Майкл Эбен, Брайан Таймэн

FreeBSD. Энциклопедия пользователя

Третье издание переработанное и дополненное

торгово-издательский дом
DiaSoft

Москва • Санкт-Петербург • Киев
2003

FreeBSD®

Brian Tiemann
Michael C. Urban

SAMS

FreeBSD®

Третье издание переработанное и дополненное

Майкл Эбен

Брайан Таймэн

ЭНЦИКЛОПЕДИЯ пользователя

издательство DiaSoft

УДК 681.3. 06(075)
ББК 32.973.2

Э 80

ЭБЕН МАЙКЛ, ТАЙМЭН БРАЙАН

Э 80 FreeBSD. Энциклопедия пользователя, 3-е изд. перераб. и доп.: Пер. с англ./Майкл Эбен, Брайан Таймэн – СПб.: ООО «ДиаСофтЮП», 2003. — 768 с.

ISBN 5-93772-106-3

Эксперты в области операционных систем и системного администрирования написали эту книгу для тех, кому нужна надежная серверная операционная система. Если раньше вам не приходилось работать с FreeBSD, книга поможет перейти на эту систему с Linux или Windows. В ней подробно рассказано об инсталляции системы, управлении загрузкой, конфигурировании системы X-Window и важнейших пользовательских сервисах.

Если вы системный администратор корпоративного сервера, книга предложит подробное описание функциональных возможностей FreeBSD, в том числе методику настройки и сборки ядра. Освещаются такие вопросы, как подключение к Internet, защита сетей и совместное использование ресурсов. Может служить в качестве справочника по настройке серверов Apache, Samba и DNS.

Для широкого круга пользователей и системных администраторов.

ББК 32.973.2

Authorized translation from the English language edition, entitled FreeBSD Unleashed, lst Edition by Urban, Michael; Tiemann, Brian, published by Pearson Education, Inc, publishing as Sams,
Copyright © 2002
All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.
Russian language edition published by DiaSoft Publishing.
Copyright © 2002

Лицензия предоставлена издательством Sams, подразделение Macmillan Computer Publishing.

Все права зарезервированы, включая право на полное или частичное воспроизведение в какой бы то ни было форме.

Материал, изложенный в данной книге многократно проверен. Но поскольку вероятность технических ошибок все равно остается, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

Все торговые знаки, упомянутые в настоящем издании, зарегистрированы. Случайное неправильное использование или пропуск торгового знака или названия его законного владельца не должно рассматриваться как нарушение прав собственности.

ISBN 5-93772-106-3 (рус.)
ISBN 0-672-32206-4 (англ.)

© Перевод на русский язык. ООО «ДиаСофтЮП», 2003
© Sams, 2002
© Оформление. ООО «ДиаСофтЮП», 2003

Гигиеническое заключение № 77.99.6.953.П.438.2.99 от 04.02.1999

Оглавление

Об авторах ... 25
О техническом редакторе ... 25
Посвящение ... 26
Благодарности ... 26
Введение ... 27
Как построена эта книга ... 29
Содержимое CD-ROM ... 30
Часть 1. Знакомьтесь: FreeBSD ... 31
Глава 1. Введение во FreeBSD ... 32
Преимущества FreeBSD ... 33
Что можно делать при помощи FreeBSD? ... 33
FreeBSD — не только для серверов ... 34
Краткая история FreeBSD и UNIX ... 35
Рождение системы BSD ... 35
BSD на платформах Intel x86 ... 36
Рождение FreeBSD ... 36
Философия UNIX ... 36
Преимущества FreeBSD ... 38
Windows 2000 и FreeBSD ... 38
Linux и FreeBSD ... 39
Талисман FreeBSD ... 40
Глава 2. Установка FreeBSD ... 41
Проверка оборудования ... 42
Создание загрузочных дискет ... 42
Создание загрузочных дискет в DOS или Windows ... 43
Создание загрузочных дисков на другой FreeBSD- или UNIX-системе ... 44
Инсталляция ... 44
Введение в программу Sysinstall ... 45
Навигация по Sysinstall ... 46
Создание разделов и выбор точек монтирования ... 46
Выбор жестких дисков ... 46
Разбиение диска на разделы ... 48
Disk Label Editor ... 49
Создание разделов и выбор точек монтирования ... 51
Оптимизация производительности ... 53
Раздел root ... 53
Раздел swap ... 54
Создание остальных разделов ... 55
Заметки о SoftUpdates ... 55
Выбор типа установки ... 56
Выбор метода установки ... 56
Настройка после инсталляции ... 57
Конфигурирование сети ... 57

Настройка консоли ... 59
Установка часового пояса ... 60
Совместимость с Linux ... 60
Конфигурирование мыши ... 60
Конфигурация X-сервера ... 61
Выбор рабочего стола по умолчанию ... 66
Установка дополнительных пакетов программ ... 66
Добавление пользователя ... 67
Установка пароля root ... 69
Выход из программы Sysinstall и перезагрузка системы ... 69
Первая загрузка FreeBSD ... 70
Отключение системы FreeBSD ... 70
Глава 3. Сложные аспекты установки ... 71
Резервное копирование ... 72
Создание раздела для FreeBSD с помощью FIPS ... 73
Запуск Scandisk и программы дефрагментации ... 73
Где найти FIPS и создание загрузочной дискеты ... 73
Работа с FIPS ... 74
Возможные проблемы с альтернативной загрузкой и их решение ... 75
Альтернативная загрузка FreeBSD и DOS, Windows 95/98/Me ... 76
Альтернативная загрузка FreeBSD и Linux ... 76
Boot-менеджер FreeBSD ... 76
Загрузка FreeBSD из LILO ... 77
Альтернативные способы установки ... 77
Установка FreeBSD по FTP ... 78
Установка по NFS ... 80
Часть 2. Использование FreeBSD ... 81
Глава 4. Первый сеанс работы во FreeBSD ... 82
Процесс запуска FreeBSD ... 83
BIOS ... 83
POST ... 83
Начальная загрузка ... 83
Этап boot0 ... 83
Этап boot1 ... 84
Этап boot2 ... 84
Этап boot3 ... 84
Ядро ... 84
Процесс init ... 86
Проверка целостности файловой системы ... 86
Сценарии конфигурации системы ... 86
Программы getty и Login ... 87
Вход во FreeBSD ... 87
Запуск системы X-Window ... 88
Введение в среду Gnome ... 88
Останов системы X-Window ... 90
Выход из FreeBSD ... 90
Закрытие FreeBSD ... 91

Команда shutdown ... 91
Замечания относительно команд halt и reboot ... 93
Глава 5. Работа в среде Gnome ... 94
Gnome ... 95
Рабочий стол ... 96
Панель Gnome ... 96
Панель задач ... 97
Работа с окнами ... 97
Виртуальные десктопы ... 98
Toggle ... 99
Опция In Group ... 99
Опция Depth ... 99
Работа с файлами и каталогами ... 99
Глава 6. Настройка среды Gnome ... 101
Добавление новых пиктограмм на рабочий стол ... 102
Создание на рабочем столе пиктограмм для запуска программ ... 102
Настройка панели Gnome ... 103
Работа с пиктограммами панели Gnome ... 103
Настройка меню Start ... 104
Работа с апплетами панели Gnome ... 105
Создание и удаление панелей ... 105
Настройка программы Gnome File Manager ... 106
Вкладка File Display ... 106
Вкладка Caching ... 107
Вкладка Desktop ... 107
Работа с Gnome Control Center ... 108
Изменение фона рабочего стола ... 108
Менеджер окон ... 109
Обработчики документов ... 109
Настройка параметров менеджера окон Sawfish ... 112
Глава 7. Работа с приложениями ... 113
Работа с текстом ... 114
Текстовый редактор gedit ... 114
Редактор vi ... 118
Графика и изображения ... 123
Программа GIMP ... 123
GQview ... 126
StarOffice ... 127
Рабочий стол StarOffice ... 127
Встроенный Web-браузер StarOffice ... 128
Электронная почта в StarOffice ... 128
StarOffice Explorer ... 129
Справочная информация в StarOffice ... 130
Мультимедиа ... 130
Проверка поддержки звуковой платы операционной системой FreeBSD ... 130
Создание файлов устройств ... 131
Микшер ... 131

Воспроизведение файлов MP3 с помощью XMMS ... 133
Воспроизведение MP3-файлов с помощью mpg123 ... 133
Микшеры для системы X-Window ... 134
Сетевые приложения ... 134
Конфигурирование веб-браузера Netscape ... 134
Веб-браузер Lynx ... 135
FTP ... 135
Приложения электронной почты ... 137
Глава 8. Работа с командным интерпретатором ... 140
Основы работы с командным интерпретатором ... 141
Типы доступных командных интерпретаторов ... 141
Командный интерпретатор Bourne (sh) ... 141
Командный интерпретатор C (csh) ... 142
Командный интерпретатор Korn (ksh или pdksh) ... 142
Командный интерпретатор Bourne Again Shell (bash) ... 143
Командный интерпретатор Tcsh (tcsh) ... 143
Какой интерпретатор выбрать? ... 143
Изменение интерпретатора ... 143
Получение справки ... 144
Поиск страниц справочных руководств ... 144
Резюме команд ... 145
Разделы справочных руководств ... 145
Простейшие способы работы с файлами в командном интерпретаторе ... 146
Как система FreeBSD хранит файлы ... 146
Начальный каталог ... 147
Вывод содержимого каталога ... 147
Перемещение по файловой системе ... 149
Можно запутаться и заблудиться... ... 149
Копирование файлов и каталогов ... 149
Перемещение и переименование файлов и каталогов ... 150
Удаление файлов и каталогов ... 151
Удаление каталогов ... 151
Команда touch ... 152
Создание ссылок ... 152
Универсальные опции ... 154
Метасимволы и символы-заместители ... 154
Несколько слов об именах файлов ... 155
Работа с нестандартными именами файлов ... 156
Команды обработки текстовых файлов ... 157
Подсчет числа строк, слов и символов ... 157
Просмотр текстовых файлов: утилиты more и less ... 157
Поиск по шаблону ... 158
Сортировка текста в файле ... 159
Замена строк с помощью tr ... 159
Вывод частей строк из текстовых файлов ... 160
Форматирование текста утилитой fmt ... 161
Конвейеры и перенаправление ввода-вывода ... 161
Дополнение командной строки и редактирование истории команд ... 163

Часть 3. Администрирование FreeBSD 164

Глава 9. Файловая система FreeBSD 165

Структура каталогов FreeBSD 166
Мониторинг использования файловой системы 169
Команда df 169
Команда du 170
Монтирование и демонтирование файловых систем FreeBSD 171
Команда mount 171
Команда umount 172
Монтирование и демонтирование файловых систем других операционных систем 173
Монтирование файловой системы Windows/MS-DOS 174
Монтирование файловой системы Linux 175
Монтирование и демонтирование файловых систем CD-ROM и флоппи-дисков 176
Монтирование компакт-дисков и дискет 176
Демонтирование компакт-дисков и дискет 177
Другие сменные носители 177
Понимание файла /etc/fstab 177
Проверка и восстановление файловых систем с помощью утилиты fsck 179
Журнальные файловые системы и мягкое обновление 181
Использование fsck для восстановления суперблоков 181
Настройка пользовательских квот 183
Блоки, файлы и индексные дескрипторы 184

Глава 10. Пользователи, группы и права доступа 187

Концепция пользователей и групп 188
Зачем нужны группы? 190
Владение файлами 191
Изменение прав владения файлом с помощью команды chown 192
Изменение групповых прав владения файлом с помощью команды chgrp 193
Права доступа к файлам и каталогам 193
Взаимосвязь между правами доступа файлов и каталогов 194
Изменение прав доступа к файлам и каталогам с помощью команды chmod 195
Списки управления доступом 197
Конфигурирование поддержки ACL в ядре 197
Конфигурирование файловых систем для использования ACL 197
Получение информации о текущих установках ACL 198
Установка маски максимальных прав доступа 198
Добавление пользователя или группы в ACL 199
Запрещение доступа с помощью ACL 201
Удаление записи из ACL 201
Удаление всех записей из ACL 201
Добавление и удаление учетных записей пользователей 201
Файлы /etc/passwd и /etc/master.passwd 203

Файл /etc/group ... 205
Управление группами ... 205
Глава 11. Конфигурация системы и стартовые сценарии ... 206
Понимание процесса начальной загрузки FreeBSD ... 207
Что может замедлить процесс загрузки? ... 210
Сценарии конфигурирования ресурсов ... 210
Файл /etc/defaults/rc.conf ... 212
Файл /etc/rc.conf ... 213
Каталоги /usr/local/etc и /usr/local/X11R6/etc ... 214
Создание сценариев для запуска программ при загрузке системы ... 215
Демон inetd и файл конфигурации inetd.conf ... 216
Система ведения журналов (syslogd) и файл syslog.conf ... 218
Заметки о файле /etc/rc.local ... 219
Глава 12. Настройка командного интерпретатора ... 221
Что такое командный интерпретатор? ... 222
Работа с командными интерпретаторами ... 223
Установка командных интерпретаторов ... 224
Файл /etc/shells ... 225
Использование альтернативных командных интерпретаторов ... 226
Изменение интерпретатора в процессе работы ... 226
Изменение командного интерпретатора по умолчанию ... 227
Программы, используемые в качестве командных интерпретаторов ... 228
Файлы инициализации интерпретатора ... 229
Файлы tcsh/csh: .cshrc, .login и .logout ... 230
Файлы bash: .profile, .shrc и .bash_logout ... 231
Настройка среды командного интерпретатора ... 233
Настройка tcsh ... 233
Настройка bash ... 235
Переменные среды и переменные командного интерпретатора ... 236
Переменные среды ... 236
Переменные командного интерпретатора ... 237
Глава 13. Программирование на языке командного интерпретатора ... 238
Пример простой программы для интерпретатора ... 240
printf ... 241
Переменные ... 242
Присвоение значения переменной ... 243
Имена переменных ... 243
Взаимодействие с пользователем ... 244
Обработка аргументов командной строки ... 245
Подстановка команд ... 246
Арифметические операции в сценариях ... 246
Циклы ... 248
Цикл while ... 249
Цикл until ... 250
Логические операторы AND/OR в циклах while и until ... 250
Цикл for ... 251
shift ... 252

Операторы true и false ... 253
Выход из цикла ... 253
Условные операторы ... 254
Операторы if ... 254
Оператор case ... 258
Логические операторы AND/OR ... 259
Код завершения ... 260
Установка кода завершения ... 261
Перехват прерываний при выходе ... 262
Функции ... 263
Файловые дескрипторы ... 263
Отладка сценариев командного интерпретатора ... 264
Расширенные возможности интерпретатора Korn ... 265
Получение и установка интерпретатора Korn ... 266
Встроенная арифметика ... 266
Массивы ... 267
Подстановка команд ... 270
Использование getopts ... 270
Глава 14. Мониторинг производительности, управление процессами и автоматизация заданий ... 271
Мониторинг производительности. Утилита top ... 272
Вывод утилиты top ... 273
Мониторинг процессов. Утилита ps ... 275
Объяснение вывода ps ... 276
Прерывание неуправляемого процесса ... 277
Команда kill ... 277
Опции команды kill ... 278
Изменение приоритетности процесса ... 278
Автоматизация заданий ... 279
Демон cron ... 279
Анатомия файла crontab ... 280
Создание и редактирование файлов crontab ... 281
Однократный запуск заданий с помощью команды at ... 282
Управление доступом к командам cron и at ... 283
Глава 15. Установка дополнительного программного обеспечения ... 284
Пакеты FreeBSD ... 285
Разделяемые библиотеки и зависимости ... 286
Получение информации об инсталлированных пакетах ... 287
Установка пакетов ... 289
Установка с помощью sysinstall ... 289
Использование утилиты pkg_add ... 291
Удаление пакетов ... 293
Обновление пакетов ... 293
Стратегия портов ... 293
Дерево портов FreeBSD ... 294
Анатомия порта FreeBSD ... 296
Инсталляция портов ... 297

Удаление инсталлированных портов ... 298
Обновление порта ... 298
Как обновить локальное дерево портов? ... 299
Замечания о запрещенных портах ... 300
Освобождение дискового пространства, занятого в процессе сборки портов ... 301
Как поступить, если сборка порта завершается неудачно ... 301
Web-сайт Fresh Ports ... 302
Глава 16. Печать ... 304
Демон lpd ... 305
Очередь печати ... 305
Конфигурирование ядра, устройства и режима соединения ... 306
Конфигурирование режима параллельного порта ... 306
Создание каталога спула ... 307
Фильтры ... 307
Текстовые фильтры ... 307
Печать файлов в формате PostScript на не-PostScript-принтерах ... 310
Фильтр печати lpf ... 311
Фильтры преобразования ... 311
Настройки в файле /etc/printcap ... 312
Установка фильтров преобразования ... 313
Включение lpd ... 314
Печать из командной строки ... 314
Печать из X-Window ... 315
Печать в StarOffice ... 316
Проверка состояния заданий печати ... 317
Удаление заданий из очереди ... 317
Управление принтером ... 319
Программа lpc в интерактивном режиме ... 319
Состояние очереди ... 319
Запрет печати и останов демона ... 319
Использование программы lpc в неинтерактивном режиме ... 323
Управление доступом к lpc ... 324
Основы сетевой печати ... 324
Решение проблем ... 324
Принтер не получает данных, задания находятся в очереди ... 324
Принтер сигнализирует о приеме данных, но не печатает ... 324
Вместо изображения или Web-страницы печатаются сотни страниц мусора ... 324
Принтер работает медленно ... 325
Вывод печатается "лесенкой" ... 325
Весь текст печатается в одной строке, поверх уже напечатанного текста ... 326
Глава 17. Конфигурирование ядра ... 327
Роль ядра ... 328
Зачем конфигурировать собственное ядро? ... 329
Использование dmesg для сбора информации о запуске ядра ... 330
Конфигурационные файлы ядра ... 331

Конфигурационный файл GENERIC ... 332
Подсказки ядру относительно устройств ... 333
Файл LINT ... 333
Создание конфигурационного файла ядра ... 334
Компиляция и установка ядра ... 335
Добавление файлов устройств в каталоге /dev ... 335
Аварийное восстановление ... 336
Глава 18. Поддержка FreeBSD ... 337
Отслеживание исходного кода FreeBSD ... 338
Объяснение ветвей исходного кода STABLE и CURRENT ... 338
Выбор цели обновления ... 339
Что такое make world? ... 340
На что следует обратить внимание перед сборкой системы ... 341
Задачи, выполняемые до сборки всей системы ... 343
Синхронизация локального дерева исходного кода с деревом STABLE, CURRENT или RELEASE ... 343
Текстовый файл UPDATING ... 345
Объединение файлов /etc/group и /etc/passwd с соответствующими новыми версиями ... 345
Объединение файла /etc/make.conf с его новой версией ... 346
Сборка системы из исходных файлов ... 347
Очистка каталога /usr/obj ... 347
Запуск записи в log-файл ... 347
make buildworld ... 348
Обновление ядра ... 349
make installworld ... 350
Использование mergemaster для проверки измененных конфигурационных файлов ... 351
Перезагрузка системы после обновления ... 353
Глава 19. О жестких дисках и файловых системах ... 354
Режимы доступа IDE/ATA ... 355
Режимы PIO ... 355
Режимы DMA ... 356
Режим Ultra DMA ... 356
Диски SCSI ... 357
Геометрия жесткого диска ... 359
Режим LBA и предел 528 Мб ... 360
Режимы Extended INT13 и предел 8.4 Гб ... 360
Практические вопросы ... 361
Разбиение жесткого диска на разделы ... 361
Разделы BIOS (Slices) ... 361
Разделы BSD ... 361
Редактор слайсов (fdisk) в программе sysinstall ... 363
Создание дисковых меток ... 364
Полная разметка раздела FreeBSD ... 364
Добавление нового диска ... 365
Сохранение изменений и форматирование диска ... 366
Подготовка файловой системы к использованию ... 367

Глава 20. Курс выживания для пользователей FreeBSD 368
Переход к FreeBSD 369
Переход от Windows NT/2000 370
Что "следует" и "не следует" делать 378
Тонкая настройка производительности 380
Настройки ядра 381
Система Soft Updates и асинхронная запись 381
Вопросы геометрии диска 382
Несколько настроек с помощью sysctl 383
Полезные справочные руководства 383
Подготовка к худшему: резервные копии 384
Создание ключевых файлов 384
Резервные копии 384
Восстановление 386
Зеркальные сервера 387
Глава 21. Знакомство с программированием на языке Perl 390
Что такое Perl? 391
Perl во FreeBSD 391
Сильные стороны Perl 392
Слабые стороны Perl 393
Основы создания сценариев на Perl 393
Переменные и операторы 394
Скаляры, массивы и ассоциативные массивы 395
Управление потоком 397
Аргументы командной строки 399
Простой сценарий на языке Perl 400
Дополнительные возможности Perl 400
Обработка текста 400
Работа с файлами 404
Функции 406
Модули Perl 407
Полезные ресурсы, посвященные Perl 409
Web-сайты 409
Книги 409
CPAN 410
Часть 4. FreeBSD и работа в сети 411
Глава 22. Основы компьютерных сетей 412
Введение в сети 413
Топологии сетей 414
Сетевые компоненты 417
Кабели 417
Хабы 419
Коммутаторы 420
Мосты 422
Маршрутизаторы 422
Сетевые протоколы 423
TCP: Transmission Control Protocol 423

UDP: User Datagram Protocol ... 424
ICMP: Internet Control Message Protocol ... 425
TCP/IP ... 425
IP-адреса ... 426
ARP и MAC-адреса ... 428
Команда arp ... 429
Подсети и сетевые маски ... 430
Маршрутизация ... 431
Шлюзы и трансляция сетевых адресов (Gateways and Network Address Translation) ... 433
Имена хостов и доменов ... 434
DHCP ... 435
Глава 23. Настройка основных сетевых служб ... 436
Конфигурирование сетевой карты ... 437
Настройка сетевых параметров при помощи sysinstall ... 438
Настройка сетевых параметров вручную ... 441
Использование ifconfig ... 441
Команда route и настройка шлюза ... 443
Имя хоста (hostname) ... 444
Сетевые настройки в файле /etc/rc.conf ... 445
Работа с /etc/netstart ... 445
Создание IP-псевдонимов ... 447
Настройка имен и IP-адресов в файле /etc/hosts ... 448
Утилита ping ... 448
Настройки DNS в файле /etc/resolv.conf ... 449
Другие файлы конфигурации сети ... 450
Глава 24. Подсоединение к Интернет с помощью PPP ... 451
Выбор Интернет-провайдера ... 452
Информация, необходимая для настройки ... 452
Сравнение kernel PPP и user PPP ... 453
Конфигурирование kernel PPP ... 453
/etc/resolv.conf ... 453
Файл /etc/ppp/options ... 453
Сценарий программы chat ... 454
Запуск демона pppd ... 455
PAP- и CHAP-аутентификация ... 455
pap-secrets and chap-secrets ... 456
Соединения по требованию и постоянные соединения ... 456
Установление и разрыв соединения ... 457
Пользовательский PPP ... 457
Файл /etc/ppp/ppp.conf ... 458
Запуск пользовательского PPP ... 460
Запуск user PPP рядовым пользователями ... 460
Соединение по требованию и постоянное соединение ... 460
Запуск программ при установлении и разрыве соединения ... 461
Решение проблем ... 461
Последние замечания ... 462

Глава 25. Настройка служб E-mail ... 463
Общие сведения об SMTP ... 464
Агенты пересылки почты (MTA) и пользовательские почтовые агенты (MUA) ... 465
Распространенные агенты пересылки почты (MTA) ... 466
Распространенные пользовательские почтовые агенты (MUA) ... 467
Настройка основных служб Sendmail ... 468
Схема размещения файлов Sendmail ... 468
Конфигурационные файлы ... 468
Вопросы разрешения DNS-адресов ... 472
Управление системой Sendmail ... 472
Mail Relay ... 474
Протокол POP3 ... 476
Настройка POP3-сервера qpopper ... 477
Инсталляция и конфигурирование qpopper ... 477
Запуск в автономном режиме (Standalone Mode) ... 478
Запуск в режиме сервера (Server Mode) ... 479
Включение шифрования по SSL ... 479
Настройка IMAP-сервера IMAP-UW ... 481
Электронная почта для отдельных рабочих станций ... 483
Использование Fetchmail для получения почты с POP3- и IMAP-серверов ... 483
Настройка Sendmail для отдельных рабочих станций ... 486
Альтернативные почтовые серверы ... 487
Postfix ... 487
Qmail ... 487
Exim ... 487
Smail ... 488
Глава 26. Конфигурирование Web-сервера ... 489
Общие сведения о протоколе HTTP ... 490
Коды ответа и перенаправление ... 492
Получение и инсталляция сервера Apache ... 494
Схема расположения файлов сервера Apache ... 495
Конфигурирование сервера Apache ... 496
Использование httpd.conf ... 496
Использование файлов .htaccess и заменяющих опций ... 497
Запуск и останов демона HTTP ... 499
Основы контроля доступа к серверу ... 500
Контроль доступа по адресу ... 501
Контроль доступа по паролю ... 502
Контроль доступа по адресу и паролю ... 504
Виртуальный хостинг ... 505
Модули сервера Apache ... 507
Встроенные модули ... 507
Динамически загружаемые модули ... 507
Сторонние модули ... 507
Серверные расширения ... 509
Основы CGI ... 511

Включение CGI в Apache ... 511
Написание CGI-программ ... 513
Глава 27. Конфигурирование FTP-сервера ... 516
Общие сведения о FTP ... 517
Обзор структуры каталогов FTP ... 519
Аутентифицированный и анонимный FTP-доступ ... 520
Настройка FTP-сервера ... 520
Управление FTP-доступом ... 522
Файл /etc/ftpusers ... 523
Файл /etc/shells ... 523
Файл /var/run/nologin ... 523
Разрешение анонимного доступа по FTP ... 524
Виртуальный хостинг ... 525
Использование альтернативных FTP-серверов ... 525
WU-FTPD ... 526
ProFTPD ... 526
Глава 28. Конфигурирование шлюза Internet ... 527
Что такое маршрутизатор? ... 528
Что такое шлюз? ... 530
Что такое NAT? ... 530
Конфигурирование шлюза NAT в ОС FreeBSD ... 530
Включение перенаправления пакетов ... 531
Включение NAT ... 532
Использование реализации PPP пользовательского уровня ... 532
Использование реализации PPP на уровне ядра или подключение к Internet по выделенной линии ... 532
Конфигурирование и включение демона natd ... 533
Включение и конфигурирование брандмауэра ... 533
Конфигурирование клиентов для использования нового шлюза ... 534
Конфигурирование клиентов Windows 95/98 ... 534
Конфигурирование клиентов Mac OS и Mac OS X ... 535
Конфигурирование клиентов FreeBSD ... 536
Конфигурирование клиентов Linux ... 537
Конфигурирование беспроводного доступа к Internet ... 538
Маршрутизация между тремя и более сетями ... 538
Динамическая маршрутизация ... 540
Маршрутизация в масштабе предприятия и демилитаризованная зона (DMZ) ... 541
Глава 29. Защита сети ... 542
Модели защиты ... 543
Правила задания паролей ... 545
Обеспечение безопасности паролей с помощью программы Crack ... 545
Устаревание паролей ... 547
Задание первоначальных паролей ... 548
Одноразовые пароли на базе системы S/Key ... 549
Система Kerberos ... 551
Проблемы со службами, передающими информацию в явном виде ... 552

Использование утилиты tcpdump для контроля передаваемой информации ... 552
Защита терминальной информации (OpenSSH) ... 554
Защита служб электронной почты (POP3 и IMAP) ... 555
Защита FTP-сервера ... 556
Защита сервера Apache ... 557
Система Apache-SSL ... 558
Сервер Apache с модулем mod_ssl ... 558
Эксплуатация защищенного Web-сервера ... 559
Плохо написанные сценарии CGI ... 559
Повышение безопасности сценариев CGI с помощью системы CGIwrap ... 560
Профили защиты системы и защита ядра (securelevel) ... 561
Использование брандмауэра ... 562
Включение поддержки брандмауэра ... 563
Конфигурирование системы IPFW ... 565
Предотвращение вторжений и взломов ... 566
Использование PortSentry ... 567
Использование файла /etc/hosts.allow ... 569
Использование системы Tripwire ... 570
Если кажется, что система взломана ... 573
Атаки на службы (DoS) ... 574
Ограничение количества порождаемых серверных процессов ... 575
Защита от атак с плацдарма ... 576
Физическая защита ... 577
Другие источники информации о защите ... 577
Страница справочного руководства man security ... 577
Списки рассылки ... 578
Руководства по защите ОС FreeBSD ... 578
Web-ресурсы ... 578
Книги ... 579
Глава 30. Сервер доменных имен ... 580
Введение в систему BIND ... 581
Структура системы доменных имен ... 581
Зоны ... 582
Файлы и программы системы BIND ... 583
Включение демона сервера имен ... 583
Запуск системы BIND в "песочнице" ... 584
Файл конфигурации системы BIND (named.conf) ... 584
Использование перенаправляющего сервера ... 586
Конфигурации главного и подчиненного сервера ... 587
Другие типы зон ... 588
Ограничение доступа к службе DNS ... 589
Создание файла зоны ... 590
Директивы ... 591
Записи SOA ... 593
Записи NS ... 594
Записи адреса ... 595
Записи CNAME ... 595

Записи MX ... 595
Записи PTR ... 596
Файлы обратного просмотра зоны DNS ... 596
Создание файла зоны localhost ... 597
Конфигурирование кэширующего сервера имен ... 597
Глава 31. Сетевая файловая система (NFS) ... 599
Основы NFS ... 600
Конфигурирование сервера NFS ... 602
Демон NFS (nfsd) ... 603
Демон монтирования NFS (mountd) ... 603
Задание общих ресурсов в файле /etc/exports ... 604
Запуск служб NFS без перезагрузки ... 605
Конфигурирование клиента NFS ... 606
Демон ввода/вывода NFS (nfsiod) ... 606
Монтирование удаленных файловых систем ... 606
Автоматическое монтирование удаленных файловых систем при загрузке системы ... 607
Демон автоматического монтирования (amd) ... 608
Глава 32. Совместное использование файлов и принтеров с Microsoft Windows ... 610
Введение в систему Samba ... 611
Особенности работы протоколов SMB/CIFS ... 611
Просмотр ... 612
Защита, рабочие группы и домены ... 613
Установка и конфигурирование системы Samba ... 614
Демоны smbd и nmbd ... 615
Файл smb.conf и система SWAT ... 615
Предоставление каталогов для общего доступа ... 617
Совместный доступ к принтерам ... 619
Управление доступом ... 619
Журнальные файлы системы Samba ... 622
Переменные системы Samba ... 622
Другие компоненты системы Samba ... 623
Дальнейшее развитие системы Samba ... 624
Файловая система smbfs ... 624
Глава 33. Протокол DHCP ... 626
Как работает протокол DHCP ... 627
Аренда IP-адреса ... 627
Преимущества использования DHCP ... 627
Конфигурирование ядра для поддержки протокола DHCP ... 629
Включение поддержки DHCP ... 629
Включение DHCP с помощью программы sysinstall ... 629
Конфигурирование поддержки протокола DHCP вручную ... 630
Программа dhclient ... 631
Сценарий /sbin/dhclient-script ... 631
Файл /etc/dhclient.conf ... 632
Демон сервера DHCP ... 633

Файл конфигурации dhcpd ... 634
Программа dhcpconf ... 634
Часть 5. X-Window ... 637
Глава 34. Дополнительные возможности настройки системы X-Window ... 638
Обновление версии 3.3.6 до 4.x ... 639
Использование SuperProbe ... 639
Конфигурирование системы X-Window с помощью сценария xf86config ... 640
Настройка мыши ... 641
Выбор клавиатуры ... 642
Настройка монитора ... 643
Настройка видеокарты ... 645
Опции в файле XF86Config ... 651
Синтаксис файла XF86Config ... 651
Раздел "Modules" ... 652
Раздел "Files" ... 652
Раздел "ServerFlags" ... 653
Раздел "InputDevice" ... 655
Раздел "Monitor" ... 658
Раздел "Device" ... 658
Раздел "Screen" ... 659
Подраздел "Display" ... 660
Раздел "ServerLayOut" ... 660
Проверка настройки X-Window ... 661
Файл .xinitrc с личными настройками ... 662
Смена менеджера окон ... 662
Автоматический запуск приложений ... 663
Установка цвета фона или фонового изображения ... 664
Работа со шрифтами ... 665
Проверка файла XF86Config ... 665
Создание каталогов и установка шрифтов ... 666
Использование удаленных клиентов X-Window ... 667
Использование программы xhost для отображения вывода удаленных графических приложений ... 667
Запуск удаленного приложения ... 669
Другие способы управления доступом клиентов ... 669
xdm ... 669
Часть 6. Приложения ... 671
Приложение А. Основные конфигурационные файлы и сценарии FreeBSD ... 672
Приложение Б. Список поддерживаемого оборудования ... 677
Требования к системе ... 678
Поддерживаемое оборудование ... 678
Дисковые SCSI-контроллеры ... 678
Приводы CD-ROM ... 679
Сетевые карты ... 680

Устройства USB ... 683
Звуковые устройства ... 683
Различные устройства ... 683
Видеокарты, поддерживаемые системой X-Window ... 684
Приложение В. Решение проблем с инсталляцией и загрузкой ... 698
Проблемы с инсталляцией системы ... 699
Загрузка с дискеты приводит к останову или перезагрузке ... 699
Загрузка с дискеты повисает в фазе Probing Devices (Проверка устройств) ... 699
Система загружается с CD-диска, но программа инсталляции показывает, что CD-ROM не найден ... 699
Геометрия жесткого диска определяется неверно ... 699
Система Micron зависает при загрузке ... 700
Сетевая карта 3Com PCI не работает с системой Micron ... 700
SCSI-контроллер HP Netserver не определяется при загрузке ... 700
В системе с видеокартой ATI Mach64 гаснет экран ... 700
Устройства, необходимые для установки FreeBSD, не обнаружены ... 700
Проблемы загрузки и вопросы, не связанные с инсталляцией ... 701
При попытке загрузиться FreeBSD выдает сообщение Missing Operating System (Отсутствует операционная система) ... 701
Менеджер загрузки FreeBSD зависает на "F?" ... 701
Менеджер загрузки FreeBSD выдает сообщение Read Error (Ошибка чтения) и зависает ... 701
Менеджер загрузки FreeBSD отсутствует, загружается Windows ... 702
FreeBSD обнаруживает меньше оперативной памяти, чем реально присутствует в системе ... 702
FreeBSD выдает сообщение Device Not Configured (Устройство не сконфигурировано) при попытке монтирования CD-диска ... 702
Программы завершают работу с ошибками Signal 11 ... 703
Система выдает странные сообщения об ошибках при запуске top, ps и других системных утилит ... 703
Забыт пароль пользователя root ... 703
Приложение Г. Источники информации ... 704
Ресурсы, связанные непосредственно с FreeBSD ... 705
Web-сайты ... 705
Списки рассылки ... 705
Общие списки ... 706
Дискуссионные группы USENET, посвященные FreeBSD ... 708
Каналы IRC ... 708
Дополнительные ресурсы, связанные с BSD ... 708
Web-сайты ... 709
Дискуссионные группы USENET ... 709
Другие ресурсы Internet ... 709
Web-сайты ... 709
Дискуссионные группы USENET ... 710
Приложение Д. Справочник по командам ... 712
Перечень команд ... 714

& ... 714
| ... 714
> ... 714
< ... 714
>> ... 714
<< ... 715
adduser ... 715
alias ... 715
at ... 715
atq ... 715
atrm ... 715
bash ... 715
batch ... 715
bc ... 716
bg ... 716
cal ... 716
cat ... 716
cd ... 716
chfn ... 716
chgrp ... 717
chmod ... 717
chown ... 718
chroot ... 718
chsh ... 718
cp ... 718
cpio ... 718
crontab ... 718
cu ... 719
cut ... 719
date ... 719
dc ... 719
dd ... 719
df ... 719
diff ... 719
dir ... 719
dmesg ... 719
du ... 719
dump ... 720
echo ... 720
ed ... 720
ee ... 720
elm ... 720
emacs ... 720
env ... 720
ex ... 720
fdformat ... 720
fetchmail ... 720
fg ... 720
file ... 720

find ... 721
finger ... 721
fmt ... 721
ftp ... 721
grep ... 721
gunzip ... 721
gzip ... 721
halt ... 722
head ... 722
hostname ... 722
ifconfig ... 722
irc ... 722
ispell ... 722
kill ... 722
less ... 722
ln ... 722
locate ... 722
login ... 723
logout ... 723
look ... 723
lpc ... 723
lpq ... 723
lpr ... 723
lprm ... 723
ls ... 723
lynx ... 724
mail ... 724
make ... 724
make install ... 724
make deinstall ... 724
make clean ... 724
make distclean ... 724
makewhatis ... 724
man ... 724
mkdir ... 724
more ... 725
mount ... 725
mv ... 725
netstat ... 725
newgrp ... 725
passwd ... 725
pdksh ... 725
ping ... 726
pkg_add ... 726
pkg_info ... 726
pkg_delete ... 726
pppd ... 726
pppstats ... 726
pr ... 726

procmail ... 726
ps ... 726
pwd ... 726
reboot ... 727
red ... 727
restore ... 727
rm ... 727
rmdir ... 727
rmuser ... 727
route ... 727
sed ... 727
shutdown ... 728
sort ... 728
su ... 728
swapon ... 728
tail ... 728
talk ... 728
tar ... 728
tcsh ... 728
telnet ... 729
top ... 729
touch ... 729
tput ... 729
tr ... 729
umount ... 729
uptime ... 729
vi ... 729
vipw ... 729
vmstat ... 730
w ... 730
wall ... 730
wc ... 730
whatis ... 730
whereis ... 730
who ... 730
whoami ... 730
Приложение E. Вопросы локализации FreeBSD ... 731
Языки, символы, кодировки ... 732
Локализация ... 734
Команда locale ... 735
Русификация консоли во FreeBSD ... 735
Настройка LOCALE ... 737
Русификация приложений ... 738
Русификация X-Window ... 739
Часто задаваемые вопросы ... 740
Несколько слов о проблемах украинизации ... 741
Полезные ссылки ... 741
Предметный указатель ... 743

Об авторах

Майкл Эбен (Michael Urban), биолог по образованию, несколько лет работал с разными операционными системами UNIX, включая FreeBSD, Linux и Solaris. Майкл — технический аналитик, системный администратор и вебмастер в Lion Research Center. Кроме того, он занимается программированием, включая разработку баз данных, доступных через Web. Если он не занят программированием на Perl или Java, то, как правило, погружен в исследования африканских львов.

Брайан Таймэн (Bryan Tiemann) — постоянный пользователь FreeBSD еще со времен учебы в Caltech. Тогда он использовал эту систему при создании развлекательного веб-сайта для фанатов киноиндустрии, который существует и поныне. Уроженец Калифорнии, Брайан прожил в этом штате всю жизнь, но сейчас он живет в Сан-Хосе, и занимается сетевым оборудованием. Помимо FreeBSD, интересуется компьютерами Macintosh, анимацией и мотоциклами.

О техническом редакторе

Тим Хикс (Tim Hicks) — главный инженер компании HomeSide Leading, Inc. Он сертифицированный специалист в области информационных технологий и системного администрирования операционных систем UNIX. Работал с HP-UX 10.10 и 11.00, а также с Solaris 8 и AIX 4.1 в течение пяти лет. Тим имеет глубокие знания и опыт работы с кластерными системами и базами данных Sybase и Oracle.

Посвящение

Моим родителям, Крису и Бонни, моей сестре Бесс, а также львам, живущим в Серенгетти. Надеюсь, что данная работа внесет определенный вклад в то, что Lion Research Center делает для них.

Майкл С. Эбен

Посвящаю эту книгу моим родителям, Кейту и Анне, и моему брату Майку. А также Дугласу Адамсу, где бы во Вселенной он ни находился.

Брайан Таймэн

Благодарности

Проект такого масштаба невозможно реализовать без помощи единомышленников. Мы хотим выразить благодарность персоналу Sams Publishing, работавшему с нами над созданием этой книги, — Кэтрин Педам, Марку Сьерзньяку, Андрею Бистеру, Нэнси Сиксмис и Дэну Шерфу. Мы особенно признательны Тиму Хиксу за техническое редактирование этой книги.

Данная книга не вышла бы в свет без помощи разработчиков FreeBSD. Мы благодарим Джордана Хаббарда и его команду за поддержку. Нельзя не упомянуть и о добровольцах-разработчиках FreeBSD по всему миру. Спасибо вам за то, что вы посвящали FreeBSD свое свободное время, дабы она превзошла коммерческие операционные системы, стоимостью в сотни, а то и тысячи долларов.

Мы благодарим доктора Крэйга Пакера и Пэйтона Вэста из Lion Research Center (**www.lionresearch.org**) за ряд фотографий, помещенных в книге. Кроме того, я хотел бы поблагодарить моего соавтора, Брайана Таймэна, за разрешение пользоваться его веб-сервером для тестирования моих программных разработок и игр.

Большое спасибо Полу Саммерсу, который выполнял тяжелую работу по администрированию сайта, пока Брайан философствовал, был хакером и весело проводил время где-то на другом конце города. Он счастлив, что у него есть многочисленные друзья, которые помогали принимать решения и усмирять страсти, бушевавшие вокруг операционных систем, и воодушевляли его на нелегкий писательский труд. Я благодарю всех членов TLK-L и **lionking.org** за мощную поддержку на протяжении всех этих дней. Они заставили меня поверить в то, что я знаю, о чем говорю.

Введение

Двадцать лет назад, когда начали продавать первые персональные компьютеры, мог ли кто-то предположить, для чего они будут использоваться в начале третьего тысячелетия? Мог ли кто-то подумать, что Microsoft станет такой влиятельной компанией, какой в то время была IBM? А кто мог предсказать рост программных продуктов с открытым исходным кодом? А то, что эти продукты будут на равных бороться с Microsoft за долю рынка? Можно ли было представить, что в 2001 году IBM будет поддерживать развитие Linux?

Но и сегодня набросать контуры будущего отнюдь не проще — очень уж быстро развиваются технологии. Предсказать, продукция каких компаний будет использоваться через 20 лет, — чрезвычайно тяжело. Тем не менее легко догадаться, что Internet будет развиваться. Ясно и то, что все реже Internet-серверами будут управлять коммерческие операционные системы. В ближайшие годы подавляющее большинство Internet-серверов будет работать под управлением операционных систем UNIX с открытым исходным кодом. Наибольшее внимание системных администраторов привлекает Linux. Эта ОС завоевала признание широких масс, а в последнее время она начинает пользоваться и поддержкой корпораций. Даже недавние попытки Microsoft представить Linux злейшим врагом интеллектуальной собственности способствовали росту ее популярности. Как бы там ни было, но программные средства с открытым кодом уверенно отвоевывают себе место под солнцем.

Однако часто незамеченным остается тот факт, что Linux — это не UNIX; это, скорее, разновидность UNIX. Да, эта операционная система выполняет те же функции, что и коммерческие версии UNIX, но, в отличие от них, она от начала и до конца разработана сообществом ее пользователей. Основные принципы этого проекта отображены в общедоступной лицензии GPL, GNU General Public License, где сказано, что любой код, созданный по данной лицензии, должен быть бесплатным и доступным всем желающим. Это положение распространяется и на коммерческое программное обеспечение, разработанное на базе открытого кода. Оно также должно выпускаться по лицензии GPL, т.е. как ПО со свободно доступным исходным кодом, поскольку эти программы созданы на базе разработок с открытым кодом.

К сожалению, до сих пор нет единой точки зрения на ПО с открытым кодом. Не прекращается давний спор между сторонниками открытого ПО и приверженцами коммерческих программ. Главная идея GPL такова: программы придуманы людьми и для людей, а значит, они никому не принадлежат и не могут использоваться для получения коммерческой выгоды. Однако многие компании, пользуясь открытым ПО, не хотят соблюдать условия GPL. По их мнению, распространение разработанного ими исходного кода равносильно публикации коммерческих секретов. Программному обеспечению с открытым кодом противостоит коммерческое ПО, т.е. ПО с закрытым исходным кодом, поставляемое исключительно в виде исполняемых программ единственным поставщиком. Это ПО нельзя модифицировать под свои нужды. Конечно, коммерческие предприятия не в восторге от отсутствия альтернативы, но общедоступная лицензия их тоже пугает, поскольку бесплатное ПО ассоциируется с ненадежным ПО.

FreeBSD (Berkeley Software Design) — это не просто открытая система, это система с изюминкой. Она основана на более либеральной BSD-лицензии для открытых программных средств, которая позволяет применять код, разработанный в Кали-

форнийском университете в Беркли, при разработке собственных программ, не требуя обязательной публикации вновь разработанного исходного кода. Лицензия BSD более дружелюбна к специалистам, разрабатывающим коммерческое ПО, чем GPL. Вот почему Apple (ранее NeXT) избрала ядро BSD в качестве основы для своей платформы NeXTSTEP, которая позже превратилась в Mac OS X. Вот почему Microsoft предпочла систему FreeBSD операционной системе Linux, анонсировав планы портирования среды программирования C# на платформу FreeBSD. Лицензия BSD поддерживает вклад множества пользователей и в то же время не создает коммерческих препятствий компаниям, вкладывающим средства в разработку BSD-программ.

Естественно, можно предположить, что Linux и FreeBSD являются конкурентами. По мнению части пользователей, так оно и есть. В процентном отношении FreeBSD сейчас превалирует среди операционных систем с открытым кодом, кроме Linux. В настоящее время она занимает 15% рынка, согласитесь, это достаточно много.

Одним из самых существенных различий между FreeBSD и Linux являются методы их продвижения. Систему FreeBSD практически не рекламируют. А вот агрессивная поддержка Linux не имеет аналогов в мире UNIX. В какой-то мере в этом есть свой смысл: Linux — пример "экстремальной" системы с открытым ПО, а FreeBSD — компромисс между энтузиазмом и корпоративной рутиной. Резонанс, вызванный поистине всенародным вкладом в Linux, способствует популярности этой операционной системы. Вместе с тем множество самодеятельных разработчиков создает более хаотичную платформу. FreeBSD — более традиционная система, поэтому вокруг нее меньше шума, но во многих отношениях она более мощная и более предсказуемая. Это правильный UNIX, с коммерческим кодом, таким же стабильным, как и традиционные версии UNIX от известных производителей "железа".

Вы уже работали с Linux и теперь ищите менее изменчивую платформу для корпоративного сетевого сервера? Или раньше вы имели дело с коммерческим UNIX, а теперь вам необходима легкая в использовании, но менее дорогая система? А может, вы системный администратор Windows-сервера, которому нужна альтернатива полностью закрытым операционным системам Microsoft? В любом из этих случаев FreeBSD станет для вас отличным выбором.

Я впервые обратился к FreeBSD в 1997 году, это была версия 2.2.2. Меня привлек тот факт, что даже на этом раннем этапе ее развития Yahoo! предпочла эту операционную систему Linux, а Hotmail (до приобретения ее Microsoft) использовала FreeBSD наряду с Solaris, высоко оценив необычайно высокую скорость обработки запросов пользователей. (В 2000 году Microsoft наконец-то смогла перевести большую часть оборудования Hotmail на Windows 2000, но, как стало известно, FreeBSD используется до сих пор, поддерживая ряд важнейших функций.)

С тех пор система FreeBSD претерпела значительные изменения. Улучшена компоновка системы; механизм защиты укреплен и усовершенствован. Поистине революционный набор портированных приложений для FreeBSD оказался весьма успешным и был перенесен в NetBSD, OpenBSD и Mac OS X; а модуль бинарной совместимости с Linux дает возможность запускать во FreeBSD программы для Linux, например, RealPlayer и StarOffice. Стандартизированые конфигурационные файлы и надежная файловая система способствуют большей предсказуемости платформы и более легкому управлению ею. Хотя FreeBSD не так популярна, как Linux, но зато ее и меньше ругают. Кроме того, она имеет ряд уникальных особенностей, которыми не может похвастать Linux.

Что касается открытых операционных системах, то Linux в ближайшем будущем, по-видимому, останется в центре внимания. FreeBSD, тем не менее, тоже продвигается вперед — благодаря компаниям, желающим выйти из под власти Microsoft. До сих пор они не решались переходить на Linux из-за жестких требований лицензии GPL и путаницы с дистрибутивами. FreeBSD предлагает таким компаниям разумную альтернативу. Другие операционные системы, выпускаемые по лицензии BSD, также имеют свои ниши на рынке: OpenBSD сфокусирована на том, чтобы оставаться самой безопасной операционной системой, а NetBSD функционирует практически на любых аппаратных средствах (и на платформе Intel x86, и на PowerPC, и даже на Sega Dreamcast).

Привлекательность FreeBSD заключается в ее универсальности; она незаменима в роли полнофункционального сервера или рабочей станции. Это особенно важно сегодня, когда все больше людей связывает свою деятельность с Internet. FreeBSD позволяет создать Web-сайт, домашнюю сеть, написать программу, заняться малым бизнесом и просто поделиться с миром своими мыслями, найти единомышленников и друзей. Научиться работать с этой системой может каждый: для этого не нужны специализированные знания, она доступна даже для финансовых менеджеров. Эта операционная система традиционно придерживается курса, который проходит точно посредине между двумя крайностями. Вполне возможно, что уже в ближайшее время она приобретет широкую известность благодаря росту числа пользователей, открывающих для себя эту операционную систему и ее неисчерпаемые возможности.

Трудно сказать, что ждет компьютерную индустрию через десять или двадцать лет. Тем не менее есть все основания предполагать, что FreeBSD будет с нами до тех пор, пока будут развиваться открытые программные средства. И не исключено, что однажды компромисс между открытыми и коммерческими средствами, который воплощает FreeBSD, станет единственно возможным решением.

Как построена эта книга

Книга состоит из шести частей:

- **Часть 1. Знакомтесь: FreeBSD**. В этой части представлены подробные указания и советы по установке и настройке FreeBSD.
- **Часть 2. Использование FreeBSD**. Здесь можно узнать о настройке среды Gnome и ее возможностях, а также о работе с приложениями и командными интерпретаторами.
- **Часть 3. Администрирование FreeBSD**. В этой части описаны программные инструментальные средства и административные процедуры, а также приведен ряд советов относительно повседневной эксплуатации FreeBSD, в том числе и по конфигурированию ядра.
- **Часть 4. FreeBSD и работа в сети**. Конфигурирование основных сетевых служб, защита сети и совместное использование ресурсов.
- **Часть 5. X-Window**. Все о настройке графического интерфейса пользователя.
- **Часть 6. Приложения**. Эта часть содержит ссылки на описания наиболее важных команд FreeBSD, перечень возможных проблем при инсталляции и загрузке, а также методы их решения.

Информация, на которую следует обратить особое внимание, выделена в блоки. Есть три типа блоков: примечание, совет и предостережение.

Примечания включают комментарии, касающихся рассматриваемой в данный момент темы, а также дают полные определения некоторых терминов.

Советы расшифровывают аббревиатуры и проясняют намеки относительно того, как сделать работу в системе FreeBSD более эффективной.

Предостережения даются для предотвращения действий пользователя, которые могут создать проблемы.

В книге используются следующие типографские соглашения:

- Строки кода, команды, операторы, переменные и любой экранный текст набраны **моноширинным шрифтом**. Команды, которые предлагается ввести с клавиатуры, набраны **полужирным моноширинным шрифтом**.
- В описаниях синтаксиса символы-заместители набраны ***полужирным курсивом***. Введите вместо символа-заместителя имя файла, параметр или необходимый элемент.
- *Курсивом* выделяются технические термины при их первом появлении в тексте.
- Специальный значок ➥ обозначает, что код в действительности должен размещаться на одной строке. Увидев значок ➥ перед строкой кода, имейте в виду, что он указывает на продолжение предыдущей строки.

1 ЧАСТЬ

Знакомьтесь: FreeBSD

Введение во FreeBSD

Установка FreeBSD

Сложные аспекты установки

1 ГЛАВА

Введение во FreeBSD

- Преимущества FreeBSD
- Что можно делать при помощи FreeBSD?
- Краткая история FreeBSD и UNIX
- Философия UNIX
- Преимущества FreeBSD
- Талисман FreeBSD

FreeBSD — это операционная система типа UNIX. Она работает на архитектурах Intel x86 и Alpha, и в настоящее время ведутся работы по переводу ее на архитектуру Sparc и PowerPC. Добровольцы со всех концов света трудятся над совершенствованием FreeBSD, и исходный код этой системы доступен для всех желающих.

Преимущества FreeBSD

Наверное, существует столько причин для пользования FreeBSD, сколько людей, ею пользующихся. Главной причиной, безусловно, является то, что FreeBSD — бесплатная система, не обременяющая пользователя дорогой лицензией. Вы можете бесплатно установить копию FreeBSD на всех своих компьютерах, сколько бы их ни было. Если вы устанавливаете сервер, вам не потребуется платить за каждое подключение или за дополнительных пользователей, как в некоторых коммерческих сетевых операционных системах. Операционная система FreeBSD имеет и другие преимущества:

- **Система чрезвычайно стабильна**. По данным компании Netcraft (**www.netcraft.com**), изучавшей сайты с самым продолжительным календарным временем беспрерывной работы, из 50 первых в ее списке сайтов 47 функционирует под управлением FreeBSD. С момента последней перезагрузки Web-сервера №1 прошло уже 1133 дня! И конечно же, он работает под FreeBSD.
- **Системой FreeBSD пользуются крупнейшие компании и интенсивно используемые сайты**: Sony; Yahoo!; The Apache Project и **freesoftware.com** — самый популярный и посещаемый FTP-сайт в мире.
- **Система открыта**. Доступно все дерево исходного кода этой операционной системы, в код можно вносить изменения, выполнять любые проверки защиты и т.д.
- **Доступны тысячи бесплатных пакетов прикладных программ**. Под FreeBSD работают тысячи свободно распространяемых программ для решения любых задач: игры, офисные приложения, графическое ПО, а также самый известный в мире Web-сервер.

Что можно делать при помощи FreeBSD?

Поскольку система FreeBSD устанавливается вместе с компиляторами для многочисленных языков программирования, ее возможности ограничены лишь возможностями имеющихся аппаратных средств. FreeBSD используется очень широко — начиная от совместной работы с файлами и заканчивая мощными высококачественными спецэффектами и генерированием компьютерной анимации. FreeBSD может работать на старом компьютере с 486 процессором и на мультипроцессорных системах, объединенных в мощные кластеры.

ПРИМЕЧАНИЕ

Спецэффекты для кинофильма "Матрица" кинокомпании Warner Brothers были созданы на кластере под управлением FreeBSD.

Ниже перечислены наиболее общие применения систем FreeBSD, не требующие ни специальных знаний в области программирования, ни значительных денежных средств:

- **Экономичное решение для совместного использования файлов и принтера**. Свободно распространяемый набор программ Samba (подробнее см. главу 32) позволяет организовать совместное использование файлов и принтеров с компьютеров, работающих под управлением разных операционных систем, в том числе и Windows. В этом случае FreeBSD может выступать в роли главного контроллера домена (Primary Domain Controller) для сети Windows.
- **Web-сервис**. Как уже говорилось, система FreeBSD управляет работой самых загруженных Web-серверов. Даже если вы пока не планируете создавать собственный Internet-сайт, FreeBSD — хорошая основа для большого корпоративного сервера.
- **Электронная почта**. С помощью FreeBSD можно запустить почтовый сервер компании даже на базе устаревшего 486-го компьютера.
- **Маршрутизация, DNS-сервер и NAT**. Даже старую систему можно превратить в полезный маршрутизатор, DNS-сервер или, скажем, в гейт для подключения к Internet нескольких пользователей через одно соединение.
- **Экономичные базы данных**. Используя бесплатные базы данных с поддержкой SQL, можно создать бесплатное решение, не уступающее многим коммерческим программам стоимостью в десятки тысяч долларов. Кстати, под FreeBSD можно запустить даже СУБД Oracle.
- **Лицензионная политика**. FreeBSD имеет очень либеральную лицензию, позволяющую бесплатно использовать исходный код системы при разработке собственных приложений. Для специалиста по разработке встроенных систем это прекрасное решение.

FreeBSD — не только для серверов

Кроме серверных решений, существует масса иных применений FreeBSD, которые вас наверняка заинтересуют. Например:

- **Разработка и тестирование Web-сайтов**. Время, когда Web-страницы отображали статический код HTML, уже стало историей. Сегодня Web-страницы используют серверные технологии, такие как встроенные и CGI-сценарии, а также запросы к базам данных, позволяющие создавать динамический и интерактивный контент. Чтобы разработать Web-сайт любой сложности, потребуется Web-сервер, пригодный для разработки и тестирования. FreeBSD прекрасно справится с этой работой.
- **Разработка баз данных в автономном режиме**. FreeBSD дает возможность разрабатывать и тестировать базу данных для Web-сайта полностью автономно.
- **Разработка программ или изучение программирования**. Изучающих языки программирования порадует тот факт, что, используя FreeBSD, можно сэкономить на компиляторах и программах отладки. Все, что понадобится при изучении программирования и написании мощных приложений, поставляется вместе с системой.
- **Недорогая рабочая станция**. Благодаря наличию в системе бесплатных программ с помощью FreeBSD можно создать недорогую и в то же время мощную рабочую станцию для выполнения практически любых задач.

Следующий раздел дает общее представление об истории FreeBSD и UNIX, а также об удивительных конструктивных особенностях UNIX, благодаря которым эта система (через 30 лет после ее создания!) является движущей силой всей компьютерной индустрии.

Краткая история FreeBSD и UNIX

У истоков операционной системы UNIX стояла Bell Laboratories компании AT&T. Два человека — Кен Томпсон (Ken Thompson) и Деннис Ритчи (Dennis Ritchie) — были главной движущей силой развития UNIX.

Операционная система UNIX родилась случайно. В середине 60-х годов AT&T Bell Laboratories совместно с другими компаниями прикладывала немало усилий для разработки новой операционной системы под названием Multics. Ее предполагалось использовать для крупномасштабных вычислений, которые выполнялись на машине класса мэйнфрейм. Кен Томпсон написал небольшую компьютерную игру, но ему не нравились ни производительность мэйнфрейма, ни стоимость машинного времени. С помощью Денниса Ритчи он переписал эту игру для работы на компьютере DEC PDP-7 и по ходу дела написал целую операционную систему.

Весной 1969 года Bell Labs вышла из проекта, и программисты Computing Science Research Center остались без вычислительной среды. К этому моменту они разработали базовую структуру файловой системы, которая впоследствии превратилась в файловую систему UNIX. Первые версии UNIX были написаны на языке ассемблер: вначале для DEC PDP-7, а затем — для DEC PDP-11. В 1973 году UNIX была переписана на C — совершенно новом языке программирования, разработанном Ритчи.

Создание языка программирования C и системы UNIX — две самые важные вехи в истории компьютерной индустрии. Язык C стал первым мультиплатформным языком, который позволил относительно легко переносить приложения, написанные на нем, между различными компьютерными платформами. Поскольку UNIX написана на языке C, она сравнительно легко переносится между платформами. Это одна из многих возможностей, благодаря которым система UNIX стала такой популярной.

Рождение системы BSD

AT&T в принципе не занималась компьютерным бизнесом (отчасти потому, что в то время это было монополией правительства). Поэтому AT&T предложила UNIX в виде исходных кодов правительственным учреждениям и университетам за сравнительно небольшую плату. Вот так система UNIX попала в 80% университетов, имевших компьютерные факультеты. Одной из первых организаций, вплотную занявшихся работой над UNIX, стала группа из Калифорнийского университета в Беркли — Computer Systems Research Group. Этому способствовал и тот факт, что в 1975 году Кен Томпсон оставил Bell Labs и перешел в отдел компьютерных исследований в Беркли. В работе над расширением системы ему активно помогал студент-выпускник Билл Джой (Bill Joy). Калифорнийские студенты и преподаватели внесли значительный вклад в систему UNIX. Измененная и скорректированная в университете версия была выпущена под названием Berkeley Software Distribution, или BSD. Наиболее значимые изменения, сделанные в Беркли, — это редактор **vi** и командный процессор C (C shell).

В конце 70-х годов произошло важное событие: Министерство обороны США объявило, что ее подразделение Advanced Research Project Agency будет использовать UNIX и что в качестве базовой принята версия разработчиков из Беркли. Одним из требований, поставленных министерством обороны, была возможность работы в сети и высокая устойчивость системы. Так, благодаря военным, UNIX стала продвигаться вперед по пути совершенствования.

В это время Билл Джой оставил университетский городок и основал компанию Sun Microsystem. Рабочие станции Sun использовали версию операционной системы, производную от BSD и известную как SunOS.

BSD на платформах Intel x86

Большая часть исходного кода BSD была доступна пользователям бесплатно. В 1991 году BSD была портирована на платформу Intel x86. Эта версия операционной системы была названа 386/BSD. А в Калифорнийском университете образовалась новая коммерческая группа, которая начала продавать коммерческую версию BSD для платформы x86.

Рождение FreeBSD

В 1993 году две совершенно разные группы одновременно пришли к выводу, что UNIX заслуживает большего внимания. В результате были созданы два новых проекта. Результатом первого проекта стала операционная система NetBSD. Здесь основное внимание уделялось доступности и универсальности системы. Если существует аппаратная платформа, то наверняка имеется и работающая на ней версия NetBSD. Второй проект породил FreeBSD. В этой разработке внимание было сконцентрировано на том, чтобы система стала проще в использовании. Иначе говоря, эта система была ориентирована на широкий круг пользователей и на платформу Intel x86. Сегодня FreeBSD — самая известная UNIX-система из семейства BSD.

В истории FreeBSD и UNIX можно обнаружить целый ряд интересных событий. Если вы хотите получить более детальную информацию, касающуюся истории FreeBSD и UNIX, обратитесь на следующие сайты:

- **http://www.bell-labs.com/history/unix/** — подробная история разработки UNIX в Bell Labs с фотографиями.
- **http://daemonz.org/bugs/history.ehtml** — история Berkeley Software Distribution.
- **http://www.freebsd.org/handbook/history.html** — детальная история разработки FreeBSD.

Философия UNIX

Почему же система UNIX остается такой же мощной и популярной, как и 30 лет назад? Причин много. Это и переносимость, и конструктивные особенности этой операционной системы, а главное — философия, заложенная в ее фундамент.

Многие видят в UNIX чрезвычайно запутанную, сложную и непонятную операционную систему. На самом деле UNIX являет собой яркий пример разработки операционной системы по принципу KISS (Keep It Simple, Stupid).

Конструктивно UNIX базируются на множестве небольших программ, каждая из которых отлично решает совершенно определенную задачу. Кроме того, разработчики UNIX заложили в эту систему совершенно замечательную идею: при необходимости

несколько маленьких, не связанных между собой программок можно объединить в единое целое, решая тем самым задачи любой сложности. Исходная философия по-прежнему живет в основных командах, доступных на всех системах UNIX. Ее ключевые элементы:

- Простые команды
- Команды, соединенные каналами (pipe)
- Преимущественно общий стиль интерфейса
- Отсутствие типов файлов

ПРИМЕЧАНИЕ

Концепция объединения известна как конвейер (piping). Ее "отцом" считается Дуг Маклори (Doug McIlory) из Bell Labs. Томпсон реализовал ее в UNIX (каналы будут подробно рассмотрены в главе 8).

Приведем пример функционирования каналов. Предположим, что существует простой текстовый файл, представляющий собой адресную книгу. Каждому пользователю в нем отведена одна строка, содержащая имя, адрес (в том числе e-mail), номер телефона и т.д. Поля файла разделены знаком тильда (~). Строки файла могут выглядеть примерно так:

```
Doe,   John~505   Some   Street~Anytowm~NY~55555~505-555-1212~jdoe@email.com
Doe,   Jane~121   Any   Street~Sometown~NY~12121~121-555-1212~jadoe@isp.com
Bar,   Foo~501   Some   Street~Anytown~NY   55555~505-123-4567~foobar@email.com
```

Файл может состоять из 50 или из 500 имен — число не имеет значения. Предположим, что на базе этого файла нужно составить список жителей города Anytown, отсортировать список в алфавитном порядке, а затем создать его бумажную копию.

Ни одна команда сама по себе не в состоянии выполнить все эти действия, но если объединить несколько команд в один канал, сделать все необходимое будет очень просто. Ниже приведен один из способов выполнения данной задачи:

```
awk 'BEGIN {FS="~"} $3 == "Anytown" {print "%s\t%s\n",$1,$6}'
➥address.txt | sort | lp
```

Этот код читает информацию из файла **adress.txt**. Используя тильду как разделитель полей, он отбирает строки, где в третьем поле указан Anytown. Затем из каждой строки выбирается первое и шестое поле (имя и номер телефона), разделяемые символом табуляции (\t), каждая строка завершается символом новой строки (\n). Результат передается на стандартный вывод. После чего полученные данные подаются по каналу команде сортировки, которая сортирует их в алфавитном порядке. После этого данные опять-таки посредством канала попадают на команду **lp**, которая выводит их на принтер, заданный по умолчанию. Окончательный документ будет содержать искомые записи:

```
Bar,   Foo   505-123-4567
Doe,   John  505-555-1212
```

Таким образом, простейшими средствами была создана работоспособная база данных, способная отыскать любое поле и вывести данные в нужном виде. При этом все необходимые операции обеспечивает всего одна строка кода! Ниже приведен второй пример, который позволяет вывести простой список адресатов:

```
awk 'BEGIN {FS="~"} $3 == "Anytown" {printf "%s\n%s\n$s, $s
$s\n\n",$1,$2,$3,$4,$5}' address.txt | lp
```

Результат работы будут выглядеть так:

```
Doe, John
505 Some Street
Anytown, NY 55555

Bar, Foo
501 Some Street
Anytown, NY 55555
```

Операционные системы типа UNIX (в том числе и FreeBSD) обладают поистине безграничным потенциалом, который каждый из вас может "высвободить" и использовать для выполнения всевозможных задач. Именно эти возможности системы и не дали UNIX выйти из игры.

Преимущества FreeBSD

В этом разделе мы сравним возможности операционной системы FreeBSD с возможностями Windows 2000 и Linux.

Windows 2000 и FreeBSD

Microsoft поступила гениально, разработав операционную систему, которой может пользоваться каждый дурак. Windows 2000 способна выполнять разнообразные задачи, не требуя от пользователя глубоких знаний внутреннего функционирования системы. С одной стороны, Windows 2000 отвечает самым высоким требованиям, предъявляемым к техническим средствам, однако многим пользователя они совершенно ни к чему. С другой стороны, Windows 2000 не имеет интерфейса к целому ряду имеющихся в ней возможностей и тонких настроек. И наконец, "искушенные" пользователи (читай: хакеры) могут легко "нарушить" границы дозволенного, установленные графическим интерфейсом системы.

Каковы же различия между FreeBSD и Windows 2000?

- **Ядро Windows 2000 невозможно изменить.** Ядро — это сердце операционной системы; оно контролирует все аспекты ее работы. FreeBSD позволяет создать новое ядро, которое будет максимально соответствовать назначению конкретной операционной системе. Благодаря этому возрастает быстродействие и снижаются требования к аппаратным ресурсам. В Windows 2000 предпочтение отдается простоте эксплуатации, а не производительности и эффективному использованию аппаратных средств.
- **Windows 2000 предлагает графический интерфейс для решения большинства задач.** Система FreeBSD основана на командной строке. С помощью графического интерфейса можно, в принципе, более или менее просто выполнить любую задачу по настройке системы. Но это наверняка потребует массу времени и не всегда полученный результат вас удовлетворит. Что касается FreeBSD, то в ней для настройки используются текстовые конфигурационные файлы, редактируя которые можно выполнить необходимые действия быстро и точно.
- **Графический интерфейс — неотъемлемая примета Windows 2000**, тогда как во FreeBSD можно и вовсе обойтись без него. Специалисту не нужны окошки для доступа к серверу, который стоит в дальней комнате. FreeBSD позволяет вы-

полнять все необходимые операции по его администрированию, используя исключительно командную строку.

- **Управлять FreeBSD очень просто.** Все административные задачи FreeBSD можно выполнять с удаленного терминала, даже самого простейшего. Управлять FreeBSD можно с машин, базирующихся на другой платформе, например, с ПК под Windows, с Macintosh и др. Хотя Windows 2000 тоже можно администрировать удаленно, но для этого нужно специальное программное обеспечение, которое подходит исключительно для Windows. Это значит, что задачи удаленного администрирования Windows-систем можно выполнять только с другой системы Windows.
- **Графический интерфейс имеет ряд ограничений,** от которых свободна командная строка. Графический интерфейс пользователя может вместить только строго определенные возможности. Рано или поздно опытному пользователю потребуется что-либо, что не предусмотрено разработчиками этой операционной системы. Вначале кажется, что работать с командной строкой сложно, но очень скоро понимаешь, что набрать нужную команду гораздо быстрее, чем добиться того же самого эффекта с помощью системы разветвленных меню.

Linux и FreeBSD

О системе Linux сейчас знают все. В последнее время она стала особенно популярной. Фактически Linux — это клон UNIX. Как и FreeBSD, это открытая операционная система, разработанная добровольцами из разных стран мира. У FreeBSD и Linux много общего. Это прекрасные операционные системы, отвечающие нуждам практически любого пользователя. Хотя для Linux создано больше программ, чем для FreeBSD, но последняя позволяет запускать практически все программы, разработанные для Linux. Более того, под FreeBSD они работает даже быстрее, чем под Linux.

Ниже приведены наиболее существенные различия между FreeBSD и Linux:

- **У FreeBSD только один дистрибьютор, а у Linux их более 30.** FreeBSD будет работать одинаково на любой системе. В случае с Linux это не так. У каждого дистрибьютора свой подход. Например, Slackware Linux использует управляющие сценарии типа BSD. Debian Linux — управляющие сценарии Sys V, а Red Hat Linux не просто использует сценарии Sys V, но и сохраняет их не там, где этого требует стандарт Sys V. Это вводит пользователей в заблуждение, когда они переходят с одного дистрибутива Linux на другой.
- **FreeBSD является полноценной операционной системой,** поддерживаемой основным составом; Linux — это только ядро, поддерживаемое Линусом Торвальдсом. Компании, занимающиеся распространением Linux, комплектуют свои дистрибутивы целым рядом программ, специально разработанных для Linux. Поскольку каждый дистрибьютор имеет собственное мнение относительно того, что должно входить в дистрибутив, вполне возможно, что программ, которые доступны в одной системе Linux, не окажется в другой. Это часто вызывает проблемы при модернизации Linux. Например, можно обновить ядро Linux и обнаружить, что необходимы еще некоторые пакеты. Поскольку FreeBSD является полноценной операционной системой, ее модернизация обычно осуществляется легче, поскольку все зависимости синхронизированы.

- **Любой пользователь имеет возможность поместить свой код в Linux.** Поступления во FreeBSD рассматривает и допускает в систему основной состав разработчиков. Процесс обновления кода отслеживается и координируется намного тщательнее, чем в Linux. Для большинства пользователей это позитивное явление, поскольку они уверены в том, что код был протестирован специалистами на отсутствие проблем.

Поскольку в системе FreeBSD поддерживается одно дерево исходного кода, она стабильнее Linux и в большей степени соответствует производственным целям. Основным недостатком FreeBSD, вызванным таким подходом, является то, что нововведения допускаются в систему медленней, чем в Linux. Но есть выбор: либо вы предпочтете стабильность производственной среды, либо остановите свой выбор на моднейших вещичках и новейших игровых устройствах, пожертвовав ради этого надежностью.

Талисман FreeBSD

Символом FreeBSD является демон. Он в полном смысле этого слова хозяин системы. Демонами называют фоновые процессы в системах UNIX, выполняющие всевозможные задачки. Демоны очень полезны. Если вы хоть раз отправляли сообщения по электронной почте или посещали Web-страницу, то наверняка пользовались их услугами, сами того не подозревая. У Windows 2000 они тоже имеются. Просто там они называются по-другому: Microsoft дала им название "сервисов".

ГЛАВА 2

Установка FreeBSD

- Проверка оборудования
- Создание загрузочных дискет
- Инсталляция
- Введение в программу Sysinstall
- Навигация по Sysinstall
- Создание разделов и выбор точек монтирования
- Выбор типа установки
- Выбор метода установки
- Настройка после инсталляции
- Выход из программы Sysinstall и перезагрузка системы
- Первая загрузка FreeBSD
- Отключение системы FreeBSD

Чтобы начать работать с системой FreeBSD, ее сначала необходимо установить на жесткий диск. Это относительно простой процесс: просто следуйте инструкциям, изложенным в этой главе, и проблем с установкой не возникнет. В зависимости от того, насколько быстро работает система в целом и CD-привод в частности, процесс установки займет от 20 минут до часа. Большая часть этого времени уйдет на копирование файлов, поэтому неотлучно сидеть перед компьютером не придется.

Прежде чем приступать к установке, тщательно все проверьте и спланируйте. Планирование потребует меньше времени, чем повторная инсталляция, которая наверняка потребуется, если установки, не будут соответствовать действительности. Чтобы процесс прошел благополучно, обязательно предварительно прочтите эту главу (желательно, и главу 3).

ПРЕДУПРЕЖДЕНИЕ

Убедитесь в том, что располагаете достаточным временем, чтобы спокойно и не торопясь выполнить все необходимые операции.

СОВЕТ

Сейчас самое время просмотреть приложение Б, где дан список оборудования, поддерживаемого FreeBSD. Не имеет смысла продолжать установку, если обнаружится, что большая часть ваших аппаратных средств не поддерживается системой FreeBSD.

Проверка оборудования

Перед тем как начать установку, необходимо удостовериться в том, что вы точно знаете, какие аппаратные средства используются в вашей системе. Убедитесь, что вам известна следующая информация:

- Изготовитель видеокарты, ее модель и объем видеопамяти.
- Технические характеристики монитора; для установки X-Window необходимо знать допустимую частоту горизонтальной и вертикальной развертки.
- Если есть модем, уточните, к какому порту он подключен и какое IRQ использует.
- Тип мыши (serial, ps/2 или bus) и число кнопок. Если мышь подсоединяется к последовательному порту, укажите его номер.
- Если есть сетевая карта, то нужно знать изготовителя карты, ее модель, адрес и IRQ.
- При наличии сети нужно знать: IP-адрес вашей машины; маску подсети; адрес шлюза; IP-адрес DNS-сервера; имя вашего домена; имя локального хоста (т.е. вашей системы).

Если какая-либо информация о вашем компьютере неизвестна, но в данный момент система работает под Microsoft Windows, недостающие сведения можно получить в Control Panel.

Создание загрузочных дискет

CD-диск с дистрибутивом FreeBSD, прилагаемый к этой книге, является загрузочным. Если ваша система не имеет CD-привода или планируется устанавливать

FreeBSD каким-либо другим способом (например, через сеть), необходимо создать две загрузочных дискеты (boot disks).

Имиджи загрузочных дискет хранятся в каталоге **floppies** на прилагаемом к книге дистрибутивном CD-диске. Если доступа к CD-приводу нет вообще, можно загрузить загрузочные имиджи с FTP-сервера **ftp.freebsd.org**. Они расположены в **/pub/FreeBSD/releases/5.0-RELEASE/floppies**. Необходимо загрузить два файла — **kern.flp** и **msfroot.flp**. Если загрузочные диски предполагается создавать в системе DOS или Windows, также потребуется программа **fdimage.exe**, расположенная в каталоге **/pub/FreeBSD/tools**.

СОВЕТ

Возможно, вы сможете быстрее загрузить нужные файлы с одного из зеркальных серверов **ftp.freebsd.org**. В большинстве случаев, за **ftp** сразу же следует число (например, **ftp1.freebsd.org** или **ftp2.freebsd.org**). Использование зеркал может значительно ускорить скачивание файлов.

Вам также потребуются две новых дискеты объемом 1,44 Мб.

ПРИМЕЧАНИЕ

Файлы, содержащие имиджи загрузочных дисков нельзя просто скопировать на гибкий диск. Они должны быть записаны с использованием специальных методов, описанных ниже.

СОВЕТ

Пользуйтесь новыми дисками для создания загрузочных дискет, поскольку утилиты записывают на диски необработанные данные, не принимая во внимание их формат. Если на диске будет дефектный сектор, при установке системы это может вызвать практически не решаемые проблемы.

Создание загрузочных дискет в DOS или Windows

Перед тем как создавать загрузочные дискеты в системе Windows, необходимо загрузиться в режиме DOS. Если вы, работая в Windows, попытаетесь создавать загрузочные дискеты из окна DOS, наверняка возникнут проблемы. Если получить доступ к CD-приводу из режима DOS невозможно, временно скопируйте в любое место на жестком диске следующие файлы с дистрибутивного CD-диска:

- **D:\TOOLS\FDIMAGE.EXE**
- **D:\FLOPPIES\BOOT.FLP**
- **D:\FLOPPIES\MFSROOT.FLP**

Предполагается, что D: — это CD-привод системы. Загрузив систему в режиме DOS, можно приступать к созданию загрузочных дискет.

Создание загрузочных дискет с дистрибутивного CD

Вставьте дистрибутивный CD-диск в CD-привод и поместите первую чистую дискету в floppy-привод. Затем наберите в командной строке DOS следующие команды. Если вашему CD-приводу соответствует другой логический диск (не D:), внесите в команды соответствующие изменения.

```
C:\> cd d:\tools
D:\TOOLS> fdimage \floppies\boot.flp a:
```

Когда программа выполнит эту задачу, вставьте в дисковод вторую дискету и введите следующее:

```
D:\TOOLS> fdimage \floppies\msfroot.flp a:
```

Если доступа к CD-приводу нет

Перейдите в каталог на жестком диске, в который вы временно скопировали файлы **fdimage.exe**, **boot.flp** и **msfroot.flp** и введите в командной строке DOS следующие команды, заменяя каталог **temp** тем каталогом, куда скопировали файлы:

```
C:\>cd temp
C:\TEMP> fdimage boot.flp a:
```

После записи первой дискеты вставьте вторую и введите следующее:

```
C:\TEMP> fdimage msfroot.flp a:
```

Создание загрузочных дисков на другой FreeBSD- или UNIX-системе

Если диски создаются на другой FreeBSD- или UNIX-системе, программа **fdimage.exe** не потребуется (однако файлы **boot.flp** и **msfroot.flp** будут по-прежнему необходимы).

Используйте UNIX-утилиту **dd** для записи файлов на дискету. В системе FreeBSD это будет иметь такой вид:

```
dd if=boot.flp of=/dev/rfd0
```

По окончании копирования вставьте вторую дискету и используйте команду **dd** для создания второго диска:

```
dd if=msfroot.flp of=/dev/rfd0
```

Теперь, после создания загрузочных дискет, можно перейти непосредственно к процессу установки.

Инсталляция

Мы предполагаем одно из двух: либо установка FreeBSD будет производиться на новый жесткий диск, либо вы уже освободили для FreeBSD достаточный объем дискового пространства (см. главу 3). Мы также предполагаем, что установка производится с CD. Если в вашем случае это не так, прочтите соответствующие разделы, описывающие установку по сети, FTP или с дискет. После этого вернитесь назад и продолжайте установку, руководствуясь предложенными здесь инструкциями.

Вставьте дистрибутивный диск в CD-привод. Если ваша система не может загружаться с CD-диска, то произведите загрузку с дискет, для чего вставьте в дисковод a: загрузочную дискету с файлом **boot.ftp**. Перезагрузите систему.

Когда система выполняет начальную загрузку, на экран выводятся разные сообщения и так называемая "волшебная палочка". Пока она вращается, все нормально — система загружается. Если вращение остановилось — система зависла. В нормальной ситуации система загрузит некоторые данные, после чего появится следующее сообщение:

```
FreeBSD/i386 bootstrap loader, Revision 0.8
    (jkh@bento.freebsd.org, Mon Nov 20 11:41:23 GMT 2000)
|
Hit [Enter] to boot immediately, or any other key for command prompt.
Booting [kernel] in 9 seconds... _
```

Для продолжения нажмите Enter. По ходу дела появится сообщение с просьбой заменить дискету на ту, которая содержит **mfsroot**. Сделайте это и нажмите Enter. Когда ядро загрузится, вы попадете в программу **UserConfig**, которая выглядит так:

```
                    Kernel Configuration Menu

Skip kernel configuration and continue with installation.

     Start kernel configuration in full screen Visual mode.
     Start kernel configuration in CLI mode.

Here you have the chance to go into kernel configuration mode, making
any changes which may be necessary to properly adjust the kernel to
match your hardware configuration.

If you are installing FreeBSD for the first time, select Visual Mode
(press Down-Arrow then ENTER).

If you need to do more specialized kernel configuration and are an
experienced FreeBSD user, select CLI mode.

If you are certain that you do not need to configure your kernel then
simply press ENTER or Q now.
```

Большинству пользователей следует выбрать Skip kernel configuration и продолжить процесс установки. Если в процессе установки возникли трудности, обратитесь к приложению Б, возможно, вы найдете там разрешение проблем, связанных с аппаратными средствами.

После выхода из программы **UserConfig**, ядро завершит процесс начальной загрузки.

Введение в программу Sysinstall

Когда ядро завершит процесс загрузки, вы попадете в программу FreeBSD **Sysinstall**. Первым на экран будет выведено основное меню, показанное на рисунке 2.1.

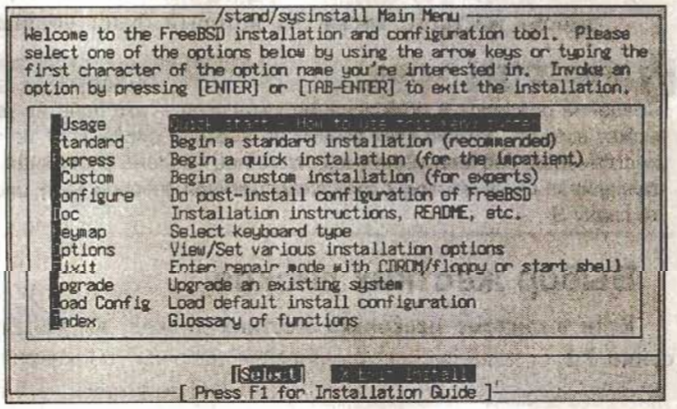


Рисунок 2.1
Основное меню Sysinstall.

Навигация по Sysinstall

Работая с **Sysinstall**, нельзя пользоваться мышью, тем не менее управлять программой очень просто. Для этого используется клавиатура, список необходимых клавиш управления приведен в таблице 2.1.

Таблица 2.1 Клавиши управления

Клавиша управления	*Команда*
Стрелка вверх	Переход к предыдущей опции меню.
Стрелка вниз	Переход к следующей опции меню.
Стрелки влево/вправо	Переключение между вариантами выбора, внизу экрана. Так, в основном меню клавиши влево и вправо переключают опции **Select** и **Exit Install**.
Клавиша пробел	В меню, где возможен выбор многочисленных опций, пробел иногда служит для выбора текущей опции.
Клавиша табуляции	Работает так же, как и курсорные клавиши влево/вправо. Используется для перехода между полями, там, где необходимо вводить информацию.

Кроме того, большую часть опций можно выбрать посредством выделенных букв. Обычно это первые буквы в названии опции. Для получения дополнительной информации об использовании **Sysinstall** нажмите Enter на выделенной опции **Usage**. После выбора опции **Standard** появится сообщение, уведомляющее о разбиения диска на разделы в стиле DOS. Просто нажмите Enter для продолжения.

Создание разделов и выбор точек монтирования

Далее процесс может протекать следующим образом:

- Если в системе только один жесткий диск, вы попадете в редактор разделов FreeBSD.
- Если в системе несколько жестких дисков, на экран будет выведено меню для выбора жесткого диска, на который будет устанавливаться FreeBSD.

ПРЕДОСТЕРЕЖЕНИЕ

Создание разделов и подключение каталогов — это одна из тех областей, где любая ошибка может вызвать потерю всех данных на жестком диске. Если на жестком диске хранится ценная информация, сначала убедитесь, что с нее сделана резервная копия. Если FreeBSD будет устанавливаться на машину с уже установленной операционной системой, предварительно прочтите главу 3.

Выбор жестких дисков

Если в системе несколько жестких дисков, вы увидите меню, показанное на рисунке 2.2.

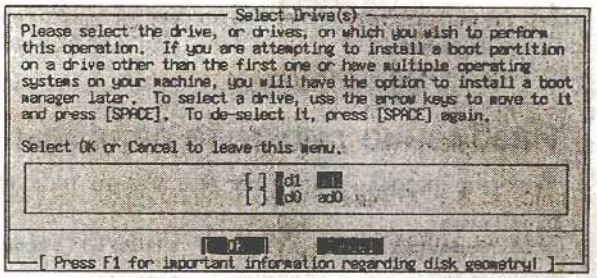


Рисунок 2.2
Выбор жестких дисков для установки.

Таблица 2.2 содержит некоторые данные, которые можно увидеть в меню, а также краткое описание.

Таблица 2.2 Данные меню

Данные	*Значение*
ad0	Первый физический ATA-диск в системе. Первый IDE-контроллер, ведущий жесткий диск. Если в системе установлена Windows или DOS, она будет расположена на этом приводе.
ad1	Второй ATA-диск в системе. В зависимости от того, каким способом устанавливается система, это может быть либо slave-диск на первичном контроллере, либо primary-диск на вторичном контроллере.
da0	da0 идентифицирует ведущий SCSI-диск. Если в системе имеются исключительно SCSI-диски, а также установлена система Windows или DOS, они будут расположены на этом приводе.
da1	Второй SCSI-диск.
fd	Дисковод гибких дисков

В вашем списке также могут присутствовать ad2 и ad3, либо da2 или da3 и т.д. Помните, что число после имени — это номер привода в системе; а также о том, что FreeBSD начинает отсчет приводов с нуля, а не с единицы.

Используйте клавишу табуляции, чтобы выбрать диски, которые будут использоваться для установки FreeBSD. Из следующего раздела вы узнаете, почему предпочтительно устанавливать систему на несколько разделов.

После выбора диска вы попадете в редактор разделов, где нужно отредактировать таблицу разбиения диска на разделы. После выхода из редактора разделов, вы возвратитесь назад, в меню, где может выбрать для редактирования другой диск. Нажмите Enter для выхода из данного меню и продолжайте установку системы.

ПРЕДУПРЕЖДЕНИЕ

Если FreeBSD устанавливается на втором жестком диске, а на первом находится другая операционная система, причем вы не планируете ничего менять на первом жестком диске, вам потребуется установить Boot-менеджер. В этом случае при загрузке вы сможете выбрать операционную систему, которая должна загружаться. Система FreeBSD предложит установить Boot-менеджер немного позже, в процессе инсталляции. Понятно, что Boot-менеджер нужно разместить на первом диске системы. Чтобы осуществить эту операцию, выберите в меню первый жесткий диск в качестве загрузочного. Если вы не хотите трогать первый жесткий диск, просто выйдите из редактора разделов. Затем выберите другой жесткий диск, на который будет устанавливаться только система FreeBSD, и разбейте его на разделы. Это даст FreeBSD возмож-

ность установить Boot-менеджер на первом диске. Если этого не сделать, Boot-менеджер не будет установлен на первый жесткий диск, и по окончании процесса установки вы не сможете загрузить систему FreeBSD.

Разбиение диска на разделы

Редактор разделов покажет вам что-то наподобие нижеследующего:

```
Disk name:          ad0                                FDISK Partition Editor
DISK Geometry: 4336 cyls/146 heads/63 sectors = 39882528 sectors (19473MB)

Offset    Size(ST)       End       Name    PType      Desc    Subtype      Flags

   0     39882528  39882527       ad0s1        3    freebsd       165          C


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

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

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

В данном примере первая строка указывает на то, что мы работаем с диском **ad0**. Вторая строка выводит информацию о геометрии данного диска. Несколько следующих строк отображают информацию относительно всех определенных на данный момент разделов системы. Если диск новый или перед установкой системы с него были удалены все разделы, весь объем диска будет представлять собой один раздел, определенный как unused. В таблице 2.3 дана информация о значении каждого информативного поля.

ПРИМЕЧАНИЕ

Глава 19 посвящена тонкостям использования жестких дисков и дает определения терминов, имеющих отношение к этой теме.

Таблица 2.3 Информация таблицы разделов

Поле	*Назначение*
Offset	Начальный сектор раздела.
Size (ST)	Размер секторов раздела.
End	Последний сектор раздела.
Name	FreeBSD определяет имя этого раздела (если оно известно).
Ptype	Число, указывающее тип раздела.
Descr	Тип раздела.
Subtype	Более детальная информация о типе раздела.
Flags	В этом поле могут употребляться такие обозначения:
	=: — Слайс (slice) выровнен правильно.

Поле	Назначение
	>: — Данный слайс простирается за 1024-тый цилиндр жесткого диска. Это положение будет рассмотрено позднее при создании этикеток диска в виде разделов boot, которые превышают объем в 1024 цилиндра. Это чревато повреждениями некоторых систем (это ограничение BIOS, не ограничение FreeBSD).
	R: — Данный слайс содержит файловую систему root.
	B: — Данный слайс использует плохо опознающую обработку BAD144.
	C: — Данный слайс является разделом FreeBSD.
	A: — Данный слайс является активным разделом (данная часть загружаема).

СОВЕТ

Нажмите **z** для переключения между единицами измерения (секторы, килобайты и мегабайты).

Чтобы предоставить FreeBSD все доступное пространство на жестком диске, выберите опцию A.

ПРЕДУПРЕЖДЕНИЕ

В предыдущих версиях FreeBSD существовала опция под названием dangerously dedicated (опасное назначение), которая использовалась для решения проблем с определением геометрии диска на старых системах. Сегодня при правильном конфигурировании FreeBSD подобные проблемы встречаются намного реже. Эта опция по-прежнему выводится на экран нажатием клавиши **F.** Однако, поскольку она намеренно не документирована, пользоваться ею не рекомендуется.

После нажатия клавиши Enter появится строка, отображающая единичный раздел типа 165, отданный под FreeBSD, и символ **C** в поле **Flags**. Нажмите **Q** для выхода из редактора разделов.

ПРИМЕЧАНИЕ

В отличие от Linux, в системе FreeBSD на одном диске обычно не создается несколько разделов. Чтобы создать отдельные разделы для разных файловых систем, используйте Disk Label Editor.

Если в системе только один жесткий диск, появится уведомление о создании разделов BSD внутри только что созданного с помощью **fdisk** раздела. Нажмите Enter для перемещения в Disk Label Editor. Если в системе несколько дисков — возвращайтесь в меню выбора диска.

Если есть необходимость разместить FreeBSD на нескольких дисках (либо из-за недостатка места для проведения полной установки, либо по другим причинам), выберите другие диски, повторяя шаги, описанные в предыдущем разделе этой главы.

Disk Label Editor

Именно здесь мы действительно создаем файловую систему, на которой будет жить FreeBSD. В первый раз редактор будет выглядеть примерно так:

```
                                    FreeBSD Disklabel Editor

Disk: ad1       Partition name: ad1s1     Free: 30033360 blocks (14664MB)
Disk: ad0       Partition name: ad0s1     Free: 39882528 blocks (19473MB)

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



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

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

Можно заметить, что в примере, вверху меню, показаны два диска. Это зависит от того, сколько дисков было выбрано для работы с FreeBSD.

Существует опция **A** (Auto Defaults). Если устанавливается рабочая станция и в вашем распоряжении для работы с FreeBSD имеется только один диск, можно использовать эту опцию как простейший способ установки файловой системы. Если на рабочей станции используется несколько дисков или вы устанавливаете сервер, необходимо вручную установить разделы, потому что опция Auto Defaults не выделяет достаточно пространства для файловой системы **/var**. К тому же, она размещает файловую систему **/tmp** в разделе **root** (определения **/var** и **/tmp** будут даны позже в этой главе). По нашему мнению, это не есть хорошо, поэтому имеет смысл потратить время на установку разделов вручную.

Сколько же разделов нужно для FreeBSD?

Необходимо создать как минимум два раздела: файловую систему **root** и раздел **swap**. Но это далеко не лучшее решение. Единственное преимущество одного большого раздела **root** над несколькими — это отсутствие проблем с нехваткой места в одном разделе и его переизбытком в другом. Тем не менее есть ряд причин, чтобы отказаться от идеи единого и неделимого **root** и создать несколько разделов.

FreeBSD — это мощная многозадачная операционная система. Она практически всегда находится в рабочем состоянии — особенно на загруженном сервере. В ней, как и в других системах UNIX, одновременно открыто несколько файлов, которые часто записываются на диск. Если происходит авария, отключается электричество или уборщик подключает свой пылесос к розетке питания сервера, файловая система может быть сильно повреждена. В зависимости от того, что конкретно записывалось в момент повреждения, поломка системы может оказаться весьма серьезной. Возможность повреждения файловой системы значительно снижается, если в момент отключения питания система не находилась в процессе записи.

Вот почему лучше пользоваться несколькими файловыми системами. Они не только помогают уменьшить последствия аварии до одной области вместо всей системы, но также защищают самую важную файловую систему — **root**. Самые важные файлы размещаются в разделах, которые редко изменяются, либо имеют атрибут только для

чтения. В правильно созданной файловой системе запись в раздел **root** осуществляется крайне редко.

Вторая причина, по которой не следует хранить все данные в разделе **root**, заключается в опасности самого факта хранения всей информации в одном разделе. Во-первых, в этой ситуации сервер оказывается уязвим для всевозможных DoS-атак (атака типа отказ в обслуживании). С другой стороны, систему нужно защищать от пользователей. Так, например, в системе, где важнейшая системная информация и данные пользователей хранятся в одном разделе, возможна следующая ситуация. Пользователь намеренно или случайно создает файл, который полностью заполняет дисковое пространство раздела. Как результат, происходит отказ системы электронной почты и системы печати, поскольку негде сохранять поступающие почтовые сообщения или задания принтера; Web-сервер прекратит работу, поскольку не сможет вести свой системный log-файл. **Syslogd** (программа, которая регистрирует сообщения компьютера) будет не в состоянии записывать важные сообщения, что само по себе даст хакерам лишний шанс вломиться в незащищенную систему. Все программы, требующие создания временных файлов, перестанут работать (или, что еще хуже, будут работать неправильно), поскольку нет свободного пространства для файловой системы **tmp**. Согласитесь, такая ситуация, по меньшей мере, неприятна.

Третья причина — управление доступом. Маркировку только для чтения может иметь лишь раздел в целом. Если нужно предохранить какие-то важные компоненты операционной системы от внешнего воздействия, желательно размещать их в отдельном разделе.

Итак, системный администратор должен придерживаться принципа разделяй и властвуй. А теперь, поскольку мы пришли к соглашению, давайте двинемся дальше и рассмотрим создание разделов.

Создание разделов и выбор точек монтирования

Я рекомендую создавать как минимум четыре раздела: один для файловой системы **root** (/), второй — для области **swap**, третий — для файловой системы пользователей (**/usr**) и последний — для файловой системы **/var**. Некоторые советуют помещать файловую систему **/var** вместе с **/usr** в одном разделе. С этим трудно согласиться, потому что определить, сколько места потребуется для **/var**, не так-то просто. Кроме того, подобный подход подвергает систему всевозможным опасностям, особенно если домашние каталоги пользователей размещены в том же разделе, что и файловая система **/usr**.

ПРИМЕЧАНИЯ

FreeBSD имеет возможность использовать больше памяти, чем имеется в действительности. Эта возможность называется *виртуальной памятью*. Используя виртуальную память, FreeBSD перемещает неиспользуемые в данный момент странички памяти из реальной памяти на жесткий диск. Когда перемещенная из диска страница памяти потребуется вновь, FreeBSD возвратит ее в память, а ненужные в данный момент данные из памяти переместит на диск. Подобные операции получили название *своппинг* (*swapping*). Часть диска, заполненная страницами памяти, называется областью подкачки, или разделом **swap**. Область подкачки дает операционной системе возможность оперировать большим объемом ОЗУ, чем есть на самом деле.

Принять правильное решение относительно размещения файловых систем вам поможет таблица 2.4, описывающая некоторые из каталогов системы FreeBSD и области применения каждого из них. Обратите внимание на то, что это не полный список; он включает только те каталоги, для которых желательно иметь отдельные разделы. Более полная информация о файловой системе FreeBSD приведена в главе 9.

Таблица 2.4 Каталоги FreeBSD и их назначение.

Каталог	*Назначение*
/	Это файловая система **root**. Она имеется на всех машинах UNIX. Это раздел занимает верхний уровень иерархии файловой системы. Все остальные файловые системы устанавливаются как подчиненные **root** (даже если они разных дисках, разделах, даже на разных компьютерах). Здесь находится ядро во всех версиях FreeBSD до версии 5.0. Файловой системе **root** необходимо выделить отдельный раздел. Оптимальным будет объем в 100 Мб.
/boot	Во всех версиях FreeBSD вплоть до 5.0 имелся каталог, где хранились конфигурационные файлы загрузчика операционной системы FreeBSD и некоторые другие файлы, необходимые для запуска системы. Начиная с версии 5.0. ядро и ряд других файлов, необходимых для запуска системы, также располагаются в этом каталоге. Теперь раздел **root** больше не подвержен прежнему ограничению (максимум 1024 цилиндра), поскольку **boot** размещается в отдельном разделе. (Обратите внимание, что раздел **/boot** должен полностью умещаться в первые 1024 цилиндра.)
/usr	Файловая система **/usr** содержит большую часть доступных обычным пользователям утилит и программ. **/usr** определенно нужен собственный раздел.
/usr/local	Здесь находится третья часть всего программного обеспечения, не являющегося частью системы (Web-серверы и базы данных). Многие предпочитают размещать **/usr/local** на отдельном разделе. Мы не рекомендуем такой подход для обычных систем. Разве что в наличии имеется несколько дисков или есть необходимость разделить **/usr** из-за недостатка дискового пространства.
/var	Этот раздел содержит буферные (spool) каталоги, используемые для организации очередей печати и работы электронной почты. Здесь же размещаются log-файлы. Мы предпочитаем помещать **/var** в отдельный раздел. Объем, необходимый для этого каталога, зависит от того, будет ли задействован сервер печати, сервер электронной почты или Web-сервер. Дело в том, что на загруженном Web-сервере log-файлы могут расти невероятно быстро. Если предполагается работа с активно используемым Web-сервером, выделите этому разделу достаточно места.
/tmp	В этом каталоге программы и пользователи могут записывать временные файлы. Каталог обычно очищается при каждой перезагрузке системы. Программам, которым необходимо записывать временные файлы большого размера, не следует разрешать использовать **/tmp**; в данном случае его можно заменить каталогом **/usr/tmp** или **/var/tmp**. Можно либо предоставить **/tmp** собственный раздел, либо разделять общее дисковое пространство между ним и **/usr**.
/home	Здесь хранятся домашние каталоги пользователя. Этот каталог часто располагается в **/usr**. Если планируется, что система будет обслуживать большое число пользователей, у каждого из которых будет множество файлов, можно выделить **/home** собственный раздел, либо отдать ему весь диск.

Оптимизация производительности

Если система располагает несколькими жесткими дисками, можно значительно ускорить ее работу, распределив задачи, интенсивно работающие с диском, между разными жесткими дисками. К примеру, если используется сервер новостей Usenet (как правило, сильно загруженный), а также Web-сервер (тоже достаточно загруженный), необходимо поместить каталог новостей на один диск, а каталог, в котором Web-сервер хранит свои страницы, — на другой.

Раздел root

Если у вас нет отдельного раздела **/boot**, то раздел **root** (/) станет первым создаваемым разделом. Для создания этого раздела в программе Disk Label Editor выделите диск, на котором должен быть создан этот раздел. Нажмите **C** для создания раздела. Будет выведено диалоговое окно с запросом о размере этого раздела (см. рис. 2.3).

Рисунок 2.3
Установка размера раздела.

Мы уже упоминали о том, что 100 Мб достаточно для раздела **root**. В соответствии с текстом в диалоговом окне можно задать размер в мегабайтах, добавляя **M** после числа. Поэтому введите в окне **100M** и нажмите Enter. Затем появится окно с запросом типа будущего раздела. Выберите опцию **FS A filesystem**, потому что именно она будет ответственна за файловую систему root, и нажмите Enter. Теперь появится окно, в котором нужно указать *точку монтирования* (*mount point*) этого раздела. *Mount point* — это каталог, к которому будет подключена файловая система. Введите в этом окне /, поскольку это корневая файловая система, и нажмите Enter. После этого экран примет такой вид:

```
                              FreeBSD Disklabel Editor

Disk: ad1       Partition name: ad1s1     Free: 30033360 blocks (14664MB)
Disk: ad0       Partition name: ad0s1     Free: 39882528 blocks (19473MB)

Part          Mount          Size  Newfs     Part      Mount      Size  Newfs
----          -----          ----  -----     ----      -----      ----  -----
ad0s1a        /              100MB UFS Y


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

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

Данный вывод показывает, что создана файловая система **ad1s1a** (имя устройства и слайс, используемый FreeBSD для обращения к этой системе файлов); она установ-

лена на /; размер — 100 Мб; тип — стандартная файловая система для FreeBSD (UFS). Далее перейдем к разделу **swap**.

Раздел swap

FreeBSD — это операционная система с виртуальной памятью: она может использовать больше памяти, чем физически доступно в системе. Это достигается путем сброса неиспользуемых в данный момент страниц памяти на жесткий диск. Конечно, доступ к данным на жестком диске выполняется намного медленней, чем к ОЗУ, поэтому пространство **swap** — это отнюдь не альтернатива наличию в системе достаточного количества памяти.

Размещение раздела **swap** может ощутимо повлиять на работу системы. Поэтому ниже приведены некоторые указания для выбора местонахождения данного раздела.

- Поместите раздел **swap** как можно ближе к началу диска. Доступ к внешним цилиндрам жесткого диска осуществляется быстрее, чем к внутренним.
- Если в системе несколько жестких дисков, то лучше поместить раздел **swap** на самом быстром приводе системы.

И все же: сколько места выделить разделу **swap**? Когда память была дефицитом, возникло правило, что размер области подкачки должен быть в 2–4 раза больше объема оперативной памяти. В настоящее время память сильно подешевела и стала гораздо более емкой. Вот почему это эмпирическое правило слегка устарело. Если на рабочей станции установлено 256 Мб памяти или более, то системе, по всей видимости, не понадобится большая область подкачки. При ОЗУ, равном 128–256 Мб, объем области подкачки можно рассчитывать по формуле 1:1. Если объем памяти больше или равен 512 Мб, то установка области подкачки, превышающей 256 Мб, для большинства систем будет лишена смысла, поскольку памяти вполне достаточно. Тем не менее совсем без области подкачки не обойтись. Ее наличие повысит общую стабильность и производительность системы.

В некоторых ситуациях ядро операционной системы не знает, как себя вести. При этом система выдает хорошо известное опытным администраторам сообщение о так называемой "панике ядра" (kernel panic). Обычно после этого приходится перезагружать систему. Но если ядро запрограммировано соответствующим образом, то оно не просто паникует, а и предпринимает попытки сбросить содержимое ОЗУ на диск, в раздел **swap**. В таких экстремальных ситуациях содержимое памяти представляет особую ценность для программистов, стремящихся выяснить причину kernel panic и предотвратить возникновение подобной ситуации в будущем. Что же произойдет, если в памяти информации больше, чем может вместить область подкачки? Возможны варианты:

- Ядро вообще откажется выгружать содержимое ОЗУ. В этом случае будет невозможно определить, что вызвало (или вызывает, если это случается регулярно) панику.
- Если паника была вызвана сбоем в структуре файловых систем, ядро, возможно, начнет выгружать содержимое памяти в раздел **swap**, пока не заполнит его до конца. Поскольку информации в памяти больше, чем вмещает раздел **swap**, будет непоправимо повреждена файловая система, следующая сразу за разделом **swap**.

Паника ядра — явление не слишком частое в системе FreeBSD, но, когда это все-таки происходит, совсем неплохо иметь функцию, позволяющую выгрузить содержимое ОЗУ на диск и исследовать в спокойной обстановке. Это даст возможность выяснить, что же вызвало подобное состояние ядра. А значит, объем области подкачки должен позволять сделать это.

ПРИМЕЧАНИЕ

Автор этой книги за семь лет работы с рабочей станцией FreeBSD столкнулся с kernel panic всего один раз.

Определившись с размером раздела **swap**, создайте его тем же способом, каким был создан раздел **root**, единственное исключение — это тип файловой системы (укажите **swap**). В разделе **swap** нет каталогов, поэтому в нем нельзя смонтировать какую-либо файловую систему.

Создание остальных разделов

После создания разделов **root** и **swap** (а возможно, и **boot**), создайте остальные разделы как файловые системы и укажите точки их монтирования в соответствующих каталогах (например, **/var**, **/usr**). Ниже приведен пример того, как может выглядеть окончательный результат:

```
                              FreeBSD Disklabel Editor

Disk: ad0      Partition name: ad0s1     Free: 0 blocks (0MB)
Disk: ad1      Partition name: ad1s1     Free: 0 blocks (0MB)

Part          Mount          Size  Newfs     Part      Mount      Size  Newfs
----          -----          ----  -----     ----      -----      ----  -----
ad0s1a        /            100MB  UFS Y
ad0s1b        swap         256MB  SWAP
ad0s1e        /var         200MB  UFS Y
ad0s1f        /tmp         100MB  UFS Y
ad0s1g        /usr       18817MB  UFS Y
ad1s1e        /home      14664MB  UFS Y

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

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

В этом примере показаны два жестких диска с разделами для системы FreeBSD, а каталог **/home** находится на втором жестком диске, занимая весь его объем.

Заметки о SoftUpdates

Начиная с версии 5.0 FreeBSD, в системе появилась опция, разрешающая запуск SoftUpdates прямо из Disk Label Editor. Более детально SoftUpdates рассматривается в главе 9, но сейчас важно знать, что эта опция может значительно ускорить работу системы. Чтобы установить SoftUpdates в файловую систему, просто выберите файловую систему и нажмите **S** для установления или снятия этой опции. Файловая си-

стема с установленной опцией SoftUpdates будет иметь значок +S сразу после типа файловой системы (например, **UFS+S**).

Закончив создание разделов, нажмите **Q** для выхода из редактора разделов. Не используйте опцию **W**, поскольку она предназначена для внесения изменений в существующую файловую систему, а не для создания новых. Следующее окно выведет запрос о том, что будет устанавливаться в систему.

Выбор типа установки

Если нет проблем с дисковым пространством, выбирайте (см. рис. 2.4) опцию **All**. Если вы ставите серверную систему и уверены, что X-Window не потребуется, выберите вариант **Develloper**. Если была выбрана опция, не включающая исходный код, то в дальнейшем будет невозможно создать новое ядро или модернизировать систему посредством **cvsup**. Если была выбрана опция, устанавливающая исключительно службы ядра, создать новое ядро будет можно, но модернизировать систему с помощью **cvsup** по-прежнему нельзя. Начинающим пользователям мы рекомендуем придерживаться стандартного варианта установки — вариант **All**.

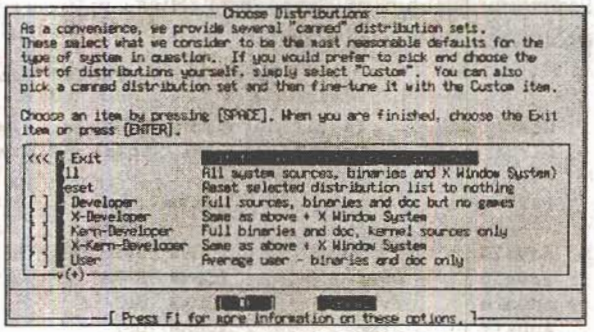


Рисунок 2.4
Меню выбора типов установки.

ПРИМЕЧАНИЕ

cvsup — это система, позволяющая автоматически поддерживать операционную систему FreeBSD в актуальном состоянии и вносить изменения, которые сделаны в системе с момента последнего запуска **cvsup**. Любые изменения автоматически загружаются с сервера и применяются к исходному коду FreeBSD. Этот процесс выполняется быстро, поскольку **cvsup** загружает только изменения (см. главу 18).

Чтобы выбрать тип установки, выделите желаемый тип и нажмите пробел. Если выбран вариант **All**, появится запрос о том, будет ли устанавливаться совокупность портов FreeBSD. Если позволяет дисковое пространство, в этом определенно есть смысл. Дело в том, что набор портов обеспечивает простую установку множества дополнительных программ, которые, возможно, снимут с ваших плечей большую часть "грязной работы". Используйте клавишу табуляции для выбора нужного варианта и нажмите Enter. Вы вернетесь в меню. Выберите **Exit** и нажмите Enter.

Выбор метода установки

Следующее окно попросит выбрать метод установки системы (см. рис. 2.5).

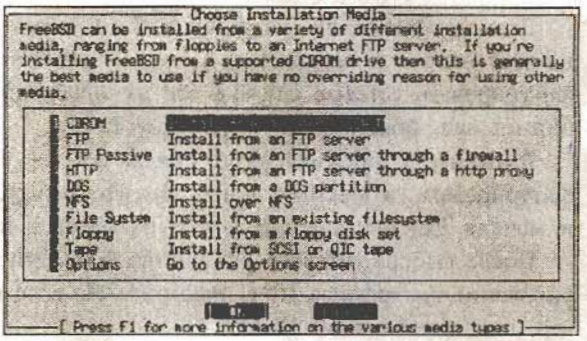


Рисунок 2.5
Выбор метода установки

Мы предполагаем, что вы устанавливаете систему с CD-диска. Если ваш случай не относится к стандартному (например, требуется установить систему по сети, с магнитной ленты или с дискет; либо же выполнить установку из уже существующей файловой системы), обратитесь к соответствующим разделам главы 3, прежде чем продолжать. Чтобы установить систему с CD-диска, выберите опцию CD-ROM и нажмите Enter. Система выведет предупреждение такого содержания:

```
                    User Confirmation Requested

Last Chance! Are you SURE you want to continue the installation?

If you're running this on a disk with data you wish to save then WE
STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!

We can take no responsibility for lost disk contents!
```

ПРЕДУПРЕЖДЕНИЕ

С этого момента на вашем жестком диске будут сделаны необратимые изменения. Это последний шанс остановить процесс инсталляции. Вы уверены, что хотите продолжать? После нажатия **Yes** созданные вами разделы отформатируются, и начнется процесс установки. Все данные, имеющиеся в выбранных для использования разделах, БУДУТ ПОТЕРЯНЫ!

Не передумали? Выберите **Yes**. Система выведет на экран окно с информацией "Starting an emergency holographic shell on vty4", затем приступит к форматированию разделов. Когда форматирование будет завершено, программа Sysinstall начнет копирование файлов на жесткий диск/диски.

Настройка после инсталляции

Когда система завершит процесс копирования файлов, программа установки поздравит вас с удачным ее завершением и сообщит о том, что сейчас программа Sysinstall перейдет к вопросам, касающимся конфигурирования системы. Для продолжения выберите **OK**.

Конфигурирование сети

Далее появится окно с запросом о том, будут ли конфигурироваться какие-либо сетевые устройства — Ethernet или SLIP/PPP. Эти действия можно выполнить прямо сейчас. Если необходимые данные неизвестны, всегда есть возможность сделать это позднее, после ознакомления с главами, рассматривающими создание и организацию сети.

Если конфигурация сетевых служб не нужна вообще, просто выберите **No** в ответ на этот вопрос и переходите к следующему разделу. Если на данный момент сконфигурировать сетевые службы все же необходимо, выберите **Yes**, после чего экран примет вид, показанный на рисунке 2.6.

Выберите из предложенного списка сетевое устройство, которое необходимо сконфигурировать, и нажмите Enter. Имейте в виду, что, в зависимости от изготовителя и модели Ethernet-адаптер может называться по-разному.

После выбора сетевого устройства поступит запрос о том, требуется ли конфигурирование интерфейса IPv6. Выберите **No** (кроме тех случаев, когда в сети есть сервер, поддерживающий IPv6).

Следующий вопрос будет о конфигурации DHCP-интерфейса. Если в сети есть DHCP-сервер, то FreeBSD попытается соединиться с ним и получить от него необходимую информацию о сети. Если в сети такого сервера нет, выберите **No**. В следующем окне можно ввести информацию о сети. Более детально этот вопрос будет рассмотрен в главах 22 и 23. А сейчас лишь несколько замечаний относительно конфигурирования сети (см. рис. 2.7).

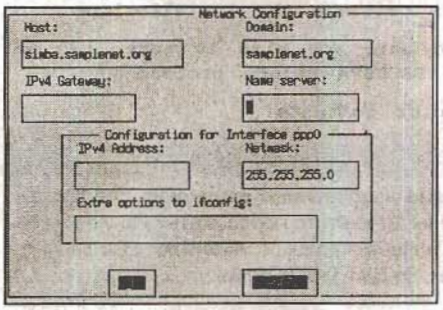


Рисунок 2.6 *Выбор сетевого интерфейса для процесса конфигурации.*

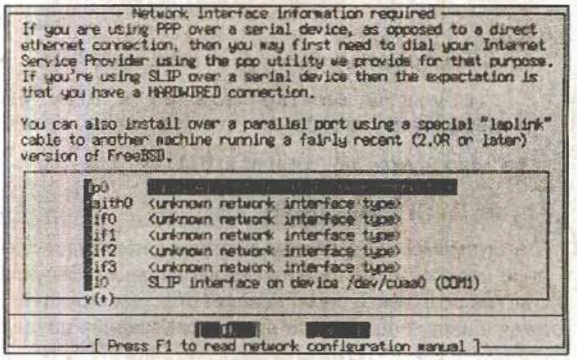


Рисунок 2.7 *Конфигурирование сетевого интерфейса.*

Табл. 2.5 разъясняет назначение отдельных полей.

Таблица 2.5 Конфигурирование сетевого интерфейса.

Поле	*Описание*
	Основные сетевые опции для данной системы
Host	Полное имя хоста (в данном примере, **simba.samplenet.org**).
Domain	Имя домена (**samplenet.org**).
IPv4 Gateway	Адрес шлюза для доступа к сети Internet.
Name Server	IP-адрес DNS-сервера.
	Сетевые параметры данной системы
IPv4 Address	IP-адрес данной системы
Netmask	Маска подсети
Extra Options to ifconfig	Дополнительные опции для ifconfig

Используйте клавишу табуляции или Enter для перемещения между полями. Когда все будет сделано, выберите **OK** для выхода из программы конфигурации. На вопрос о перезагрузке ответьте **no**, поскольку процесс установки практически закончен, так что система в любом случае будет скоро перезагружаться. Теперь **Sysinstall** задаст целый ряд вопросов относительно сети:

- **Станет ли данный компьютер шлюзом в сеть?**

  Если данный компьютер выступает в роли шлюза, который другие системы используют для доступа в Internet, маршрутизатора и т.п., выберите **Yes**.

- **Будет ли разрешен обычный FTP-доступ к этому компьютеру (без анонимного подключения по FTP)?** Мы предполагаем, что здесь лучше выбрать **Yes**. Всегда есть возможность сконфигурировать анонимный FTP-доступ позднее.

- **Будет ли данный компьютер NFS-сервером?** NFS-сервер позволяют совместно использовать каталоги и файлы на жестком диске. Выберите **No**. Если служба NFS установлена неправильно, она может стать причиной неполадок в системе защиты (см. главу 31).

- **Будет ли данный компьютер сконфигурирован как NFS-клиент?** Если вы собираетесь монтировать файловые системы, расположенные на других машинах, нужно будет установить NFS-клиент. Для получения детальной информации о конфигурации сетевой файловой системы см. главу 31.

- **Требует ли данный компьютер специальных мер безопасности?** Если ваша машина не требует максимально высокого уровня защиты, выбирайте **No**. Позднее можно будет установить параметры настройки функции защиты при помощи редактирования конфигурационных файлов системы.

После выбора в этом варианте **No**, на экране появится другое окно, уведомляющее о службах, которые будут доступны в данном режиме. Политика высокого уровня защиты блокирует службы, подобные X-Window, в результате чего даже запуск X-сервера на локальном компьютере будет невозможен. Поэтому мы предлагаем использовать профиль среднего уровня защиты по умолчанию (the default medium security profile), пока сервер будет работать в среде, где уровень защиты является критическим.

Настройка консоли

Далее можно установить параметры консоли FreeBSD, которые определят установки клавиатуры, выбор экранных шрифтов, скринсейвера и т.д. Все это можно сделать из меню, которое показано на рис. 2.8.

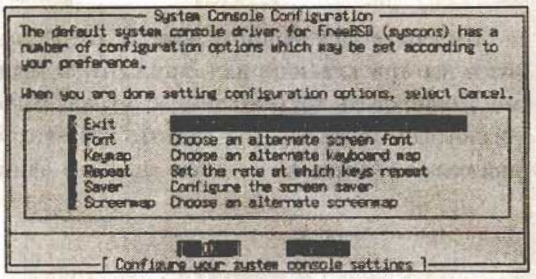


Рисунок 2.8
Установка параметров консоли.

Большая часть опций не требует пояснения. Выберите необходимые варианты и следуйте инструкциям, появляющимся на экране.

Установка часового пояса

Далее программа Sysinstall спросит о том, какой часовой пояс следует установить. Скорее всего, время в вашем часовом поясе не совпадает с временем по Гринвичу. Поэтому выберите **No**. Следующее меню запросит сведения о регионе, стране и другую информацию о местонахождении компьютера. Введите необходимые данные и нажмите **Yes**.

Совместимость с Linux

Следующий запрос будет выглядеть так: "Would you like to enable Linux binary compatibility?" (Установить бинарную совместимость с Linux?). После нажатия **Yes** программа Sysinstall установит mini Linux filesystem в **/usr**, разделяемые библиотеки и другие программы, необходимые для запуска программ, написанных под Linux. Это очень полезная опция, так что выберите **Yes**. Мы, например, написали ряд сценариев для этой книги, используя StarOffice для Linux.

ПРИМЕЧАНИЕ

Программы Linux в системе FreeBSD выполняются не через эмулятор, а поддерживаются на уровне ядра. Поэтому Linux-программы во FreeBSD работают также быстро, как и в Linux. Мало того, некоторые приложения работают в системе FreeBSD даже быстрее, чем в самой Linux!

Конфигурирование мыши

Во FreeBSD имеется демон, позволяющий выполнять операции типа "вырезать и вставить текст" с помощью мыши. Следующий запрос касается того, относится ли ваша мышь к породе USB-мышей. Если это так, нажмите **Yes**, и на экране появится меню конфигурирования мыши (см. рис. 2.9).

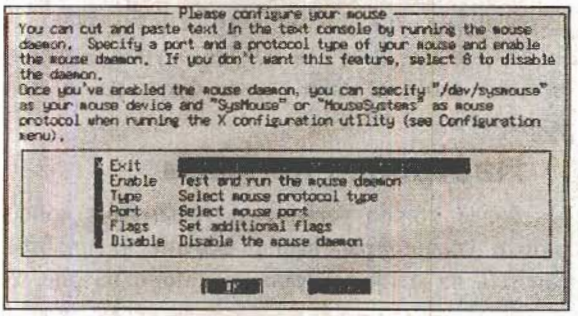


Рисунок 2.9
Конфигурирование мыши.

Выберите тип мыши и порт, к которому она подсоединена. В X-Widow используются все три клавиши для стандартной мыши. Если мышь двухкнопочная, необходимо эмулировать трехкнопочную мышь. Выберите в меню опцию **Flags** и введите ***-3*** в диалоговом окне. Другие важные параметры — это **-r high** (для более быстрого передвижения указателя) и **-r low** (для его замедления).

Конфигурация X-сервера

ПРИМЕЧАНИЕ

Этот раздел применим только в отношении FreeBSD версии 4.4, работающей на XFree86 3.3.6. Если версия системы — 5.0, необходимо использовать XFree86 4.x. В этом случае следуйте инструкциям в главе 34.

Если в компьютере установлена система X-Window, на экране появится запрос о конфигурации X-сервера. Выберите **Yes**, и появится меню, как на рис. 2.10.

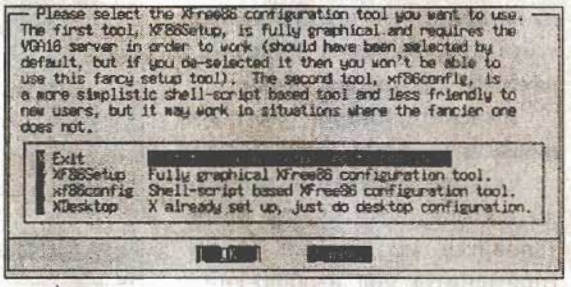


Рисунок 2.10
Выбор способа конфигурации X-сервера.

Если до этого момента X-Window не устанавливалась, выберите опцию XF86Setup для конфигурирования X-сервера.

На данном этапе экран может некоторое время оставаться пустым, а затем отобразится фон с буквой "X" в центре. Рано или поздно, так или иначе появится экран, изображенный на рис. 2.11. Это займет некоторое время, поэтому будьте терпеливы.

ПРИМЕЧАНИЕ

Избегайте движений мышью до тех пор, пока не завершится часть программы XF86Setup по установки мыши. В противном случае, программа могут возникнуть проблемы ее установкой.

В первую очередь нужно сконфигурировать работу мыши в X, чтобы можно было пользоваться ею при установке всего остального.

Рисунок 2.11
Главное меню программы установки XF86Setup.

Конфигурирование мыши

Итак, перед вами экран с системой помощи, показанный на рис. 2.12.

Ознакомившись с его содержанием, нажмите Enter. Появится экран, показанный на рис. 2.13. Для перемещения между опциями используйте клавишу табуляции. Выберите **Sysmouse** в качестве протокола мыши.

Выберите опцию **/dev/sysmouse** и нажмите Enter. Держите клавишу табуляции нажатой до тех пор, пока курсор не остановится на кнопке "Apply" (применить) под изображением мыши. Нажмите Enter, и через секунду мышь будет активирована. Проверьте, передвигается ли курсор по экрану, попробуйте, действуют ли кнопки мыши. Если возникли проблемы с мышью, загляните в приложение В. Если мышь двухкнопочная и необходимо эмулировать трехкнопочный вариант нажатием сразу двух кнопок, выберите опцию **Emulate3Buttons**.

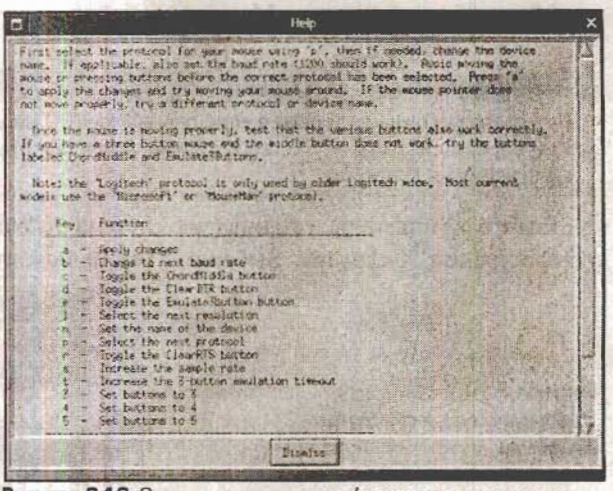


Рисунок 2.12 *Экран помощи в конфигурировании мыши.*

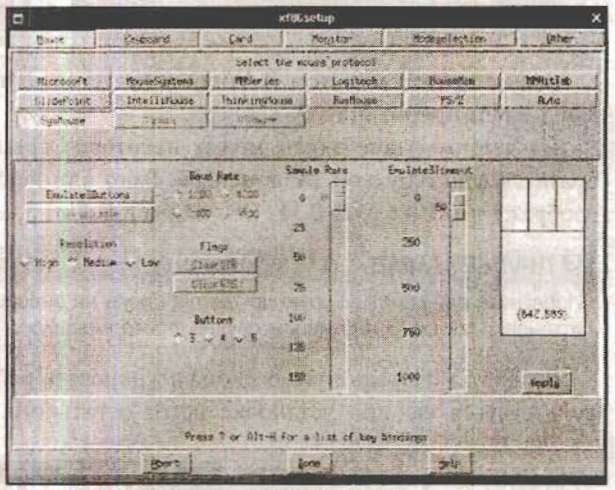


Рисунок 2.13 *Экран конфигурирования мыши.*

Конфигурирование клавиатуры

Кликните на кнопке Keyboard в меню, расположенном в верхней части экрана. Здесь выбирается модель и тип клавиатуры (см. рис. 2.14). Выбор по умолчанию — это "Generic 101-key PC". Однако многие пользуются клавиатурой со 104 клавишами (Enhanced Windows keyboard), поскольку на ней имеются добавочные клавиши. Для X-Window есть программы и утилиты, использующие эти добавочные клавиши. Итак, выберите вариант, соответствующий вашей клавиатуре.

На правой стороне экрана можно переназначить некоторые клавиши так, чтобы *они выполняли другие функции.*

Конфигурирование видеокарты

Далее кликните на закладке Card в верхней части экрана, после чего на экране появится меню, в котором программе XF86Setup следует указать тип вашей видеокарты (см. рис. 2.15).

Прокрутите весь список и найдите нужный тип видеокарты, после чего кликните на нем для выбора. Если нужной видеокарты в списке все-таки не окажется, не все потеряно. Попробуйте поискать использующийся в ней чипсет, вы сможете сконфигурировать видеокарту вручную, используя опцию "Detailed Setup" (см. главу 34, о сложных моментах конфигурирования X-Window). Выбрав видеокарту, кликните на вкладку "Monitor", чтобы выбрать параметры монитора.

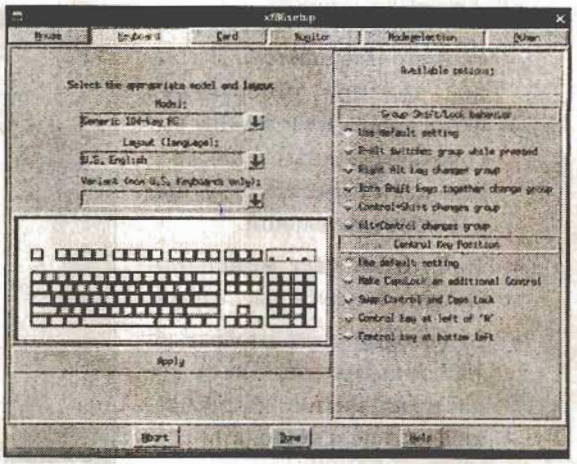


Рисунок 2.14 *Конфигурирование клавиатуры.*

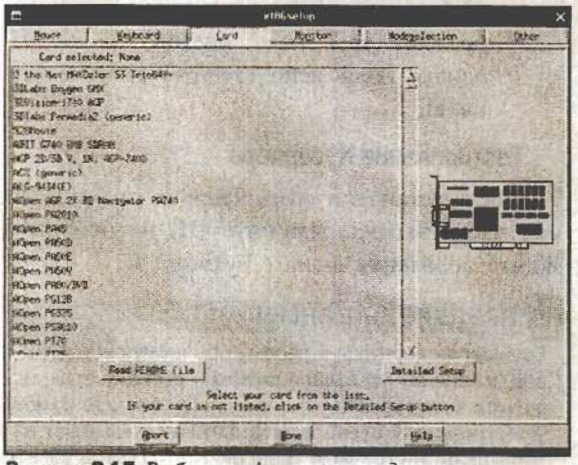


Рисунок 2.15 *Выбор конфигурации видеокарты.*

Конфигурирование параметров монитора

ВНИМАНИЕ!

С настройкой монитора следует быть особенно внимательным, поскольку неправильная установка его технических характеристик может повлечь за собой порчу оборудования. Не указывайте тип монитора, частота горизонтальной развертки которого выходит за пределы возможностей вашего монитора. В случае сомнений выбирайте более консервативный вариант.

Вы увидите новый экран, такой как на рис. 2.16.

Введите в соответствующие поля частоту вертикальной и горизонтальной развертки для вашего монитора. Данную информацию можно найти в инструкции по эксплуатации монитора. Можно также задать диапазон частот горизонтальной и вертикальной развертки.

Выбор видеорежима

X-Window работает наилучшим образом при разрешении не менее 1024×768. Если монитор 17-дюймовый, эти параметры будут в самый раз. Если 15-ти дюймовый — терпимо. Для 15-дюймового монитора можно уменьшить разрешение до 800×600. С 14-дюймовым монитором в X-Window работать тяжело, в любом случае разрешение должно быть не менее 800×600, в крайнем случае — 640×480. Если монитор 19- или 21-дюймовый, желательно выбрать разрешение не менее 1024×768. При желании можно выбрать несколько разрешений. В этом случае можно будет изменить разрешение прямо в X-Window (видеокарта должна поддерживать эти видеорежимы). В нижней части экрана выберите количество цветов, воспроизводимых в данном режиме:

- 8-битовый цвет поддерживает всего 256 цветов. Понятно, что при отображении фотографий или других высококачественных изображений цветопередача будет неудовлетворительной.
- 16-битовый цвет может отображать более 64 тыс. цветов.
- 24-битовый цвет отображает 16,7 млн. цветов. Такое количество цветов человеческий глаз не в состоянии различать.
- 32-битовый цвет также отображает 16,7 млн. цветов, однако здесь используется четыре байта на пиксель, а не 3, как у 24-битного цвета.

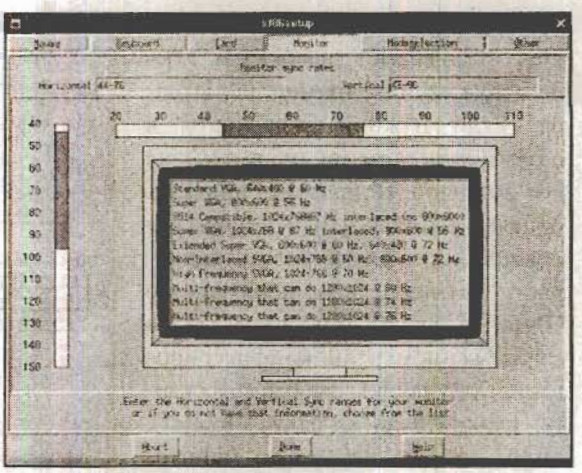


Рисунок 2.16 *Конфигурирование частот обновления монитора.*

Тестирование X-сервера

После установки разрешения и глубины цвета кликните на кнопке "done" в нижней части экрана. Нажмите **OK** на появившемся сообщении о запуске X-сервера. Ждите появления фона с буквой "X".

ПРЕДУПРЕЖДЕНИЕ

Если экран вернулся к своему обычному состоянию, но имеет какой-то искаженный и странный вид, и/или вы услышали высокий воющий звук, исходящий от монитора, НЕМЕДЛЕННО выключите монитор или одновременно нажмите комбинацию клавиш CTRL-ALT-BACKSPACE для уничтожения X-сервера. Оба сигнала указывают на то, что частоты обновления монитора были завышены, и строчный трансформатор готов взлететь на воздух.

Если монитор остался в затемненном состоянии или вошел в экономичный режим питания, или же произвольно выключился (контрольная лампа источника питания замигала, изменился цвет), это, возможно, означает, что выбранные частоты обновления превысили возможности монитора, и он отключился для предотвращения поломки. Нажмите комбинацию клавиш CTRL-ALT-BACKSPACE для уничтожения X-сервера и попытайтесь еще раз.

И вот наконец-то на экране появилось окно с надписью "Congratulations. You've got a running X server!" Если изображение на экране смещено от центра или имеет неправильный размер, воспользуйтесь **xvidtune** для корректировки изображения.

Использование xvidtune

При запуске **xvidtune** на экране появится "угрожающее" сообщение о непоправимых последствиях неправильного использования этой программы, которые выражаются в повреждении монитора и/или видеокарты. Скажите **OK**, и экран примет такой вид:

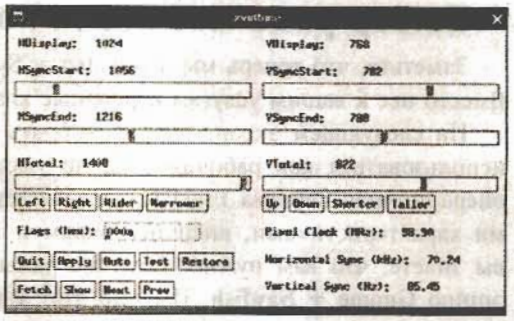


Рисунок 2.17
Работа программы xvidtune.

В данной программе используются следующие клавиши:

Таблица 2.6 Клавиши управления xvidtune.

Клавиша	*Действие*
Left, Right, Up и Down	Перемещение изображения на экране в соответствующих направлениях.
Wider (шире), Narrower (уже), Shorter (короче), Taller (выше)	Соответствующее изменение изображения на экране.
Quit	Выход из xvidtune.
Apply	Применение изменений к текущему видеорежиму.
Auto	В активном состоянии изменения вносятся незамедлительно, без нажатия клавиши "Apply". Это позволяет видеть результаты предпринимаемых действий в реальном времени.
Test	Проверка дисплея с учетом внесенных изменений. Начальный экран будет восстановлен через несколько секунд. При параллельной активизации **Auto** не действует.
Restore	Восстанавливает первоначальные параметры.
Fetch	Сохраняет текущие параметры X-сервера. Воспользуйтесь функцией **Restore** для восстановления параметров, активных на момент выбора **Fetch**.
Show	Отображает текущий модели из файла XF86Config в стандартный вывод данных (более детальная информация — в главе 34).
Next, Prev	Функция переключения на следующий или предыдущий видеорежим, соответственно (при условии, что выбрано несколько разрешений).

Если в программе **xvidtune** вносятся изменения частот вертикальной и горизонтальной развертки, убедитесь, что их значения не выходят за рамки возможностей вашего монитора.

После внесения изменений нажмите **Apply** для установки окончательных параметров, затем **Quit** для выхода из программы **xvidtune**.

Сохранение конфигурации и выход из программы XF86Setup

После выхода из **xvistune** кликните на опции Save Configuration and Exit. Ответьте **Yes** на вопрос о создании символической связи 'X' с сервером.

Выбор рабочего стола по умолчанию

Заметьте, что теперь мы вернулись к Sysinstall. Здесь нельзя пользоваться мышью. Вместо нее к вашим услугам курсорные клавиши, клавиши табуляции, пробел и Enter.

На следующем экране можно выбрать рабочий стол, который в X-Window будет использоваться как рабочий стол по умолчанию. В отличие от Microsoft Windows, операционная система FreeBSD предлагает широкий выбор рабочих столов с разными характеристиками, внешним видом и т.д. Если вы опытный пользователь Linux, вы знаете, что вам нужно. Если же раньше вы не работали с X-Window, выберите опцию **Gnome + Sawfish**. Именно этот вариант мы будем использовать в последующих разделах.

ПРЕДУПРЕЖДЕНИЕ

Используя **xf86cfg**, помните, что неправильно введенные параметры частот обновления могут повредить монитор. Более детально **xf86cfg** будет рассмотрен в главе 34.

Установка дополнительных пакетов программ

Ответьте **Yes** на вопрос о просмотре набора пакетов FreeBSD. Ознакомьтесь с набором, чтобы получить представление об имеющихся программах. Когда появится запрос о способе установки, выберите CD-ROM.

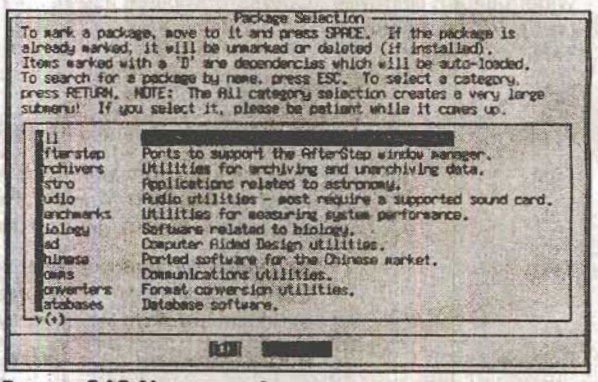


Рисунок 2.18 *Установка дополнительных пакетов.*

Внимательно рассмотрите систему установки пакетов (курсорные клавиши используются для перемещения между пакетами и категориями, пробел — для выбора или отмены выбора пакета, Enter — для возвращения к основному меню). В уже установленных пакетах программ после имени будет стоять буква **X**. Отмена выбора этих пакетов приведет к их удалению. Пакеты, для которых указана буква **D**, являются зависимостями, то есть они требуются для некоторых других пакетов. В нижней части экрана помещено краткое описание отмеченного пакета.

Следующие разделы содержат описание категорий. Как мы полагаем, пользователь просмотрит и установит предложенные нами пакеты.

Командные интерпретаторы

- **Bash-2.04** — мощный и простой в обращении командный интерпретатор с интерфейсом командной строки. Пользователи Linux знакомы с ним, поскольку практически во всех версиях Linux — это командный интерпретатор по умолчанию.

Gnome

Если вы последовали нашему совету и установили **Gnome + Sawfish** в качестве среды рабочего стола по умолчанию, то теперь можете установить некоторые пакеты

из этой категории, чтобы работа с Gnome стала продуктивнее (и/или забавнее). Ниже приведены некоторые пакеты программ, которые будут полезны для дальнейшей работы:

- **eog-0.5_1** — The Eye of Gnome Image Viewer. Позволяет просматривать графические файлы, понимает большинство известных форматов.
- **gaddr-1.1.4** — Простая адресная книга для Gnome.
- **gedit-0.9.3** — Графический текстовой редактор для Gnome, напоминает Notepad, но обладает большим набором функций.
- **glunarclock-0.11** — Графическая программа, отображающая на панели Gnome картинки с текущей фазой Луны. Применяется для отслеживания следующего перевоплощения пользователя в оборотня.
- **gno3dtet-1.6.0** — 3D-версия Тетриса для Gnome.
- **gnomeapplets-1.2.4** — Мелкие приложения для панели Gnome.
- **gnomefind-1.0** — Помогает отыскать потерянный файл, если пользователь забыл, куда он его положил.
- **gnomegames-1.2.0** — Пакет, содержащий несколько программ развлекательного характера.
- **gnomemc-4.5.51** — Gnome Multiple Commander. Программа управления файлами для Gnome, напоминает Windows Explorer.
- **gnomemedia-1.2.0** — Прикладные мультимедиа-задачи для Gnome (CD-плейер и т.д.).
- **gpaint-0.1.1** — Программа, напоминающая Paint в Windows.

Выбрав необходимые пакеты, убедитесь, что находитесь в основном меню установки пакетов (см. рис. 2.18). Воспользуйтесь клавишей табуляции для перехода к кнопке **Install**, нажмите Enter. **Sysinstall** приступит к установке выбранных вами пакетов.

Добавление пользователя

Затем появится запрос о добавлении в систему новых учетных записей пользователей. Выберите **Yes** для создания обычной учетной записи. Помните, что для пользователя **root** нет ограничений в системе, поэтому заходить в систему как **root** следует лишь для выполнения конкретных задач администрирования. Для повседневной работы используйте учетную запись обычного пользователя. Поверьте, это правило родилось на базе горького опыта, и начинающим системным администраторам не стоит им пренебрегать.

На экране, изображенном на рис. 2.19, выберите **User** для добавления нового пользователя.

На экране появится форма для нового пользователя, которую необходимо заполнить (см. рис. 2.20).

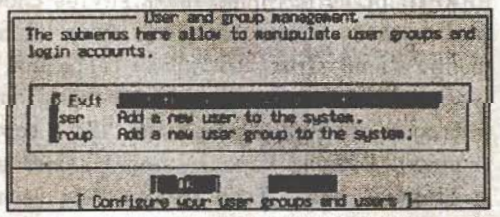


Рисунок 2.19 *Добавление новых пользователей и групп пользователей.*

Используйте клавишу табуляции для передвижения между полями. Значение каждого поля подробно рассмотрено в следующих разделах.

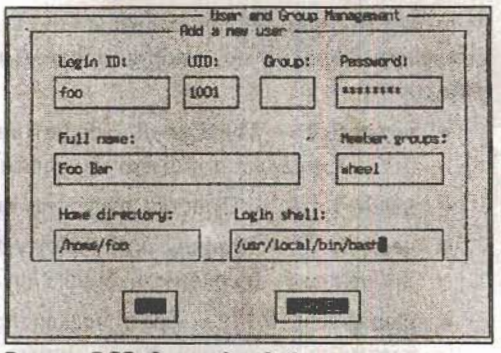


Рисунок 2.20 *Форма для добавления нового пользователя.*

Login ID:

Это имя, которое пользователь вводит для регистрации в системе. Оно должно включать не более восьми алфавитно-цифровых символов. Это могут быть инициалы, уменьшительные имена или сочетание имени и фамилии. Каждый пользователь должен иметь уникальное пользовательское имя. И самое главное — пользуйтесь буквами нижнего регистра!

UID:

Идентификатор пользователя — это просто уникальное число, назначенное системой для данного пользователя. Не заполняйте это поле: система подберет число самостоятельно.

Group ID:

Исходная группа, членом которой становится пользователь. Не заполняйте это поле, система сделает это автоматически.

Password:

Пароль, используемый для регистрации в системе. Здесь предоставляется несколько советов по выбору хорошего пароля, который непросто взломать.

- Пароль должен содержать не менее восьми алфавитно-цифровых символов.
- Используйте комбинацию букв верхнего и нижнего регистров и по крайней мере один символ вроде ! или $.
- Не используйте слова, имеющиеся в словарях. Программы, разработанные для взламывания паролей, перебирают словарь и легко подберут ваш пароль.
- Не используйте данные, которые легко ассоциируются с вами. Другими словами, имя ребенка, девушки, кота, год рождения и т.д. — это плохие пароли.

С другой стороны, нужно придумать что-то такое, что легко запомнить. Вот несколько приемов: объединить части двух-трех слов в одно (не забудьте о верхнем и нижнем регистрах!), вставить внутрь слова несколько цифр, использовать небуквенные символы вместо похожих букв. Пользователь также может использовать "замещение клавиатуры", составляя секретное слово, что-либо для пользователя значащее (например, так: первый верхний слева, второй верхний справа, первый нижний слева и второй нижний справа плюс 123 в верхнем регистре составляют вполне надежный пароль). Выбрав пароль, внимательно введите его, поскольку поступит запрос на повторный ввод. Если пароль был введен неправильно, зарегистрироваться будет невозможно (если пользователь ошибся случайно, см. приложение В относительно способов изменения пароля).

После окончательного выбора пароля, нигде его не записывайте, включая ящики рабочего стола. Не записывайте его на клеющиеся листки и не наклеивайте на монитор. Не наносите в виде татуировки на руку. Не записывайте пароль в блокнот. Никому никогда его не раскрывайте: ни сослуживцам, ни жене, ни детям.

Full name:

Можно ввести свое имя, а можно и не заполнять эту графу. Однако некоторые программы, такие как электронная почта, используют это поле. Так что лучше все-таки ввести свое имя.

Member groups:

В этом поле лучше ввести **wheel**, в результате чего пользователь станете членом группы **wheel**, что даст возможность при необходимости стать пользователем **root**.

Home directory:

Местоположение домашнего каталога на жестком диске. Не заполняйте это поле, система лучше знает.

Login shell:

Командный интерпретатор по умолчанию. Если установлен Bash, то в это поле нужно ввести **/usr/local/bin/bash** (не забудьте о косой черте!). Если пользователь предпочитает другой командный интерпретатор, его нужно указать в этом поле. Выберите с помощью клавиши табуляции кнопку **OK** и нажмите Enter. Учетная запись пользователя будет создана, и пользователь вернется к меню, показанному на рис. 2.20. Выберите опцию **Exit**.

Установка пароля root

Нажмите Enter на сообщении об установке пароля **root**. В нижней части экрана появится окно с приглашением на ввод нового пароля **root**:

```
Changing local password for root.
New password:
```

Введите новый пароль. Обратите внимание, что в процессе ввода пароль на экране не отображается. Это сделано, чтобы никто не мог подсмотреть его, заглядывая через плечо. Нажмите Enter. Система затребует вторичный ввод для подтверждения правильности. Если пароль был введен одинаково в обоих случаях, программа установки выведет запрос о переходе в конфигурационное меню. Выберите здесь **No**, в результате чего вернетесь в основное меню.

Выход из программы Sysinstall и перезагрузка системы

В меню выберите при помощи клавиши табуляции **Exit Install** и нажмите Enter. Система попросит удалить CD-диски и дискеты из соответствующих приводов. Теперь перезагрузите систему.

ПРИМЕЧАНИЕ

Может случиться, что пользователь нажал кнопку Eject (выдвижение) на CD-ROM, но ничего не произошло. Не паникуйте. Система FreeBSD не причинила вреда CD-приводу; она заблокировала его для того, что в процессе установки или в ходе любого другого процесса диск нельзя было случайно извлечь. В любом случае выберите **Yes** и подождите, пока система начнет перезагрузку.

Первая загрузка FreeBSD

Во FreeBSD по умолчанию используется трехэтапный процесс загрузки, в котором задействованы три программы, последовательно вызывающие друг друга (об этом более подробно см. главу 4). Каждая их этих программ базируется на функциях предыдущей программы и каждая следующая становится все более сложной.

Затем стартует ядро, которое определяет имеющиеся устройства и выполняет их инициализацию. При этом по экрану проходит ряд сообщений. После завершения процесса своей загрузки ядро передает управление пользовательскому процессу с именем **init**, который выполняет проверку дисков на возможность использования. Затем **init** запускает пользовательский процесс настройки ресурсов, выполняющий монтирование файловых систем, настройку сетевых адаптеров для работы в сети и вообще запуск всех процессов, обычно выполняемых в системе FreeBSD при загрузке.

После завершения **init** вы получите следующее сообщение системы:

```
FreeBSD/i386 (simba.samplenet.org) (ttyv0)
login:
```

Естественно, вместо (**simba.samplenet.org**) будет стоять имя вашего хоста и сети. Если сеть не установлена, здесь будет имя системы по умолчанию, вероятно **Amnesiac**. Зарегистрируйтесь как **root**.

На экране появится сообщение с информацией о системе:

```
Copyright  1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.

FreeBSD 4.4 RELEASE (GENERIC) #0: Mon Mar 26 00:47:09 CST 2001

#
```

Символ # — приглашение командной строки, это означает, что система ожидает ваших команд. Добро пожаловать во FreeBSD!

Отключение системы FreeBSD

Прежде чем выключить компьютер, нужно корректно завершить работу системы. Для полной остановки системы пользователь **root** должен выполнить команду **/sbin/shutdown -h now**. FreeBSD завершит работу и выдаст сообщение "System halted". Теперь можно выключить питание компьютера.

Важно понимать, что нельзя просто так взять и выключить машину, работающую под FreeBSD. Чтобы система в дальнейшем смогла нормально загрузиться, она должна корректно размонтировать все разделы и отметить их как чистые (clean). Простое выключение питания может привести к порче данных на жестком диске.

ГЛАВА 3

Сложные аспекты установки

- Резервное копирование
- Создание раздела для FreeBSD с помощью FIPS
- Работа с FIPS
- Возможные проблемы с альтернативной загрузкой и их решение
- Boot-менеджер FreeBSD
- Загрузка FreeBSD из LILO
- Альтернативные способы установки

Если предстоит поставить FreeBSD на машину, на которой уже установлена другая операционная система, необходима осторожность. Часто случается, что система FreeBSD устанавливается на рабочую станцию, на которой уже функционирует Windows. Данная глава покажет, каким образом две операционные системы смогут уживаться на одной машине. Будет также кратко рассмотрена установка FreeBSD на машину, работающую под Linux. Кроме того, мы вкратце коснемся сетевой файловой системы (NFS) и протокола передачи файлов (FTP).

Резервное копирование

Прежде чем что-либо предпринимать, необходимо сделать копии разделов, с которыми работают пользователи. Хотя создать пространство для размещения FreeBSD можно без разрушения хранящейся на машине информации, тем не менее следует рассчитывать на худшее: могут произойти ошибки, и система даст сбой. Сделайте копии всего, что требуется сохранить.

ПРЕДУПРЕЖДЕНИЕ

Пусть процедура резервного копирования станет для вас привычной. Жесткие диски — это механические устройства, которые не застрахованы от поломок. Быть может, если это случится, резервные копии, хранящие данные всех пользователей, спасут вашу фирму от краха.

К оборудованию, использующемуся для резервного копирования, относятся устройства записи компакт-дисков, приводы Zip или Jaz, лентопротяжные устройства, а если объем копируемых данных невелик — гибкие диски. (На самом деле дискеты не подходят для создания резервных копий по целому ряду причин. — Прим. ред.) Копировать операционную систему или установленные программы не требуется, поскольку их легче проинсталлировать заново. Главное — это файлы, включающие пользовательские данные, а также конфигурационные и рабочие файлы системы (например, база данных почтовых сообщений).

В этой книге процедуры резервного копирования не рассматриваются. Когда сделаны резервные копии всех необходимых файлов, потребуется освободить пространство на жестком диске для установки FreeBSD. Этого можно достичь несколькими способами:

- Просто удалите разделы и запустите процесс установки. При этом все существующие в системе данные будут потеряны.
- Используйте коммерческую программу разбивки диска на разделы, например, Partition Magic. Если в системе есть программа Partition Magic, однозначно воспользуйтесь ею. Для получения подробных инструкций обратитесь к руководству Partition Magic.
- Используйте утилиту FIPS. Это свободно распространяемая программа, позволяющая разбить существующий раздел, чтобы выделить пространство для размещения FreeBSD. Данная утилита включена в компакт-диск, поставляемый с этой книгой. Мы рассмотрим FIPS позднее.

Создание раздела для FreeBSD с помощью FIPS

FIPS — это программа, предназначенная для работы в режиме DOS. Она разбивает существующий раздел DOS на два раздела так, как вы укажете. Созданный программой новый раздел можно использовать для установки FreeBSD. Обратите внимание на то, что FIPS работает только с разделами типа DOS (FAT16 или FAT32). FIPS не может работать с другими типами разделов, как-то NTFS (Windows NT/2000), EXT2FS (Linux) или HPFS (OS/2).

При работе с FIPS следует учитывать некоторые ограничения, присущие данной утилите:

- FIPS не может разбить расширенный раздел DOS — только первичный. Если пользователь работает с Windows 95 OSR2 или Windows 98, для него это не проблема, поскольку есть только один первичный раздел, занимающий весь диск.
- На диске должно быть не более трех уже существующих разделов. Дело в том, что FIPS создает новый первичный раздел, а их общее число не может превышать четырех.

Запуск Scandisk и программы дефрагментации

Перед использованием FIPS необходимо запустить Scandisk DOS или Windows и устранить проблемы, связанные с диском. Затем запустите программу дефрагментации диска.

Утилите FIPS нужно нефрагментированное свободное пространство в конце раздела для создания нового раздела. Разбиение невозможно, если последние сектора диска содержат какие-либо данные. При дефрагментации все данные перемещаются в начало диска, не оставляя "дыр" в середине. Процесс дефрагментации может занять от нескольких минут до нескольких часов.

Где найти FIPS и создание загрузочной дискеты

FIPS имеется на CD-диске в каталоге TOOLS под именем **FIPS.EXE**. Потребуются также файлы с именем **RESTORRB.EXE** и **ERRORS.TXT**. Загрузить FIPS можно также с FTP-сервера FreeBSD **ftp.freebsd.org** или с одного из зеркальных сайтов. Обычно FIPS находится в каталоге **/pub/FreeBSD/tools/fips.exe**. Там же хранятся файлы **restorrb.exe** и **errors.txt**. В DOS или Windows нужно создать загрузочную дискету и скопировать на нее три вышеперечисленных файла. Ниже приведен пример создания загрузочной дискеты:

```
C:\> format a: /s
Insert new diskette in drive A:
and press ENTER when ready...

Checking existing disk format.
Verifying 1.44M
Format complete.
System transferred
```

```
Volume label (11 characters, ENTER for none)?

     1,457,664 bytes total disk space
        388,608 bytes used by system
     1,069,056 bytes available on disk

           512 bytes in each allocation unit.
           2,088 allocation units available on disk.

Volume Serial Number is is 031B-0831
Format another (Y/N)?n
C:\>d:
D:\>cd tools
D:\TOOLS>copy fips.exe a:\
      1 file(s) copied
d:\TOOLS>copy restorerb.exe a:\
      1 file(s) copied
D:\TOOLS>copy errors.txt a:\
      1 file(S) copied
D:\TOOLS>
```

Теперь загрузите систему с только что созданной дискеты. Когда система выполнит загрузку, введите в командной строке DOS команду **fips** для запуска утилиты FIPS.

Работа с FIPS

Когда FIPS запустится в первый раз, то автоматически выведет предупреждение о том, что программа не предназначена для работы в многозадачной среде. Нажмите любую клавишу для продолжения. Если в системе несколько дисков, FIPS уточнит, с каким из них предполагается работать. Выберите нужный диск. Затем программа выведет на экран таблицу разделов выбранного диска. Она будет выглядеть примерно так:

```
     |        |   Start        |      |          End     | Start  |Number of|
Part.|bootable|Head Cyl.  Sector|System|Head Cyl. Sector|  Sector|Sectors  |  MB
-----+--------+----------------+------+----------------+--------+---------+----
1    |    yes |   1     0     1|   06h|  12  983    32|      32|   409312| 199
2    |     no |   0     0     0|   00h|   0    0     0|       0|        0|   0
3    |     no |   0     0     0|   00h|   0    0     0|       0|        0|   0
4    |     no |   0     0     0|   00h|   0    0     0|       0|        0|   0

Checking root sector ... OK

Press any Key
```

Если на диске несколько разделов, программа спросит, какой из них будет разбиваться. Если на диске только один раздел, появится приглашение **Press any Key**. FIPS считывает загрузочный сектор и выводит информацию похожую на ту, что приведена ниже:

```
Bytes per sector: 512
Sectors per cluster: 8
Reserved sectors: 1
Number of FATs: 2
Number of rootdirectory entries: 512
Number of sectors (short): 0
```

```
Media descriptor byte: f8h
Sectors per FAT: 200
Sectors per track: 32
Drive heads: 13
Hidden sectors: 32
Number of sectors (long): 409312
Physical drive number: 80h
Signature: 29h
```

Затем появится запрос о номере стартового цилиндра нового раздела. Будет выведен объем нового и старого разделов. Курсорные клавиши позволяют увеличивать или уменьшать номер цилиндра, с которого начнется новый раздел. Кроме того, можно использовать курсорные клавиши up и down для более быстрого увеличения и уменьшения номера стартового цилиндра. По окончании нажмите клавишу Enter.

СОВЕТ

Запишите информацию о номере стартового цилиндра раздела, который создаете. В процессе установки FreeBSD это даст возможность убедиться, что система устанавливается на правильный раздел.

Далее FIPS покажет новую таблицу разделов. Вы сможете отредактировать ее или продолжить создание раздела. Если выбрать Continue, FIPS в последний раз уточнит, записать ли сделанные изменения в таблицу разделов. Скажите Yes и перезагрузите систему.

ПРЕДУПРЕЖДЕНИЕ

Очень важно, ничего не записывать на жесткий диск до перезагрузки системы. В противном случае диск может быть испорчен. Дело в том, что до перезагрузки DOS не подозревает, что таблица разделов изменена.

После перезагрузки системы необходимо снова запустить FIPS с опцией **-t**, это позволит убедиться, что разбиения на разделы выполнено правильно. Если все-таки были обнаружены ошибки, восстановите предыдущую таблицу разделов, запустив **RESTORRB.EXE** с последующей перезагрузкой.

ПРЕДУПРЕЖДЕНИЕ

Если в файловую систему на диске были внесены какие-либо изменения, будет невозможно использовать **RESTORRB.EXE** для восстановления старой таблицы разделов. Поэтому очень важно запустить **fips** с опции **-t** сразу же после перезагрузки системы.

Если вывод **fips -t** не сообщил об ошибке, извлеките загрузочную дискету из дисковода и перезагрузите систему. Затем запустите программу Scandisk на разделе, который подвергался изменению.

Возможные проблемы с альтернативной загрузкой и их решение

Если на жестком диске установлено две операционные системы, нужно выбрать, какая из них будет загружаться по умолчанию. Существует ряд проблем, связанных с альтернативной загрузкой.

Первой проблемой является то, что вся необходимая для загрузки FreeBSD информация должна находиться на первых 1024 цилиндрах жесткого диска. Это означает, что либо раздел **root** не должен превышать по объему 1024 цилиндра, либо нужно использовать отдельный загрузочный раздел. В этом случае вышеуказанного ограничения на размер раздела **root** нет.

Если для DOS или Windows необходимо больше пространства, чем предоставляют 1024 цилиндра, необходимо разбить раздел DOS или Windows на два логических раздела — C и D. Между разделами C и D необходимо поместить маленький раздел для загрузки с него FreeBSD. Этот раздел позднее, во время установки, будет использоваться как **/boot**. Для него достаточно 30 Мб.

Следующим важным моментом является необходимость установки DOS или Windows перед установкой FreeBSD. Windows (и DOS) предполагает, что она единственная операционная система на жестком диске, и при инсталляции переписывает главную загрузочную запись (master boot record). Если сначала установить FreeBSD, то последующая установка DOS или Windows может повредить загрузчик FreeBSD, и загрузить систему будет невозможно.

Альтернативная загрузка FreeBSD и DOS, Windows 95/98/Me

Система FreeBSD поставляется с программой управления загрузкой (boot manager), позволяющей при запуске системы выбрать ту или иную операционную систему. Допустим, что DOS, Windows 95, Windows 98 или Windows Me уже установлены. Тогда в процессе установки FreeBSD будет выведено меню, позволяющее установить загрузчик и добавить уже имеющиеся операционные системы в загрузочное меню.

Альтернативная загрузка FreeBSD и Linux

Если кроме FreeBSD на машине будет установлена ОС Linux, которая будет загружаться из загрузчика FreeBSD, установите LILO — в загрузочный раздел Linux, а не в MBR. В результате можно будет загрузить Linux из Boot-менеджера FreeBSD.

Boot-менеджер FreeBSD

Установить boot-менеджер FreeBSD можно в процессе инсталляции. После установки можно сконфигурировать загрузчик с помощью программы **boot0cfg**. Программой **boot0cfg** можно управлять из командной строки. Существует несколько опций, которые вас наверняка заинтересуют. **boot0cfg -B** устанавливает загрузчик в MBR. Это единственный способ восстановить boot-менеджер FreeBSD в случае его повреждения. Если необходимо внести изменения в конфигурацию boot-менеджера, следует установить его заново, используя соответствующую команду. Приведенный ниже список опций (см. табл. 3.1) поможет внести необходимые изменения в конфигурацию boot-менеджера:

Таблица 3.1 Конфигурационные опции boot-менеджера

Опция	*Описание*
-v	**boot0cfg** выводит расширенную информацию об указанном слайсе.
-b image	Здесь **image** — имя используемого загрузочного образа. По умолчанию — это **/boot/boot0**.
-d drive	Здесь **drive** является номером привода, используемым BIOS для обращения к диску. Обычно это **0x80** для первого привода и **0x81** для второго и т.д.
-f file	Здесь **file** — имя файла, в котором хранится копия исходной MBR на случай возникновения проблем. Если файл уже существует, он будет перезаписан.

Опция **-o** также поддерживается и содержит список опций, отделенных друг от друга запятой. Приведем некоторые из них:

Таблица 3.2 Конфигурационные опции boot-менеджера

Опция	*Описание*
packet	Если BIOS поддерживает данную опцию, он даст **boot0cfg** команду использовать расширения **int 0x13** вместо CHS для работы с дисками. Это позволяет обойти ограничение в 1024 цилиндра, описанное выше. Однако в случае, если ПК BIOS не поддерживает данную опцию, при следующей перезагрузке система может зависнуть.
noupdate	Boot-менеджера может делать записи в MBR по умолчанию и модернизировать его. Это может создавать проблему, если у вас установлено аппаратное антивирусное средство, блокирующее запись в MBR. **noupdate** не позволяет boot-менеджеру производить записи в MBR.

boot0cfg также поддерживает опцию **-s**, где **n** — это число от 1 до 5, указывающее раздел для загрузки по умолчанию. Имеется также опция **-t n**; здесь **n** обозначает время ожидания в ticks перед загрузкой операционной системы по умолчанию (в одной секунде примерно 18,2 ticks).

Загрузка FreeBSD из LILO

Если вы уже работали с Linux и хотите запустить FreeBSD, используя Linux LILO, сделать это очень просто. Отредактируйте файл **/etc/lilo.conf** в Linux и добавьте такие строки:

```
other=/dev/hda2
      table=/dev/hda
      label=FreeBSD
```

Возможно, потребуется изменить и другие строчки, чтобы указать другие устройства.

После изменения конфигурационного файла необходимо установить LILO заново. В качестве пользователя **root** введите команду **lilo**.

Альтернативные способы установки

Если по каким-то причинам установка FreeBSD с компакт-диска невозможна, есть и другие методы. Это, например, установка с FTP- или NFS-сервера.

Установка FreeBSD по FTP

Можно установить систему FreeBSD непосредственно с FTP-сервера. Однако в этом случае предполагается, что в вашем распоряжении постоянное и быстрое соединение с Internet. Выполнение установки через модем займет очень много времени, и нет гарантий, что соединение не оборвется. Обязательно уточните, есть ли возможность зарегистрироваться на FTP-сервере в качестве пользователя **anonymous**. Это обычный способ регистрации на общедоступных FTP-серверах. Если FreeBSD будет устанавливаться с сайтов FreeBSD, то такая регистрация там возможна. При установке системы FreeBSD с FTP-сервера, не предоставляющего анонимный вход в сеть (например, с FTP-сервера в локальной сети), необходимо предварительно сконфигурировать учетную запись пользователя.

Конфигурирование пользователей

В меню **sysinstall** (см. рис.3.1) выберите Options и нажмите Enter, после чего экран примет вид, показанный на рис. 3.2.

Рисунок 3.1
Основное меню sysinstall системы FreeBSD.

Рисунок 3.2
Меню Options.

Выберите опцию FTP Username и нажмите пробел. В диалоговом окне введите *имя, которое будете использовать при регистрации на FTP-сервере*, нажмите Enter. Затем появится запрос на ввод пароля. Введите нужный пароль и нажмите Enter. Когда закончите, нажмите Q для выхода, после чего вы вернетесь в основное меню **sysinstall**.

Выбор способа установки FTP

Далее следуйте инструкциям главы 2, пока не появится экран с вопросом о выборе способа установки (Choose Installation Media). Затем последует запрос на выбор сайта (см. рис. 3.3).

Если установку намечено производить с одного из зеркальных сайтов FreeBSD, выберите сайт из списка. Появившееся диалоговое окно (см. рис. 3.4) выдаст приглашение на ввод имени FTP-сервера и пути к дистрибутивным файлам FreeBSD. Рисунок 3.4 показывает пример FTP-сайта с именем **lion**, расположенного в сети **samplenet.org**, а файлы FreeBSD размещены в каталоге **/FreeBSD.**

Теперь нужно выполнить конфигурирование сети (см. рис. 3.5).

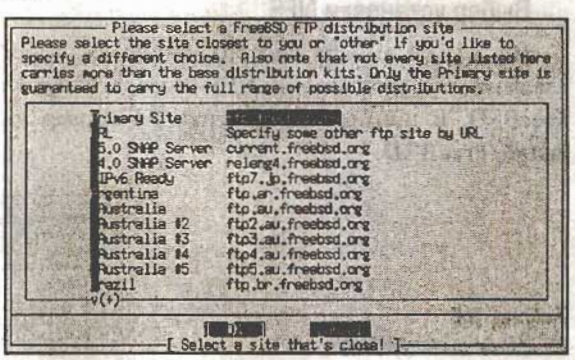


Рисунок 3.3
Выбор FTP-сервера, с которого будет выполняться установка FreeBSD.

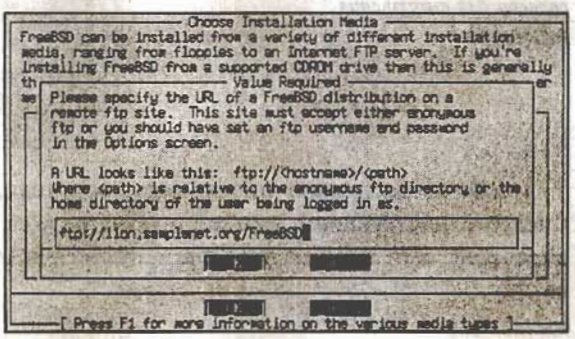


Рисунок 3.4
Конфигурирование FTP-сервера .

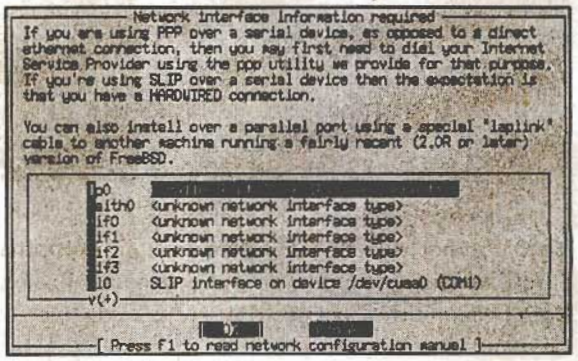


Рисунок 3.5
Конфигурирование сети.

Этот процесс выполняйте согласно инструкциям главы 2. По окончании конфигурирования сети установка будет протекать аналогично тому, как описано в главе 2. Когда копирование файлов будет закончено, двигайтесь дальше, руководствуясь указаниями главы 2. Можете пропустить раздел по конфигурированию сети, поскольку уже сделали это раньше.

Установка по NFS

NFS расшифровывается как *сетевая система файлов* (Network File System). Систему можно установить через сетевую файловую систему, предполагая, что в сети есть NFS-сервер и на нем имеется дистрибутив FreeBSD.

Выбор установки NFS

На экране выбора способа установки выберите NFS. Затем будет выведено приглашение на ввод имени NFS-сервера и пути к каталогу, содержащему дистрибутив FreeBSD. В примере на рисунке 3.6 сервер называется **lion**, а каталог установки — **install/FreeBSD**.

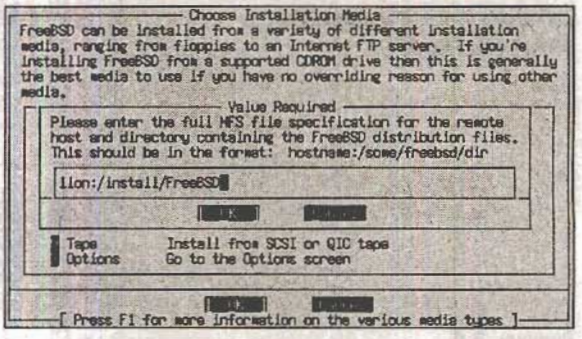
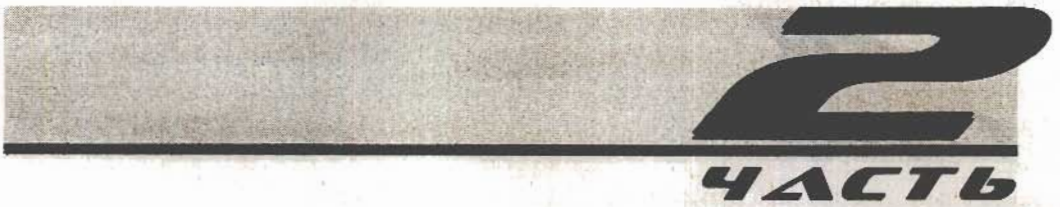


Рисунок 3.6
Конфигурирование NFS-сервера для выполнения установки.

После ввода данной информации необходимо сконфигурировать сеть (рис. 3.5). Следуйте инструкциям соответствующего раздела главы 2. По окончании конфигурирования сети процесс установки нужно продолжить точно так, как описано в главе 2.



Использование FreeBSD

- Первый сеанс работы во FreeBSD
- Работа в среде Gnome
- Настройка среды Gnome
- Работа с приложениями
- Работа с оболочкой

4 ГЛАВА

Первый сеанс работы во FreeBSD

- Процесс запуска FreeBSD
- BIOS
- Начальная загрузка
- Ядро
- Процесс init
- Вход во FreeBSD
- Запуск системы X-Window
- Введение в среду Gnome
- Останов системы X-Window
- Выход из FreeBSD
- Закрытие FreeBSD

В настоящей главе освещаются вопросы входа в операционную систему FreeBSD и запуска системы X-Window. Она также познакомит вас со средой рабочего стола, которая называется Gnome.

Процесс запуска FreeBSD

Когда вы впервые запускаете FreeBSD, на экране появляется поток сообщений ядра системы и различных процессов, которые начинают выполняться. В следующих разделах мы кратко рассмотрим процедуру запуска операционной системы FreeBSD.

BIOS

ПРИМЕЧАНИЕ

Далее мы будем говорить об операционной системе FreeBSD, работающей на компьютерах с процессорами x86 компании Intel. В вычислительных системах Alpha процесс запуска FreeBSD в общих чертах похож, но в деталях немного отличается.

BIOS (*Basic Input/Output Services — Базовые службы ввода/вывода*) представляет собой небольшое по объему программное обеспечение, которое зашито в микросхеме ПЗУ на системной плате компьютера. Кроме всего прочего, задача BIOS заключается в тестировании аппаратного обеспечения компьютера при его включении и запуске загрузчика операционной системы.

POST

При включении компьютера на нем автоматически запускается тест, который называется *POST* (*Power On Self Test — Самотестирование при включении питания*). На этом этапе BIOS также проверяет работу аппаратного обеспечения компьютера, находит объем оперативной памяти и устройства PnP, чтобы определить, какие ресурсы они могут использовать.

Начальная загрузка

Затем BIOS ищет в системе устройство, с которого можно произвести начальную загрузку. Какие устройства будут проверяться и в каком порядке — зависит от настроек BIOS. Обычно загрузка выполняется с первого жесткого диска системы, но им может также быть и дискета, и CD-диск. Возможна также загрузка операционной системы с сервера через сеть. Загрузка осуществляется с первого обнаруженного устройства загрузки. Когда устройство, с которого можно загрузиться обнаружено, BIOS начинает процедуру *начальной загрузки*. Начальная загрузка — это многоэтапный процесс, который начинается с этапа *boot0*.

Этап boot0

BIOS считывает содержимое сектора 0 жесткого диска, с которого выполняется начальная загрузка. Сектор 0 называют также *главной загрузочной записью* (*Master Boot Record*, или *MBR*). Программа, расположенная в MBR, имеет длину всего 512 байтов. Она содержит информацию о диске, позволяющую вывести на экран меню разделов данного диска, с которых можно выполнить начальную загрузку. Вот пример информации программы boot0:

```
F1 DOS
F2  FreeBSD
F3  Drive  1

Default:  F2
```

С помощью функциональных клавиш можно выбрать раздел, который требуется загрузить (и, следовательно, операционную систему). После этого boot0 загружает содержимое *загрузочного сектора* выбранного раздела. Если на жестком диске находится только операционная система FreeBSD, то меню на экран не выводится и загрузочный сектор загружается автоматически. Это начало следующего этапа начальной загрузки, который называется *boot1*.

Этап boot1

Программа boot1 находится в первом секторе загружаемого раздела. Как и программа boot0, она имеет в длину 512 байтов. Ее задача — найти и загрузить программу *boot2*.

Этап boot2

Программа boot2 способна загружать с жесткого диска реальные файлы. Обычно это программа, которая называется *загрузчик*. Программа-загрузчик выполняет следующий этап начальной загрузки.

Этап boot3

Обычно программа-загрузчик хранится в каталоге **/boot/loader**. Она позволяет изменять различные параметры запуска операционной системы FreeBSD, в частности, выбирать ядро, которое должно быть загружено. Работа с ядром будет подробно рассматриваться в главе 11. Итак, загрузчик находит и загружает ядро операционной системы и передает ему управление. На этом начальная загрузка заканчивается.

Ядро

Стандартное ядро обычно находится в каталоге **/boot/kernel** и запускается загрузчиком. Оно управляет всеми обращениями к аппаратным ресурсам со стороны программ и пользователей. Сообщения, видимые на экране при запуске FreeBSD, представляют собой сообщения ядра, которые информируют о том, что ядро находит и инициализирует аппаратные средства компьютера. Большая часть этих сообщений пробегает по экрану слишком быстро. Но после регистрации и входа в систему можно воспользоваться командой **dmesg | more** и просмотреть эти сообщения; переход к следующей странице сообщений осуществляется нажатием клавиши "пробел". Ниже в качестве примера приводятся некоторые сообщения ядра системы.

```
Copyright 1992-2001 The FreeBSD Project.
Copyright 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
     The Regents of the University of California. All rights reserved.
FreeBSD 5.0-CURRENT #0: Sun Apr 8 15:17:26 CDT 2001
     root@simba.samplenet.org:/usr/obj/usr/src/sys/SIMBA
```

Первые три строки в объяснениях не нуждаются. Это просто информация об авторских правах. В четвертой строке дается название операционной системы и версия ядра. В ней также указывается дата и время сборки ядра.

Первая часть пятой строки содержит имя и локальный электронный адрес пользователя, собравшего ядро. В данном случае ядро было собрано пользователем **root** и сделано это было на той же вычислительной системе, которая использовалась для написания этой книги. Вторая часть пятой строки содержит информацию о том, где находится каталог, в котором было собрано ядро.

```
CPU: AMD-K6(tm) 3D+ Processor (400.91-MHz 586-class CPU)
    Origin = AuthenticAMD    Id = 0x591    Stepping = 1
     Features=0x8021bf<FPU,VME,DE,PSE,TSC,MSR,MCE,CX8,PGE,MMX>
     AMD  Features=0x80000800<SYSCALL,3DNow!>
```

Данные строки описывают тип процессора и содержат некоторую информацию о его возможностях.

```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model Generic PS/2 mouse, device ID 0
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq
↪2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5 drive> on fdc0 drive 0
```

Ядро обнаружило мышь типа PS/2. Мышь является устройством с именем **psm0**. Также были обнаружены контроллер накопителей на гибких дисках и накопитель на гибких дисках. Имя контроллера — **fdc0**, а имя самого накопителя — **fd0**.

```
sbc0: <Creative SB AWE64> at port 0x220-0x22f,0x330-0x331,0x388-↪0x38b
irq 5 drq 1,5 on isa0
pcm1: <SB16 DSP 4.16> on sbc0
```

Ядро нашло звуковую плату SoundBlaster AWE 64 и выдала информацию об используемых этой платой ресурсах. Обратите внимание, что в данном случае **sbc0** — это имя группы устройств, связанных с данной звуковой платой.

```
unknown: <PNP0303> can't assign resources
unknown: <PNP0f13> can't assign resources
unknown: <PNP0501> can't assign resources
unknown: <PNP0700> can't assign resources
unknown: <PNP0401> can't assign resources
unknown: <PNP0501> can't assign resources
```

Некоторые из этих сообщений вы можете увидеть на экране своего компьютера во время запуска FreeBSD. Они означают, что ядро обнаружило в системе некоторые устройства Plug and Play, но они ему неизвестны, поэтому оно не может выделить им ресурсы. Наличие этих сообщений не влияет на стабильность системы.

```
ad0: 19473MB <Maxtor 92049U6> [39566/16/63] at ata0-master UDMA33
ad1: 14664MB <IBM-DJNA-351520> [29795/16/63] at ata0-slave UDMA33
```

Ядро нашло накопители на жестких дисках и выдала информацию об имени накопителя, его объеме, производителе, модели, геометрии и о том, к какому контроллеру подключен накопитель. Она также сообщает, какой режим доступа использует накопитель. В данном случае оба накопителя используют Ultra DMA 33 (см. главу 19).

```
Mounting rootfrom ufs:/dev/ad0s1a
```

Это сообщение означает, что программа ядра смонтировала корневую файловую систему. После того как корневая файловая система будет смонтирована, ядро передает управление процессу, который называется **init**. На этапе работы процесса **init** выдаются сообщения светло-серого цвета. Это позволяет отличать сообщения ядра от

других сообщений. Сообщения ядра — ярко-белого цвета; другие сообщения — светло-серого.

Процесс init

Когда операционная система FreeBSD закрывается надлежащим образом, она с целью проверки того, что все данные записаны, выполняет для каждого диска программу **sync**, демонтирует файловые системы и устанавливает для этих файловых систем флажок *clean* (*чисто*). Если операционная система FreeBSD будет закрыта некорректно, то флажок clean установлен не будет.

Проверка целостности файловой системы

Одно из первых действий, выполняемых процессом **init**, — это проверка того, установлен ли флажок clean. Если флажок установлен, процесс **init** монтирует файловую систему для использования. Если флажок не установлен, процесс **init** запускает программу **fsck**; она проверяет, не повреждена ли файловая система, и исправляет все ошибки, которые в состоянии исправить. Программа **fsck** подобна программе Scandisk из операционной системы Windows и данный процесс напоминает процедуру **Your system was not properly shut down** (Ваша система не была закрыта надлежащим образом) в Windows. Если программа **fsck** обнаруживает ошибку, которую не может исправить, она переводит систему в однопользовательский режим, чтобы это мог сделать системный администратор.

Предположим, что флажок clean был установлен. В этом случае процесс **init** приступает к монтированию всех перечисленных в файле **/etc/fstab** файловых систем, для которых установлен флажок **mount at boot**.

СОВЕТ

При необходимости программу **fsck** можно запускать вручную. Более подробную информацию о программе **fsck** и ее параметрах можно найти в главе 9.

Сценарии конфигурации системы

После того, как файловые системы будут смонтированы, процесс **init** считывает *сценарии конфигурации* системы, или *сценарии управления выполнением* (*run control scripts, rc scripts*), которые находятся в каталогах **/etc** и **/etc/defaults**. Кроме того, **init** проверяет каталог **/user/local/etc/rc.d** на наличие дополнительных сценариев, которые должны запускаться процессом **init** автоматически при запуске операционной системы (например, сценарии запуска Web-серверов, серверов баз данных и др.). Эта часть процесса загрузки аналогична обработке файлов **config.sys**, **autoexec.bat**, **system.ini** и тех частей системного реестра, которые связаны с параметрами запуска операционной системы Windows.

ПРИМЕЧАНИЕ

Следует добавить, что процесс **init** проверяет также наличие файла **/etc/rc.local** и считывает его содержимое; он может использоваться для запуска таких программ, как Web-серверы. Однако этот файл не рекомендуется использовать, и в будущих версиях FreeBSD он может не поддерживаться. Поэтому все сценарии запуска рекомендуется помещать в каталог **/usr/local/etc/rc.d**, а не в файл **/etc/rc.local**.

РЕЖИМЫ ИСПОЛНЕНИЯ В BSD И В SYS V

Если у вас за плечами опыт работы в SYS V UNIX, то последний раздел может вызвать у вас легкое недоумение. Поэтому даем пояснение. В процессе **init** операционной системы BSD отсутствует такое понятие, как уровни исполнения. Практически, вы имеете два режима — однопользовательский и многопользовательский с сетевой поддержкой. В отличие от уровней исполнения операционной системы SYS V, в BSD многопользовательский режим без сетевой поддержки отсутствует. И файл **inittab**, который имеется в SYS V, в BSD также отсутствует. Кроме того, параметры запуска задаются, как правило, одним файлом — **rc.conf**.

Программы getty и Login

После считывания сценариев процесс **init** запускает программу консоли (и программы нескольких виртуальных терминалов). Как правило, это программа **getty**. Другая программа, часто используемая вместо **getty**, — это **xdm**, она сразу же после загрузки системы запускает графический сеанс X-Window. В файле **/etc/ttys** задается, какая именно программа будет запускаться. Будем считать, что в нашем случае это **getty**.

Программа **getty** инициализирует терминал и устанавливает различные параметры, относящиеся к типу терминала и безопасности системы. Напомним еще раз, что эти параметры и их значения задаются в файле **/etc/ttys**. Затем **getty** запускает программу входа в систему — **login**, которая подтверждает правильность регистрационного имени пользователя и пароля.

Вход во FreeBSD

Когда все процессы запуска операционной системы завершатся, вы увидите на экране следующую информацию:

```
FreeBSD/i386  (amnesiac)  (ttyp0)

login:
```

Если при установке операционной системы вы не сконфигурировали никакой сетевой среды, то по умолчанию для вашей системы будет использоваться имя хоста **amnesiac**. Чуть позже мы расскажем, как его изменить.

Введите регистрационное имя обычного пользователя, которое вы назначили себе во время установки. Нажмите Enter и при появлении приглашения введите пароль. Обратите внимание, что пароль не будет отображаться на экране.

После того как вы введете пароль, программа входа в систему сравнивает его с паролем, хранимым в базе данных. Если сравнение пройдет успешно, вы увидите на экране что-нибудь вроде:

```
Last login: Tue Apr 10 15:19:17 on ttyp0
Copyright  1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights
➥reserved.
FreeBSD 5.0-RELEASE (GENERIC) #0: Sun Apr  8 15:17:26 CDT 2001

bash$
```

Если это ваш первый вход в систему в качестве обычного пользователя, вы, по-видимому, не увидите первую строку с информацией о последнем входе в систему

(Last Login). Однако при последующих подключениях к системе эта строка будет присутствовать всегда.

ПРЕДУПРЕЖДЕНИЕ

Обязательно обращайте внимание на информацию о последнем входе в систему. Если, например, данная строка информирует вас о том, что время последнего подключения — это Sat Sept 15 14:05:29, а вы в этот день отдыхали за городом и наверняка не работали, это значит, что кто-то посторонний хозяйничал в системе, пользуясь вашими учетными данными. В этом случае НЕМЕДЛЕННО меняйте пароль (воспользуйтесь командой **passwd**) и поставьте в известность системного администратора о взломе защиты. (А может, **root** — это вы?)

Остальная информация — это информацией об авторских правах и о ядре (когда оно было собрано). Последняя строка — это приглашение командного интерпретатора. Операционная система FreeBSD ждет от вас ввода команды.

ПРЕДУПРЕЖДЕНИЕ

Обратите внимание на различие между нынешним приглашением и тем, что было в прошлый раз, когда вы регистрировались и входили в систему как пользователь **root**. У пользователя **root** приглашением является символ фунта — #. У обычного пользователя приглашением, как правило, является либо **$** (в стиле командного интерпретатора Борна), либо **%** или **>** (в стиле командного интерпретатора C). Когда вы входите в систему как пользователь **root**, решеточка (#) постоянно напоминает вам о необходимости быть внимательным при вводе потенциально опасных команд.

Если вы неправильно введете регистрационное имя или пароль, операционная система выдает в ответ следующее сообщение:

```
Login incorrect
login:
```

В этом случае попытайтесь снова ввести свои регистрационные данные.

СОВЕТ

Если три раза подряд неправильно ввести регистрационное имя или пароль, то создается впечатление, что система зависла. На самом деле это не так. Эта задержка — средством защиты, помогающее резко снизить эффективность программ разгадывания паролей, пытающихся взломать защиту. С каждой последующей неудачной попыткой задержка может увеличиваться в определенной прогрессии. Так что подождите несколько секунд и приглашение **login:** появится снова.

Запуск системы X-Window

Если при установке операционной системы вы сконфигурировали X-Window, то в командной строке теперь можно ввести **startx**, запуская тем самым систему X-Window.

Введение в среду Gnome

Если при установке в качестве среды рабочего стола вы выбрали среду Gnome, используемую по умолчанию, и установили соответствующие пакеты (см. главу 2), то на экране должно появиться изображение, похожее на то, что показано на рис. 4.1.

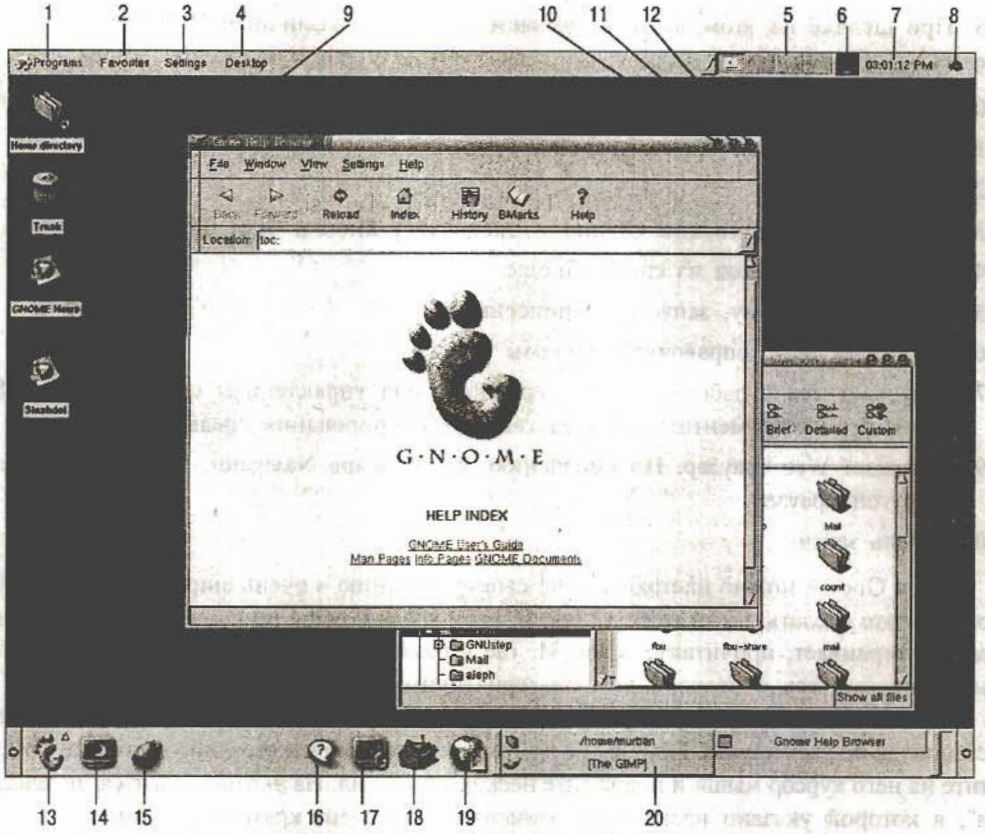


Рисунок 4.1 *Среда рабочего стола Gnome.*

Не правда ли, напоминает Windows или Macintosh? Но есть также и некоторые различия. Вот назначение всех элементов, показанных на рис. 4.1:

1. Здесь можно запускать программы, добавленные в меню Gnome. Это меню аналогично меню Start — Programs в Windows.
2. Можно создать список избранных URL-адресов Web-сайтов, сайтов FTP, программ и т.д.
3. Есть возможность конфигурировать среду Gnome.
4. На рабочем столе можно размещать пиктограммы или выходить из этой среды.
5. Можно переключаться между четырьмя виртуальными рабочими столами. Направленная вниз стрелка с левой стороны открывает список окон.
6. При поступлении электронной почты, здесь появляется трехмерный вращающийся конверт. Чтобы сконфигурировать такую возможность, щелкните правой кнопкой мыши на этом месте.
7. На часах можно щелкнуть правой кнопкой и изменить формат отображения времени. Если установлен Gnome-PIM, то из меню часов, открываемого правой кнопкой, можно также обратиться к планировщику событий.

8. При щелчке на этом месте открывается доступ к различным Web-ресурсам.
9. Открывает оконное меню.
10. Сворачивает окно в пиктограмму.
11. Разворачивает или восстанавливает окно.
12. Закрывает окно.
13. Пиктограмма с оттиском Gnome — эквивалент кнопки Start (Пуск) в Windows.
14. Осуществляет выход из среды Gnome.
15. Блокирует систему, запуская скринсейвер.
16. Открывает окно справочной системы Gnome.
17. Запускает сеанс работы в окне терминала под управлением среды Gnome. 18. Запускает инструментальное средство конфигурирования среды Gnome.
19. Запускает Web-браузер. По умолчанию это Netscape Navigator. Но можно задать и другой браузер.
20. Панель задач.

Среду Gnome можно настраивать по своему желанию в очень широких пределах. О том, как это сделать, вы узнаете в главе 6. Если среда Gnome по каким-либо причинам вас не устраивает, прочитайте главу 34, где в разделе "Как заменить менеджер окон" вы познакомитесь с несколькими альтернативными программами.

Обратите внимание, что для большинства элементов среды Gnome параметр ToolTips установлен по умолчанию. Если вы не помните назначение какого-либо элемента, поместите на него курсор мыши и подождите несколько секунд. На экране появится "подсказка", в которой указано назначение элемента и дано его краткое описание.

Останов системы X-Window

Существует три способа закрыть среду Gnome и вернуться в командную строку:

- На панели Gnome в нижней части экрана щелкните на пиктограмме, на которой изображен экран монитора с изображением луны (пора бай-бай!).
- Щелкните на меню **Desktop** в верхней части экрана и выберите команду **Log Out**.
- На панели Gnome щелкните на символе Gnome (отпечаток лапы) и выберите команду **Log Out**.

В любом случае на экране появится диалоговое окно с вопросом: действительно ли вы хотите выйти из среды. В этом диалоговом окне будет также флажок, который называется **Save current setup** (Сохранить текущую настройку). Если установить этот флажок, то Gnome запомнит все выполнявшиеся апплеты панели, все открытые окна менеджера файлов и все открытые окна терминалов. Когда вы в следующий раз запустите среду Gnome, она будет иметь ту же самую настройку.

Выход из FreeBSD

Если к вашему компьютеру могут получить доступ другие люди, а вы собираетесь отлучиться (пусть даже на несколько минут), нужно выйти из системы. Это предот-

вращает доступ к ней других пользователей под вашим именем. Чтобы выйти из FreeBSD, введите **exit** в командной строке. На экране вновь появится приглашение на ввод регистрационного имени, которое вы видели при загрузке FreeBSD.

СОВЕТ

Перед выходом из системы неплохо было бы очистить экран. Для этого введите в командной строке **clear**. После этого выходите из системы, как было описано выше.

Закрытие FreeBSD

Перед выключением питания очень важно всегда правильно закрывать операционную систему FreeBSD (как и любую другую операционную систему UNIX). Если этого не сделать, может произойти серьезное повреждение файловой системы.

Команда shutdown

Правильное закрытие системы осуществляется с помощью команды **shutdown**. Синтаксис команды **shutdown** имеет следующий вид:

```
shutdown [действие] [когда] [широковещательное сообщение]
```

Первый параметр означает действие, которое должна выполнить команда **shutdown**, второй указывает время, когда это действие будет выполняться, а третий параметр представляет собой сообщение, которое будет послано всем пользователям, зарегистрированным в данный момент в системе. В таблице 4.1 перечислены все параметры, которые могут использоваться для обозначения *действия*, выполняемого командой **shutdown**:

Таблица 4.1 Параметры команды shutdown

Действие	*Что означает*
не указано	Отключает всех пользователей и переводит систему в однопользовательский режим.
-h	Останавливает систему.
-p	Останавливает систему и выключает электропитание (если это аппаратно и программно возможно).
-r	Перезагружает систему.
-k	Отключает всех пользователей и запрещает дальнейшие регистрацию и вход в систему (за исключением root). Однако система остается в многопользовательском режиме и подключенной к сети.
-o	Закрывает систему, не посылая сигнал процессу **init**. Это не очень хорошо, поскольку в этом случае не запускаются сценарии, закрывающие отдельные программы.
-n	Если при этом задан также и параметр **-o**, то перед закрытием кэш файловой системы (диска) не будет сбрасываться на диск. Это может привести к потере данных.

Временные параметры команды **shutdown** могут быть заданы несколькими способами. Ключевое слово **now** означает, что действие должно быть выполнено немедленно. Возможен также формат **+n**, где *n* указывает через сколько минут будет выполнено указанное действие (это дает пользователям время сохранить файлы и закрыть

программы). Допустим и формат **yymmddhhmm**, позволяющий задать точное время выполнения действия. Здесь **yy** — год, **mm** — месяц, **dd** — день, **hh** — часы (в 24-часовом формате) и **mm** — минуты. Если год и месяц не заданы, то это означает, что действие должно быть выполнено сегодня. Если вы зададите время, которое уже прошло, то на экране появится сообщение, предупреждающее об этом.

Часть команды **shutdown**, называемая "широковещательным сообщением", представляет собой предупреждение, которое через регулярные интервалы времени посылается всем подключенным к системе пользователям. Эти сообщения начинают передаваться за 10 часов до предстоящего закрытия системы и становятся все более частыми по мере приближения момента закрытия.

Ниже дан типичный процесс закрытия системы, в данном случае с 10-минутной задержкой.

```
# shutdown -h +10 Hard disk needs to be replaced
```

В результате команда **shutdown** начнет выполняться в фоновом режиме. Через 10 минут система остановится. Кроме того, на всех терминалах пользователей будет отображено следующее широковещательное сообщение:

```
*** System shutdown message from root@simba.samplenet.org ***
System going down in 10 minutes
Hard disk needs to be replaced
```

За пять минут до предстоящего закрытия система создает файл **/var/run/nologin**. Таким способом запрещается дальнейшая регистрация для входа в систему, а когда кто-нибудь пытается это сделать, у него на экране отображается содержимое этого файла. В этот файл помещается широковещательное сообщение и указывается время закрытия системы. Так, в данном случае, всякий, кто попытается зарегистрироваться в системуе увидит на экране следующее сообщение:

```
NO LOGINS: System going down at 17:57
Hard disk needs to be replaced.
```

Когда наступает время закрытия, выполняются следующие действия:

- В процесс **init** посылается сигнал TERM, который прекращает создание любых новых процессов.
- Процесс **init** читает содержимое файла **/etc/rc.shutdown** и выполняет все сценарии закрытия для отдельных программ, которые там содержатся.
- Всем процессам посылается сигнал TERM и дается время на то, чтобы мягко, то есть корректно, завершить свою работу.
- Всем процессам, которые в течение разумного промежутка времени не отвечают на сигнал TERM, посылается сигнал KILL, который нельзя проигнорировать. Он жестко завершает процесс, грубо прерывая его выполнение.
- С помощью команды **sync** данные из кэша синхронизируются с носителем информации, файловые системы демонтируются и отмечаются как clean.
- Происходит останов ядра.

Кроме того, команда **shutdown** производит в системный журнал запись, в которой отмечается время закрытия системы и кто это сделал. Обратите внимание, что закры-

вать систему может только **root** (либо зарегистрировавшийся как **root**, либо ставший им с помощью команды **su**).

Чтобы закрыть систему сейчас, введите **su** и нажмите Enter. Когда появится соответствующее приглашение, введите пароль **root**. Предполагая, что в системе нет других пользователей, вводим следующую команду:

```
shutdown -h now
```

Когда произойдет закрытие системы, вы увидите на экране следующее сообщение:

```
System halted
Please press any key to reboot
```

И только после этого можно спокойно выключать электропитание вычислительной системы.

Замечания относительно команд halt и reboot

Для останова и перезагрузки системы можно использовать еще две команды: **halt** и **reboot**, соответственно. Однако лучше не привыкать пользоваться этими командами. Ни одна из них не выполняет сценарий **rc.shutdown**, что может привести к неправильному завершению работы некоторых программ. Кроме того, ни одна из этих программ не позволяет задавать временную задержку и ни одна из них не предупреждает пользователей о предстоящем закрытии системы. Поэтому для останова системы всегда используйте команду **shutdown**.

ПРИМЕЧАНИЕ

Если раньше вы работали в DOS и/или Windows, у вас могла выработаться привычка перезагружать систему, используя Ctrl+Alt+Delete. По умолчанию FreeBSD перехватывает сигнал, вырабатываемый этой комбинацией клавиш, и выполняет действия, эквивалентные команде **reboot**. Это может вызвать проблемы, если обычные пользователи имеют доступ к клавиатуре сервера. В главе 11 вы узнаете, как сконфигурировать FreeBSD таким образом, чтобы нажатие комбинации клавиш Ctrl+Alt+Delete не приводило к перезагрузке системы.

5 ГЛАВА

Работа в среде Gnome

- Менеджеры окон
- Gnome
- Рабочий стол
- Панель Gnome
- Работа с окнами
- *Апплеты для среды Gnome*
- Работа с файлами и каталогами
- Менеджеры окон

Менеджер окон (Window Managers) выполняется поверх X-сервера. Внешний вид графического интерфейса системы X-Window определяет именно менеджер окон. X-сервер обеспечивает для графической среды своего рода "каркас"; а менеджер окон управляет тем, как выглядит и функционирует графическая среда. Имеются десятки различных менеджеров окон, предназначенных для работы в системе X-Window.

Gnome

На рис. 5.1 изображен сеанс X, который выполняется в среде Gnome. Вдоль верхнего и нижнего краев экрана расположены панели. Верхняя панель обеспечивает доступ к различным функциональным возможностям, а нижняя панель содержит кнопки выполняющихся приложений.

Для управления отдельными окнами используется менеджер окон, совместимый со средой Gnome. Менеджер окон сам определяет, что он работает в среде Gnome и соответствующим образом модифицирует свою работу. В данном случае для управления окнами в среде Gnome применяется менеджер окон Sawfish. Это, пожалуй, наилучший менеджер окон для среды Gnome (возможно, вам больше понравится IceWM и Enlightenment). Как менять менеджеры окон, рассказывается в главе 6.

Существует несколько разных сред рабочего стола, предназначенных для системы X-Window. В системах UNIX с открытым исходным кодом (таких как, FreeBSD и Linux) наиболее популярны Gnome и KDE. Похоже, что среда Gnome де-факто стала стандартной средой рабочего стола для операционных систем UNIX, так как ее поддержали компании Sun Microsystems, Hewlett-Packard и IBM. Но если вы захотите поработать в среде KDE, то она кратко рассматривается в главе 7.

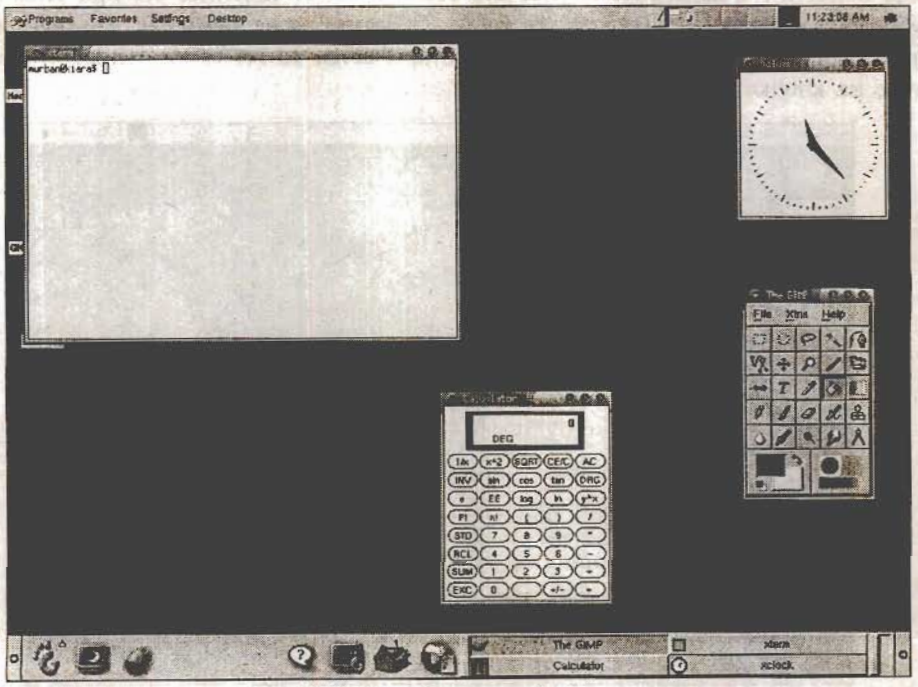


Рисунок 5.1 *Работа в среде Gnome.*

Рабочий стол

Двойной щелчок на пиктограмме, расположенной на рабочем столе, приводит к открытию данного файла; удерживая в нажатом положении левую кнопку мыши, можно перетаскивать разные элементы. Щелчок правой кнопкой мыши на рабочем столе или на каком-нибудь элементе открывает меню.

СОВЕТ

Если пиктограммы на рабочем столе отсутствуют и щелчок правой кнопкой мыши на рабочем столе не приводит к какому-либо видимому эффекту, то, вероятно, во время инсталляции вы не установили **gnomemc**.

Панель Gnome

Среда Gnome позволяет создавать столько панелей, сколько требуется, и производить их всестороннюю настройку — как панель будет сориентирована, где она будет располагаться и что будет на ней находиться. По умолчанию в среде Gnome создается две панели: одна расположена вдоль верхнего края экрана, а другая — вдоль нижнего.

Щелчок на кнопке с отпечатком лапы (это символ среды Gnome) приводит к открытию меню, аналогичного меню Start в операционной системе Windows. Маленькая направленная вверх стрелка (находится справа от кнопки Gnome) указывает, что щелчок на данном элементе открывает подменю. Обратите внимание на пунктирную линию в верхней части этого меню. Если на ней щелкнуть, то меню откреплается от панели и его можно поместить в любом месте рабочего стола. Следует заметить, что при этом меню не убирается с панели, а создается его копия. Обратите также внимание, что при щелчке правой кнопкой на пустом месте панели открывается то же самое меню, что и при щелчке на кнопке Gnome.

Рисунок 5.2 *Плавающие меню в среде Gnome.*

Панель задач

На панели задач находятся кнопки программ, выполняемых в среде в данное время. Темная пиктограмма на кнопке означает, что окно данной задачи свернуто. Если щелкнуть на кнопке, то окно соответствующей задачи переместится на передний план. При щелчке на кнопке правой клавишей мыши открывается подменю. Вот пункты этого подменю:

- **Iconify/Restore** (Свернуть/Восстановить) — минимизация окна или его разворачивание до нормального размера.
- **Shade/Unshade** (Свернуть/Развернуть) — "Сворачивает" окно так, что остается видна только область заголовка). Разворачивает окно, если оно было свернуто.
- **Stick/Unstick** (Приклеить/Отклеить) — "Приклеенное" окно — это окно, которое отображается на всех виртуальных рабочих столах.
- **Close window** (Закрыть окно) — Закрывает окно и заканчивает работу программы, выполняемой в этом окне.
- **Kill app** (Ликвидировать приложение) — также заканчивает работу приложения, выполняемого в окне, но делает это с помощью сигнала SIGKILL. Как правило, ее следует использовать только тогда, когда программа зависла и не реагирует на действия пользователя.

Работа с окнами

Вы сразу же заметите, что раскрывающееся меню в среде Gnome содержит значительно больше команд, чем аналогичные меню в Windows. Вот некоторые из наиболее важных команд с пояснениями:

- **Delete** — В обычном порядке завершает работу приложения в данном окне.
- **Destroy** — Завершает работу приложения с помощью сигнала SIGKILL, грубо прерывая его работу.
- **Toggle** — Открывает подменю, команды которого будут подробно обсуждаться ниже.
- **Maximize/Minimize** — Выполняет ту же самую функцию, что и аналогичные команды в Windows.
- **In Group** — Эта команда будет подробно обсуждаться чуть ниже.
- **Send window to** — Позволяет перемещать или копировать окна в другие виртуальные рабочие пространства.
- **Depth** — Позволяет поместить окно на передний план, переместить его на задний план и т.д.
- **Frame type** — Позволяет изменять различные атрибуты рамки окна.
- **Frame style** — Позволяет полностью изменить внешний вид окна, например, внешний вид области заголовка и кнопок, расположенных в области заголовка. Обратите внимание, что эта команда может изменить и то, как будут функционировать органы управления.

- **History** — При щелчке на одной из команд **Remember** (Запомнить) запоминается соответствующий атрибут окна. Когда программа будет открыта в следующий раз, окно появится на том же самом месте и будет иметь те же самые атрибуты.

Иногда может потребоваться открыть больше окон, чем умещается на одном экране. Здесь приходят на помощь виртуальные десктопы.

Виртуальные десктопы

Виртуальные десктопы позволяют работать сразу с несколькими рабочими столами на одном физическом мониторе. Переключаться между ними можно либо с помощью панели Desk Guide, которая расположена на правой стороне панели, проходящей вдоль верхнего края экрана. Второй вариант — щелкнуть средней кнопкой мыши на рабочем столе, а затем выбрать команду **Workspaces** и нужное рабочее пространство из всплывающего списка.

"Приклеенные" окна

Если вам требуется, чтобы какое-то окно было доступно на всех виртуальных десктопах, для этого окна можно установить свойство **sticky** (приклеенное). Имеется два способа установки этого атрибута:

- Правой кнопкой мыши щелкнуть на кнопке окна, расположенной на панели задач, а затем щелкнуть на команде **Stick** (Приклеить).
- Щелкнуть на оконном меню в верхней левой части области заголовка, выбрать **Toggle** (Переключить), а затем — **Stick**.

Чтобы "отклеить" окно, просто повторите ту же самую процедуру. Переместить окно с одного виртуального десктопа на другой можно так:

- Установить для окна свойство **sticky**, переключиться на тот рабочий стол, куда требуется переместить окно, а там сбросить свойство **sticky**.
- Щелкнуть на оконном меню в верхней левой части области заголовка, выбрать пункт **Sent window to**, а затем выбрать **Previous workspace** или **Next workspace**.

Создание и удаление виртуальных десктопов

По умолчанию FreeBSD предоставляет пользователю четыре виртуальных десктопа; однако их число можно увеличивать или уменьшать. Это делается двумя способами:

- Находясь в последнем по списку виртуальном десктопе, щелкните средней кнопкой мыши на рабочем столе, затем на пункте **Workspaces**, а потом на **Insert workspace**. Будет создано новое пустое рабочее пространство и вы будете находиться в нем.
- Находясь в первом или последнем десктопе, щелкните средней кнопкой мыши на рабочем столе, а затем щелкните, соответственно, на команде **Move workspace left** (Переместить десктоп влево) или **Move workspace right** (Переместить десктоп вправо). Будет создано новый рабочий стол и содержимое текущего десктопа переместится в него.

Чтобы ликвидировать десктоп, щелкните средней кнопкой мыши на рабочем столе, на **Workspaces**, а затем на команде **Merge with next** (Объединить со следующим) или **Merge with previous** (Объединить с предыдущим).

Toggle

Если щелкнуть на оконном меню в верхнем левом углу области заголовка, то на экране появится раскрывающееся меню. Одним из пунктов этого меню является **Toggle**, который открывает одноименное подменю. Ниже перечислены свойства окна, которые содержатся в подменю **Toggle** (Переключить).

- **Sticky**
- **Minimize**
- **Shaded** — Затемненное.
- **Ignored** — Установка этого свойства приводит к тому, что менеджер окон полностью игнорирует данное окно, как будто оно не существует. Это означает, что оно будет отсутствовать в списке окон, его нельзя перемещать или изменять его размеры, и у него будет отсутствовать область заголовка и границы.
- **Focusable** — Если это свойство установлено, то окну можно передавать фокус (сделать его активным). Если это свойство сброшено, то окно можно помещать на передний план и таким образом делать видимым, но его нельзя делать активным (то есть нельзя ничего делать с тем, что выполняется в этом окне).
- **Cyclable** — Если это свойство установлено, то окно включается в совокупность окон, отображаемых на экране при переборе окон. Если оно сброшено, то окно не включается в эту совокупность.
- **In window list** — Если это свойство установлено, то окно будет присутствовать в списке окон, отображаемом с помощью щелчка средней кнопкой мыши на рабочем столе и выбора команды **Windows (Окна)**. Если это свойство сброшено, то окно не будет присутствовать в списке окон.
- **In GNOME task list** — Кнопка окна либо отображается на панели задач среды Gnome, либо нет.

Опция In Group

Опция **In Group** используется для создания групп окон. Эти группы отображаются как подменю в списке окон при щелчке средней кнопкой мыши на рабочем столе.

Опция Depth

Опция Depth (Глубина) позволяет пересылать окно на передний или задний план. В отличие от операционной системы Windows, в среде Gnome окно может находиться на заднем плане и в то же время иметь фокус и быть активным.

Работа с файлами и каталогами

Для запуска File Manager сделайте двойной щелчок на пиктограмме **Home directory**, расположенной на рабочем столе. Менеджер файлов показан на рис. 5.3.

Удерживая нажатой левую кнопку мыши, можно перетаскивать объекты в любое место. Перетаскивание с помощью средней кнопкой мыши приводит к открытию подме-

ню, в котором можно выбрать операцию, которую требуется выполнить над объектом (переслать, скопировать или создать связь). При щелчке на объекте правой кнопкой мыши открывается контекстное подменю.

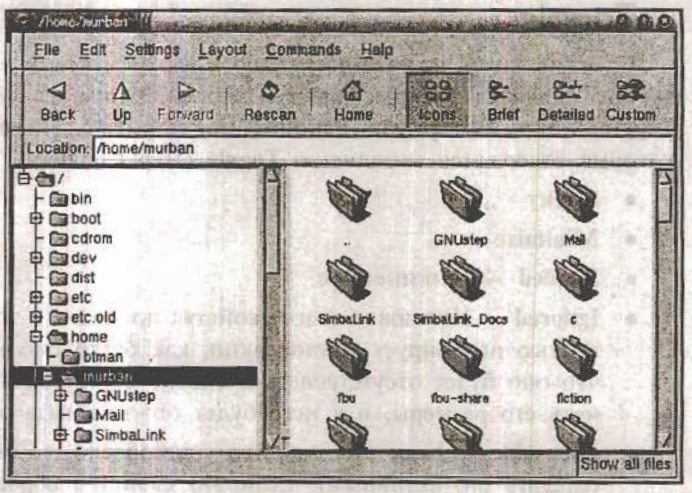


Рисунок 5.3
Программа File Manager.

ПРЕДОСТЕРЕЖЕНИЕ

Обратите внимание на важное различие между командами **Move to trash** (Отправить в корзину) и **Delete** (Удалить) в этом подменю. Команда **Move to trash** отправляет объект в корзину, из которой его можно вернуть, как и в Windows. Команда **Delete** не отправляет объект в корзину, а совсем его удаляет, то есть восстановить данный объект уже нельзя.

Если на рабочем столе необходимо создать ярлык, то объект можно перетащить из окна File Manager прямо на рабочий стол. Если в окне File Manager щелкнуть на пиктограмме **Home**, вы попадете в свой "домашний" каталог.

ГЛАВА 6

Настройка среды Gnome

- Добавление новых пиктограмм на рабочий стол
- Настройка панели Gnome
- Создание и удаление панелей
- *Настройка программы Gnome File Manager*
- Работа с программой Gnome Control Center
- Настройка параметров менеджера окон Sawfish

Добавление новых пиктограмм на рабочий стол

Новые пиктограммы можно добавлять на рабочий стол несколькими способами. Так, их можно просто перетаскивать на рабочий стол из окна программы Gnome File Manager. Если перетаскивать пиктограмму с помощью левой кнопки мыши, то она переместится со своего места на рабочий стол. При перетаскивании пиктограммы с помощью средней кнопки мыши открывается меню с набором команд, которые можно выполнить по отношению к этой пиктограмме. С помощью команды **Link** можно создать символическую ссылку.

Новые пиктограммы на рабочем столе можно также создавать, щелкая правой кнопкой мыши на рабочем столе и выбирая во всплывающем меню команду **New**. В результате появится дополнительное меню с несколькими командами. Описание этих команд приводится в таблице 6.1.

Таблица 6.1 Команды создания новых элементов на рабочем столе

Команда	*Описание*
Terminal	Открывает на рабочем столе новое окно с командной строкой.
Directory	Создает на рабочем столе новый каталог.
URL link	Создает на рабочем столе ярлык для Web-сайта. При двойном щелчке на этом ярлыке происходит запуск браузера Netscape и загрузка указанной Web-странички.
Launcher	Создает на рабочем столе иконку для запуска программы.

Создание на рабочем столе пиктограмм для запуска программ

Создать пиктограмму для запуска программ можно двумя способами. Во-первых, с помощью Gnome File Manager перейдите в каталог, где находится программа. Затем, используя среднюю кнопку мыши, перетащите пиктограмму программы из окна менеджера файлов на рабочий стол и щелкните на команде **Link**. Второй способ — щелкнуть правой кнопкой на пустом месте рабочего стола, выбрать команду **New**, а затем — **Launcher**. По сравнению с первым методом второй обеспечивает более широкие возможности в отношении задания свойств пиктограммы. Кроме того, второй способ иногда бывает также и более быстрым. При первом способе необходимо знать, где находится программа. При втором способе все, что необходимо знать, — это имя исполняемого файла программы. FreeBSD устанавливает исполняемые файлы в те каталоги, для которых заданы пути поиска. На рис. 6.1 изображено диалоговое окно создания новой пусковой пиктограммы.

Предположим, например, что на рабочем столе необходимо создать ярлык для программы **xcalc** (калькулятор). Ниже приведена процедура (с описанием всех параметров), которая для этого используется.

1. **Name** — Это имя пиктограммы, отображаемое на рабочем столе под пиктограммой. Оно предназначено только для пользователей и никак не связано с настоящим именем программы.

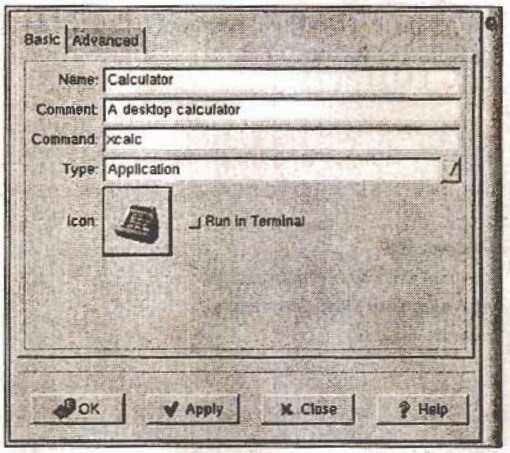


Рисунок 6.1
Создание пиктограммы для запуска калькулятора.

2. **Comment** — Комментарий, предназначен только для удобства пользователей.
3. **Command** — Команда, используемая для запуска программы. В данном случае имя исполняемого файла программы — **xcalc**.
4. **Type** — Щелкните справа на стрелке, направленной вниз, и появится раскрывающееся меню. В данном случае нужно выбрать **Application**.
5. **Run in Terminal** — Если флажок установлен, приложение запускается в окне терминала. Он предназначен для приложений, работающих в этом режиме. Для **xcalc** оставьте флажок сброшенным. Чтобы выбрать для приложения подходящую пиктограмму, щелкните на кнопке **No Icon**. По умолчанию в среде Gnome используются пиктограммы из каталога **/usr/X11R6/share/gnome/pixmaps.**
6. **Advanced** — Эта вкладка содержит редко используемые параметры (например, комментарии на разных языках и выполнение какого-либо предварительного действия перед главным, для которого предназначена пиктограмма).

Настройка панели Gnome

Чтобы получить доступ к большинству параметров настройки панели Gnome, необходимо щелкнуть правой кнопкой на пустом месте панели, выбрать последовательно команды **Panel** и **Properties**, а затем команду в одном из подменю меню **Properties**. Можно также в меню **Panel** выбрать команду **Global Preferences** с тем, чтобы изменить глобальные параметры, относящиеся к панели Gnome.

Работа с пиктограммами панели Gnome

Пиктограммы на панели Gnome бывают трех разных видов. Первый вид — пиктограмма для запуска программ. Второй — пиктограмма интегрированного приложения среды Gnome (например, пиктограмма с оттиском лапы, посредством которой открывается меню Start). Третий — пиктограмма работающего апплета Gnome. При щелчке правой кнопкой на пиктограммах разных видов открываются разные меню. Например, чтобы переместить пиктограмму, щелкните на ней правой кнопкой и затем выберите команду **Move**. Чтобы зафиксировать пиктограмму на новом месте, щелкните на ней левой кнопкой мыши. А если нужно сменить Web-браузер, запускаемый по умолчанию, щелкните на пиктограмме запуска браузера правой

кнопкой и выберите команду **Properties**. На экране появится диалоговое окно, изображенное на рис. 6.2.

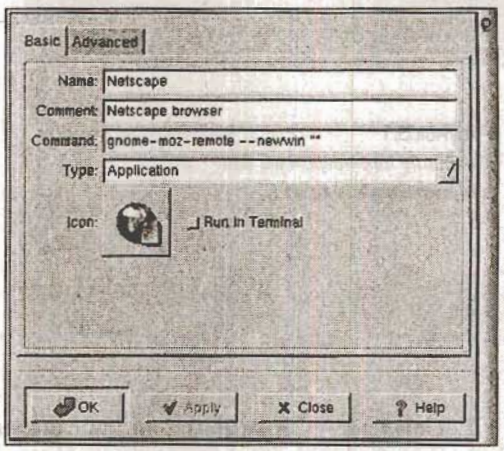


Рисунок 6.2
Диалоговое окно Properties для пиктограммы веб-браузера на панели Gnome.

Настройка меню Start

Пиктограмма Gnome (с оттиском лапы) подобна кнопке Start в Windows. Приложения, написанные для среды Gnome, после установки автоматически появятся на панели. Другие приложения придется добавлять в меню вручную.

Чтобы добавить, удалить или изменить команды в этом меню, щелкните правой кнопкой мыши на пиктограмме с оттиском Gnome и затем выберите команду **Edit Menus**. Редактор меню среды Gnome изображен на рис. 6.3.

В левой части представлена древовидная структура меню. Знак плюс (+) слева от команды указывает на то, что данная команда представляет собой подменю и содержит ряд других команд. Если щелкнуть на знаке плюс, то подменю развернется, и на экране отобразятся входящие в него команды.

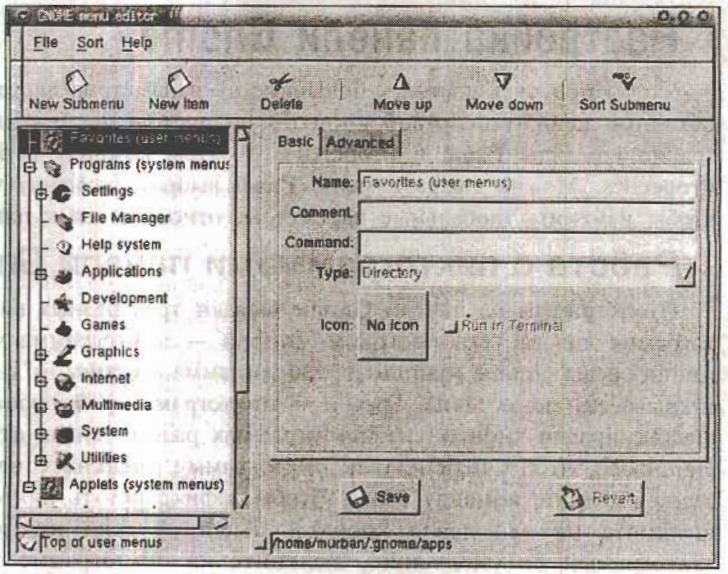


Рисунок 6.3
Редактор меню среды Gnome.

Если щелкнуть на команде **New Submenu** (Новое подменю) или **New Item** (Новая команда), то соответствующий элемент будет вставлен в меню перед тем элементом, который в данный момент подсвечен. После этого, используя правую сторону окна редактора меню, можно редактировать новый элемент точно так же, как это делалось при создании пиктограммы **xcalc**. Не забудьте после редактирования нажать кнопку **Save**.

Если в каком-нибудь подменю щелкнуть на любой команде и затем выбрать команду **Sort Submenu**, то все команды данного подменю будут отсортированы в алфавитном порядке. Если последовательно выбрать **Sort** и **Sort Submenu Recursive**, то будет отсортировано как текущее подменю, так и все подменю, входящие в него.

Работа с апплетами панели Gnome

Апплеты панели — это небольшие приложения, которые запускаются непосредственно на панели Gnome. Имеющиеся апплеты для панели Gnome могут выполнять всевозможные задачи: от контроля состояния батареек в переносном компьютере до проверки электронной почты. Примером могут служить четыре апплета, которые запускаются по умолчанию. Это панель задач, часы, пейджер (позволяет переключаться из одного рабочего пространства в другое) и проверка электронной почты.

Чтобы удалить апплет с панели, щелкните на нем и затем выберите команду **Remove from Panel**. В некоторых случаях для появления этой команды вам придется щелкать правой кнопкой на границе апплета. Примером может служить панель задач. Если щелкнуть на одной из кнопок, имеющихся здесь, то откроется меню команд, которые можно выполнять по отношению к данной программе. Если же щелкнуть на вертикальной полоске в форме углубления, расположенной на левом крае панели задач, то откроется меню с командами, относящимися к самому апплету (например, перемещения или удаления).

Чтобы добавить апплет на панель Gnome, щелкните правой кнопкой на пустом месте панели, выберите команду **Applets**, выберите одну из категорий апплетов, присутствующих в подменю, а затем щелкните на апплете, который нужно добавить.

Для настройки большинства апплетов необходимо щелкнуть правой кнопкой на апплете или на его границе (как в случае с панелью задач) и затем выбрать команду **Properties**.

Создание и удаление панелей

Чтобы удалить с рабочего стола существующую панель, щелкните на ней правой кнопкой, выберите команду **Panel**, а затем — команду **Remove this panel**. Вы получите предупреждение о том, что теряете панель и все расположенные на ней пиктограммы апплетов. Если на панели имеется запущенный апплет, то при ее удалении вы можете получить предупреждение, в котором сообщается, что "The applet appears to have died unexpectedly" (Апплет неожиданно прекратил работу), и спрашивается, не перезагрузить ли апплет. Если вы щелкните на кнопке **Reload**, то апплет будет перезагружен на другую панель.

Чтобы создать новую панель, щелкните правой кнопкой на какой-нибудь существующей панели, выберите **Panel** и **Create panel**, а потом щелкните на том типе панели, который вам требуется создать. В таблице 6.2 перечислены различные типы панелей.

Таблица 6.2 Различные типы панелей в среде Gnome

Тип панели	*Описание*
Кромочная	Панель, которая имеет протяженность от одного края рабочего стола до другого.
Присоединяемая	Панель, которая динамически увеличивается и уменьшается до размера, необходимого для того, чтобы уместить на ней всю требуемую информацию.
Скользящая	Скользящая панель изменяет свои размеры в соответствии с размерами самой большой пиктограммы, расположенной на этой панели.
Плавающая	Плавающая панель может находиться в любом месте рабочего стола, а не только у одной из его сторон.

Если требуется изменить местоположение панели на экране или ее ориентацию (горизонтальная или вертикальная), выберите команду **All properties**. Набор параметров изменяется в зависимости от того, с панелью какого типа вы работаете.

Настройка программы Gnome File Manager

Программа Gnome File Manager напоминает Windows Explorer. Чтобы настроить программу Gnome File Manager в соответствии со своими потребностями, выберите меню **Settings**, а затем — **Preferences** (см. рис. 6.4).

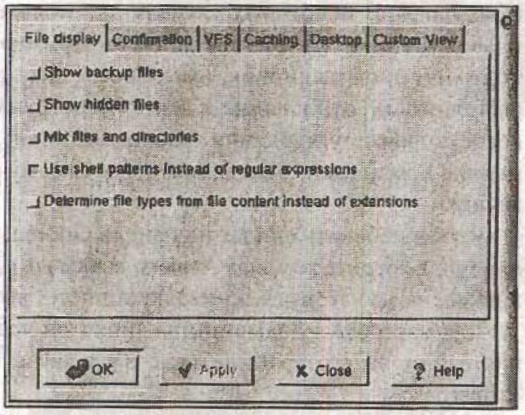


Рисунок 6.4. *Настройка программы Gnome File Manager.*

Вкладка File Display

Первая вкладка позволяет настраивать то, как файлы отображаются на экране, а также то, как производится поиск файлов. Ниже приводятся значения всех параметров:

- **Show backup files** — Имена файлов резервных копий обычно заканчиваются тильдой (~). По умолчанию Gnome File Manager не отображает эти файлы. Если установить флажок, то файлы резервных копий будут отображаться на экране.
- **Show hidden files** — Скрытые файлы называют также "файлами с точкой", поскольку их имена начинаются с точки. Такие файлы обычно не отображаются в содержимом каталогов. Если установить этот флажок, то указанные файлы будут отображаться.
- **Mix files and directories** — Как правило, сначала идет список всех каталогов в алфавитном порядке, а потом — список всех файлов. Данный флажок позволяет отсортировать их все вместе.
- **Use shell patterns instead of regular expressions** — Этот параметр определяет, как использовать шаблоны для поиска файлов с помощью утилиты **find**. О регулярных выражениях можно почитать в главе 8.

- **Determine file types from file content instead of extensions** — Определять типы файлов по их содержанию, а не по расширению.

Параметры на вкладке **Confirmation** в объяснениях не нуждаются.

Вкладка **VFS** содержит параметры, которые определяют, как менеджер файлов работает с виртуальными файловыми системами. Параметры этой вкладки применяются, главным образом, к FTP-серверам. Программа Gnome File Manager обеспечивает четкий доступ по протоколу FTP, позволяя выполнять операции над файлами, расположенными на сервере FTP, так, как если бы они находились на вашем собственном диске. На вкладке VFS имеются следующие параметры:

- **VFS timeout** — время бездействия (простоя) в секундах, по истечении которого соединение с FTP-сервером закрывается.
- **Anonymous FTP password** — Большинство анонимных служб FTP ожидают, что при регистрации вы введете в качестве пароля свой адрес электронной почты, поэтому именно его следует ввести в этом поле.
- **Always use FTP proxy** — Всегда использовать уполномоченный сервер FTP. Если вы не можете связываться с сервером FTP напрямую, а только через уполномоченный сервер, то этот параметр требуется установить.

Вкладка Caching

Параметры, находящиеся на вкладке **Caching**, могут повысить эффективность работы менеджера файлов. Вот эти параметры и их назначение:

- **Fast directory reload** — Если этот параметр установлен, то менеджер файлов кеширует содержимое каталогов (запоминает его в оперативной памяти). Поэтому при повторном обращении к некоторому каталогу менеджеру файлов не требуется повторно считывать его содержимое с жесткого диска.
- **Compute totals before copying files** — Этот параметр, вероятней всего, изменять не следует. По умолчанию он установлен. Менеджер файлов подсчитывает общее число файлов, выбранных для копирования, поэтому после окончания копирования он может выдать вам информацию о состоянии.
- **FTP directory cache timeout** — Данный параметр определяет время в секундах, в течение которого информация в кэше действительна. При кешировании содержимого каталогов новые файлы не отображаются до тех пор, пока не будет нажата кнопка **Rescan**. Если только у вас нет высокоскоростного соединения с сервером FTP, этот параметр трогать не следует. Когда содержимое каталогов не кешируется, производительность менеджера файлов может очень сильно снизиться.
- **Allow customization of icons in icon view** — Пользователь может сам настраивать внешний вид пиктограмм. Когда данный параметр установлен, время, необходимое для отображения содержимого каталога, увеличивается многократно. Этот параметр, по-видимому, лучше оставить сброшенным.

Вкладка Desktop

Позволяет настраивать различные аспекты отображения пиктограмм на рабочем столе.

- **Automatic icon placement** (Автоматическое размещение пиктограмм) — Если этот параметр установлен, пиктограммы автоматически выравниваются по левому краю рабочего стола.
- **Snap icons to grid** (Привязать пиктограммы к сетке) — Когда этот параметр установлен, пиктограммы на рабочем столе выравниваются по невидимой сетке.
- **Use shaped icons** (Использовать пиктограммы в чистом виде) — Когда этот параметр установлен, у пиктограмм на рабочем столе прозрачный фон. Когда он сброшен, у пиктограмм непрозрачный фон в форме квадрата.
- **Use shaped text** (Использовать текст в чистом виде) — Когда этот параметр установлен, может быть тяжело читать текст под пиктограммами (если используются обои пестрой расцветки). Когда он сброшен, текст отображается на фоне непрозрачного прямоугольника.

Работа с Gnome Control Center

Программа Gnome Control Center (Центр управления среды Gnome) эквивалентна программе Windows Control Panel (Панель управления). Здесь можно настраивать почти все аспекты функционирования среды Gnome. Чтобы запустить программу Gnome Control Center, на нижней панели щелкните на пиктограмме, отображающей ящик для инструментов.

Изменение фона рабочего стола

Возможно, вам уже надоело смотреть на голубовато-зеленый фон рабочего стола. Gnome Control Center поможет его изменить. Щелкните на строке **Background** и справа появятся параметры фона (см. рис. 6.5).

В качестве обоев для рабочего стола можно использовать картинку, которая вам нравится. Gnome позволяет использовать изображения наиболее распространенных форматов, включая BMP, JPEG, GIF, PNG и XPM.

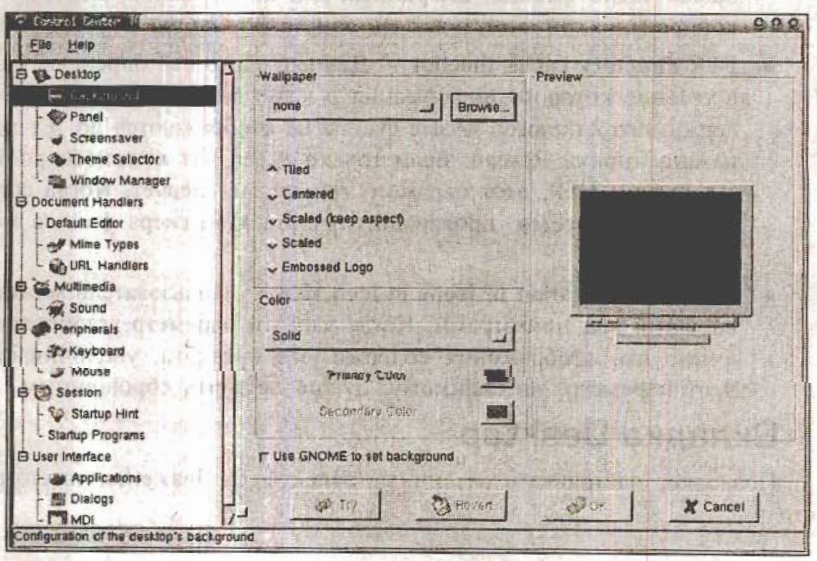


Рисунок 6.5
Настройка фона рабочего стола.

Меню Gnome позволяет настроить все что угодно (см. рис 6.6). Так, выбрав пункт Theme Selector, можно изменять тему рабочего стола, используемую средой Gnome. Тысячи различных тем вместе с инструкциями по их установке можно найти по адресу **gtk.themes.org**.

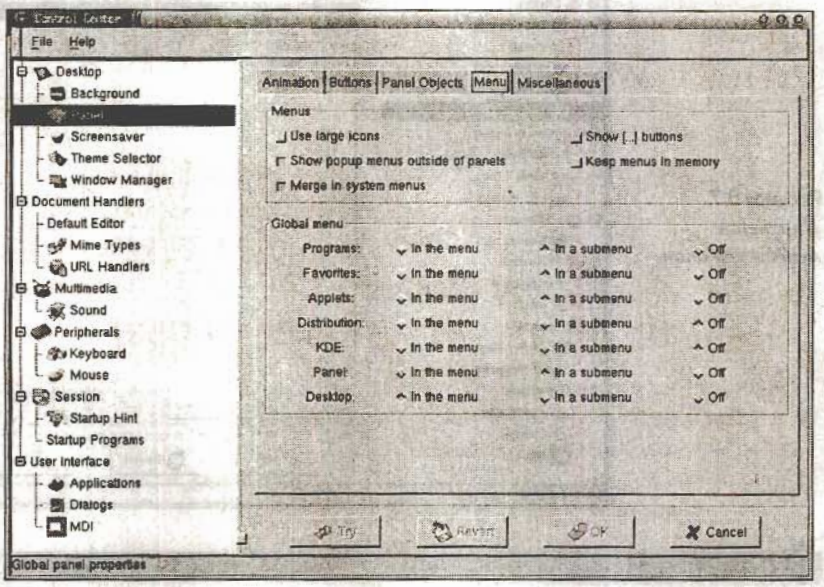


Рисунок 6.6 *Настройка параметров меню Gnome.*

Менеджер окон

По умолчанию в среде Gnome используется менеджер окон Sawfish. Именно с этим менеджером окон мы и работали до настоящего момента. Но если вам не нравится менеджер окон Sawfish, то здесь (см. рис. 6.7) можно выбрать другой менеджер окон. Обратите внимание на то, что при изменении менеджера окон, некоторые органы управления и меню могут измениться. В настоящее время широко используются три менеджера окон, хорошо согласующиеся со средой Gnome, — Sawfish, IceWM и Enlightenment.

Обработчики документов

Параметры данного раздела определяют способы обработки различных документов. Например, как Gnome File Manager обрабатывает файлы различных типов.

Редактор, используемый по умолчанию

Если вы новичок во FreeBSD, то, возможно, с помощью раскрывающегося меню установите в качестве редактора по умолчанию **gEdit**, поскольку **Emacs** более труден в изучении.

Типы mime

Типы mime определяют, как программа Gnome File Manager обрабатывает файлы различных типов и какие приложения должны использоваться для открытия этих файлов при двойном щелчке на пиктограмме. Хотя программа Microsoft Word не работает в операционной системе FreeBSD, в ней работают другие текстовые процессоры, спо-

собные обрабатывать документы Word. Для этого можно использовать, например, текстовый процессор AbiWord с открытым исходным кодом.

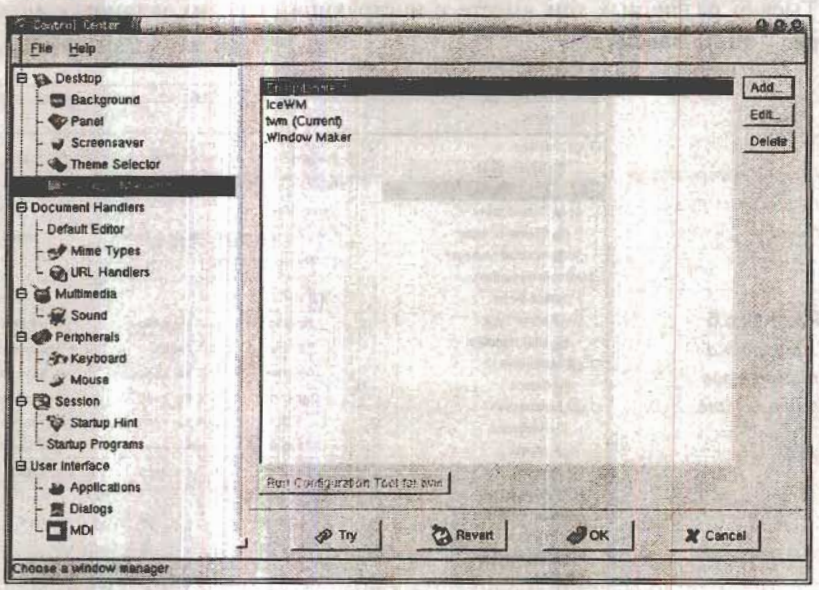


Рисунок 6.7. *Изменение менеджера окон.*

СОВЕТ

Программа Abiword имеется в коллекции портов операционной системы FreeBSD в каталоге "editors". Более подробная информация имеется в главе 15.

Когда вы щелкаете на пиктограмме документа Word, появляется диалоговое окно с вопросом: с помощью какого приложения открыть этот документ? Конечно, отвечать на этот вопрос каждый раз, когда открывается документ, быстро надоест. Поэтому вам потребуется установить тип mime, чтобы программа Gnome File Manager автоматически распознавала документы данного типа как документы Word и выбирала для их открытия соответствующую программу (AbiWord). Просматривая список типов mime в окне центра управления (см. рис. 6.8), вы обнаружите, что среде Gnome ничего неизвестно о существовании документов Microsoft Word.

ПРИМЕЧАНИЕ

Mime — это сокращение от multipurpose Internet mail extensions (многоцелевые почтовые расширения сети Internet). Стандарт mime первоначально был разработан для передачи по электронной почте файлов, а не просто текста. Позднее он был распространен и на WWW. Когда вы посещаете Web-страницу, сервер сначала посылает вам заголовок, информирующий браузер о том, какого типа документ он собирается вам отправлять. Информация в этом заголовке представляет собой тип mime. С его помощью браузер определяет, как обработать документ, который он сейчас получит (отобразить в окне браузера, открыть внешнее приложение для обработки документа или сгрузить документ на диск). Многие типы файлов имеют стандартные названия типов mime, которые хранятся в самих файлах и известны браузерам и почтовым клиентам. Подробная информация о типах mime имеется в документах RFC 2045 и 2046.

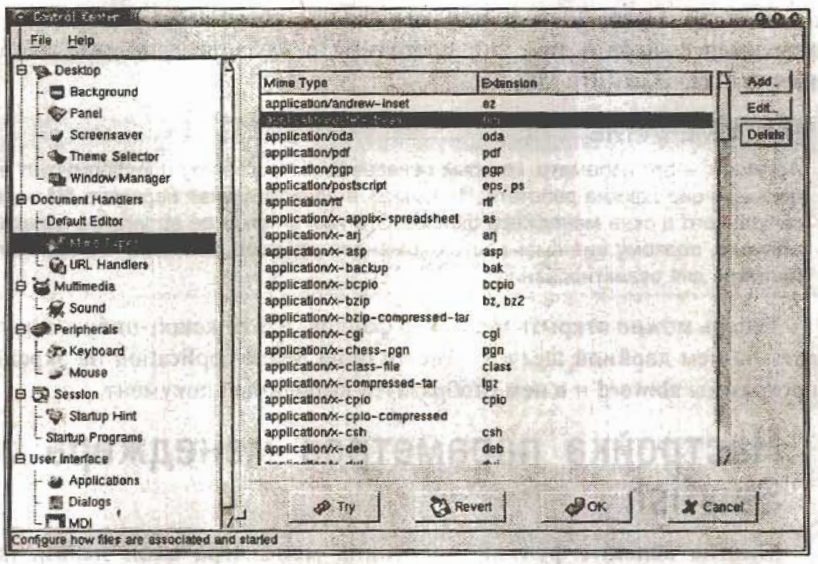


Рисунок 6.8 *Список типов mime в окне центра управления.*

Хотя среде Gnome безразлично, как вы назовете тип mime, следует придерживаться указанных правил. В Internet имеется несколько мест, где можно найти список распространенных типов mime, скажем, по адресу **http://www.isi.edu/in-notes/iana/assignments/media-types/media-types**. Так, например, тип mime для документов Microsoft Word называется application/msword. Чтобы создать тип mime, щелкните на кнопке **"Add"** и на экране появится диалоговое окно (см. рис. 6.9).

Поля Regular Expression в данном случае следует оставить пустыми. Если они заполняются, то менеджер файлов будет стараться найти в документе эти регулярные выражения, чтобы определить тип mime.

Теперь этот тип mime необходимо отредактировать, чтобы связать его с определенным приложением. Поэтому щелкните на новом типе mime и затем — на кнопке **Edit**. На рис. 6.10 показано диалоговое окно Edit.

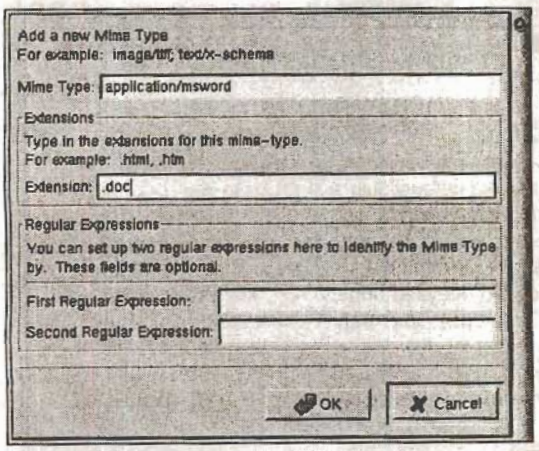


Рисунок 6.9 *Создание типа mime для документов Microsoft Word.*

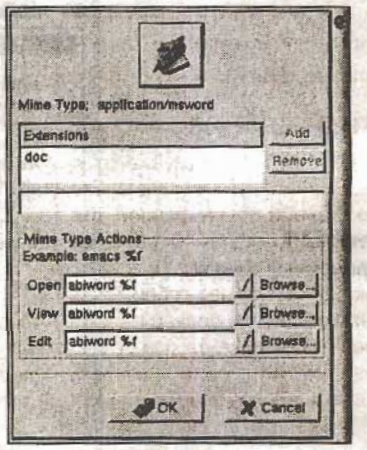


Рисунок 6.10 *Редактирование нового типа mime для документов Word.*

В полях Open, View и Edit мы вводим **abiword %f**. Символы **%f** информируют менеджер файлов о том, что программе в качестве аргумента необходимо передать имя файла. Нажмите OK.

ПРИМЕЧАНИЕ

Аргумент — это параметр, который передается в программу и информирует ее о том, с какими данными она должна работать. Например, в данном случае параметр **%f** содержит имя файла, выбранного в окне менеджера файлов. Это имя в качестве аргумента передается в программу **abiword**, поэтому она вызывается командой **abiword filename** и открывает файл с именем **filename** для редактирования.

Теперь можно открыть менеджер файлов, найти какой-нибудь документ Word и сделать на нем двойной щелчок. Вместо окна Select application на экране откроется окно программы **abiword** и в нем отобразится выбранный документ.

Настройка параметров менеджера окон Sawfish

Многие аспекты функционирования менеджера окон нельзя настроить в окне программы Gnome Control Center. Вам придется использовать программу Sawfish Configurator. Чтобы запустить конфигуратор Sawfish, щелкните средней кнопкой на пустом месте рабочего стола, выберите **Customize**, а затем — **All features** (см. рис. 6.11).

Чтобы со средой рабочего стола было легче работать, сделайте следующие изменения. Щелкните на строке **Focus** и установите флажок параметра **Focus windows when they are first displayed**. Это приведет к тому, что при открытии новые окна будут автоматически делаться активными. Убедитесь также, что установлен флажок параметра **Dialog windows inherit the focus from their parents** (Диалоговые окна наследуют фокус от своих материнских окон). Флажок параметра **Raise windows when they are first focused** (Поднимать окна на передний план, когда им впервые передается фокус) также должен быть установлен. Чтобы выйти из конфигуратора Sawfish после того, как сделаете все изменения, щелкните на кнопке **Ok**.

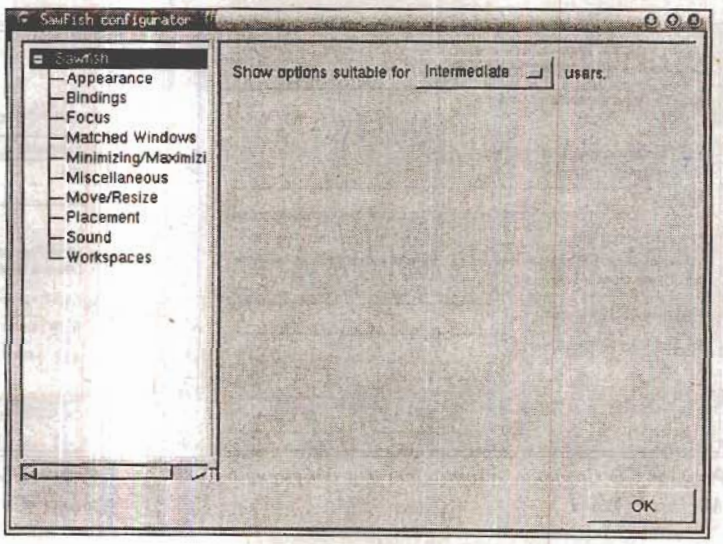


Рисунок 6.11
Конфигуратор Sawfish.

ГЛАВА 7

Работа с приложениями

- Работа с текстом
- Графика и изображения
- Офисный комплект StarOffice
- Мультимедиа
- Сетевые приложения

В операционной системе FreeBSD можно запускать, помимо тысяч своих родных приложений, также и большинство приложений операционной системы Linux. Среди этого множества разнообразных приложений можно найти приложения практически для любых целей. Все приложения, освещаемые в настоящей главе, распространяются бесплатно.

Работа с текстом

Одним из наиболее распространенных приложений, которое нужно всем, является текстовый редактор. В отличие от текстового процессора, он работает с простым текстом. Он не может запоминать изменения в шрифтах, поля и другую подобную информацию, относящуюся к документу. Текстовый редактор обычно используется для следующих целей:

- **Редактирование файлов конфигурации системы**. Для этого требуется текстовый редактор, который может записывать в файл простой текст. Использование текстового процессора для редактирования конфигурационных файлов недопустимо, поскольку операционная система не поймет символы форматирования и шрифтовыделения, которые текстовый процессор сохраняет вместе с документом. (Конфигурационные файлы описаны в главе 11.)
- **Создание и модификация исходного кода программ**. Исходный код программ записывается в виде простого текста. Главы 13 и 21 познакомят вас с двумя популярными языками программирования для операционной системы FreeBSD; для написания кода на этих языках используется текстовый редактор.
- **Создание и модификация Web-страниц**. Web-страницы пишутся на языке HTML. Результатом является файл, содержащий простой текст, понятный Web-браузеру. Многие люди по-прежнему предпочитают писать Web-страницы в простом текстовом редакторе. Зачастую это быстрее, чем в программах типа WYSIWYG (what you see is what you get — что видишь, то и получаешь) с графическим интерфейсом.
- **Сложный набор текста**. Если вы ученый или инженер, вас может заинтересовать TeX и его расширенный макропакет LayTeX, так как они обладают уникальными возможностями представления всевозможных математических уравнений.

Вначале мы рассмотрим два текстовых редактора с графическим интерфейсом, предназначенных для работы в системе X-Window, поскольку они более привычны тем, у кого за плечами опыт работы в Windows или Macintosh. А затем перейдем к редакторам для FreeBSD, работающим в текстовом режиме.

Текстовый редактор gedit

Редактор **gedit** — это, фактически, стандартный текстовый редактор для среды рабочего стола Gnome. Он похож на программу Notepad из Windows в том отношении, что их элементы управления функционируют во многом одинаково. Однако **gedit** обладает гораздо более широкими возможностями, чем программа Notepad. Редактор **gedit** можно найти в пакетах, входящих в компакт-диск FreeBSD, а также в дереве портов в каталоге **editors**. На рис. 7.1 показано, как выглядит редактор **gedit**.

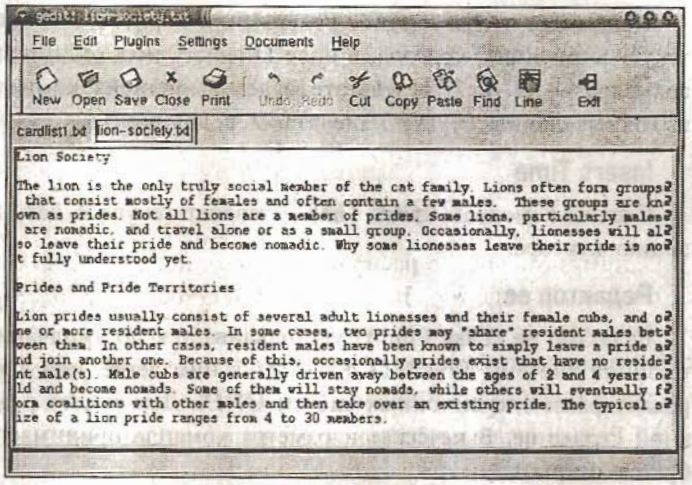


Рисунок 7.1
Текстовый редактор gedit.

Как и следует ожидать, меню File дает возможность открывать, сохранять и распечатывать файлы. Меню Edit позволяет проводить поиск, делать изменения и тому подобное. Другие пункты меню обеспечивают доступ к более сложным функциональным возможностям.

Сменные модули

Редактор **gedit** может использовать сменные модули, расширяющие его функциональные возможности. Чтобы выбрать какой-нибудь сменный модуль, щелкните на меню Plug-ins и найдите в списке требуемый модуль. По умолчанию вместе с редактором **gedit** устанавливаются пять сменных модулей. Ниже дается их краткое описание.

Browse

Если установлен веб-браузер Lynx, то модуль **Browse** позволяет загружать текст с Web-страниц непосредственно в редактор. Затем его можно сохранять как текстовый файл, редактировать и так далее. Обратите внимание, что загружается только текстовая часть страницы, а все ссылки теряются.

Diff

Сменный модуль **Diff** сравнивает два существующих документа, а затем создает третий, отражающий различия в этих двух документах. Устанавливая соответствующий флажок, можно сравнивать, как документы, открытые в редакторе, так и хранящиеся в файлах на диске.

E-mail

Сменный модуль **E-mail** дает возможность отправлять текущий документ по электронной почте нужному адресату.

Shell Output

Сменный модуль **Shell Output** позволяет выполнить команду командного интерпретатора и автоматически вставить выходные данные в текущий документ на позицию курсора.

Например, чтобы листинг содержания корневого каталога вставить в документ на текущую позицию курсора, в поле Directory необходимо ввести /, а в поле Enter shell command — **ls -l**. В результате будет выполнена команда **ls** с параметром **-l** и ее выходные данные будут напечатаны в документе.

Insert Time

Сменный модуль **Insert Time** вставляет текущую дату и время в документ на позицию курсора.

Редактор ee

Редактор **ee** (Easy Editor) устанавливается по умолчанию при инсталляции операционной системы FreeBSD. Работать с ним легче, чем с традиционными текстовыми редакторами операционных систем UNIX. Для вызова редактора **ee** введите в командной строке **ee**. В качестве параметра команда принимает имя файла, который требуется открыть.

Большинство действий в редакторе **ee** выполняются с помощью определенных клавиатурных комбинаций. Символ карат (^) означает клавишу Ctrl. В таблице 7.1 приведены эти комбинации и их назначение.

Таблица 7.1 Клавиатурные комбинации, используемые в редакторе ee.

Комбинация клавиш	*Действие*
Ctrl+o	Позволяет вводить значения символов в коде ASCII. Это необходимо при вводе специальных символов, отсутствующих на клавиатуре; в данном случае вводятся непосредственно коды ASCII.
Ctrl+c	Выводит приглашение, позволяющее вводить команды, перечисленные в меню. Если вводить команду не требуется, просто нажмите клавишу Enter.
Ctrl+y	Выводит приглашение на выполнение поиска. Введите выражение, которое необходимо найти в файле, и нажмите Enter. Будет выполнен поиск первого совпадения с заданным шаблоном после текущего положения курсора.
Ctrl+x	Повторно выполняет поиск, заданный комбинацией клавиш Ctrl+y; в результате будет найден второе совпадение.
Ctrl+g и Ctrl+v	С помощью этих комбинаций клавиш осуществляется переход к последующей и предыдущей страницам. То же самое можно сделать с помощью клавиш Page Up и Page Down.

Настройка и конфигурация редактора ee

Если в вашем home-каталоге имеется файл **.init.ee**, то при каждом запуске редактор **ee** считывает из него параметры конфигурации. Этот файл можно создать вручную. Или же можно во время работы в редакторе **ee** нажать клавишу Esc, в результате чего откроется меню (см. рис. 7.2); затем для доступа к параметрам нажмите **e** или выберите строку **settings** и нажмите Enter.

В таблице 7.2 приведены установочные параметры и их назначение.

Рисунок 7.2 *Меню редактора ee.*

Рисунок 7.3 *Установочные параметры редактора ee.*

Таблица 7.2 Установочные параметры редактора ee.

Параметр	*Описание*
Tabs to spaces	По умолчанию этот параметр сброшен. Будучи установлен, вызывает преобразование символов табуляции в эквивалентное число пробелов. Это полезно для некоторых языков программирования, где пробелы играют важную роль (языки Python и FORTRAN).
Case-sensitive search	По умолчанию сброшен. Параметр определяет, будет ли поиск проводиться с учетом регистра.
Margins observed	По умолчанию сброшен, т.е. перевод курсора не происходит до тех пор, пока вручную не будет вставлен символ новой строки. Когда этот параметр установлен, он вызывает автоматический перевод курсора на начало новой строки при достижении правого поля документа.
Auto paragraph format	По умолчанию сброшен. Если он установлен, редактор пытается автоматически форматировать абзац при наборе текста — аналогично тому, как это делают текстовые процессоры. Обратите внимание, что установка этого параметра автоматически приводит к установке параметра Margins observed. Сброс параметра Margins observed автоматически приводит к сбросу данного параметра.
Eightbit characters (Восьмибитовые символы)	По умолчанию установлен. При этом отображаются 8-битовые символы расширенного кода ASCII. Когда он сброшен, эти символы не отображаются.
Info window	По умолчанию установлен. При этом в верхней части экрана отображается окно menu/help. Когда он сброшен, это окно не отображается.
Emacs key bindings	По умолчанию установлен. При этом клавиатурные комбинации имеют то же значение, что и в редакторе Emacs. Вероятно, не следует изменять данный параметр.
Right margin	Столбец, в котором задается правое поле. По умолчанию оно устанавливается по ширине стандартного 80-колоночного дисплея.
16 bit characters	Он определяет, как обрабатываются 16-битовые символы (как два 8-битовых символа или как один 16-битовый).

Последний параметр в этом меню — Save editor configuration — вызывает запись информации о конфигурации редактора в файл **.init.ee**. Если этот файл уже существует, на место старой информации записывается новая. Если вы запишите этот файл в свой home-каталог, эта конфигурация будет использоваться по умолчанию для всех файлов, которые вы создаете и открываете в редакторе **ee**. Если записать файл конфигурации в текущий каталог, то эти параметры будут перекрывать используемые по умолчанию параметры из файла конфигурации в home-каталоге каждый раз, когда вы запускаете редактор **ee** из данного каталога.

Редактор vi

Редактор **vi** — один из первых редакторов, разработанных для операционных систем UNIX. Он и по сей день остается одним из самых мощных редакторов и стандартно поставляется практически с каждой операционной системой типа UNIX. К сожалению, среди новичков редактор **vi** пользуется репутацией программы, известной своей загадочностью и трудностью в изучении. В нем отсутствуют меню, и все действия осуществляются с помощью клавиш и клавиатурных комбинаций. Понятно, что на их изучение требуется время.

Так зачем же изучать редактор **vi**? Есть, по меньшей мере, две причины. Во-первых, он имеется в любой операционной системе типа UNIX, с которой вам быть может, придется работать. Как знать, в операционной системе типа UNIX, он может оказаться единственным редактором...

Во-вторых, когда вы изучите различные комбинации клавиш и команды, окажется, что в вашем распоряжении очень мощное средство. Если вы хорошо печатаете, то редактор **vi** обеспечит возможность очень быстрой работы, так как для набора большинства команд не придется убирать руки с базовых клавиш.

Чтобы запустить редактор **vi**, необходимо ввести команду **vi**. Если за ней последует имя файла, он откроется для редактирования, а если вы зададите имя несуществующего файла, то **vi** будет считать, что это новый файл.

Сразу после запуска редактор **vi** будет находиться в командном режиме. В этом режиме нажатия клавиш интерпретируются как команды редактору, а не как текст, вводимый в документ. Чтобы переключиться в режим ввода текста, необходимо нажать одну из клавиш: a, i или o.

Клавиша **a** означает *append* (*присоединить*). В этом режиме вводимый текст вставляется после символа, на котором находится курсор.

Клавиша **i** означает *insert* (*вставить*). В этом режиме вводимый текст вставляется перед символом, на котором находится курсор.

И наконец, клавиша **o** означает *open* (*открыть*). Это приводит к тому, что после строки, на которой находится курсор, в текст вставляется новая строка. Затем курсор перемещается на новую строку и редактор **vi** переходит в режим insert, разрешая ввод текста на новой строке.

Имеется также несколько других команд для установки режима ввода текста, которые применяются не так часто. Это команда **O**, которая добавляет пустую строку над текущей строкой; и команда **A**, которая начинает вставлять текст в конце текущей строки.

Если из режима ввода текста вы хотите вернуться в командный режим, нажмите клавишу Escape. При переходе в командный режим редактор **vi** по умолчанию выдает звуковой сигнал.

Перемещение по тексту в редакторе vi

Перемещаться по тексту в режиме ввода текста можно, как правило, с помощью курсорных клавиш или клавиш Page Up / Page Down. Однако на терминале это не всегда функционирует. Кроме того, на некоторых терминалах эти клавиши могут отсутствовать. На этот случай имеются другие клавиши, позволяющие перемещаться по документу в командном режиме.

Чтобы воспользоваться этими клавишами перемещения, нажмите клавишу Escape и перейдите в командный режим. В этом режиме можно пользоваться клавишами **h**, **j**, **k** и **l** для перемещения курсора влево, вниз, вверх и вправо, соответственно. Вот советы, которые помогут запомнить, в каком направлении перемещает курсор каждая из этих клавиш:

- Клавиша l — крайняя справа и поэтому перемещает курсор вправо.
- Клавиша h — крайняя слева и перемещает курсор влево.
- Клавиша j немного похожа на стрелку, направленную вниз. Она перемещает курсор вниз.
- С буквы k начинается слово kaif, при котором душа улетает вверх — вместе с курсором.

В командном режиме имеется еще несколько клавиш перемещения. В таблице 7.3 приведены различные клавиши и их функции.

Таблица 7.3 Клавиши перемещения, функционирующие в командном режиме редактора vi

Клавиша	*Действие*
h	Перемещает курсор влево на один символ.
j	Перемещает курсор вниз на один символ.
k	Перемещает курсор вверх на один символ.
l	Перемещает курсор вправо на один символ.
w	Перемещает курсор вперед на одно слово.
b	Перемещает курсор назад на одно слово.
e	Перемещает курсор в конец следующего слова.
0	Перемещает курсор в начало строки.
$	Перемещает курсор в конец строки.
)	Перемещает курсор в начало следующего предложения.
(	Перемещает курсор в начало предыдущего предложения.
}	Перемещает курсор в начало следующего абзаца.
{	Перемещает курсор в начало предыдущего абзаца.
G	Перемещает курсор в конец текущего документа.

Клавиша	Действие
^	Перемещает курсор к первому символу строки, не являющемуся пробелом.
H	Перемещает курсор на первую строку на экране.
L	Перемещает курсор на последнюю строку на экране.

Обратите внимание, что с каждой командой этой таблицы по умолчанию используется число 1. Клавиша j перемещает курсор вниз на одну строку, клавиша k перемещает его вверх на одну строку, клавиша w перемещает вправо на одно слово и так далее. Все эти команды можно модифицировать, вводя перед ними число. Так, следующая команда перемещает курсор вниз не на одну строку, а на пять:

```
5j
```

Следующая команда перемещает курсор на 75-ую строку файла, редактируемого в данный момент:

```
75G
```

А вот команда, которая перемещает курсор на пятую снизу строку экрана:

```
5L
```

Данный синтаксис справедлив для всех команд в таблице 7.3, за исключением команды ^, которая перемещает курсор к первому символу документа, не являющемуся пробелом.

Прочие клавиши перемещения

Помимо описанных выше клавиш перемещения курсора, имеется еще несколько клавиатурных комбинаций, выполняющих прокрутку текста по экрану (они перечислены в таблице 7.4).

Таблица 7.4 Прокрутка текста в редакторе vi

Комбинация клавиш	Действие
z, затем Enter	Перемещает строку, на которой находится курсор, вверх экрана.
z, затем -	Перемещает строку, на которой находится курсор, вниз экрана.
z, затем .	Перемещает строку, на которой находится курсор, в центр экрана.
Ctrl+u	Прокручивает текст на пол-экрана вверх.
Ctrl+d	Прокручивает текст на пол-экрана вниз.
Ctrl+f	Прокручивает текст вперед на целый экран.
Ctrl+b	Прокручивает текст назад на целый экран.
Ctrl+e	Прокручивает текст вниз на одну строку.
Ctrl+y	Прокручивает текст вверх на одну строку.

Команды редактирования текста

В редакторе vi клавиши Backspace и Delete не выполняют тех действий, которых от них можно ожидать. Для удаления текста и тому подобного придется пользоваться

различными клавишами в командном режиме редактора **vi**. В таблице 7.5 перечислены различные команды редактирования текста, имеющиеся в редакторе **vi**.

Таблица 7.5 Команды редактирования текста в редакторе vi

Клавиша	*Действие*
D	Удаляет текст от позиции курсора до конца строки.
dd	Удаляет всю текущую строку целиком.
*n*dd	Здесь *n* — число удаляемых строк. Например, команда **5dd** удаляет текущую строку и четыре строки, следующие за ней.
r*c*	Здесь *c* — символ. Эта команда заменяет символ в позиции курсора символом, который указан за r.
R	Текст, вводимый после этой команды, замещает текущий текст, начиная с позиции курсора. Этот режим действует до тех пор, пока не будет нажата клавиша Escape, возвращающая редактор в командный режим.
S	Удаляет текущую строку и начинает ввод текста на новой пустой строке.
x	Удаляет символ в позиции курсора и сдвигает следующие за ним символы влево.
X	Удаляет символ перед курсором и сдвигает следующие за ним символы влево.
~	Заменяет букву на позиции курсора той же буквой другого регистра.
J	Объединяет текущую строку с предыдущей.

Операции над файлами и выход из редактора vi

Это операции загрузки и сохранения файлов в редакторе **vi**. Соответствующие команды даны в таблице 7.6.

Таблица 7.6 Операции над файлами в редакторе vi

Клавиша	*Действие*
ZZ	Сохраняет изменения в текущем файле и осуществляет выход из программы.
:wq	Сохраняет изменения в текущем файле и осуществляет выход из программы (то же, что и ZZ).
:w	Сохраняет изменения в текущем файле.
:w!	Сохраняет изменения в текущем файле, замещая существующий файл с таким же именем.
:q	Осуществляет выход из программы. Если имеются несохраненные изменения, редактор vi выдает сообщение об этом и программа не закрывается.
:q!	Осуществляет выход из редактора vi, даже если имеются несохраненные изменения. Изменения при этом теряются.
:e *filename*	Загружает заданный файл в редактор vi для редактирования. Если заданный файл не существует, то создается новый файл.
:e!	Отбрасывает все изменения и перезагружает с диска старый вариант файла.

Поиск в тексте и замена текста в редакторе vi

В редакторе vi имеется несколько команд для выполнения поиска и замены. Они перечислены в таблице 7.7.

Таблица 7.7 Команды поиска и замены в редакторе vi

Клавиша (и)	*Действие*
/шаблон	Здесь *шаблон* — это фрагмент текста (слово, словосочетание, фраза и т. п.), который требуется найти. Редактор vi осуществляет поиск в файле в прямом направлении до первого совпадения с заданным шаблоном.
/	Повторяет поиск заданного образца в прямом направлении до обнаружения следующего совпадения.
?шаблон	Здесь *шаблон* — это фрагмент текста, который требуется найти. Редактор vi осуществляет поиск в файле в обратном направлении до первого встреченного совпадения с заданным шаблоном.
?	Повторяет поиск шаблона в обратном направлении до обнаружения очередного совпадения.
%	Перемещает курсор на соответствующую парную скобку. Эта команда полезна для программистов.
:s/шаблон1 /шаблон2	Заменяет в текущей строке каждое совпадение *шаблона1* на *шаблон2*.
:%s/шаблон1 /шаблон2	Заменяет в файле каждое совпадение *шаблона1* на *шаблон2*.

Копирование, вырезание и вставка текста в редакторе vi

Для копирования текста в буфер в редакторе **vi** используется команда **y** (**y** — это сокращение от yank — выдерживать). Она, преимущественно, помещает текст в буфер. В таблице 7.8 приведены разновидности команды **yank**.

Таблица 7.8 Разновидности команды Yank в редакторе vi

Команда	*Действие*
yw	Помещает в буфер слово, на котором в данный момент находится курсор.
y$	Помещает в буфер текст от текущей позиции курсора до конца данной строки.
yy	Помещает в буфер всю текущую строку.
*n*yy	Здесь *n* — число строк, заносимых в буфер. Например, команда **5yy** помещает в буфер текущую строку, а также четыре строки, следующие за ней.

Текст из буфера можно вставить на любое место документа, перемещая на требуемое место курсор и применяя команду p или P. Команда p вставляет текст в документ после курсора. Команда P вставляет текст перед курсором. Текст продолжает оставаться в буфере и после того, как вы вставите его в документ. Поэтому можно снова использовать команду p или P, чтобы вставить текст в документ еще раз, в другом месте.

ПРЕДОСТЕРЕЖЕНИЕ

Редактор **vi** хранит в буфере текст только самой последней операции копирования текста в буфер или удаления. Другими словами, если вы выполнили команду **dd**, чтобы удалить строку текста, а потом выполнили команду **yy**, чтобы скопировать строку текста, то в буфере текст, с которым оперировала команда **dd**, будет замещен текстом операции **yy**. Это означает, что текст из операции **dd** будет потерян, т.е. операцию удаления отменить будет невозможно.

Графика и изображения

Имеется несколько графических программ и программ редактирования изображений, работающих под управлением FreeBSD. К ним относится программа редактирования изображений Gimp и программа создания графических фигур и иллюстраций Xfig. Все программы, рассмотренные ниже, доступны в коллекции портов FreeBSD.

Программа GIMP

GIMP — это The GNU Image Manipulation Program. Это современный редактор изображений с открытым исходным кодом, сравнимый по возможностям с дорогостоящими коммерческими графическими программами. Если раньше вы работали с Adobe Photoshop, то большинство элементов GIMP будет вам знакомо. GIMP распознает большинство форматов файлов изображений, включая BMP, GIF, JPEG, PNG, PCX и TIFF, а также формат PostScript.

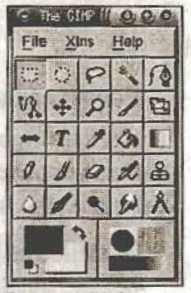


Рисунок 7.4
Главная панель управления GIMP.

GIMP можно найти в каталоге **graphics** коллекции портов FreeBSD. После того как GIMP будет установлен, его можно запускать, вводя **gimp** в окне теминала системы X-Window или в диалоговом окне Run. На рис. 7.4 изображена главная панель управления программы GIMP.

Открытие файла

Для открытия уже существующего изображения, выберите пункт меню **File** или нажмите **Ctr+O**. Появится окошко Load Image, позволяющее выбрать нужный файл. Выбранное изображение раскроется в отдельном плавающем окне. Если щелкнуть правой кнопкой мыши на окне изображения, появится набор всплывающих меню, которые могут быть использованы для обработки изображения. GIMP предоставляет практически все инструменты, эффекты и технологии, которые требуются от современного компьютерного редактора изображений! На рис. 7.5 показано изображение, загруженное в программу GIMP.

Получение снимка экрана в программе GIMP

Программа GIMP позволяет делать снимки экрана и получать таким образом изображения. Для этого на панели управления щелкните на меню File, а затем на команде Acquire (Получить). В открывшемся подменю щелкните на команде Screen Shot (Снимок экрана).

Рисунок 7.5 *GIMP и инструменты редактирования изображений.*

Редактирование изображений

Когда в программе GIMP открыто окно изображения, доступ к большинству элементов управления осуществляется с панели управления программы GIMP или щелчком правой кнопкой мыши на изображении.

Изменение режима отображения

Режим отображения — это цветовой режим, используемый для отображения картинок. Имеется три возможности: режим RGB (16,7 миллиона цветов), режим Grayscale (Шкала серого) и режим Indexed (Индексированные цвета; используется палитра, содержащая не более 256 цветов). Для доступа к этому меню необходимо щелкнуть правой кнопкой на окне редактирования изображения, выбрать команду Image, а затем — команду Mode.

Яркость и цветовой баланс

Для регулировки яркости, контрастности, цветового баланса и пр. необходимо щелкнуть правой кнопкой на изображении, затем на команде Image и Colors. Обратите внимание, что для регулировки большинства этих параметров у изображения должен быть цветовой режим RGB или Grayscale. Если установлен цветовой режим Indexed (часто применяется для Web-изображений), то ряд параметров будет недоступен. Если требуется изменить свойства изображения, отображаемого в режиме Indexed, то сначала придется установить режим RGB или Grayscale. После выполнения регулировок, можно опять установить режим Indexed.

Фильтры

В программе GIMP имеется много фильтров, доступ к которым осуществляется щелчком правой кнопкой на изображении и выбором команды Filters (Фильтры).

Чтобы функционировало большинство этих фильтров, изображение должно быть в формате RGB.

Команда Script-Fu

Команды в подменю Script-Fu создают разные интересные эффекты, например, придают изображениям вид старых фотографий, добавляют на изображение пятна кофе, придают изображению вид вышивки на ткани и т.д. Для доступа к командам в подменю Script-Fu щелкните правой кнопкой мыши где-нибудь на изображении и затем выберите команду Script-Fu.

Инструментальные средства GIMP

На панели управления программы GIMP имеется много разных инструментов, предназначенных для внесения изменений в изображения. В табл. 7.9 описаны функции всех инструментов, имеющихся в программе GIMP. Двойной щелчок на пиктограмме инструмента приводит к открытию диалогового окна с параметрами конфигурации для данного инструмента.

Таблица 7.9 Инструменты для работы с изображениями, имеющиеся в программе GIMP.

Пиктограмма	*Функция*
1	Выделение прямоугольной области. Удерживайте левую кнопку в нажатом положении и перемещайте указатель мыши по изображению.
2	Выделение овальной области. Удерживайте левую кнопку мыши в нажатом положении и перемещаайте указатель мыши по изображению.
3	Выделение произвольной области. Удерживайте левую кнопку мыши в нажатом положении и, перетаскивая указатель мыши по изображению, выделяйте требуемую область произвольной формы.
4	Волшебная палочка. Щелчок левой кнопкой мыши приводит к тому, что инструмент выделяет смежные с данной точкой области, имеющие похожий цвет.
5	Кривая Безье. Данный инструмент позволяет выделять области изображения с помощью кривой Безье.
6	Интеллектуальные ножницы. Этот инструмент позволяет выделять на изображении области различной формы.
7	Перемещение. Позволяет перемещать слои, а также части изображения, выделенные с помощью одного из инструментов выделения.
8	Масштаб изображения. При щелчке левой кнопкой мыши изображение увеличивается. Если во время щелчка левой кнопкой удерживать в нажатом положении клавишу Ctrl, то изображение уменьшится.
9	Инструменты кадрирования. Позволяют вырезать часть изображения, а также изменить его размер.
10	Вращение и перспектива. С помощью этого инструмента на изображение накладывается сетка. Перетаскивая эту сетку можно вращать изображение.
11	Переворот. При щелчке на изображении данный инструмент переворачивает изображение, давая, преимущественно, его зеркальное отражение.

Пиктограмма	Функция
12	Текст. Данный инструмент открывает диалоговое окно, позволяющее добавлять к изображению текст. Можно менять тип шрифта, его размер и т.д.
13	Пипетка. Если щелкнуть на этом инструменте, а затем где-нибудь на изображении, то цвет данной точки станет цветом переднего плана.
14	Заливка. Если щелкнуть на какой-нибудь области изображения, то произойдет заливка этой области либо цветом переднего плана, либо цветом фона, либо в соответствии с шаблоном. Двойной щелчок позволяет изменить конфигурацию инструмента.
15	Градиент. Позволяет создавать цветовые переходы. По умолчанию осуществляется переход от цвета переднего плана к цвету фона.
16	Карандаш. Когда левая кнопка мыши удерживается в нажатом положении, этот инструмент рисует резкие, четкие линии.
17	Кисть. Рисует менее резкими, чем карандаш, более расплывчатыми мазками.
18	Ластик. Этот инструмент стирает выбранные области изображения и заменяет их цветом фона.
19	Аэрограф.
20	Штамп. Это инструмент для клонирования. Можно выделить определенную область изображения, а затем скопировать ее в другое место. Используется для ретуширования.
21	Размытость/Резкость. Если провести данным инструментом по какой-нибудь области изображения, то эта область станет либо более размытой, либо, наоборот, более резкой.
22	Чернильная ручка. Данный инструмент рисует подобно чернильной ручке.
23	Осветлитель. Если провести этим инструментом по какой-нибудь области изображения, то она станет более светлой.
24	Палец (Smudge tool). Данный инструмент рисует широкими мазками, создавая эффект использования акварельных красок.
25	Измерительный инструмент. Этот инструмент служит для измерения расстояний и углов на изображении.

В GIMP имеется 86 типов кистей, 168 различных шаблонов, около 40 цветовых палитр, 100 сценариев и более 200 сменных модулей, которые можно использовать для создания изображений и работы с ними.

GQview

GQview — это программа просмотра изображений для среды Gnome. Она находится в каталоге **graphics** коллекции портов FreeBSD и может работать с файлами большинства популярных форматов. Две наиболее приятные особенности GQview — это режим демонстрации слайдов и полноэкранный режим отображения изображений.

Чтобы использовать GQview для этих целях, просто сохраните все слайды, предназначенные для демонстрации, в отдельном каталоге. Введите **gqview** в окне терминала X. Затем выберите соответствующий каталог на левой стороне окна программы

GQview. Обратите внимание: чтобы сделать каталог текущим, необходимо, в отличие от других программ, щелкнуть на его пиктограмме только один раз. Находясь в текущем каталоге, щелкните на клавише v и программа переключится в полноэкранный режим. Если требуется, чтобы переход к очередному слайду осуществлялся автоматически, без щелчка кнопкой мыши, щелкните правой кнопкой мыши на экране и выберите команду Start slideshow.

StarOffice

StarOffice — это офисный комплект с полным набором функциональных возможностей, который распространяется компанией Sun Microsystems. Его можно приобрести на компакт-диске за умеренную плату или бесплатно загрузить из сети Internet. По своим функциональным возможностям StarOffice не уступает таким пакетам, как Microsoft Office, Lotus SmartSuite и т.д. В состав комплекта StarOffice входят: текстовый процессор, электронная таблица, программа подготовки презентаций, программа рисования, клиент электронной почты, HTML-редактор и база данных. StarOffice может быть установлен из каталога **editors** коллекции портов FreeBSD.

СОВЕТ

StarOffice — это программа для разных платформ, которая доступна также в Microsoft Windows, Solaris x86 и Solaris Sparc. Посетите Web-сайт **www.sun.com/products/staroffice/get.html**. Здесь вы найдете информацию о том, как сгрузить StarOffice или заказать компакт-диски компании Sun. Пакет Deluxe комплектуется документацией и в розницу стоит $39.95.

Рабочий стол StarOffice

На рис. 7.6 изображен рабочий стол комплекта StarOffice. В StarOffice предусмотрено несколько разных способов создания новых документов:

- Дважды щелкнуть на одной из пиктограмм новых документов, расположенных на рабочем столе.

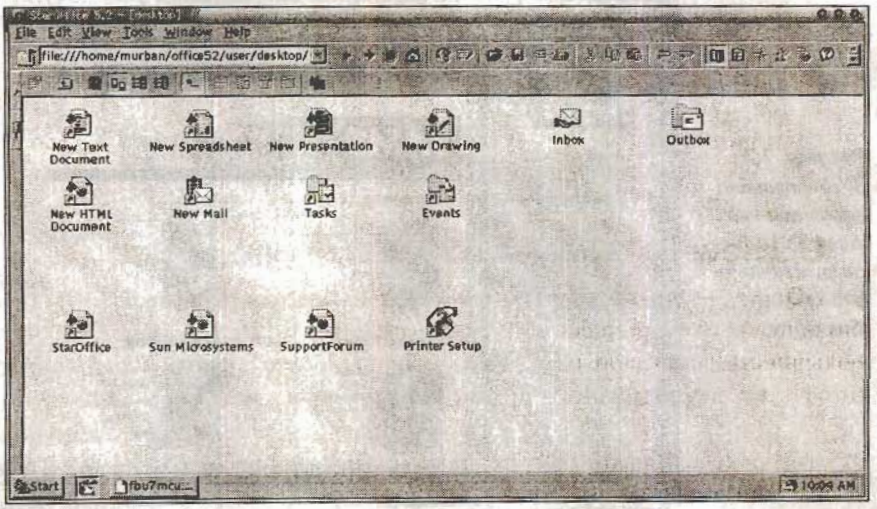


Рисунок 7.6 *Рабочий стол StarOffice.*

- Выбрать меню File, New и один из имеющихся типов документов.
- Выбрать меню File, New и From Template (по шаблону).

Встроенный Web-браузер StarOffice

Чтобы обратиться к Web-браузеру, в меню File выберите команду Open; затем в диалоговом окне Filename введите URL того Web-сайта, который хотите открыть.

Поскольку Web-браузер интегрирован в комплект StarOffice, содержимое Web-сайтов можно открывать прямо в окне текущего документа (см. рис. 7.7). Переключаться с Web-страницы на текущий документ можно с помощью стрелок Back и Forward. Они аналогичны кнопкам Back и Forward, имеющимся в Web-браузерах. Однако в StarOffice их функции распространяются на все типы документов. Выбрав меню Tools, а затем команду Option, можно настроить по своему усмотрению различные аспекты работы Web-браузера.

Электронная почта в StarOffice

В StarOffice включена также интегрированная программа электронной почты. Однако перед использованием ее необходимо сконфигурировать. В панели меню выберите Tools, а затем Options. В диалоговом окне, которое появится на экране, щелкните на знаке плюс (+) рядом с разделом General, а потом на строке User Data (см. рис 7.8).

Потребуется также создать новую учетную запись электронной почты. Щелкните правой кнопкой мыши где-нибудь на рабочем столе StarOffice. Во всплывающем меню выберите тип формуляра электронной почты, который вы хотите создать. Вероятно, это будет формуляр типа POP3 (или IMAP). Введите адрес почтового сервера; имя пользователя и пароль, используемые для доступа к серверу.

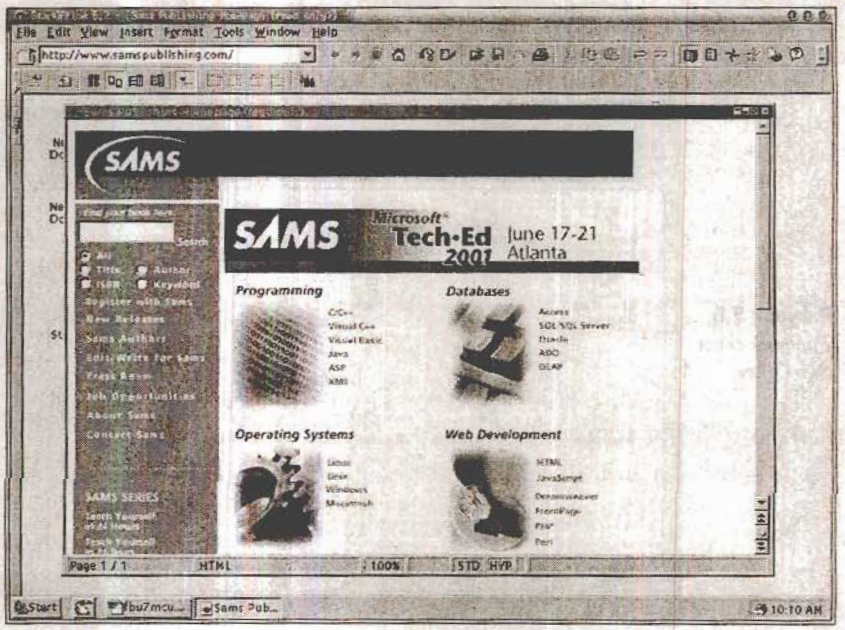


Рисунок 7.7 *Web-страница издательства Sams Publishing, загруженная в веб-браузер StarOffice.*

Чтобы программа автоматически отслеживала поступление свежей почты, на вкладке Contents установите флажок параметра Include in update function и число минут между проверками. Возможно, потребуется установить флажки параметров Remove messages from server и Save document contents locally. В противном случае ваша электронная почта будет оставаться на сервере и будет недоступной в то время, когда у вас не установлено соединение с сетью Internet. Для отправки сообщений необходим также почтовый ящик для исходящей почты. На вопрос программы, не хотите ли вы его создать, ответьте Yes.

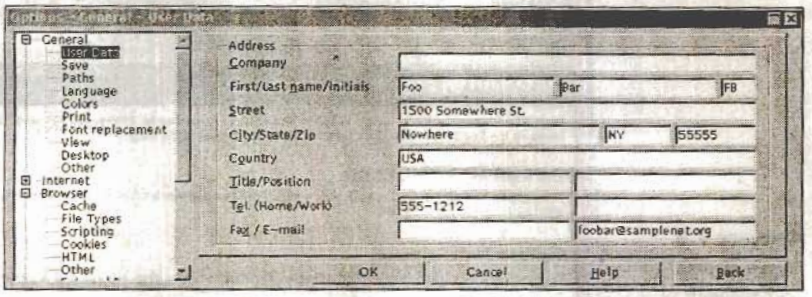


Рисунок 7.8 *Сведения о пользователе StarOffice. Убедитесь, что ваш электронный адрес указан в нижнем правом поле.*

Введите имя этого выходного почтового ящика, вероятно, прекрасно подойдет простое имя вроде **Outbox**. Щелкните на вкладке SMTP. В поле Server введите адрес сервера исходящей почты. Кроме того, проверьте, верна ли информация в поле Sender. В поле Reply to ничего вводить не надо.

Щелкая правой кнопкой на пиктограмме входного или выходного почтового ящика и выбирая команду Properties, вы можете в любое время изменить введенную информацию.

Прием и отправка почты

Когда поступает свежая электронная почта, пиктограмма, имеющая вид земного шара и расположенная рядом с часами на панели задач StarOffice, начинает мигать. Дважды щелкните на этой пиктограмме, и StarOffice немедленно проверит наличие свежей почты.

Создать новое почтовое сообщение можно несколькими способами:

- В любом приложении StarOffice выбрать File, New и Mail.
- Щелкнуть правой кнопкой на рабочем столе StarOffice; во всплывающем меню последовательно выбрать New, Documents и Mail.
- Во входном или выходном почтовом ящике щелкнуть на пиктограмме New Mail.

StarOffice Explorer

StarOffice Explorer (Проводник) обеспечивает быстрый доступ к документам, шаблонам, корзине и так далее. Explorer может пребывать в одном из трех режимов: visible (видимый), hidden (скрытый) и turned off (отключенный). Обратите также внимание на корзину (Recycle Bin). Стоящее за ней число в круглых скобках означает число файлов, находящихся в ней.

Справочная информация в StarOffice

Для получения справочной информации в StarOffice существует два основных способа. Во-первых, это Help Agent. Он предоставляет контекстно-зависимую справку, касающуюся того, что вы делаете в текущий момент. Во-вторых, это меню Help, расположенное на панели меню. Данная справка (см. рис. 7.9) автоматически открывается раздел справочной системы, относящийся к той программе комплекта StarOffice, в которой вы в данное время работаете.

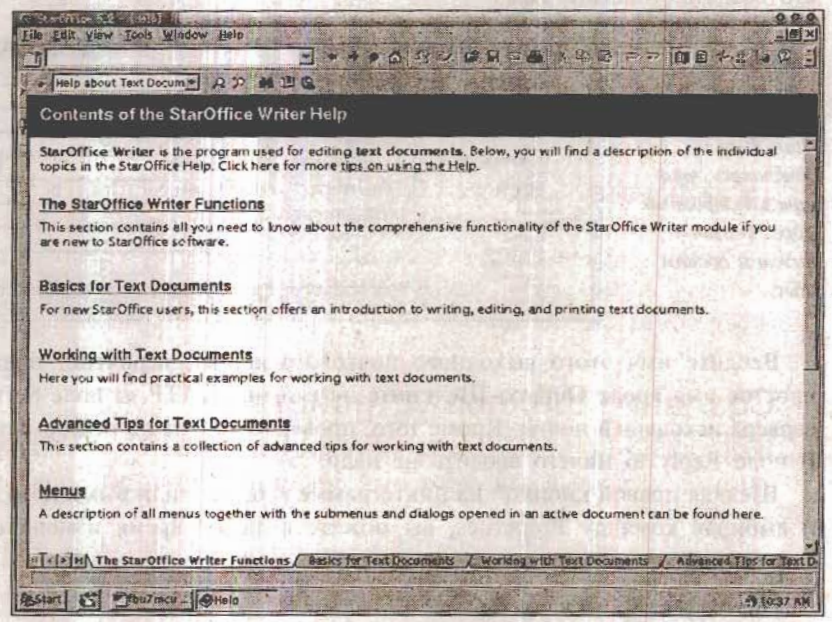


Рисунок 7.9 *Главное окно справочной системы StarOffice.*

Вкладки, названия которых видны внизу окна, позволяют быстро переходить к другим темам. Раскрывающееся меню вверху слева также позволяет выбирать разные темы. Пиктограмма с изображением бинокля служит для проведения поиска в предметном указателе справочной системы. Одна из приятных особенностей справочной системы StarOffice состоит в том, что она позволяет пользователю записывать собственные комментарии и замечания непосредственно в файл справки.

Мультимедиа

Поддержка мультимедиа, практически отсутствовавшая во FreeBSD еще несколько лет назад, к настоящему времени стала достаточно хорошей. Если у вас установлена одна из распространенных звуковых плат, то весьма вероятно, что FreeBSD поддерживает ее.

Проверка поддержки звуковой платы операционной системой FreeBSD

Ко времени написания этой книги не было точно известно, будет ли драйвер звуковой платы включен в ядро, используемое по умолчанию. Чтобы узнать, загру-

жен ли драйвер PCM, проверьте, присутствует ли в файле конфигурации ядра строка **device pcm**. Если эта строка есть, поддержка звуковой платы включена в ядро. Файл конфигурации ядра находится в каталоге **/sys/i386/conf/GENERIC**.

Введите команду **dmesg | more** и просмотрите ее выходные данные, касающиеся ядра. Найдите информацию, указывающую на то, что во время загрузки системы звуковая плата была обнаружена. Например, в моей системе выходные данные команды **dmesg** содержат следующие строки:

```
sbc0: <Creative SB AWE64> at port 0x220-0x22f,0x330-0x331,0x388-0x38b irq 5 drq
1,5 on isa0
pcm1: <SB16 DSP 4.16> on sbc0
```

Эти строки говорят о том, что FreeBSD обнаружила звуковую плату SoundBlaster AWE 64; они также содержат адреса, которые она использует.

Если FreeBSD не обнаруживает вашу звуковую карту, то можно попытаться заставить карту работать, используя драйверы компании 4Front Technologies. Их можно получить по адресу **www.opensound.com**. Если вы решите использовать драйверы компании 4Front Technologies, то вам, вероятно, придется удалить или закомментировать строку **device pcm** в файле конфигурации ядра, а затем собрать ядро снова. Полностью все инструкции по конфигурированию и сборке ядра содержатся в главе 17.

Создание файлов устройств

Во FreeBSD 4.4 для звуковых карт требуется создать соответствующий файл устройства, необходимый для работы со звуком. Для этого как **root** введите в каталоге **/dev** следующую команду:

```
./MAKEDEV snd0
```

Во FreeBSD 5.0 используется файловая система DEVFS и в этом случае файлы устройств создаются автоматически при каждой загрузке системы.

Микшер

Как только файлы устройств будут созданы, вы сможете обращаться к микшеру FreeBSD из командной строки. В окне терминала X введите команду **mixer**. На экран будут выведены строки, подобные следующим:

```
Mixer vol      is currently set to  75:75
Mixer bass     is currently set to  50:50
Mixer treble   is currently set to  50:50
Mixer synth    is currently set to  75:75
Mixer pcm      is currently set to  75:75
Mixer speaker  is currently set to  75:75
Mixer line     is currently set to  75:75
Mixer mic      is currently set to   0:0
Mixer cd       is currently set to  75:75
Mixer igain    is currently set to   0:0
Mixer ogain    is currently set to  50:50
```

Они показывают, на какие уровни установлены регуляторы сигналов (в процентах от 0 до 100) во всех звуковых устройствах системы. Понятно, что число слева от двоеточия означает уровень сигнала в левом звуковом канале, а число справа от

двоеточия — уровень сигнала в правом звуковом канале. Вот, что означают некоторые элементы вывода:

- **Mixer vol** — Это главный регулятор громкости для всех звуковых устройств.
- **Mixer pcm** — Это устройство, через которое проходит большинство звуковых сигналов. Данный регулятор громкости определяет громкость файлов **.wav**, файлов MP3, файлов Real Audio и т.д.
- **Mixer synth** — Это регулятор громкости синтезатора. Он, как правило, влияет на файлы MIDI.
- **Mixer cd** — Это регулятор громкости воспроизведения компакт-дисков.
- **Mixer line** — Это регулятор громкости устройства, подключенного к гнезду звуковой платы.
- **Mixer igain** — Это регулятор усиления входного сигнала.
- **Mixer ogain** — Это регулятор усиления выходного сигнала.

Чтобы изменить любой из этих уровней, необходимо ввести команду **mixer**, а затем — имя устройства и требуемый уровень. Так, например, чтобы установить регулятор громкости **cd** на уровень 90%, как для левого, так и для правого канала, введите:

```
mixer cd 90
```

Если для левого и правого каналов требуется установить разные значения, то необходимо ввести оба этих значения, разделенные двоеточием. Например, чтобы установить уровень 100% для левого канала и 80% для правого, введем:

```
mixer cd 100:80
```

При загрузке системы большинство регуляторов микшера по умолчанию устанавливается на уровень 75%. Добиться, чтобы при загрузке системы по умолчанию устанавливались другие уровни, можно разными способами. Можно, например, в каталоге **/usr/local/etc/rc.d** создать файл запуска. Этот файл включает команды микшера. Файлу можно дать любое имя, например **mixerset**. Вот пример информации, которую может содержать данный файл:

```
mixer vol 80:80
mixer cd 90:90
mixer pcm 50:50
```

Как пользователь **root** создайте этот файл в каталоге **/usr/local/etc/rc.d**. Затем сделайте файл исполняемым, выполнив следующую команду:

```
chmod u+x mixerset
```

Теперь команды, которые содержатся в данном файле, будут выполняться при каждой загрузке системы и устанавливать в микшере требуемые уровни громкости.

Второй способ изменения уровней, используемых по умолчанию, заключается в добавлении команд в свой файл **profile**. Если у вас командный интерпретатор типа **sh**, то команды необходимо добавлять в файл **.profile**. Если у вас командный интерпретатор типа C, то — в файл **.login**. Оба эти файла расположены в вашем home-каталоге. За более подробной информацией обратитесь к главе 12.

Все, что вам потребуется сделать, — это открыть файл **.profile** или **.login** в текстовом редакторе и добавить требуемые команды микшера. Например:

```
mixer vol 80:80
mixer cd 50:50
mixer pcm 90:90
```

В отличие от изменений в файле запуска, данные изменения может делать не только пользователь **root**, но они не действуют при загрузке системы. Эти изменения вступают в действие тогда, когда вы регистрируетесь и входите в систему. Преимущество данного метода состоит в том, что если системой пользуются несколько человек, то каждый может создать собственный набор параметров микшера в соответствии со своими предпочтениями.

Воспроизведение файлов MP3 с помощью XMMS

Во FreeBSD можно воспроизводить компакт-диски в формате MP3. Программа XMMS — это точная копия Winamp, предназначенная для FreeBSD и других систем UNIX. Она находится в каталоге **audio** коллекции портов FreeBSD.

После установки программы XMMS ее можно запускать с помощью команды **xmms**. Ее можно вводить, как в окне терминала X, так и в диалоговом окне Run. На рис. 7.10 изображено окно XMMS.

Рисунок 7.10 *Проигрыватель XMMS для файлов MP3.*

XMMS даже поддерживает скины (skins) Winamp. И их даже не требуется распаковывать для этого. Просто поместите упакованные скин-файлы Winamp в подкаталог **.xmms/Skins** своего home-каталога. Щелкните на пиктограмме в верхнем левом углу окна XMMS; затем в раскрывающемся меню выберите последовательно команды Options и Skin Browser. В результате откроется диалоговое окно, в котором можно выбирать желаемый скин.

Кроме того, на веб-сайте **www.xmms.org/skins.html** имеется большое число скинов XMMS. Как и скины Winamp, их можно просто скопировать в каталог **.xmms/Skins**, не распаковывая предварительно. Скины XMMS обычно распространяются в виде упакованных файлов **tar** с расширением **.tar.gz**. Это популярный формат архивных файлов для систем UNIX.

XMMS позволяет также подключать различные сменные модули для создания визуальных и звуковых эффектов. Многие из них находятся в каталоге **audio** коллекции портов FreeBSD.

Воспроизведение MP3-файлов с помощью mpg123

Если вы не работаете с системой X-Window, но хотите слушать музыку в формате MP3, можно воспользоваться MP3-проигрывателем, запускаемом из командной строки. Одной из лучших и наиболее популярных программ этого типа является **mpg123**. Она находится в каталоге **audio** коллекции портов FreeBSD.

В наиболее простой форме **mpg123** запускается командой **mpg123 filename**, где **filename** — это имя файла, который вы хотите послушать. Эта команда допускает символы подстановки (см. гл. 8). Например, команда **mpg123 *.mp3** будет воспроизводить все файлы текущего каталога, имеющие расширение **.mp3**. Можно также в командной строке ввести список файлов, которые вы хотите воспроизвести. Параметр **-Z** означает воспроизведение файлов в случайном порядке. Можно также использовать параметр **-@**, за которым следует имя текстового файла. В этом случае **mpg123** трактует содержимое данного текстового файла как список файлов, которые необходимо воспроизвести. Текстовый файл должен содержать список файлов MP3 — по одному файлу на каждой строке. Чтобы увидеть полный список всех параметров, введите команду **man mpg123**.

Микшеры для системы X-Window

В данном разделе рассматривается программа **xmixer**, расположенная в каталоге **audio** коллекции портов FreeBSD. Запускать программу **xmixer** можно двумя способами. Первый способ — ввести команду **xmixer** в окне терминала X или в диалоговом окне Run. В **xmixer** используется комплект инструментов Athena (набор программиста для проектирования графических интерфейсов). Второй способ запуска программы **xmixer** — ввести команду **xgmixer**. Это тот же самый **xmixer**, но в данном случае запускается версия программы, в которой используется комплект инструментов GTK.

Во FreeBSD имеется несколько других аудиоприложений, включая проигрыватели компакт-дисков, проигрыватели MP3, проигрыватели сигнала в формате Real Audio и инструментальные средства для преобразования (ripping) звуковых дорожек компак-дисков в формат MP3. Чтобы узнать, какие приложения у вас имеются, просмотрите каталог **audio** в коллекции портов FreeBSD.

Сетевые приложения

Во FreeBSD имеется много сетевых приложений. В данном разделе рассматриваются веб-браузеры, программы электронной почты, программы передачи данных по протоколу FTP и т.д.

Конфигурирование веб-браузера Netscape

В операционной системе FreeBSD имеется веб-браузер Netscape Communicator. Последняя версия, доступная во FreeBSD, — 4.76. Если вам требуется версия Netscape 6, то придется запустить во FreeBSD версию Netscape для Linux. Кроме того, версию для Linux следует запускать и в тех случаях, когда необходимо использовать такие сменные модули, как Flash и Shockwave. Эти сменные модули доступны только в версиях Netscape для Linux. Все браузеры Netscape, включая версии для Linux, находятся в каталоге **www** коллекции портов.

После установки браузера Netscape его можно запускать командой **netscape** (версию 4.76) или **netscape6** (версию 6). В данном разделе освещается версия 4.76, но версия 6 на нее похожа. Чтобы изменить параметры браузера Netscape, на панели меню щелкните на меню Edit, а затем — на команде Preferences.

Веб-браузер Lynx

Lynx — это веб-браузер, который работает в текстовом режиме, графические возможности у него отсутствуют. Его все еще можно использовать, но путешествовать по всемирной Сети с помощью Lynx становится все труднее, так как он не может работать с изображениями, фреймами и Java-апплетами. Поскольку на многих сайтах для доступа к изображениям используются ссылки, а не теги ALT, это может вызвать проблемы, которые в Lynx трудно или даже невозможно обойти. Более подробную информацию по браузеру Lynx можно получить, введя команду **man lynx**.

FTP

FTP — сокращение от *File Transfer Protocol* (*Протокол передачи файлов*). Это широко распространенный метод передачи файлов из одной вычислительной системы в другую. Чтобы начать FTP-сеанс, в командной строке или в окне терминала X введите **ftp** и имя сервера, с которым вы хотите связаться:

```
ftp ftp.freebsd.org
```

Если соединение будет успешно установлено, то на экране появятся примерно такие строки:

```
Connected to ftp.beastie.tdk.net.
220 ftp.beastie.tdk.net FTP server (Version DG-4.1.73 983302105) ready.
Name (ftp.freebsd.org:murban):
```

В строке приглашения **Name** необходимо ввести свое регистрационное имя для сервера FTP. Если оно такое же, как и регистрационное имя в системе, то просто нажмите Enter. На открытых серверах FTP можно зарегистрироваться под именем **anonymous**. Когда в качестве имени пользователя вы введете **anonymous** и нажмете Enter, удаленный хост ответит вам:

```
331 Guest login ok, send your e-mail address as password.
Password:
```

Введите в качестве пароля свой адрес электронной почты. После регистрации вы можете получить приветственное сообщение и что-то вроде следующего:

```
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Для перемещения по FTP-серверу в командной строке этого приглашения можно вводить большинство команд, которые используются в командной строке локального командного интерпретатора (см. главу 8). В таблице 7.10 приведен список команд, наиболее широко применяемых в FTP-сеансе.

Таблица 7.10 Широко применяемые команды FTP

Команда	*Действие*
ls	Выдает содержание каталога на удаленном хосте.
cd	Изменяет каталог на удаленном хосте.
pwd	Отображает текущий каталог, в котором вы находитесь на удаленном хосте.

Команда	*Действие*
lcd	Изменяет каталог на локальном хосте.
binary	Осуществляет передачу данных в двоичном режиме (этот режим должен использоваться для всех видов данных, за исключением простого текста).
ascii	Осуществляет передачу данных в режиме ASCII (этот режим должен использоваться ТОЛЬКО для простого текста).
put filename	Копирует **filename** на удаленный хост. Если путь к **filename** не задан, то предполагается, что он находится в текущем каталоге. Если файл назначения не задан, то копируемый файл помещается в текущий каталог удаленного хоста и имеет то же имя, что и локальный файл (то есть **filename**). (Эта команда действует только тогда, когда у вас есть разрешение производить запись в каталог удаленной машины.)
mput file1 file2	Копирует на удаленный хост несколько файлов, перечисленных в команде. Файлы помещаются в текущий каталог удаленного хоста. (Предполагается, что у вас есть разрешение производить запись в каталог удаленной машины.)
get filename	Копирует файл с удаленного хоста в локальную вычислительную систему. Если путь не задан, то предполагается, что файл находится в текущем каталоге удаленного хоста. Если файл назначения не задан, то данный файл копируется в локальную систему с тем же самым именем (то есть **filename**).
mget file1 file2	Копирует с удаленного хоста несколько файлов, перечисленных в команде. Файлы помещаются в текущий каталог локальной системы.
mkdir dirname	Создает на удаленной машине каталог с именем **dirname** (при условии, что у вас есть на это право).
rmdir dirname or rm dirname	На удаленной машине удаляет каталог с именем **dirname** (при условии, что у вас есть на это право).
del filename	На удаленной машине удаляет файл с именем **filename** (при условии, что у вас есть на это право).
bye or quit	Закрывает соединение с удаленным хостом и осуществляет выход из программы FTP, возвращая вас в командную строку.

Чтобы получить список доступных команд, в командной строке с приглашением **ftp>** можно ввести команду **help**. Чтобы получить краткое описание какой-нибудь команды, введите **help** и имя команды. Например:

```
ftp> help
Commands may be abbreviated.  Commands are:

!               disconnect      mdelete         preserve        runique
$               edit            mdir            progress        send
account         exit            mget            prompt          sendport
append          form            mkdir           proxy           site
ascii           ftp             mls             put             size
bell            get             mode            pwd             status
binary          gate            modtime         quit            struct
bye             glob            more            quote           sunique
case            hash            mput            recv            system
cd              help            msend           reget           tenex
cdup            idle            newer           rename          trace
```

```
chmod          image          nlist          reset          type
close          lcd            nmap           restart        umask
cr             less           ntrans         restrict       user
debug          lpwd           open           rhelp          verbose
delete         ls             page           rmdir          ?
dir            macdef         passive        rstatus
ftp> help mdir
mdir                list contents of multiple remote directories
ftp>
```

За более подробной информацией обратитесь к справочной странице **man ftp**. Если вы предпочитаете работать с графической программой-клиентом FTP в системе X-Window, то в каталоге **ftp** коллекции портов имеется несколько таких программ-клиентов.

Приложения электронной почты

Приложения электронной почты — одни из первых приложений сети Internet. FreeBSD не испытывает недостатка в программах-клиентах электронной почты. Имеются как клиенты для командного интерпретатора, так и графические клиенты для системы X-Window.

Программа Balsa

Balsa — это графический почтовый клиент для среды Gnome (находится в каталоге **mail** коллекции портов FreeBSD). Если Balsa установлена, ее можно вызвать командой **balsa**. Когда вы запускаете ее первый раз, она выводит на экран ряд вопросов, чтобы правильно настроиться на получение и отправку почты. За более подробной информацией по конфигурированию и применению программы Balsa обратитесь к веб-сайту **www.balsa.net**.

Программа Pine

Pine — это программа-клиент электронной почты, работающая в текстовом режиме с простым интерфейсом. Управляемая с помощью меню, она отличается простотой и интуитивно понятна для среднего пользователя. Pine находится в каталоге **mail** коллекции портов FreeBSD. На рис. 7.11 изображено главное меню программы Pine.

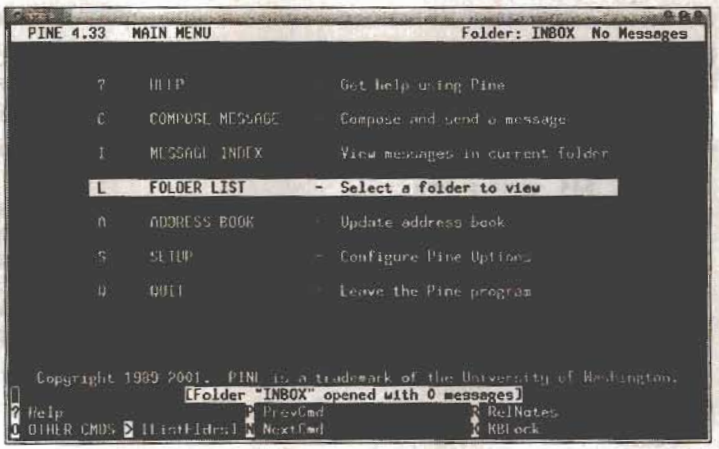


Рисунок 7.11
Главное меню программы Pine.

Программа Pine очень популярна среди конечных пользователей. Поскольку она работает в текстовом режиме, с ее помощью можно проверять электронную почту с удаленного неинтеллектуального терминала. К сожалению, у Pine репутация программы, написанной плохо и имеющей множество проблем, связанных с безопасностью. Хотя все известные на настоящий момент ошибки исправлены, многие специалисты по проблемам безопасности считают, что в защите Pine имеется еще много необнаруженных дыр.

ПРЕДОСТЕРЕЖЕНИЕ

Открывать любую электронную почту, будучи зарегистрированным как **root**, — плохая идея, а в случае Pine — особенно. Это связано с природой некоторых потенциальных брешей в защите Pine. Почту пользователя **root** следует пересылать на имя простого пользователя, а затем читать электронную почту, используя учетную запись этого пользователя. Подробнее о работе с электронной почтой можно почитать в главе 25.

Программа Mutt

Mutt — это еще одна программа-клиент электронной почты, работающая в текстовом режиме. Она находится в каталоге **mail** коллекции портов FreeBSD. Mutt — распространенная собачья кличка. Такое имя программе дали потому, что она представляет собой своего рода "дворняжку", то есть помесь двух почтовых программ-клиентов. В Mutt попытались объединить преимущества программ Elm (очень старого клиента электронной почты) и Pine (клиента, более простого в использовании, чем Elm). Многие пользователи Mutt согласятся, что эти цели были успешно достигнуты.

Mutt — это очень сложная программа с широким набором возможностей. Ее конфигурирование и использование выходят за рамки настоящей книги. Веб-сайт программы Mutt — **www.mutt.org** — предлагает очень хорошую онлайновую документацию, в которой рассказано, как конфигурировать эту мощную программу и как с ней работать.

Программы uuencode и uudecode

Теперь несколько слов о программах **uuencode** и **uudecode**. Фактически, **uuencode** — это метод пересылки двоичных файлов в виде простого текста. Часто клиент электронной почты автоматически кодирует и декодирует такие присоединенные файлы. Однако так бывает не всегда. Иногда вы можете получить электронную почту, содержащую присоединенный файл, который, как вам покажется, представляет собой несколько сотен строк текстового мусора. Он будет выглядеть примерно так:

```
begin 644 gunzip.exe
M35HZ`4D`````.I6___/6H`````.`,4('````$Q:.3'_\ 8X&W<F_S9(`/M&
M`'_XN&\ #'EX9+[F@(%FO__!X/$#,N058OL@^P$N0'B7^,1OR,7OZX!`?=
M#B;_^\ 7X#A3HQ?@*%`[HH?_K'\ (AD,1>_(/&!-MW<.,_#>X(-GX#@S@)O]W
MBP<FT<==2+Y5W+D/V4`U-KD/U8SV*0^!8HD/@/E/@1=*<6E/@?N'C>@J/X
M$8BC^`\ H]7#NGONEY/@.$+C>ML'N^`X(N,_N^!#A[O@OZ^[X$/)=_^[X$/CN
M^!`#.[X#E[UE=6QT;Z$!\ `107?C48(%L-'^P;[FE(#QP4&,,@WBC/B&)
```

Строка **begin** содержит имя файла, в котором будет сохранена данная информация после ее декодирования **uudecode**. Чтобы декодировать присоединенный файл,

сохраните его (или сообщение электронной почты, которое содержит весь этот текстовый мусор) в каталоге, где будете выполнять декодирование. Предположим, что вы сохранили файл под именем **program.txt**. Теперь для декодирования файла наберите следующую команду:

```
uudecode program.txt
```

В результате файл будет декодирован и его содержимое запишется в файл, который будет иметь имя, указанное в строке **begin** закодированного файла. В нашем случае декодированная информация запишется в выходной файл с именем **gunzip.exe**.

Если вам необходимо закодировать файл, то это также легко сделать. Предположим, что требуется закодировать файл, который называется **gunzip.exe**.

```
uuencode gunzip.exe gunzip.exe > gunzip.txt
```

В этой команде три аргумента представляют собой: имя файла, который требуется закодировать (**gunzip.exe**); имя выходного файла, который будет создан при декодировании закодированного файла (обычно такое же самое — **gunzip.exe**); выходные данные перенаправляются в файл **gunzip.txt**, который будет содержать закодированный текст. Если мы этого не сделаем, то выходные данные будут выведены на экран. Перенаправление позволяет послать выходные данные непосредственно в почтовую программу, если вы хотите отправить кому-нибудь закодированный файл.

8 ГЛАВА

Работа с командным интерпретатором

- Основы работы с командным интерпретатором
- Типы доступных командных интерпретаторов
- Выбор интерпретатора
- Получение справки
- Простейшие способы работы с файлами в командном интерпретаторе
- Команды обработки текстовых файлов
- Конвейеры и перенаправление ввода-вывода
- Дополнение командной строки и редактирование истории команд

До этого момента мы имели дело в основном с графическим интерфейсом системы X-Window и менеджером окон Gnome. Хотя это достаточно простой способ работы с системой, воспользоваться реальными возможностями FreeBSD позволяет только командный интерпретатор. Философия разработки, стоящая за командной строкой и интерпретаторами UNIX, — это то, что делает эту систему одной из самых мощных, хотя ее исходная версия была разработана более 30 лет назад. В этой главе рассказано о нескольких командных интерпретаторах, включенных в состав FreeBSD, приведено сравнение их возможностей и продемонстрированы основные принципы работы с интерпретатором.

Основы работы с командным интерпретатором

Командный интерпретатор UNIX представляет собой интерфейс к UNIX. Фактически это программный уровень, который предоставляет среду для ввода команд, обеспечивая тем самым взаимодействие между пользователем и ядром операционной системы. Если подходить к этому вопросу в более широком смысле, то рабочий стол Windows или Macintosh также можно рассматривать как интерпретатор (оболочку). Фактически, командный интерпретатор преобразует язык, понятный человеку, в команды, понятные ядру. Взаимосвязь между оборудованием системы, ядром и интерпретатором показана на рис. 8.1.

Рисунок 8.1 *Взаимосвязь между оборудованием, ядром и интерпретатором (оболочкой).*

О роли ядра подробно рассказано в главе 17. *Ядро* представляет собой специальный элемент программного обеспечения, управляющий взаимодействием других программных компонентов с аппаратными средствами компьютера. Обычно, пользователю задумываться о работе ядра не приходится, поскольку оно выполняет свою работу в фоновом режиме.

Типы доступных командных интерпретаторов

В состав FreeBSD включено несколько командных интерпретаторов, начиная от простейших и заканчивая самыми изощренными. Здесь рассказывается не обо всех, а лишь о наиболее популярных из них. Начнем с одного из самых ранних интерпретаторов, который до сих пор используется повсеместно: это командный интерпретатор Bourne, широко известный как **sh**.

Командный интерпретатор Bourne (sh)

Исходная версия командного интерпретатора Bourne была разработана в Bell Labs Стивеном Борном (Steven Bourne) для операционной системы AT&T UNIX. Во FreeBSD, как и во многих других версиях UNIX, этот интерпретатор заменен команд-

ным интерпретатором, совместимым со стандартом POSIX, который представляет собой улучшенную версию Bourne и поддерживает такие функции, как, например, редактирование командной строки, историю команд и псевдонимы — элементы, отсутствующие в исходной версии Bourne.

Командный интерпретатор C (csh)

Командный интерпретатор **csh** — традиционный интерпретатор BSD. Разработчики в Berkeley, столкнувшись со всеми ограничениями исходной версии Bourne, создали собственный вариант командного интерпретатора для операционной системы BSD. Интерпретатор C получил такое название потому, что его синтаксис очень похож на язык программирования C. **csh** имеет немало преимуществ перед Bourne, как-то: управление заданиями и история команд. Кроме того, в нем имеется конфигурационный файл выхода из системы, чего нет в интерпретаторе Bourne. **csh** также поддерживает конфигурационный файл, чтение которого выполняется при запуске каждой новой копии интерпретатора, не являющейся начальной. Исходная версия Bourne таких свойств не имеет.

csh, в определенном смысле, является полной противоположностью Bourne: он удобен при интерактивном использовании, но очень сложен в программировании. Некоторые программисты, тем не менее, предпочитают **csh**, однако его синтаксис более запутан и предъявляет больше требований к пользователю.

Хотя **csh** удобен для интерактивного использования и в свое время был очень популярен, сегодня он несколько устарел. В наши дни появились командные интерпретаторы, еще более удобные для работы в таком режиме.

Командный интерпретатор Korn (ksh или pdksh)

Компания AT&T выпустила командный интерпретатор Korn в 1986 году. Его автором является Дэвид Корн (David Korn). Этот интерпретатор был ответом AT&T на **csh**. Интерпретатор Korn совместим снизу вверх с Bourne. Он, как и интерпретатор C, поддерживает управление заданиями, историю команд, псевдонимы и конфигурационный файл, предназначенный для последующих процессов интерпретатора. Кроме того, Korn обеспечивает редактирование истории команд в стиле **vi** и Emacs (т.е. возможность отыскать одну из предыдущих команд и отредактировать ее перед повторным запуском). Этот интерпретатор имеет несколько свойств, полезных при программировании в нем, например, расширяемость новыми командами и совместимость синтаксиса со многими другими интерпретаторами. Интерпретатор Korn весьма популярен и входит в состав многих коммерческих версий UNIX.

Существует и общедоступная версия интерпретатора Korn: **pdksh**. Ее можно установить и во FreeBSD, воспользовавшись набором портов, о котором рассказано в главе 15.

Пользователям, хорошо знакомым с редакторами **vi** и Emacs, интерпретатор Korn понравится, но те, кто лишь начинает работать с UNIX, скорее предпочтут командный интерпретатор **bash** (о нем рассказано далее), поскольку команды редактирования **vi** и Emacs не всегда интуитивно понятны (однако впоследствии редактирование становится более эффективным).

Командный интерпретатор Bourne Again Shell (bash)

Командный интерпретатор Bourne Again Shell (игра слов: Bourne Again означает снова Bourne и созвучно с Born Again — заново родившийся) представляет собой совместимый с Bourne интерпретатор, разработанный фондом свободно распространяемого программного обеспечения (Free Software Foundation) (FSF). Интерпретатор **bash** используется по умолчанию практически во всех дистрибутивах Linux. Он похож на Korn, однако содержит больше свойств, включая встроенную систему справки, интуитивно понятное редактирование командной строки, в котором используются клавиши курсора, чрезвычайно мощные функции истории, а также большой набор переменных среды.

Однако **bash** не является стандартом для многих коммерческих версий UNIX. Поэтому программы, написанные для **bash**, нельзя портировать во многие версии UNIX.

Командный интерпретатор Tcsh (tcsh)

Командный интерпретатор **tcsh** представляет собой улучшенную версию интерпретатора C. (Буква "t" напоминает об операционной системе TENEX для DEC PDP-10; в исходном варианте **tcsh** должен был имитировать поведение командной строки этой ОС.) В **tcsh** добавлены новые свойства, включая дополнение имен файлов и редактирование командной строки с помощью интуитивных средств управления, похожих на средства **bash** (также используются клавиши курсора). Кроме того, **tcsh** имеет несколько удобных свойств, не поддерживаемых в **bash**, включая распознавание потенциально опасных команд (например: "Are you sure you want to delete ALL files?").

tcsh — это командный интерпретатор, очень удобный для интерактивного использования. Во многом он даже лучше, чем **bash**. К сожалению, в том, что касается программирования, у него не намного меньше проблем, чем у исходного интерпретатора C. Хотя некоторые из них и были устранены, желательно создавать сценарии в стиле Bourne, избегая программирования в командном интерпретаторе C.

Какой интерпретатор выбрать?

Итак, какой же интерпретатор выбрать? Большинство стандартных команд FreeBSD работает одинаково во всех интерпретаторах. Различия играют важную роль только тогда, когда дело доходит до таких задач, как изменение переменных интерпретатора, установка псевдонимов и создание сценариев.

Если вы не планируете создавать программы на языке интерпретатора, воспользуйтесь Korn, **bash** или **tcsh**. В ином случае отдайте предпочтение Korn или **bash**, поскольку все, чему вы научитесь, будет пригодно и для создания сценариев.

Изменение интерпретатора

В основном мы будем говорить о командном интерпретаторе **bash**, поскольку этот интерпретатор наиболее прост для изучения. Чтобы узнать о том, какой командный интерпретатор используется в данный момент, введите следующую команду (где

foo — имя, с которым вы регистрируетесь в системе, а $ — приглашение командной строки):

```
$ grep foo /etc/passwd
```

Вам не ясно назначение этой команды? Оно станет понятным еще до того, как вы закончите чтение этой главы. Команда возвратит строку следующего вида:

```
foo:*:1001:1001::Foo Bar:/home/foo:/usr/local/bin/ksh
```

Строка на вашем экране может быть несколько другой. Сейчас нас интересует только последнее поле (в примере в нем содержится строка **/usr/local/bin/ksh**). В нем указан командный интерпретатор, используемый по умолчанию, в данном случае Korn. Это поле может иметь одно из следующих значений:

- **/bin/sh**. Командный интерпретатор Bourne, поддерживающий стандарт POSIX.
- **/bin/csh**. Командный интерпретатор C, который в настоящее время заменен интерпретатором **tcsh**. Поэтому **/bin/csh** на самом деле отвечает интерпретатору **tcsh**.
- **/bin/tcsh**. Командный интерпретатор **tcsh**.
- **/usr/local/bin/ksh**. Командный интерпретатор Korn.
- **/usr/local/bin/bash**. Командный интерпретатор **bash**.

Если интерпретатор **bash** не запущен по умолчанию, просто введите **bash** в командной строке. В главе 12 будет рассказано о том, как изменить командный интерпретатор пользователя по умолчанию.

ПРЕДУПРЕЖДЕНИЕ

UNIX и FreeBSD учитывают регистр в командах и именах файлов. Поэтому **Grep**, **grep** и **greP** — это с точки зрения FreeBSD совершенно разные вещи. Если вы, вводя команды, получаете сообщение об ошибке: **No such file or directory** (Нет такого файла или каталога), убедитесь, что команда введена в правильном регистре.

Получение справки

Разумеется, в одной главе невозможно рассказать обо всех командах, поддерживаемых интерпретатором. Во FreeBSD включен большой набор справочных руководств, в которых задокументированы все команды и их возможные опции. Для доступа к странице справочного руководства используется команда **man имя_команды**. Например, чтобы прочесть справку о самой команде **man**, введите команду **man man**. Для просмотра содержимого страниц воспользуйтесь курсорными клавишами, пробелом и клавишами Page Up/Page Down.

Поиск страниц справочных руководств

Иногда известно, что делает команда, но ее название почему-то не вспоминается. В этом случае можно воспользоваться опцией **-k** команды **man**. Предположим, что вы хотите узнать о командах поиска в текстовых файлах определенных слов или фраз и вывода тех участков файла, где они содержатся. Вначале нужно запустить команду **man -k search**. Ее вывод приведен на листинге 8.1.

Листинг 8.1 Использование команды man -k

```
bios(9), bios_sigsearch(9), bios32_SDlookup(9), bios32(9) - Interact
with PC BIO
S
bsearch(3)              - binary search of a sorted table
devclass_find(9)        - search for a devclass
device_find_child(9)    - search for a child of a device
lkbib(1)                - search bibliographic databases
lookbib(1)              - search bibliographic databases
lsearch(3), lfind(3)    - linear searching routines
manpath(1)              - determine user's search path for man pages
res_query(3), res_search(3), res_mkquery(3), res_send(3), res_init(3),
dn_comp(3
), dn_expand(3) - resolver routines
tsearch(3), tfind(3), tdelete(3), twalk(3) - manipulate binary search
trees
Search::Dict(3), look(3) - search for key in dictionary file
bash$
```

Однако это немного не то, что вы искали. Результаты относятся к процедурам поиска в языке программирования С. При следующей попытке используйте слово "pattern" (шаблон) (как показано в листинге 8.2), поскольку вам требуются команды поиска шаблонов в текстовых файлах.

Листинг 8.2 Следующая попытка применения команды man -k

```
$ man -k pattern
gawk(1) - pattern scanning and processing language
glob(3), globfree(3)        - generate pathnames matching a pattern
grep(1), egrep(1), fgrep(1), zgrep(1) - print lines matching a pattern
lptest(1)                   - generate lineprinter ripple pattern
menu_pattern(3)             - get and set a menu's pattern buffer
bash$
```

Теперь вывод выглядит более полезным, особенно набор команд **grep**. Этот набор команд представляет собой средства для поиска шаблонов в текстовых файлах, что и требовалось найти. Теперь, применив команду **man grep**, можно прочесть справочное руководство по **grep** и узнать, на что способна эта команда.

Резюме команд

В других случаях известно имя команды, но неизвестно, какие функции она выполняет (набор команд можно увидеть, например, в каталоге **/usr/bin** или **/usr/games**). Тогда можно воспользоваться опцией **-f** команды **man** для получения краткого (в одну строку) описания команды. Предположим, например, что вы нашли команду **pom** в каталоге **/usr/games** и хотите узнать, для чего она предназначена:

```
$ man -f pom
pom(6) - display the phase of the moon
bash$
```

Обратите внимание *на цифру (6) после имени команды:* она указывает номер раздела справочных руководств, к которому принадлежит команда.

Разделы справочных руководств

Справочные руководства разбиты на девять разделов, описание которых приведено в табл. 8.1.

Таблица 8.1 Разделы справочных руководств

Раздел	*Описание*
1: User Commands (Пользовательские команды)	Информация о таких пользовательских командах, как **ls**, **rm**, **cp** и **grep**. К этому разделу вам придется обращаться наиболее часто при работе во FreeBSD.
2: System Calls (Системные вызовы)	Содержит информацию о различных API FreeBSD. Этот раздел в первую очередь предназначен для программистов на C.
3: Subroutines (Процедуры)	Здесь содержатся описания библиотечных функций. Они также предназначены для программистов на C.
4: Devices (Устройства)	Информация для программистов на C об интерфейсах к драйверам устройств во FreeBSD.
5: File Formats (Форматы файлов)	Информация о форматах различных конфигурационных файлах системы, например, файлах **crontab** для запуска заданий по графику и файлах **rc**, управляющих запуском системы.
6: Games (Игры)	Наиболее важный раздел справочных руководств. Здесь содержатся инструкции к играм и другим развлекательным средствам FreeBSD.
7: Miscellaneous (Разное)	То, что не подходит для других разделов, размещается здесь.
8: System Administration (Системное администрирование)	Содержит информацию о командах, предназначенных для администрирования системы, например, **fsck** (команда для проверки файловых систем).
9: Kernel Interfaces (Интерфейсы ядра)	Информация для программистов об интерфейсе ядра на C.

Некоторые команды имеют более одной записи. Например, запись **crontab** представлена в разделе 1 (команда **crontab**) и в разделе 5 (формат файла **crontab**). По умолчанию команда **man** выводит самую первую запись, т.е. для **crontab** будет выведено руководство из раздела 1. Если же вы хотите обратиться к руководству из раздела 5, воспользуйтесь командой **man 5 crontab**. Чтобы осуществить поиск в определенных разделах справочных руководств, необходимо указать опцию **-S** с последующим списком их номеров, разделенным двоеточием.

Простейшие способы работы с файлами в командном интерпретаторе

Если вам доводилось работать с MS-DOS, большая часть информации из этой главы будет даваться вам легко. В ином случае вам потребуются некоторые вспомогательные сведения.

Как система FreeBSD хранит файлы

Как и большинство современных операционных систем, FreeBSD сохраняет файлы в иерархической древовидной структуре. Файловую систему FreeBSD можно рассматривать как систему шкафчиков с документацией. Жесткий диск можно рассматривать как группу таких шкафчиков. Файловая система FreeBSD организована в виде нескольких каталогов. Каталоги представляют собой индивидуальные шкафчики в

группе. Каталоги содержат файлы и другие каталоги. Последние аналогичны папкам и бумагам в шкафчике. Такой же тип файловой системы используется в системах Windows и Macintosh. Единственная разница заключается в том, что в Windows и Macintosh используется графическое представление папок и файлов в графическом диспетчере файлов, а в консоли FreeBSD — текстовое.

В Windows или DOS при регистрации пользователь, как правило, попадает в корневой каталог. Во FreeBSD дело обстоит иначе. После регистрации пользователь попадает в свой начальный каталог, который расположен на несколько уровней ниже корневого каталога. Структура файловой системы подробно рассматривается в главе 9. Здесь мы ограничимся рассмотрением начального каталога.

Начальный каталог

Каждый пользователь в системе UNIX (за некоторым исключением) имеет собственный начальный каталог (home-каталог). Последний представляет собой нечто вроде личного "виртуального жесткого диска". В своем каталоге можно удалять, копировать, перемещать файлы, создавать и удалять каталоги. Начальный каталог это место в файловой системе FreeBSD, предназначенное для пользователя. Следующие несколько разделов посвящены командам для работы с файлами и каталогами.

Вывод содержимого каталога

Для вывода содержимого каталога используется команда **ls**. Сама по себе без опций она выводит, например, следующее:

```
$ ls
Mail      letter-to-boss    program        proposal-draft
fbu       mail              program.c
```

Однако этот вывод ничего не говорит о содержимом файлов и не позволяет отличить файлы от каталогов. **ls** поддерживает много опций, изменяющих поведение этой команды. Одной из наиболее полезных опций является **-F**, которая выводит информацию о типе каждого файла в списке. Например:

```
$ ls -F
Mail/       letter-to-boss      program*       proposal-draft
fbu@        mail/               program.c
```

В таком формате команда позволяет узнать об элементах списка более подробно. Вот значение каждого символа:

- /: Элемент является подкаталогом текущего каталога.
- *: Элемент является выполняемым файлом или сценарием.
- @: Элемент является ссылкой на другой элемент файловой системы на жестком диске.

Обратите внимание на выполняемый файл "**program**". В отличие от Windows, FreeBSD не требует, чтобы выполняемые файлы имели расширение **.com**, **.exe** или **.bat**. На то, что файл является выполняемым, указывает бит прав доступа, который хранится в индексном дескрипторе файла. Во FreeBSD вам никогда не встретится выполняемый файл с расширением **.exe** или **.com**.

В списке представлены далеко не все символы. Чтобы ознакомиться с ними, найдите описание опции **-F** в справочном руководстве **man ls**.

Современные версии команды **ls** поддерживают опцию **-G**, которая позволяет выводить файлы разного типа различным цветом.

По умолчанию команда **ls** не выводит скрытые файлы, называемые еще *dot-файлы*, т.е. файлы, имя которых начинается с точки. Для их отображения используется опция **-a**:

```
$ ls -a
.               .forward        .mailrc         ..profile       Mail
..              .hushlogin      .mysql_history  .project        fbu
.addressbook    .login          .rhosts         .login.conf     mail
.muttrc         .sh-history     .cshrc          .mail_aliases
```

Большинство скрытых файлов являются конфигурационными. В них может храниться информация о настройках командного интерпретатора, адресная книга почтовой программы или история команд интерпретатора. Два основных преимущества того, что эти файлы скрыты: они не загромождают листинг каталога и их нельзя случайно удалить, поскольку dot-файлы не удовлетворяют шаблонам с символами-заместителями.

Получить более подробную информацию о файлах и каталогах позволяет опция **-l** (см. листинг 8.3):

Листинг 8.3 Пример вывода команды ls -l

```
$ ls -l
total 3
drwx------  2 murban murban    512 Feb 15 16:04 Mail
lrwxr-xr-x  1 murban murban     15 Mar 20 06:55 fbu ->
/home/murban/documents/books/fbu
-rw-r--r--  1 murban murban    782 Mar 15 09:21 letter-to-boss
drwx------  2 murban murban    512 Mar 24 15:12 mail
-rwx------  1 murban murban  15221 Feb 21 18:11 program
-rw-r--r--  1 murban murban   1571 Feb 21 17:51 program.c
-rw-r--r--  1 murban murban   2521 Feb 25 18:51 proposal-draft
```

Каждая запись содержит семь информационных полей. Вот назначение каждого из них (слева направо):

- **Права доступа и другие атрибуты файла**. Первое поле в списке **ls -l** содержит права доступа и атрибуты файла. О значениях в этом поле подробно рассказано в главе 10. В этом поле указано, кто имеет права на чтение, запись и выполнение каждого файла (или каталога).
- **Количество ссылок**. В этом поле содержится количество ссылок, существующих для данного файла. Обычные файлы содержат не менее одной ссылки (на себя), а каталоги — не менее двух (на себя и на родительский каталог).
- **Имя пользователя-владельца файла**. В этом поле содержится информация о том, какому пользователю и какой группе принадлежит файл.
- **Имя группы-владельца файла**. См. предыдущее поле.
- **Размер файла**. Размер файла в байтах.
- **Дата и время последней модификации файла**. Время, когда файл был в последний раз изменен.

- **Имя файла**. В этом поле указано имя файла. Ссылки выглядят следующим образом:

```
fbu -> /home/murban/documents/books/fbu
```

В данном случае элемент листинга **fbu** представляет собой ссылку на **/home/murban/documents/books/fbu**. Ссылка используется для доступа к каталогу, чтобы не вводить полный путь к нему.

Перемещение по файловой системе

Для изменения текущего каталога используется команда **cd**. Имя каталога можно указать в различном формате, как показано в табл. 8.2.

Таблица 8.2 Использование команды cd

Команда	*Результат*
cd	Делает текущим начальный каталог пользователя.
cd /	Путь к каталогу, начинающийся с символа /, называется "абсолютным". Он начинается с корневого каталога системы (который представляет собой каталог самого верхнего уровня). Данная команда перемещает пользователя в корневой каталог.
cd /usr/local/bin	Перемещает пользователя в каталог "**bin**", расположенный в каталоге "**local**", который в свою очередь расположен в "**usr**", а последний — в корневом каталоге.
cd bin	Перемещает пользователя в каталог "**bin**" (если таковой существует), расположенный непосредственно в текущем каталоге. Если такого каталога нет, **cd** возвращает ошибку. Имена каталогов или файлов, которые не начинаются с /, рассматриваются относительно текущего рабочего каталога.
cd ../	Специальная запись "**../**" обозначает родительский каталог по отношению к текущему. Эта команда перемещает пользователя в каталог на один уровень вверх. Замыкающий символ косой черты указывать не обязательно.
cd ../bin	Перемещает пользователя в каталог "**bin**", расположенный в родительском каталоге (по отношению к текущему).
cd ../../bin	Перемещает пользователя в каталог "**bin**", расположенный в каталоге на два уровня выше текущего.

Можно запутаться и заблудиться...

Иногда легко забыть, в какой месте иерархии файловой системы вы в данный момент находитесь. Команда **pwd** выводит имя текущего каталога. (Кроме того, приглашение интерпретатора можно настроить таким образом, что в нем будет указан текущий каталог. Об этом рассказано в главе 12.)

Копирование файлов и каталогов

Для копирования файлов применяется команда **cp**. Ее аргументом является список файлов, за которым следует местоположение (файл или каталог), куда их следует скопировать. Если указано более одного файла или используются символы-замести-

тели, точкой назначений должен быть каталог. Несколько примеров приведено в табл. 8.3.

Таблица 8.3 Использование команды cp

Команда	*Результат*
cp file1 file2	Скопировать существующий **file1** в новый файл с именем **file2**.
cp file1 /archive	Скопировать **file1** в каталог **"archive"**, который находится в корневом каталоге.
cp file1 mystuff/newfile	Скопировать **file1** в каталог **mystuff** в текущем каталоге. Если **"newfile"** — подкаталог, **file1** копируется в подкаталог **"newfile"** со своим прежним именем. Если **"newfile"** — это не подкаталог, **file1** копируется в подкаталог **"mystuff"** с новым именем **"newfile"**.
cp file1 file 2 /archive	Скопировать **file1** и **file2** в каталог **"archive"**, расположенный в корневом каталоге.

ПРЕДУПРЕЖДЕНИЕ

По умолчанию команда **cp** перезаписывает содержимое существующего файла с именем, указанным как точка назначения. Например, **cp file1 file2** перезаписывает **file2** файлом **file1**, если **file2** уже существует. Опция **–i** (интерактивный режим) заставляет команду **cp** запрашивать подтверждение на перезапись файла.

Если нужно рекурсивно скопировать каталог и все, что в нем находится, включая и подкаталоги, следует воспользоваться опцией **-R**. Например:

```
$ cp -R dir1 /dir2
```

Эта команда копирует все, что находится в каталоге **dir1**, в новый каталог с именем **dir2**. Если **dir2** уже существует, в нем создается новый подкаталог **dir1** и все содержимое размещается в последнем. Если **dir1** уже существует в **dir2**, содержимое подкаталога **dir1** в текущем каталоге добавляется к содержимому подкаталога **dir1** в **dir2**. Файлы в каталоге назначения, имеющие те же имена, что и копируемые, перезаписываются новым содержимым, поэтому опцию **-R** нужно применять с осторожностью (или в сочетании с **-i**, в этом случае утилита выдает предупреждение до того, как перезаписать файл).

ПРЕДУПРЕЖДЕНИЕ

Будьте осторожны с командной строкой и следите за тем, чтобы не запустить, например, такую команду:

```
$ cp -R /* /old
```

Она рекурсивно копирует все содержимое корневого каталога в подкаталог **old**, который также содержится в корневом. Это значит, что все содержимое **old** копируется в подкаталог внутри него, также имеющий имя **old**. В результате возникает бесконечный цикл, копирующий содержимое каталога **old**. На достаточно быстром жестком диске выполнение такой команды очень скоро поглотит все свободное пространство.

Перемещение и переименование файлов и каталогов

Для перемещения или переименования файлов и каталогов используется команда **mv**. Несколько примеров приведено в табл. 8.4.

Таблица 8.4 Использование команды mv

Команда	*Результат*
mv file1 file2	Переименовывает **file1** в **file2**.
mv /dir1 /dir2	Переименовывает каталог **dir1** в **dir2**. Если каталог **dir2** уже существует и не является пустым, утилита выдает сообщение об ошибке.
mv file1 /dir2	Перемещает **file1** в каталог **dir2**, расположенный в корневом каталоге.
mv file1 /dir2/file2	Если **file2** является каталогом, **file1** перемещается в **/dir2/file2**. Если **file2** не существует, **file1** перемещается в каталог **dir2** под именем **file2**.

Удаление файлов и каталогов

Для удаления файлов используется команда **rm**. Запущенная без опций, команда **rm** удаляет файлы, список которых задан в командной строке. Каталоги при этом не удаляются. Если необходимо удалить каталог и все, что в нем находится, применяется опция **-R**. Несколько примеров использования этой команды приведено в табл. 8.5.

Таблица 8.5 Использование команды rm

Команда	*Результат*
rm file1	Удаляет **file1**.
rm file1 file2	Удаляет файлы **file1** и **file2**.
rm -R dir1	Удаляет каталог **dir1** и все, что в нем находится.

Список некоторых (но не всех) опций, поддерживаемых командой **rm**, приведен в табл. 8.6:

Таблица 8.6 Опции команды rm

Опция	*Результат*
-f	Заставляет **rm** произвести удаление, не задавая вопросов, даже если файл предназначен только для чтения. Эту опцию следует использовать с особой осторожностью.
-i	Запрашивает подтверждение на удаление каждого файла. Полезно при использовании **rm** с символами-заместителями (о них далее) или опцией **-R**.
-P	Три раза перезаписывает содержимое файла случайной последовательностью байтов для его полного физического удаления. Таким образом, уменьшается шанс восстановления файла. Эту опцию используют для удаления особо важных файлов.
-W	Пытается восстановить файл, удаленный командой **rm**.

Удаление каталогов

Для удаления каталогов используется команда **rmdir**. Однако она способна удалить лишь пустой каталог. Запущенная с опцией **-p**, эта команда удаляет каталог и все подкаталоги в нем, если все они также являются пустыми. Если требуется удалить каталоги вместе с файлами, необходимо воспользоваться командой **rm -R**.

Команда touch

Команда **touch** служит двум основным целям: созданию пустого файла или изменению даты и времени последнего доступа или модификации существующего файла. Базовый синтаксис:

- **touch имя_файла**, где **имя_файла** указывает на файл, который нужно создать или изменить.
- **touch** поддерживает несколько опций, управляющих датой и временем последнего доступа или модификации файла. Подробнее о них можно узнать из справочного руководства.

Создание ссылок

Как уже упоминалось, FreeBSD дает возможность создавать ссылки на определенные точки файловой системы. Ссылки позволяют избежать ввода длинных путей или навигации по нескольким уровням подкаталогов в средствах с графическим интерфейсом. Существует два вида ссылок: жесткие (hard links) и символические (soft links). Между ними есть важные различия.

Жесткие ссылки

Жесткая ссылка (*hard link*) — это элемент файловой системы, указывающий на тот же индексный дескриптор (физическая точка на жестком диске), что и другой файл. На самом деле существует только один физический файл. Но ему отвечает два или более элемента каталога, указывающих на одни и те же данные на жестком диске.

Жесткие ссылки создаются командой **ln** по умолчанию. Например:

```
$ ln /home/foo/documents/books/fbsd/file1.txt ./file.txt
```

В результате в текущем каталоге создается элемент под названием **file1.txt**, указывающий на ту же точку жесткого диска, что и **/home/foo/documents/books/fbsd/file1.txt**. Теперь к этому файлу можно обратиться, используя, например, команды **vi file1.txt** или **vi documents/books/fbsd/file1.txt**. Естественно, первый вариант удобнее. Именно этой цели и служат ссылки. Если посмотреть состояние ссылки командой **ls -l**, то последняя выведет следующее:

```
-rw------- 2 foo bar 26896 Mar 25 19:18 file.txt
```

Ссылка выглядит так же, как и предыдущий файл, не правда ли? Это так, потому что это и есть обычный файл в любом контексте его использования. О том, что это жесткая ссылка говорит лишь цифра 2 в поле, следующем за правами доступа. Она указывает, что существует два элемента каталога, указывающих на один индексный дескриптор (физическое местоположение на жестком диске). Все изменения, вносимые в файл или его атрибуты, подействуют и на другой элемент каталога, указывающий на те же данные на жестком диске. Например, если изменить права доступа к этому файлу, они отразятся на исходном файле **/home/foo/documents/books/fbsd/file1.txt** и на файле, созданном командой **ln**. Время модификации и размер файла для обеих ссылок совпадают.

При удалении элемента каталога **/home/foo/documents/books/fbsd/file1.txt** счетчик ссылки уменьшится на единицу. Однако сам файл на жестком диске не будет

удален, поскольку существует ссылка, указывающая на него (созданная ранее). Файл не удаляется до тех пор, пока счетчик ссылок на него не станет равным нулю.

Жесткие ссылки обладают двумя важными ограничениями:

- Их нельзя использовать как ссылки на каталоги.
- Они не могут "пересекать границы" файловых систем.

Чтобы создать ссылку на каталог или ссылку на файл, находящийся в другой файловой системе, необходимо воспользоваться символическими ссылками.

Символические ссылки

Символические ссылки (*soft links*, *symbolic links* или *symlinks*) во многом подобны ярлыкам в Windows. В отличие от жесткой, символическая ссылка представляет собой отдельный файл, имеющий собственный индексный дескриптор на жестком диске. В этом файле содержится ссылка, указывающая на другой файл. Для создания символической ссылки командой **ln** используется опция **-s**. Например:

```
$ ln -s /home/foo/documents/books/fbsd/file1.txt file.txt
```

Команда **ls -l** для этого файла выведет следующее:

```
lrwxr-xr-x 1 foo bar 31 Mar 25 19:56 file.txt -> /home/foo/documents/
books/fbsd/file1.txt
```

Основные отличия символической ссылки от жесткой:

- Права доступа ссылки не отражают прав доступа к файлу. Кроме того, атрибуты символической ссылки (права доступа, владелец, группа) нельзя изменить. Это необходимо проделать с самим файлом.
- Счетчик ссылки равен одному, а не двум. Дело в том, что символическая ссылка является файлом, который указывает на другой файл. Это не элемент каталога, указывающий на область с данными на жестком диске (т.е. не жесткая ссылка).
- Размер, приведенный для символической ссылки, не равен размеру реального файла, на который она указывает.
- Время последней модификации ссылки не совпадает с временем последней модификации файла. Оно говорит лишь о том, когда была изменена ссылка.
- После символа -> указано имя реального файла.

В отличие от жесткой ссылки, при удалении исходного файла, на который указывает символическая ссылка, данные физически удаляются, и ссылка становится недействительной. Если удаляется ссылка, исходный файл не претерпевает никаких изменений.

СОВЕТ

Если вы пытаетесь обратиться к файлу, но система выдает сообщение "**No such file or directory**" (Нет такого файла или каталога), хотя он присутствует в листинге, выводимом командой **ls**, возможно, что этот файл является устаревшей символической ссылкой, отвечающей уже несуществующему файлу. Используя опцию **-l**, проверьте, не является ли файл такой ссылкой, а также существует ли исходный файл. Следует отметить, что удаление или просто перемещение файла делают символические ссылки на него нефункциональными.

Можно сформулировать следующее общее правило: используйте жесткие ссылки при работе с файлами в пределах одной файловой системы. Они обладают теми преимуществами, что выводят реальную информацию о файле и не устаревают при удалении или перемещении исходного файла. Если же требуется создать ссылку на каталог или ссылку на файл, расположенный в другой файловой системе, необходимо воспользоваться символическими ссылками.

ПРИМЕЧАНИЕ

Жесткие ссылки имеют еще одно преимущество перед символическими: они не используют индексных дескрипторов. Вначале это не кажется проблемой, но дело в том, что количество индексных дескрипторов на диске ограничено. Если файлов (даже небольшого размера) много, свободные дескрипторы могут закончиться, хотя на диске будет еще достаточно свободного места. В этом случае система не сможет создавать новые файлы до тех пор, пока какие-либо из файлов не будут удалены, благодаря чему освободятся индексные дескрипторы.

Универсальные опции

Большинство команд, упоминаемых в этом разделе, поддерживают опции, приведенные в табл. 8.7.

Таблица 8.7 Универсальные опции большинства команд интерпретатора

Опция	*Результат*
-i	Запуск в интерактивном режиме, запрос подтверждения, если действие может причинить ущерб существующему файлу.
-v	Информативный режим. Программа выводит сообщение о каждой своей операции.
-f	Программа выполняет действия без запросов, даже если права доступа и запрещают его (например, удаление файла, когда права доступа установлены только для чтения; предполагается, что при этом пользователь имеет право на запись в каталог, где расположен файл).

ПРЕДУПРЕЖДЕНИЕ

Как можно заметить, большинство команд UNIX выполняют указанные действия над файлами и каталогами, не задавая лишних вопросов, даже в тех случаях, когда они могут уничтожить существующие файлы. UNIX не ведет пользователей за руку, как Windows. В UNIX по умолчанию предполагается, что пользователь знает, что он делает, когда просит систему выполнить что-либо. Поэтому в тех случаях, когда вы не уверены в своих действиях (например, не знаете, есть ли в каталоге, предназначенном для копирования содержимого другого каталога, файлы с такими же именами), используйте опцию **-i**. Тогда большинство команд будет запрашивать подтверждения, если возникнет угроза повреждения существующих файлов.

Метасимволы и символы-заместители

Все предыдущие команды поддерживают метасимволы и символы-заместители. Они замещают один или несколько неизвестных символов.

ПРЕДУПРЕЖДЕНИЕ

Пользователям DOS: если вы считаете, что знакомы с символами-заместителями и решите пропустить эту секцию, помните, что поведение таких символов в UNIX и во FreeBSD ОТЛИ-

ЧАЕТСЯ от их поведения в DOS. Частично это связано с тем, что в UNIX точка не рассматривается отдельно, а считается обычным символом в имени файла. Будьте осторожны с символами-заместителями в UNIX, особенно, если вы привыкли к ним в DOS.

Три основных оператора замены, используемые в командной строке FreeBSD, приведены в табл. 8.8.

Таблица 8.8 Символы-заместители

Оператор	*Назначение*
?	Совпадает с любым одиночным символом. Например, **file?.txt** совпадает с **file1.txt**, **file2.txt** и **fileA.txt**. Не совпадает с **file10.txt** и **fileAB.txt**.
*	Совпадает с любой последовательностью символов. Например, **f*** совпадает с именами файлов **f**, **foo**, **file**, **file1.txt**, **file2.txt**, **fileA.txt**, **file10.txt** и **fileAB.txt**.
[]	Совпадает с диапазоном символов (см. далее).

Совпадение с диапазоном символов

Квадратные скобки позволяют указать диапазон символов для поиска совпадения. Например, **file[1-3]** совпадает с **file1**, **file2**, **file3**, но не с **file4**. Совпадение можно указать не только для цифр, но и для букв и для других символов, например, **file[a-c]** совпадает с именами **filea**, **fileb** и **filec**.

СОВЕТ

При работе с диапазонами символов важно помнить, что совпадение основано на ASCII-коде символа. Например, **file[A-b]** совпадает не только с **fileA** и **fileb**, но и с **fileB**, **fileC** и т.д. до **fileZ**. Так происходит потому, что в кодовой таблице прописные буквы предшествуют строчным. Поэтому, когда диапазон начинается с прописной буквы и заканчивается строчной, в него входят все прописные буквы — от указанной до **Z**.

Похожий формат используется, когда нужно указать совпадение с одним символом из списка. Например, **file[1234]*** совпадает с именами файлов, которые начинаются с **file1**, **file2**, **file3** или **file4** (дальше могут следовать любые символы). Обратите внимание, что в приведенном шаблоне используются два символа-заместителя. Это допускается интерпретатором.

И наконец, с символами-заместителями можно использовать логический оператор НЕ. Например, **file[!1234]*** совпадает с именами всех файлов, которые НЕ начинаются с **file1**, **file2**, **file3** или **file4**.

ПРЕДУПРЕЖДЕНИЕ

При использовании символов-заместителей с такими командами, как **rm**, нужно быть предельно осторожным, поскольку ошибки могут быть непоправимыми. Предположим, что вместо **rm note*** была введена команда **rm note ***. В последнем случае интерпретатор рассматривает пробел как разделитель аргументов. Такая опечатка сильно меняет смысл команды: вместо "Удалить все файлы, имя которых начинается с **note**", получается "Удалить файл **note** и ВСЕ файлы в текущем каталоге"! *Дважды и даже трижды проверяйте аргументы* команд типа **rm** и *пользуйтесь* опцией **–i**.

Несколько слов об именах файлов

Хотя технически UNIX позволяет включить в имя файла любой символ (некоторые символы нельзя просто ввести, их следует указывать как двусимвольные после-

довательности, начинающиеся с косой черты), желательно все-таки ограничиться только буквами, цифрами, точкой, дефисом (-) и символом подчеркивания (_). Нежелательно начинать имя с дефиса, поскольку в UNIX дефис, как правило, интерпретируется как символ, за которым следует опция. Такие имена лишь затрудняют работу с файлами. В имя можно включать и пробелы. Однако в этом случае его необходимо заключить в кавычки, чтобы интерпретатор мог воспринять имя целиком, а не как список нескольких файлов. Я рекомендую воспользоваться распространенной практикой: заменять пробелы на символы подчеркивания. Тогда имена легко читаются, но не требуют кавычек при работе с ними.

Избегайте применения в именах файлов специальных символов.

Работа с нестандартными именами файлов

Рано или поздно может случиться так, что в вашем каталоге появятся файлы с нестандартными именами. Рассмотрим, например, такой листинг:

```
*             File\1.txt      file4.txt          file6.txt
File 1        file"3".txt     file5.txt
```

Файлы с такими именами предназначены для демонстрационных целей. Избегайте таких имен на практике!

Первый файл имеет имя *. Как же его удалить? Первой реакцией нового пользователя UNIX будет команда **rm ***. Конечно, она удалит этот файл, но вместе с ним она удалит и все остальные файлы в текущем каталоге, поскольку интерпретатор трактует звездочку как символ-заместитель! Разумеется, это не то, что нужно было бы сделать.

В таких ситуациях используется специальный escape-символ, поддерживаемый командным интерпретатором. Это символ обратной косой черты (\). Он указывает, что следующий за ним символ следует рассматривать в его первичном (а не в специальном) значении. Поэтому команда **rm -i *** удалит нужный файл, не затрагивая больше никаких файлов текущего каталога.

Следующий файл называется **File\1.txt**. Само имя включает экранирующий символ. Команда **rm File\1.txt** выдает сообщение об ошибке **File1.txt: No such file or directory**. Несложно понять, что нужно экранировать сам символ обратной косой черты: **rm -i File\\1.txt**.

Далее, перейдем к файлу **File 1**. Командный интерпретатор трактует пробелы как разделители аргументов, поэтому команда **rm File 1** вместо удаления нужного файла пытается удалить два файла: **File** и **1**. Чтобы имя воспринималось как один аргумент, необходимы кавычки: **rm -i "File 1"**.

А что делать с файлом **file"3".txt**? Попытка выполнить команду **rm file"3".txt** приводит к сообщению **file3.txt: No such file or directory**. Все дело в том, что кавычка (") является специальным символом, который также необходимо экранировать: **file\"3\".txt**.

СОВЕТ

Обратите внимание, что во всех примерах команда **rm** используется с опцией **-i**. Это следует считать хорошей привычкой, особенно при работе с файлами, имеющими нестандартные имена. Таким образом, вы всегда сможете убедиться, что команда выполняет именно то действие, которое требуется.

Куда же делся этот файл?

Иногда требуется найти файл (например, вы не помните, в каком каталоге создали или сохранили его). В таких случаях во FreeBSD используется команда **find**, имеющая несколько нестандартный синтаксис. Например:

```
$ find . -name "lostfile*"
```

Эта команда производит поиск файлов, имя которых начинается с **lostfile** в текущем каталоге, включая и все подкаталоги. Отметьте, что поиск, производимый из корневого каталога, проверяет все подкаталоги системы, включая и подкаталоги, размещенные на других дисках или в сетевых файловых системах NFS, смонтированных на текущий момент. Такой поиск требует достаточно много времени. Команда имеет несколько опций, указывающих, как глубоко в иерархии каталогов проводить поиск, а также обрабатывать все файловые системы или только текущую. Утилита **find** позволяет искать файлы не только по имени, но и по владельцу, группе и размеру. Об этих и многих других свойствах **find** рассказано на странице ее справочного руководства **man find**.

Кроме того, существует и другая поисковая команда, **locate**, которая в действительности не ищет файл на диске, а просматривает базу данных, содержащую список файлов, имеющихся в системе. Естественно, что поиск выполняется во много раз быстрее. Недостаток состоит в том, что в системе могут присутствовать новые файлы, которые еще не включены в базу данных **locate**, и поэтому не могут быть найдены этой утилитой. В зависимости от настроек база данных обновляет ежедневно или еженедельно (этим можно управлять). За подробными сведениями обратитесь к странице справочного руководства **man locate**.

Команды обработки текстовых файлов

Одной из первоначальных целей, которые компания AT&T ставила, разрабатывая UNIX, была обработка текстовых данных. UNIX и FreeBSD включают множество команд обработки текстовых данных в интерпретаторе. Здесь мы рассмотрим наиболее полезные из них.

Подсчет числа строк, слов и символов

Для подсчета числа строк, слов и символов в текстовом файле используется команда **wc**. Без опций она выводит все три величины. Например, вывод команды, данный в примере, сообщает, что в текстовом файле, в котором хранится данная глава, содержится 1160 строк, 7823 слов и 51584 символов:

```
$ wc fbu8mcu
    1160     7823    51584 fbu8mcu.html
```

Команда **wc** поддерживает несколько опций, управляющих выводом информации: **-l** — число строк, **-w** — слов и **-c** — символов.

Просмотр текстовых файлов: утилиты more и less

Раньше две этих команды были отдельными, но теперь **more** представляет собой жесткую ссылку на **less**. Несмотря на свое имя, команда **less** (меньше) является более мощной, чем **more** (больше).

Команда **less** (или **more**) используется для поэкранного отображения текстовых файлов. Кроме того, она позволяет во время просмотра производить поиск и прокручивать содержимое файла на любое число строк вперед или назад.

Несколько примеров команд, поддерживаемых утилитой **less** (в интерактивном режиме), приведены в табл. 8.9.

Таблица 8.9 Команды, поддерживаемые программой less

Команда	*Назначение*
/pattern	Здесь **pattern** задает шаблон поиска строки в файле.
ПРОБЕЛ или **f**	Прокручивает содержимое на один экран вперед. Если перед нажатием клавиши пробел ввести число, **less** прокрутит содержимое вперед на указанное число строк.
b	Прокручивает содержимое на один экран назад. Если перед нажатием **b** ввести число, **less** прокрутит содержимое назад на указанное число строк.
Клавиши курсора "↑" и "↓"	Прокручивает файл на одну строку.
# g	Здесь # обозначает число. Введите его и нажмите **g**. **less** прокрутит файл до указанной вами строки.
# %	Здесь # обозначает число от 0 до 100. По этой команде **less** прокручивает файл до той части, которая задана в процентах.

Эти опции используются с **less** наиболее часто. Их гораздо больше, в чем не трудно убедиться, заглянув в справочное руководство этой команды (его длина более 2000 строк!).

Поиск по шаблону

Для поиска по шаблону в текстовых файлах используется набор команд **grep**. Существует три разных команды: стандартная программа **grep**, которая ищет вхождение строки по простому шаблону, **egrep**, поддерживающая регулярные выражения, и **fgrep**, которая производит поиск указанных строк. В справочных руководствах некоторых ранних версий UNIX **fgrep** расшифровывалось как "fast grep" (быстрая grep), поскольку предполагалось, что эта команда быстрее, чем стандартная **grep**. На самом деле **fgrep** почти всегда работает медленнее, чем **grep**.

Предположим, что в текстовом файле **textfile** нужно найти вхождение строки **cat**. Для этого можно воспользоваться командой **grep** в простейшем формате:

```
$ grep cat textfile
```

Эта команда выводит на экран каждую строку файла, содержащую подстроку **cat**. Обратите внимание, что команда ищет вхождение шаблона, а не отдельные слова. Это значит, что такому критерию поиска удовлетворяет не только слово **cat**, но и ***catnip***, ***catbird***, **catfish** и **concatenate**, поскольку в каждом из них содержится подстрока **cat**. Если же нужно найти именно слово, необходимо окружить его пробелами и заключить весь шаблон в кавычки:

```
$ grep " cat " textfile
```

Распространенные опции **grep**: **-i** — поиск без учета регистра, **-c** — не выводить строки с вхождениями шаблона, а напечатать только количество вхождений в файле, **-n** — выводить номер каждой строки с вхождением (и саму строку), **-v** — обратить операцию и выводить строки, не включающие указанный шаблон.

Рассмотрение более мощной утилиты **egrep** выходит за пределы этой главы, но о работе с регулярными выражениями будет подробно рассказано в главе 13.

Сортировка текста в файле

Иногда необходимо отсортировать текстовый файл определенным образом. Например, в алфавитном порядке или в порядке возрастания/убывания числового поля. Для этого используется команда **sort**.

По умолчанию **sort** сортирует значение по ASCII-коду и не игнорирует ведущие пустые символы. Часто используемые опции приведены в табл. 8.10.

Таблица 8.10 Опции команды sort

Опция	*Результат*
-d	Сортировка в стиле "телефонной книги". Игнорируются все символы, кроме букв, цифр и тире.
-b	При сортировке в строках игнорируются ведущие пустые символы.
-f	При сортировке строчные буквы (внутренне) преобразуются в прописные (но в выводе сохраняются исходные значения). Фактически, эта опция отключает учет регистра.
-n	Сортировка в соответствии с числовым значением поля.
-t	Опция изменяет символ-разделитель полей. По умолчанию для отделения полей друг от друга **sort** использует пустой символ.
-u	Если во входных данных присутствуют одинаковые строки, в отсортированном выводе печатается только одна такая строка.
-r	Вывод отсортированных данных в обратном порядке.
-o	Выводит результат не на экран, а в файл, имя которого следует за **-o**. Эта опция дает тот же эффект, что и перенаправление вывода в файл (подробнее о перенаправлении ввода-вывода рассказано далее в этой главе).

Если в командной строке указано несколько файлов, **sort** конкатенирует их. Если при этом используется опция **-m**, **sort** работает быстрее, но, чтобы сортировка происходила правильно, каждый из входных файлов должен быть уже отсортирован.

Замена строк с помощью tr

Для поиска строки в файле и ее замены другой строкой используется команда **tr**. Она имеет следующий синтаксис:

```
$ tr 'a-z' 'A-Z'
```

Эта команда заменяет строчные буквы прописные. По умолчанию программа **tr** читает данные из стандартного входного потока (обычно, это клавиатура) и выводит их в стандартный выходной поток (как правило, на экран). В большинстве случаев такое поведение бесполезно, поэтому **tr** используется вместе с перенаправлением

ввода-вывода. Здесь показана основная форма запуска **tr**, когда программа получает данные из файла и направляет их в файл:

```
$ tr 'a-z' 'A-Z' < file1 > file2
```

Эта команда читает файл **file1**, заменяет буквы нижнего регистра буквами верхнего регистра, затем сохраняет новый файл в **file2**.

Команда **tr** поддерживает опцию **-d**. При этом **tr** просто удаляет все вхождения указанного символа в файле. Например, следующая команда удаляет все вхождения символов A и B в **file1** и сохраняет результат в **file2**:

```
$ tr -d 'AB' < file1 > file2
```

Вывод частей строк из текстовых файлов

Иногда необходимо вывести лишь часть строки из файла. Для вывода определенных полей файла применяется команда **cut**. Предположим, например, что файл **phone.txt** содержит адресную книгу:

```
Doe, John~105 Some Street~Anytown~NY~55555~123-555-1212
Doe, Jane~105 Some Street~Anytown~NY~55555~123-555-1212
James, Joe~251 Any Street~Sometown~CA~51111~321-555-1212
```

Чтобы просмотреть первые пять символов каждой строки можно воспользоваться командой **cut -c1-5 phone.txt**:

```
Doe,
Doe,
James
```

Более полезным применением **cut** является вывод определенных полей. По умолчанию команда **cut** для разделения полей использует символ табуляции, однако его можно заменить любым другим.

```
$ cut -f1 phone.txt
```

В данном случае адресная книга не содержит символов табуляции, поэтому, чтобы указать нужный символ, следует воспользоваться опцией **-d**.

```
$ cut -f1 -d'~' phone.txt
Doe, John
Doe, Jane
James, Joe
```

Здесь команда **cut** отображает только первое поле. При этом символом-разделителем является тильда (~).

Аналогично, можно вывести список всех пользователей системы, применив **cut** к файлу **/etc/passwd**:

```
$ cut -f1 -d':' /etc/passwd
frank
bob
alice
joe
simba
lee
```

Форматирование текста утилитой fmt

Команда **fmt** форматирует текст, разбивая его на строки по 65 символов (по умолчанию). Ей можно пользоваться, например, при подготовке текстового файла к пересылке по электронной почте или для других несложных задач форматирования. Например:

```
Until he extends his circle of compassion to include all living things, man
will not himself find peace.
-- Dr. Albert Schweitzer
```

Первая строка содержит 105 символов, т.е. слишком много, чтобы отобразить на дисплее в текстовом режиме (и даже в графическом, если разрешение невелико). В результате, либо почтовая программа разорвет строку в произвольном месте (например, посредине слова), либо текст выйдет за пределы экрана, заставляя читающего прокручивать его по горизонтали.

```
$ fmt quote.txt
Until he extends his circle of compassion to include all living
things, man will not himself find peace.
-- Dr. Albert Schweitzer
```

Этот вывод можно направить почтовой программе или в файл.

Следующий пример лучше демонстрирует возможности команды **fmt**:

```
Until he extends
his
circle of
compassion to
include all living
things
man
will not himself
find peace
-- Albert Schweitzer
```

После выполнения команды **fmt** текст будет выглядеть следующим образом:

```
Until he extends his circle of compassion to include all living
things man will not himself find peace
-- Dr. Albert Schweitzer
```

В этом разделе приведено большинство команд, полезных при работе с текстом. Комбинируя их, можно решать довольно сложные задачи, как-то: анализ log-файлов Web-сервера, поиск в них общих тенденций посещения и т.д. Конечно же, возможности этих команд ограничены. После знакомства с ними вы, вероятно, будете использовать для обработки текста такие средства, как **sed** и **awk**. Их рассмотрение выходит за рамки этой главы. Тем не менее эти средства по умолчанию включены в состав FreeBSD и применяются для решения сложных задач.

Далее мы обсудим, как все перечисленные команды комбинируются для выполнения различных операций.

Конвейеры и перенаправление ввода-вывода

Один из элементов, делающих систему UNIX столь мощной, — это возможность использовать вывод одной команды как ввод другой. Кроме того, вывод можно пе-

ренаправлять по-разному. Например, вывод команды **ls**, который обычно поступает на экран, можно с легкостью направить в файл:

```
$ ls > filelist.txt
```

В результате будет создан текстовый файл **filelist.txt**, содержащий листинг текущего каталога. Такая операция и называется *перенаправлением вывода*.

Если листинг каталог не помещается на экране, вывод команды **ls** можно направить программе **more**:

```
$ ls | more
```

Вспомните, что команда **more** отображает текст поэкранно. Это позволяет прочесть листинг любой длины.

Предположим, что приведенную выше цитату необходимо переслать по электронной почте. Чтобы не вводить ее заново, можно использовать файл, содержащий ее, как ввод команды **mail**. Существует два способа решения этой задачи, которые дают один и тот же результат. Например, можно воспользоваться командой **cat**, которая выводит содержимое файла на экран и перенаправить ее вывод программе **mail** с помощью конвейера:

```
$ cat quote.txt | mail useraddress
```

Можно воспользоваться *перенаправлением ввода*, которое приводит к тому же эффекту:

```
$ mail useraddress < quote.txt
```

В последнем случае команда **mail** получает ввод не с клавиатуры, а из файла **quote.txt**. Последняя команда более эффективна, поскольку не требует запуска утилиты **cat**, — перенаправление обрабатывает сам командный интерпретатор.

В одной команде можно использовать перенаправление как ввода, так и вывода. Например, программу **tr**, которой мы пользовались ранее, запускается так: прочесть из файла **file1**, записать в **file2**:

```
$ tr 'a-z' 'A-Z' < file1 > file2
```

(Если вы забыли, каким знаком воспользоваться — < или >, вспомните, что стрелка указывает направление пересылки данных.)

В одной команде можно использовать несколько конвейеров:

```
$ cut -f1 -d' ' access.log | sort | uniq -c | more
```

Это простой способ извлечения полезной информации из log-файла Web-сервера, совместимого с NCSA, а именно: первая утилита читает значение поля 1, где содержится сетевой адрес машины, обращавшейся к Web-серверу, затем значение пересылается программе **sort** для последующей сортировки, а затем — программе **uniq -c**. Последняя утилита подсчитывает количество вхождений идентичных строк и выводит только разные строки, перед которыми отображается количество вхождений. Затем весь вывод пересылается программе **more**. Таким образом, данная команда выводит информацию о том, сколько раз и с каких адресов происходили обращения к Web-серверу.

Конвейеры можно комбинировать и с перенаправлением ввода-вывода:

```
$ cut -f1 -d' ' /var/log/httpd-access.log | sort | uniq -c > hits.txt
```

Эта команда совпадает с предыдущей, но вывод не отображается на экран, а записывается в файл.

Дополнение командной строки и редактирование истории команд

В предыдущих разделах мы изучили основные команды интерпретатора. Здесь мы перейдем к дополнительным свойствам, имеющимся у более современных интерпретаторов.

Если неизвестно, установлен ли в системе **bash** (или любая другая программа), это легко определить, воспользовавшись таким свойством **tcsh**, как дополнение командной строки. Достаточно ввести две или три буквы имени (уникально задающих имя команды) и нажать клавишу Tab. Если введенное число букв задает команду однозначно, то оставшаяся часть имени будет автоматически дополнена интерпретатором.

Если введенной информации недостаточно для определения имени, интерпретатор выдаст сигнал "звонок" (предупреждение, вид которого зависит от терминальной программы). Для вывода списка всех возможных дополнений команды следует или повторно нажать Tab (**bash**) или Ctrl+D (**tcsh**). Обратите внимание, что комбинация Ctrl+D используется также для удаления символа справа от курсора и (если она вводится в пустой строке) для завершения сеанса работы с **tcsh**. Поэтому будьте осторожны!

```
$ bas[Tab]
basename   bash          bashbug
```

Дополнение по клавише Tab работает и для имен файлов:

```
$ ls show[Tab]
showchars.cgi*      showfavepics.cgi*  showprofile.cgi*   showuploads.cgi*
showcomments.cgi*  showpopular.cgi*   showrequests.cgi*
```

Еще одним фундаментальным свойством современных интерпретаторов является редактирование истории команд. По мере ввода команды сохраняются в буфере текущего сеанса работы. Максимальное их число указано в конфигурационном файле (в **tcsh** по умолчанию 100 команд). Прокручивать список введенных ранее команд позволяют курсорные клавиши. Для повторного запуска следует нажать клавишу Enter. Для исправления ошибок или редактирования команды необходимо воспользоваться стрелками влево или вправо.

Кроме того, каждая команда заносится в файл истории: в **tcsh** — это **.history**, а в **bash** — это **.bash_history**. Сей файл позволяет "вспоминать" команды не только текущего, но и предыдущих сеансов работы. Тем не менее, хотя правами на чтение файла истории обладает только пользователь-владелец, в большинстве случаев, его наличие рассматривается как дыра в защите, поскольку он хранит все команды, выполненные пользователем. Файл истории можно удалить. Для удаления файла пользователю достаточно ввести команду **rm ~/.history** или **rm ~/.bash_history**, в зависимости от интерпретатора.

Администрирование FreeBSD

- Файловая система FreeBSD
- Пользователи, группы и права доступа
- Конфигурация системы и стартовые сценарии
- Настройка командного интерпретатора
- Программирование на языке командного интерпретатора
- Мониторинг производительности, управление процессами и автоматизация заданий
- Установка дополнительного программного обеспечения
- Печать
- Конфигурирование ядра
- Поддержка FreeBSD
- О жестких дисках и файловых системах
- Курс выживания для пользователей FreeBSD
- Основы программирования на языке Perl

9 ГЛАВА

Файловая система FreeBSD

- Структура каталогов FreeBSD
- Мониторинг использования файловой системы
- Монтирование и демонтирование файловых систем FreeBSD
- Монтирование и демонтирование файловых систем других операционных систем
- Монтирование и демонтирование файловых систем CD-ROM и флоппи-дисков
- Понимание файла **/etc/fstab**
- Проверка и восстановление файловых систем с помощью утилиты **fsck**
- Установка и использование пользовательских квот файловой системы

Чтобы понимать, как работает операционная система FreeBSD, необходимо четко представлять, как она управляет файлами. Важно также и то, чем ее методы отличаются от методов других операционных систем.

FreeBSD использует *FFS* (BSD Fast File System, Система быстрого доступа к файлам BSD). Широко распространено заблуждение, что в современных версиях UNIX семейства BSD применяется *UFS* (*Universal File System*, Универсальная файловая система) или *UNIX File System* (Файловая система UNIX). Пользователи даже по-разному интерпретируют эту аббревиатуру! Причина этого заблуждения кроется в том, что многие утилиты, о которых будет рассказано в этой главе (например, **mount**), ссылаются на UFS как на файловую систему по умолчанию. Никто не возбраняет и вам упоминать о файловой системе BSD как о UFS, но при этом следует помнить, что "настоящая" UFS использовалась лишь в ранних версиях семейства BSD и не применяется во FreeBSD.

FFS — распространенная файловая система, используемая во FreeBSD, OpenBSD, NetBSD и других системах, включая Mac OS X (чья исходная версия, Darwin, была основана на ядре FreeBSD). В Linux, как правило, применяется Ext2FS, в Windows NT — NTFS, а Windows 95/98/Me — VFAT. Эта информация будет полезна при обсуждении монтирования файловых систем, принадлежащих другим операционным системам.

ПРИМЕЧАНИЕ

Более полный обзор различных типов файловых систем, существующих в современном компьютерном мире, можно найти по адресу **http://www.penguin.cz/~mhi/fs/**.

Структура каталогов FreeBSD

Для тех, кто работал с UNIX или UNIX-подобной операционной системой, структура каталогов FreeBSD, без сомнения, покажется знакомой (см. рис. 9.1).

Тем не менее между файловой системой FreeBSD и файловыми системами операционных систем Linux, Solaris и других вариантов UNIX существуют ключевые различия. Для пользователей, работавших в Windows или Macintosh, структура каталогов файловой системы FreeBSD покажется таинственной. Как и во многих других вопросах, связанных с UNIX, корни традиционной схемы именования каталогов теряются в далеком прошлом. Мы постараемся пролить свет на эту тайну, по крайней мере, давайте познакомимся с этой системой (см. табл. 9.1).

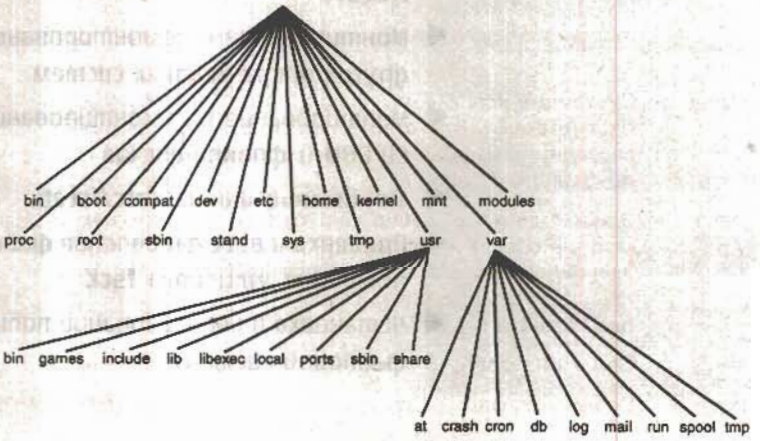


Рисунок 9.1
Иерархическая структура файловой системы FreeBSD, которая начинается с "/", то есть с корневого каталога (root).

С помощью команды **ls /** можно увидеть содержимое корневого каталога файловой системы FreeBSD. Обратите внимание на следующие элементы: каталоги обозначены /, символические ссылки — @, а выполняемые файлы — *.

Таблица 9.1 Ключевые элементы файловой системы FreeBSD

Каталог	*Назначение*
bin/	Здесь находятся выполняемые файлы, скомпилированные со статическими библиотеками. Поэтому этим каталогом можно воспользоваться в случае аварийной загрузки, когда нет доступа к программам, использующим динамические библиотеки, или другим файловым системам, кроме /.
boot/	Здесь находятся конфигурационные и исполняемые файлы, необходимые для загрузки системы. Кроме того, начиная с версии FreeBSD 5.0 в этом каталоге размещается ядро. Оно управляет всеми устройствами и сетью, а также выполняет ряд других задач. Подробную информацию о ядре вы найдете в главе 17.
compat@	Символическая ссылка на структуру каталогов, обеспечивающую совместимость с другими операционными системами, например с Linux.
dev/	Специальный каталог. Файлы внутри него являются устройствами, точнее, файлами специальных типов, которые обеспечивают программам интерфейс к устройствам, поддерживаемым ядром.
etc/	В ранних версиях системы здесь размещались разнообразные файлы, которые не могли быть помещены в другие каталоги. Теперь здесь хранятся системные конфигурационные файлы, в том числе базы данных пользовательских паролей и сценарии начальной загрузки.
home@	В зависимости от настройки системы этот элемент может быть каталогом, а может быть символической ссылкой на **/usr/home**. В нем собраны все начальные каталоги обычных пользователей.
mnt/	Пустой каталог, предназначенный для монтирования других дисков.
modules/	Здесь находятся загружаемые модули ядра.
proc/	Файловая система процессов (procfs). Она представляет собой интерфейс к таблице процессов. Используется некоторыми программами, однако не является важным компонентом работы операционной системы (т.е. ее можно благополучно демонтировать).
root/	Начальный каталог пользователя root. Он расположен за пределами каталога **/home** по соображениям безопасности и, кроме того, доступен при аварийной загрузке.
sbin/	Системные исполняемые файлы, скомпилированные со статическими библиотеками. Они отличаются от файлов, которые хранятся в **/bin**, тем, что способны изменять поведение системы, тогда как программы из **/bin** представляют собой лишь пользовательские утилиты.
stand/	Здесь содержится набор жестких ссылок на программы (hard-linked programs), которые обеспечивают среду "мини-FreeBSD" для инсталляции системы и аварийного запуска в автономном режиме (standalone mode). Скорее всего, вам понадобится лишь одна программа из этого каталога, — **sysinstall**. О ней рассказано в главе 2.
sys@	Ссылка на файлы с исходным кодом ядра, если они были установлены.
tmp/	Временные файлы. Данный каталог доступен для записи всем пользователям.

Каталог	Назначение
usr/	"Шлюз" к остальной части системы: программы, использующие динамические библиотеки, пользовательские файлы и программы, устанавливаемые администратором. В последующих главах будет подробно обсуждаться содержимое этого каталога.
var/	Изменяющиеся файлы. Здесь содержатся runtime-файлы, используемые программами, log-файлы, спул (spool) и другие элементы, чье содержимое изменяется в процессе нормальной работы операционной системы.

Перечисленные выше файлы и каталоги составляют ядро файловой системы FreeBSD. Следует отметить, что между структурой каталогов FreeBSD и UNIX-подобных операционных систем, например, Linux, существуют важные различия.

Структура FreeBSD является строго контролируемой, так как следует правилу: "Все, что устанавливает администратор, размещается в каталоге **/usr/local**". Некоторые операционные системы предоставляют пользователям определенную свободу при установке программ, но только не FreeBSD — в ней поддерживается четкая структура портированных программ и пакетов (см. главу 15). Хотя программа может по умолчанию размещать свои библиотеки в каталоге **/var/lib**, а конфигурационные файлы — в **/etc**, FreeBSD изменяет сценарии инсталляции таким образом, чтобы соответствующие файлы попали в каталоги **/usr/local/lib** и **/usr/local/etc**. Фактически, все конфигурационные файлы любого программного обеспечения размещаются в каталоге **/usr/local/etc**, а если программа устанавливает сценарий, запускать который нужно на этапе начальной загрузки системы, то он находит свое место в каталоге **/usr/local/etc/rc.d**. Содержимое этого каталога выполняется при загрузке системы после запуска основных сценариев из аналогичного каталога **/etc/rc.d**.

Следствием столь строго управляемой структуры является достаточно простая поддержка операционной системы FreeBSD, особенно ее воссоздание на новой машине (это нужно делать, например, при обновлении оборудования). Теоретически, все содержимое каталога **/usr/local** можно скопировать с одной машины на другую и все программное обеспечение будет работать также успешно, как и ранее. Однако на самом деле это довольно рискованная операция, которая зачастую приводит к непредвиденным последствиям. Тем не менее это идеал, к которому стремится FreeBSD.

Для тех, кто привык к Linux или Solaris, FreeBSD имеет очевидный недостаток — те усилия, которые, вполне возможно, придется приложить для переноса программ с других систем во FreeBSD. Так, в частности, придется изменить пути к файлам. Например, программы, написанные на языке Python для Linux, где интерпретатор Python вызывается как **/usr/bin/python**, не будут выполняться во FreeBSD, поскольку Python, не являясь частью основной инсталляции, следовательно, должен находиться в каталоге **/usr/local/bin/python**. Все это несложно исправить для одного-двух файлов, но при переносе инсталляции, содержащей сотни подобных программ, это уже проблематично. В интересах более легкой совместимости платформ проще нарушить структурные требования FreeBSD, создав символическую ссылку **/usr/bin/python**, указывающую на **/usr/local/bin/python**.

Более подробное описание организации файловой системы FreeBSD предоставляет команда **man hier**.

Мониторинг использования файловой системы

Одна из концепций UNIX-подобных операционных систем, незнакомая пользователям Windows, заключается в идее *точек монтирования* (mount points). В Windows/DOS каждому диску системы присвоена буква (например, C:), и каждый из них имеет независимую файловую систему. Например, машина с двумя дисками и приводом CD-ROM имеет диски **C:**, **D:** и **E:**, с которыми может работать пользователь. В UNIX дело обстоит иначе. Существует только одна общесистемная структура каталогов, и все диски монтируются как различные точки в ней. Здесь уместна следующая аналогия: система Windows подобна винограднику, в котором кусты расположены в ряд, UNIX же представляет собой одно большое дерево, где меньшие деревья примыкают к стволу как ветки и вместе формируют единую иерархическую крону (см. рис. 9.2).

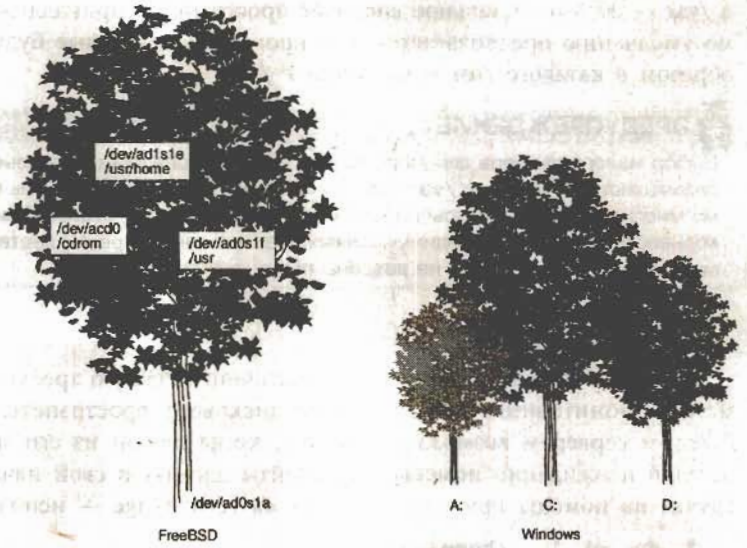


Рисунок 9.2
Структура файловых систем FreeBSD (UNIX) и Windows.

Преимуществом структуры последнего типа является легкость, с которой в систему можно добавить дисковое пространство: достаточно просто подмонтировать новый диск или раздел в какой-либо точке иерархии (мы продемонстрируем такой подход далее в этой главе). Недостаток же заключается в том, что в такой файловой системе гораздо сложнее переместить все содержимое с одного диска на другой. В Windows или Mac OS это осуществляется при помощи простого перетаскивания, поскольку каждый диск имеет отдельную файловую систему. Во FreeBSD такая задача относится к разряду более громоздких. Таким образом, иерархическая структура больше подходит для сервера, оборудование которого не меняется в течение долгого времени, чем для рабочей станции, на которой данные зачастую "живут" дольше, чем аппаратное обеспечение.

Команда df

Команда ***df*** (disk free — свободное пространство) — самый простой способ получить информацию об используемом дисковом пространстве; ознакомление с выво-

дом команды **df** является частью ежедневной проверки состояния системы (подробнее об этом см. главу 14). Вывод команды содержит основную информацию о файловых системах и именах устройств, используемых ими.

Вывод команды **df** выглядит следующим образом:

```
Filesystem   1K-blocks      Used     Avail  Capacity  Mounted  on
/dev/ad0s1a      49583     28427     17190       62%  /
/dev/ad0s1f    4254901   1959405   1955104       50%  /usr
/dev/ad0s1e      19815     12058      6172       66%  /var
procfs               4         4         0      100%  /proc
```

Каждый слайс (slice) или раздел (partition) считается файловой системой и может быть смонтирован в любую точку структуры каталогов. Вывод команды показывает, что система имеет три слайса на основном диске IDE (**/dev/ad0**), они смонтированы как /, **/usr** и **/var**. Такие установки производятся при инсталляции FreeBSD по умолчанию. Обратите внимание, что корневой файловой системе (/) отведено всего 50 Мб, а **/var** — 20 Мб. Остальное дисковое пространство принадлежит **/usr**. Это значит, что по умолчанию предполагается, что программы и данные будут размещаться главным образом в каталоге **/usr** и частично в **/var**.

ПРЕДУПРЕЖДЕНИЕ

Выбор малого размера для **/var** может оказаться опасным, поскольку log-файлы традиционно размещаются в каталоге **/var/log**. Эти файлы могут достигать очень больших размеров, поэтому многие администраторы предпочитают перемещать каталог **/var** в **/usr/var**, создавая в корневом каталоге необходимую символическую связь. Преимущества и недостатки различных методик разбиения диска на разделы обсуждаются в главе 19.

Команда du

Команды **df** иногда бывает недостаточно: зачастую требуются более специфичные методы мониторинга использования дискового пространства. Так, при управлении сетевым сервером возможна ситуация, когда любой из его многочисленных пользователей неожиданно помещает гигабайты данных в свой начальный каталог. В этом случае на помощь приходит команда ***du*** (disk usage — использование диска).

```
# du -d 1   /home/
22572       /home/bob
9           /home/fred
31          /home/alice
1520        /home/tom
66211       /home/pat
```

Команда **du** рекурсивно выводит размер всех каталогов, расположенных в текущем каталоге или в указанном в качестве параметра. Параметр **-d**, сопровождаемый числом, задает глубину рекурсии (без него вывод команды будет чрезвычайно велик). Опция **-s** задает режим сводки (summary), в этом случае вывод содержит одну строку с суммарным размером указанного каталога. Перечень опций команды **du**, включая обработку символических ссылок, можно найти на странице справочного руководства: **man du**.

Должность системного администратора накладывает на него бремя слежения за файловыми системами. Было бы неплохо, если бы система могла выполнять эту задачу самостоятельно. И такое решение существует: это *квоты* (quotas). К их обсуждению мы вернемся в конце этой главы.

Монтирование и демонтирование файловых систем FreeBSD

Здесь мы продемонстрируем многосторонность UNIX-подобных файловых систем. Предположим, что система заполнила один диск (**/dev/ad0**) и администратор добавляет другой диск как вторичное устройство первого контроллера (primary slave). Система опознает его как **/dev/ad1**. (Диски SCSI обозначаются как **/dev/da0** и т.д.) После разбиения диска на разделы и указания его метки (эта процедура подробно рассмотрена в главе 19) появится одна или несколько файловых систем, которые можно добавить в любую точку структуры каталогов системы. (Это напоминает прививку новой ветки к дереву.)

Команда mount

Предположим, например, что в систему постоянно добавляются пользователи, которые норовят заполнить свои начальные каталоги файлами большого размера. Становится очевидным (благодаря командам **df** и **du**), что раздел **/usr** почти полон — в основном за счет **/usr/home**. Не остается ничего другого как добавить новый диск, предназначенный для хранения начальных каталогов пользователей.

Покупается новый жесткий диск размером 50 Гб и разбивается на три раздела в пределах одного слайса FreeBSD — **/dev/ad1s1e** (100 Мб), **/dev/ad1s1f** (8 Гб) и **/dev/ad1s1g** (40 Гб). Третий (наибольший) раздел предназначен для начальных каталогов пользователей, остальные два — для добавления дискового пространства в других частях системы. В данный момент нас интересует только превращение старого раздела **/home** в новый 40-гигабайтный раздел, не содержащий только начальные каталоги пользователей.

У вас **/home** — это, скорее всего, символическая ссылка на **/usr/home**, которую следует удалить командой **rm /home**. (Реальный каталог **/usr/home** при этом не пострадает.) Если же **/home** — каталог, его необходимо переименовать, например, **mv /home /home.old**.

Теперь для новой файловой системы нужно создать *точку монтирования*. Точки монтирования должны быть каталогами, необязательно пустыми; однако следует помнить, что после монтирования устройства в непустой каталог, его содержимое становится недоступным. (Для одновременного доступа требуются объединенные файловые системы (union filesystems), которые на момент написания этой книги еще не поддерживались.) Для создания новой точки монтирования выполняется команда **mkdir /home**.

Теперь все готово к монтированию новой файловой системы. Для стандартных файловых систем FreeBSD оно выполняется командой **mount** без параметров. 40-гигабайтный раздел обозначается как **/dev/ad1s1g** и монтируется командой:

```
# mount /dev/ad1s1g /home
```

Если выполнение команды происходит без ошибок, новой файловой системой сразу же можно пользоваться. Проверить это позволяет команда **df**:

```
Filesystem  1K-blocks     Used    Avail  Capacity  Mounted on
/dev/ad0s1a     49583    28427    17190       62%  /
/dev/ad1s1g  39245453  3362491 32119340       12%  /home
/dev/ad0s1f   4254901   892410  3140012       22%  /usr
/dev/ad0s1e     19815    12058     6172       66%  /var
procfs              4        4        0      100%  /proc
```

Все прошло успешно! Теперь все файлы из **/usr/home** (или **/home.old**, в зависимости от настроек вашей системы) можно переместить в новую файловую систему **/home**.

Однако подобные операции не всегда проходят так легко. Монтирование файловых систем — это одна из тех областей системного администрирования, где потенциально существуют ловушки для неосторожных сисадминов. Так, например, команда может выдать сообщение об ошибке — Incorrect super block (Некорректный суперблок) или еще менее информативное — Invalid argument (Неверный аргумент). Обычно причиной таких сообщений является неправильное указание имени устройства в командной строке **mount** (соглашение об именах во FreeBSD действительно может запутать администратора). Одна и та же файловая система может быть обозначена как **/dev/ad1s1e**, как **/dev/acd0** или как **/dev/ad3s1** — в зависимости от того, адресуется она в режиме слайсов (slice mode) или в специализированном режиме (dedicated mode). Кроме того, определенные имена устройств с суффиксами могут быть взаимозаменяемы с короткими именами. Все эти тонкости более подробно обсуждаются в главе 19.

Команда **mount** имеет опцию **-f**, которая требует произвести операцию монтирования в любом случае. Но если файловая система не монтируется, то для этого, видимо, существует определенная причина. Имеет смысл выяснить и устранить ее, а не использовать опцию **-f**. Такой причиной может оказаться неформатированный диск, файловая система неизвестного типа или "грязная" файловая система, т.е. система, останов которой был выполнен некорректно. В последнем случае файловая система может иметь ряд несоответствий, которые необходимо вначале устранить утилитой **fsck** (как это сделать, будет показано далее в этой главе), а уж потом выполнять монтирование.

Весьма полезно поле **options**, в котором указываются кодовые слова (флажки), разделенные запятыми. С их помощью устанавливаются такие свойства, как доступ к файловой системе только для чтения или для чтения-записи. По умолчанию возможен доступ для чтения-записи, но опция **-r** (или **rdonly**) позволяет монтировать файловую систему только для чтения. Полный список опции **mount** можно получить, выполнив команду **man mount**.

Команда umount

Приходит время, когда файловую систему необходимо демонтировать — для этого служит команда **umount** (а не "unmount"). Чтобы демонтировать файловую систему **/home**, используется следующая команда:

```
# umount /home
```

Тот же результат дает и команда **umount /dev/ad1s1g**, а **umount -a** демонтирует все файловые системы, кроме корневой.

Демонтирование файловых систем представляет собой более простую процедуру, чем их монтирование. Однако демонтируемая файловая система не должна использоваться в момент этой операции. Это значит, что такие системы, как **/usr** и **/var**, как правило, можно демонтировать только в однопользовательском режиме. В примере с разделом **/home** все подключенные к системе пользователи будут находиться, скорее всего, в своих начальных каталогах, поэтому, прежде чем демонтировать раздел **/home**, их необходимо отключить. Чаще всего, начинающие пользователи, экспериментируя с командами **mount** и **umount**, этого не понимают.

Запомните: ***нельзя находиться внутри демонтируемой файловой системы!*** Иначе команда демонтирования выдаст сообщение об ошибке: Device busy (Устройство

занято). Поэтому возьмите себе за правило: перед демонтированием любой файловой системы выполнять команду **cd /**.

Как и **mount**, команда **umount** имеет опцию **-f**, приводящую к насильственному демонтированию. И опять-таки прибегать к ней следует лишь в случае крайней необходимости; беспорядочные операции с файловой системой могут потенциально дестабилизировать работу всей операционной системы.

Монтирование и демонтирование файловых систем других операционных систем

Все изложенное выше применимо к стандартным файловым системам FreeBSD. Что же делать, если нужно смонтировать диск, принадлежащий другой операционной системе, например, Linux, Windows 98 или NT? Такую операцию тоже можно выполнить. В основном ядре FreeBSD по умолчанию поддерживает файловые системы, перечисленные в табл. 9.2.

Таблица 9.2 Файловые системы, поддерживаемые стандартным ядром

Файловая система	*Название*
FFS	Быстрая файловая система (Berkeley Fast Filesystem)
MFS	Файловая система в памяти (Memory Filesystem)
NFS	Сетевая файловая система (Network Filesystem)
MSDOSFS	Файловая система MS-DOS (MS-DOS Filesystem)
CD9660	Файловая система ISO 9660 (CD-ROM)
PROCFS	Файловая система процессов (Process Filesystem)

Поддерживаются и другие виды файловых систем, однако для их использования необходимо заново скомпилировать ядро с соответствующими опциями (см. главу 17).

Таблица 9.3 Дополнительные файловые системы

Файловая система	*Название*
FDESC	Файловая система дескрипторов (File Descriptor Filesystem)
KERNFS	Файловая система ядра (Kernel Filesystem)
NTFS	Файловая система Windows NT (NT Filesystem)
NULLFS	Файловая система NULL (NULL Filesystem)
NWFS	Файловая система Novell NetWare (NetWare Filesystem)
PORTAL	Файловая система Portal (Portal Filesystem)
UMAPFS	Файловая система UID map (UID map Filesystem)
UNION	Объединенная файловая система (Union Filesystem)
CODA	Файловая система CODA (CODA Filesystem)
EXT2FS	Файловая система Ext2 (Ext2 Filesystem) (Linux)

Некоторые из этих файловых систем стабильнее, чем другие; как было сказано ранее, объединенные файловые системы, чья разработка еще не завершена, могут

повредить системе, поэтому монтировать их не рекомендуется. По поводу завершенности и надежности Ext2FS и NTFS также высказывалось немало опасений. Если файловая система не была включена в стандартное ядро, для этого, видимо, имелись определенные причины.

В идеальном случае, файловая система, не включенная в ядро, поддерживается как загружаемый модуль в каталоге **/modules**; такой модуль загружается автоматически при попытке монтирования файловой системы. Именно так обстоит дело со многими системами, включая CODA, PORTAL, NWFS, NULL, NTFS, UNION и другими типами, включенными в стандартное ядро, как модули.

Для монтирования файловых систем, поддерживаемых ядром по умолчанию (например, MSDOSFS), и даже тех, которые изначально не скомпилированы с ядром, FreeBSD имеет несколько удобных утилит. Они находятся в каталоге **/sbin**:

```
mount_cd9660*     mount_mfs*      mount_portal*
mount_devfs*      mount_msdos*    mount_procfs*
mount_ext2fs*     mount_nfs*      mount_std*
mount_fdesc*      mount_ntfs*     mount_umap*
mount_kernfs*     mount_null*     mount_union*
mount_linprocfs*  mount_nwfs*
```

Каждая из них похожа на **mount** и работает аналогичным образом. Фактически, утилиты **mount_*** представляют собой расширения опции **-t** команды **mount**. Эта опция поддерживается только для нескольких внутренних типов файловых систем. Если задан неизвестный аргумент, то происходит вызов соответствующей утилиты из предыдущего списка. Например, команда **mount -t nfs /mnt**, на самом деле, заменяется на **mount_nfs /mnt**. Зная это, проще сразу запустить необходимую утилиту.

ПРИМЕЧАНИЕ

Что касается стабильности и степени завершенности частично поддерживаемых файловых систем, рекомендую обратиться к страницам справочных руководств **man mount_***. Они расскажут, чего именно не хватает в поддержке конкретной файловой системы. Например, из **man mount_ntfs** можно узнать о проблемах, которые могут возникнуть при записи файлов, и об отсутствии поддержки компрессии, а **man mount_union** предупреждает об ограниченной функциональности этой файловой системы и потенциальной возможности потери данных. Обязательно ознакомьтесь с этими руководствами!

Монтирование файловой системы Windows/MS-DOS

Вначале рассмотрим монтирование диска Windows 98 как файловой системы MSDOSFS во FreeBSD. Для этого применяется команда **mount_msdos**. Она имеет несколько специальных опций, о которых рассказывать здесь нет необходимости, например, **-W** и **-L**, управляющих локальной кодовой таблицей, применяемой в длинных именах файловой системы FAT32/VFAT. В примере мы ограничимся установками по умолчанию, т.е. кодировкой ISO 8859-1, а также предположим, что чтение и запись в файловой системе производятся с правами пользователя **root**.

```
# mount_msdos /dev/ad1s1 /mnt
```

Помните: то, что в системах DOS и Linux называется "разделами" (partitions) во FreeBSD называется слайсами (slices), дабы избежать путаницы с традиционными разделами (partitions) BSD, являющимися подразделами слайсов. Поскольку файловая система

Windows 98 на устройстве **/dev/ad1s1** использует слой, а не раздел, для обращения к ней достаточно указать номер слайса без дополнительных суффиксов, которые использовались для монтирования стандартных файловых систем FreeBSD, как было рассказано в начале этого раздела (там именем устройство было **/dev/ad1s1g**).

Если используется дополнительный раздел DOS (extended DOS partition), команда может выдать ошибку — Invalid argument (Неверный аргумент). В этом случае обратите внимание, что разделы внутри дополнительного раздела DOS нумеруются начиная с 5, поэтому имя первого такого устройства: **/dev/ad1s5**.

При монтировании файловой системы MS-DOS возможны и дополнительные шаги: монтирование с UID/GID пользователя, не являющегося **root**; монтирование с маской прав доступа, управляющей доступом пользователей к содержимому файловой системы; вывод длинных имен файлов, а не коротких и т.д. Полностью все опции изложены на странице справочного руководства **man mount_msdos**.

СОВЕТ

Флоппи-диск Windows содержит имена файлов, которые могут отображаться в стандартном коротком формате MS-DOS 8.3 или же в длинном формате Windows 95/98 (при этом дополнительная информация извлекается из метаданных файловой системы). Если дискета монтируется с помощью команды **mount_msdos**, FreeBSD попытается найти на диске длинные имена файлов. В случае успеха все имена будут отображаться в длинном формате. Если же система обнаружит только короткие имена, будет использоваться короткий формат. Заставить команду **mount_msdos** принудительно отображать имена файлов в длинном или коротком формате можно при помощи опции **-l** или **-s**, соответственно.

Монтирование файловой системы Linux

В большинстве Linux-систем используется файловая система Ext2FS. Поскольку поддержка Ext2FS не включена в стандартное ядро, монтирование этой файловой системы требует дополнительного шага. (Поддержка Ext2FS вскоре будет доступна в виде загружаемого модуля, хотя не момент написания книги ее еще не было.) Действия по компиляции и установке ядра обсуждаются в главе 17. При этом необходимо добавить следующую строку:

```
options     EXT2FS
```

После того как в системе запущено новое ядро, для монтирования файловой системы Linux достаточно воспользоваться командой:

```
# mount_ext2fs /dev/ad1s1 /mnt
```

За исключением поддержки в ядре, команда **mount_ext2fs** ведет себя почти идентично стандартной **mount**, поэтому сюрпризов не предвидится.

ИСПОЛЬЗОВАНИЕ FDISK ДЛЯ ПОЛУЧЕНИЯ ИНФОРМАЦИИ О РАЗДЕЛАХ

Определить, с какой файловой системой предстоит работать, позволяет команда **fdisk**:

```
# fdisk /dev/ad1
******* Working on device /dev/ad1 *******
parameters extracted from in-core disklabel are:
cylinders=1247 heads=255 sectors/track=63 (16065 blks/cyl)

Figures below won't work with BIOS for partitions not in cyl 1
parameters to be used for BIOS calculations are:
cylinders=1247 heads=255 sectors/track=63 (16065 blks/cyl)
```

```
Media sector size is 512
Warning: BIOS sector numbering starts with sector 1
Information from DOS bootblock is:
The data for partition 1 is:
sysid 131,(Linux filesystem)
      start 63, size 2104452 (1027 Meg), flag 0
            beg: cyl 0/ sector 1/ head 1;
            end: cyl 130/ sector 63/ head 254
The data for partition 2 is:
sysid 130,(Linux swap or Solaris x86)
      start 2104515, size 787185 (384 Meg), flag 0
            beg: cyl 131/ sector 1/ head 0;
            end: cyl 179/ sector 63/ head 254
The data for partition 3 is:
sysid 131,(Linux filesystem)
      start 2891700, size 17141355 (8369 Meg), flag 0
            beg: cyl 180/ sector 1/ head 0;
            end: cyl 1023/ sector 63/ head 254
The data for partition 4 is:
<UNUSED>
```

Каждый раздел (или слайс в терминологии FreeBSD) имеет системный идентификатор (**sysid**). Файловой системе Linux Ext2FS отвечает 131, FreeBSD — 165. Коды других файловых систем, которые распознает утилита **fdisk**, находятся в файле **/usr/src/sbin/i386/fdisk/fdisk.c**.

Монтирование и демонтирование файловых систем CD-ROM и флоппи-дисков

После изучения монтирования различных файловых систем перейдем к сменным носителям — CD-ROM и флоппи-дискам. Диски CD-ROM обычно монтируются как CD9660, дискеты же, как правило, могут принадлежать либо FFS (стандарт FreeBSD), либо MS-DOS.

Монтирование компакт-дисков и дискет

В случае CD-ROM основная задача состоит в установлении правильного имени устройства. Приводы IDE обычно имеют имена вида **/dev/acd0c**, приводы SCSI — **/dev/cd0c**, а различные нестандартные приводы могут иметь и другие префиксы. Суффикс **c** указывает, что система адресуется ко всему диску в "специализированном" режиме (dedicated mode).

СОВЕТ

О текущем соглашении об именах можно узнать по адресу **http://www.FreeBSD.org/handbook/disks-naming.html**.

```
# mount_cd9660 /dev/acd0c /cdrom
```

Монтирование флоппи-диска также не вызывает затруднений. Скоре всего, имя нужного устройства это **/dev/fd0**; его не следует путать с записью, которую содержит ядро, — **fdc0**. Последняя отвечает реальному контроллеру флоппи-дисков шины ISA, тогда как имена **fd0** и **fd1** указывают на приводы, присоединенные к этому контроллеру.

```
# mount /dev/fd0 /floppy
# mount_msdos /dev/fd0 /floppy
```

Следует проявить осторожность: флоппи-диски могут быть защищены от записи, а диски CD-ROM физически доступны только для чтения, однако FreeBSD на этапе монтирования не проверяет возможность записи на носитель! Если монтировать защищенную от записи дискету без опции **-r** или **rdonly**, то при попытке записи возникнет ошибка ввода-вывода. Что произойдет дальше, зависит от стабильности приложения, осуществляющего запись, — начиная от простого вывода сообщения об ошибке и заканчивая полным зависанием системы. Поэтому при монтировании защищенной от записи дискеты или диска CD-ROM средствами командной строки обязательно указывайте опцию **-r** или **rdonly**, предотвращающую попытку записи на такой носитель на уровне ядра!

Демонтирование компакт-дисков и дискет

Поскольку компакт-диски и дискеты являются сменными носителями, существует потенциальный риск удалить носитель до того, как он будет реально демонтирован. Для сравнения: Windows динамически монтирует и обновляет состояние устройств при каждом обращении для чтения или записи. В Mac OS монтирование диска и физическая вставка-удаление носителя полностью контролируются программным обеспечением, что делает их взаимозависимыми. FreeBSD (и другие версии UNIX для x86) не имеют подобных излишеств, поэтому системный администратор должен сам заботиться о том, чтобы все смонтированные носители находились на месте.

Большинство приводов CD-ROM блокируется при монтировании носителя операционной системой (этому способствует программный механизм открытия устройства). Они не открываются в ответ на нажатие кнопки до тех пор, пока устройство не будет демонтировано. Дискеты, к сожалению, можно вытащить в любой момент, а компакт-диски — аварийно, открыв привод при помощи скрепки. Если смонтированное устройство удалено из привода до демонтирования, а программа пытается произвести чтение или запись, произойдет дестабилизация работы системы, о которой говорилось ранее.

Напрашивается следующий вывод: следует всегда помнить о демонтировании компакт-диска (**umount/cdrom**) или дискеты (**umount/floppy**) и выполнять эту операцию перед удалением носителя из привода. Это гарантирует стабильную работу системы.

Другие сменные носители

Количество сменных носителей, появляющихся на рынке, постоянно растет. Сегодня они включают в себя внешние устройства USB и FireWire, перезаписываемые компакт-диски и DVD. Приводы Zip и подобные им устройства стали уже достаточно привычными. FreeBSD поддерживает приводы Zip для параллельного порта как устройство **vp0**, а USB-приводы Zip — как **umass**. Вскоре к ним присоединятся десятки других устройств, каждое из которых имеет свои особенности по записи, смене носителя и монтируемости. Здесь мы рассмотрели лишь основы работы со сменными носителями на примере их патриархов — компакт-дисков и дискет.

Понимание файла /etc/fstab

Возникает вопрос: существует ли простой способ работать с монтируемыми устройствами автоматически, поскольку вся гибкость, которой обладают средства монтирования, становится только помехой при интенсивном использовании системы. После того как вы разберетесь, какие команды монтируют второй жесткий диск IDE,

сетевой диск NFS, дискету MS-DOS и SCSI-привод CD-ROM, нужно ли постоянно помнить эти команды? Нет! Существует более удобное средство — файл **/etc/fstab**.

Его содержимое можно просмотреть с помощью команды **cat /etc/fstab**:

```
#  Device      Mountpoint FStype     Options     Dump  Pass#
/dev/ad0s1b    none       swap       sw          0     0
/dev/ad0s1a    /          ufs        rw          1     1
/dev/ad0s1g    /home      ufs        rw          2     2
/dev/ad0s1f    /var       ufs        rw          2     2
/dev/ad0s1e    /usr       ufs        rw          2     2
/dev/acd0c     /cdrom     cd9660     ro,noauto   0     0
/dev/fd0       /floppy    msdos      rw,noauto   0     0
proc           /proc      procfs     rw          0     0
```

В этом файле содержится все, что системе нужно знать об определенной точке монтирования: какое устройство прикреплено к ней, какой тип файловой системы используется на нем, опции монтирования, а также в каком порядке следует производить проверку файловых систем при загрузке системы. *Файл* ***fstab*** тесно связан с командой **mount**. В комплексе два этих средства значительно облегчают управление файловой системой.

Основная функция **fstab** — задать системе профиль монтируемых устройств, которые следует активизировать на этапе загрузки. Когда в файле заданы все точки монтирования, их можно одновременно подключить одной командой: **mount -a**. Именно так происходит во время загрузки после того, как система выполняет проверку файловых систем командой **fsck -p**, т.е. в режиме preen (чистка), проверяющем, что все файловые системы имеют статус clean (чистый). Затем выполняется команда **mount -a -t nonfs** — монтирование всех файловых систем, перечисленных в файле **/etc/fstab**, кроме NFS.

Мало того, такие установки упрощают работу с файловыми системами. Если точка монтирования указана в файле **/etc/fstab**, не нужно в точности запоминать формат команды **mount**; достаточно указать только точку монтирования, например:

```
# mount /home
```

Команда читает всю необходимую информацию из файла **fstab**, где указано, что на устройстве **/dev/ad0s1g** установлена файловая система UFS (точнее, FFS), которую необходимо смонтировать для чтения-записи. Аналогично, для монтирования флоппи-диска достаточно ввести команду:

```
# mount /floppy
```

Согласитесь, вполне дружественный подход к пользователю!

Опция **noauto**, указанная для записей **/cdrom** и **/floppy** сообщает команде **mount**, что эти файловые системы не монтируются на этапе загрузки. Как и с ресурсами NFS, нельзя поручиться, что диск CD-ROM или флоппи-диск будет доступен во время загрузки системы, поэтому опция **noauto** предотвращает потерю времени на бесполезные попытки монтирования. Однако ничего не мешает смонтировать нужную файловую систему после загрузки, — пример такой команды приведен выше.

В четвертом столбце файла **fstab** задаются любые опции команды **mount**, применимые к файловой системе. Можно использовать, например, любую из опций, указанных на страницах справочного руководства **man mount** или **man mount_***, если это нестандартная файловая система.

Самый правый столбец файла **/etc/fstab** — поле "Pass#"; оно содержит флажок, необходимый **fsck** (об этой утилите рассказано далее). Числа, большие нуля, задают порядок проверки файловых систем. Корневой файловой системе присвоен номер

прохода, равный 1, т.е. она проверяется первой; затем проверяются смонтированные файловые системы, которым присвоен номер 2 (если позволяет оборудование, проверка выполняется параллельно). Число ноль отключает проверку файловой системы. Именно это необходимо для компакт-дисков, дискет, разделов подкачки и других ресурсов, которые или не могут быть повреждены в принципе, или их состояние не имеет значения для системы.

Левее колонки, указывающей номер прохода (pass #), находится колонка с номером уровня дампа (dump level number). Он используется командой **dump**. Это старая утилита UNIX, предназначенная для резервного копирования на основе уровней. Номер уровня дампа сообщает утилите **dump**, на каком уровне необходимо запустить резервное копирование файловой системы. Например, файловые системы, для которых он равен 1, копируется лишь тогда, когда уровень дампа равен 1 или меньше (нулевой уровень дампа означает: "полное резервное копирование всех файловых систем"). Обратите внимание, что нулевой номер уровня дампа в **/etc/fstab** полностью предотвращает резервное копирование файловой системы.

Полное обсуждение процедур резервного копирования и восстановления содержится в главе 20. Там же подробно рассказано об утилите **dump**, а также других средствах резервного копирования и зеркалирования, например, CVSup.

Проверка и восстановление файловых систем с помощью утилиты fsck

Программа **fsck** (сокращение от File System Consistency checK — проверка согласованности файловой системы) похожа на Microsoft ScanDisk и другие дисковые утилиты, по крайней мере, по своей интерактивной природе и роли в процессе загрузки. Ее основная задача заключается в проверке файловых систем из **/etc/fstab** и их пригодности для монтирования. Такой режим называется preen (чистка) и задается опцией **-p**. Кроме того, он устраняет все несогласованности в файловых системах, не отмеченных как clean (чистый) корректным методом останова системы.

Вероятно, вам придется сталкиваться с работой программы **fsck** только при загрузке системы. Обычно она запускается после идентификации всех устройств, выдавая на экран что-то вроде:

```
/dev/ad0s1e:
103469  files,  858450  used,  9066025  free
(25777  frags,  1130031  blocks,  0.3%  fragmentation)
```

ПРИМЕЧАНИЕ

Не беспокойтесь о фрагментации, о которой сообщает **fsck**. Помните, что даже фрагментация около 2–3% (вряд ли вам доведется увидеть большее значение) намного меньше той, что присуща традиционным настольным операционным системам. Для диска с системой DOS/VFAT нередко приходится наблюдать значение 50% и более. Поэтому утилиты дефрагментации в большом почете на рынке настольных систем. В файловых системах UNIX разработаны механизмы, которые автоматически размещают данные в соседних секторах, поэтому фрагментация всегда находится на низком уровне. Жесткий диск с файловой системой UNIX не придется дефрагментировать никогда (я бы не был столь категоричен — Прим. ред.). Обратитесь к разделу "Блоки, файлы и индексные дескрипторы" далее в этой главе, чтобы разобраться с механизмом хранения данных и фрагментации.

Проблемы обычно возникают тогда, когда работа системы не была завершена нормально — при сбое питания или при некорректном выключении системы. Файловые системы UNIX следят за своей структурой посредством синхронной записи на диск метаданных, которая требует нескольких циклов записи. Если несвоевременный останов происходит в процессе такой записи, метаданные могут быть повреждены. В этом случае использование файловой системы будет невозможно до ее корректировки. Для этих целей и предназначена **fsck**.

Если файловая система находится в поврежденном (unclean) состоянии, утилита **fsck** работает в режиме исследования (investigative mode). Программа последовательно, блок за блоком "проходит" по файловой системе, проверяя целостность метаданных. Эта процедура может занимать длительное время и зависит от размера и скорости диска. Если утилита находит несоответствие, которое нельзя устранить автоматически, гарантируя при этом целостность данных (подробнее см. **man fsck**), **fsck** запрашивает подтверждение пользователя. В большинстве случаев следует такое подтверждение дать. Но, если программа запросила подтверждение, то несоответствие, скорее всего, столь значительно, что часть данных будет потеряна. Обычно, это файл или файлы, запись которых происходила в момент аварии системы. Как правило, потери данных при этом невелики.

По окончании работы **fsck** система может выдать приглашение #. В этом случае необходимо продолжить загрузку командой **boot** или перезагрузить машину командой **reboot**. Последний вариант предпочтительнее, поскольку позволяет убедиться, что сервер приходит в рабочее состояние без каких-либо проблем при загрузке.

Утилита **fsck** выполняется не только при загрузке, ее можно запустить из командной строки для любой смонтированной файловой системы. Однако делать этого все же не стоит, особенно если система работает и загружена различными задачами! Важно, чтобы проверяемая на согласованность файловая система не изменялась. Если все же необходимо запустить **fsck**, следует перейти в однопользовательский режим (single-user mode):

```
# shutdown +5
```

Через пять минут после запуска команды многопользовательский режим (multiuser mode) будет отключен. Естественно, что, начиная с этого момента, все действия можно будет выполнить только с консоли. В однопользовательском режиме удаленно администрировать систему нельзя.

После того как система приведена в "неподвижное" состояние, можно запустить **fsck**. Подобный подход может оказаться необходимым, если вы обнаружите, что вывод команды **dmesg** (он ежедневно высылается пользователю **root**) содержит сообщение о некорректном индексном дескрипторе, и захотите разобраться в причинах проблемы, не перегружая систему. После запуска **fsck** для одного или всех устройств (используется синтаксис вида **fsck -p /dev/ad1s1g**) нужно просто выйти из однопользовательского командного интерпретатора (командой **exit**) и вернуться к обычному (многопользовательскому) состоянию системы.

Однако иногда команду **fsck** можно вполне безопасно запускать и в многопользовательском режиме, например, при монтировании второго диска с несущественными данными или новой файловой системой. Запуск **fsck** на этапе загрузки проверяет только файловые системы, перечисленные в файле **/etc/fstab**. Вместо добавления нового устройства в файл **fstab** и перезагрузки можно просто попытаться смонтировать устройство. Если попытка завершится неудачей, следует запустить команду **fsck** (с синтаксисом, приведенным выше). А затем вновь попытаться смонтировать диск.

Все это можно смело проделать и в многопользовательском режиме, поскольку никто не будет записывать данные на устройство, которое еще не смонтировано!

Журнальные файловые системы и мягкое обновление

Для решения проблемы синхронной записи созданы различные решения. Одно из них — это журнальные (logging) файловые системы, в которых все метаданных до непосредственной записи на диск заносятся в журнал. Это заметно ускоряет работу утилиты **fsck**, поскольку ей не требуется анализировать всю файловую систему — местоположение несогласованностей известно заранее, поэтому их остается лишь исправить.

Однако FreeBSD не поддерживает журнальных файловых систем. В ней используется другое решение проблемы — мягкое обновление (Soft Updates). В то время как журнальные файловые системы заносят в log-файл все операции записи, мягкое обновление (начиная с версии FreeBSD 5.0 оно встроено в стандартное ядро) — это потенциально более развитая методика, при которой выполняются упорядоченные операции записи, не требующие применения внешнего log-файла. Мягкое обновление обеспечивает целостность метаданных и согласованность файловой системы точно так же или даже лучше, чем журнальные файловые системы. Кроме того, оно имеет преимущество в производительности: файловая система становится доступной сразу на этапе загрузки, а проверка согласованности происходит автоматически в фоновом режиме.

Поддержка мягкого обновления во FreeBSD 5.0 осуществляется демоном **diskcheckd**. Этот демон (включен по умолчанию) запускается в фоновом режиме и производит периодическую проверку целостности файловой системы, устраняя необходимость в запуске **fsck** при загрузке и уменьшая риски, связанные с аварийным остановом системы. Конфигурационным файлом этого демона является **/etc/diskcheckd.conf**; инструкции и примеры его использования приведены в справочном руководстве, вызываемом командой **man diskcheckd**. Все ошибки, найденные **diskcheckd**, записываются службой **syslogd** (о ней рассказано в главе 11). Во время работы демона **diskcheckd** его состояние можно просмотреть командой **ps**:

```
# ps -ax | grep diskcheckd
  251  ??  Ss     0:00.28 diskcheckd: ad0 13.26% (diskcheckd)
```

ПРИМЕЧАНИЕ

Утилита **fsck** в FreeBSD очень похожа на **fsck** других операционных систем за тем исключением, что ей не хватает нескольких удобных элементов интерфейса, например, строки состояния, которая есть у **fsck** в Linux. Ключевые свойства всех утилит, конечно же, одинаковы.

Более подробную информацию о мягком обновлении и сравнении его с журнальными файловыми системами можно найти по адресу **http://www.mckusick.com/softdep/** и **http://www.ece.cmu.edu/~ganger/papers/CSE-TR-254-95/**.

Использование fsck для восстановления суперблоков

Скорее всего, изложенное ниже вам не понадобится, однако, как говорит закон Мэрфи, лучший способ убедиться, что меры предосторожности не понадобятся, это своевременно принять их.

Распространенным типом сбоя файловой системы является повреждение суперблока: когда система по каким-либо причинам не может прочесть информацию из блока, содержащего важнейшие данные о файловой системе. Этот блок называется *суперблоком* и размещается в секторах с 16 по 31 от начала устройства. Он является настолько жизненно важным для файловой системы, что FreeBSD содержит альтернативные суперблоки в начале каждой группы цилиндров. Если основной суперблок будет поврежден, на устройстве можно будет отыскать множество его резервных копий. Первая из них всегда начинается с сектора 32, остальные расположены на диске с определенным интервалом.

Предположим, что необходимо смонтировать файловую систему (например, сменный жесткий диск), который при последнем использовании был работоспособен. Однако при запуске команда **mount** выдает сообщение об ошибке: "/dev/ad1s1h on /mnt: Incorrect super block". Положение легко исправить с помощью **fsck**.

```
#    fsck  /dev/ad1s1h
**   /dev/ad1s1h
BAD  SUPER  BLOCK:  MAGIC  NUMBER  WRONG
LOOK  FOR  ALTERNATE  SUPERBLOCKS?  [yn]  y
USING  ALTERNATE  SUPERBLOCK  AT  32
**  Last  Mounted  on  /home2
**  Phase  1  -  Check  Blocks  and  Sizes
**  Phase  2  -  Check  Pathnames
**  Phase  3  -  Check  Connectivity
**  Phase  4  -  Check  Reference  Counts
**  Phase  5  -  Check  Cyl  groups
148  files,  15660  used,  7038840  free  (208  frags,  879829  blocks,  0.0%
fragmentation)
UPDATE  STANDARD  SUPERBLOCK?  [yn]  y
*****  FILE  SYSTEM  WAS  MODIFIED  *****
```

Во FreeBSD все выглядит просто: на других платформах необходимо использовать опции командной строки, указывающие расположение резервного суперблока, причем, фиксированное местоположение имеет только суперблок на секторе 32. Утилита **fsck** копирует первый резервный суперблок на место основного и после этого файловая система становится пригодной к монтированию.

ПРИМЕЧАНИЕ

Определить, где именно на устройстве расположены все суперблоки, позволяет утилита **newfs** с параметром **-N** (при этом выводятся данные о файловой системе и никаких изменений не вносится):

```
# newfs -N /dev/ad1s1h
Warning: 2672 sector(s) in last cylinder unallocated
/dev/ad1s1h: 14558608 sectors in 3555 cylinders of 1 tracks, 4096 sectors
  7108.7MB in 223 cyl groups (16 c/g, 32.00MB/g, 7936 i/g)
super-block backups (for fsck -b #) at:
 32, 65568, 131104, 196640, 262176, 327712, 393248, 458784, 524320, 589856,
  655392, 720928, 786464, 852000, 917536, 983072, 1048608, 1114144, 1179680,
  1245216, 1310752, 1376288, 1441824, 1507360, 1572896, 1638432, 1703968,
  1769504, 1835040, 1900576, 1966112, 2031648, 2097184, 2162720, 2228256,
 2293792, 2359328, 2424864, 2490400, 2555936, 2621472, 2687008, 2752544,
 2818080, 2883616, 2949152, 3014688, 3080224, 3145760, 3211296, 3276832,
  3342368, 3407904, 3473440, 3538976, 3604512, 3670048, 3735584, 3801120,
  3866656, 3932192, 3997728, 4063264, 4128800, 4194336, 4259872, 4325408,
  4390944, 4456480, 4522016, 4587552, 4653088, 4718624, 4784160, 4849696,
  4915232, 4980768, 5046304, 5111840, 5177376, 5242912, 5308448, 5373984,
  5439520, 5505056, 5570592, 5636128, 5701664, 5767200, 5832736, 5898272,
```

```
5963808, 6029344, 6094880, 6160416, 6225952, 6291488, 6357024, 6422560,
6488096, 6553632, 6619168, 6684704, 6750240, 6815776, 6881312, 6946848,
7012384, 7077920, 7143456, 7208992, 7274528, 7340064, 7405600, 7471136,
7536672, 7602208, 7667744, 7733280, 7798816, 7864352, 7929888, 7995424,
8060960, 8126496, 8192032, 8257568, 8323104, 8388640, 8454176, 8519712,
8585248, 8650784, 8716320, 8781856, 8847392, 8912928, 8978464, 9044000,
9109536, 9175072, 9240608, 9306144, 9371680, 9437216, 9502752, 9568288,
9633824, 9699360, 9764896, 9830432, 9895968, 9961504, 10027040, 10092576,
10158112, 10223648, 10289184, 10354720, 10420256, 10485792, 10551328,
10616864, 10682400, 10747936, 10813472, 10879008, 10944544, 11010080,
11075616, 11141152, 11206688, 11272224, 11337760, 11403296, 11468832,
11534368, 11599904, 11665440, 11730976, 11796512, 11862048, 11927584,
11993120, 12058656, 12124192, 12189728, 12255264, 12320800, 12386336,
12451872, 12517408, 12582944, 12648480, 12714016, 12779552, 12845088,
12910624, 12976160, 13041696, 13107232, 13172768, 13238304, 13303840,
13369376, 13434912, 13500448, 13565984, 13631520, 13697056, 13762592,
13828128, 13893664, 13959200, 14024736, 14090272, 14155808, 14221344,
14286880, 14352416, 14417952, 14483488, 14549024
```

Как можно видеть, существует множество резервных копий. Вы также можете указать определенный суперблок (если не хотите, чтобы он выбирался автоматически), например, команда **fsck -b 2490400 /dev/ad1s1h** использует суперблок из сектора 2490400.

Настройка пользовательских квот

Помните пользователей, которые беспрерывно загружали файлы, что вынудило вас купить новый жесткий диск и перенести туда их каталоги? Предположим, что вы хотите запретить любому пользователю превышать лимит в 20 Мб без вашего специального разрешения. Сделать это позволяют *квоты* (quotas).

Поддержка квот не встроена в стандартное ядро. Чтобы разрешить их использование, в конфигурацию ядра необходимо добавить следующую строку:

```
options        QUOTA
```

О компиляции ядра подробно рассказано в главе 17. Кроме того, несколько флажков поддержки квот содержатся в файле **/etc/rc.conf**. В него необходимо добавить такие строки:

```
enable_quotas="YES"
check_quotas="NO"
```

Первая строка включает глобальную поддержку квот, вторая заставляет систему во время загрузки пропустить довольно длительную проверку согласованности (**quotacheck -a**), которая удостоверяет, что базы данных квот синхронизованы нужным образом. Чтобы разрешить эту проверку, установите значение YES (или удалите строку YES, которая используется по умолчанию).

Последний шаг заключается во включении (или выключении) квот в файловых системах. Для этого в четвертом поле файла **/etc/fstab** добавляется опция **userquota** и/или **groupquota**. Например:

```
/dev/ad0s1g    /home      ufs    rw,userquota,groupquota 2      2
/dev/ad0s1f    /var       ufs    rw,userquota                   2      2
/dev/ad0s1e    /usr       ufs    rw,gropquota                   2      2
```

После выполнения всех настроек следует перегрузить компьютер. Теперь все готово к назначению квот пользователям. Это можно сделать для каждого пользователя отдельно (на сильно загруженном сервере это затруднительно) или для диапазона

UID (установив квоту для одного пользователя и применив ее как прототип для определенного диапазона UID). Чтобы установить квоту для первого пользователя, необходимо собрать утилиту **edquota**, которая позволяет редактировать атрибуты в текстовом файле (наподобие программы **chfn**, о которой рассказано в следующей главе). Текстовый редактор, который использует **edquota**, указан в переменной среды **EDITOR**, по умолчанию это **vi**, но его можно заменить на более дружественный к пользователю, например, **pico** или **ee**, командой **setenv EDITOR pico**.

```
# edquota -u bob
Quotas for user test:
/usr: blocks in use: 65, limits (soft = 50, hard = 75)
        inodes in use: 7, limits (soft = 50, hard = 60)
/var: blocks in use: 0, limits (soft = 50, hard = 75)
        inodes in use: 0, limits (soft = 50, hard = 60)
```

После установления квоты для одного пользователя ее можно распространить и на целый ряд других:

```
# edquota -p bob 1001-9999
```

Эта команда применяет указанные выше настройки квот к заданному диапазону UID (который может включать и идентификаторы еще не созданных пользователей).

Обратите внимание на разницу между блоками и индексными дескрипторами. Используются пределы для обоих значений. *Блоки* относятся к общему объему, а индексные дескрипторы понимаются в значении "файлы".

Блоки, файлы и индексные дескрипторы

Существуют различные способы размещения данных на диске. Их необходимо знать, чтобы понимать вывод таких команд, как **df** и **fsck**. Здесь рассказано о том, как хранятся данные.

Размер блока данных по умолчанию равен 8192 байта. Это единица хранения данных. Блок разделен на восемь фрагментов по 1024 байта. Файл, который занимает не полный блок, сохраняется фрагментарно вместе с другими файлами. Именно это подсчитывает утилита **fsck**, когда выводит информацию о фрагментации.

Когда файл, находящийся в одном блоке с другим файлом, увеличивается так, что ему уже не хватает места в этом блоке, FreeBSD перемещает весь файл в другой блок данных вместо того, чтобы просто разместить фрагмент файла в следующем блоке (как это делает система Windows). Таким образом, все файлы, меньшие 8192 байт, всегда сохраняются в отдельных блоках. Кроме того, файл произвольного размера занимает наименьшее возможное число блоков и имеет фрагменты только в одном блоке данных. Такой подход уменьшает время доступа и предотвращает высокую степень фрагментации, присущую другим системам.

Фактически, файл — это индексный дескриптор. Последний является фундаментальной группой связанных данных, которые мы и рассматриваем как файл. Такая модель требуется настоящей многопользовательской операционной системе для эффективного разделенного доступа к файлам. *В индексном дескрипторе содержится* следующая информация:

- Тип файла и режимы доступа к нему
- UID и GID владельца файла

- Размер файла
- Время последнего доступа и изменения файла, а также номер модифицированного индексного дескриптора
- Число блоков данных, выделенных файлу
- Прямые и непрямые указатели на эти блоки данных

Доступ к блокам данных во FreeBSD осуществляется по указателям, хранимым в индексных дескрипторах, а именно: существует 12 прямых указателей, каждый из которых указывает на отдельный блок. Таким образом, к файлу размером до 96 килобайт возможен прямой доступ. Кроме того, есть три уровня непрямых указателей: одинарный, двойной и тройной. *Одинарный* непрямой указатель содержит ссылку на блок файловой системы, содержащий указатели на блоки данных. Такой блок файловой системы содержит 2048 дополнительных адресов 8-килобайтных блоков, что дает доступ к файлу размером до 16 мегабайт. *Двойной* непрямой указатель ссылается на блок файловой системы, содержащий 2048 блоков, содержащих одинарные непрямые указатели, каждый из которых ссылается на блок файловой системы с 2048 адресами 8-килобайтных блоков. Это дает доступ к файлу размером до 32 гигабайт. И наконец, на вершине иерархии находится *тройной* непрямой указатель, который делает возможным доступ к файлу размером до 70 терабайт! Следует отметить, что в UFS максимальный размер файла все-таки ограничен и равен одному терабайту.

Теперь несколько слов о жестких и мягких ограничениях и периоде отсрочки.

- *Жесткое ограничение* (*hard limit*) выполняется неукоснительно — если пользователь достиг установленного предела дискового пространства, системе запрещено выделять ему больше.
- *Мягкое ограничение* (*soft limit*) не запрещает пользователю создавать больше файлов или использовать дополнительное дисковое пространство. Но после превышения установленного лимита начинает действовать *период отсрочки* (*grace period*), равный по умолчанию семи дням (эту установку можно изменить опцией **edquota -t**). По окончании периода отсрочки мягкие ограничения превращаются в жесткие. Такой подход позволяет пользователям на короткий период времени использовать больше дискового пространства, чем им выделено. Период отсрочки обнуляется, когда объем дискового пространства, занимаемого пользователем, становится ниже предела.

По установлении ограничений их можно просмотреть следующим образом:

```
# quota bob
Disk quotas for user bob (uid 1015):
Filesystem  blocks  quota  limit   grace   files  quota  limit grace
/home       1812    20000  40000           37     0      0
```

Команда **quota** показывает информацию о квотах для заданного или текущего (если аргумент опущен) пользователя. Если пользователь выходит за установленные пределы, это отмечается звездочкой (*), а колонка с периодом отсрочки (grace) содержит срок до его окончания:

```
# quota bob
Disk quotas for user bob (uid 1015):
Filesystem blocks  quota limit   grace   files  quota  limit  grace
/home      28121*  20000 40000   6days   189    0      0
```

ПРИМЕЧАНИЕ

Чтобы убедиться, что квоты установлены правильно, используется команда **mount** без аргументов. Вот вывод этой команды в системе, где квоты установлены для раздела **/home**:

```
/dev/ad0s1a  on  /  (ufs,  local)
/dev/ad0s1f  on  /home  (ufs,  local,  with  quotas)
/dev/ad0s1e  on  /usr  (ufs,  local)
procfs  on  /proc  (procfs,  local)
```

Если флажок **with quotas** отсутствует, это значит, что файловая система была смонтирована без квот. Проверьте файлы **/etc/fstab**, **/etc/rc.conf** и конфигурацию ядра. Если все выглядит верно, попробуйте перезагрузить систему.

Отключить квоты можно одним из трех способов: глобально, установив опцию **enable_quotas="NO"** в файле **/etc/rc.conf**; для каждой файловой системы в файле **/etc/fstab** или для определенного пользователя посредством команды **edquota**, установив мягкие и жесткие ограничения нулями. При желании, эти настройки можно применить к диапазону UID с помощью команды **edquota -p**.

Управление файловой системой — это непростая задача, но, если вы разобрались с концепциями, изложенными здесь, вам сразу же станет понятны преимущества файловых систем UNIX. В многопользовательских операционных системах, каковой является и FreeBSD, имеются возможности, отсутствующие у настольных систем, например, работа с несколькими типами операционных систем, мониторинг использования дискового пространства, применение квот. Другие особенности работы с файловыми системами рассматривается в главе 19, где мы обсудим форматирование и разметку новых жестких дисков.

10 ГЛАВА

Пользователи, группы и права доступа

- Концепция пользователей и групп
- Зачем нужны группы?
- Владение файлами
- Права доступа к файлам и каталогам
- Списки управления доступом (ACL)
- Добавление и удаление учетных записей пользователей

В этой главе рассказано о пользователях и правах доступа к файлам. Это концепции, которые составляют основу систем UNIX. Администрирование многопользовательской системы предполагает целый ряд ограничений, к которым не привыкли пользователи настольных систем. Однако именно такой подход позволяет избежать неконтролируемых сложностей, которые неизбежны на машинах, где все операции выполняются с правами суперпользователя (яркий пример — Windows).

Традиционные настольные операционные системы — Windows 95/98/Me или классический вариант Mac OS — создают впечатление многопользовательских систем, каковыми, тем не менее, не являются. Все, что они предлагают, — это возможность выбора профилей, или способов отображения данных в соответствии с настройками пользователей. Любое действие, предпринимаемое на локальной машине (чтение файлов, инсталляция программ, останов систем и т.д.), разрешено, поскольку предполагается, что только один пользователь имеет абсолютный контроль над системой. Доступ к сетевым ресурсам осуществляется с правами гостя (guest), при этом ни пользователь, ни компьютер не участвуют в каком-либо реальном процессе аутентификации в домене или его эквиваленте.

Windows NT и 2000 являют собой успешные шаги на пути к настоящей многопользовательской системе, такой как FreeBSD. Каждый пользователь в этих системах имеет отдельную учетную запись и набор прав, управляющий его доступом к файлам, принтерам и другим ресурсам. Обе системы имеют концепцию пользователя **root**, полностью распоряжающегося локальной машиной. Обе системы поддерживают группы и различные уровни доступа. Фактически, в Windows NT/2000 права доступа так сложны (наследование снизу вверх, наследование сверху вниз, доменные и локальные пользователи, до трех учетных записей с правами **root**), что в них может утонуть даже самый опытный пользователь. Но одним из важнейших элементов многопользовательской функциональности, отсутствующей у настольных машин, является возможность удаленного доступа.

Работая с NT-машиной, пользователь большую часть времени проводит за консолью, сидя непосредственно перед компьютером. Удаленный доступ требует использования специального программного обеспечения, так называемого терминального сервера (terminal server). Фактически, пользователь компьютера, работающего под NT, не имеет возможности запускать процессы непосредственно на сервере. Доступ к приложению на сервере возможен только при помощи его клиентской части, причем и клиентский, и серверный компоненты должны быть запущены предварительно. Windows 2000 включает в свой состав программное обеспечение терминального сервера, позволяющее запускать приложения с графическим интерфейсом на сервере.

FreeBSD поддерживает возможность удаленного доступа. Фактически, в системах UNIX мало что происходит на консоли. Для большинства операций используются терминальные соединения (например, Telnet или SSH), предоставляющие пользователю интерфейс командной строки по сети. В результате к машине, работающей под FreeBSD, возможен полноценный доступ из любых точек сети, причем одновременно с разных машин. Поэтому права доступа пользователей и групп играют в жизни администратора FreeBSD куда большую роль, чем в NT-системе, где администратор всегда знает, какие приложения в данный момент выполняются.

Концепция пользователей и групп

Модель пользователей и прав доступа, применяемая во FreeBSD (и большинстве систем UNIX), является одноуровневой. Есть только два типа пользователей: обыч-

ные и *суперпользователь*, или ***root***. Права доступа обычных пользователей так или иначе ограничивают их действия в системе. Пользователь **root** — единственный, кто свободен от каких бы то ни было ограничений. Другие модели прав доступа (скажем, в Windows NT/2000) включают в себя более сложные модели, что позволяет обеспечить определенные функции системы, такие как аутентификация и процессы системного уровня. А чем проще модель, тем более изощренным должен быть администратор. Ему придется очень крепко подумать, чтобы настроить, допустим, Web-сервер с правильными правами доступа (этот вопрос мы обсудим в главе 26). Следует отметить, что система со сложной схемой прав доступа является менее защищенной, поскольку содержит больше элементов, подверженных сбоям.

Каждый пользователь FreeBSD имеет ограниченные права доступа и постоянное место для работы в системе — начальный каталог (home directory). Повысить свой статус в системе до уровня **root** дает возможность команда **su**. Она запрашивает пароль **root** — фактически, ключ к царству. Пароль **root** — это самая важная информация в любой UNIX-системе. Получение доступа с правами **root** позволяет создавать, изменять или уничтожать абсолютно все, что содержит система. При регистрации в системе с правами **root** следует проявлять повышенную осторожность и всегда помнить, что злонамеренный пользователь, дабы получить доступ суперпользователя может прослушивать сеть. Поэтому пароль **root** ни в коем случае нельзя передавать по сети как обычный текст (подробности сетевой защиты изложены в главе 29). Кроме того, необходимо взять за правило менять его хотя бы раз в месяц. Здесь дополнительные предосторожности никогда не помещают.

Чтобы иметь возможность запустить команду **su** необходимо принадлежать к элитной группе, называемой **wheel**. Хотя FreeBSD имеет только два типа пользователей — обычные и **root**, группа **wheel** весьма эффективно создает ограниченный круг особо доверенных лиц: тех, кому позволено получать привилегии **root** (с помощью **su**). Используя возможности команды **su**, можно возложить часть административных задач на других пользователей.

ПРИМЕЧАНИЕ

FreeBSD отличается от многих дистрибутивов Linux и других версий UNIX тем, что при удаленном доступе (посредством Telnet или даже SSH) не позволяет пользователю **root** регистрироваться в системе. Это одна из важных мер безопасности. Для получения доступа в качестве **root** необходимо зарегистрироваться как обычный пользователь (естественно, принадлежащий к группе **wheel**) и запустить команду **su**. Такой подход затрудняет доступ неавторизованным пользователям, поскольку в этом случае им кроме пароля **root** требуется еще и пароль кого-либо из группы **wheel**. Опытный хакер найдет способ его узнать, но это потребует от него дополнительных затрат сил и времени.

Если вам потребуется отключить это свойство, сделать это можно в файле **/etc/ttys**, добавив ключевое слово **secure** в поле справа от поля **network** в разделе **Pseudo terminals** (Псевдотерминалы):

```
ttyp0     none                                network          secure
ttyp1     none                                network          secure
ttyp2     none                                network          secure
```

Однако помните, что это чрезвычайно рискованный шаг. Подобных действий следует избегать, регистрируясь в системе как обычный пользователь и запуская команду **su**.

Пользователей системы можно разделить на реальных людей, подключающихся к системе, и псевдопользователей (таких как **bin**, **operator**, **daemon**, **nobody** и другие). Последние необходимы системе для того, чтобы управлять процессами. Важно понимать,

что процессы, как и файлы, принадлежат определенным пользователям и при взаимодействии с другими процессами и файлами подчиняются ограничениям, наложенным на них правами доступа.

Пользователи никогда не работают с файлами непосредственно: выполняемые ими команды запускают процессы (имеющие установленные для пользователя права доступа), а процессы выполняют заданные операции над файлами и взаимодействуют с другими процессами (см. рис. 10.1). Процесс, владельцем которого является первый пользователь (user 1), может работать только с файлами и процессами, принадлежащими ему, но если он попытается изменить что-либо, принадлежащее второму пользователя (user 2), в доступе будет отказано. При простейших системных установках пользователь может вносить изменения лишь в принадлежащие ему файлы и процессы.

Теперь посмотрим, что происходит, если user 1 — это **root**. В этом случае его процессы имеют "абсолютную власть" над любыми процессами. Если один из процессов пользователя **root** — это программа, изменяющая определенные настройки системы в конфигурационном файле, то система станет уязвимой для атак, поскольку хакеры могут воспользоваться ошибкой в этой программе и, произведя запрос с "неправильными" параметрами, нарушить работу системы или получить контроль над ней. Последствия могут быть совершенно непредсказуемыми. Поэтому большинство системных процессов запускается с правами доступа псевдопользователей, а не пользователя **root**.

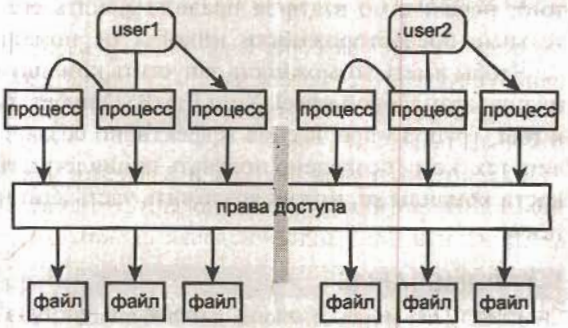


Рисунок 10.1
Пользователь запускает процессы, которые взаимодействуют с файлами и другими процессами.

Зачем нужны группы?

Каждый пользователь принадлежит к определенной первичной группе. Зачастую она включает только его самого и носит его имя. При необходимости это можно изменить, сделав первичной для всех пользователей группу **users**. Наличие отдельной группы для каждого пользователя обладает большей гибкостью (об этом далее) и представляет собой более защищенную модель. Более подробную информацию о "персональных" группах можно почерпнуть из **man adduser**.

Всякий пользователь может принадлежать к любой группе в системе, включая персональные группы других пользователей или группу **wheel** (группы добавляются в файл **/etc/group**). Единственным пользователем, управляющим принадлежностью пользователей к различным группам является **root**.

Ранее было продемонстрировано назначение группы **wheel**: это перечень пользователей, которые могут получить права доступа **root**, применив команду **su**. Группы имеют и другое назначение. Как правило, группы необходимы для того, чтобы пре-

доставить нескольким пользователям идентичные права доступа к набору файлов или процессов, скажем, позволить группе программистов, работающих над общим проектом, вносить изменения в общие файлы с исходным кодом (см. рис. 10.2). Недопустимо, чтобы несколько пользователей регистрировались под одной учетной записью или использовали один пароль. Группы же позволяют им совместно "владеть" набором файлов и иметь необходимые для работы с ним права доступа.

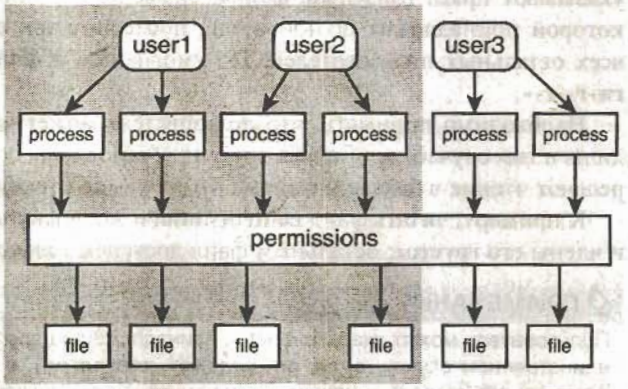


Рисунок 10.2
Два пользователя из одной группы работают с одним набором файлов, защищенным от других пользователей.

Владение файлами

Далее следует разобраться с моделью владения файлами. Все варианты UNIX имеют одинаковую структуру: каждый файл или каталог принадлежит и пользователю, и группе. Тем не менее это не означает, что все пользователи или члены группы имеют одинаковые права доступа.

Рассмотрим права доступа и владения для набора файлов (см. листинг 10.1). Для получения листинга необходимо запустить команду **ls** с опциями **-l** (подробный вывод) и **-a** (выводить все файлы, включая скрытые, т.е. файлы, имена которых начинаются с точки):

Листинг 10.1 Набор файлов с правами доступа и владельцами

```
# ls -la /home/frank
total 3126
drwxr-xr-x     3 frank    users      512 May  12   2000 .
drwxr-xr-x    52 root     users     9216 Mar   7  13:37 ..
-rw-r--r--     1 bob      users   291090 Jan  23   2000 1.bmp
-rw-rw-r--     1 bob      bob       2703 Dec  22   1998 contents.html
-rw-r--r--     1 frank    users     3657 Jan   9  14:11 file.txt
-rw-r--r--     1 bob      users    92195 Sep  11  21:31 1.uu
drwxr-xr-x     2 root     users      512 Jan   2  14:19 files
drwxr-xr-x    12 root     wheel     1024 Feb  18   1999 more-files
```

Запутанно выглядит, не правда ли? В этом разделе вы узнаете, что означает строка **drwxr-xr-x**, задающая *режим доступа* к файлу.

Для каждой разновидности пользователей — пользователь (user), группа (group) и другие (others) — существует набор битов полномочий. Эти биты предоставляют возможность читать файл, модифицировать его и выполнять, иначе говоря предоставляют три вида доступа: чтение (read), запись (write) и выполнение (execute). Смысл этих битов для файлов следующий:

- **r** — файл можно читать;
- **w** — файл можно модифицировать, удалять и переименовывать;
- **x** — файл можно выполнять.

Итак, права доступа к файлу определяется записью типа **-rwxrw-r--**. Первый дефис означает, что данный объект является файлом, следующие три символа (**rwx**) указывают права владельца файла, символы **rw-** определяют права членов группы, к которой принадлежит пользователь, последние три символа (**r--**) относятся к правам всех остальных пользователей. По умолчанию к файлам применяется режим доступа **rw-r--r--**.

Необходимо понимать, что пользователь может читать принадлежащие ему файлы лишь в том случае, если права доступа установлены соответствующим образом (т.е. разрешают чтение владельцу файла). То же самое относится и к группе.

К примеру, читать файл **contents.html** и записывать в него могут только пользователь и члены его группы; остальным файл доступен только для чтения.

ПРИМЕЧАНИЕ

Пользователь может удалять файлы, принадлежащие другому пользователю, только в одном-единственном случае: ***если он владеет каталогом, в котором эти файлы находятся***. Команда **rm** (remove — удалить) запрашивает подтверждение: следует ли отменить действие прав доступа и владения файлом и удалить его.

```
# ls -l tempfile
-rw-r--r--  1  root  users  0  Aug  7  21:44  tempfile
#  rm  tempfile
override  rw-r--r--  root/users  for  tempfile?  y
```

Изменение прав владения файлом с помощью команды chown

Суперпользователь имеет возможность изменить права владения любым файлом или каталогом в системе. Это одно из тех действий, выполнять которые может только **root**: обычные пользователи не могут передавать свои файлы друг другу.

Для изменения хозяина файла применяется команда **chown** (change owner — изменить владельца):

```
#  chown  bob  file.txt
```

Она изменяет имя *владельца* файла **file.txt** (но не группу-владельца) на bob. Если вы помните, раньше им был frank, но теперь только bob может читать и записывать в файл, а frank может только читать его.

Команда **chown** применима и к каталогам:

```
#  chown  bob  /home/frank
```

Эта команда, применяемая к каталогу, выполняется над элементом "." (см. листинг 10.1). Этот элемент указывает на текущий каталог, в то время как элемент ".." указывает на родительский каталог. Права доступа к каталогу **/home/frank** в листинге 10.1 показывают, что выполнять запись в этот каталог может только его владелец. Но с этой минуты лишь bob — единственный пользователь, имеющий возможность создавать или удалять файлы в этом каталоге, даже если они ему не принадлежат. Следует отметить, что пользователь может изменять файлы, которыми он владеет, в

каких бы каталогах они ни находились. Взаимосвязь между правами владения и правами доступа показана в табл. 10.1.

СОВЕТ

Каждый администратор должен знать полезную опцию команды **chown**: **-R**. Она заставляет команду выполняться рекурсивно, т.е. для текущего каталога, всех файлов в нем и всех файлов во всех подкаталогах, находящихся ниже в иерархии. Эта опция применяется, например, при воссоздании учетной записи, когда необходимо изменить права владения всеми пользовательскими файлами:

```
# chown -R bob /home/frank
```

Изменение групповых прав владения файлом с помощью команды chgrp

В этом разделе рассматривается похожая на **chown** команда — **chgrp**. Она изменяет права владения группы, а не пользователя. Действует она подобным образом:

```
# chgrp users contents.html
```

После этого права владения файлом **contents.html** выглядят так:

```
-rw-rw-r--  1 bob  users  2703 Dec 22  1998 contents.html
```

Поскольку и пользователь, и группа имеют право записи, в данной ситуации любой пользователь из группы users может записывать в файл точно так же, как и bob.

По умолчанию во FreeBSD для каждого пользователя создается новая группа: например, группа bob является первичной для пользователя bob. Все файлы создаются с правами владения: пользователь bob, группа bob. Если другой пользователь (например, frank) принадлежит к группе bob, он может записывать в файлы. Это механизм, который предоставляет пользователям bob и frank одинаковые возможности доступа к файлам.

chgrp — это просто другой способ запуска команды **chown**; при желании можно использовать следующий синтаксис:

```
# chown bob.users contents.html
```

После этого файлом **contents.html** владеют пользователь bob и группа users. Чтобы изменить только группу владельца применяется формат:

```
# chown .users contents.html
```

И **chown**, и **chgrp** поддерживают опцию **-R**, как было рассказано ранее.

Права доступа к файлам и каталогам

Снова обратимся к файлам из листинга 10.1. Пришло время изучить строки, задающие права доступа к каталогам. Прежде всего, каталоги распознаются по первому биту в строке — **d**. Это просто флажок, не связанный с правами доступа.

Полномочия на работу в каталоге действуют по тому же принципу, что и полномочия на файлы, но здесь есть отличия:

- **r** — каталог можно читать (выполнить команду **ls**);
- **w** — каталог можно модифицировать (создавать или удалять файлы), удалять и переименовывать;
- **x** — в каталоге можно выполнять операции над файлами, в том числе производить поиск файлов в нем.

Так, например, запись **drwxr-xr-x** означает, что данный объект является каталогом (**d**), его владелец может выполнять в этом каталоге любые действия, а его группа и остальные могут только читать и выполнять поиск. **Обратите внимание**: чтобы содержимое каталога можно было просмотреть, у него должен быть установлен бит выполнения — **x**. Для каталогов этот бит имеет смысл "поиск". Если выполняемый файл является сценарием, то пользователь для выполнения этого файла должен иметь право на чтение и выполнение. Для выполнения двоичного файла достаточно иметь только разрешение на выполнение.

Взаимосвязь между правами доступа файлов и каталогов

В табл. 10.1 показано взаимосвязь между пользователем, группой, к которой он принадлежит, и правами доступа, которые предоставляет каталог в зависимости от своего режима.

Таблица 10.1 Возможности манипулирования файлами при различных правах доступа к ним и содержащим их каталогам

Действия	*Каталог недоступен для записи*		*Каталог доступен для записи*	
Создать файл	-		+	
	Пользователь не имеет прав на запись в файл	*Пользователь имеет право на запись в файл*	*Пользователь не имеет прав на запись в файл*	*Пользователь имеет право на запись в файл*
Перезаписать содержание файла	-	+	-	+
Редактировать содержание файла	-	+	-	+
Переименовать файл	-	-	+	+
Удалить файл	-	-	+	+

При попытке удалить файл (**rm**), право записи в который отсутствует, система выдает следующий запрос:

```
# rm file.txt
override rw-r--r-- bob/users for 1.uu? y
```

Заметьте, что система, хоть и спрашивает, удалить файл все равно не позволит! Запрос появляется при всякой попытке удалить файл, не принадлежащий пользователю, причем даже положительный ответ ("**y**") не приводит к его удалению. Вывод этого запроса можно отключить с помощью опции **rm -f**.

Во FreeBSD тот файл, который можно удалить, можно также перенести в другой каталог посредством команды **mv** (move — переместить). При этом файл сначала копируется, *а затем удаляется*. Файл, который можно только читать, можно лишь скопировать.

Кроме стандартных прав доступа, применимых к пользователю, группе и всем остальным пользователям, существует несколько дополнительных режимов. О них будет рассказано далее, после обсуждения механизма изменения прав доступа к файлам.

Изменение прав доступа к файлам и каталогам с помощью команды chmod

Для изменения прав доступа к файлам или каталогам используется команда **chmod** (change mode — изменить режим). Команду **chmod** можно использовать двумя способами: с числовыми или символическими аргументами.

Изменение режимов с численными аргументами

Самый простой способ изменения прав доступа заключается в установке трехзначного восьмеричного числа, которое уникальным образом задает права доступа для каждого типа владельца. Каждая цифра определяет режим владельца: пользователя, группы и всех остальных. Число, задающее режим, получается путем суммирования чисел, отвечающих битам прав доступа. Возможные значения битов приведены в табл. 10.2.

Таблица 10.2 Биты режимов прав доступа и их значения

Бит	*Значение*
0	нет доступа
1	доступ на выполнение (для каталогов — поиск)
2	доступ на запись
4	доступ на чтение

Таким образом, режиму "чтение и запись" соответствует **6**, режиму "чтение и выполнение" — **5**, а режиму "чтение, запись и выполнение" — **7**.

Комбинация цифр формирует трехзначное число, задающее стандартные права доступа к файлу. Несколько примеров приведено в табл. 10.3.

Таблица 10.3 Права доступа в численной форме

Режим	*Значение*
755	чтение/запись/выполнение для владельца, чтение/выполнение для группы и остальных
644	чтение/запись для владельца, только чтение для группы и остальных
600	чтение/запись для владельца, для группы и остальных доступа нет

Применить права доступа к файлу или каталогу позволяет команда вида:

```
# chmod 755 testscript.sh
```

Есть еще четвертая цифра, которая управляет "дополнительными" свойствами: задает особое поведение файлов и каталогов при определенных обстоятельствах. Ниже приведены значения битов, формирующих четвертую цифру.

- **0** — обычные права доступа.
- **1** — бит устойчивости. Он устанавливается только для каталогов: владелец имеет право удалять или переименовывать только файлы, которыми он владеет, причем лишь при наличии права на запись в этот каталог.

- **2** — установить идентификатор группы, **setgid**. Когда такой бит установлен для выполняемого файла, последний выполняется с правами группы, владеющей файлом, а не с правами пользователя, запустившего его.
- **4** — установить идентификатор пользователя, **setuid**. Когда такой бит установлен для выполняемого файла, последний выполняется с правами пользователя, владеющего файлом, а не с правами того, кто его запустил. Четвертая цифра принадлежит наибольшему разряду (другими словами, она находится слева). В предыдущем примере права доступа **755** эквивалентны **0755**. Значение дополнительного бита формируется из тех же соображений, что изложены выше, поэтому, например, **3755** создает каталог с установленным битом устойчивости и битом **setgid** в дополнение к обычным правам доступа **755**.

Изменение режимов с символическими аргументами

Хотя восьмеричная система является достаточно прозрачной, существует и другой способ, легче запоминающийся. Это символический метод. Вместо указания числа в аргументе команды **chmod** ей задается от одного до трех флажков. Строку, содержащую их, можно отформатировать различными способами. Здесь указаны наиболее распространенные из них.

ПРИМЕЧАНИЕ

Полное описание гибкого синтаксиса команды **chmod** содержится в справочном руководстве **man chmod**.

Примеры символических режимов приведены в табл. 10.4. Каждый из них задается строкой символов. Первые символы задают, чьи права меняются (**u** — владелец; **g** — группа, **o** — остальные, **a** — всех (устанавливается по умолчанию, если первый символ не задан)). Второй символ — вид изменения (+,-,=), а третий — биты прав доступа.

Таблица 10.4 Символические режимы прав доступа

Строка режима	*Значение*
go+w	добавить право на запись для группы и остальных пользователей
+x	добавить право на выполнение для всех
o-r	удалить право на чтение для остальных пользователей
ugo=rw	установить всем права на чтение и запись
a=rw	совпадает с ugo=rw
+t	добавить бит устойчивости (sticky bit)
+s	добавить биты **setuid** и **setgid**

```
# chmod g+w file.txt
```

Символический метод проще запоминается и поэтому удобнее для большинства операций с командой **chmod**.

СОВЕТ

Как и команды **chown** и **chgrp**, **chmod** поддерживает опцию **-R**.

Списки управления доступом

ПРИМЕЧАНИЕ

Раздел, посвященный спискам управления доступом, относится только к FreeBSD 5.0, поскольку FreeBSD 4.x их не поддерживает.

Списки управления доступом (Access Control Lists, ACL) появились во FreeBSD версии 5.0. Они позволяют более строго управлять тем, какие пользователи имеют доступ к файлам и каталогам. Вместо разрешения или запрещения доступа на основе категорий владельца, группы и всех остальных, можно предоставлять права доступа отдельным пользователям и группам. Для них можно установить маску максимальных прав доступа (maximum permission mask), которая отменяет реальные права, установленные в ACL.

Для поддержки ACL ядро необходимо скомпилировать с расширениями UFS. Опции, которые необходимо добавить, приведены в следующем разделе.

Конфигурирование поддержки ACL в ядре

На время написания книги еще не было известно, будут ли во FreeBSD 5.0 расширения UFS (UFS Extensions) включены в ядро по умолчанию. Если они включены, то нет необходимости перекомпилировать ядро. Если же нет, то следует добавить соответствующие опции конфигурации и заново собрать ядро.

Прежде всего, необходимо взглянуть на конфигурационный файл ядра и проверить, включена поддержка расширений UFS или нет. Если вы не собирали ядро самостоятельно, то на платформах Intel x86 его конфигурационный файл находится в **/usr/src/sys/i386/conf/GENERIC**, а на платформах Alpha — в **/usr/src/sys/alpha/conf/GENERIC**. Если ACL поддерживается, файл должен содержать следующие строки:

```
options         UFS_EXTATTR
options         UFS_EXTTR_AUTOSTART
options         UFS_ACL
```

Если эти опции присутствуют, значит, ничего менять не нужно. Если нет, необходимо добавить их и собрать новое ядро. Как сконфигурировать, собрать и установить новое ядро, мы подробно обсудим в главе 17.

После установки нового ядра и перезагрузки системы необходимо сконфигурировать файловые системы, с которыми будут использоваться ACL.

Конфигурирование файловых систем для использования ACL

Для управления дополнительными атрибутами файловых систем применяется команда **extattrctl**. Ее необходимо применять к каждой файловой системе, с которой ACL будет использоваться. Например, для включения ACL на файловой системе, смонтированной как **/usr**, нужно придерживаться следующей процедуры:

Сначала, зарегистрировавшись как **root**, создайте каталоги **/usr/.attribute** и **/usr/.attribute/system**:

```
mkdir   /usr/.attribute   /usr/.attribute/system
```

Затем перейдите в каталог **/usr/.attribute/system** (**cd /usr/.attribute/system**) и запустите команду **extattrctl** для инициализации атрибутов ACL на файловой системе:

```
extattrctl initattr -p /usr 388 posix1e.acl_access
```

На больших файловых системах выполнение команды занимает несколько минут.

По завершении ту же команду необходимо запустить с другим аргументом (именем файла):

```
extattrctl initattr -p /usr 388 posixle.acl_default
```

Возможно, придется подождать несколько минут.

ПРЕДУПРЕЖДЕНИЕ

Файлы, создаваемые предыдущими командами, могут достигать чрезвычайно больших размеров — до нескольких гигабайт на больших файловых системах. Поэтому до запуска команд убедитесь, что на жестком диске имеется достаточно свободного пространства.

После выполнения этих команд необходимо перегрузить систему или заново смонтировать файловые системы, чтобы изменения вступили в силу. После этого ACL для файловой системы будут активизированы.

Получение информации о текущих установках ACL

Для получения информации о существующих свойствах ACL-файлов или каталогов используется команда **getfacl**. Например, для вновь созданного файла **acltest.txt** без установок ACL с правами доступа по умолчанию 644 команда **getfacl acltest.txt** возвратит следующий результат:

```
#file:acltest.txt
#owner:0
#group:0
user::rw-
group::r--
other::---
```

В первой строке указано имя файла. В двух следующих — идентификатор пользователя и группы владельца файла (в данном случае это **root**). В трех последних строках содержатся текущие права доступа к файлу. Обратите внимание, что остальные пользователи не имеют доступа к файлу.

Установка маски максимальных прав доступа

Маска максимальных прав доступа (*maximum permissions mask*) управляет правами, которые предоставляются пользователю (или группе), добавленному в список ACL. Обратите внимание, что маска применяется к тем пользователям и группам, которым в записи ACL предоставлен доступ к файлу или каталогу. Она не влияет на права доступа владельца файла, группы по умолчанию, а также всех остальных пользователей. Для тех, кто не перечислен в списке ACL, применяются стандартные права доступа к файлу.

Для добавления, изменения или удаления записей из списка ACL применяется команда **setfacl**. Она имеет следующий синтаксис:

```
setfacl action permissions filename
```

Здесь ***action*** — действие (добавление записи, изменение, удаление и т.д.), ***permissions*** — устанавливаемые права доступа ACL, ***filename*** — имя файла или каталога. В следующем примере файлу **acltest.txt** добавляется маска максимальных прав доступа ACL на чтение:

```
setfacl -m m::r acltest.txt
getfacl acltest.txt
#file:acltest.txt
#owner:0
#group:0
user::rw-
group::r--
mask::r--
other::---
```

В синтаксисе команды **setfacl** опция **-m** означает, что ACL-запись добавляется или изменяется. **m::** указывает, что устанавливается маска, а **r** означает просто read — права на чтение. Чтобы добавить права на запись, необходимо использовать синтаксис **m::rw**. Последний аргумент, конечно же, имя файла.

После установки маски вывод команды **getfacl** несколько изменяется: команда отображает те же права доступа и установленную маску.

Добавление пользователя или группы в ACL

Чтобы добавить в список ACL пользователя или группу используется упоминавшаяся ранее команда **setfacl**. Например, для предоставления права на чтение пользователю foobar применяется следующий синтаксис:

```
setfacl -m u:foobar:r acltest.txt
getfacl  acltest.txt
#file:acltest.txt
#owner:0
#group:0
user::rw-
user:foobar:r--
group::r--
mask::r--
other::---
```

Как можно видеть, в списке ACL появилась новая запись для пользователя foobar. Теперь он может читать файл, хотя стандартные права доступа и не позволяют ему этого.

Для предоставления тому же пользователю прав на чтение и запись применяется такая команда:

```
setfacl -m u:foobar:rw acltest.txt
#file:acltest.txt
#owner:0
#group:0
user::rw-
user:foobar:rw-
group::r--
mask::rw-
other::---
```

Вывод команды **getfacl** показывает, что foobar имеет право записи в файл. Обратите внимание, что вместе с этим маска автоматически обновляется до максимальных прав доступа на чтение и запись. Если такое поведение нежелательно (другими словами, не нужно, чтобы маска обновлялась), при установке ACL следует использовать опцию **-n**. Например, следующая команда устанавливает в списке ACL права на чтение и запись файла для пользователя foobar, но не обновляет маску даже в том случае, когда она не содержит права на запись:

```
setfacl -n -m u:foobar:rw acltest.txt
```

СОВЕТ

Порядок следования опций важен. Опция **-n** должна предшествовать опции действия (в данном случае **-m**). Если их поменять местами, команда выдаст ошибку: **setfacl: acl_from_text() failed: Invalid argument**.

ПРЕДУПРЕЖДЕНИЕ

Помните, что при добавлении новой записи в ACL без опции **-n** маска максимальных прав доступа обновляется автоматически. Например, если текущая маска разрешает только чтение и в ACL добавляется пользователь с правами на чтение и запись, маска обновляется автоматически также с правами на чтение и запись. Если вам это не нужно, используйте опцию **-n**.

СОВЕТ

Когда опция **-n** не используется, маска автоматически повышает права доступа до тех, которые имеет добавляемый в ACL пользователь (если они выше). Точно так же маска может быть автоматически понижена. Например, после удаления единственной ACL-записи пользователя с правами на запись, права на запись будут автоматически удалены из маски. Предотвратить понижение прав доступа в маске позволяет та же опция **-n**.

После предоставления пользователю foobar прав на чтение и запись файла список ACL выглядит следующим образом:

```
#file:acltest.txt
#owner:0
#group:0
user::rw-
user:foobar:rw-
group::r--
mask::rw-
other::---
```

Если затем вновь необходимо изменить маску так, чтобы максимальные права доступа в ACL включали только чтение, используется следующая команда:

```
setfacl  -m  m::r  acltest.txt
```

После этого вывод команды **getfacl** выглядит так:

```
#file:acltest.txt
#owner:0
#group:0
user::rw-
user:foobar:rw-                 #  effective:  r--
group::r--
mask::r--
other::---
```

Обратите внимание, что пользователь foobar все равно имеет права на чтение и запись. Однако — в силу действия маски — эффективные права этого пользователя включают только чтение.

В одной командной строке можно задать несколько списков ACL, разделенных запятой. Например:

```
setfacl -n -m u:foobar:rw,u:guest:r,g:visitors:r acltest.txt
```

Эта команда предоставляет доступ на чтение и запись пользователю foobar, на чтение — пользователю **guest** и группе **visitors**. Кроме того, использование опции **-n**

предотвращает обновление маски, даже если текущая маска запрещает некоторые из прав доступа, указанные в операторе.

Запрещение доступа с помощью ACL

Так же как список ACL используется для предоставления пользователям и группам доступа к файлам и каталогам, его можно применять и для решения обратной задачи — запрещения доступа тем, кто в ином случае имел бы его. Например, следующая команда создает запись ACL для пользователя foobar, не содержащую прав доступа:

```
setfacl -m u:foobar: acltest.txt
```

В этом случае пользователю foobar запрещен какой-либо доступ к файлу **acltest.txt**, даже если он является членом группы-владельца этого файла, которой доступ к нему разрешен. ACL имеет более высокий приоритет, чем стандартные права доступа к файлу.

Удаление записи из ACL

Опция **-x** команды **setfacl** используется для удаления записи из ACL. Например, следующая команда полностью удаляет запись пользователя foobar:

```
setfacl -x u:foobar: acltest.txt
```

Удаление всех записей из ACL

Для удаления всех записей из ACL применяется опция **-b**. Например:

```
setfacl -b acltest.txt
```

Более подробная информация об ACL, включая возможность установки ACL для каталогов по умолчанию, содержится на страницах справочных руководств по командам **getfacl** и **setfacl**.

Добавление и удаление учетных записей пользователей

А теперь мы поговорим о том, как расширить пользовательскую базу системы, то есть о добавлении учетных записей пользователей и групп. Для *добавления учетных записей* пользователей в систему применяется сценарий **adduser**. Эта программа отличается от сценариев **adduser** или **useradd**, используемых в Linux, но выполняет ту же задачу. Запустить сценарий **adduser** может только пользователь **root**.

При первом запуске сценарий **adduser** настраивает конфигурационный файл с набором значений по умолчанию. При следующем запуске можно воспользоваться опцией **-s** (silent — тихий), когда используются эти значения по умолчанию и программа запрашивает только основные данные о каждом пользователе. В листинге 10.2 показан исходный запуск сценария:

Листинг 10.2 Примера сеанса работы adduser

```
Use option ``-silent'' if you don't want to see all warnings and
~questions.

Check   /etc/shells
Check   /etc/master.passwd
Check   /etc/group
Enter your default shell: csh date ksh no sh tcsh [ksh]:
```

```
Your default shell is: ksh -> /usr/local/bin/ksh
Enter your default HOME partition: [/home]:
Copy dotfiles from: /usr/share/skel no [/usr/share/skel]:
Send message from file: /etc/adduser.message no
[/etc/adduser.message]:
Use password-based authentication (y/n) [y]: y
Enable account password at creation (y/n) [y]: y
Use an empty password (y/n) [n]: n

Ok, let's go.
Don't worry about mistakes. I will give you the chance later to
➥correct any input.
Enter username [a-z0-9_-]: joe
Enter full name []: Joe User
Enter shell csh date ksh no sh tcsh [ksh]:
Enter home directory (full path) [/home/joe]:
Uid [1005]:
Enter login class: default []:
Login group joe [joe]:
Login group is ``joe". Invite joe into other groups: guest no
[no]:
Use password-based authentication (y/n) [y]: y
Use an empty password (y/n) [n]: n
Enter password []:
Enter password again []:
Enable account password at creation (y/n) [y]: y

Name:       joe
Password:   ****
Fullname:   Joe User
Uid:        1005
Gid:        1005 (joe)
Class:
Groups:     joe
HOME:       /home/joe
Shell:      /usr/local/bin/ksh
OK? (y/n) [y]:
Added user ``joe"
Send message to ``joe" and: no root second_mail_address [no]:

Joe User,

your account ``joe" was created.
Have fun!

See also chpass(1), finger(1), passwd(1)

Add anything to default message (y/n) [n]:
Send message (y/n) [y]: n
Copy files from /usr/share/skel to /home/joe
Add another user? (y/n) [y]: n
Goodbye!
```

Нажатие клавиши Enter после каждого запроса устанавливает значение по умолчанию — оно показано в квадратных скобках. В некоторых случаях значение выбирается из списка (символом-разделителем в нем служит пробел), например, при установке командного интерпретатора; если список включает значение **no**, то оно запрещает или отменяет все остальные возможности. Например, выбор опции **no** для командного интерпретатора создает пользователя, который такового не имеет, а это значит, что пользователь не может зарегистрироваться в системе.

СОВЕТ

Во FreeBSD 4.0 и выше командный интерпретатор **csh**, встроенный в основную часть системы, на самом деле представляет собой интерпретатор **tcsh**, т.е. версию **csh** с более широкими возможностями. **tcsh** можно указать в качестве командного интерпретатора, но это значение совпадает с **csh**.

При желании можно создать файл **/etc/adduser.message**, который будет направляться каждому новому пользователю в его почтовый ящик. Пользователь сможет прочесть его при первой регистрации в системе с помощью почтовой программы (подробнее о почтовой службе — в главе 25).

Файлы, начинающиеся с точки, упоминаемые в выводе сценария, представляют собой скрытые файлы конфигурации командного интерпретатора (их список выводится лишь при вызове команды **ls** с опцией **-a**):

```
.cshrc          .login_conf       .mailrc     .rhosts
.login          .mail_aliases     .profile    .shrc
```

Назначение этих файлов подробно объясняется в главе 12.

Каждый пользователь и группа имеет численный эквивалент своего имени — идентификатор пользователя и идентификатор группы (UID и GID, соответственно). Именно этот идентификатор указан в информации о владении файлом или каталогом. Он также используется при управлении процессами. При удалении учетной записи пользователя, права владения его файлами продолжают принадлежать оставшемуся UID удаленного пользователя.

Удаление пользователя выполняется достаточно просто. Для этого применяется команда **rmuser**, аргументом которой служит имя пользователя (для **adduser** оно не требуется), как показано на листинге 10.3.

Листинг 10.3 Пример сеанса работы rmuser

```
# rmuser joe
Matching password entry:

joe:IRBpIrE/nkDQo:1008:1008::0:0:Joe    user:/home/joe:/bin/csh

Is this the entry you wish to remove? y
Remove user's home directory (/home/joe)? y

Updating password file, updating databases, done.
Updating group file: (removing group joe -- personal group is empty)
➥done.
Removing user's home directory (/home/joe): done.
Removing files belonging to joe from /tmp: done.
Removing files belonging to joe from /var/tmp: done.
Removing files belonging to joe from /var/tmp/vi.recover: done.
```

Строка matching password entry (проверка пароля) относится к базе данных, в которой хранится вся информация о пользователях. Именно о ней и пойдет речь дальше.

Файлы /etc/passwd и /etc/master.passwd

Все операционные системы, подобные UNIX, содержат файл **/etc/passwd**, но в зависимости от платформы их роль разная. В некоторых системах это единственное хранилище пользовательской информации (включая и пароли). В этом случае добавление новой учетной записи пользователя означает добавление еще одной строки в этот файл с помощью текстового редактора (**vi**, **pico**, **ee**, **emacs** и др.). В современных операционных системах применяется теневое хранение паролей: в зашифрованном

виде они хранятся не в **/etc/passwd**, а в файле, доступном для чтения только пользователю **root**. Имя этого файла зависит от системы. В одних системах это **/etc/shadow**, в других — **/etc/security/master.passwd**. Во FreeBSD этот файл называется **/etc/master.passwd**.

Оба файла **passwd** представляют собой обычные текстовые базы данных, где каждому пользователю отвечает одна строка, поля в которой разделяются двоеточием (:). В каждой строке указано: имя пользователя, его идентификатор, идентификатор первичной группы, начальный каталог, начальный командный интерпретатор и полное имя пользователя (которое, в свою очередь, является списком следующих полей, разделенных запятой: "Адрес офиса", "Рабочий телефон", "Домашний телефон").

Права доступа к файлу **/etc/passwd** установлены как **0644**, а **/etc/master.passwd** — как **0600**. Такая схема защиты означает, что все пользователи имеют доступ к информации, содержащейся в **/etc/passwd**, и только **root** — к информации из файла **/etc/master.passwd**, который отличается от первого только тем, что во втором поле содержатся зашифрованные пароли пользователей. Во FreeBSD пароли шифруются посредством алгоритма MD5.

```
/etc/passwd:
joe:*:1008:1008:Joe    User:/home/joe:/bin/csh

/etc/master.passwd:
joe:$1$32iknJXS$TnJUJj9LzYGwWRZonOu/I0:1008:1008:Joe      User:/home/joe: .
➥/bin/csh
```

Однако информация об учетных записях пользователей может хранится не только в этих двух файлах. Текстовые базы данных приемлемы в системе с небольшим числом пользователей. Если в системе 25 тысяч пользователей, то для проверки информации при регистрации потребуется уйма времени.

Поэтому во FreeBSD используются два дополнительных файла **/etc/pwd.db** и **/etc/spwd.db**. Они представляют собой ассоциативные таблицы в формате **db**, отвечающие файлам **/etc/passwd** и **/etc/master.passwd**, включая и права доступа. Они обеспечивают быстрый механизм поиска в больших базах данных и автоматически генерируются программой **pwd_mkdb** при изменении информации об учетных записях посредством команд **chfn**, **passwd** или **adduser/rmuser**.

Команда **chfn** (change full name — изменить полное имя) представляет собой средство для изменения информации о пользователе. Как и команда **edquota**, речь о которой шла в главе 9, **chfn** запускает текстовый редактор, имя которого задано в переменной **EDITOR** (по умолчанию — **vi**). Он позволяет изменить любое из текстовых полей, а после сохранения файла и выхода программа перезаписывает файл **/etc/master.passwd** и автоматически запускает **pwd_mkdb -p** для обновления остальных трех файлов.

Важно отметить, что главным файлом базы данных является **/etc/master.passwd**. Если необходимо перестроить список пользователей или перенести его с другой FreeBSD-машины, можно просто разместить новый файл **master.passwd** в каталоге **/etc** и запустить следующую команду:

```
# pwd_mkdb -p /etc/master.passwd.new
```

В этом примере предполагается, что новый файл размещен в каталоге **/etc** под именем **master.passwd.new**. Файл **/etc/master.passwd** будет заменен новым, а **/etc/pwd.db**, **/etc/spwd.db** и **/etc/passwd** перестроены заново. Опция **-p** указывает программе **pwd_mkdb** генерировать новый файл **/etc/passwd**; если она опущена, **/etc/passwd** не претерпевает никаких изменений. Рекомендуется всегда использовать опцию **-p**, чтобы файлы оставались синхронизованными.

ПРИМЕЧАНИЕ

Во FreeBSD файл **/etc/passwd** не используется, он предназначен лишь для совместимости со сторонними приложениями. Большинство современных утилит черпает информацию о пользователях из файла **/etc/pwd.db**.

Файл /etc/group

Действия с группами похожи на действия с пользователями (текстовые базы данных содержатся в каталоге **/etc**), но поскольку группы обычно не имеют паролей, файл **/etc/group** (эквивалент файла **/etc/passwd** для групп) не требует дополнительных мер защиты (кроме ограничения прав на запись одним только **root**).

Ниже приведен пример строки из файла **/etc/group**. Обратите внимание, что она содержит лишь четыре поля: имя группы, "пустое" поле (в котором при необходимости сохраняются пароли), идентификатор группы и разделенный запятой список ее пользователей.

```
wheel:*:10:root,bob,frank
```

Файл **/etc/group** не имеет соответствующей ассоциативной базы данных, так как к информации в нем обращаться приходиться достаточно редко. Кроме того, в системе, как правило, бывает гораздо меньше групп, чем пользователей. Правило использования "уникальных групп" во FreeBSD делает последний аргумент спорным. Возможно, в следующих версиях системы появятся ассоциативные базы данных и для групп.

Управление группами

В силу отсутствия паролей, пользовательского доступа и ассоциативной базы данных, добавление группы в систему — дело чрезвычайно простое. Достаточно открыть файл **/etc/group** в любом текстовом редакторе и создать новую строку в формате, приведенном выше. Важно присвоить новой группе уникальный идентификатор.

Новые группы добавляются автоматически сценарием **adduser** в режиме создания уникальных групп для каждого пользователя. Обычно идентификатор такой группы совпадает с идентификатором пользователя, но это не обязательно. Скорее всего, вам потребуется добавлять только группы системного назначения, например, для таких задач, как запуск Web-сервера или работа с базой данных. Идентификатор таких групп обычно должен принимать значения в диапазоне от 100 до 1000. Числа, большие 1000, используются для уникальных групп (для совпадения с идентификаторами соответствующих пользователей). Числа, меньшие 100, принадлежат группам, являющимся частью операционной системы.

Для добавления пользователя в группу достаточно добавить его имя в четвертом поле. Если в нем уже содержатся какие-либо имена, их необходимо разделять запятой, как показано ранее. Для удаления пользователя из группы нужно просто удалить его имя.

11

ГЛАВА

Конфигурация системы и стартовые сценарии

- Понимание процесса начальной загрузки FreeBSD
- Сценарии конфигурирования ресурсов
- Демон **inetd** и файл конфигурации **inetd.conf**
- Система ведения журналов (**syslogd**) и файл **syslog.conf**
- Заметки о файле **/etc/rc.local**

В процессе начальной загрузки операционная система FreeBSD, прежде чем предоставить пользователю доступ к командной строке, выдает немало информации о системе. Этот процесс значительно сложнее, чем в таких системах, как Windows или Mac OS. Тем не менее, когда система настроена правильно и в ней нет аппаратных конфликтов, полная загрузка (включая запуск всех служб, или "демонов") происходит достаточно быстро.

Поскольку все UNIX-подобные операционные системы слегка отличаются друг от друга, процесс начальной загрузки FreeBSD может показаться незнакомым даже ветеранам Linux. Так, например, утилита **fsck** не имеет удобного индикатора состояния проверки или опций, имеющихся в аналогичной утилите из дистрибутивов Linux. А boot-менеджер работает несколько иначе, чем LILO (загрузчик Linux). Кроме того, FreeBSD не имеет столь большого числа уровней исполнения (runlevel), как Linux или Solaris. Поскольку модель FreeBSD проще и прямолинейнее, ее легче понять, с другой стороны, она не имеет ряда возможностей, имеющихся в других системах.

Понимание процесса начальной загрузки FreeBSD

Процесс начальной загрузки (*bootstrapping process*) FreeBSD включает множество этапов, каждый из которых имеет строго определенное назначение. Не завершив любой из них, нельзя перейти к следующему. Все эти этапы мы рассмотрим в данной главе.

При включении машины выполняется проверка оборудования и установка конфигураций, заданных в BIOS и CMOS. Процедура проверки оборудования запускает тест памяти, а также поиск устройств IDE или SCSI. Информация о ходе этого процесса отображается на экране, после чего он очищается. Этот шаг не зависит от операционной системы — он происходит всегда, какая бы система ни была установлена. После этого на экран выводится таблица с данными о найденных устройствах и настройках. Затем BIOS читает главную загрузочную запись (Master Boot Record) (MBR) ведущего устройства первого контроллера в поисках информации о первом загрузочном блоке. Задача MBR и первых двух загрузочных блоков — запустить загрузчик, который конфигурирует ядро. Каждый из последующих загрузочных блоков несколько сложнее, чем предыдущий. Размер двух первых из них ограничен 512 байтами (т.е. размером MBR и загрузочного сектора слайса). Ниже мы рассмотрим назначение всех блоков.

- Загрузочный блок 0 (**boot0**). Этот первичный блок находится в MBR и содержит список доступных слайсов, из числа которых с помощью функциональных клавиш можно выбрать нужный для загрузки.

```
F1  FreeBSD
F2  Linux
F3  ??

Default:  F1
```

  Нажатие соответствующей функциональной клавиши позволяет выбрать нужный слайс. Если подождать несколько секунд, то система выберет слайс, установленный по умолчанию.

- Загрузочный блок 1 (**boot1**). Он содержит несложную программу, которая запускается из загрузочного сектора слайса (slice), выбранного в **boot0**. Про-

грамма использует сокращенную версию утилиты **disklabel**, разбивающую слайс на разделы (о чем подробно рассказано в главе 19) для нахождения и запуска загрузчика **boot2**. Заметьте, что **boot1** не имеет пользовательского интерфейса.

- Загрузочный блок 2 (**boot2**). Наконец, система достигает загрузочного блока, которому отведено достаточно места, чтобы он мог содержать код основного загрузчика, который умеет читать файлы из загружаемой файловой системы. В ранних версиях FreeBSD этот блок выдавал запрос, позволяющий указать, откуда загружать ядро (кроме установки по умолчанию). Теперь он автоматически запускает программу, называемую *загрузчиком* (*loader*), которая поддерживает пользовательский интерфейс и имеет другие возможности.
- Загрузчик (loader). Эта программа находится в каталоге **/boot**. Она читает конфигурационные файлы **/boot/defaults/loader.conf** и **/boot/loader.conf**, а затем загружает ядро и указанные в них модули. (Файл **/boot/loader.conf** содержит установки, отменяющие **/boot/defaults/loader.conf**, и работает подобно файлу **/etc/rc.conf**, о котором рассказано далее в этой главе.)

  Загрузчик ожидает нажатия клавиши в течение 10 секунд. Если его не последует, он загружает ядро в состоянии, заданном по умолчанию. Если нажать Enter, программа предложит интерфейс командной строки, с помощью которого можно полностью управлять процедурой загрузки ядра: для запуска в *однопользовательском режиме* (*single-user mode*) наберите **boot -s**, для загрузки предыдущего ядра — **boot kernel.old**, а для загрузки с CD-ROM — **boot -C**. Кроме того, можно по одному загружать или выгружать модули, просматривать содержимое файлов (**more**) и выполнять другие задачи (подробности вы найдете в **man boot**). В большинстве случаев обращаться к этим командам нет необходимости. Обычно система загружается в состоянии по умолчанию. Пусть, например, требуется загрузить определенный модуль ядра (скажем, **/modules/portal.ko**) на этапе загрузки, до того как система загрузится целиком. Кроме того, необходимо просмотреть все модули, находящиеся в памяти на текущий момент. Все это можно выполнить посредством приглашения **ok**, как показано ниже:

```
ok load portal.ko
/modules/portal.ko   text=0x1d18   data=0x1f4+0x4
syms=[0x4+0x8d0+0x4+0x6bf]
ok  lsmod
  0x100000:  kernel  (elf  kernel,  0x355be4)
  0x455be4:  /boot/kernel.conf  (userconfig  script,  0x4c)
  0x456000:  portal.ko  (elf  module,  0x3eb0)
```

  Нажатие клавиши **?** в приглашении выдает список доступных команд. В примере были продемонстрированы команды **load** (предназначена для загрузки модулей) и **lsmod** (выводит на экран текущий список модулей, находящихся в памяти). Эта информация чрезвычайно полезна, если возникнут проблемы, а также в ситуации, когда имеет значение порядок загрузки модулей. Более подробно о возможностях командной строки загрузчика смотри в справочном руководстве **man loader**.

На этом фаза процесса загрузки, связанная с загрузочными блоками, завершается. На финальной стадии управление берет система FreeBSD: ядро загружается в память, проверяет все доступные устройства, выполняет сценарии *конфигурирования*

ресурсов (resource configuration), создающие необходимую рабочую среду, и запускает различные системные службы.

- Ядро (kernel). Когда загрузчик закончит свою работу и передаст управление следующей стадии, ядро начинает проверять все найденные устройства и выводить результаты на экран. Именно на этой стадии на экран выводится основная масса сообщений. Все сообщения заносятся в файлы **dmesg**, прочесть которые позволяет команда **dmesg**.

СОВЕТ

dmesg представляет собой довольно примитивное средство, которое лишь выводит на экран все сообщения, накопленные в процессе загрузки системы. Достаточно просто ввести команду **dmesg**, или **dmesg | less** для более удобного чтения.

- **init**. После того как ядро полностью загружено в память, управление передается процессу **init**, и начинается заключительная стадия процедуры загрузки. Об этом говорит сообщение Automatic reboot in progress (Происходит автоматическая перезагрузка). При этом **init** запускает Resource Configuration script — сценарий конфигурирования ресурсов (**/etc/rc**). Этот сценарий в первую очередь проверяет целостность всех файловых систем, указанных в файле **/etc/fstab**, о чем говорилось в главе 9.

  Если утилита **fsck** не находит таких проблем, которые она не может решить самостоятельно, она монтирует все файловые системы (**mount -a -t nonfs**) и процесс загрузки продолжается. Если же **fsck** сталкивается с нерешаемой проблемой, система переходит в однопользовательский режим, который позволяет запустить **fsck** вручную и устранить повреждения. Чтобы продолжить загрузку системы, из однопользовательского режима следует выйти.

Если все идет по плану, система выдает приглашение для регистрации в системе. Весь этот процесс, как правило, занимает не больше минуты.

ЗАЩИТА ПРОЦЕССА ЗАГРУЗКИ

Следует отметить, что по умолчанию, когда система входит в однопользовательский режим (**boot -s** в командной строке загрузчика **loader**), она не запрашивает пароль, предоставляя при этом доступ с правами пользователя **root**. Если кто-либо, кроме администратора, имеет возможность работать непосредственно на консоли и перезагружать машину, то система имеет очень серьезную дыру в защите.

К счастью, эту прореху легко заштопать. В файле **/etc/ttys**, содержащем терминальные настройки для различных способов доступа (консоль, виртуальные терминалы, последовательные терминалы и сетевые (псевдо) терминалы), необходимо заменить установку **secure** на **insecure** в строке **console**:

```
console none                unknown off insecure
```

Эти установки сообщают системе, что консоль является незащищенной точкой доступа. В результате система требует ввода пароля при загрузке в однопользовательском режиме.

ПРИМЕЧАНИЕ

Терминальный метод **console** используется только в однопользовательском режиме. При загрузке в полном многопользовательском режиме на одном физическом терминале доступно несколько виртуальных терминалов (переключаться между ними позволяют комбинации клавиш Alt+F1, Alt+F2 и т.д.). Эти терминалы всегда предоставляет приглашение для регистрации в системе. В этом случае установки **secure** и **insecure** выполняют следующую функцию: при

попытке входа с правами **root** система требует зарегистрироваться в качестве обычного пользователя, а затем получить права доступа **root** посредством команды **su**.

А вот и вторая дыра в защите, которую желательно устранить, если обычные пользователи имеют доступ к физическому терминалу (даже в том случае, если консоль уже защищена). Проблема в том, что нажатие комбинации клавиш Ctrl+Alt+Delete заставляет систему перезагрузиться независимо от прав доступа. Это можно сделать в любой момент. В некоторых случаях такое поведение системы не является предметом для опасений (например, если сервер заперт в отдельной комнате), однако в большинстве ситуаций так оставлять нельзя. Чтобы устранить такое неприятное явление необходимо заново собрать ядро, предварительно включив следующую опцию в его конфигурации:

```
options                 SC_DISABLE_REBOOT
```

Подробный рассказ об опциях ядра и его конфигурировании вы найдете в главе 17.

Что может замедлить процесс загрузки?

В некоторых случаях загрузка системы требует больше времени. Поскольку процесса загрузки выполняется поэтапно, несложно установить, где именно возникла проблема. Чаще всего, она связана с **sendmail** или **httpd** (Apache); в обоих случаях причиной является неправильная работа сетевого соединения или неправильная конфигурация параметров сети.

Обоим системам, и Sendmail, и Apache, необходимо установить имя хоста локальной машины; для этого им требуется реверсивный поиск в DNS, сконфигурированном в **/etc/resolv.conf**. (Настройка TCP/IP изложена в главе 23.) Если сеть настроена неправильно, система, прежде чем сообщить о проблеме, довольно долго пытается что-то сделать, после чего процесс загрузки продолжается. Сетевые тайм-ауты зачастую достаточно длительны, поэтому именно сеть — самая распространенная причина замедления процесса загрузки. Что можно в этом случае предпринять? Постарайтесь, чтобы сервер, указанный первым в файле **/etc/resolv.conf**, был всегда доступен с FreeBSD-машины; в этом случае Sendmail и Apache быстро определят имя хоста локальной машины и запустятся без задержки. Если это невозможно, в качестве первого сервера имен следует указать 127.0.0.1 (**localhost**) и запретить остальные.

Между прочим, именно поэтому процесс **init** запускает команду **mount -a -t nonfs** на начальном этапе подготовки загрузки в многопользовательском режиме. Сетевые файловые системы (NFS) имеют очень большой тайм-аут, а процесс, ожидающий его окончания, практически невозможно прекратить. Команда **mount**, с которой мы уже сталкивались в главе 9, позволяет избежать начального монтирования ресурсов NFS (перенося этот процесс на более позднюю стадию). Однако процесс **init** монтирует все файловые системы, включая и NFS, во время выполнения сценария **/etc/rc**. Все ресурсы NFS, указанные для автоматического монтирования в файле **/etc/fstab** (например, без опции **noauto**) имеют потенциальную возможность надолго затормозить загрузку системы. Если вы не уверены в постоянной доступности ресурсов NFS, никогда не монтируйте их автоматически на этапе загрузки!

Сценарии конфигурирования ресурсов

Все файлы в каталоге **/etc**, имя которых начинает с **rc**, являются сценариями конфигурирования ресурсов. Иначе говоря, это программы, запускающие какую-

либо часть системы FreeBSD в соответствии с конфигурацией. Одни из них вызываются рекурсивно из каких-либо программ, другие не выполняют никаких задач в отдельной конфигурации, а некоторые и вообще никогда не запускаются. Тем не менее в **/etc** присутствует один **rc**-файл, который всегда подвергается редактированию для изменении поведения системы при загрузке. Это **/etc/rc.conf**. Все сценарии конфигурирования ресурсов, используемые во FreeBSD, перечислены в табл. 11.1.

Таблица 11.1 Сценарии конфигурирования ресурсов

Имя сценария	*Описание*
/etc/rc	Главный сценарий **config**.
/etc/rc.diskless1	Процесс **init** читает этот сценарий при бездисковой загрузке по протоколу BOOTP.
/etc/rc.diskless2 /etc/defaults/rc.conf	Процесс **init** читает этот файл на раннем этапе, внося в список задачи, обязательные к исполнению.
/etc/rc.conf	Этот файл отменяет настройки по умолчанию из **/etc/defaults/rc.conf**. Это ЕДИНСТВЕННЫЙ файл в каталоге **/etc**, который может подвергаться редактированию!
/etc/rc.sysctl	Устанавливает переменные ядра. По умолчанию он ничего не выполняет.
/etc/rc.serial	Настраивает терминалы и другие последовательные устройства.
/etc/rc.pccard	Запускает демон PC-карт для ноутбуков.
/etc/rc.network	Настраивает работу сети по протоколу TCP/IP.
/etc/rc.network6	Совпадает с сценарием **rc.network**, а также содержит службы IPv6.
/etc/rc.atm	Запускается сценарием **rc.network**; устанавливает ATM-службы для машин WAN.
/etc/rc.firewall	Запускается сценарием **rc.network**; конфигурирует брандмауэр **ipfw**.
/etc/rc.firewall6	Запускается сценарием **rc.network6**; конфигурирует брандмауэр **ip6fw**.
/etc/rc.i386	Установки, специфичные для платформы x86 (например, опции консоли и APM).
/etc/rc.shutdown	Запускается процессом **init** на этапе останова системы.
/etc/rc.suspend	Сценарии, используемые демоном управления питанием APM.
/etc/rc.devfs	Конфигурирует файловую систему устройств (device filesystem).
/etc/rc.local	Устаревший метод добавления пользовательских расширений сценариев конфигурирования. Вместо этого необходимо *использовать метод* ***rc.d***!
/usr/local/etc/rc.d/ /usr/local/X11R6/etc/rc.d/	Дерево каталогов, содержащее новые сценарии, добавляемые пользователем (или автоматически инсталлируемые программами).

Из перечисленных файлов нас интересуют только **/etc/defaults/rc.conf**, **/etc/rc.conf** и дерево каталогов **rc.d**. Все остальное трогать не следует, дабы последующие инсталляции FreeBSD могли обновлять конфигурацию, сохраняя пользовательские настройки.

Файл /etc/defaults/rc.conf

Взгляните на сценарий **/etc/rc**. Легко видеть, что он полностью автоматизирован; принцип его работы заключается в проверке определенных переменных или файлов, управляющих конфигурацией системы; если они существуют, сценарий запускает отдельный цикл, получающий параметры из этих переменных и файлов. Содержимое файла **/etc/rc** редактировать не следует. В других системах изменения в процессе начальной загрузки достигаются путем внесения изменений в сценарии конфигурирования ресурсов. Модель FreeBSD предполагает максимально возможный уровень автоматизации и абстрагирования, при котором настройки сосредоточены в файлах, в которых указаны лишь параметры.

В ранних версиях FreeBSD существовал только один конфигурационный файл — **/etc/rc.conf**. В нем содержались все переменные, используемые сценарием **/etc/rc** и другими сценариями. Поэтому любая модификация системы означала изменение этого файла. Вскоре такая схема стала неуправляемой, поскольку число переменных выросло, и значение этого файла возросло. Администратору, обновляющему систему, приходилось ценой больших усилий объединять старый вариант **rc.conf** с новым. Практически, это ни чем не отличалось от прямого редактирования файла **/etc/rc**.

Решение состояло в создании каталога **/etc/defaults**, в котором была размещена копия **rc.conf** со всеми настройками по умолчанию. Теперь файл **/etc/rc.conf** все еще существует, но он может быть пустой, тем не менее система все равно будет загружаться. Этот файл предназначен для внесения настроек, которые отменяют действие **/etc/defaults/rc.conf**. Обычно они связаны с сетевой конфигурацией (IP-адрес, имя хоста, адрес шлюза и т.д.) и запуском демонов, например, **sendmail** и **sshd**.

Рассмотрим типичный блок кода из файла **/etc/defaults/rc.conf**, приведенный в листинге 11.1.

Листинге 11.1 Фрагмент файла /etc/defaults/rc.conf

```
# Демон named можно запустить в режиме sandbox, подробнее см. в man
#  security.
#
named_enable="NO"           # Запустить named, или DNS-сервер (или NO
                            # (НЕТ)).
named_program="named"       # путь к named, если требуется другая его
                            # копия.
named_flags=""                          # Флажки named.
#named_flags="-u  bind  -g  bind"       # Флажки named.
```

Из приведенного сценария видно, что сценарий **/etc/rc** делает по умолчанию, — он вообще не запускает демон **named**.

Обычно переменные в файле **/etc/defaults/rc.conf** группируются в подобные блоки, с общим префиксом и значением **YES/NO**. Как правило, первая переменная служит главным "переключателем". Если она равна **NO**, то значения остальных переменных игнорируются; если она равна **YES**, то рассматриваются только незакомментированные переменные (например, вторая строка **named_flags** в предыдущем примере).

Файл /etc/rc.conf

Предположим, что требуется запустить **named**. В простейшем случае все, что требуется, — это просто отредактировать файл **/etc/rc.conf** (файл, отменяющий настройки по умолчанию) и добавить в него следующую строку:

```
named_enable="YES"
```

Остальные переменные **named_*** копировать в файл **/etc/defaults/rc.conf** нет необходимости; помните, что каждая переменная из файла по умолчанию загружается в память процессом **init**, поэтому важно лишь, чтобы главный "переключатель" был установлен в значение **YES**. Если это так, все переменные будут использоваться при последующем запуске цикла в сценарии **/etc/rc** (в данном случае для **named**).

Другими переменными можно воспользоваться для более тонкой настройки. Предположим, программа сервера имен называется **mynamed**. Кроме того, нужно создать пользователя и группу **bind**, от имени которых будет работать сервер (это делается в целях безопасности). Все, что нужно сделать (предполагая, что **mynamed** имеет такие же опции командной строки, как и **named**), — это добавить в файл **/etc/rc.conf** две следующих строки:

```
named_program="mynamed"
named_flags="-u  bind  -g  bind"
```

Любопытных приглашаем посмотреть, что с этими переменными делает процесс **init**:

```
#  grep  "named_enable"  /etc/rc*
/etc/rc.network:          case  ${  named_enable}     in
```

Итак, сервер запускается из сценария **/etc/rc.network**. В этом файле несложно найти фрагмент кода, выполняющий именно эту задачу:

```
network_pass2()  {
        echo  -n  'Doing  additional  network  setup:'
        case  ${named_enable}  in
        [Yy][Ee][Ss])
                echo  -n  ' named';     ${named_program:-named}
➥${named_flags}
                ;;
        esac
```

Мы видим, что все переменные **named_*** из обоих файлов **rc.conf** участвуют в процессе; теперь при загрузке системы, когда на экране после строки **Doing additional network setup:** появляется **named**, это значит, что система применяет указанные настройки и автоматически запускает сервер имен.

ПРИМЕЧАНИЕ

О создании сценариев командного интерпретатора читайте в главе 13. Это поможет вам в дальнейшем при изучении сценариев конфигурирования ресурсов.

Чаще всего, в файле **/etc/rc.conf** содержатся переменные, отвечающие за конфигурацию TCP/IP. Очевидно, что они различны для разных систем, поэтому настройки FreeBSD по умолчанию не подходят. В листинге 11.2 показано содержимое файла **/etc/rc.conf** сразу после новой инсталляции системы:

Листинг 11.2 Файл /etc/rc.conf после новой инсталляции

```
# Этот файл содержит настройки, отменяющие /etc/defaults/rc.conf
# Пожалуйста, вносите изменения только в этот файл.

# Включение сетевых демонов.
# -- sysinstall generated deltas -- #
kern_securelevel="1"
kern_securelevel_enable="YES"
linux_enable="YES"
sendmail_enable="YES"
sshd_enable="YES"
portmap_enable="NO"
nfs_server_enable="NO"
inetd_enable="NO"
network_interfaces="fxp0 lo0"
ifconfig_fxp0="inet 10.6.7.101 netmask 255.0.0.0"
defaultrouter="10.6.1.1"
hostname="freebsd1.testnetwork.com"
usbd_enable="YES"
```

Некоторые из этих переменных являются избыточными по отношению к файлу с настройками по умолчанию, тем не менее их полезно иметь в этом файле, поскольку многие из свойств (например, NFS-сервер) теперь управляются из одного файла.

Многие программы при инсталляции настраивают свою загрузку самостоятельно; однако файл **/etc/rc.conf** не предназначен для них. Предполагается, что изменения в него может вносить только администратор и программа **sysinstall** (когда она вносит изменения в основную часть системы). Для программ, инсталлируемых пользователем (портов и пакетов), а также для сценариев, о которых файл **/etc/rc** не знает, существует другая система размещения сценариев начальной загрузки и конфигурационных файлов — иерархия **/usr/local/etc**.

Каталоги /usr/local/etc и /usr/local/X11R6/etc

Помните, в главе 9 было сказано: "Все, что устанавливает администратор, размещается в каталоге **/usr/local**"? Это справедливо не только для программ и разделяемых библиотек, но и для сценариев начальной загрузки. Каталог **/usr/local/etc** является локальным эквивалентом **/etc**, и именно в нем размещаются все конфигурационные файлы программ, устанавливаемых администратором.

ПРИМЕЧАНИЕ

Важно отметить, что разница между файлами в каталоге **/etc** и каталоге **/usr/local/etc** заключается *не* в том, что первые нельзя редактировать, а вторые можно. Многие файлы из **/usr/local/etc** точно так же не подвергаются редактированию, поскольку не содержат каких-либо пользовательских конфигураций или не предполагают изменения (хотя этот каталог включает и файлы, предназначенные для настройки конфигураций). Различие заключается в том, что файлы из **/etc** управляют основой системы, только теми программами, которые являются частью инсталляции FreeBSD. Файлы же из **/usr/local/etc** содержат конфигурации программ, инсталлируемых администратором из коллекции портов или других пакетов. Кроме того, каталог **/usr/local/etc** служит для хранения начальных сценариев, которые вы будете писать сами.

Локальный каталог **/usr/local/etc** содержит конфигурационные файлы, редактируемые администратором и предназначенные для настройки соответствующих программ. Подробнее об этом говорится в главе 15, где речь пойдет о портах и пакетах.

В данный момент нас интересует подкаталог **rc.d** (аббревиатура от Resource Configuration for Daemons — Конфигурация ресурсов для демонов).

Процесс **init** обращается к каталогу **/usr/local/etc/rc.d** после того, как запускает все сценарии **/etc/rc.***. Все выполняемые файлы из этого каталога, имеющие расширение **.sh**, запускаются в алфавитном порядке. Примеры файлов, устанавливаемых в этот каталог, включают **apache.sh**, **mysql-server.sh** и **samba.sh**. Эти сценарии являются настраиваемой частью портов или пакетов, причем каждый из них воспринимает аргументы **start** и **stop**; когда **init** запускает каждый сценарий, используется аргумент **start**. Заметьте, что сценарии из этого каталога можно выполнять и на этапе исполнения, например, для запуска новой службы без перезагрузки всей системы:

```
# /usr/local/etc/rc.d/apache.sh start
```

Некоторые порты или пакеты содержат файлы с суффиксом **.sample** (например, **samba.sh.sample**). Дело в том, что многие пакеты для успешного запуска требуют специальной настройки. Apache, например, запускается сразу же после инсталляции, не требуя какой-либо модификации в файлах **config** (хотя позже вам, без сомнения, придется их изменять). Поэтому этот пакет устанавливает файл сценария **apache.sh**, который может запустить программу после первой же перезагрузки. С другой стороны, пакет Samba необходимо вначале настроить; если выполнить сценарий сразу после инсталляции, он не сможет запустить демон. Поэтому нужно внести требуемые изменения, а затем переименовать файл (без **.sample**), чтобы к нему мог обратиться начальный процесс **init**.

Каталог **/usr/local/X11R6/etc** полностью аналогичен **/usr/local/etc**, за тем исключением, что он предназначен для программ с интерфейсом X11 — панелей GNOME, графических средств, игр, менеджеров окон и т.д. В этом каталоге также имеется подкаталог **rc.d**. Сценарии из него запускаются немедленно после обращения к файлам из **/usr/local/etc/rc.d**. Локальные каталоги начальных сценариев конфигурируются. Для этого необходимо внести требуемые каталоги в строку **rc.conf**:

```
local_startup="/usr/local/etc/rc.d /usr/X11R6/etc/rc.d" # каталоги с
➥начальными сценариями.
```

Создание сценариев для запуска программ при загрузке системы

Только большой мечтатель может решить, что ему не придется настраивать систему для выполнения задач начальной загрузки. Вам наверняка потребуется запускать собственный демон или очищать каталог общего доступа при каждой загрузке системы. Для этого достаточно написать собственный сценарий и поместить его в каталог **/usr/local/etc/rc.d**.

Помните, что процесс **init** требует, чтобы всякий сценарий из каталога **rc.d** воспринимал аргумент **start** и имел расширение **.sh** (иначе он не будет выполняться автоматически). Ниже приведен пример сценария со страницы справочного руководства **man rc**:

```
#!/bin/sh -
#
#       сценарий инициализации/останова пакета foobar

case "$1" in
start)
        /usr/local/sbin/foo -d && echo -n ' foo'
```

```
        ;;
*)
        echo "unknown option: $1 - should be 'start'" >&2
        ;;
esac
```

Файл сценария можно назвать, например, **foo.sh**. Убедитесь, что он на самом деле выполняемый (**chmod +x foo.sh**)!

С технической точки зрения программе не обязательно следует быть сценарием командного интерпретатора: ее можно написать на языке perl, C или любом другом. Главное, чтобы выполняемый файл имел расширение **.sh**. Тем не менее не стоит обманывать систему! Мы вернемся к созданию сценариев командного интерпретатора в главе 13.

Демон inetd и файл конфигурации inetd.conf

Хотя основная часть системы включает в себя достаточно много демонов, имеющих файлы **.conf** в каталоге **/etc**, наиболее важный среди них — **inetd**, суперсервер. Давайте обсудим его конфигурирование.

Задача **inetd** заключается в прослушивании соединений на указанных сетевых портах и запуск соответствующих процессов сервера в случае поступления запросов. Например, **inetd** отвечает за **telnet**-соединения. Если система позволяет подключаться по протоколу **telnet**, можно открыть соединение и получить приглашение для регистрации, не запуская предварительно на сервере процесса **telnetd**. Каждый раз, когда система получает запрос на соединение к порту 23, она создает новый процесс **telnetd** для его обработки. Выполняемые программы, запускаемые демоном **inetd** (и подобными ему), находятся в каталоге **/usr/libexec**. Путь к этим программам обычно не включен в настройку, так как запускать их из командной строки не предполагается; обычно они порождаются другими процессами, при этом им передаются определенные ресурсы (например, переменные среды и сетевые соединения).

ПРИМЕЧАНИЕ

Использование демона **inetd** исключает необходимость выполнять главный процесс **telnetd**, представляющий опасность с точки зрения безопасности, если в **telnetd** обнаружится уязвимое место. Многие демоны (**sshd**, **httpd**, **sendmail** и другие) запускаются в "самостоятельном" режиме, а не вызываются демоном **inetd**. Главный процесс (выполняемый с правами **root**) прослушивает новые соединения и порождает новые процессы, которыми владеет пользователь, который не имеет прав доступа. Это повышает гибкость и скорость программы, но создает определенный риск для безопасности системы. **inetd** также запускается с правами доступа **root**, поэтому его взлом столь же опасен для системы. Тем не менее, чем больше демонов выполняется с такими правами, тем больше вероятность найти дыру в защите.

Просмотр файла **/etc/inetd.conf** показывает, что почти все записи в нем запрещены. Разрешенные перечислены в табл. 11.2.

Таблица 11.2 Системные службы, управляемые демоном inetd и разрешенные по умолчанию

Служба	*Описание*	*Используемые порты/ресурсы*
ftp	File Transfer Protocol (Протокол передачи файлов)	Порт 21/TCP
telnet	Удаленный терминал	Порт 23/TCP

Служба	*Описание*	*Используемые порты/ресурсы*
comsat	Сервер biff (уведомление пользователей о входящей почте)	Порт 512/UDP
ntalk	chat-сервер командной строки	Порт 518/TCP,UDP
ftp (IPv6)	File Transfer Protocol (Протокол передачи файлов)	IPv6
telnet (IPv6)	Удаленный терминал	IPv6

Другие службы, которые вам, скорее всего, понадобятся, перечислены в табл. 11.3.

Таблица 11.3 Другие полезные службы inetd

Служба	*Описание*	*Используемые порты/ресурсы*
pop3	Post Office Protocol (Почтовый протокол)	Порт 110/TCP
imap4	Interim Mail Access Protocol (Протокол доступа к почтовой службе)	Порт 143/TCP
smtp	Qmail (альтернатива SMTP-серверу Sendmail)	Порт 25/TCP
netbios-ssn netbios-ns	Разделяемый доступ к файлам на платформе Windows с помощью Samba	Порт 139/TCP Порт 137/TCP
finger	Просмотр информации о пользователе	Порт 79/TCP

Для включения какой-либо из этих служб, необходимо просто удалить символ комментария в начале строки и перезапустить сервер **inetd** следующим образом:

```
# ps -waux | grep inetd
root   110   0.0   0.6  1032  752   ??   Ss   11:57PM   0:00.01  inetd
# kill -HUP 110
```

СОВЕТ

Если система работает на уровне безопасности 1 или выше (данная опция выбирается при инсталляции, уровень 1 называется Medium — средний), то демон **inetd** не будет запущен. Это еще раз подтверждает, насколько опасны службы, выполняющиеся в **inetd**. Когда система работает на этом уровне защиты, демон все же можно запустить с помощью команды **inetd -wW**. Чтобы разрешить его постоянную работу, из файла **/etc/rc.conf** нужно устранить следующую строку:

```
inetd_enable="NO"
```

Демон **inetd** представляет собой одну из областей FreeBSD, где нет полной автоматизации. Такой подход направлен на предотвращение неправильных конфигураций. Если вам потребуется разрешить службы в файле **/etc/inetd.conf**, помните, что вы создаете нестандартную конфигурацию, которая требует дополнительного внимания. Например, службы **cvs** содержат предупреждение о дыре в защите, причиной которой может стать неправильное значение параметра. Службы Samba (**netbios-ssn** и **netbios-ns**) рассчитывают, что выполняемые файлы **smbd** и **nmbd** находятся в каталоге **/usr/local/sbin**, однако они присутствуют там лишь в том случае, если система Samba установлена из портов или пакетов. Запуск Samba из **inetd**, а не в самостоятельном режиме также представляет собой альтернативную, нестандартную конфигурацию.

Подобным образом и другие службы (например, **pop3**) пытаются запускать файлы, установленные в каталоге **/usr/local/libexec**. Поэтому помните, что раз этот каталог находится внутри **/usr/local**, то в нем находятся только те программы, которые были явно установлены администратором. Инсталляция порта **pooper** разместит в этом каталоге необходимый выполняемый файл и службу можно будет разрешить в конфигурации **inetd**. Но если вы выберете пакет **qpopper** (еще один вариант POP3-сервера), выполняемый файл будет иметь имя **qpopper** вместо **popper**. Поэтому строку в конфигурационном файле нужно будет изменить соответствующим образом:

```
pop3 stream tcp   nowait root /usr/local/libexec/qpopper qpopper
```

Существует множество разных ловушек для неопытных сисадминов. Поэтому старайтесь не изменять службы **inetd** больше, чем это действительно требуется. Подробную информацию о синтаксисе и методах работы с **inetd** ищите в **man inetd**.

Система ведения журналов (syslogd) и файл syslog.conf

Системные сообщения записываются в файл **/var/log**. Запись обеспечивает механизм, называемый **syslogd**, или демон системной журнализации. Его поведением управляет файл **/etc/syslog.conf**, в котором заданы log-файлы разных служб. Каждая служба, о которой знает демон (это может быть **auth**, **authpriv**, **console**, **cron**, **daemon**, **ftp**, **kern**, **lpr**, **mail**, **mark**, **news**, **ntp**, **security**, **syslog**, **user**, **uucp** и **local0** — **local7**) имеет несколько уровней "строгости", которые позволяют управлять ведением журналов. Эти уровни (приведены в порядке убывания строгости) — **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info** и **debug**.

Каждый демон или служба, выполняемые во FreeBSD, могут заносить информацию в log-файлы с помощью демона **syslogd**; Sendmail и другие почтовые программы с помощью процедур **syslog()** могут посылать сообщения, используя средство **mail**; обработка сообщений демонов системой **syslogd** определяется файлом **syslog.conf**.

По умолчанию в файле **syslog.conf** определено несколько правил ведения журналов:

```
*.err;kern.debug;auth.notice;mail.crit          /dev/console
*.notice;kern.debug;lpr.info;mail.crit;news.err    /var/log/messages
security.*                                      /var/log/security
mail.info                                       /var/log/maillog
lpr.info                                        /var/log/lpd-errs
cron.*                                          /var/log/cron
*.err                                           root
*.notice;news.err                               root
*.alert                                         root
*.emerg                                         *
```

Они означают: сообщения **err** от всех служб, **debug** от ядра, сообщения авторизации **notice**, а также сообщения **crit** почтовых программ выводятся на системную консоль (т.е. их можно просматривать на мониторе, подключенном к FreeBSD-машине). Аналогично все сообщения, связанные с защитой, направляются в файл **/var/log/security**, а все сообщения от почтовых программ уровня **info** записываются в файл **/var/log/maillog**. Все остальное сохраняется в файле **/var/log/messages** — основном log-файле системы. (Если вы работали с Linux, то заметите, что этот файл является эквивалентом **syslog**.)

Некоторые типы сообщений не записываются в log-файлы, а направляются механизмам обработки других типов. В установках **syslog.conf** по умолчанию задано, что

сообщения всех служб уровней **err**, **notice** и **alert** печатаются на терминале пользователя **root**, а сообщения **emerg** — на терминалах всех пользователей. В таблице 11.4 показаны возможные действия для сообщений **syslogd** и синтаксис для них.

Таблица 11.4 Синтаксис для действий syslogd

Синтаксис	*Действие*
/path/to/file	Сообщения записываются в указанный файл.
@some.hostname.com	Сообщения перенаправляются демону **syslogd** на машине **some.hostname.com** (с использованием сетевой службы **syslog**).
user1	Сообщения выводятся на любой терминал, с которого пользователь **user1** зарегистрировался в системе.
root,user1,user2	Сообщения отображаются на терминалах всех указанных пользователей.
*	Сообщения выводятся всем зарегистрированным пользователям.
| "mail root"	Сообщения пересылаются пользователю **root** по электронной почте.

Подробную информацию о конфигурировании **syslogd** можно найти в справочных руководствах **man syslogd** и **man syslog.conf**.

ПРИМЕЧАНИЕ

Каждый log-файл из каталога **/var/log** обновляется в соответствии со своим набором правил. Например, файл **/var/log/maillog** архивируется и создается заново каждый день программой **periodic**. Остальные файлы, например **/var/log/cron** и **/var/log/messages**, обновляются другими методами (зачастую самими программами, которые записывают информацию в них). При архивировании, как правило, используется утилита **gzip**. Для поиска информации в заархивированных файлах следует воспользоваться командой **gzcat** в сочетании с обычной утилитой **grep**:

```
# gzcat /var/log/messages.2.gz | grep "rejected"
```

Заметки о файле /etc/rc.local

Сценарий конфигурирования ресурсов **/etc/rc.local**, упомянутый ранее, ныне фактически устарел. С этим файлом хорошо знакомы администраторы, использовавшие ранние версии дистрибутивов FreeBSD или Linux. Он служил той же цели, что и каталоги **/usr/local/etc/rc.d**, — обеспечивал механизм, изменяющий поведение системы при загрузке.

Метод **rc.d**, описанный ранее, оказывается более предпочтительным для решения этой задачи. Он лучше структурирован: все задачи решаются отдельными сценариями с общим интерфейсом, расположенными в каталоге **/usr/local** за пределами **/etc**. Тем не менее, если вам нужно использовать сценарий **/etc/rc.local**, он все еще поддерживается, хотя и не создается в инсталляции по умолчанию. Вы можете сами создать этот файл и разместить в нем любой набор команд интерпретатора. Обратите внимание, что файл запускается до других сценариев из каталога **rc.d**.

Поддержка **rc.local** существует лишь для обеспечения обратной совместимости, и, возможно, будет когда-нибудь устранена из FreeBSD. Поэтому прежде всего нужно проверить поддержку **rc.local** системой в файле **/etc/rc**, как показано в листинге 11.3.

Листинг 11.3 Фрагмент файла /etc/rc, демонстрирующий поддержку /etc/rc.local

```
# grep -A 5 rc.local /etc/rc
# Используйте традиционный (хоть и устаревший) файл rc.local, если
# он существует. Если вы используете этот файл, скопируйте в него
# содержимое файла /etc/defaults/rc.conf и добавьте собственные
# переменные в файл /etc/rc.conf, как показано ниже. Не пытайтесь
# разместить локальные расширения непосредственно в файле /etc/rc.
# Use /etc/rc.local
#
# ---- rc.local ----
#         if [ -r /etc/defaults/rc.conf ]; then
#                 . /etc/defaults/rc.conf
#                 source_rc_confs
#         elif [ -r /etc/rc.conf ]; then
#                 . /etc/rc.conf
--
# ---- rc.local ----
#
if [ -r /etc/rc.local ]; then
        echo -n 'starting local daemons:'
        sh /etc/rc.local
        echo '.'
fi

# В каждом корректном каталоге из $local_startup, произвести поиск
# сценариев с расширением *.sh
#
```

Здесь нас в первую очередь интересует код, запускающий **sh /etc/rc.local**. Если он содержится в этом файле и не закомментирован, можно использовать файл **rc.local**. Но лучше использовать более современный метод — сценарии из каталога **rc.d**.

12 ГЛАВА

Настройка командного интерпретатора

- Что такое командный интерпретатор?
- Работа с командными интерпретаторами
- Использование альтернативных командных интерпретаторов
- Файлы инициализации интерпретатора
- Настройка среды командного интерпретатора
- Переменные среды и переменные командного интерпретатора

Даже, если вы будете управлять системой FreeBSD при помощи графического интерфейса X-Window, вам, как администратору, все равно придется иметь дело с командным интерпретатором. Он предоставляет гибкость, необходимую для решения самых сложных задач, но в то же время требует определенного опыта. В этой главе мы ознакомимся с командным интерпретатором и овладеем основными принципами работы с ним.

Что такое командный интерпретатор?

Об основах работы с командным интерпретатором уже говорилось в главе 8. В ней был представлен интерфейс командной строки, а также обсуждались различные варианты интерпретаторов, которыми можно пользоваться во FreeBSD. В этой главе командный интерпретатор рассматривается более подробно: что можно делать с его помощью и как его можно настроить для своих нужд.

Как известно, интерпретатор (shell) представляет собой интерфейс командной строки, существенно отличающийся от графических сред Windows или Mac OS. По своей функциональности он похож на интерпретатор COMMAND.COM системы MS-DOS, предшествовавшей Windows, однако гораздо сложнее. Он играет значительно большую роль в управлении системой, чем графическая среда.

Иногда интерпретатор называют *оболочкой* (*shell*). Этот термин применяется в системах UNIX для описания различных уровней работы операционной системы, в архитектуре которой различают внутренние, автоматизированные функции, и внешние, вызываемые пользователем. Внутренняя часть системы называется *ядром* (мы подробно рассмотрим его в главе 17). С внешней стороны находится пользователь. А что окружает ядро? Естественно, оболочка! С ее помощью пользователь взаимодействует с ядром и программами. Взаимосвязи проиллюстрированы рисунком 12.1.

Одной из задач интерпретатора является обеспечение безопасного и структурированного доступа к ядру. Он воспринимает пользовательские команды и с помощью функций ядра системы "превращает" их в процессы, работающие с файлами и устройствами. Он предотвращает выполнение ненадежного кода, который может привести к краху ядра, а также облегчает поиск и запуск программ в системе.

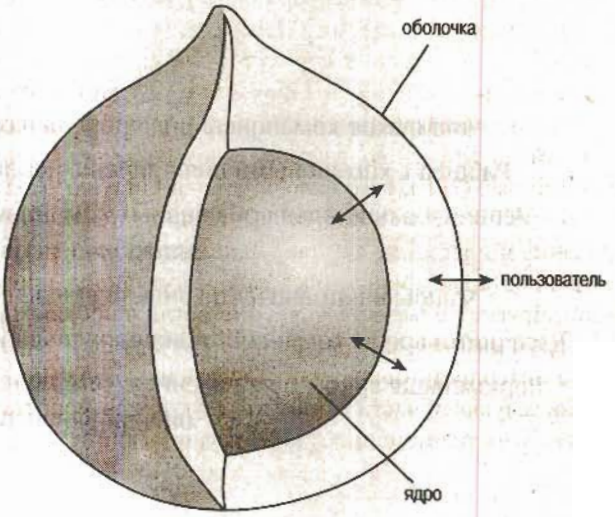


Рисунок 12.1
Ядро, окруженное оболочкой

В практическом смысле интерпретатор — это программа, способная порождать новые процессы. Она запускается системой при каждой регистрации в ней, выводит приглашение командной строки, выполняет команды пользователя и прекращает свою работу после выхода из системы. Фактически, это все, что должен выполнять командный интерпретатор, поэтому многие программы могут действовать в этом качестве.

Термин оболочка (shell) применяется и в несколько иных аспектах. Например, оболочкой является программа, запущенная поверх какой-либо другой и обеспечивающая интерфейс с последней. Windows 95 в техническом смысле является оболочкой, хотя и менее явной, чем ее предшественники (Windows 3.1, DESQview или интерпретатор MS-DOS). Многие программы поддерживают возможность "выхода" в DOS (т.е. к интерфейсу командной строки). Такой подход присущ программам, которые предшествовали многозадачным операционным системам, когда система позволяла работать только с одним терминальным окном. В данном случае "выход в оболочку" (shelling out) означает просто запуск из программы интерпретатора (т.е. командной строки DOS или UNIX).

В главе 8 мы рассмотрели различные программы FreeBSD, специально предназначенные для запуска сценариев и, таким образом, являющиеся командными интерпретаторами. Они включают **sh**, **csh**, **bash**, **ksh** и **tcsh**. Поведение этих командных интерпретаторов, а также используемые в них языки программирования слегка различаются. В главе 13 будет рассказано о создании сценариев в среде командного интерпретатор Bourne (**sh**), наиболее популярного среди пользователей. Здесь же мы разберем конфигурирование избранного интерпретатора для своих нужд.

Работа с командными интерпретаторами

Потребностям большинства пользователей удовлетворит командный интерпретатор FreeBSD по умолчанию, **/bin/tcsh**. Тот факт, что во FreeBSD по умолчанию используется **tcsh**, а не **bash** (повсеместно используемый в Linux), отражает небольшие различия философского характера между традициями BSD и структурой System V (на которой основана большая часть архитектуры Linux), эти аргументы уходят корнями в юность системы UNIX. Сейчас эти философские различия несколько утратили актуальность. Коммерческие варианты UNIX обычно следуют одному из путей, который не изменяется годами. Однако между Linux и FreeBSD существуют различия, связанные с разницей в структуре лицензий GNU и BSD. Именно здесь спрятан ответ на вопрос, почему в Linux предпочитают **bash**, а в FreeBSD — **tcsh**.

Поскольку **bash** является GNU-программой, а в Linux предпочтение отдается программному обеспечению, разработанному и распространяемому по лицензии GNU (GNU Public License, GPL), **bash** используется в Linux по умолчанию. **bash** и **tcsh** очень похожи, однако иногда доступ к ним осуществляется по-разному. Кроме того, процесс конфигурирования и поведение на этапе исполнения тоже несколько различаются. Тем не менее читатели, которые привыкли к Linux, могут инсталлировать **bash** и пользоваться знакомой средой. Другие пользователи вполне могли в прошлом работать с системами, где интерпретатором был **ksh** или **zsh**. И здесь нет проблем: мы рассмотрим, как установить различные интерпретаторы и использовать их совместно.

В последующей части главы исследуются интерпретаторы **tcsh** и **bash**. В ней показаны различия в их применении и конфигурировании.

Установка командных интерпретаторов

Устанавливать программное обеспечение в FreeBSD предпочтительнее из портов или пакетов (подробнее об этом см. главу 15). В этой главе частично изложены инструкции по инсталляции программного обеспечения из пакетов или набора портов. Далее представлены основные команды и процедуры, предназначенные для установки новых интерпретаторов.

Инсталляция из пакетов

Наиболее простой способ установки нового командного интерпретатора заключается в использовании программы **sysinstall** и ее *менеджера пакетов* (*package manager*). Следует запустить **/stand/sysinstall**, выбрать **Configure**, а затем **Packages**. В качестве устройства инсталляции нужно выбрать CD-ROM (если у вас есть компакт-диск с дистрибутивом FreeBSD) или FTP.

После того, как на экране появится список пакетов, необходимо перейти к разделу **Shells** и просмотреть список доступных интерпретаторов. На каждом устанавливаемом интерпретаторе нужно нажать пробел или поставить крестик (**X**). После этого следует нажать клавишу Enter и вернуться к главному экрану программы. С помощью клавиши курсора "стрелка вправо" выберите пункт **Install**.

В результате будет загружено и установлено заданное программное обеспечение. После этого нужно выйти из программы **sysinstall**, выбрав опцию **Exit** в верхней части меню **Configuration**, а затем **Exit Install** — в нижней. Процесс установки интепретатора(ов) завершен. Проверить правильность инсталляции можно, просмотрев содержимое каталога **/usr/local/bin**, а также страницы соответствующего справочного руководства, например **man bash**.

Установка из портов

Альтернативный способ установки программного обеспечения (к нему мы еще вернемся в главе 15) заключается в использовании коллекции портов. Это строго структурированный метод компиляции программного обеспечения с его последующей установкой, гарантирующий, что все файлы будут размещены там, где требуется (в частности, в каталоге **/usr/local**, в полном соответствии с правилами иерархии во FreeBSD). Если набор портов установлен, он находится в каталоге **/usr/ports** и содержит ту же структуру, которой подчинены пакеты, управляемые программой **sysinstall**.

Вначале следует перейти в каталог **/usr/ports/shells** и просмотреть его содержимое. Это относительно небольшой раздел коллекции портов. В других категориях (подкаталогах внутри **/usr/ports**) находится большее число программ, доступных для установки. Каждый из каталогов, показанных на листинге ниже, содержит **Makefile**, различные контрольные суммы и файлы исправлений, а также список файлов пакета.

```
44bsd-csh/          es/osh/rc/vshnu/
Makefileesh         /pash/ruby-shell/wapsh/
README.html         flash/pdksh/sash/zsh/
bash1/              ksh93/perlsh/scsh/sh-devel/
bash2/              mudsh/pkg/tcsh/
```

Остается лишь перейти в нужный каталог и запустить утилиту **make**: система распакует необходимый дистрибутивный файл, внесет исправления, сконфигурирует и скомпилирует его. Для инсталляции пакета следует ввести команду **make install**. За-

вершает процесс команда **make clean**, предназначенная для удаления временных файлов, созданных в процессе сборки:

```
# cd /usr/ports/shells/bash2
# make
  ...
# make install
  ...
# make clean
  ...
```

По завершении процесса новый интерпретатор будет установлен и готов к использованию.

ПРИМЕЧАНИЕ

Если используется **tcsh**, может потребоваться выполнить команду **rehash**, которая заставляет командный интерпретатор заново прочесть файлы конфигурации, чтобы обновить пути, к которым могли добавиться новые программы. В ином случае доступ к ним может оказаться невозможным без выхода из системы и повторного входа в нее (фактически, без перезапуска командного интерпретатора).

Файл /etc/shells

Очень важным системным файлом, о котором непременно следует знать, является **/etc/shells**. В нем содержится список всех командных интерпретаторов, установленных в системе, с полными путями к ним. Вот пример файла **/etc/shells** работающей системы с тремя дополнительными интерпретаторами:

```
# cat /etc/shells
# $FreeBSD: src/etc/shells,v 1.3 1999/08/27 23:23:45 peter Exp $
#
# Список доступных интерпретаторов для chpass(1).
# Ftpd не позволяет подключаться к системе тем пользователям,
# которые не используют один из следующих интерпретаторов.
/bin/sh
/bin/csh
/bin/tcsh
/usr/local/bin/bash
/usr/local/bin/zsh
/usr/local/bin/ksh
```

Задача файла **/etc/shells** — указать системе список командных интерпретаторов, которые вы, как системный администратор, считаете допустимыми. Таким образом, система не разрешает пользователю заменить свой интерпретатор, заданный по умолчанию, на программу, которая не предназначена для таких целей. Наибольшую проблему представляют программы с установленным битом **setuid** или те, что запускаются с эффективным UID пользователя, владеющего ими (об этом речь шла в главе 10). Если программа с битом **setuid** принадлежит **root**, все ее действия выполняются с правами **root**. Естественно, нежелательно, чтобы обычный пользователь мог получить такие полномочия, просто зарегистрировавшись в системе. Программа **chsh** не позволит рядовому пользователю заменить свой командный интерпретатор по умолчанию на программу, не входящую в список **/etc/shells**.

Командный интерпретатор пользователя указан в базе данных **/etc/master.passwd** в десятом поле каждой записи (или в седьмом поле записи в файле **/etc/passwd**). Об этом было рассказано в главе 8:

```
/etc/master.passwd:
foo:*:$1$LXZkCuzD$7Oa8LyRgbjYOb.XrXiBad.:1001:1001::999066364:0:Foo
   Bar:/home/foo:/usr/local/bin/ksh
/etc/passwd:
foo:*:1001:1001::Foo Bar:/home/foo:/usr/local/bin/ksh
```

Если в этом поле указан интерпретатор (в данном примере **/usr/local/bin/ksh**), содержащийся в **/etc/shells**, пользователь сможет зарегистрироваться в системе и использовать такие службы, как FTP. Если же нет, то в доступе к службам типа FTP будет отказано. Файл **/etc/shells**, кроме того, используется для создания списка доступных интерпретаторов во время процесса добавления новой учетной записи посредством команды **adduser** (см. главу 10).

При каждой инсталляции нового интерпретатора из портов или пакетов, запись о нем автоматически добавляется в файл **/etc/shells**. Обратите внимание на следующий момент: хотя только **root** может заменить командный интерпретатор пользователя программой, не включенной в **/etc/shells**, пользователь может зарегистрироваться в системе (прямо с консоли или по telnet/ssh) независимо от того, включен его интерпретатор в этот файл или нет. Список интерпретаторов управляет также доступом к системе по FTP, и в этом кроется самая распространенная ловушка для администраторов. Предположим, что новый интерпретатор инсталлируется не из портов или пакетов (и запись о нем не добавляется в файл **/etc/shells** автоматически), после чего устанавливается как командный интерпретатор некоторого пользователя (проводить такие настройки имеет только **root**). В результате, пользователь может без проблем зарегистрироваться в системе с помощью **telnet** или **ssh**, однако при попытке использовать протокол FTP система отказывает ему в доступе. Решение состоит в том, чтобы добавить командный интерпретатор пользователя в файл **/etc/shells**. Рекомендуется всегда использовать порты или пакеты для инсталляций подобного рода.

Использование альтернативных командных интерпретаторов

Каждый пользователь может изменить командный интерпретатор, используемый им по умолчанию. Для этого может быть масса причин: кто-то привык к Linux и **bash**, кому-то необходима специально настроенная среда, мигрировавшая из другой системы и требующая специального формата конфигурационных файлов (этот вопрос мы обсудим позже). В любом случае, изменение командного интерпретатора — дело несложное.

Изменение интерпретатора в процессе работы

Простейший способ перехода к другому интерпретатору состоит в его запуске. Если по умолчанию запущен **tcsh**, а требуется **bash** (предполагается, что он установлен в системе), достаточно просто ввести команду **bash** во время сеанса работы **tcsh**.

```
# bash
bash-2.04#
```

Помните: теперь выходить из системы нужно будет дважды: один раз из процесса **bash**, а второй — из исходного процесса **tcsh**. Только при выходе из *начального командного интерпретатора* (*login shell*) происходит отключение от системы.

Изменение командного интерпретатора по умолчанию

Конечно же, запускать новый интерпретатор после каждой регистрации в системе неэффективно, даже если это делается и не вручную, а путем добавления команды к файлу **.login** (сценарий инициализации, выполняемый после запуска интерпретатора), поскольку это приводит к выполнению программы внутри программы. К счастью, каждый пользователь имеет возможность выбрать в качестве интерпретатора любую программу, включенную в файл **/etc/shells**. Для выполнения этой задачи предназначена программа **chsh** (*change shell* — изменить командный интерпретатор). Ее поведение различно на разных платформах. На одних, скажем, в Linux, **chsh** — это интерактивное средство командной строки, запрашивающее новые значения по одному. Во FreeBSD программа **chsh** аналогична программам **chpass**, **chfn** и другим средствам управления учетными записями пользователей: все они являются жесткими ссылками друг на друга, поскольку выполняют схожие функции.

Поведение **chsh** сводится к открытию файла с информацией о пользователе в текстовом редакторе, позволяющем редактировать значения доступных полей. По умолчанию используется редактор **vi**. Его можно заменить другим редактором, например, **ee** или **pico**, установив соответствующим образом значение переменной среды **EDITOR**. (К переменным среды мы еще вернемся чуть ниже.)

Для изменения своего интерпретатора достаточно ввести **chsh**. Пользователь **root** может изменить командный интерпретатор любого пользователя, указав в командной строке имя последнего в качестве аргумента.

```
# chsh frank
#Changing user database information for frank.
Shell: /bin/tcsh
Full Name: Frank Allen
Office Location:
Office Phone:
Home Phone:
Other information:
~
~
```

Начинающим пользователям работать с **vi** непросто. Об этом редакторе написаны целые книги. Однако для профессионалов эта программа предлагает один из самых удобных видов интерфейса. Сейчас мы рассмотрим только команды, необходимые для изменения интерпретатора пользователя. Внутренние команды **vi** необходимо вводить точно, иначе информацию можно случайно удалить или исказить без возможности ее восстановления. Если вы допустили ошибку, введите **:q!**, а затем нажмите Enter. Это позволит вам выйти из программы без сохранения изменений.

```
:q!
```

Предположим, что требуется изменить командный интерпретатор пользователя frank с **/bin/tcsh** на **/usr/local/bin/bash**. Эта задача выполняется в несколько этапов. Находясь в редакторе **vi** (после запуска команды **chsh frank**), нужно поместить курсор на слово **tcsh**. Затем следует нажать **c** (change — изменить) и **w** (word — слово). В конце изменяемого слова появится знак **$**, как показано ниже:

```
Shell: /bin/tcs$
```

Вместо **tcsh** нужно ввести **bash** и нажать Escape для выхода из этого режима. Теперь командным интерпретатором будет **/bin/bash**. Затем нужно поместить курсор на первом символе косой черты (перед именем каталога **bin**) и нажать **i** (insert — вставить) — редактор перейдет в режим добавления текста. Ввести **/usr/local** и вновь нажать Escape. Полная строка должна выглядеть как **/usr/local/bin/bash**.

Чтобы перейти к командной строке редактора нужно ввести **:w** (write — записать), нажать Enter и сохранить файл. Для выхода нужно ввести **:q** и нажать Enter.

Сохраненный файл на самом деле, является временным файлом в каталоге **/etc**. Если внесенные изменения корректны, они будут автоматически прочитаны и внесены в файлы **/etc/master.passwd** и **/etc/passwd**. С этого момента при каждой регистрации пользователя в системе его интерпретатором будет **/usr/local/bin/bash** вместо **/bin/tcsh**. Если изменения некорректны (например, заданная программа не включена в список **/etc/shells**), программа предоставляет возможность повторно отредактировать настройки пользователя или отменить операцию.

Программы, используемые в качестве командных интерпретаторов

Командным интерпретатором пользователя может быть и программа, специально для этого не предназначенная. Существуют программы (например, **/sbin/nologin**), которые только выдают текстовое сообщение и завершают работу. Некоторые программы предназначены для взаимодействия с другими службами, а не просто с файловой системой UNIX (например, TinyFugue). Их применение может оказаться более предпочтительным для пользователя. Если в качестве командного интерпретатора пользователя указать несуществующую программу, это полностью предотвратит возможность его регистрации в системе. В качестве интерпретатора можно использовать даже **/usr/bin/mail** (программа для чтения почтовых сообщений). В этом случае единственным доступным действием будет чтение почты. Подобные методы часто применяются в правилах использования учетных записей на сервере.

Если указанного интерпретатора не существует, программа **chsh** выдает предупреждение, но все-таки вносит его в базу данных пользователей. В этом нет ошибки — это просто один способов предотвращения регистрации пользователей в системе. Помните, что работа начального командного интерпретатора предполагает такую последовательность действий: программа корректно запускается и подключается к **pty** (псевдотерминал), регистрация в системе проходит успешно и не прекращается до тех пор, пока не завершается работа интерпретатора и **pty** не освобождается. Если интерпретатор не запускается, к терминалу ничего не подключается и пользователь просто не получает приглашения командной строки. То же самое происходит при ошибке (в зависимости от терминальной программы), которая может в случае проблем с авторизацией, сбоя протокола или (при соединении по **telnet**) явной ошибки запуска командного интерпретатора после вывода текстовых сообщений, например:

```
FreeBSD/i386 (stripes.somewhere.com) (ttyp2)

login: frank
Password:
Last login: Sat May  5 18:35:52 from w044.z064002043.
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
          The Regents of the University of California.  All rights
                reserved.
```

```
FreeBSD 4.3-RELEASE (STRIPES) #0: Wed Jan 31 18:45:43 PST 2001
----------------------------------------------------------------------
Welcome to the system! Today's news: nothing.

You have mail.
login: /usr/local/bin/foosh: No such file or directory
Connection closed by foreign host.
```

Поведение системы можно варьировать, изменяя, в частности, выводимые перед запуском интерпретатора сообщения. Например, если пользователь или группа нарушили правила работы в системе (например, имела место попытка взлома), им можно отказать в регистрации, установив в качестве интерпретатора несуществующую программу и добавив объяснения правил пользования системой в файл **/etc/motd** (этот текстовый файл выводится после успешной проверки пароля). Достаточно отредактировать файл **/etc/motd** в текстовом редакторе и это сообщения будут выводиться всем пользователям, включая и тех, для кого не указан реальный командный интерпретатор.

Более правильное решение — установить в качестве начального интерпретатора пользователя программу, выводящую некоторый текст, предназначенный именно для него. Примером может служить **/sbin/nologin** — программа, запрещающая пользователю регистрацию в системе. Это всего лишь сценарий, который выводит строку текста и завершает работу. Сообщения могут быть, например, такими:

```
You have mail.
This account is currently not available.
Connection to stripes.somewhere.com closed.
```

Обратите внимание, что программа **/sbin/nologin** выполняет все необходимые процедуры: присоединяется к псевдотерминалу, завершает работу и освобождает его. Хотя она не предоставляет приглашения командной строки, тем не менее работает не менее корректно, чем любой интерпретатор. Такой способ запрещения регистрации в системе подходит для любой терминальной программы, включая **telnet** и **ssh**, причем ошибок или предупреждений не выводится. Он достаточно элегантен. Фактически, вместо интерпретатора можно указать любую программу — интерактивную или нет. Это может быть **/bin/ls** или **/usr/bin/finger**. Иногда полезнее создать собственный сценарий или программу, запускаемую в тех случаях, когда определенные пользователи регистрируются в системе. Таким путем система может поддерживать различные службы, обеспечивая, например, пользователям со статусом гость возможность регистрироваться в системе и запускать ограниченный набор команд, взаимодействуя с меню. Или же просто получить определенную информацию. Возможности безграничны!

ПРЕДУПРЕЖДЕНИЕ

Если программа-обработчик подобного типа написана на C или другом подобном языке, где требуется явно выделять блок памяти перед его использованием, следует внимательно следить за возможностью переполнения буфера! Это одна из самых распространенных дыр в защите сетевых служб. Найдя брешь такого типа, особенно, если программа выполняется с правами пользователя **root** (если ее бит **setuid** установлен и владельцем является **root**), атакующий может получить полный контроль над системой. В этой области необходимо быть предельно *осторожным!*

Файлы инициализации интерпретатора

Пришло время обсудить, как настраивается командный интерпретатор. Он позволяет создавать псевдонимы для упрощения команд, автоматически запускать опреде-

ленные программы при входе в систему, устанавливать переменные среды и т.д. За решение этих задач отвечают файлы инициализации и конфигурации, существующие на глобальном и пользовательском уровнях.

Поскольку **tcsh** и **bash** имеют разные наборы конфигурационных файлов, мы последовательно расскажем о них. FreeBSD включает в себя конфигурационные файлы системного и пользовательского уровня для обоих интерпретаторов, поэтому при желании от **tcsh** к **bash** можно перейти во всей системе. Естественно, что так делать не обязательно.

ПРИМЕЧАНИЕ

При создании новой учетной записи пользователя, конфигурационные файлы по умолчанию копируются из каталога **/usr/share/skel** (приставка **dot** перед именем каждого файла опускается) в начальный каталог пользователя. При необходимости можно скопировать все конфигурационные файлы в каталог **/usr/local/share/skel**, внести в них изменения, а затем указать в файле **/etc/adduser.conf** новое местонахождение файлов, применяемых по умолчанию. Таким образом, можно внести глобальные изменения в конфигурации командных интерпретаторов еще до создания учетных записей пользователей.

Файлы tcsh/csh: .cshrc, .login и .logout

Первый файл, к которому при запуске обращается **tcsh**, — это конфигурационный файл системного уровня **/etc/csh.cshrc**. Сразу же за ним следует обращение к **/etc/csh.login**. Обратите внимание, что во FreeBSD существуют оба файла, однако их содержимое закомментировано (см. листинг 12.1).

Листинг 12.1 Глобальные файлы /etc/csh.cshrc и /etc/csh.login

```
# cat /etc/csh.cshrc
# $FreeBSD: src/etc/csh.cshrc,v 1.3 1999/08/27 23:23:40 peter Exp $
#
# Файл системного уровня .cshrc для csh(1).

# cat /etc/csh.login
# $FreeBSD: src/etc/csh.login,v 1.19.2.1 2000/07/31 20:13:26 rwatson
➥Exp $
#
# Файл системного уровня .login для csh(1).
# Уберите комментарий, чтобы перейти к поведению системы по
умолчанию версии 4.2, где информация о диске показана в K-блоках
# setenv BLOCKSIZE K
#
# Для установки языков и кодовых страниц обратитесь к
# login.conf(5), а также к опциям charset и lang.
# Полный locale-лист содержится в /usr/share/locale/*
#
# Чтение системных сообщений
# msgs -f
# Включение терминальных сообщений
# mesg y
```

Листинг показывает, что эти файлы не выполняют никаких функций, однако ими можно воспользоваться. Всякое изменение, внесенное в эти файлы, например, включение терминальных сообщений, применяется глобально и отменяется лишь в случае, если в конфигурационных файлах пользователя соответствующая опция установлена по-другому. Конфигурационные файлы пользователя аналогичны системным, но обра-

щение к ним происходит чуть позже. Эти файлы находятся в начальном каталоге пользователя: **.cshrc** и **.login**.

Файл **.cshrc** (его первоисточник — **/usr/share/skel/dot.cshrc**) по умолчанию производит настройку среды командного интерпретатора. В частности, он создает несколько псевдонимов:

```
alias h          history 25
alias j          jobs -l
alias la         ls -a
alias lf         ls -FA
alias ll         ls -lA
```

Устанавливает пути поиска программ:

```
set path = (/sbin /bin /usr/sbin /usr/bin /usr/games /usr/local/
↪sbin /usr/local/bin /usr/X11R6/bin $HOME/bin)
```

А также задает переменные среды, управляющие поведением некоторых системных заданий:

```
setenv   EDITOR    vi
setenv   PAGER     more
setenv   BLOCKSIZE          K
```

Этот файл конфигурации выполняет и некоторые другие функции. Просмотрите файл **.cshrc** из вашего начального каталога.

В файле **.login**, который выполняется следующим, пользователь размещает программы, которые должны запускаться при его регистрации в системе. Например, по умолчанию в файл **.login** (из **/usr/share/skel/dot.login**) включена программа **fortune** и условный оператор, однако эта строка закомментирована:

```
#  Вывод  сообщения  при  входе  в  систему:
#  [  -x  /usr/games/fortune  ]  &&  /usr/games/fortune  -s
```

Администратор или сам пользователь могут указать процессы, которые должны запускаться при выходе из системы. К примеру, это может быть сценарий, удаляющий все временные файлы из каталога **/tmp**, владельцем которых является данный пользователь. Сценарий достаточно включить в файл **/etc/csh.logout**, не имеющий по умолчанию выполняемого содержимого:

```
# cat /etc/csh.logout
# $FreeBSD: src/etc/csh.logout,v 1.3 1999/08/27 23:23:41 peter Exp $
#
# Файл системного уровня .logout для csh(1).
```

Процесс выхода из системы обращается и к файлу **.logout** в начальном каталоге пользователя, если он там есть. По умолчанию этот файл не устанавливается.

ПРИМЕЧАНИЕ

Все вышеперечисленные файлы выполняются при запуске **tcsh** в качестве начального интерпретатора. Однако командный интерпретатор может быть запущен и при других обстоятельствах. Например, для выполнения сценария (когда требуемый интерпретатор указан непосредственно в строке сценария). В этом случае файлы **.login** и **.logout** (а также их эквиваленты системного уровня) игнорируются.

Файлы bash: .profile, .shrc и .bash_logout

bash работает аналогично **tcsh**: вначале исполняет файлы инициализации системного уровня, а затем переходит к пользовательским. Вначале происходит чтение файла

/etc/profile, который (как и **/etc/csh.cshrc**) не выполняет никаких действий, но содержит примеры того, что он может делать (как показано на листинге 12.2).

Листинг 12.2 Глобальный файл /etc/profile

```
# cat /etc/profile
#  $FreeBSD: src/etc/profile,v  1.12.2.1  2000/07/31 20:13:26 rwatson Exp $
#
# Файл системного уровня .profile для sh(1).
#
# Уберите комментарий, чтобы перейти к поведению системы по
# умолчанию версии 4.2, где информация о диске показана в K-блоках
# BLOCKSIZE=K; export BLOCKSIZE
#
# Для установки языков и кодовых страниц обратитесь к
# login.conf(5), а также к опциям charset и lang.
# Полный  locale-лист содержится в /usr/share/locale/*
# О более полном управлении локализацией рассказано в справочном
# руководстве setlocale(3).
#
# Чтение системных сообщений
# msgs -f
# Включение терминальных сообщений
# mesg y
```

Обратите внимание, что файл **/etc/profile** совмещает функциональность **/etc/csh.cshrc** и **/etc/csh.login**. Это единственный глобальный файл **bash** в системе, глобального сценария для выхода из системы не существует.

Далее обрабатывается пользовательский файл **.profile**, в котором переменные среды (включая **PATH**) устанавливаются и экспортируются интерпретатору в стиле Bourne:

```
# при желании удалите /usr/games и /usr/X11R6/bin
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/games:/usr/local/bin:/usr/
  X11R6/bin:$HOME/bin; export PATH

BLOCKSIZE=K;        export BLOCKSIZE
EDITOR=vi;          export EDITOR
PAGER=more;         export PAGER

# переменная ENV содержит имя файла, вызываемого при всяком
# интерактивном использовании sh.
ENV=$HOME/.shrc; export ENV
```

В последней строке переменной **ENV** присваивается значение **.shrc**. Соответствующий файл находится в начальном каталоге пользователя. Его чтение происходит после файла **.profile**. В нем устанавливаются псевдонимы подобно тому, как **tcsh** создает их в **.cshrc**, а также несколько других опций, например, изменение приглашения командной строки (по умолчанию оно отключено):

```
# Несколько полезных псевдонимов
alias h='fc -l'
alias j=jobs
alias m=$PAGER
alias ll='ls -laFo'
alias l='ls -l'
alias g='egrep -i'

# # set prompt: ``username@hostname$ ''
# PS1="`whoami`@`hostname | sed 's/\..*//'`"
# case `id -u` in
#           0) PS1="${PS1}# ";;
```

```
#          *) PS1="${PS1}$ ";;
# esac
```

Как и в случае с файлом **.logout** для **tcsh**, интерпретатор **bash** выполняет файл **.bash_logout**, если он присутствует в начальном каталоге. По умолчанию он не устанавливается.

ПРИМЕЧАНИЕ

Перечисленные выше файлы используются интерпретатор **bash** тогда, когда он выполняется как начальный командный интерпретатор. В ином случае он не обращается к **.profile**, а читает файл **.bashrc**. Обычно этот файл в FreeBSD отсутствует.

Настройка среды командного интерпретатора

Мы уже видели несколько примеров того, как настраивается среда командного интерпретатора (обоих типов) с помощью его конфигурационных файлов. Это является собой замечательный пример расширения функциональности командного интерпретатора. Следует отметить, что существуют дополнительные возможности, примеры которых не представлены в конфигурационных файлах по умолчанию. Мы рассмотрим, как они реализуются в **tcsh** и **bash**. В большинстве случаев это встроенные команды, которые можно запустить или напрямую из командной строки, или в одном из конфигурационных файлов интерпретатора.

ПРИМЕЧАНИЕ

Подробную информацию о большом числе встроенных команд, часть из которых представлена в **csh/tcsh**, часть в **bash**, а часть — в обоих, можно найти в справочном руководстве **man builtin**.

Настройка tcsh

Одной из самых распространенных настроек является создание *псевдонимов* (*alias*). Она заключается в замене команды любым выбранным именем. Псевдонимы позволяют существенно упростить работу, особенно, когда существует набор часто используемых сложных команд. Примеры простейших псевдонимов приведены в **.cshrc**:

```
alias ll              ls -lA
```

В результате интерпретатор заменяет команду **ll /usr/local** на **ls -lA /usr/local**. Однако как поступить в том случае, когда необходимо производить подстановку аргумента — изменять параметры командной строки в зависимости от введенных значений? Эту задачу позволяет решить чрезвычайно гибкое и сложное средство синтаксического анализа командной строки **tcsh**. Полностью о нем рассказано в справочном руководстве **man tcsh**. В данном примере отметим лишь, что к первому аргументу команды можно обратиться как **\!^** (или **\!:1**), а к последующим — **\!:2**, **\!:3** и т.д. Поэтому можно воспользоваться, например, такой командой:

```
alias lookup     grep \!^ /etc/passwd
```

Теперь для извлечения информации о пользователе достаточно применить команду **lookup frank**. Для удаления псевдонима используется команда **unalias**:

```
unalias lookup
```

Интерпретатор позволяет изменить внешний вид приглашения командной строки. Это достигается путем установки переменной **prompt**: вывод любой команды можно использовать, заключив его в обратные апострофы, как это делается, например, в сценариях на языке Perl. Кроме того, можно отобразить переменную, содержащую "номер" команды в истории. Для этого применяется символ !. Ниже приведен достаточно сложный пример, где применяются многие свойства, включая использование **sed** (stream **ed**itor — текстовый процессор, обрабатывающий блоки текста в неинтерактивном режиме) для сокращения имени хоста до его первого элемента:

```
# set prompt="{  `whoami`@`hostname | sed 's/\..*//'`:!}   "
{  root@www:23}
```

В приглашении командной строки бывает удобно отображать каталог, в котором в данный момент пребывает пользователь, используя результат команды **pwd** (present working directory — текущий рабочий каталог):

```
# set prompt="{  `pwd`:!}   "
{  /root:24}
```

Установка пути поиска команд в **tcsh** также предполагает присвоение значения переменной среды (о переменных среды будет рассказано чуть позже). К массиву значений, заключенному в скобки, нельзя добавить значение "на ходу". Нужно добавить новый элемент к строке, а затем перезапустить команду **set**, именно поэтому настраивать пути лучше всего в одном из конфигурационных сценариев, а не прямо из командной строки. Синтаксис команды представляет собой заключенный в скобки список путей, разделенных пробелами:

```
set path = (/sbin /usr/sbin /bin /usr/bin /usr/local/bin /usr/
contrib/bin /usr/X11R6/bin /usr/local/sbin /usr/games . /usr/local/
mystuff)
```

Еще одним полезным встроенным средством является **stty**. Оно позволяет переопределить кодовую таблицу символов. Это требуется в ситуациях, когда терминальная программа пересылает другим программам неожиданные символы, например, возникает путаница между delete и backspace или символами новой строки. Вначале необходимо просмотреть текущие настройки:

```
# stty -a
speed 38400 baud; 60 rows; 80 columns;
lflags: icanon isig iexten echo echoe -echok echoke -echonl echoctl
        -echoprt -altwerase -noflsh -tostop -flusho pendin
              -nokerninfo -extproc
iflags: -istrip icrnl -inlcr -igncr ixon -ixoff ixany imaxbel
              -ignbrk brkint -inpck -ignpar -parmrk
oflags: opost onlcr -oxtabs
cflags: cread cs8 -parenb -parodd hupcl -clocal -cstopb -crtscts
              -dsrflow -dtrflow -mdmbuf
cchars: discard = ^O; dsusp = ^Y; eof = ^D; eol = <undef>;
        eol2 = <undef>; erase = ^?; intr = ^C; kill = ^U;
        lnext = ^V; min = 1; quit = ^\; reprint = ^R;
        start = ^Q; status = ^T; stop = ^S; susp = ^Z;
        time = 0; werase = ^W;
```

Теперь, если требуется установить символ удаления (erase) как **^H**, следует воспользоваться командой:

```
# stty erase ^H
```

Кроме того, можно включить режим просмотра (watch), когда система сообщает о пользователях, зарегистрированных в ней, и используемых псевдотерминалах. Это достигается последовательностью команд:

```
set watch=(1 any any)
set who="%n has %a %l from %M."
```

Свойство **tcsh**, которое наверняка имеет смысл отключить, — это переменная среды **autologout**. По умолчанию она установлена значением 60 минут. Отключение терминала через час простоя раздражает пользователей. Отключение этой переменной достигается одной командой в конфигурационном сценарии:

```
unset autologout;
```

И в заключение, если настройка переменных выполнялась путем редактирования конфигурационных сценариев, нет нужды выходить из системы и заново регистрироваться в ней, чтобы настройки вступили в силу, — достаточно перечитать конфигурацию встроенной командой **rehash**.

Настройка bash

Псевдонимы в **bash** настраиваются несколько иначе, чем в **tcsh**. Псевдоним и заменяющий его текст разделяются знаком =, а не табуляцией или пробелом. Кроме того, в этом командном интерпретаторе нет механизма подстановки аргументов, подобного тому, что применяется в **tcsh**, поэтому, например, создание псевдонима **lookup** в **bash** требует более сложных команд.

```
alias ll='ls -laFo'
```

Интересно выглядит настройка приглашения командной строки в **bash**. Поскольку **bash** реализует настройку переменных среды по-другому, необходимо настроить первичную (primary) и вторичную (secondary) переменные (**PS1** и **PS2**), которые интерпретатор отображает пользователю. Чтобы получить приглашение, похожее на адрес электронной почты, применявшееся ранее в **tcsh**, используется следующая команда (обратите внимание на экранирующий символ обратной косой черты перед символом **!**, хотя последний не имеет специального значения, присущего ему в **tcsh**):

```
# PS1="{ `whoami`@`hostname | sed 's/\ ..*//'`:\!} "
{ root@www:17}
```

Чтобы получить приглашение, содержащее текущий рабочий каталог, применяется команда:

```
# PS1="{ `pwd`\ :!} "
{ /root:18}
```

Установка путей поиска команд в **path** производится также с помощью переменной среды **PATH**. В данном случае это список каталогов, разделенных двоеточием. Кроме того, для передачи этой переменной интерпретатору сразу же за списком следует команда **export**.

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/games:/usr/local/bin:/usr/
➥X11R6/bin:$HOME/bin; export PATH
```

О необходимости экспорта переменных (т.е. пересылки их командному интерпретатору) рассказано в следующем разделе.

Переменные среды и переменные командного интерпретатора

Одним из наибольших различий между **tcsh** и **bash** является их методология работы с переменными. Как будет ясно из главы 13, существует два типа переменных, используемых для работы. А сейчас рассмотрим, чем они отличаются и как их следует применять.

Переменные среды

Независимо от того, какой интерпретатор используется, он имеет переменные среды. Они содержат значения, доступные запускаемым программам во время всего сеанса работы. Более того, эти переменные управляют поведением некоторых программ. Например, утилита **chfn** запускает текстовый редактор, заданный в переменной среды **EDITOR**, а переменная **TERM** сообщает интерпретатору, как следует форматировать текст, чтобы он корректного отобразился на экране. Переменная **BLOCKSIZE** управляет форматом вывода чисел утилитами **du** и **df**: килобайты, полукилобайты и пр. Программе не требуется быть начальным командным интерпретатором, чтобы обратиться к переменным среды, — любая программа имеет доступ к ним точно так же, как и та, что их установила.

В любом из вышеуказанных командных интерпретаторов можно просмотреть список переменных среды, воспользовавшись командой **printenv**, вывод которой приведен на листинге 12.3.

Листинг 12.3 Пример вывода команды printenv

```
# printenv
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/local/bin:/usr/contrib/bin:/usr/
➥X11R6/bin:.
MAIL=/var/mail/frank
BLOCKSIZE=1k
FTP_PASSIVE_MODE=YES
USER=frank
LOGNAME=frank
HOME=/home/frank
SHELL=/bin/tcsh
SSH_CLIENT=192.168.173.230  50095  22
SSH_TTY=/dev/ttyp0
TERM=vt100
HOSTTYPE=FreeBSD
VENDOR=intel
OSTYPE=FreeBSD
MACHTYPE=i386
SHLVL=1
PWD=/home/frank
GROUP=users
HOST=stripes.somewhere.com
REMOTEHOST=192.168.173.230
PATHSET=true
EDITOR=vi
VISUAL=pico
```

Для установки переменной среды в командном интерпретаторе используется один из двух возможных вариантов синтаксиса, зависящий от интерпретатора. Чтобы ус-

тановить переменную **COLOR** значением **gold** в **tcsh** применяется следующая команда:

```
# setenv COLOR gold
```

В **bash** переменную следует вначале установить в рамках локального сеанса, а затем экспортировать среде:

```
# COLOR=gold
# export COLOR
```

Обратите внимание, что переменная среды должна иметь значение. Она не может быть равна null или вовсе не определена.

Переменные командного интерпретатора

Существует и другой набор переменных, доступных во время сеанса работы. Эти *переменные командного интерпретатора* применяются только во время текущего сеанса и не связаны с другими процессами. Они доступны только процессу интерпретатора. В терминах программирования, переменные интерпретатора являются локальными, а переменные среды — глобальными, которые наследуются всеми программами.

Переменные интерпретатора должны принадлежать к нижнему регистру в **tcsh** и к верхнему — в **bash**. Их можно просмотреть посредством команды **set**. Установка переменных интерпретатора в **tcsh** производится также с помощью команды **set**. В **tcsh** можно определить переменную интерпретатора, не присваивая ей значения:

```
# set history 100
# set noclobber
```

В **bash** достаточно присвоить значение переменной, не экспортируя ее:

```
# VISUAL=pico
```

С помощью команды **unset** переменную можно удалить:

```
# unset autologout
```

Подробно переменные среды и интерпретатора описаны в главе 13, мы также разберем, как их свойства наследования применяются при создании сценариев.

13 ГЛАВА

Программирование на языке командного интерпретатора

- Пример простой программы для интерпретатора
- Переменные
- Взаимодействие с пользователем
- Арифметические операции в сценариях
- Циклы
- Условные операторы
- Код завершения
- Функции
- *Файловые дескрипторы*
- Отладка сценариев командного интерпретатора
- Расширенные возможности интерпретатора Korn

В главе 8 было рассказано о том, как работать с командной строкой интерпретатора в интерактивном режиме. Следует отметить, что его возможности этим не ограничиваются — в интерпретатор встроен мощный язык программирования. Этот язык используется для самых разных задач: от автоматизации повторяющихся команд до написания сложных интерактивных программ обработки данных, получения информации из небольших баз данных или ее пополнения.

Благодаря модульной структуре операционной системы FreeBSD любую ее команду можно вызвать из программного интерпретатора. Если одна команда FreeBSD не может выполнить нужную задачу, можно использовать "конвейеры", состоящие из цепочек команд. *Конвейер* позволяет направить вывод одной команду на ввод другой. Об этом подробнее рассказано далее в этой главе. FreeBSD поддерживает сотни модульных команд, каждая из которых выполняет небольшую специализированную задачу. Существуют команды для поиска в текстовом файле, объединения двух файлов, форматирования текста в столбцы, извлечения определенных полей из текстового файла, подсчета количества символов, слов и строк, выполнения математических операций, резервного копирования, архивации и разархивации файлов и т.д. Если нет команды или их набора, которые способны выполнить нужную операцию, вполне возможно, что в Internet можно найти свободно распространяемую программу для решения этой задачи.

Программирования на языке командного интерпретатора зачастую избегают из-за наличия Perl и других языков, считая его просто "вариантом прославленного языка пакетного программирования DOS", что совершенно неверно. Конечно, в своей простейшей форме сценарии командного интерпретатора похожи на bat-файлы DOS, используемые лишь для запуска определенных команд, содержащихся в этом файле. Однако, потратив время на изучение работы интерпретатора, осознав, как он обрабатывает команды, и каким образом вывод одной команды с помощью конвейера можно отправить на ввод другой, вы сможете писать очень серьезные и сложные программы. Особенно хорошо это будет получаться после изучения сотни специализированных команд FreeBSD, предназначенных для решения узких задач.

Вот шесть основных причин, по которым стоит изучать программирование на языке командного интерпретатора в FreeBSD:

- Его легко выучить. Если вы часто работаете с командной строкой FreeBSD, вы, наверняка, уже знакомы со многими командами, используемыми в сценариях.
- Он экономит время. Для решения некоторых задач, требующих часы или даже дни программирования на C, достаточно 5–10 минут программирования на языке интерпретатора.
- Этот язык позволяет автоматизировать утомительную работу. Скажем, если требуется внести одно и то же изменение в 100 файлов, достаточно воспользоваться циклом "for" для автоматизации этого процесса.
- Он позволяет изучить новые полезные методы работы с системой. При программировании вы столкнетесь с командами, о которых никогда раньше не слышали, и узнаете о новых способах решения известных задач. Вы лучше изучите командную строку FreeBSD и поймете, какой мощью она обладает.
- Он развивает креативный потенциал. Вы всегда сможете создать нужную вам программу на языке командного интерпретатора. Одним из главных преимуществ FreeBSD является философия "Делай по-своему". Существует множество

способов достижения цели, и не обязательно общепринятый прием является наилучшим.

- Он обучает думать. Возможно, это идеальный язык для того, чтобы научиться программировать, поскольку он, работая со знакомыми командами, позволяет сосредоточиться на логике программы.

Итак, перейдем к изучению одного из самых мощных средств FreeBSD.

ПРИМЕЧАНИЕ

Сначала мы расскажем о командах, присущих всем командным интерпретаторам Bourne. Вторая часть главы посвящена расширенным возможностям Korn и других командных интерпретаторов, поддерживающих стандарт POSIX. Если нужна программа, способная работать в разных системах, и заранее неизвестно, какие интерпретаторы будут применяться, имеет смысл ограничиться набором общих команд.

ПРИМЕЧАНИЕ

О программировании в интерпретаторах семейства **csh** в этой главе не рассказывается. Следует отметить, что подобные интерпретаторы не слишком удобны для программирования, поскольку им недостает многих полезных свойств (включая функции и обмен потоков **STDOUT** и **STDERR**). Рекомендуется использовать командные интерпретаторы семейства Bourne.

Пример простой программы для интерпретатора

По традиции обучение начинается с написания программы "Hello, World!". Вот один из возможных способов ее написания на языке интерпретатора Bourne:

```
1.   #!/bin/sh
2.
3.   # Легендарная программа "Hello, World!",
4.    # реализованная для интерпретатора Bourne.
5.
6.   echo
7.   echo "Hello World!"
8.   echo
9.   exit 0
```

Внимание! Номера строк приведены только для удобства чтения кода. При его вводе их следует опустить. Введите текст и сохраните файл. Затем сделайте его выполняемым. Для этого используется следующая команда:

```
chmod u+x hello
```

Эта команда устанавливает права владельца на запуск файл (предполагается, что он назван **hello**).

Теперь файл можно запустить. Пример вывода программы показан ниже:

```
bash$ ./hello
Hello World!
bash$
```

Рассмотрим каждую строку кода подробно.

- **Строка 1:** В этой строке содержится "магическая последовательность" символов **#!**, сообщающая FreeBSD, что за ней следует имя командного интерпретатора, в котором данный сценарий должен выполняться. В данном случае это

интерпретатор Bourne, **/bin/sh**. Для сценария на языке Perl первая строка выглядела бы так: **/usr/bin/perl**, на языке Python — **/usr/local/bin/python** и т.д.

- **Строки 2 и 5:** Это просто пустые строки. Командный интерпретатор игнорирует пустые символы, если они не взяты в кавычки (подробнее об этом далее). Такие строки используются лишь для улучшения читаемости программы.
- **Строки 3 и 4:** Это комментарии. Они начинаются с символа # и простираются до конца строки. Интерпретатор игнорирует их. Включать в код комментарии следует обязательно: чтобы по прошествии некоторого времени иметь возможность разобраться в том, что делает программа.
- **Строка 6:** Команда **echo** выводит свои аргументы в стандартный вывод, **STDOUT** (обычно, это экран). Его можно также перенаправить в файл или на какое-либо устройство. В данном случае команда **echo** выводит в **STDOUT** пустую строку.
- **Строка 7:** Команда **echo** выводит строку "Hello, World!" в **STDOUT**. Кавычки сообщают интерпретатору, что все находящееся между ними следует обрабатывать как один аргумент (т.е. строка воспринимается как один элемент, а не как их набор). Дело в том, что без кавычек интерпретатор воспринимает пустой символ как разделитель аргументов. Далее в этой главе вы увидите, зачем их применяют. Поэтому стоит приобрести привычку всегда использовать кавычки вокруг строковых выражений. В данном случае кавычки не являются обязательными.
- **Строка 8:** Команда **echo** выводит пустую строку.
- **Строка 9:** Команда **exit** завершает работу программы и возвращает код выхода родительской программе. (Обычно это командный интерпретатор, но может быть и какая-либо другая программа.) Код завершения, равный 0, указывает, что программа нормально завершила работу, код, отличный от 0, сообщает об ошибке. Обычно, код завершения читает вызывающая программа и на его основе принимает решение: какое действие предпринять.

В такой короткой программе устанавливать статус завершения необязательно. Если он не указан явно, программа возвращает статус завершения последней выполненной команды. Тем не менее лучше сразу привыкать устанавливать код завершения, тогда при разработке более объемных и сложных сценариев будет меньше ошибок. Далее в этой главе мы увидим, как этим приемом можно воспользоваться для автоматического принятия решения о том, какое действие следует выполнять дальше.

printf

В предыдущей программе вместо команды **echo** можно было применить команду **printf**. Она выполняет такое же действие, но позволяет управлять форматированием вывода. В следующем примере показано, как выглядит программа, использующая **printf** вместо **echo**:

```
1.      #!/bin/sh
2.
3.   # Легендарная программа "Hello, World!",
4.    # реализованная для интерпретатора Bourne.
5.
6.    printf  "\nHello  World!\n\n!"
7.    exit  0
```

Вывод этой программы полностью совпадает с выводом предыдущей.

Если вам доводилось программировать на C, синтаксис команды **printf** вам хорошо знаком. Символ обратной косой черты представляет собой управляющий символ, указывающий, что следующий за ним символ имеет специальное значение (или, наоборот, отменяющий специальное значение символа, если последний им обладает). В данном случае "n" — символ новой строки. Он позволяет отказаться от двух команд **echo**, выводящих пустые строки, включив их в аргумент команды **printf**. Таким образом, символы новой строки можно включить в печатаемую строку. В конце строки содержится два таких символа, поскольку **printf** автоматически не добавляет его к концу строки. Чтобы вставить одну пустую строку, необходимо указать два символа. Вот пример, показывающий, что команда **printf** не переходит на новую строку автоматически:

```
printf "Hello, "
printf  "World!\n\n"
```

Здесь две строки кода печатают одну фразу "Hello, World!", хотя отдельные слова выводятся разными операторами. Это связано с тем, что **printf** не добавляет символ новой строки к печатаемой строке. Если бы мы в данном случае воспользовались командой **echo** вместо **printf**, "Hello" и "World!" оказались бы в разных строках.

Команда **printf** поддерживает следующие символы форматирования вывода:

Таблица 13.1 Управляющие символы форматирования вывода команды printf

Символ форматирования	*Описание*
\a	Звуковой сигнал. Во времена телетайпов на машине звенел настоящий звонок. В наши дни этот символ подает сигнал на динамик компьютера.
\b	Печатает символ "backspace".
\f	Печатает символ "переход к новой странице".
\n	Символ перехода к новой строке.
\r	Выводит символ возврата каретки. (возвращается к первому символу строки)
\t	Печатает символ табуляции.
\v	Печатает символ вертикальной табуляции.
\'	Символ одинарной кавычки.
\"	Символ двойной кавычки.
\\	Символ обратной косой черты.
\num	Печатает символ, заданный восьмеричным ASCII-кодом *num*. Вероятно, вам никогда не придется пользоваться им в сценариях.

Конечно же, простого вывода сообщений на экран недостаточно для решения задач. Следующим свойством языка сценариев являются переменные.

Переменные

Из алгебры известно, что для обозначения неизвестных величин в уравнениях используются буквы, называемые *переменными*. В программировании они подчиня-

ются подобной концепции, хотя в общем случае переменные могут содержать не только числа, а и строки (т.е. последовательности символов).

При программировании на языке командного интерпретатора используются два типа переменных: переменные интерпретатора и переменные среды. Основное различие состоит в том, что переменные среды доступны другим сценариям и программам, вызываемым из вашего сценария, а переменные сценария — только ему самому.

В сценариях, не в пример программам, с переменными можно обращаться довольно свободно — нет, например, необходимости декларировать их до использования. Все переменные в программах командного интерпретатора хранятся как строки.

Присвоение значения переменной

В простейшем случае значение переменной присваивается следующим образом:

```
myvar=5
```

Теперь в переменной **myvar** хранится значение 5. Для доступа к информации в переменной перед ее именем следует указывать символ **$**. Например, команда **echo ${myvar}** выводит 5 в поток **STDOUT**, т.е. обычно на экран.

ПРИМЕЧАНИЕ

В присвоении переменных важную роль играют пустые символы. Выражение **myvar=5** присваивает 5 переменной **myvar**. А выражение **myvar = 5** приводит к ошибке, поскольку интерпретатор трактует **myvar** как имя команды, которую он пытается запустить, а не как имя переменной.

Фигурные скобки не являются обязательными, однако они улучшают читаемость кода, выделяя имена переменных. Использовать их или нет — решайте сами.

Значение одной переменной можно присвоить другой, например:

```
myvarB=$myvar
```

Это выражение присваивает содержимое переменной **myvar** переменной **myvarB**. Если **myvarB** уже содержит какое-либо значение, оно будет заменено новым.

Кроме того, переменной можно присвоить вывод команды или ввод из потока **STDIN** (обычно это клавиатура). Мы увидим пример такого присваивания далее в этой главе.

Создание переменной среды похоже на создание переменной интерпретатора. Единственное различие заключается в том, что ее необходимо экспортировать. Для этого используется оператор **export**. Например, выражение:

```
MYVAR=5
export  MYVAR
```

создает переменную интерпретатора **MYVAR**, а затем экспортирует ее, делая переменной среды, которая будет доступна другим программам, запущенным из данного командного интерпретатора.

Имена переменных

В *именах переменных учитывается регистр. Они* могут содержать буквы, цифры и символ подчеркивания. Имя переменной **не может** начинаться с цифры. Кроме того, не следует вначале имени переменной использовать подчерк. Желательно применять описательные имена, чтобы код программы было легко читать. Например, предположить, что содержит переменная **avg_rainfall** (средн_осадки), гораздо проще, чем гадать о назначении переменной с именем **xyz123**.

СОВЕТ

По соглашению в именах локальных переменных используются символы нижнего регистра, а в именах переменных среды — верхнего. Некоторые программисты предпочитают использовать смешанные имена для локальных переменных, что позволяет отличать их от других команд интерпретатора, которые, как правило, состоят из символов только нижнего регистра.

Взаимодействие с пользователем

Кроме присвоения значений переменным и их использования в сценарии, командный интерпретатор предоставляет возможность обрабатывать ввод из потока **STDIN**. Обычно этот ввод набирается пользователем (или читается из файла, если применяется перенаправление потока **STDIN**). Для чтения пользовательского ввода существует команда **read**. Ниже приведена несколько модифицированная версия программы "Hello, World", которая читает ввод из потока **STDIN**, которым, как правило, является клавиатура:

```
1.  #!/bin/sh
2.
3.  # Модифицированная программа Hello, World, воспринимающая ввод
    # с клавиатуры
4.
5.  echo
6.  echo -n "Please enter your name: "
7.  read name
8.  echo
9.  echo "Hello, ${name}!"
10. echo
11. exit 0
```

Программа выводит следующее:

```
Please enter your name: Mike
Hello, Mike!
```

В этой программе присутствуют три новых аспекта, подлежащих обсуждению:

- **Строка 6:** Использована новая опция команды **echo**. Опция **-n** отменяет печать символа новой строки, который обычно сопровождает строку, выводимую оператором **echo**. В результате курсор остается в той же строке, что и фраза **Please enter your name:**.
- **Строка 7:** Команда **read** читает ввод из потока **STDIN**, в данном случае — с клавиатуры. Пользователь вводит строку текста. После нажатия клавиши Enter, **read** сохраняет ее в переменной **name**. Команда **read** предполагает использование кавычек, поэтому строка сохраняется точно в том виде, в каком ее ввел пользователь, со всеми пустыми символами.
- **Строка 9:** Команда **echo** используется для вывода строки **Hello**, за которой следует содержимое переменной **name**. Фигурные скобки не являются обязательными.

Команда **read** в качестве аргументов может иметь несколько переменных (см. Листинг 13.1). В этом случае всякий пустой символ трактуется как символ-разделитель между значениями, присваиваемыми разным переменным. Пример показан в листинге 13.1.

Листинг 13.1 Использование нескольких переменных в аргументе команды read

```
#!/bin/sh

echo
echo -n "Введите три числа, разделенные пробелами или символом
➥табуляции: "
read var1 var2 var3
echo
echo "The value of var1 is: ${var1}"
echo "The value of var2 is: ${var2}"
echo "The value of var3 is: ${var3}"
echo
exit 0
```

Вот пример запуска программы:

Введите три числа, разделенные пробелами или символом табуляции: 557 2024 57240

```
The value of var1 is: 557
The value of var2 is: 2024
The value of var3 is: 57240
```

Не имеет значения, сколько пустых символов присутствует во вводе. При присвоении значений нескольким переменным **read** интерпретирует любое число смежных пустых символов как один разделитель аргументов.

Если число аргументов, прочитанных из потока ввода, меньше, чем число переменных в списке, оставшимся переменным не присваивается никаких значений. С другой стороны, если аргументов больше, чем переменных, все оставшиеся значения присваиваются последней переменной. Например, при другом запуске программы ей указывается более трех чисел:

Enter three numbers separated by spaces or tabs: 1 2 3 4 5 6 7 8 9 0

```
The value of var1 is: 1
The value of var2 is: 2
The value of var3 is: 3 4 5 6 7 8 9 0
```

Обработка аргументов командной строки

Получить информацию от пользователя можно также при чтении аргументов командной строки. Командный интерпретатор автоматически сохраняет значения этих аргументов в специальных переменных. Фактически, для их чтения не требуется специального программного кода. В командную строку можно включить до девяти аргументов. Они хранятся в переменных **$1-$9**. Командный интерпретатор, поддерживающий стандарт POSIX, позволяет обращаться к дополнительным аргументам как **${10}**, **${11}** и т.д. Однако этого следует избегать, особенно если программы будут использоваться и в других системах, поскольку обычный интерпретатор Bourne обрабатывает только переменные **$1-$9**. Переменная **$0** содержит имя самой программы, переменная **$@** — все аргументы, а **$#** — количество аргументов. Например:

```
#!/bin/sh
echo
echo "The name of the program is: $0"
echo "The total number of arguments received is: $#"
echo "The complete argument string is: #@"
echo "Your first name is: $1"
echo "Your last name is: $2"
echo
exit 0
```

Вот пример запуска программы:

```
bash$ ./yourname Michael Urban

The name of the program is: ./yourname
The total number of arguments received is: 2"
The complete argument string is: Michael Urban"
Your first name is: Michael
Your last name is: Urban

bash$
```

ПРИМЕЧАНИЕ

Командный интерпретатор FreeBSD, поддерживающий стандарт POSIX, имеет команду **getopts** — более гибкий способ обработки аргументов командной строки в сценарии. Мы изучим эту команду при обсуждении расширенных возможностей программирования в интерпретаторе Korn. Если вы создаете Bourne-сценарии (т.е. они используют **sh**), предназначенные для других UNIX-систем, не следует применять **getopts**, поскольку эта команда является нестандартной для Bourne. В результате программа может оказаться неработоспособной в других системах.

Подстановка команд

Подстановка команд позволяет запустить команду и присвоить ее вывод переменной. Выполняемую команду следует заключить в символы `. Их не следует путать с одинарными кавычками. ` представляет собой символ обратной кавычки (на клавиатуре он, как правило, совмещен с символом тильды ~). Например, оператор:

```
TodayDate=`date`
```

запускает команду **date** и присваивает ее вывод переменной **TodayDate**. После этого к значению переменной можно обращаться обычным образом.

Арифметические операции в сценариях

Хотя исходный командный интерпретатор Bourne не имеет встроенной арифметики, подобные действия все-таки можно производить, воспользовавшись подстановкой команд и командой **expr**. Например, оператор:

```
var3=`expr var1 + var2`
```

складывает значения переменных **var1** и **var2** и сохраняет результат в **var3**. Заметьте, что аргументы и оператор необходимо разделять символом пробела. Команда **expr var1+var2** желаемого результата не даст.

Команда **expr** позволяет выполнять лишь простейшие арифметические операции с целыми числами. Попытка использования чисел с плавающей точкой приведет к ошибке. Кроме того, изменение порядка операций с помощью скобок не поддерживается. При делении дробная часть отбрасывается. Например, результатом 5 / 2 команды **expr** будет 2. Чтобы найти остаток от деления используется оператор деления с остатком (%). Например, **expr 5 % 2** возвращает 1.

Символы, имеющие специальное значение для интерпретатора, следует экранировать. Например, команда **expr 2 * 2** не дает результата, так как интерпретатор рассматривает символ * как символ-заместитель. Для выполнения операции умножения оператор (*) следует экранировать: **expr 2 * 2**. Символ обратной косой черты предотвращает использование символа в его специальном значении.

Команда **expr** позволяет работать и с логическими выражениями (значением которых является истина или ложь). Если выражение истинно, **expr** возвращает 1. В ином случае **expr** возвращает 0. Например, оператор:

```
expr 2 + 2 = 4 + 1
```

возвращает 0 (поскольку выражение не является истинным), а

```
expr 2 + 2 = 3 + 1
```

возвращает 1 (выражение истинно).

Операция **!=** обращает значение логического выражения. Название этой операции — "не равно". Поэтому, например, выражение **expr 5 != 3** возвращает 1 (истина).

Команда **expr** позволяет также обрабатывать неравенства. Помните: поскольку символы < и > имеют для интерпретатора специальное значение, при использовании с **expr** их необходимо экранировать. Для неравенств команда также возвращает значение истина/ложь. Например, выражение **expr 5 \> 4** возвращает 1 (неравенство истинно). Кроме того, поддерживаются операторы больше или равно (\>=) и меньше или равно (\<=).

expr не ограничивается сравнением лишь чисел. Эта команда позволяет сравнивать также и строки. Например, выражение **expr "The quick brown fox jumped over the lazy dog" = "The quick brown fox jumped over the lazy dg"** возвращает 0 (в данном случае равенство ложно).

И хотя **expr** подходит для простых операций в сценариях, возможности этой команды весьма ограничены. Для операций с числами с плавающей запятой (с точки зрения программиста — с плавающей точкой), а также обработки сложных выражений, где порядок операций изменен, применяется команда **bc**. Она представляет собой специальный язык программирования, предназначенный для математических действий. **bc** можно использовать для обработки выражений, включенных в сценарии. В листинге 13.2 показан пример программы, использующей **bc** для вычисления длины окружности и площади круга.

Листинг 13.2 Использование bc для вычисления длины окружности и площади круга

```
1.  #!/bin/sh
2.
3.   # Данная программа вычисляет длину окружности
4.   # и площадь круга.
5.
6.   pi="3.14159265"              # Переменной присваивается число p.
7.
8.   # Пользователю выводится сообщение и запрашивается радиус
9.   # окружности.
10.
11.  echo
12.  echo "This program computes both the circumference and the area of"
13.  echo "a circle."
14.  echo
15.  echo -n "Please enter the radius of the circle: "
16.  read radius
17.
18.  # Вычисляемые значения сохраняются в переменных.
19.
20.  circumference=`echo "$radius*$pi" | bc -lr`
21.  area=`echo "$radius^2*$pi" | bc -lr`
22.
```

```
23. # Программа выводит результаты и завершает работу.
24.
25. printf "\n\nThe circumference is:\t$circumference\n"
26.  printf  "The  area  is:\t\t$area\n\n"
27. exit 0
```

Большинство концепций, используемых в данной программе, уже были объяснены, поэтому мы не будем рассматривать каждую строку кода, а остановимся лишь на новых и важных операторах.

- **Строка 6:** В этой строке переменной **pi** присваивается значение числа p. Мы специально упоминаем об этом, поскольку этот пример иллюстрирует правильный подход к программированию. С той же легкостью вместо переменной **pi** можно было использовать в программе число 3.14159265. Но присвоение числа переменной служит двум основным целям. Во-первых, это делает код программы более удобным для чтения. Во-вторых, такой сценарий гораздо легче поддерживать. Если позднее окажется, что требуется более точное значение числа p, достаточно будет изменить лишь один оператор. Если бы мы использовали не переменную, а явное значение, его пришлось бы заменять другим во всем сценарии.
- **Строка 16:** Как показано ранее, команда **read** читает данные из потока **STDIN**. В данном случае введенное пользователем значение сохраняется в переменной **radius**.
- **Строка 20:** В строке 20 для вычисления длины окружности и сохранения результата в переменной **circumference** используется подстановка команд и конвейер. Поскольку **bc** не воспринимает аргументы командной строки, здесь используется команда **echo**, вывод которой с помощью конвейера перенаправляется **bc**.
- **Строка 21:** Эта строка подобна строке 20. Символ ^ в выражениях **bc** представляет собой оператор возведения в степень. В этом случае он возводит значение переменной **$radius** в степень 2 (как известно из геометрии, площадь круга равна радиусу в квадрате, умноженному на p). Скобки здесь не требуются, поскольку операция возведения в степень предшествует операции умножения.
- **Строки 25 и 26:** В этих строках результат выводится пользователю. Здесь применяется команда **printf**, поскольку она позволяет форматировать вывод лучше, чем **echo**. Применение символа табуляции **\t** форматирует числа в аккуратный столбец.

Данный пример демонстрирует простейший способ использования команды **bc**. Она является мощным средством, возможности которого гораздо шире, чем показывает эта программа. **bc** представляет собой язык программирования, о свойствах которого можно узнать на странице справочного руководства. Там же рассказано о применении этой команды в сценариях командного интерпретатора и отдельных программах.

Циклы

В некоторых случаях требуется повторять действие до тех пор, пока определенное выражение не станет истинным (или, наоборот, перестанет быть истинным). Здесь в игру вступают операторы циклов. Командный интерпретатор Bourne поддерживает три вида циклических конструкций: **while**, **until** и **for**. Рассмотрим каждую из них.

Цикл while

Цикл **while** выполняет операторы, заключенные в нем, до тех пор, пока условие цикла является истинным. Можно рассуждать и по-другому: цикл **while** повторяет последовательность операторов до тех пор, пока условие цикла не станет ложным. Если уже при первом проходе цикла условие ложно, операторы внутри цикла выполняться не будут. Например, в следующей программе цикл **while** используется для вывода на экран целых чисел от 1 до 20.

```
1.   #!/bin/sh
2.    #  Count  from  1  to  20
3.    i=1
4.    while  [  $i  -le  20  ]
5.    do
6.              echo  $i
7.              i=`expr  $i  +  1`
8.    done
9.exit  0
```

В этой программе представлены несколько новых концепций:

- **Строка 3:** Переменной **i** присваивается начальное значение 1. **i** часто применяется как счетчик цикла (это давно сложившаяся традиция), поэтому здесь нет необходимости использовать описательное имя.
- **Строка 4:** Команда **while** содержит условие, заключенное в квадратные скобки. На самом деле, они представляют собой сокращенную запись команды под названием **test**. Последняя часто используется в сценариях командного интерпретатора.

Команда **test** использует достаточно прозрачный синтаксис, близкий к Фортрану: **-le** в данном примере обозначает "меньше или равно". Таким образом, цикл выполняется до тех пор, пока значение переменной **i** меньше или равно 20. Операторы сравнения, поддерживаемые данной командой, приведены в таблице 13.2.

Таблица 13.2 Операции сравнения команды test

Опция	*Значение*
-eq	Истина, если операнды равны
-ne	Истина, если операнды не равны
-gt	Истина, если первый операнд больше второго
-ge	Истина, если первый операнд больше или равен второму
-lt	Истина, если первый операнд меньше второго
-le	Истина, если первый операнд меньше или равен второму

- **Строка 5:** Оператор **do** указывает, что все операторы, следующие за ним, должны выполняться при каждой итерации цикла. Все выражения между **do** и **done** являются телом цикла.
- **Строка 6:** Здесь выводится значение переменной **i**.
- **Строка 7:** Здесь используется подстановка команд, значение переменной **i** увеличивается на единицу, а затем новое значение вновь присваивается переменной **i**.

- **Строка 8:** Завершающий оператор цикла. В этой точке программа вновь возвращается к оператору **while** и вновь проверяет условие цикла. Если значение переменной **i** все еще меньше или равно 20, операторы в теле цикла выполняются снова. Если **i** больше 20, цикл завершается и управление передается оператору, следующему за оператором **done** (в данном случае, это просто оператор **exit** в строке 9, который завершает программу с кодом успешного выполнения, 0).

Обратите внимание, что операторы внутри цикла выровнены с отступом по левому краю. Это позволяет легко выделить цикл при чтении исходного кода. Интерпретатор игнорирует отступ, выполняя команды обычным образом. Я рекомендую всегда выделять циклы, облегчая их поиск и проверку в программе.

СОВЕТ

Пробел между **[** и тестируемым условием является обязательным. Его отсутствие приведет к ошибке. Например, **[$VarA -gt 5]** выполняется, а **[$VarA -gt 5]** возвращает ошибку.

Цикл until

Цикл **until** по смыслу противоположен циклу **while**. Он выполняет последовательность операций до тех пор, пока условие не станет истинным. В этом случае цикл завершается. Если условие истинно уже при первом запуске, операторы в теле цикла не запускаются.

Циклы **while** и **until** очень похожи. Как правило, любой из них можно использовать в программе и добиваться одного и того же результата, изменяя условие. Например, предыдущую программу можно переписать, заменив цикл **while** на **until**. При этом в строке 4 потребуется лишь два изменения:

```
4.  until  [$i  -gt  20  ]
```

Программа выполняет ту же функцию. Единственное различие заключается в том, что теперь цикл выполняется до тех пор, пока значение переменной **i** не станет большим 20, тогда как цикл **while** выполнялся до тех пор, пока **i** было меньше или равно 20. В обоих случаях программа дает один и тот же результат.

Логические операторы AND/OR в циклах while и until

Циклы **while** и **until** позволяют работать с логическими операторами AND/OR. Логическое выражение **AND** возвращает значение истина лишь в том случае, когда оба операнда истинны, а выражение **OR** — когда лишь один из операндов имеет значение истина. Ниже приведен пример кода с логическим оператором **AND**:

```
#!/bin/sh
# Пример логического оператора AND
VarA=1
VarB=5
while  [  $VarA  -eq  1  ]  &&  [  $VarB  -gt  7  ]
do
          echo  "VarA  is  equal  to  1  and  VarB  is  greater  than  7"
done
exit  0
```

В предыдущем примере оператор **echo** не запускается, ведь хотя **$VarA** равно 1, **$VarB** не больше 7. Поскольку цикл **while** в данном случае требует, чтобы оба условия были истинными, тест не проходит и возвращает значение 0 (ложь).

Ниже приведен блок кода, аналогичный предыдущему, за исключением того, что в цикле **while** используется оператор **OR**. Теперь достаточно, чтобы лишь одно выражение было истинным:

```
#!/bin/sh
# Пример логического оператора OR
VarA=1
VarB=5
while [ $VarA -eq 1 ] || [ $VarB -gt 7 ]
do
        echo "VarA is equal to 1 or VarB is greater than 7"
done
exit 0
```

В данном случае достаточно, чтобы только одно из условий давало значение истина. А поскольку **$VarA** равно 1, программа будет выполнять бесконечный цикл с оператором **echo**. (Чтобы прервать цикл, необходимо нажать комбинацию клавиш **Ctrl-C**.)

Цикл for

Цикл **for** отличается от **while** и **until**. Вместо проверки истинности условия цикл **for** выполняет операторы внутри тела цикла в зависимости от количества аргументов в списке. Цикл **for** содержит переменную, которая при каждой итерации получает следующий аргумент из списка. Цикл **for** продолжается до тех пор, пока список не будет исчерпан. После этого управление передается первому оператору, следующему за телом цикла. Вот пример программы, где цикл **for** и команда **bc** используются для вывода квадратных корней чисел от 10 до 20.

```
1.  #!/bin/sh
2.  # Вывод квадратных корней чисел 10 - 20
3.  for num in `jot 10 10 20`
4.  do
5.      square_root=`echo "scale=5; sqrt($num)" | bc -lr`
6.      echo $square_root
7.  done
8.  exit 0
```

Программа выводит следующее:

```
3.16227
3.31662
3.46410
3.60555
3.74165
4.00000
4.12310
4.24264
4.35889
4.47213
```

- **Строка 3:** Оператор цикла **for**. В этой строке представлена новая команда **jot**. Она производит набор полезных операций с числами, включая печать строки чисел и генерирование случайных значений. В данном случае **jot** печатает строку из 10 чисел, начиная с 10 и заканчивая 20 (10 10 20). Оператор цикла **for** пос-

ледовательно присваивает каждое из значений переменной **num** и выполняет операторы в теле цикла.

- **Строка 5:** Команда **bc** используется для вычисления квадратного корня из текущего значения переменной **$num**. Выражение **scale=5** указывает **bc**, что в выводе числа после десятичной точки следует сохранить пять значащих цифр. Обратите внимание на точку с запятой после выражения **scale**. Этот знак препинания используется для разделения операторов, содержащихся в одной строке. Функция **sqrt** команды **bc** возвращает квадратный корень из заданного числа. Поскольку интерпретатор заменяет переменную ее значением, **bc** получает число, а не имя переменной. В заключение, вывод команды перенаправляется команде **bc**. Опция **-l** указывает на необходимость предварительной загрузки математической библиотеки, где содержится функция **sqrt**.

shift

Команда **shift** похожа на цикл **for**. Чтобы выполнить цикл один раз для каждого аргумента командной строки, переданного сценарию, можно воспользоваться оператором **while** и командой **shift**.

Как отмечалось ранее, аргументы командной строки хранятся в переменных от **$1** до **$9**. Каждый запуск команды **shift** сдвигает переменные на одну позицию влево. Это значит, например, что информация, сохраненная в **$1** отбрасывается, а значение переменной **$2** присваивается **$1**. Например:

```
1.    #!/bin/sh
2.    # Программа иллюстрирует применение команды shift.
3.    while [ $# -ne 0 ]
4.    do
5.          echo "The value of \$1 is now $1."
6.          shift
7.    done
8.    echo
9.    exit 0
```

Вот вывод этой программы:

```
bash$ ./shift1 a b c d e
The value of $1 is now a.
The value of $1 is now b.
The value of $1 is now c.
The value of $1 is now d.
The value of $1 is now e.

bash$
```

- **Строка 3:** Здесь начинается цикл **while**. Переменная **$#** содержит общее число аргументов командной строки. Цикл **while** выполняется до тех пор, пока значение **$#** не становится равным нулю. Если **$#** равно нулю, следовательно, все аргументы использованы, после чего цикл завершается.
- **Строка 5:** В этой строке выводится текущее значение переменной **$1**. Обратите внимание, что для печати строкового значения **$1** на экране символ **$** необходимо экранировать символом обратной косой черты, поскольку сам по себе он имеет специальное значение.

- **Строка 6:** После выполнения команды **shift** переменные сдвигаются на одну позицию влево. Значение **$1** отбрасывается (оно больше недоступно), **$2** смещается в **$1**, **$3** — в **$2** и т.д.

Одним из распространенных вариантов применения команды **shift** (и циклов **for**) является обработка имен файлов, заданных как аргументы командной строки, и выполнение операций над каждым из них.

Операторы true и false

В программировании сценариев используются операторы **true** и **false**. Их единственным назначением является возвращение значения истина (0) или ложь (1), соответственно. Этими операторами можно воспользоваться для создания бесконечных циклов. В следующем разделе будет рассказано, как прервать бесконечный цикл. Далее в этой главе вы встретитесь с ситуациями, где такие циклы оказываются полезными. Ниже приведен пример программы с бесконечным циклом.

```
1.  #!/bin/sh
2.   # Пример бесконечного цикла.
3.   while true
4.   do
5.          echo "This line will print forever."
6.   done
7.   echo "This line will never print since the program will"
8.   echo "never get past the loop."
9.   exit 0 # Программа не сможет завершить работу корректно,
10.         # поскольку никогда не дойдет до этой точки.
```

Очевидно, здесь невозможно привести вывод этой программы, поскольку она не прекращает свою работу. В строке 3 условие проверяется на истинность. Так как аргумент (**true**) никогда не возвратит значения ложь, цикл будет повторяться бесконечно. Строки 7–10 никогда не будут выполнены, поскольку программа не сможет выйти из цикла. (Для завершения работы программы воспользуйтесь комбинацией клавиш **Ctrl-C**.)

Выход из цикла

Иногда возникает необходимость программно прервать бесконечный цикл. Для этого используется один из двух следующих операторов: **break** или **continue**.

Оператор break

Оператор **break** прерывает цикл немедленно, независимо от того, выполнено ли условие окончания цикла. В листинге 13.3 приведена небольшая модификация предыдущего примера с бесконечным циклом.

Листинг 13.3 Выход из цикла

```
1.  #!/bin/sh
2.  # Бесконечный цикл.
3.  while true
4.   do
5.       echo "This line will print forever... But..."
6.       break
7.   done
8.   echo
9.   echo "The break statement in the loop causes the loop"
10.  echo "to terminate immediately and go the first statement"
```

```
11. echo "after the loop."
12. exit 0
```

Вот вывод программы:

```
This line will print forever... But...

The break statement in the loop causes the loop
to terminate immediately and go to the first statement
after the loop.
bash$
```

(Оператор **break** — причина того, что бесконечный цикл был прерван, и программа перешла к первому оператору после цикла.)

Оператор continue

Оператор **continue** заставляет цикл перейти к началу и проверить условие. Операторы цикла, следующие за **continue**, не выполняются. Пример еще одной модификации программы с бесконечным циклом приведен на листинге 13.4:

Листинг 13.4 Перезапуск цикла до его окончания

```
1.  #!/bin/sh
2.  #  Бесконечный  цикл.
3.  while  true
4.  do
5.      echo  "This  line  will  print  forever."
6.      continue
7.      echo  "But  this  line  will  never  print even though it is inside"
8.      echo  "the  loop  because  the  preceding  continue  statement"
9.      echo  "causes  the loop to jump back  to  the  top  and  re-evaluate"
10.     echo  "the  test."
11. done
12. echo  "This  line  will  never  print  since  the  program  will"
13. echo  "never  get  past  the  loop."
14. exit  0 #  Программа  не  сможет  завершить  работу  корректно,
15.         #  поскольку  она  никогда  не  дойдет  до  этой  точки.
```

Эта программа будет безостановочно выводить строку "This line will print forever". Остальные операторы в теле цикла выполняться не будут, так как оператор **continue** заставит цикл переходить в начало и проверять условие цикла.

Условные операторы

Условные операторы выполняются в том случае, когда определенное условие (или несколько условий) истинно. Они могут принимать одну из трех общих форм: **if**, **case** и **AND/OR**.

Операторы if

Операторы **if** проверяют числовые выражения. Если условие истинно, выполняются операторы внутри блока **if**. Если оно ложно, есть две возможности:

- Ничего не происходит. Операторы внутри блока не запускаются, и программа продолжает выполняться дальше.
- Если в блок **if** включен оператор **else**, выражения из последнего выполняются, если условие ложно. Другими словами, поток управления программы следует правилу: "Сделать это, если условие истинно, или то, если оно ложно, но не оба действия одновременно".

Например, в следующей программе оператор **if** используется для проверки количества аргументов командной строки, переданных программе. Если их число больше или равно 1, программа выполняет операции внутри блока. Если они отсутствуют, программа завершает работу, не выполняя никаких действий.

```
1.   #!/bin/sh
2.   # ifprog: Примеры использования оператора if.
3.   if [ $# -ge 1 ]
4.   then
5.           echo "You supplied $# command line arguments."
6.   fi
7.   echo
8.   echo "Program exiting..."
9.   echo
10.  exit 0
```

Вот два примера работы программы:

Запуск 1:

```
bash$ ./ifprog file1 file2 file3
You supplied 3 command line arguments.
Program exiting...
bash$
```

Запуск 2:

```
bash$ ./ifprog
Program exiting...
bash$
```

Оператор **if** в строке 3 проверяет число аргументов командной строки, используя переменную **$#**, в которой оно содержится. Если оно больше или равно 1, выполняются операторы, заключенные между ключевыми словами **then** и **fi**. (**fi** — это **if** наоборот. Сей оператор отмечает конец блока.) Если нет, управление передается оператору, следующему за **fi**. В данном случае он информирует пользователя о выходе из программы.

Программу можно сделать более дружественной к пользователю, добавив оператор **else**, сообщающий пользователю, как следует запускать программу (вместо безмолвного выхода из нее). Пример приведен на листинге 13.5.

Листинг 13.5 Дружественная версия предыдущей программы

```
1.   #!/bin/sh
2.   # ifprog: Примеры использования оператора if.
3.   if [ $# -ge 1 ]
4.   then
5.           echo "You supplied $# command line arguments."
6.   else
7.           echo "Usage: $0 file1 file2..."
8            echo
9            exit 1
10.  fi
11.  echo
12.  echo "Program exiting..."
13.  echo
14.  exit 0
```

Если количество аргументов командной строки меньше 1, программа выполняет выражения в операторе **else** и выводит пользователю сообщение (помните, переменная **$0** содержит имя запущенной программы?). Обратите внимание на строку 9, где программа

прекращает работу с кодом завершения, равным 1 (выполнению программы помешали ошибки). Если программа исполняет оператор **else**, операторы в строках 10-14 не выполняются.

Часть **then** оператора **if** является обязательной, а часть **else** — нет. Как вы видели в примере, операторы в блоке **then** выполняются тогда, когда условие истинно. Но иногда возникает необходимость выполнить действия лишь тогда, когда выражение ложно. В этом случае нужно воспользоваться двоеточием. Например:

```
if [ $myvar -gt 5 ]
then
    : # Не предпринимать никаких действий, выйти из блока if
else
    # Операторы, которые выполняются, если условие ложно.
fi
```

elif

В некоторых случаях возникает необходимость проверить два или несколько разных условий и предпринять различные действия в зависимости от результатов каждого этапа. Для этих целей используется оператор **elif**.

elif является аббревиатурой от **else if**. Когда используется оператор **elif**, программа вначале выполняет оператор **if**. Если его условие истинно, выполняется его код, а затем управление передается следующему оператору(т.е. оператору, расположенному после **fi**). Если условие ложно, проверяется условие в первом операторе **elif**. Если оно истинно, выполняются операторы из его блока, и программа переходит к концу блока **if**. Если оно ложно, проверяется следующий оператор **elif** и т.д. Фактически, условия проверяются до тех пор, пока одно из них не даст значение истина. Если такого условия нет, ничего не происходит либо запускаются операторы, заключенные в блоке **else** (если он присутствует). В листинге 13.6 представлены многие из изученных концепций, включая **if**, **elif** и **else**. Программа реализует простой алгоритм угадывания числа.

Листинг 13.6 Простая игра по угадыванию числа

```
1.  #!/bin/sh
2.  # Игра по угадыванию числа
3.  clear
4.  guess_count=1    # Счетчик инициализируется значением 1
5.  echo
6.  echo "Number guessing game written in bourne shell script."
7.  echo
8.  echo -n "Enter upper limit for guess: "
9.  read up_limit
10. rnd_number=`jot -r 1 1 $up_limitr    # Получение случайного числа
11. echo
12. echo "Irve thought of a number between 1 and $up_limit."
13. echo
14. echo -n "Please guess a number between 1 and $up_limit: "
15. read guess
16. # Сравнение со случайным числом.
17. while true
18. do
19. if [ $guess -gt $rnd_number ]
20. then
21.         echo
22.         echo "Your guess was too high. Please try again."
23.         guess_count=`expr $guess_count + 1`
24.         echo -n "Please guess a number between 1 and $up_limit: "
```

```
25.         read guess
26.    elif [ $guess -lt $rnd_number ]
27.    then
28.         echo
29.         echo "Your guess was too low. Please try again."
30.         guess_count=`expr $guess_count + 1`
31.         echo -n "Please guess a number between 1 and $up_limit: "
32.         read guess
33.    else
34.         break
35.    fi
36.  done
37.  # Мы достигаем этой точки, если игрок отгадал число.
38.  echo
39.  echo "Correct!"
40.  echo
41.  echo "You guessed the number in $guess_count guesses."
42.  echo
43.  exit 0
```

Вот пример запуска программы:

```
Enter upper limit for guess: 10
Irve thought of a number between 1 and 10.

Please guess a number between 1 and 10: 5

Your guess was too low. Please try again.
Please guess a number between 1 and 10: 8

Your guess was too high. Please try again.
Please guess a number between 1 and 10: 7

Correct!

You guessed the number in 3 guesses.
```

Большинство концепций, примененных в этой программе, вам уже известны. Строка 3 просто очищает экран. Строка 4 инициализирует значением 1 переменную **guess_count**, в которой хранится число попыток, сделанных игроком. В строках 23 и 30 значение **guess_count** увеличивается на 1 при каждой неудачной попытке. В строке 10 используется подстановка команд: **jot** выполняется для присвоения переменной **rnd_number** случайного числа в диапазоне от 1 до верхнего предела, заданного пользователем. В строке 17 начинается бесконечный цикл. В строке 19 проверяется условие: число введенное игроком больше, чем случайное число, выбранное программой. Если да, игроку выводится сообщение, значение переменной **guess_count** увеличивается на 1, а затем запрашивается новое число. После того, как игрок вводит новое число, оно сохраняется в переменной **guess** и цикл **while** выполняется вновь. Если введенное число не больше случайного числа, оператор **elif** проверяет, меньше ли оно. Если да, игроку выводится сообщение, значение переменной **guess_count** увеличивается на 1, а затем запрашивается новое число. После ввода цикл **while** выполняется вновь. И наконец, если ни одно из условий не истинно, запускается блок **else** (здесь не нужны проверки, ведь если число ни больше и ни меньше, следовательно, оно равно загаданному). Оператор **else** прерывает бесконечный цикл, и программа передает управление первому оператору, следующему за оператором **done**. Последние выражения программы сообщают пользователю о том, что он догадался, и о количестве предпринятых попыток (для этого выводится значение переменной **guess_count**).

Операторы **if** поддерживают также логические операторы **AND (&&)** и **OR (||)**.

Оператор case

Если одну и ту же переменную необходимо проверить в различных условных операторах, существует более эффективный метод, чем применение нескольких операторов **if**. Оператор **case**, аргументом которого является переменная, содержит блоки, выполнение которых зависит от значения переменной. В листинге 13.7 оператор **case** используется для случайного выбора цитат.

Листинг 13.7 Использование case для случайного выбора цитат

```
1.  #!/bin/sh
2.  # Генератор случайных цитат.
3.  quote_num=`jot -r 1 1 5`
4.  case "$quote_num" in
5.      1) echo
6.         echo "\"Until he extends his circle of compassion to include"
7.         echo "all living things, man will not himself find peace.\""
8.         echo" -- Albert Schweitzer"
9.         echo ;;
10.     2) echo
11.        echo "\"With regard to excellence, it is not enough to
➥know, but"
12.        echo "we must try to have and use it.\""
13.        echo "-- Aristotle"
14.        echo ;;
15.     3) echo
16.        echo "\"Imagination is more important than knowledge.
➥Knowledge"
17.        echo "is limited. Imagination encircles the whole world.\""
18.        echo "-- Albert Einstein"
19.        echo ;;
20.     4) echo
21.        echo "\"It is not the strongest of the species that
➥survive, nor"
22.        echo "the most intelligent, but the one most responsive to
➥change.\""
23.        echo "-- Charles Darwin"
24.        echo ;;
25. esac
26. exit 0
```

В строке 3 команда **jot** используется для того, чтобы сгенерировать случайное число от 1 до 4. Блок **case** начинается в строке 4. Применяется следующий синтаксис: **case variable in**, где *variable* — имя переменной, значение которой проверяется. Первая проверка происходит в строке 5. Слева от скобки находится условие, обрабатываемое оператором **case**. Подобно оператору **if**, оператор **case** останавливается на первом же совпадении, запускает операторы из соответствующего блока, а затем переходит к оператору **esac** (это **case** наоборот). Конец каждого условного блока обозначается двойной точкой с запятой. Она может находиться в отдельной строке или же в строке с последним оператором блока, как показано в этом примере.

Оператор **case** поддерживает символы-заместители. Например:

```
case "$myvar" in
     a) #операторы для "a"
        ;;
     b) #операторы для "b"
        ;;
```

```
    *) #операторы для других значений
       ;;
esac
```

В этом фрагменте кода проверяется равенство переменной **$myvar** значению **a** или **b**. Если это так, то выполняются соответствующие операторы. Если нет, последним условием является сравнение с символом-заместителем, который совпадает с любым значением. Поэтому если переменная **$myvar** не равна ни **a**, ни **b**, исполняется последний фрагмент кода.

Кроме того, **case** поддерживает и другие символы-заместители, включая **?** (который работает точно так же, как и в командной строке интерпретатора), и символ конвейера, позволяющий передавать оператору **case** диапазон опций. Например, **Y | y)** совпадает с **Y** или **y** при проверке условия. Несколько следующих подряд символов следует заключать в квадратные скобки. Так, например, **[Yy][Yy][Ee][Ss])** дает совпадение со строкой "y" или "yes", содержащей любую комбинацию символов верхнего или нижнего регистра.

CGI-ПРОГРАММИРОВАНИЕ

Возможно, вы сталкивались с Web-сайтами, где представлены свойства, подобные случайному выбору цитат, или, к примеру, когда при каждой загрузке страницы на ней отображается другой рисунок. Один из методов реализации этого подхода заключается в использовании оператора **case**. Программы, аналогичные приведенной ранее, называются CGI-программами. CGI-интерфейс позволяет Web-серверу запускать внешние программы и пересылать их вывод браузеру. Подробнее о CGI-программировании рассказано в главе 26.

Логические операторы AND/OR

Логические условные операторы **AND/OR** в некоторых случаях заменяют операторы **if**. Код завершения первой команды используется как условие запуска второй. Например:

```
tar cvfz backup.tar.gz documents/2000/* && rm -r documents/2000
```

Эта команда означает следующее: "Если первая операция прошла успешно, выполнить вторую. Если нет — вторая команда не выполняется". Другими словами: "Необходимо выполнить команды A и B. Но если команда A невыполнима, то не следует исполнять и B". В данном случае первая команда архивирует все файлы из каталога **documents/2000** в файл **backup.tar.gz**. Если этот процесс завершается успешно (команда **tar** возвращает код 0), выполняется команда после оператора **&&**, которая удаляет каталог **documents/2000**. Если же процесс архивирования не завершен успешно (команда **tar** возвращает код, отличный от 0), команда после **&&** не исполняется (понятно, что нет смысла удалять каталог, если он не был корректно заархивирован).

Символом || обозначается оператор **OR**. Он означает следующее: "Если A невыполнимо, исполнить B. Но если A завершилось успешно, B не выполнять". Например:

```
tar cvfz backup.tar.gz documents/2000/* || echo "Archive operation
↪failed."
```

В этом случае, если операция архивирования завершается успешно (**tar** возвращает 0), команда после оператора || не запускается. Если же нет (**tar** возвращает значение, отличное от 0), выполняется команда, указанная после ||, — на экран выводится сообщение об ошибке.

Код завершения

Большинство программ в FreeBSD по окончании работы возвращают код завершения. При успешном завершении это 0. Число, отличное от 0, как правйло, указывает на ошибку. Некоторые программы возвращают разные коды — в зависимости от возникшей проблемы. Часто об этом рассказано на соответствующих страницах справочных руководств.

Код завершения последней команды хранится в переменной **"$?"**. Вот несколько примеров:

```
bash$ ls > /dev/null
bash$ echo $?
0
bash$

bash$ ls -2 /dev/null
ls: illegal option -- 2
usage: ls [-ABCFGHLPRTWabcdfgiklnoqrstu1] [file ...]
bash$ echo $?
1
bash$
```

В первом примере переменная **$?** установлена значением 0 (для краткости вывод команды **ls** перенаправлен на устройство **/dev/null**). Во втором примере указана неверная опция. Команда **ls** выдает сообщение об ошибке и устанавливает **$?** значением 1.

Код завершения используется в сценариях для принятия решений. Такой подход продемонстрирован в примере с логическими операторами **AND/OR**. В нем было показано, как успешное или неуспешное выполнение одной команды является условием запуска другой. Логические операторы **AND/OR**, используемые в предыдущем разделе, можно заменить одним оператором **if**:

```
if tar cvfz backup.tar.gz documents/2000/*
then
        rm -r documents/2000
else
        echo "Archive operation failed"
fi
```

В последнем случае скомбинированы два примера из предыдущего раздела. Оператор **if** читает код завершения команды **tar**. Если он равен 0, выполняются операторы из блока **then**, т.е. каталог удаляется. Если же он отличен от 0, исполняются выражения в блоке **else**.

Другое полезное применение этого свойства связано с командой **test**. Ранее мы уже рассматривали, как команда **test** используется с математическими выражениями. Кроме того, командой **test** можно пользоваться для проверки существования файла. Например:

```
if [-f program.conf]
then
      : # ничего не выполнять
else
        touch program.conf
done
```

Данный фрагмент кода проверяет существование файла **"program.conf"**. Если файл существует, возвращается 0 и запускается блок **then**. В данном примере этот блок не выполняет никаких действий. (Об этом говорит символ двоеточия. Остальная часть строки —

комментарий). Если файла не существует, выражение возвращает 1, после чего исполняется блок **else**, где файл создается посредством команды **touch**.

Команда **test** позволяет проверить не только существование файла, а и его атрибуты. Все опции команды **test** и их значение приведены в таблице 13.3.

Таблица 13.3 Проверка атрибутов файла в сценариях

Опция	*Действие*
-f	Файл существует и это обычный файл
-d	Файл существует и является каталогом
-s	Файл существует и его размер больше нуля
-c	Файл существует и является специальным файлом символьного устройства
-b	Файл существует и является специальным файлом блочного устройства
-r	Файл существует и доступен для чтения
-w	Файл существует и доступен для записи
-x	Файл существует и доступен для выполнения

Установка кода завершения

Большинство сценариев в этой главе устанавливают код завершения, и теперь вы знаете, для чего он используется. Как вы видели, для возврата значения применяется оператор **exit**. Если код завершения не указан явно, программа возвращает код последней выполненной команды. Он сохраняется в переменной **$?** точно так же, как это происходит для любой программы. Код завершения сценария могут использовать другие программы для принятия решений. Особенно это полезно в случае, когда, например, один сценарий вызывается из другого.

Кроме того, помните, что ваши возможности не ограничены лишь значениями 0 и 1. Можно использовать любые числа. Желательно возвращать код 0 в случае успешного завершения, а другие значения при возникновении ошибок. Для обработки различных ошибок в одной программе можно использовать несколько кодов завершения. Это позволит программе, вызывающей сценарий, предпринять различные действия. Вот пример кода, где используется несколько кодов завершения:

```
if  [-r  program.conf]
then
      : # ничего не выполнять
else
       exit  1
fi
if  [touch  /tmp/program.lock]
then
       :  #  ничего  не  выполнять
else
      exit 2
fi
#  операторы  основной  программы
exit  0
```

Вначале сценарий проверяет возможность чтения файла **program.conf**. При отрицательном результате программа завершает работу с кодом 1. При положительном — переходит к следующему оператору **if** и пытается создать файл **/tmp/program.lock**.

Если процесс создания не завершается успешно, программа прекращает работу с кодом 2. В заключение, если обе операции были выполнены успешно, программа выполняет основной блок и возвращает 0. Этот сценарий можно вызывать из другой программы, проверяющей статус выхода. В зависимости от его значения родительская программа может предпринимать различные действия.

Перехват прерываний при выходе

Как известно, выполнение программы во FreeBSD можно прервать, послав какой-либо из сигналов **kill**, а также посредством различных комбинаций клавиш (например, **Ctrl-C**). Проблема заключается в том, что если программа создает временные файлы, а пользователь прерывает ее работу по **Ctrl-C**, созданные файлы не будут удалены. За короткий срок таких файлов может накопиться достаточно много. К счастью, командный интерпретатор позволяет перехватывать подобные прерывания. Ниже приведен короткий пример, иллюстрирующий перехват прерываний:

```
#!/bin/sh
# Программа демонстрирует перехват прерываний
trap 'echo "Interrupt received. Quitting." 1>&2' 1 2 3 15
echo -n "Enter a number: "
readln num
exit 0
```

Программа устанавливает перехват прерываний 1, 2, 3 и 15. Предпринимаемые действия задаются в одинарных кавычках. Если при запуске программы, после того как она выдаст приглашение "Enter a number", нажать **Ctrl-C**, она получит сигнал 2 (INT). Поскольку программа перехватывает это прерывание, она выдаст сообщение "Interrupt received" и завершит работу.

Здесь мы познакомились с новым аспектом команды **echo**. Фактически, это перенаправление вывода, обеспечиваемое самим интерпретатором. Команда **1>&2** в операторе **echo** перенаправляет вывод в поток **STDERR**. Поэтому специальное сообщение нельзя случайно перенаправить другой команде с помощью конвейера или в файл вместе с остальным выводом программы. Рекомендуется все сообщения об ошибках перенаправлять в поток **STDERR** командой **1>&2**.

Обычно перехват прерываний используется для операций типа удаления временных файлов. Если требуется выполнить несколько действий, желательно воспользоваться функцией, перехватывающей прерывание (о функциях рассказано далее в этой главе).

Если вы хотите исключить возможность выхода из программы по **Ctrl-C**, можно установить перехват прерывания, не исполняющий никаких действий. Например:

```
trap '' 2
```

Данный оператор приводит к тому, что сигнал 2 полностью игнорируется.

Программа может перехватывать несколько прерываний и выполнять различные действия в зависимости от того, как осуществляется выход из нее. Перехват сигнала 0 установлен для всех вариантов выхода из программы.

Перехват остальных сигналов осуществляется лишь тогда, когда они посылаются программе. Список наиболее часто перехватываемых сигналов приведен в таблице 13.4.

ПРИМЕЧАНИЕ

Сигнал 15 (посылаемый по умолчанию командой **kill**) и другие сигналы команды **kill** можно перехватывать, однако это не относится к сигналу 9 (**SIGKILL**). Он используется в качестве

последней возможности прервать работу программы, когда остальные методы не помогают. Поэтому его нельзя ни перехватить, ни игнорировать.

Таблица 13.4 Распространенные сигналы прерываний и их действие

Сигнал прерывания	*Действие*
0	Выход
1	HUP — Обрыв сеанса (или отсоединение)
2	INT — Прерывание (Ctrl-C)
3	QUIT — Выход (Ctrl-\)
15	TERM — Обычная команда **kill**

Функции

Функции представляют собой группы операторов, вызываемые одной командой. Их можно рассматривать как "минипрограммы внутри программ". Использование функций в сценариях облегчает программирование по двум причинам. Прежде всего, если определенный набор операций требуется выполнить в нескольких местах программы, достаточно воспользоваться лишь одной командой — именем функции. Во-вторых, если вы захотите изменить то, как исполняется операция, достаточно будет внести изменения лишь в одном фрагменте кода — в теле функции. Ниже приведен пример функции, которая удаляет временные файлы при завершении работы сценария. Здесь же показано, как вызывается эта функция:

```
#!/bin/sh
on_exit()  {
        rm  -rf  /tmp/myprogram.*
        mv  logfile  logfile.old
        mail  foo@bar.com  <  report.txt
}
trap  on_exit  0  1  2  3  15
```

Итак, как создается функция? Задается ее имя, за которым следуют круглые скобки и открывающая фигурная скобка. Все, что находится между фигурными скобками, представляет собой тело функции. Функция вызывается точно так, как и любая команда, — по имени. При этом выполняются все операторы, заключенные в фигурные скобки.

Между вызовом функции и вызовом другой программы существует важное различие. Функция выполняется текущим интерпретатором, а отдельная программа запускается в другой копии командного интерпретатора. Это значит, что функциям доступны и переменные среды, и внутренние переменные вызывающей ее программы. Отдельной программе, исполняемой другим интерпретатором, они недоступны.

Файловые дескрипторы

Файловые дескрипторы представляют собой числовые идентификаторы, устанавливаемые ядром при запуске каждого нового процесса. Процесс использует их для записи вывода и чтения ввода. По умолчанию командный интерпретатор открывает три файловых дескриптора:

F.D. 0 **STDIN**. Это стандартный входной поток. Обычно ввод поступает с клавиатуры, однако его можно перенаправить из файла или какого-либо другого источника.

F.D. 1 **STDOUT**. Это стандартный выходной поток. Обычно вывод поступает на экран, однако, как вы понимаете, его также можно перенаправить.

F.D. 2 **STDERR**. Это стандартный поток ошибок. Обычно выводится на экран, но и его можно перенаправить.

Применение файловых дескрипторов в сценариях делает написание программ более простым и эффективным. Для открытия файлового дескриптора используется команда **exec**. Ниже приведен пример использования файлового дескриптора:

```
#!/bin/sh
# открытие файлового дескриптора в потоке F.D. 1 (STDOUT)
exec > testfile.txt
# вывод информации в STDOUT, который теперь направлен в файл
  ↪testfile.txt
echo "Line 1 of the file"
echo "Line 2 of the file"
echo "Line 3 of the file"
echo "Line 4 of the file"
echo "line 5 of the file"
exit 0
```

Оператор **exec** во второй строке примера перенаправляет поток **STDOUT** в файл **testfile.txt**. В результате, операторы **echo** выводят информацию в **testfile.txt**, а не на экран, хотя в каждом из них поток не перенаправлен явно. Если в файл нужно вывести более одной строки текста, эффективнее использовать файловый дескриптор, а не перенаправление вывода в каждом операторе. Такую схему легче программировать.

Этот же подход применяется при работе с файловым дескриптором **STDIN** и командой **read**. Например:

```
#!/bin/sh
# открытие файлового дескриптора в потоке F.D. 0 (STDIN)
exec < testfile.txt
while read string
do
        echo $string
done
exit 0
```

Если файл **testfile.txt**, созданный в предыдущем примере, существует, то вывод программы будет выглядеть так:

```
Line 1 of the file
Line 2 of the file
Line 3 of the flie
Line 4 of the file
Line 5 of the file
```

Что же здесь интересного? Пример иллюстрирует важную концепцию использования файлового дескриптора с командой **read**. Поскольку между последовательными вызовами **read** файл не закрывается, **read** помнит последнюю прочитанную строку и передвигает указатель на следующую строку файла. Таким образом, **read** автоматически последовательно читает все строки файла.

Отладка сценариев командного интерпретатора

Независимо от сложности создаваемых вами сценариев, рано или поздно в них обнаружатся ошибки. Хотя интерпретатор не имеет полноценного отладчика, он

обеспечивает простейшие возможности для мониторинга всех выполняемых действий. Трассировка включается посредством опции **-xv** в строке **#!/bin/sh**. Выглядит это следующим образом:

```
#!/bin/sh -xv
```

Лучше всего применять ее совместно с перенаправлением вывода команде **more** или **less**, а также обоих потоков **STDOUT** и **STDERR** в определенный файл. Это позволит просмотреть и вывод самого сценария, и сообщений об ошибках. Вот несложный пример:

```
#!/bin/sh -xv
# Пример возможностей трассировки в сценарии командного
  интерпретатора
result=`echo "2 * 12 / (2 + 2)` | bc`
echo $result
exit 0
```

Для запуска программы и перенаправления потоков **STDOUT** и **STDERR** команде **more**, применяется следующая команда:

```
./xvtest 2>&1 | more
```

Вот вывод программы:

```
1.      ./xvtest 2>&1 | more
2.    #!/bin/sh -xv
3.    # Пример возможностей трассировки в сценарии командного
   ↪интерпретатора
4.    result=`echo "2 * 12 / (2 + 3)" | bc`
5.    + echo 2 * 12 / (2 + 3)
6.    + bc
7.    + result=4
8.    echo $result
9.    + echo 4
10.   4
11.   exit 0
12.   + exit 0
```

Здесь можно увидеть все действия, выполненные программой. Строки со знаком + представляют собой результаты этих действий. Например, в строке 3 результат вычислений присваивается переменной **result**. Предпринятые действия показаны в строках 4, 5 и 6. В строке 4 мы видим команду **echo**, в строке 5 — запуск **bc** и, наконец, в строке 6 — присвоение результата (4) переменной **result**.

В этом же примере показано, как работает "раскрытие" переменных. Обратите внимание на строки 8–10. В строке 8 интерпретатор читает выражение с оператором **echo**. В строке 9 раскрывается значение переменной, в результате чего выражение превращается в "**echo 4**". В строке 10 печатается реальный вывод оператора **echo**.

Расширенные возможности интерпретатора Korn

Командный интерпретатор Korn поддерживает несколько дополнительных свойств, которые делают программирование на нем более мощным и простым. Однако их применение следует тщательно обдумать. Причина заключается в следующем: почти любая система UNIX может запустить сценарий на языке интерпретатора Bourne, однако далеко

не каждая способна запустить сценарий интерпретатора Korn. Если сценарий предназначен для локальной системы, у вас не возникнет никаких проблем. Если же вы пишете сценарий для общего пользования, даже если он предназначен для использования на различных системах в пределах вашей организации, лучше ограничиться возможностями интерпретатора Bourne.

В этом разделе рассматриваются расширенные возможности интерпретатора Korn.

СОВЕТ

Большинство методов, о которых рассказано далее, поддерживаются и командными интерпретаторами стандарта **POSIX**, который входит в состав FreeBSD как **/bin/sh**. Если сценарий предназначен для других систем, где **POSIX** поддерживается не всегда, не следует использовать расширенные возможности, поскольку указание **#!/bin/sh** в первой строке сценария говорят системе, что она способна выполнить его. Это может привести к серьезным ошибкам.

Получение и установка интерпретатора Korn

Прежде всего, в системе необходимо установить командный интерпретатор Korn. Общедоступная версия Korn (The Public Domain Korn Shell) имеется на дистрибутивном компакт-диске FreeBSD как пакет (который можно инсталлировать из командной строки или с помощью утилиты **sysinstall**), а также как часть дерева портов в каталоге **/usr/ports/shells/pdksh**.

Чтобы для обработки программы вызывался интерпретатор Korn, необходимо заменить первую строку сценария **#!/bin/sh** на **#/usr/local/bin/ksh**. Korn является полностью совместимым снизу вверх с интерпретатором Bourne, т.е. любой сценарий, написанный для последнего запускается и в Korn. Очевидно, что обратное утверждение не будет верным: сценарий Korn не исполняется в интерпретаторе Bourne.

После инсталляции перейдем к рассмотрении свойств командного интерпретатора Korn.

Встроенная арифметика

Интерпретатор Korn имеет набор встроенных арифметических функций. Это значит, что в нем нет необходимости запускать программу **expr**, как это делалось в Bourne. Как и **expr**, арифметика в Korn ограничена целыми числами. Но поскольку это внутренние функции интерпретатора, они выполняются значительно быстрее, чем **exec**.

Воспользоваться встроенными арифметическими функциями Korn можно двумя способами. В первом из них используется оператор **let**. Например:

```
let x=7+5
```

В этой строке кода переменной **x** присваивается значение 12.

Во втором способе выражение заключается в двойные скобки, например:

```
if ((x < z))
```

Такой код читать легче, чем конструкции с оператором **let**. Кроме того, математические сравнения удобнее использовать в такой форме, а не с синтаксисом, присущим команде **test** в Bourne. В Korn можно пользоваться привычной записью < и > вместо **-lt** и **-gt**. В синтаксисе Korn символы, которые могут иметь специальное значение для интерпретатора, экранировать не нужно: можно, например, использовать запись **((5 * 3))** вместо **5 * 3**.

Математические операторы, поддерживаемые в арифметике Korn, приведены в таблице 13.5.

Таблица 13.5 Математические операторы Korn Shell Math Operators

Оператор	*Описание*
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
==	Равно
!=	Не равно
&&	Истинно, если оба выражения не равны нулю
||	Истинно, если одно из выражений не равно нулю
=	Присваивает значение выражения справа выражению слева
+=	Добавляет выражение в правой части к переменной в левой и сохраняет в ней результат
-=	Вычитает выражение в правой части из переменной в левой и сохраняет в ней результат
*=	Умножает выражение в правой части на переменную в левой и сохраняет в ней результат
/=	Делит выражение в правой части на переменную в левой и сохраняет в ней результат
%=	Делит выражение в правой части на переменную в левой и сохраняет в ней остаток от деления

Массивы

Командный интерпретатор Korn поддерживает массивы. Массив представляет собой переменную, содержащую несколько элементов. Они используются для группировки связанных элементов.

Массив можно рассматривать как ящик с множеством отделений. Каждое отделение имеет номер. Доступ к каждому из них осуществляется по имени массива, за которым следует номер. Для загрузки массива используется команда **set**. Предположим, к примеру, что нужно для определенного региона создать массив под названием **temperature**, содержащий среднюю температуру каждого месяца в году. Для этого используется следующая команда:

```
set -A temperature 57 52 58 61 63 65 71 70 68 66 64 62
```

Она создает массив **temperature** с 12 элементами в нем. Эту команду можно ввести в командной строке интерпретатора (запустить Korn можно с помощью команды **ksh**).

Для доступа к различным элементам массива используется индекс, следующий за именем массива. Индекс начинается с 0. Кроме того, поддерживаются символы-заместители. Вот несколько примеров:

```
ksh$ echo ${ temperature[0]}
57
ksh$  echo  ${  temperature[11]}
62
ksh$  echo  ${  temperature[*]}
57  52  58  61  63  65  71  70  68  66  64  62
ksh$
```

Массив может содержать до 512 элементов (0–511).

Для изменения значения одного элемента массива используется обычный оператор присваивания, например:

```
temperature[0]=55
```

Необходимо отметить следующие тонкости:

- В отличие от переменных, при обращении к массиву фигурные скобки являются обязательными. Команда **echo $temperautre[1]** не даст желаемого результата. Правильный синтаксис: **${ temperature[1]}**.
- Массив нельзя экспортировать как переменную среды.

Почему же не создать отдельную переменную для каждого месяца? Можно было бы поступить и так. Для сравнения ниже приведен пример, показывающий, как применение массивов значительно сокращает время, требуемое для программирования:

```
1.    #!/usr/local/bin/ksh
2.    # Иллюстрация массивов и расчет средней температуры.
3.    set -A temperature 57 52 58 61 63 65 71 70 68 66 64 62
4.    i=0
5.   printf "\nMonth\t\tTemperature\n\n"
6.   while ((i < 12))
7.    do
8.                  (( month = $i + 1 ))
9.                   printf "$month\t\t${temperature[$i]}\n"
10                   (( total_temp += ${temperature[$i]} ))
11.                 (( i += 1 ))
12.  done
13.  avg_temp=$(( total_temp / 12 ))
14.  echo
15.   echo    "Average temperature for whole year: $avg_temp"
16.   echo
  exit 0
```

Вот вывод программы:

```
Month                Temperature

1                    57
2                    52
3                    58
4                    61
5                    63
6                    65
7                    71
```

```
8                    70
9                    68
10                   66
11                   64
12                   62
Average temperature for whole year: 63
```

Сценарий может показаться сложным, но в нем не так уж много нового, хотя синтаксис несколько отличается от того, который мы использовали ранее. В этой программе присутствует и новая логика, с которой мы не сталкивались раньше. Рассмотрим, как работает эта программа.

- **Строка 1:** Обратите внимание, что в первой строке сценария указан командный интерпретатор **/usr/local/bin/ksh**. Если вы случайно укажете **#!/bin/sh**, программа работать не будет, а станет выдавать странные ошибки.
- **Строка 3:** В строке 3 для загрузки массива **temperature** двенадцатью значениями (средняя температура за каждый месяц) используется команда **set**.
- **Строка 4:** Счетчик цикла инициализируется нулевым значением.
- **Строка 6:** В строке 6 начинается цикл, исполняющийся до тех пор, пока **i** меньше 12.
- **Строка 8:** Здесь создается новая переменная **month**. Ее значение устанавливается равным **$i + 1**. Она используется для вывода номера текущего месяца. Почему к **$i** добавляется 1? Это связано с тем, что элементы в массиве нумеруются как 0–11 (и именно эти значения переменная принимает в цикле), а месяцы нумеруются как 1–12.
- **Строка 9:** Здесь выводится текущий месяц, за которым следуют два символа табуляции и значение элемента из массива **temperature**. Обратите внимание, что здесь используется подстановка переменных. Значение **$i** заменяется числом, сохраняемым в **$I**, а оно используется как элемент, извлекаемый из массива.
- **Строка 10:** В этой строке читается значение переменной **$total_temp**, к нему добавляется значение элемента массива **temperature** с индексом **$i**, а затем результат сохраняется в **$total_temp**. В итоге, переменная содержит суммарное значение среднемесячных температур для подсчета среднегодовой температуры.
- **Строка 11:** Здесь к текущему значению переменной **$i** добавляется 1. Если **$i** меньше 12, цикл продолжается.
- **Строка 13:** По завершении цикла подсчитывается среднегодовая температура: значение **$total_temp** делится на 12. Результат присваивается переменной **avg_temp**.
- **Строка 15:** Программа выводит значение среднегодовой температуры.

ПРИМЕЧАНИЕ

Согласно этой программе, среднегодовая температура равна 63. Это хороший пример того, что *интерпретатор поддерживает лишь целочисленную арифметику. Реальное значение составляет* 63.083333, однако интерпретатор отбрасывает десятичную часть, поскольку не умеет обрабатывать числа с плавающей запятой. Там, где требуется более высокая точность, необходимо воспользоваться командой **bc**, как это было сделано в примерах, приведенных ранее.

Вы заметили, как данный подход позволил сократить код программы? Вместо того чтобы обрабатывать каждый месяц вручную, мы смогли воспользоваться цик-

лом, который автоматизировал этот процесс. Если бы для сохранения среднемесячных температур использовались разные переменные, потребовалось бы гораздо больший объем программного кода.

ПРИМЕЧАНИЕ

Если вы хотите попрактиковаться в программировании, замените цикл **for** в приведенном примере на **while**. Помните, что индекс массива может быть звездочкой (символом-заместителем) — в этом случае массив выводит все элементы.

Подстановка команд

Командный интерпретатор Korn поддерживает более прозрачную форму подстановки команд, чем та, что применяется в Bourne. Вместо заключения команды в обратные кавычки применяется следующий синтаксис:

```
today_date=$(date)
```

Старый стиль также поддерживается. Выбор того, каким из них пользоваться, — за вами.

Использование getopts

getopts представляет собой более удобный метод обработки аргументов командной строки, чем синтаксис интерпретатора Bourne, показанный ранее в этой главе. Команда **getopts** дает возможность использовать стандартный синтаксис ***-option***, поддерживаемый большинством команд FreeBSD. Кроме того, она позволяет обрабатывать аргументы опций.

Общий синтаксис команды **getopts** выглядит так:

```
getopts опции переменная
```

где *опции* представляют собой набор опций, а *переменная* — имя переменной, в которой они будут сохранены. Если после опции стоит двоеточие, значит, для этой опции необходимо указать значение. Это значение сохраняется в специальной переменной **$OPTARG**. Другая специальная переменная под названием **$OPTIND** хранит значение текущего обрабатываемого аргумента.

Команда **getopt** запускается для каждой указанной опции. Если ей воспользоваться в цикле **while**, каждая итерация цикла будет выполнена для последующего аргумента командной строки. Ниже приведен пример того, как работает синтаксис команды **getopt**:

```
getopts abc:d: MyVar
```

Таким образом, **getopts** распознает опции **a**, **b**, **c** и **d**. Кроме того, опции **c** и **d** поддерживают значения, сохраняемые в переменной **$OPTARG**. Сама опция хранится в переменной **MyVar**. Предположим, например, что программа, содержащая команду **getopts**, называется **myprog** и запускается следующим образом:

```
./myprog -a -c foobar
```

Команда **getopts** выполняется дважды (например, в цикле **while**). После ее первого выполнения переменная **$MyVar** содержит **a**, после второго — **c**, а переменная **$OPTARG** — строку **foobar**. Использование этих переменных, а также упоминавшейся ранее переменной **$OPTIND**, позволяет применять команду **getopts** для принятия решений в сценариях и в тех же самых процедурах, примеры которых были приведены в этой главе, таких как циклы и условные операторы.

14 ГЛАВА

Мониторинг производительности, управление процессами и автоматизация заданий

- Мониторинг производительности. Утилита top
- Вывод утилиты top
- Мониторинг процессов. Утилита ps
- Объяснение вывода ps
- Прерывание неуправляемого процесса
- Изменение приоритетности процесса
- Автоматизация заданий
- Демон cron
- Однократный запуск заданий с помощью команды at
- Управление доступом к командам cron и at

Одним из фундаментальных различий между настольными и серверными операционными системами, такими как FreeBSD, является управление процессами. FreeBSD позволяет контролировать любой процесс в системе, начиная от тривиальных и заканчивая жизненно важными. Настольная система (Windows, например) позволяет управлять (в очень ограниченном смысле) лишь определенными процессами уровня приложений. Даже таблицу процессов в Windows мы видим, как правило, только в экстремальной ситуации. А вот во FreeBSD она составляет неотъемлемую часть ежедневного администрирования системы. Нажав комбинацию клавиш Ctrl+Alt+Delete в Windows можно получить список запущенных процессов, которые при желании можно прервать. Собственно, на этом все возможности и заканчиваются, так как из этого списка неясно, что делает каждый из процессов и сколько системных ресурсов он потребляет. FreeBSD позволяет получить эту информацию, а также предоставляет возможность перезапустить процесс, изменить его приоритет, передать ему какой-либо сигнал прерывания и т.д. Главное, система помогает установить, какой из процессов вызвал проблемы. Настольные системы представляют собой "машины с запаянным капотом". FreeBSD позволяет открыть капот в любой момент и разобраться в текущей ситуации.

Все это вовсе не значит, что FreeBSD менее стабильная система, чем настольные платформы с графическим интерфейсом. Администратор не может дестабилизировать систему, просматривая характеристики текущих процессов. Как вы увидите далее, таблица процессов и средства, взаимодействующие с ней, представляют собой наглядное воплощение того, что собой представляет UNIX-система. Это полностью доступная машина, где все "движущиеся части" видны и настраиваемы. Таким образом, во власти администратора настраивать все, что делает система, и изменять ее работу в нужную сторону. В этом и состоит сущность UNIX.

Мы рассмотрим несколько инструментальных средств, предназначенных для мониторинга процессов: **ps**, **top** и **kill**. Одни из них более дружественные к пользователю, другие — более гибкие. Кроме того, мы обратимся и к программе **cron**, представляющей собой планировщик, запускающий периодические задания. Вот вам и еще одно свойство, отличающее серверы от настольных машин.

Мониторинг производительности. Утилита top

Простейшей утилитой мониторинга процессов является **top**. Ее название восходит к тем временам, когда она служила для вывода списка десяти основных процессов — в порядке убывания потребления ресурсов процессора. Теперь программа **top**, входящая в состав FreeBSD, по умолчанию отображает все запущенные процессы независимо от их состояния. После инсталляции системы в ней, как правило, около 30 выполняющихся процессов.

Достоинствами **top** являются: интерактивность и выполнение в режиме реального времени. Будучи запущена, утилита выводит на терминал обновляемую ежесекундно информацию о состоянии системы в текущий момент. Ей можно передать такие команды, как **kill** и **renice** (о них рассказано далее в этой главе), или же опции, определенным образом фильтрующие вывод. Эти свойства делают утилиту незаменимым инструментом для устранения проблем на перегруженном сервере, средством тонкой настройки определенных задач или мониторинга системы. Утилиту можно запустить в одном из окон графической среды и, работая в этой среде, параллельно присматривать за системой.

Вывод утилиты top

Запустите программу — и вы увидите вывод, похожий на листинг 14.1:

Листинг 14.1 Пример вывода утилиты top

```
last pid: 30283;  load averages:  0.51,  0.89,  0.87    up 52+15:48:43
11:19:03
126 processes:  1 running, 124 sleeping, 1 zombie
CPU states:  0.7% user,  0.0% nice,  2.8% system,  0.7% interrupt,
↪95.8% idle
Mem:  142M Active,  35M Inact,  59M Wired,  7496K Cache,  35M Buf,  4256K
↪Free
Swap:  500M Total,  48M Used,  452M Free,  9% Inuse

  PID USERNAME PRI NICE   SIZE    RES STATE    C   TIME   WCPU    CPU COMMAND
19460 mysql     2    0 25908K 2692K poll     0 112:20  2.20%  2.20% mysqld
30283 bob       2    0  1360K  976K sbwait   1   0:00  6.02%  1.56% qpopper
30282 root     29    0  2076K 1236K CPU1     0   0:00  3.14%  0.93% top
  245 root      2    0   868K  232K select   1 177:39  0.00%  0.00% healthd
18427 root      2    0  7592K 5092K select   0  80:24  0.00%  0.00% named
86694 frank    10    0  1700K   56K nanslp   0  76:46  0.00%  0.00% elm
   86 root      2  -12  1296K  412K select   0   6:28  0.00%  0.00% ntpd
80717 root     10    0  2132K  472K nanslp   0   5:53  0.00%  0.00% telnetd
61945 root      2    0  1868K  360K select   0   3:29  0.00%  0.00% inetd
   80 root      2    0   916K  320K select   0   3:26  0.00%  0.00% syslogd
56054 root      2    0  2168K  584K select   0   3:12  0.00%  0.00% sshd
73772 root      2    0  8956K 2540K select   0   3:12  0.00%  0.00% httpd
40567 www       2    0  9880K 2872K sbwait   1   0:57  0.00%  0.00% httpd
40581 www       2    0 10008K 3796K sbwait   0   0:55  0.00%  0.00% httpd
```

По умолчанию **top** отображает все системные процессы независимо от того, кто является их владельцем, в каком состоянии эти процессы находятся (активный режим, холостой или "зомби") и сколько ресурсов процессора потребляют. Первый полезный блок информации находится во второй строке, это число процессов. Оно различно в разных системах, но, в большинстве случаев, все текущие процессы не помещаются на экране. Нажатие клавиши "i" переводит **top** в режим, когда отображаются только активные процессы.

Следующее, на что стоит обратить внимание, это показатель **load averages** (средняя загруженность) в первой строке. Набор выводимых значений позволяет "на глаз" оценить загруженность системы. Три числа представляют собой среднюю загруженность системы (связанную с количеством запущенных заданий) за последние 1, 5 и 15 минут, соответственно. Чем эти значения меньше, тем лучше.

СРЕДНЯЯ ЗАГРУЖЕННОСТЬ

Со временем, вы увидите, какое приложение приводит к высокой загруженности системы. Обычно на серверах эти значения не превышают 1. В настольных системах, где активно применяются графические средства, они достигают 2–3. А 5 — это уже высокая загруженность. Определенные демоны перестают обрабатывать новые запросы при достижении определенного уровня загруженности. Так, например, для **sendmail** — это 12. Если показатель достигает 20–30, в системе наверняка образовалось зацикливание (его называют также *состояние гонки — race condition*), когда процессы возникают быстрее, чем система успевает завершать их. Таким образом, система все время замедляется, ее загруженность растет. Это состояние называют "спиралью смерти" (death spiral).

Это один из тех редких случаев, когда необходимо перезагрузить UNIX-систему, поскольку столь сильно загруженный сервер не может вернуться к нормальному состоянию (или завершение всех процессов займет столько времени, что проще и быстрее перезагрузить систему). В такой ситуации соединение по **telnet** или **ssh** может просто перестать отвечать (а открыть новое система не позволит). Поэтому регистрация с консоли (а иногда даже отключение питания) может оказаться единственным средством спасения.

Заголовок вывода также содержит информацию о текущем состоянии оперативной памяти системы. Здесь отображается состояние блоков памяти, включая и область подкачки (виртуальную память). Информация о памяти содержится в четвертой и пятой строках листинга 14.1. Не следует думать, что блок **Free** (Свободно) отвечает всей свободной памяти в системе. В нем показан тот объем памяти, который вообще не был задействовано с момента загрузки системы. Количество памяти, занятое процессами (т.е. программами, запущенными в данный момент и не простаивающими), указано в блоке **Active** (Активно). Остальные поля описывают другие состояния использования, которые могут быть или не быть взаимоисключающими. Поэтому сумма всех значений не обязательно равно физическому объему оперативной памяти в системе. Фактически, эта сумма, как правило, больше.

Назначение полей **Swap** (Подкачка) очевидно. В случае необходимости данные постранично перебрасываются в виртуальную память (копируются на диск и освобождают оперативную память). Утилита **top** отображает два поля: **Used** (Занято) и **Free** (Свободно). Поля **Swap** более информативны, чем поля отвечающие реальной оперативной памяти: если в области подкачки находится достаточно большой объем данных (в блоке **Used** мы видим 50% или более), значит, оперативной памяти недостаточно для выполнения всех текущих процессов. Если эта ситуация наблюдается часто, возможно, следует увеличить объем RAM. Ситуация, когда система FreeBSD исчерпала всю область подкачки, бывает редко. Но даже в этом случае (как, впрочем, и в большинстве UNIX-систем) ситуация в целом останется благоприятной: система будет выводить сообщения об ошибках, но стабильность не пострадает. Тем не менее иногда вы будете сталкиваться с непредсказуемым поведением или нестабильностью. Поэтому желательно, чтобы объем задействованной области подкачки был минимальным. Дело в том, что в оперативной памяти все выполняется гораздо быстрее...

Обратите внимание, что в столбце **CPU** процессы приведены в порядке убывания значений. Здесь показано, какой процент циклов процессора в данный момент использует каждый процесс. Не стоит думать, что сумма значений равна 100, поскольку большую часть времени процессор не загружен. Взгляните вновь на заголовок: строка **CPU states** сообщает, сколько ресурсов процессора задействовано в различных состояниях. Эти значения можно приблизительно соотнести с процентами в столбце **CPU**. Поле **WCPU** (weighted CPU, взвешенное значение ресурсов процессора) содержит процентное значение, о котором будет рассказано при обсуждении команды **ps** в следующем разделе.

Работа процессора разбита на циклы, несколько миллионов в секунду. Каждый из циклов является частью определенного процесса. Со временем процесс использует достаточное число циклов, чтобы это значение можно было измерить в секундах. Именно об этом и сообщает столбец **TIME**. Не путайте формат (здесь используется двоеточие) с форматом записи времени часы:минуты. На самом деле, эта запись указывает количество секунд использования процессора системой и пользователями, соответственно. Процессу может понадобиться несколько минут или даже часов, чтобы набралось измеримое зна-

чение. Поэтому, если вы видите большое число (как у процесса **mysqld** в листинге 14.1), оно, вероятно, отвечает процессу, выполняемому в течение недель. Некоторые процессы требуют много ресурсов процессора на этапе запуска. Последний случай легко заметить по значению в столбце **CPU**.

Далее мы обратимся к столбцам **SIZE** и **RES**. Значение **SIZE** — это размер памяти, выделенной процессу, включая такие компоненты, как текст, данные и стек. Поскольку их части доступны всей системе, это значение не отражает точно, сколько памяти использует процесс. **RES** показывает объем резидентной памяти, это значение следует добавить к используемой памяти. Оба значения являются "корректными", но **RES** отвечает традиционному объему памяти, используемому процессом (именно его сообщают такие системы, как Windows и Mac OS).

Остальные поля в выводе **top** либо не важны, либо очевидны. Колонка **C** сообщает, какой из процессоров используется, если их больше одного. **PID** — это идентификатор процесса, т.е. число, присваиваемое каждому процессу при запуске, **OWNER** — соответственно, пользователь, запустивший его. **STATE** сообщает, в каком из возможных состояний находится процесс, это значение не слишком информативно, кроме случая **zomb** или **zombie** (отвечает дочернему процессу, который завершился, но еще не освободил место в таблице процессов).

Чтобы правильно отсортировать информацию, утилите **top** можно задать несколько команд в интерактивном режиме. Нажатие клавиши **i** отображает только активные процессы, нажатие **u** запрашивает имя пользователя, после чего отображаются только его процессы. Для вывода информации обо всех процессах в качестве имени пользователя следует указать +. Клавиша **k** отвечает команде **kill**, при этом запрашивается идентификатор процесса. Клавиша **t** включает (отключает) отображение характеристик самого процесса **top**. Эти и другие опции подробно объясняются в справочном руководстве **man top**.

Набор перечисленных свойств делает **top** удобным средством, позволяющим получить сводку о том, что происходит в системе, и управлять процессами. Но это не полное решение: **top** не предоставляет подробной информации о самих процессах, а его интерактивная природа не позволяет использовать его в сценариях или конвейерах с другими программами. Для этих целей используется **ps**.

Мониторинг процессов. Утилита ps

В отличие от интерактивной программы **top**, выполняемой в режиме реального времени, **ps** работает наподобие **ls** (и названия у них похожи!). Утилита отображает мгновенное состояние всех процессов на момент ее запуска. Она выводит ту же информацию, что и **top**, а также некоторые дополнительные подробности.

Если запустить **ps** без аргументов, она по умолчанию выведет список процессов текущего пользователя (т.е. запущенных во время текущего сеанса работы в системе). Утилита поддерживает большое число опций командной строки, позволяющих добиваться вывода различных результатов. Все они документированы в **man ps**. Опции задаются, как всегда:

```
# ps -waux
```

Данная комбинация параметров задает следующее: вывод в широком (wide) формате (**w**), когда он не урезается по ширине экрана, а переносит длинные строки;

вместе с процессами выводятся имена пользователей (**u**), причем, отображаются все процессы в системе (**a**), в том числе и те, что не присоединены к терминалу (**x**). Короче говоря, данный набор опций позволяет увидеть все процессы в системе в максимально детализованном формате.

Вывод команды можно фильтровать, используя встроенные опции. Например, опустите опцию **x** — увидите лишь процессы, присоединенные к терминалу, опустите **a** — будут выведены процессы только текущего пользователя (или того, что задан опцией **-U**, например, **-U frank**). Другие опции описаны в руководстве **ps**. Кроме встроенных фильтров, **ps** можно применять в сочетании с **grep** (мы уже обсуждали это в главе 9), отсортировывая процессы, например, по имени:

Листинг 14.2 Пример вывода утилиты ps, отфильтрованный с помощью grep

```
# ps -ax | grep httpd
40563  ??  S      0:54.73 /usr/local/sbin/httpd
40564  ??  S      0:55.30 /usr/local/sbin/httpd
40565  ??  S      0:56.03 /usr/local/sbin/httpd
40566  ??  S      1:00.16 /usr/local/sbin/httpd
40567  ??  S      1:05.13 /usr/local/sbin/httpd
```

Объяснение вывода ps

Рассмотрим пример вывода команды **ps -waux**:

Листинг 14.3 Пример вывода команды ps

```
USER   PID  %CPU %MEM VSZ  RSS  TT  STAT STARTED      TIME COMMAND
root     1   0.0  0.0 528   72  ??  ILs  31Jan01   1:05.94 /sbin/init --
root     2   0.0  0.0   0    0  ??  DL   31Jan01  14:41.63 (pagedaemon)
root     3   0.0  0.0   0    0  ??  DL   31Jan01   2:38.47 (vmdaemon)
root     4   0.0  0.0   0    0  ??  DL   31Jan01   0:33.21 (bufdaemon)
root     5   0.0  0.0   0    0  ??  DL   31Jan01  79:52.61 (syncer)
root    24   0.0  0.0 208    0  ??  IWs     -      0:00.00 adjkerntz -i
root    80   0.0  0.1 916  320  ??  Ss   31Jan01   3:27.80 syslogd -s
```

Процессы в выводе утилиты **ps** отсортированы по своим идентификатором (если не используются специальные опции сортировки по другим категориям). В итоге, в выводе присутствует та же информация, которую отображает **top**, но в несколько ином формате и с небольшими вариациями.

ПРИМЕЧАНИЕ

Единственное исключение состоит в следующем: столбец **%CPU** в выводе **ps** не совпадает с **CPU** в **top**, он является эквивалентом **WCPU**. В этом значении учитываются те циклы процессора, когда процесс находится в "резидентном" состоянии. В большинстве случаев разница незаметна, однако иногда между значением **ps** и **top** наблюдается значительное различие. В любом случае, **%CPU** представляет собой среднее значения за минуту, поэтому они не дают в сумме 100%.

Для просмотра метрических характеристик больше подходит утилита ***top***; ***ps*** предназначена в основном для просмотра идентификаторов определенных процессов и полных команд, посредством которых они были запущены. Она работает быстрее, чем **top**, и может быть использована в сценариях. Например, можно создать сценарий, извлекающий идентификатор определенного процесса и посылающий ему сиг-

налы (все это реализуется посредством команд интерпретатора и конвейеров). Это весьма полезно при работе с плохо контролируемым процессом или при изменении приоритета задачи.

Прерывание неуправляемого процесса

Предположим, что система работает медленнее, чем должна. После получения прав доступа **root** (посредством **su**) и запуска **top** администратор получает на экране следующую информацию:

Листинг 14.4 Вывод команды top, где представлен процесс, вышедший из-под контроля

```
last pid: 67469;  load averages:  8.32,  5.49,  2.47    up 53+01:04:22
➥20:34:42
90 processes:   1 running, 88 sleeping, 1 zombie
CPU states:  93.2% user,  0.0% nice,  0.2% system,  0.8% interrupt, 5.8%
➥idle
Mem: 153M Active, 23M Inact, 60M Wired, 7252K Cache, 35M Buf, 5112K Free
Swap: 500M Total, 44M Used, 456M Free, 8% Inuse

  PID USERNAME PRI NICE   SIZE    RES STATE  C   TIME   WCPU    CPU COMMAND
19460 frank      2    0 25908K 2816K poll   0 131:15  0.00% 92.43% testprog
67468 root      28    0  2036K 1024K CPU0   0   0:00  0.43%  0.20% top
  245 root       2    0   868K  232K select 1 178:48  0.00%  0.00% healthd
18427 root       2    0  7592K 5124K select 0  81:01  0.00%  0.00% named
```

Несложно установить, какой именно процесс тормозит работу системы. Похоже, что пользователь **frank** запустил экспериментальную программу. Возможно, она написана некорректно и поэтому съедает порядка 92% ресурсов процессора. Иногда такой "сбежавший" из-под контроля процесс — результат бесконечного цикла (блока кода, который не способен достичь точки, где он будет прерван) или утечки памяти (когда программа пытается получить все больше памяти, даже если в этом нет необходимости). Скорее всего, такая программа не сможет сама корректно завершить работу. Со временем, другим процессам становится все сложнее запускаться и нормально прекращать свою работу. Поэтому процесс **testprog** следует удалить.

Нажмите клавишу **k**. **top** выводит приглашение **kill** — интерфейс команды **kill**, точнее, ее упрощенной версии. Ей требуется идентификатор процесса, который будет прерван (в данном случае **19460**). Поскольку администратор обладает полномочиями **root**, процесс будет прекращен немедленно, и система вздохнет свободно. Теперь можно, например, послать пользователю сообщение о том, к чему привел запуск его программы.

Иногда такая схема не работает. Требуется более тонкий контроль над сигналами, которые посылаются процессам. Функциональность **kill**, обеспечиваемая утилитой **top**, подходит для прерывания несложных процессов. В общем случае применяется утилита **kill**.

Команда kill

Программа **kill** (убить) не просто прерывает процессы. Она представляет собой сигнальный механизм, посредством которого процессы могут передавать команды друг другу. Пользователь может удалить любой из своих процессов, но только **root**

способен устранять процессы других пользователей. Тем не менее основная задача программы **kill** — завершение работы процесса:

```
# kill 12553
```

Опции команды kill

Формат команды, показанный выше, заставляет систему переслать процессу определенный тип сигнала — **TERM**. Последний представляет собой универсальное сообщение "выход", которое понимают все программы UNIX. Однако иногда этого сигнала бывает недостаточно. Существует набор различных сигналов, наиболее важные из которых приведены в таблице 14.1.

Таблица 14.1 Широкоиспользуемые сигналы kill

Сигнал	*Символическое имя и значение*
1	**HUP** — отсоединение; другими словами, прерывание и перезапуск
2	**INT** — прерывание
3	**QUIT** — выход
6	**ABRT** — аварийное завершение
9	**KILL** — сигнал завершения процесса, который нельзя игнорировать
14	**ALRM** — тревога
15	**TERM** — корректное завершение процесса

Формат команды позволяет указывать номер сигнала или его символическое имя:

```
# kill -9 12553
# kill -HUP 12553
```

Первая команда посылает низкоуровневый сигнал **super-kill**, который останавливает любой процесс — независимо от его состояния и потребляемых ресурсов. Им следует пользоваться лишь тогда, когда сигнал **kill -TERM** не дает желаемого результата, поскольку безусловное завершение процесса является, в определенном смысле, некорректным: файлы и соединения могут остаться открытыми. Вторая команда заставляет процесс корректно завершить работу и перезапустить себя заново с теми же аргументами, что и раньше. При этом все входные файлы перечитываются, а процесс получает новый идентификатор. Эта команда оказывается полезной в том случае, когда вы внесли изменения в конфигурационные файлы и хотите, чтобы они вступили в силу (это часто требует перезапуска процесса).

Другие сигналы служат определенным целям, но в повседневной практике вы вполне сможете ограничиться лишь теми, о которых здесь было рассказано подробно. Большинство остальных сигналов программы просто игнорируют (если только они не были специально разработаны для реагирования на них).

Изменение приоритетности процесса

Мы обсудили команду **kill**, средство, которым можно воспользоваться и из командной строки, и интерактивно из утилиты **top**. Существует и другое средство, позволяющее повлиять на приоритетность процесса, — команда **renice**, изменяющая приоритетность (уровень **nice**) любого текущего процесса.

Приоритетность процесса представляет собой целочисленное значение в диапазоне от -20 до 20, причем наиболее высокое значение — это -20. Обратимся к выводу команды **top**: значения в столбце **NICE** представляют собой приоритеты процессов. Большинство процессов исполняются с нулевой приоритетностью. Она принята по умолчанию, так что в большинстве случаев указывать какой-либо определенный приоритет не требуется. Однако некоторые службы необходимо запускать на определенных уровнях **nice**, чтобы они могли запуститься в определенное время или же, наоборот, не препятствовали выполнению более важных процессов.

Обычные пользователи (не **root**) могут только понизить приоритетность процесса (т.е. установить более высокое значение **nice**). Для изменения приоритетности процесса на значение 10 используется следующий формат команды **renice**:

```
# renice -10 1442
```

ПРИМЕЧАНИЕ

В синтаксисе команды уровню **nice** предшествует дефис. Таким образом, в примере задан положительный (т.е. более низкий) приоритет. Чтобы указать отрицательный приоритет (более высокий), необходимо воспользоваться такой командой:

```
# renice --10 1442
```

Работая с **top** в интерактивном режиме, достаточно нажать **r** и программа выдаст приглашение **renice**, в котором необходимо ввести уровень приоритетности (от -20 до 20) и идентификатор процесса. Результат будет немедленно отображен в столбце **NICE**. Это простое интерактивное решение проблемы с программой **testprog**: вместо завершения процесса его приоритет устанавливается равным 20. Теоретически это позволит другим процессам выполняться безболезненно.

Установить уровень приоритетности процесса на этапе его запуска позволяет команда **nice**. Перед требуемой командой необходимо указать **nice** и значение приоритета:

```
# nice 10 ls
```

Таким образом, процесс **ls** запускается с уровнем приоритетности 10.

Автоматизация заданий

После того, как мы научились управлять процессами, так сказать вручную, мы исследуем, как можно запускать процессы автоматически, без вмешательства пользователя. Автоматизация заданий позволяет системе FreeBSD регулярно выполнять такие задачи, как проверка защиты, обновление состояния системы, очистка log-файлов и многое другое. В некоторых операционных системах планировщиками являются отдельные приложения (например, в Mac OS используется Software Update), но преимущество FreeBSD (и других UNIX-подобных систем) заключается в том, что планировщик является демоном, который способен запустить любую программу командной строки согласно установленного графика.

Демон cron

Как и во многих системах UNIX, планировщик FreeBSD называется **cron**. Программа **cron** для FreeBSD была написана Полом Викси (Paul Vixie), и эта же стандартная версия применяется в большинстве дистрибутивов Linux. Как и всякий уважаю-

щий себя демон, **cron** выполняется постоянно и проверяет свои входные файлы (называемые файлами **crontab**) каждую минуту. Таким образом он устанавливает, есть ли изменения в этих файлах, и проверяет, есть ли задания, которые необходимо выполнить в данный момент. Процесс **cron** сам по себе не требует перезапуска, он автоматически ежеминутно перечитывает изменения.

Существует глобальный файл **crontab** (**/etc/crontab**) и каталог (**/var/cron/tabs**), в котором пользователи могут создавать свои собственные файлы **crontab**. Казалось бы, изменения удобнее внеси в файл **/etc/crontab**. Однако в действительности, как и в случае с каталогами **/usr/local/etc/rc.d** и **/etc/rc.local** (см. главу 11), файл **/etc/crontab** следует оставить без изменений. Правильное решение — создать личный файл **crontab** пользователя **root** в каталоге **/var/cron/tabs**. Чуть далее мы разберем, как это сделать.

Анатомия файла crontab

Рассмотрим файл **crontab** отдельного пользователя:

```
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.tqWGz91396 installed on Thu Feb  1 09:29:43 2001)
# (Cron version -- $FreeBSD: src/usr.sbin/cron/crontab/crontab.c,v
  ↪1.12.2.1 2000 /11/09 11:05:36 dwmalone Exp $)
0 3 1,15 * * cat ~frank/faq.txt | mail -s "FAQ Auto-Post"
  ↪mylist@testsystem.com
```

В этом файле содержится лишь одно задание: посылка текстового файла на адрес списка рассылки. Оно запускается в 03:00 в первый и пятнадцатый день каждого месяца. Как же это достигнуто? Задание указывается в первых пяти полях строки данных, разделенных пустыми символами. (Три первых строки являются комментариями, генерируемыми автоматически. Демон **cron** не обрабатывает их.) Порядок полей и допустимые значения приведены в таблице 14.2.

Таблица 14.2 Поля даты и времени в файле crontab

Поле	*Допустимое значение*
minute (минуты)	0-59
hour (часы)	0-23
day of month (число месяца)	1-31
month (месяц)	1-12
day of week (день недели)	0-7

Каждое поле может содержать несколько значений, разделенных запятыми, или их диапазон в формате, например, "1–10". Символ звездочки (*) указывает на все возможные интервалы времени. Месяц и день недели (четвертое и пятое поля) могут быть заданы посредством символических имен (трехбуквенных аббревиатур). Имена могут быть перечислены в списке, разделенном запятой, но не в формате диапазона значений. В пятом поле (день недели) значения 0 и 7 отвечают воскресенью.

ПРИМЕЧАНИЕ

Если команду необходимо запускать каждые n минут или n часов, в файле можно указать "пошаговое" значение (например, "***/n**"). В поле минут ***/5** будет обозначать "каждые пять минут", т.е. будет эквивалентом записи 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55. О форматировании команд запуска заданий рассказано на странице руководства **man(5) crontab**.

Для часто используемых графиков запуска в первых пяти полях можно применять сокращенную запись. Примеры приведены в таблице 14.3.

Таблица 14.3 Примеры символической записи интервалов

Строка	*Эквивалентное значение*
@reboot	Запустить один раз, при запуске системы
@yearly	0 0 1 1 *
@annually	Совпадает с **@yearly**
@monthly	0 0 1 * *
@weekly	0 0 * * 0
@daily	0 0 * * *
@midnight	Совпадает с **@daily**
@hourly	0 * * * *

После пяти полей даты и времени в форматах файла **/etc/crontab** и файлов отдельных пользователей проявляются различия. Глобальный файл **/etc/crontab** содержит дополнительное поле перед полем **command** — поле **who** (кто), определяющее какой пользователь запустит процесс (а значит, будет его владельцем):

```
1       3       *       *       *   root   periodic daily
```

Поле **command** может быть сколь угодно сложным: в нем можно внести все, что можно указать в командной строке. В том числе можно воспользоваться точкой с запятой (;), разделяющей несколько команд, которые будут исполнены последовательно как части одного автоматизированного задания. Важно отметить, что для команд, запускаемых из глобального файла **/etc/crontab**, демон **cron** не учитывает значение переменной **PATH** пользователя, указанного в поле **who**, так как в начале файла задано значение **PATH**, включающее основные системные каталоги. Из этого следует, что каждая команда должна быть указана с полным путем к ней, иначе демон **cron** не сможет ее запустить.

Если команда, выполняемая **cron**, сопровождается выводом информации, весь вывод собирается в сообщение, пересылаемое затем владельцу файла **crontab** (или задания, если используется глобальный файл **/etc/crontab**).

Создание и редактирование файлов crontab

Обладая полномочиями **root**, несложно редактировать файл **/etc/crontab**, добавляя в него любые задания. Однако этого следует избегать (о чем уже говорилось ранее). Поэтому мы рассмотрим, как создавать файлы **crontab** отдельных пользователей. Поскольку файл **crontab** размещается в центральном каталоге (**/var/cron/tabs**) с правами доступа **0600**, механизм защиты позволяет пользователям создавать свои файлы, никак не влияя на чужие. Этот механизм реализует программа **crontab**:

```
# crontab -e
```

Если вы, как **root**, хотите отредактировать файл **crontab** пользователя Frank, применяется синтаксис:

```
# crontab -e -u frank
```

crontab подобен обсуждавшимся ранее средствам **chfn** и **edquota**. Он вызывает редактор, заданный переменной среды **VISUAL** (или **EDITOR**, если предыдущая не установлена). Содержимое файла (за исключением трех первых строк с комментариями) отображается в редакторе, позволяющем внести изменения, сохранить файл и выйти из него. Временный файл, который на этапе редактирования находится в каталоге **/tmp**, копируется в каталог **/var/cron/tabs**, при этом к нему добавляется заголовок. Файл **crontab** будет активным уже через минуту, когда демон **cron** перечитает его.

Однократный запуск заданий с помощью команды at

Итак, демон **cron** идеально подходит для периодического выполнения заданий. Но как быть с заданием, которое нужно запустить лишь один раз (например, перенести что-либо на более позднее время)? Такую задачу можно решить и с помощью **cron**, добавив в файл **crontab** запись **crontab -u**, удаляющую его. Существует и более удобный способ — программа **at**.

Фактически, **at** — это набор команд: **at** (запуск заданий), **atq** (отображение очереди заданий) и **atrm** (удаление заданий из очереди). Существует также команда **batch**, подобная **at**, которая запускается тогда, когда загруженность системы меньше определенного значения (по умолчанию — 1.5).

Когда команда **at** используется для создания заданий, она читает команды построчно, как сценарий в стиле **/bin/sh**. Эти команды можно указать или в командной строке (стандартный ввод) или в файле. В любом случае за именем команды следует строка с указанием времени. Она имеет интуитивно понятный вид, как ясно из таблицы 14.4.

Таблица 14.4 Примеры синтаксиса команды at

Командная строка	*Значение*
at 10pm	Запускается в 10:00 вечера текущего или (если уже больше 22:00) следующего дня
at 8:00am May 15	Запускается в 8:00 утра 15 мая
at midnight Jan 1 2002	Запускается на первой секунде 2002 года
at teatime tomorrow	Запускается в 16:00 следующего дня

За подробным обсуждением опций форматирования команды **at** следует обратиться к странице руководства **man at**. После ввода такой строки и нажатия Enter программа переходит в интерактивный режим, ожидая построчного ввода команд. В конце каждой строки, как обычно, следует нажимать Enter. По окончании ввода для выхода следует нажать **Ctrl+D**. После этого задание будет размещено в очереди. Можно просто указать входной файл:

```
# at -f mycommands noon + 5 days
```

Команда **at** читает команды из файла **mycommands** (подобного пакетному файлу DOS) и, используя интерпретатор **/bin/sh**, запускает их в полдень через пять дней от текущего момента.

Просмотреть состояние очереди заданий позволяет команда **atq**:

```
# atq
Date                    Owner       Queue       Job#
23:00:00   03/28/01     root        c           2
```

Отменить выполнение задания можно с помощью команды **atrm**:

```
# atrm 2
```

ПРИМЕЧАНИЕ

За выполнение заданий **at** отвечает команда **atrun**, которая запускается каждые пять минут (в стандартной инсталляции FreeBSD) и читает очередь заданий всех пользователей. Она выполняет те задания, время которых уже пришло. Изменить интервал запуска команды **atrun** можно в файле **/etc/crontab**, однако необходимости в этом, видимо, нет.

Управление доступом к командам cron и at

Запуск заданий по графику является столь мощным средством, что администратор должен ограничить возможности пользователей в доступе к командам **cron** и **at**. Предположим, например, что один из пользователей запускает робот IRC **eggdrop**, и всякий раз, когда вы прерываете этот процесс, он перезапускается, поскольку пользователь настроил запуск этого задания в файле **crontab**, если оно не выполняется. Пользователь не отвечает на почтовые сообщения. Что же делать? Вы можете или запретить учетную запись пользователя (это, однако, варварский метод) или ограничить его доступ к командам **cron** и **at**. Именно для этого и предназначены файлы **deny** и **allow**.

В нормальном состоянии файлов **/var/cron/allow** и **/var/cron/deny** не существует: каждый пользователь имеет возможность создавать файлы **crontab**. Файл **/var/cron/allow** (когда он существует в системе) содержит список (обычный текстовый файл, где имена пользователей заданы по одному в строке) пользователей, которым разрешено создавать файлы **crontab**. Можно поступить и по-другому: внести в файл **/var/cron/deny** тех пользователей, которым запрещено создавать файлы **crontab**. Если в системе присутствуют оба файла, преимущество имеет **/var/cron/allow**.

Файлы **/var/at/at.allow** и **/var/at/at.deny** работают аналогично. Обратите внимание на несколько иной формат имен.

15 ГЛАВА

Установка дополнительного программного обеспечения

- Пакеты FreeBSD
- Установка пакетов
- Удаление пакетов
- Обновление пакетов
- Стратегия портов
- Дерево портов FreeBSD
- Инсталляция портов
- Удаление инсталлированных портов
- *Обновление порта*
- Как обновить локальное дерево портов?
- Web-сайт Fresh Ports

До настоящего момента мы рассматривали темы, общие для различных типов системы UNIX. Большинство из них, и коммерческие, и распространяемые с открытым кодом, во многом похожи, включая структуру файловой системы, автоматизацию процессов и общие принципы администрирования. Теперь мы перейдем к более специфической теме.

Что отличает систему FreeBSD от ее BSD-собратьев, Linux и коммерческих версий UNIX? Главное — это модель, согласно которой администратор устанавливает новое программное обеспечение.

В Solaris, если требуется добавить новое приложение, необходимо найти двоичный файл, скомпилированный под используемую аппаратную платформу и самостоятельно инсталлировать его (обычно без помощи программ установки, которые размещают файл в нужном месте). Это связано с тем, что Solaris, как правило, поставляется без **gcc** (стандартного компилятора GNU C/C++). Другие системы, скажем, IRIX и HP-UX, содержат **gcc**, однако для них необходимо подыскать исходный код программы, запустить сценарий конфигурирования, скомпилировать (зачастую это легко сказать, но трудно сделать) и только потом инсталлировать готовое приложение.

Различные версии Linux продвинулись значительно дальше в этом вопросе, перейдя к концепции "пакетов", где в одном файле содержится двоичный файл, все необходимые библиотеки, а также информация, управляющая правильной инсталляцией приложения. В частности, RPM (Red Hat Package Manager), поставляемый по лицензии GNU, представляет собой диспетчер пакетов, используемый на многих платформах, включая и FreeBSD. Именно его популярность вознесла Red Hat на вершину популярности, сместив предыдущего фаворита — Slackware. RPM позволил администраторам хранить записи об установленном программном обеспечении и обновлять, деинсталлировать или добавлять новые пакеты с беспрецедентной легкостью.

Во FreeBSD процесс установки программного обеспечения организован еще более удобно. В системе имеется собственный диспетчер пакетов (набор утилит **pkg_***), о котором будет рассказано далее, а также дополнительная подсистема, называемая "портами". Последняя позволяет компилировать программное обеспечение из исходных файлов. Сначала мы рассмотрим пакеты FreeBSD, а затем перейдем к портам.

Пакеты FreeBSD

Джордан Хаббард (Jordan Hubbard), один из главных разработчиков FreeBSD, выполнил большую часть работы по созданию системы пакетов. Система, появившаяся во FreeBSD, была унаследована NetBSD и другими платформами. С годами благодаря усилиям независимых разработчиков она значительно улучшилась.

Система пакетов — это способ упаковки программного обеспечения (включая файлы **config**, разделяемые библиотеки и документацию), позволяющая с в дальнейшем извлечь все необходимое на другой машине. При этом автоматически формируется конфигурация, обеспечивающая работу программного обеспечения на этой системе. Ранние версии *диспетчеров пакетов выполняли именно такие задачи. RPM и упаковщик FreeBSD, кроме* того, предоставляют множество дополнительных функций. Они поддерживают базу данных, в которой указано, версии каких пакетов установлены на данной машине, позволяют установить файл с FTP-сервера, отслеживают зависимости и дополнительные пакеты, которые необходимо установить для корректного запуска данного пакета, обеспечивают возможность обновления или деинсталляции пакетов.

Различия эти связаны с разной философией разработки систем FreeBSD и Linux. Дело в том, что существует множество различных дистрибутивов Linux, содержащих разные версии **glibc** (набор основных разделяемых библиотек, не имеющий большого значения во FreeBSD) и запускающихся на разных аппаратных платформах. RPM должен быть достаточно сложным приложением, чтобы уметь обрабатывать все возможные ситуации. Интерфейс командной строки несколько туманен и требует хорошего знания документации. В отличие от Linux, операционная система FreeBSD разрабатывается централизованно и изначально была предназначена для одной аппаратной платформы, поэтому модель пакетов в ней значительно проще.

Как уже обсуждалось в главе 9, иерархия каталогов **/usr/local** предназначена для тех элементов системы, которые устанавливает администратор, это относится в первую очередь к программному обеспечению, инсталлируемому из портов и пакетов. Структура каталогов внутри **/usr/local** приведена в табл. 15.1.

Таблица 15.1 Структура каталогов внутри /usr/local

Подкаталог	*Назначение*
bin	Двоичные файлы (скомпилированные программы)
etc	Конфигурационные файлы
include	Заголовочные файлы C, используемые при компиляции нового программного обеспечения
info	Вспомогательные данные для создания документации
lib	Разделяемые библиотеки
libexec	Вспомогательные двоичные файлы, используемые другими программами
man	Страницы справочных руководств установленного программного обеспечения
sbin	Системные двоичные файлы (программы, изменяющие поведение системы)
share	Материалы, не зависящие от платформы (файлы данных, документация и т.д.)
var	Рабочие файлы для установленного программного обеспечения

Другими словами, это та же самая иерархия, что и в каталоге **/usr** (за исключением каталога справочных руководств, которые находятся в **/usr/local/man**, а не в **/usr/local/share/man**, как можно было бы ожидать). Пакетная система FreeBSD направляет все инсталлируемое программное обеспечение в каталог **/usr/local** за пределы дерева **/usr**, поддерживая тем самым строгое разделение между базовой и пользовательской инсталляцией.

Разделяемые библиотеки и зависимости

Разделяемая библиотека (*shared library*) представляет собой централизованный файл, который обеспечивает вызов прекомпилированных функций. Используя разделяемую библиотеку, программа получает доступ к определенным функциям, которые не нужно компилировать вместе с ней. Такой подход уменьшает размер файлов и избыточность информации. Он называется динамической сборкой (dynamic linking) в противоположность статической сборке (static linking), когда все необходимые функции компилируют-

ся вместе с программой. Разделяемые библиотеки существуют практически на всех платформах. Схема именования их, естественно, различна. Так, например, в Windows они называются DLL (Dynamic Link Libraries, библиотеки динамической компоновки).

FreeBSD включает в исходную инсталляцию большое число разделяемых библиотек, достаточное для поддержки базового программного обеспечения, а также ряда программ, которые будут инсталлированы позднее. Но это вовсе не значит, что вам никогда не встретится программа, которой требуется библиотека, не включенная в состав FreeBSD. Основные разделяемые библиотеки находятся в каталоге **/usr/lib**, а все, что инсталлирует администратор, попадает в каталог **/usr/local/lib**. Все программы автоматически ищут разделяемые библиотеки в каталоге **/usr/lib**, а затем в **/usr/local/lib**. Пути поиска можно отредактировать в файле **/etc/rc.conf** (см. главу 11).

Каждый пакет содержит список зависимостей, включающий и разделяемые библиотеки, и выполняемые файлы (программы). Если файл, указанный в зависимостях, не установлен, пакетная система автоматически отслеживает это и инсталлирует его, прежде чем продолжить инсталляцию.

Получение информации об инсталлированных пакетов

Первый шаг в работе с пакетами — это получение информации о том, что уже инсталлировано. Для работы с пакетами используются следующие утилиты: **pkg_add**, **pkg_delete**, **pkg_info**, **pkg_update**, **pkg_version** и **pkg_create**. Каждая из них совершенно прозрачно соответствует своему названию. Кстати, это одна из отличительных особенностей стиля FreeBSD: использовать программы с "говорящими" именами значительно проще, чем запоминать массу не вполне очевидных ключей и параметров одной программы. Вышеуказанные утилиты взаимодействует с базой данных в каталоге **/var/db/pkg**, к которой можно легко обратиться с помощью команды **ls**. В этом каталоге каждому установленному пакету соответствует каталог, содержащий информацию о списке пакета (т.е. о файлах в нем) и о его зависимостях. **pkg_info** использует эту базу данных для вывода краткого описания каждого пакета, как видно из листинга 15.1.

Листинг 15.1 Пример вывода команды pkg_info

```
# pkg_info
ImageMagick-5.2.7_2 An X11 package for display and interactive
                    manipulation of i
analog-4.16         An extremely fast program for analyzing WWW
                    logfiles
apache-1.3.19       The extremely popular Apache http server. Very
                    fast, very c
arc-5.21e.8         Create & extract files from DOS .ARC files
aub-2.0.5           Assemble usenet binaries
autoconf-2.13       Automatically configure source code on many Un*x
                    platforms
dict-1.4.9          Dictionary Server Protocol (RFC2229) client
elm-2.4ME+68        A once-popular mail user agent, unofficial clone
emacs-19.34b        GNU editing macros
```

Информацию о каждом отдельном пакете позволяет получить ключ **-v**. Как и в RPM, имена пакетов во FreeBSD включают несколько частей, разделенных дефисами, — имя пакета и его версию. Однако RPM-файлы, как правило, предназначены

для различных платформ, поэтому в их именах присутствует, кроме того, и "платформная" часть. Во FreeBSD пакетный файл, как правило, имеет, например, такой формат имени — **name-version.tgz**, например **bzip-0.21.tgz**. Большинство пакетов имеет расширение **.tgz** (tar-архивы). Это сокращение от **.tar.gz** (традиционный способ архивирования структуры каталогов в UNIX: вначале весь каталог упаковывается в один файл программой **tar**, а затем сжимается **gzip**). При использовании таких утилит, как **pkg_info -v** необходимо указывать полное имя пакета, включая его версию, как показано в листинге 15.2.

Листинг 15.2 Информативный вывод команды pkg_info

```
# pkg_info -v pgp-2.6.3ia
Information for pgp-2.6.3ia:

Comment:
PGP MIT or International version - Public-Key encryption for the
masses

Depends on:
Description:
PGP (Pretty Good Privacy) is a public key encryption pack-
age to protect E-mail and data files.  It lets you commu-
nicate securely with people you've never met, with no
secure channels needed for prior exchange of keys.  It's
well featured and fast, with sophisticated key management,
digital signatures, data compression, and good ergonomic
design.
WWW:  http://www.pgpi.org/

Packing list:
        Package name:  pgp-2.6.3ia
        CWD to /usr/local
File:  man/man1/pgp.1.gz
        Comment:  MD5:a0ab17d1fe83aaf159cb80fa1abf5462
File:  bin/pgp
        Comment:  MD5:625e99562f936a3d9b0ac3c5d5a94ba9
File:  lib/pgp/pgp.hlp
        Comment:  MD5:d5da3783ea26bc60f4b7584df4227866
File:  lib/pgp/pgpdoc1.txt
        Comment:  MD5:260ca85cd0263275cb7df6cd276e2b9f
File:  lib/pgp/pgpdoc2.txt
        Comment:  MD5:e3defe467fbf5c5c4809f8b5c13404a1
File:  lib/pgp/language.txt
        Comment:  MD5:bcec0f56b207846725fe7e4a612383ef
File:  lib/pgp/config.txt
        Comment:  MD5:b2518ad2566a9a4bce071936311d3c93
        Deinstall directory remove:  lib/pgp
        UNEXEC 'if [ -f %D/info/dir ]; then if sed -e '1,/Menu:/d' %D/
➥info/dir | grep -q '^[*] '; then true; else rm %D/info/dir; fi; fi'
```

Листинг сообщает все, что требуется знать о пакете: от его описания до зависимостей и списка файлов (для каждого из них контрольная сумма рассчитана по алгоритму MD5). В некоторых файлах содержится дополнительная информация, скажем, "сценарии деинсталляции" — задания, запускаемые *диспетчером при удалении* пакета. В приведенном примере каталог **lib/pgp** из локального дерева (**/usr/local/lib/pgp**) при деинсталляции удаляется. Из листинга видно, что пакет создает каталог для своих разделяемых библиотек в **/usr/local/lib**. Это возможно, поскольку поиск в каталогах с разделяемыми библиотеками производится рекурсивно. Аналогично, диспетчер

пакетов запускает строку **UNEXEC** для удаления файла **/usr/local/info/dir**. Обратите внимание, что для этого используется набор условных операторов. Таким образом, корректно созданный пакет при деинсталляции не оставляет следов своего пребывания в системе.

Как узнать, что текущая версия пакета является последней? Для этого используется утилита **pkg_version**. Она работает лишь в том случае, если установлен набор портов. Последний необходимо инсталлировать. О портах рассказано далее в этой главе. Для понимания работы **pkg_version** важно отметить, что набор портов имеет листинг текущих версий каждого пакета. Программа **pkg_version** сравнивает версию каждого установленного пакета с версией из набора портов и сообщает, какие пакеты находятся в актуальном состоянии, а какие нет (см. листинг 15.3).

Листинг 15.3 Пример вывода команды pkg_version

```
# pkg_version -v
ImageMagick-5.2.7_2          <      needs updating (port has 5.2.9_1)
apache-1.3.19                =      up-to-date with port
arc-5.21e.8                  =      up-to-date with index
aub-2.0.5                    =      up-to-date with index
autoconf-2.13                =      up-to-date with index
bash-2.04                    *      multiple versions (index has 1.14.7,2.04)
bnc-2.8.2                    =      up-to-date with index
bulk_mailer-1.12             <      needs updating (index has 1.13)
bzip2-0.9.5d                 <      needs updating (index has 1.0.1)
cclient-4.8                  <      needs updating (index has 2000c)
cvsup-16.1                   =      up-to-date with index
demoroniser-1.0              =      up-to-date with index
```

Без опции **-v** утилита **pkg_version** не отображает третьего столбца, о версии сообщает лишь символ во втором. Кроме того, в первом столбце в имени пакета не отображается подстрока версии. В любом случае, эта утилита позволяет быстро проверить, какие части системы требуют обновления.

Установка пакетов

Теперь приступим к установке программного обеспечения. Первый шаг — определить, что доступно. Прежде всего, следует отметить, что все, что доступно для Linux, доступно и для FreeBSD, поскольку совместимость на уровне двоичных файлов позволяет запускать любые приложения, включая проигрывание аудио- и видеофайлов с помощью программного обеспечения Linux.

Если в системе установлен набор портов, просмотреть все доступное программное обеспечение по категориям можно в каталоге **/usr/ports**. О наборе портов мы расскажем чуть далее. Существует и более удобный интерфейс **/stand/sysinstall**.

Установка с помощью sysinstall

В главном меню программы **sysinstall** (см. рис.15.1) выберите пункт Configure, а затем Packages. Если у вас компакт-диск с дистрибутивом FreeBSD, в качестве инсталляционного носителя выберите CD-ROM. Если вы подключены к Internet, выберите FTP.

После получения списка пакетов (независимо от того, какой носитель вы выбрали) на экране появится меню, приведенное на рис. 15.1. Для просмотра различных категорий необходимо воспользоваться стрелками курсора и клавишей Enter.

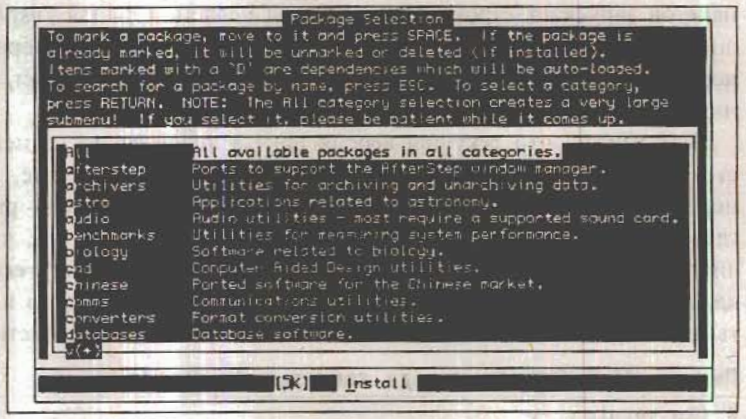


Рисунок 15.1.
Меню Packages в программе sysinstall.

О ВЕТВЯХ ВЕРСИЙ

О системе ветвления версий во FreeBSD будет рассказано в главе 18. Здесь нужно сделать несколько замечаний о терминах, важных для понимания работы пакетов и портов.

В разделе Options программы **sysinstall** существует поле, в котором устанавливается имя релиза. Если вы пользуетесь "релизом" FreeBSD (т.е. версией, поставляемой на компакт-диске), то поле Release Name установлено корректно. Но если вы обновили систему до промежуточного состояния ветвью **–STABLE** или **–CURRENT** (подробнее об этом — в главе 18), в поле может быть установлено значение, приводящее к некорректной работе, в частности, **#.#–STABLE** (где # обозначает главный и второстепенный номер версии). Это значение получено непосредственно из вывода команды **uname**, сообщающей информацию о версии ядра. Причина некорректной работы заключается в том, что в поле Release Name указан каталог на FTP–сервере, в котором **sysinstall** ищет пакеты. А текущее значение поля (отражающее версию системы) может не совпадать с реальным именем каталога, в котором находятся нужные пакеты.

Список каталогов релизов можно найти по адресу **ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/** (затем следует перейти в каталог, отвечающий вашей платформе). Если вы используете версию 4.2–STABLE, то вам нужен каталог релиза 4.x–STABLE. После выпуска следующей полной версии в текущей ветви, этот каталог будет переименован в 4.2–STABLE. Эта система именования не снабжена "защитой от дурака", поэтому необходимо следить за тем, чтобы она работала правильно. Указанный FTP–сервер безусловно заслуживает доверия, поскольку там имеются все требуемые пакеты. Достаточно скопировать имя каталога из последнего релиза в вашей ветви, и все будет работать правильно.

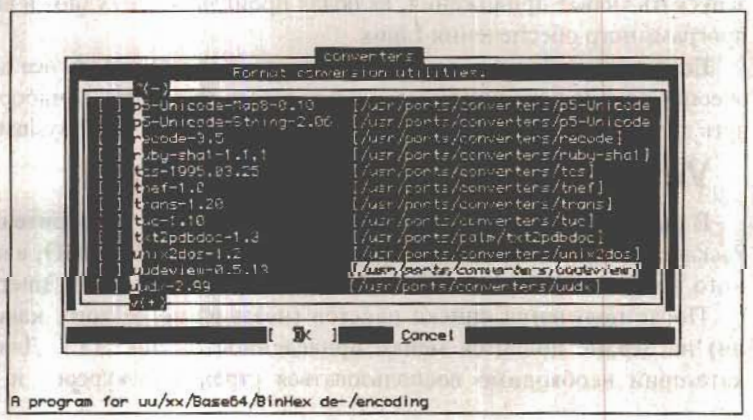


Рисунок 15.2.
Просмотр категории в меню Packages.

В нижней части экрана, показанного на рис. 15.2, в одной строке присутствует короткое описание пакета. Нужный пакет следует отметить с помощью пробела (возле него появится символ "X"). Для выхода из категории необходимо нажать Cancel (нажмите клавишу "стрелка вправо"). После того, как вы выбрали все необходимые вам пакеты, выберите Install в нижней части экрана.

ПРИМЕЧАНИЕ

Отметьте, что это единственный способ выхода из меню Packages. Если даже вы не выбрали ни одного пакета для инсталляции, все равно необходимо нажать Install для выхода в главное меню утилиты **sysinstall** (программа сообщит вам о том, что ничего не устанавливается).

На данной стадии утилита загрузит каждый пакет из списка, распакует его в каталог **/usr/tmp** и установит с помощью программы **pkg_add**. Последняя представляет собой средство командной строки, используемое диспетчером пакетов FreeBSD (его можно применять, не прибегая к **sysinstall**). Если пакет содержит какие-либо зависимости, утилита **sysinstall** сначала установит все, что необходимо, а затем вернется к выбранному пакету. Чтобы выйти из **sysinstall**, воспользуйтесь опцией Exit Install.

Итак, все пакеты установлены. Документация (страницы справочных руководств) готовы к использованию, конфигурационные файлы — в каталоге **/usr/local/etc**, а выполняемые — в **/usr/local/bin**. Чтобы обновить список доступных программ, известных командному интерпретатору, воспользуйтесь командой **rehash** (или выйдите из системы и зарегистрируйтесь повторно). После этого программное обеспечение готово к использованию.

Как мы видели в главе 11, некоторые программы необходимо конфигурировать специальным образом. Перейдите в каталог **/usr/local/etc** и проверьте файл **config** (обычно, он имеет название типа **<имя программы>.conf** или, по крайней мере, содержит имя программы). Если имя файла заканчивается подстрокой **.sample**, это значит, что перед использованием программы необходимо изменить конфигурацию и переименовать файл (устранив **.sample**). Для этого файл следует открыть в текстовом редакторе и произвести необходимые настройки. О них можно узнать либо из комментариев, содержащихся в файле, либо на странице справочного руководства.

СОВЕТ

Не все программное обеспечение, доступное во FreeBSD, существует в виде пакетов. Иногда оно является слишком новым или быстро разрабатываемым, поэтому в листинге пакетов присутствует далеко не все, что есть. Полный набор программного обеспечения, которое можно установить, содержится в современной версии набора портов. Об этом будет рассказано далее.

Использование утилиты pkg_add

Процесс инсталляции имеет несколько опций. Поскольку местонахождение библиотек, выполняемых и конфигурационных файлов стандартно, система не выдает диалогового окна, запрашивающего, куда их следует устанавливать, какие части подлежат инсталляции и т.д., т.е. опции, которыми обычно сопровождается установка настольных операционных систем. Это прекрасный пример, во-первых, "закрытого" подхода FreeBSD (по сравнению с Linux), а во-вторых, централизованной модели распространения всего программного обеспечения с открытым кодом, существующего для платформы FreeBSD.

Иногда пакет требует дополнительного управления, большего, чем может обеспечить **sysinstall**. Примеры включают в себя пакеты, имеющие интерактивные сценарии предварительной инсталляции (позволяют устанавливать опции, специфичные для пакета). Именно в таких ситуациях применяется утилита **pkg_add**. Важно понимать, что она вызывается при инсталляции каждого пакета — или напрямую, или из **sysinstall**.

Программа **pkg_add** предназначена для работы с уже загруженными файлами **.tgz** (из дистрибутивного каталога) или с файлами на удаленном сервере. Например, две следующих процедуры эквивалентны:

```
# fetch ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/4.x-STABLE/
➥packages/www/roxen-1.3.111.tgz
#  pkg_add  roxen-1.3.111.tgz
```

и

```
#  pkg_add  ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/4.x-STABLE/
➥packages/www/roxen-1.3.111.tgz
```

Конечно же, вторая процедура удобнее не только потому, что она позволяет избежать нескольких шагов в загрузке и инсталляции пакета, но и потому, что она выполняет всю работу в каталоге **/usr/tmp** и удаляет ненужные файлы по окончании работы. Программа **pkg_add** отслеживает зависимости так же, как если бы она использовалась из **sysinstall**. Если посредством **pkg_add** добавляется пакет, имеющий зависимости, утилита автоматически загружает и инсталлирует все требуемые пакеты и лишь затем продолжает работу. В результате, по окончании работы в системе будет установлена полноценно функционирующая программа.

СОВЕТ

Использование опций **pkg_add -nv** позволяет увидеть список шагов, которые производит программа, не проводя реальной инсталляции (иногда это называют "холостой инсталляцией" — dry install).

В некоторых случаях инсталляция пакета завершается инструкциями о том, как осуществить конфигурирование. Это одно из преимуществ установки пакетов средствами командной строки, а не с помощью интерактивной программы **sysinstall**. С другой стороны, **sysinstall** не сообщает о размерах файлов в пакетах, поэтому при загрузке файлов вы даже приблизительно не знаете сколько времени это займет. Если для поиска пакета в FTP-каталоге вы воспользовались браузером, вы увидите его размер и сможете оценить время загрузки.

ПРИМЕЧАНИЕ ОБ ИСТОЧНИКАХ ПАКЕТОВ

Структура каталогов на FTP-сервере FreeBSD такова, что каждый пакет представляет собой символическую ссылку, поэтому размеры файлов напрямую недоступны. Однако, следуя по ссылкам можно перейти в каталог, содержащий непосредственно файлы.

Очевидно, что утилита **pkg_add** работает с любыми **.tgz**-файлами, не только с теми, что получены с сайта FreeBSD. Опасайтесь файлов из "подозрительных" источников. Добавление пакета требует определенного доверия, поскольку при этом автору пакета позволено размещать файлы в каталогах с правами на выполнение файлов, поэтому есть возможность перезаписать при этом существующие файлы. Вирусы и "троянские кони" не очень распространены в мире UNIX, однако осторожный администратор должен всегда помнить о них, особенно при инсталляции новых пакетов!

Рекомендуется работать с **sysinstall** или, по крайней мере, загружать **.tgz**-файлы с FTP-сервера FreeBSD. Это гарантирует, что эти пакеты проверены, а контрольные суммы **checksum** по алгоритму MD5 удостоверяют их целостность.

Кроме того, утилита позволяет указывать несколько пакетов, работать в информативном режиме, отключать возможность запуска сценариев до и после инсталляции и даже не вносить пакет в базу данных установленного программного обеспечения. Эти и другие возможности описаны на странице руководства **man pkg_add**.

Удаление пакетов

Утилита **/stand/sysinstall** не поддерживает интерфейса для удаления пакетов. Обычно он и не требуется, поскольку после инсталляции пакета вы обладаете всеми средствами для сбора информации о нем.

Для просмотра списка установленных пакетов используется утилита **pkg_info**. Можно просто просмотреть листинг каталога **/var/db/pkg**. Имена каталогов совпадают с именами пакетов. Если вам известно имя пакета, который требуется удалить, можно воспользоваться утилитой **pkg_delete**:

```
# pkg_delete roxen-1.3.111
```

Если пакет имеет зависимости, **pkg_delete** проверяет их и не продолжает процесс удаления, если только он не запущен с опцией **-f**, форсирующей деинсталляцию. Кроме того, утилита пытается исполнить сценарии деинсталляции и выполнить все "требуемые" операторы. Если условия не будут соблюдены, **pkg_delete** не завершит работу успешно (если только она не выполняется с опцией **-f**). Как программа **pkg_add**, эта утилита поддерживает набор опций, включая информативный режим (**-v**) и режим "холостой инсталляции" (**-n**).

Обновление пакетов

Программа **pkg_update** позволяет обновить установленный пакет новой версией **.tgz** файла. Утилита проверяет все зависимости, чтобы новая версия была установлена корректно. Необходимо загрузить пакет **.tgz** и запустить **pkg_update**, например:

```
# pkg_update newpackage.tgz
```

Стратегия портов

Итак, мы узнали о пакетах и средствах работы с ними во FreeBSD. Теперь мы познакомимся с портами FreeBSD.

UNIX-способ инсталляции нового программного обеспечения — его компиляция — не менее удобный, чем использование пакетов. При этом администратор поступает следующим образом: вначале он находит HTTP- или FTP-сайт с дистрибутивом (раньше использовались такие службы, как **gopher** или **archie**), загружает исходный код, упакованный в файле **.tar.gz** или **.tgz** и распаковывает его во временный каталог. Затем он знакомится с различными файлами **README**, содержащими специальные инструкции и запускает сценарий **configure**, проверяющий наличие в системе определенных функциональных вызовов (поскольку в мире UNIX они могут быть различными на разных платформах). Следующим шагом является компиляция программного обеспечения с помощью специальной утилиты компилятора **make**, которая читает файл сборки и необходимые шаги из файла под названием **Makefile** в главном каталоге исходного кода. После корректной (хочется надеяться) компиля-

ции, администратор находит скомпилированный выполняемый файл и вручную копирует его в точку, предназначенную для двоичных файлов (например, **/usr/local/bin**). Иногда в файле **Makefile** существует цель **install** и команда **make install** приводит к копированию файлов в соответствующие каталоги.

Вся эта процедура до недавнего времени была очень неточной. Иногда она работала, иногда нет. И нужно признать, что в большинстве случаев она работала так себе. Поддержка была невозможной, производительность — непредсказуемой, и за UNIX закрепилась репутация запутанной и сложной в использовании системы.

Однако появились порты FreeBSD. Они представляют собой способ компиляции программного обеспечения непосредственно из исходных файлов с помощью строго регламентированной, структурированной и автоматизированной процедуры, гарантирующей безопасность и целостность устанавливаемых программ, исходный код которых получен прямо с сайтов их разработчиков. Из этого следовало, что можно пользоваться последними разработками программных продуктов, не ожидая пока появятся скомпилированные пакеты (которые, кстати, могут и не запуститься на системах, настроенных нестандартным образом). Этот подход позволяет устанавливать тысячи компонентов программного обеспечения в соответствующие каталоги файловой системы FreeBSD автоматически. Все это достигается благодаря широкой сети разработчиков портов, которые следят за изменениями в них и поддерживают сценарии, обеспечивая корректное выполнение процедур сборки и установки программ в FreeBSD. Таким образом, "порт" — это просто набор сценариев и исправлений, расположенный в определенной точке системы FreeBSD и содержащий специальный **Makefile**, позволяющий устанавливать программное обеспечение одной командой: **make install**. Никакой загрузки, конфигурирования и копирования — все это будет сделано автоматически.

Система портов FreeBSD оказалось настолько удачной, что ее адаптировали и другие системы, в частности, OpenBSD и NetBSD. Успех связан с удовлетворением потребностей тех администраторов систем UNIX, которые предпочитают сами компилировать программное обеспечение (допустим, из соображений защиты), простотой отслеживания версий и поддержки пакетов.

Дерево портов FreeBSD

Набор портов находится в каталоге **/usr/ports**. Загляните в этот каталог, там вы увидите нечто похожее на листинг 15.4. Каждая категория, которая присутствовала в программе **sysinstall**, присутствует здесь как каталог. (Скорее всего, ваш список не совпадет на 100% с этим листингом, поскольку категории постоянно реорганизуются.)

Листинг 15.4 Каталоги в /usr/ports

```
# ls -sF /usr/ports/
total  1447
   1 .cvsignore      10 devel/          1 news/
1296 INDEX            3 distfiles/      1 palm/
  11 LEGAL            4 editors/        3 print/
   4 Makefile         2 emulators/      1 russian/
   1 Mk/              1 french/         1 science/
   2 README           1 ftp/            4 security/
   4 README.html      6 games/          1 shells/
   1 Templates/       1 german/         3 sysutils/
   1 Tools/           5 graphics/       4 textproc/
```

```
  1 archivers/          1 hebrew/             1 ukrainian/
  1 astro/              1 irc/                1 vietnamese/
  4 audio/              9 japanese/           6 www/
  1 benchmarks/         1 java/               3 x11/
  1 biology/            2 korean/             1 x11-clocks/
  1 cad/                3 lang/               1 x11-fm/
  2 chinese/            4 mail/               1 x11-fonts/
  1 comms/              3 math/               2 x11-servers/
  2 converters/         1 mbone/              3 x11-toolkits/
  2 databases/          4 misc/               2 x11-wm/
  1 deskutils/          7 net/
```

Несколько первых каталогов, названия которых начинаются с прописных букв (здесь используется то преимущество UNIX, что в именах файлов различается регистр), являются структурными элементами системы портов, помогающими этой системе работать. Остальные каталоги — это категории портов. Загляните в один из них — вы увидите сотни каталогов различных портов (см., например, листинг 15.5).

Листинг 15.5 Порты внутри категории (каталога)

```
# ls -sF /usr/ports/audio/
total 230
  1 Maaate/                   1 mpg123/
  5 Makefile                  1 mpg123.el/
 11 README.html               1 mpg321/
  1 afsp/                     1 mpmf20/
  1 amp/                      1 mq3/
  1 ascd/                     1 musicbox/
  1 aumix/                    1 musicbrainz/
  1 aureal-kmod/              1 mutemix/
  1 autozen/                  1 mxv/
  1 bladeenc/                 1 napster/
```

Такой просмотр портов не слишком эффективен, особенно для больших категорий. Почитайте файл **README.html**. Помните, мы упоминали краткое описание, присущее каждому пакету? Такие описания хранятся и в этом файле. Кроме того, **README.html** присутствуют на обоих уровнях каталогов, а также внутри почти каждого каталога портов. Просмотреть структуру в гипертекстовом виде позволяет Web-браузер локальной машины: Netscape, если вы используете рабочую станцию с X-Window, или **lynx** при удаленном доступе (последний вариант показан на рис. 15.3).

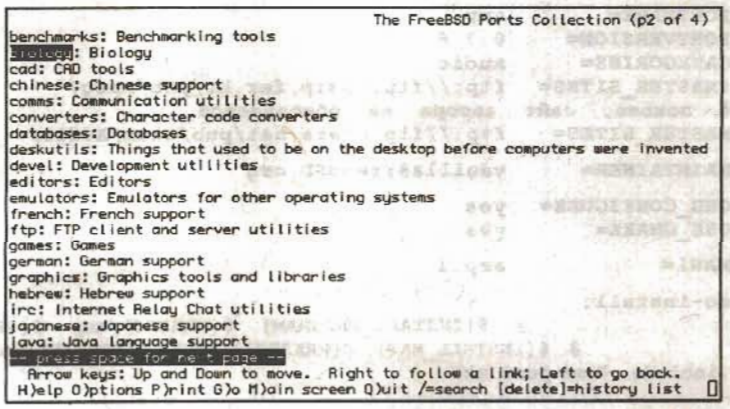


Рисунок 15.3.
Просмотр категорий в файле **README.html** *с помощью браузера* **lynx**.

Анатомия порта FreeBSD

Внутри каждого каталога порта содержится несколько файлов небольшого размера, полностью определяющих все действия, необходимые для компиляции и установки корректно загруженного архива с исходным кодом. Подробное описание этих файлов приведено в табл. 15.2.

Таблица 15.2 Файлы в каталоге порта

Файл	*Назначение*
Makefile	Содержит определенные переменные, используемые в процессе сборки, а также контактную информацию лица, поддерживающего данный порт.
README.html	Содержит краткое описание, или комментарий порта, в формате HTML.
distinfo	Контрольная сумма по алгоритму MD5, используемая для проверки целостности загруженного tar-архива.
files/ patch-aa patch-ab	Исправления, вносимые в исходный код после распаковки.
pkg-comment	Краткое описание порта
pkg-descr	"Длинное" описание порта; обычно включает и URL сайта разработчика дистрибутива.
pkg-plist	*Список распаковки*, в котором перечислены все файлы инсталляции, а также ключевые слова, сообщающие системе, какие действия следует производить при деинсталляции порта.

Файл **Makefile** содержит важнейшие элементы, необходимые для сборки, конфигурирования, установки и поддержки порта. Рассмотрим пример подобного файла (см. листинг 15.6).

Листинг 15.6 Типичный Makefile-файл порта

```
# makefile нового набора портов:        amp
# Дата создания:    23 июня 1997 года
# Автор:              Vanilla I. Shu <vanilla@MinJe.com.TW>
#
# $FreeBSD: ports/audio/amp/Makefile,v 1.10 2000/04/08 21:23:11 mharo
➥Exp $
#

PORTNAME=       amp
PORTVERSION=    0.7.6
CATEGORIES=     audio
#MASTER_SITES=  ftp://ftp.rasip.fer.hr/pub/mpeg/
# похоже, сайт автора не обновляется.
MASTER_SITES=   ftp://ftp.clara.net/pub/unix/Audio/

MAINTAINER=     vanilla@FreeBSD.org

GNU_CONFIGURE=  yes
USE_GMAKE=      yes

MAN1=           amp.1

do-install:
        @ ${INSTALL_PROGRAM} ${WRKSRC}/amp ${PREFIX}/bin
        @ ${INSTALL_MAN} ${WRKSRC}/amp.1 ${PREFIX}/man/man1
.include <bsd.port.mk>
```

Список переменных сообщает, где берет свое начало исходный код порта, какая версия (для **Makefile**) является текущей и как связаться с *лицом, поддерживающим порт* (*maintainer*). Последний является добровольцем, обычно не связанным с разработчиком программного обеспечения, чьей задачей является проверка того, что порт компилируется и устанавливается корректно, а также, что последняя версия программного продукта доступна как порт.

Большинство файлов **Makefile** содержат такие цели, как **clean**, **install** и **all**. Файлы **Makefile** в портах обычно не содержат этих целей, однако они включают центральный **Makefile** (**bsd.port.mk**, который находится в каталоге **/usr/ports/Mk** вместе с другими подключаемыми файлами). Этот файл содержит все стандартизованные цели сборки. Отдельные файлы **Makefile** в портах служат для отмены одних установок или добавления других аналогично тому, как файл **/etc/rc.conf** соотносится с **/etc/defaults/rc.conf**), устанавливая переменные, необходимые для сборки, и определяя дополнительные цели, используемые при автоматизированном процессе компиляции.

Инсталляция портов

Предположим, что вы собираетесь установить определенный порт. Все, что требуется, — это перейти в каталог порта командой **cd** и запустить команду **make**. Она скомпилирует программное обеспечение. Для инсталляции достаточно ввести **make install**.

На первый взгляд, это простой процесс, однако он подразумевает большое число действий, выполняемых за кулисами. В действительности команда **make** запускает последовательность **make**-целей (описанных в табл. 15.3), каждая из которых зависит от всех предыдущих. Любую из этих целей можно указать явно. В этом случае будут скомпилированы все цели до нее, а затем она сама.

Таблица 15.3 Цели make в Makefile-файле порта

Цель	*Действие*
fetch	Загружает исходный tar-архив с дистрибутивного сайта в каталог **/usr/ports/distfiles**
checksum	Проверяет подлинность архива с помощью MD5 **checksum**
extract	Распаковывает tar-архив в рабочий подкаталог
patch	Вносит в исходный код все исправления из каталога **files**
configure	Запускает сценарий конфигурирования, который подготавливает исходный код к сборке
build	Компилирует программный продукт

Таким образом, команда **make extract** загрузит файл с исходным кодом, проверит его контрольную сумму MD5 **checksum** и распакует его. Хотя существуют и другие цели (перечисленные в файле **/usr/ports/Mk/bsd.port.mk**), в большинстве случаев пользоваться ими не придется.

После завершения каждого этапа процесса (за исключением **fetch** и **checksum**) в рабочем подкаталоге создается файл **.extract_done**. Таким способом система следит за выполнением процесса сборки. Чтобы проверить, что этап **fetch** завершен, система просто ищет файл порта в каталоге **/usr/ports/distfiles**. Этап **extract** явно запускает

проверку файла **checksum**. Последующие этапы следуют один за другим и являются независимыми, а проверка выполнения предыдущих шагов заключается в поиске соответствующих им **.*_done**-файлов. Затем завершить процесс командой **make**.

Зависимости в портах обрабатываются так же автоматически, как и в пакетах. Они читаются из файла **Makefile** в фазе **fetch**, а затем загружаются, собираются и устанавливаются. Всякий раз, когда процесс обнаруживает зависимость (установленную или нет), он отображает это в своем выводе.

После инсталляции порта запись о нем вносится в базу данных **/var/db/pkg**. Теперь он ничем не отличается от пакета. Для сбора информации о нем и сравнения версий можно пользоваться утилитами **pkg_***, как если бы порт был установлен с помощью **sysinstall**.

Удаление инсталлированных портов

В портах также поддерживается цель **deinstall** (**make deinstall**). Этой целью можно воспользоваться лишь для той версии порта, которая указана в файле **Makefile**,т.е. происходит деинсталляция в точности того программного пакета, который был установлен. Если же порт обновлен более поздней версией пакета, команда **make deinstall** не сможет удалить его. Потребуется утилита **pkg_delete**.

Обновление порта

Если версия пакета изменилась, можно воспользоваться командой **make install** для возврата к его предыдущей версии. Однако этого лучше избегать, поскольку большинство установленных файлов в обеих версиях совпадают. Поэтому попытка удаления (с помощью **pkg_delete**) более ранней версии порта или пакета (после установки поверх него новой версии) приведет к удалению большинства файлов последней. Если вывод утилиты **pkg_version** похож на приведенный ниже, значит, в базе данных пакетов записана информация о более ранних версиях пакетов. Исправить ситуацию можно, лишь деинсталлировав все версии пакета и установив текущую версию:

```
apache-1.3.12                  <      needs updating (index has 1.3.19)
apache-1.3.14                  <      needs updating (index has 1.3.19)
apache-1.3.17                  <      needs updating (port has 1.3.19)
apache-1.3.19                  =      up-to-date with port
```

Чтобы избежать подобной ситуации, всегда проверяйте предыдущую версию порта до установки новой. А чтобы увидеть, что уже установлено в системе, используйте **pkg_info**, **pkg_version** или просто структуру каталога **/var/db/pkg**. Если существует ранняя версия, удалите ее:

```
# pkg_delete apache-1.3.12
```

Не нужно беспокоиться о **config**-файлах порта. В порте **apache**, например, файл **pkg-plist** содержит команды **@unexec**, которые сравнивают конфигурационные файлы порта с теми, что установлены, и удаляют последние лишь в случае их полного совпадения. Несмотря на это, до удаления пакета или порта желательно создать резервные копии всех важных **config**-файлов.

Впрочем, нет ничего страшного в том, чтобы оставить и старые версии портов. Это позволяет вести историю обновлений системы. Единственный недостаток: если

файлы порта изменятся, устаревшие файлы будут все равно присутствовать в системе, занимая дисковое пространство.

Как обновить локальное дерево портов?

Программное обеспечение изменяется. В наборе содержится более 4000 портов, причем каждый из них подвергается дальнейшей разработке (одни быстрее, другие медленнее). Фактически, системный администратор не способен инсталлировать или обновлять программное обеспечение с той скоростью, с которой оно разрабатывается. Лучшее, на что можно надеяться, — это поддержка набора портов на более или менее современном уровне. При этом такие средства, как **pkg_version**, позволяют следить за тем, что именно следует обновлять.

Для поддержки синхронизации портов наилучшим средством является **CVSup**. Подробнее оно описано в главе 18, но, поскольку оно является важной частью эффективной работы с портами, о его использовании кратко рассказано и здесь.

Прежде всего, необходимо установить пакет **cvsup-bin** — из пакетов или из портов. Фактически, предпочтительнее воспользоваться пакетом, поскольку **CVSup** — это средство, написанное на языке Modula-3. Если оно компилируется из исходного кода, потребуется компиляция нескольких больших приложений Modula-3. Чтобы пропустить эту процедуру, примените пакетную инсталляцию.

Убедитесь, что вы должным образом подключены к Internet. Затем перейдите в каталог порта **/usr/ports/net/cvsupit**. Это псевдопорт, поскольку на самом деле он ничего не устанавливает. Вместо этого он создает центральный **config**-файл, используемый **CVSup**. Запустите **make**, а затем **make install**. На экране будет отображено конфигурационное меню порта **cvsupit** (выбор ветви), показанное на рис. 15.4.

Рисунок 15.4
Меню выбора ветви исходного кода ***cvsupit****.*

Выберите пункт ".", а не имя ветви. Он обозначает HEAD — верхушку дерева исходного кода, где текущие версии портов хранятся на главном сервере. В остальных меню примите значения по умолчанию и выберите из списка сервер **CVSup** (обычно серверы с большими номерами оказываются менее загруженными). При запросе подтверждения на запуск обновления **CVSup** выберите **No**.

Откройте файл **/etc/cvsupfile** в текстовом редакторе. Закомментируйте строки, начинающиеся с **src-** (дескрипторы, описывающие исходные коды, которые не подлежат обновлению). Для этого поместите символ # в начале каждой строки.

Запустите обновление **CVSup** следующей командой (как указано в **cvsupit**):

```
# /usr/local/bin/cvsup -g -L 2 /etc/cvsupfile
```

Программа соединится с выбранным сервером **CVSup** и начнет синхронизацию набора портов. Она отображает все изменения между установленной и текущей версией. **CVSup** обновляет только те файлы, которые были изменены, причем только различающиеся части (используется утилита **diff**). Это позволяет обновить полный набор портов, пересылая минимальный объем данных. Обновление посредством **CVSup** можно эффективно произвести даже в сети с небольшой полосой пропускания. Подробнее об этом см. в главе 18.

По завершении синхронизации набор портов придет в современное состояние. С помощью команды **pkg_version -v** можно увидеть, какие порты или пакеты необходимо обновить.

Если у вас есть возможность, рекомендуется сделать синхронизацию портов ежедневной процедурой, чтобы порты не устаревали больше, чем на 24 часа. Для этого добавьте приведенную выше команду **CVSup** в **periodic**-файл (об этом рассказано в главе 14): в каталоге **/usr/local/etc** создайте каталог **periodic**, а в нем — каталог **daily**. Создайте файл с названием **100.cvsup-ports** и поместите в него следующие команды:

```
#!/bin/sh
/usr/local/bin/cvsup -g -L 2 /etc/cvsupfile
```

Теперь каждую ночь, когда запускается ежедневный сценарий **periodic**, набор портов будет автоматически синхронизироваться. Вывод **CVSup** будет высылаться вам по электронной почте.

Замечания о запрещенных портах

Иногда порт присутствует в наборе портов, однако система не позволяет собрать его. Этим управляет переменная **FORBIDDEN** в файле **Makefile**. Если эта переменная установлена каким-либо текстовым значением, оно будет выводиться при попытке собрать порт:

```
# cd /usr/ports/lang/perl5
# make
===>   perl-5.005 is forbidden: perl is in system.
```

Существует несколько причин, по которым порт может быть отмечен как "запрещенный". В данном примере программа является частью основной системы, поэтому ее сборка из портов будет излишней и может даже помешать нормальной работе системы. В других случаях причиной может быть дыра в защите, которая не была устранена к моменту обновления. Желательно синхронизировать порты с сервером **CVSup** как можно чаще.

Попытка собрать запрещенный порт редко приводит к хорошему результату, хотя это и можно сделать, удалив строку **FORBIDDEN** из **Makefile**. Некоторые запрещенные порты обеспечивают возможность провести сборку, отменив запрет путем установки *переменной, как, например, порт security/ssh:*

```
# cd /usr/ports/security/ssh
# make
===>   ssh-1.2.27_3 is forbidden: OpenSSH is a superior version of SSH
which has been included in the FreeBSD base system since 4.0-RELEASE.
```

```
This port is now deprecated and will be removed at some point in the
future. To override this warning set the REALLY_WANT_SSH environment
variable and rebuild.
```

Как сказано в инструкции, необходимо установить указанную переменную среды и снова попытаться провести сборку. Если используется **csh** или **tcsh**, выполните команду:

```
# setenv REALLY_WANT_SSH yes
```

В **bash** аналогичная команда имеет вид:

```
# REALLY_WANT_SSH=yes
```

После установки переменной и повторного запуска **make** компиляция порта проходит корректно.

Освобождение дискового пространства, занятого в процессе сборки портов

По окончании сборки и установки порта в его каталоге присутствует рабочий подкаталог, заполненный файлами с исходным кодом и скомпилированными объектными файлами. Они занимают некоторый объем дискового пространства, поэтому после установки рекомендуется возвратить порт в его исходное состояние, удалив все лишние файлы с помощью команды:

```
# make clean
```

Эта команда удаляет рабочий каталог и все файлы в нем для порта и всех его зависимостей. Она не удаляет лишь tar-архивы из каталога **/usr/ports/distfiles**, поэтому последние следует удалить вручную.

СОВЕТ

Если вы не испытываете проблем с дисковым пространством, рекомендуется сохранять **tar**-архивы в каталоге **/usr/ports/distfiles**. Это позволит системе работать с уже имеющимся файлом, а не загружать его заново, если вам потребуется заново собрать или установить порт.

Преимущество портов заключается также в том, что при исправлении ошибки в защите в каком-либо из портов разработчики часто предлагают не новый исходный код, а **patch**-файл, который можно поместить в каталог с имеющимся исходным кодом и скомпилировать его. В этом случае нет необходимости загружать новый **tar**-архив.

Периодически следует запускать команду **make clean** в самом верхнем каталоге набора портов. Файлы **Makefile** присутствуют на обоих уровнях: в **/usr/ports** и на уровне категории. Это позволяет, например, собрать все порты в категории одновременно (запустив команду **make**, например, в **/usr/ports/www**). Более того, таким образом можно удалить все оставшиеся после компиляции файлы во всех портах с помощью одного рекурсивного процесса. Для этого достаточно выполнить **make clean** в каталоге **/usr/ports**.

Как поступить, если сборка порта завершается неудачно

Набор портов, несмотря на удобство, несовершенен. В силу множества изменений, которым подвергается дерево исходного кода, особенно взаимосвязанные порты (например, связанные с GNOME и KDE), вы можете столкнуться с определенными проблемами. В общем случае, это выглядит так: процесс компиляции завершается

с сообщением "*** **Error code 1**" или чем-то подобным. Когда это происходит, можно попытаться предпринять несколько шагов.

Прежде всего, проверьте, что вы начали компиляции "чистого" порта. Если уже существует рабочий каталог, это значит, что вы вполне могли смешать старые и новые исходные файлы, а это зачастую приводит к невозможности сборки порта. Запустите **make clean**, чтобы вернуться к исходному состоянию порта.

Если это не помогает, посетите Web-сайт FreeBSD по адресу **http://www.freebsd.org** и следуйте ссылке Mailing Lists. Произведите поиск в соответствующих списках рассылки. В критерии поиска укажите список Ports, имя порта и текст, связанный с ошибкой (присутствующий в последних строках вывода компилятора). Если нечто подобное случалось и у других пользователей, вы наверняка найдете какую-либо полезную информацию.

Вам может помочь лицо, поддерживающее порт. Оно указано в файле **Makefile** в переменной **MAINTAINER**, где есть его адрес электронной почты. Обязательно пошлите ему вежливое сообщение, содержащее значащую часть вывода компилятора, а также информацию о системе (например, вывод команды **uname -a**). Тот, кто поддерживает порт, обычно очень занят, поэтому заранее поблагодарите за любую возможную помощь!

Комплекты обновления (Upgrade Kits)

Иногда порт не поддается сборке, сообщая, что версия системы "слишком ранняя". Это значит, что вам следует или обновить всю систему, или установить *комплект обновления* (upgrade kit). Последний вариант предпочтительнее. Несмотря на имя, он не обновляет саму систему и не вмешивается в ее работу.

Комплект обновления представляет собой набор более новых **include**-файлов, которые требуются набору портов для сборки определенных целей. Большинство **include**-файлов находится в каталоге **/usr/ports/Mk**, поэтому их обновление можно провести с помощью **CVSup**. Неизбежное препятствие состоит в том, что некоторые требуемые файлы не находятся внутри иерархии **/usr/ports**, а разбросаны по основной системе, например, в **/usr/share** или **/etc**. Комплект обновления представляет собой стандартный пакет FreeBSD, содержащий новые версии всех требуемых файлов, позволяя производить сборку портов даже тогда, когда дерево портов новее, чем основная система. Как и любой пакет, комплект обновления устанавливается с помощью команды **pkg_add**.

Комплекты обновления распространяются с **http://www.freebsd.org/ports**, причем всегда имеется возможность обновления от любого релиза до текущей версии в соответствующей ветви **-STABLE**. Можно установить последовательность комплектов обновления. Такая ситуация означает, что возникла необходимость обновления всей системы. Этот вопрос обсуждается в главе 18.

Web-сайт Fresh Ports

Вывод процесса синхронизации **CVSup** полезен, однако не похож на идеальный способ отслеживания измененных портов. Существует удобный Web-сайт (см. рис. 15.5), полезный всякому администратору FreeBSD, — Fresh Ports (**www.freshports.org**).

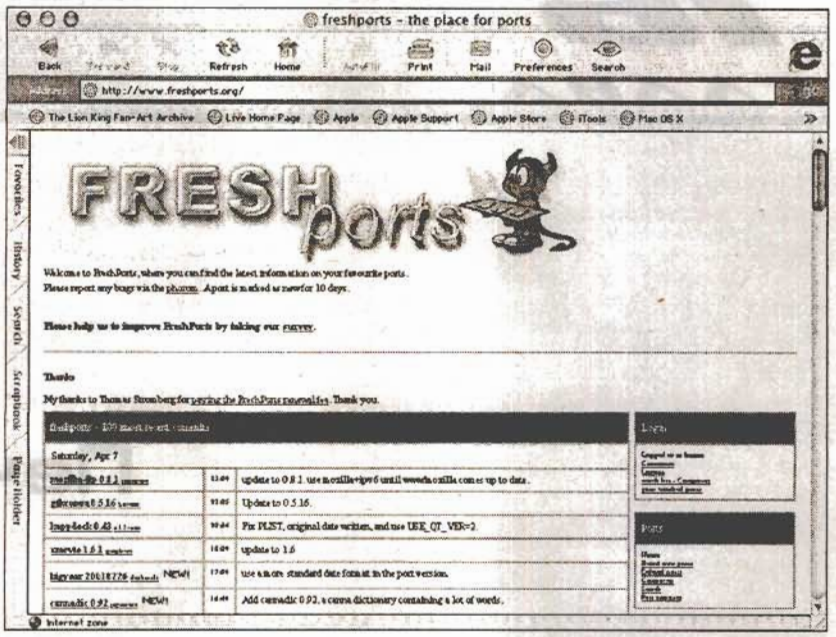


Рисунок 15.1 *Web-сайт Fresh Ports.*

Поддерживаемый независимо от основного проекта FreeBSD, Web-сайт Fresh Ports представляет собой базу данных, связанную с основным CVS-хранилищем набора портов. Здесь отслеживаются все изменения и вносятся в список в хронологическом порядке. На этом Web-сайте можно зарегистрироваться и настроить список требующихся вам портов. При каждом изменении вам будет высылаться электронное сообщение, описывающее точные изменения, произведенные в портах. Если изменилась версия порта или была исправлена серьезная дыра в защите, такой порт стоит обновить, если же была исправлена опечатка в сообщении, устанавливать заново такой порт вряд ли имеет смысла.

16 ГЛАВА

Печать

- Демон lpd
- Конфигурирование ядра, устройства и режима соединения
- Создание каталога спула
- Фильтры
- Фильтры преобразования
- Настройки в файле /etc/printcap
- Включение lpd
- Печать из командной строки
- Печать из X-Window
- Печать в StarOffice
- Проверка состояния заданий печати
- Удаление заданий из очереди
- Управление принтером
- Основы сетевой печати
- Решение проблем

Настройка принтера во FreeBSD — одна из самых сложных задач, с которыми вы столкнетесь при администрировании этой системы. Печать во FreeBSD — это нечто большее, чем просто установка драйвера, выбор принтера из Панели управления (Control Panel) и, собственно, печать. Процесс настройки печати во FreeBSD требует модификации одного или нескольких файлов и, возможно, установки фильтров, если будет печататься не только текст. Если принтер поддерживает язык PostScript, настройка несколько упростится. Если нет, принтер все равно можно будет использовать во FreeBSD. Последний случай требует лишь дополнительного конфигурирования, поскольку необходимо установить программное обеспечение, которое позволит посылать PostScript данные на не-PostScript принтер.

ПРИМЕЧАНИЕ

GDI-принтеры (иногда их называют Win-принтеры) не работают с FreeBSD. Они обычно дешевле, чем другие принтеры, поскольку в них часть функций управления принтером перенесена с оборудования на программный драйвер. Недостатком таких принтеров является несколько большее потребление ресурсов процессора, а также отсутствие драйверов (обычно разработчики поставляют драйверы только для Windows). Существует длинный список операционных систем, которые не работают с такими принтерами. В нем присутствует и FreeBSD.

Итак, далее предполагается, что принтер подключен к параллельному порту. Система FreeBSD позволяет воспользоваться и принтерами, подключенными к последовательному порту, однако такой способ печати является слишком медленным и устаревшим. Он используется на очень старых моделях принтеров. Предполагается также, что вам точно известно, поддерживает ваш принтер язык PostScript или нет. Узнайте об этом из документации.

Демон lpd

Демоном, обслуживающим принтер, является **lpd**. Это программное обеспечение выполняется в фоновом режиме и ожидает запросов на печать. При их получении оно пропускает их через фильтры, преобразует данные в другой формат (если нужно) и посылает их в очередь печати.

Очередь печати

Очередь печати (*print queue*) — это определенная область на жестком диске, где сохраняются данные, ожидающие вывода на принтер. Каждый принтер, присоединенный к системе, имеет свою отдельную очередь и область спула. Данные хранятся в спуле до тех пор, пока не будут выведены на принтер. Спул может содержать несколько заданий одного или разных пользователей.

Когда пользователь отправляет задание на печать, оно размещается в очереди. После этого спулер пересылает задание принтеру, как только тот освободится. Задания могут иметь различный приоритет, определяющий порядок их печати. У спулера есть несколько преимуществ по сравнению с прямым выводом на принтер (используемым, например, в DOS):

- Он дает возможность использовать принтер сразу нескольким пользователям, поскольку данные размещаются в очереди, и программа, переславшая их, может продолжать работу.

- Возможна печать в фоновом режиме. Опять-таки программа, пославшая запрос на печать, может переслать данные и продолжить работу. Это значит, что не нужно ждать окончания печати: программу можно даже закрыть, все равно задание, поставленное в очередь, будет выведено на принтер.
- Содержит определенный уровень защиты от сбоев. Если, к примеру, принтер будет заново инициализирован, задания в очереди не будут потеряны, и их не нужно будет пересылать снова. После того, как принтер перейдет в нормальный режим, оставшиеся задания из очереди будут выведены на печать автоматически.

Спул, обычно, расположен в каталоге **/var/spool**. Спул первого принтера в системе, как правило, находится в каталоге **/var/spool/lpd** или **/var/spool/output/lpd**. Эти настройки можно изменить в соответствующем конфигурационном файле. Об этом будет рассказано далее.

Конфигурирование ядра, устройства и режима соединения

В ядре по умолчанию включена поддержка параллельного устройства, поэтому, если вы не собирали специальное ядро без нее, у вас не должно быть никаких проблем. Проще всего проверить поддержку параллельного устройства в выводе команды **dmesg**. Для этого используется следующий синтаксис: **dmesg | grep lpt0**. Он проверяет поддержку первого параллельного устройства в ядре (в DOS это устройство называется **lpt1**, а устройства FreeBSD почти всегда нумеруются с 0). Если вывод команды примерно такой:

```
lpt0:  <Printer>  on  ppbus0
lpt0:  Interrupt-driven  port
```

это значит, что параллельный порт (и принтер) поддерживается. Если такая команда ничего не выводит, то в ядре FreeBSD отсутствует поддержка параллельного порта. В конфигурационном файле ядра необходимо добавить строку **device lpt** и заново собрать ядро. Кроме того, в конфигурации должна присутствовать и строка **device ppbus**. О том, как правильно сконфигурировать и собрать ядро, рассказано в главе 17.

Конфигурирование режима параллельного порта

Режим работы параллельного порта можно настроить с помощью программы **lptcontrol**. Для ее корректной работы необходимо, чтобы к параллельному порту был подключен принтер, а пользователь, запустивший ее, обладал правами **root**.

Синтаксис команды: **lptcontrol -x -d /dev/lpt0**, где **-x** — одна из следующих опций, определяющая устанавливаемый режим:

Таблица 16.1 Опции режима программы lptcontrol

Опция	*Описание*
-i	Режим, управляемый прерыванием
-p	Режим запроса (polled mode)
-e	Расширенный режим (поддерживаемый принтером)
-s	Стандартный режим (отключение расширенного режима)

Если принтер и BIOS поддерживают ECP, EPP или другие расширенные формы соединения, следует воспользоваться опцией **-e**.

Опция **-d** не является обязательной. Если устройство не указано, **lptcontrol** обращается к **/dev/lpt0** — устройству по умолчанию.

Вот пример команды, устанавливающей расширенный режим для первого принтера:

```
lptcontol -e -d /dev/lpt0
```

Чтобы эти настройки устанавливались автоматически при загрузке системы, необходимо добавить команду **lptcontrol** в один из начальных сценариев: или в файл **/etc/rc.local**, или в созданный для этого файл в каталоге **/usr/local/etc/rc.d**. В последнем случае необходимо установить права файла на выполнение. (Например, командой **chmod 655 имя_файла**.)

По завершении этой процедуры можно перейти к настройке демона **lpd** и спулера.

Создание каталога спула

Первое, что необходимо сделать, — это создать для принтера каталог спула. Обычно он располагается в **/var/spool/lpd**, хотя его можно поместить и в любой другой точке файловой системы.

Выберите имя для конфигурируемого принтера, оно может быть любым. Кроме того, для доступа к принтеру можно создать несколько псевдонимов.

После того, как выбрано имя принтера, создайте каталог спула для него. Например:

```
mkdir   /var/spool/lpt/laserjet
```

Затем необходимо изменить владельца каталога и права доступа к нему, чтобы пользователи не могли просматривать чужие задания на печать. Всеми каталогами спула должен владеть пользователь **daemon** и группа **daemon**. Для них должны быть установлены права на чтение, запись и выполнение. Остальным пользователям не предоставляется никаких прав доступа. Для этого используются две следующих команды:

```
chown   daemon.daemon   /var/spool/lpd/laserjet
chmod   770   /var/spool/lpd/laserjet
```

После настройки каталога спула необходимо настроить текстовый фильтр **lpd**.

Фильтры

Фильтры выполняют большую часть работы при печати. Когда **lpd** пересылает данные фильтру, он подает фильтру печатаемый файл на STDIN, а поток STDOUT — на принтер.

Текстовые фильтры

Как следует из названия, *текстовый фильтр* (*text filter*) используется демоном **lpd** при печати обычного текста. Текстовый фильтр может быть простым сценарием командного интерпретатора, выводящим неформатированные данные на принтер посредством команды **cat**, или сложной программой, переводящей данные в другой формат. Примером сложного фильтра может служить тот, что используется в программе GhostScript. На входе он получает данные в формате PostScript и преобразует их к виду, понятному не-PostScript-принтерам.

Простейший фильтр **lpd** выглядит следующим образом:

```
#!/bin/sh
/bin/cat  &&  exit  0
exit  2
```

Как правило, фильтры хранятся в каталоге **/usr/local/libexec**. Этот фильтр можно назвать, например, **/usr/local/libexec/if-text**. После сохранения файла его необходимо сделать выполняемым, чтобы **lpd** смог запустить его. Для этого достаточно воспользоваться командой **chmod 555 /usr/local/libexec/if-text**.

Первая строка фильтра показывает, что он является сценарием интерпретатора **/bin/sh** (стандартный командный интерпретатор).

СОВЕТ

Если что-либо связанное со сценариями интерпретатора вам непонятно, обратитесь к главе 13, прежде чем продолжать чтение. Настройка фильтров печати зачастую требует навыков программирования на языке командного интерпретатора.

Во второй строке вызывается программа **cat**, которая по умолчанию читает данные из потока STDIN и посылает их на STDOUT. Оператор **&&** указывает, что необходимо выполнить или оба оператора, или ни одного. Другими словами, если команда **cat** завершается успешно, сценарий заканчивает работу с кодом завершения 0, указывающим на успех. Если возникает ошибка, часть строки после **&&** не выполняется, а управление передается третьей строке, которая завершает сценарий с кодом 2. (Если **lpd** получает код завершения 2, это значит, что ошибка связана с фильтром, поэтому демон не пытается распечатать файл снова.)

Этот простой фильтр годится для печати обычного текста на большинстве принтеров, не поддерживающих языка PostScript. В противоположность последним, PostScript-принтеры не способны обрабатывать текстовые данные. Такому принтеру требуется более сложный фильтр, преобразующий обычный текст в формат PostScript.

ЧТО ТАКОЕ POSTSCRIPT?

PostScript представляет собой сложный язык программирования, разработанный для печати и форматирования графики и текста. Он не зависит от устройства в том смысле, что любой принтер, поддерживающий язык PostScript, способен печатать документ PostScript без каких-либо специальных драйверов. Этот язык был разработан компанией Adobe в 1985 году, и на сегодняшний день его поддерживает большинство принтеров.

К сожалению, принтеры, не поддерживающие язык PostScript, не печатают документы в этом формате. Они выводят его как обычный текст, представляющий собой лишь набор операторов, а не требуемое изображение или текст. Для перевода документов PostScript в формат, понятный обычным принтерам, существует свободно распространяемая программа GhostScript. Она позволяет программно эмулировать PostScript. Об этой программе рассказано далее в этой главе.

Поэтому вам обязательно нужно будет установить программу, преобразующую обычный текст в PostScript. Она называется **a2ps** и содержится в дереве портов FreeBSD. (О том, как инсталлировать порты, см. главу 15.)

Фильтр для PostScript-принтера должен уметь самостоятельно проверять, какие данные он получает. Если они имеют формат PostScript, фильтр не обрабатывает их, а в первозданном виде пересылает принтеру. (Файл PostScript представляет собой обычный текстовый файл, содержащий команды, понятные PostScript-принтеру.) В противном случае фильтр обрабатывает данные, преобразуя их в формат PostScript, а потом пересылает принтеру.

Все файлы в формате PostScript начинаются с последовательности "**%!**". Сценарий должен проверить два первых символа первой строки на совпадение с "**%!**". Если это так, файл принадлежит формату PostScript и его нужно сразу переслать принтеру. Если нет, это обычный текстовый файл, который с помощью программы **a2ps** необходимо преобразовать в PostScript.

Для этого используется следующий сценарий:

Листинг 16.1 Пример фильтра PostScript

```
#!/bin/sh
# Простой фильтр для PostScript-принтеров

read header
ps_test=`expr "$header" : '\(..\)'`
if [ "$header" = "%!" ]
then
      # Это PostScript-файл, его нужно переслать без изменений.
      Echo "$header" && cat && printf "\004" && exit 0
      exit 2
else
      # Файл является обычным текстом. Его необходимо преобразовать.
      (echo "$header"; cat) | /usr/local/bin/a2ps && printf "\004" &&
exit 0
      exit 2
fi
```

Файл можно сохранить (например, как **/usr/local/libexec/if-ps**) и сделать выполняемым с помощью команды **chmod 555 /usr/local/libexec/if-ps**, чтобы **lpd** мог запустить его.

Сценарий считывает первую строку ввода, пересылаемого ему демоном **lpd**, и сохраняет ее в переменной **header**. Затем с помощью команды **expr** программа читает два первых символа файла и сохраняет их в переменной **ps_test**. Затем оператор **if** проверяет значение двух этих символов, хранимых в **ps_test**. Если они равны **%!** (т.е. файл в формате PostScript), выполняется оператор **then**, в котором необработанные данные пересылаются принтеру. Вначале посылается первая строка, а затем (с помощью команды **cat**) весь остальной ввод. После этого оператор **printf** выводит специальный символ **\004**, и работа программы завершается с кодом 0. Оператор **&&** соединяет несколько выражений и имеет следующий смысл: "Выполнить или все команды, или ни одной". Если по каким-либо причинам одна из команд не завершится успешно, все оставшиеся команды не будут выполнены. Таким образом, программа завершит работу со статусом 2. Последний информирует **lpd** о том, что данные не были распечатаны, но предпринимать второй попытки не нужно.

Если два первых символа не равны **%!**, файл не является файлом PostScript, а значит, выполняется оператор **else**. В этом случае программа выводит первую строку и весь остальной ввод (командой **cat**) программе **/usr/local/bin/a2ps**. Программа **a2ps** преобразует файл в формат PostScript и пересылает его принтеру.

И наконец, после данных принтеру посылается специальный символ **\004** (делает это команда **printf**), после чего программа завершает работу с кодом 0. Как и в предыдущем случае (операторе **then**), если какая-либо из команд не завершается успешно, программа возвращает код 2.

Программа **a2ps** имеет множество опций, поэтому, если вы часто печатает текстовые файлы на PostScript-принтере, ознакомьтесь с ее справочным руководством. Вызов **a2ps** можно модифицировать под свои нужды.

Печать файлов в формате PostScript на не-PostScript-принтерах

Если вы используете не-PostScript-принтер, то сталкиваетесь с противоположной проблемой. Обычный текст необходимо передавать на принтер без изменений, но данные в формате PostScript нужно преобразовывать в формат, понятный принтеру.

В любом случае, PostScript часто используется как конечный формат в различных UNIX и во FreeBSD, когда дело доходит до печати более сложных форматов, чем обычный текст. Почти все программы обработки текстов, растровой и векторной графики способны выводить данные в PostScript-файл (или пересылать их непосредственно принтеру). Хотя некоторые приложения (например, StarOffice) могут включать драйвер для вашего принтера, для большинства программ UNIX это не так. Для них нужно просто выводить данные в формат PostScript. Таким образом, если вы не сможете обрабатывать PostScript, ваши возможности печати будут сильно ограничены.

К счастью, компания Aladdin разработала свободно распространяемую программу GhostScript, эмулирующую работу PostScript-принтера. Она получает данные в формате PostScript и преобразует их к форме, понятной не-PostScript-принтеру.

ПРИМЕЧАНИЕ

GhostScript поддерживает большое число принтеров. Их список находится на **http://www.cs.wisc.edu/~ghost/doc/printer.htm**.

GhostScript содержится в наборе портов FreeBSD в категории **print**. Об инсталляции портов рассказано в главе 15.

Здесь понадобится сценарий наподобие того, что приведен в предыдущем разделе. Однако его действия будут совершенно противоположными: если данные пересылаются не в формате PostScript, их необходимо вывести прямо на принтер, в ином случае их следует обработать программой GhostScript для преобразования в формат, понятный не-PostScript-принтерам. Ниже приведен сценарий (взят из FreeBSD Handbook), который обеспечивает печать файлов PostScript на принтере HP Deskjet 500, не поддерживающем язык PostScript.

Листинг 16.2 Пример фильтра GhostScript для принтера HP DeskJet 500

```
#!/bin/sh
  #
  #    ifhp - Печать PostScript-данных (эмулируемых Ghostscript) на
  #    принтере DeskJet 500
  #    Установлен в каталоге    /usr/local/libexec/hpif
  #
  #    Рассматривать символ LF как CR+LF:
  #
  printf "\033&k2G" || exit 2
  #
  #    Прочесть два первых символа файла
  #
  read first_line
  first_two_chars=`expr "$first_line" : '\(..\)'`

  if [ "$first_two_chars" = "%!" ]; then
      #
      #    Если это PostScript, использовать Ghostscript для
      #    преобразования и печати.
```

```
        #
        #    Обратите внимание, что файлы PostScript представляют собой
        #    интерпретируемые программы, которые могут выводить данные в
        #    поток stdout, смешиваясь с тем, что выводится на принтер.
        #    Поэтому поток stdout перенаправлен в stderr, а дескриптор
        #    потока 3 — в stdout. В него и выводит данные
        #    Ghostscript. Упражнение для внимательного читателя:
        #    перехватить вывод Ghostscript в stderr и переслать его по
        #    электронной почте пользователю, запустившему задание.
        #
        exec  3>&1  1>&2
        /usr/local/bin/gs  -dSAFER  -dNOPAUSE  -q  -sDEVICE=djet500  \
                  -sOutputFile=/dev/fd/3  -  &&  exit  0

        #
        /usr/local/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=djet500 \
                                -sOutputFile=-  -  && exit 0
    else
        #
        #    Обычный текст, или HP/PCL, который необходимо выводить
        #    напрямую. По окончании вывода печатается специальный
        #    символ, извлекающий последнюю страницу.
        #
        echo  $first_line  &&  cat  &&  printf  "\033&l0H"  &&
    exit  0
    fi

    exit  2
```

Сценарий может показаться сложным, но это не так. Фактически, сценарий читает первую строку ввода, извлекает из нее два первых символа и проверяет, равны ли они %!. Если да, вводом является PostScript и запускается оператор **then**, т.е. вывод пересылается программе GhostScript. Если нет, файл рассматривается как обычный текст, который пересылается непосредственно принтеру.

ПРИМЕЧАНИЕ

GhostScript представляет собой очень мощную программу с большим числом опций. Ее документацию можно найти по адресу **http://www.cs.wisc.edu/~ghost/index.html**.

Фильтр печати lpf

ПРИМЕЧАНИЕ

FreeBSD содержит программу под названием **lpf**, действующую как фильтр печати. Она имеет массу возможностей, включая ведение учета (сколько страниц напечатано, кем именно и т.д.). Информация об этой программе содержится в справочном руководстве **man cap**.

Фильтры преобразования

Фильтры преобразования (conversion filters) подобны текстовым фильтрам за исключением того, что они предназначены для преобразования различных форматов файлов в формат, понятный принтеру. Они позволяют печатать различные типы файлов непосредственно из командной строки с помощью **lpr**, исключая необходимость их открытия в соответствующих программах или преобразования вручную. Применяемый фильтр указывается в командной строке как опция команды **lpr** (это команда, используемая для печати).

В табл. 16.2 перечислены опции преобразования, поддерживаемые командой **lpr**:

Таблица 16.2 Опции преобразования команды lpr

Опция	*Описание*
-d	Файл в формате DVI (используется системой TeX.)
-f	Фильтр для печати файлов с исходным кодом на языке FORTRAN.
-c	Файл содержит данные программы **cifplot**. Последняя не включена в состав FreeBSD.
-g	Файл содержит данные, полученные с помощью процедур вычерчивания (plot routines) UNIX. Эти процедуры включены в FreeBSD.
-d	Файл содержит данные программы **troff**, независящей от устройства. Они не поддерживаются никаким программным обеспечением, включенным в FreeBSD.
-t	Файл содержит команды C/A/T для устаревших версий программы **troff**.
-v	Файл содержит растровое изображение (для различных устройств печати изображений). Большинство пользователей не применяет этот формат.

По умолчанию эти фильтры не устанавливаются. Кроме того, название опции и вызываемый фильтр не зашиты жестко в систему, поэтому если фильтр **cifplot** не используется, опцию **-c** можно назначить какому-либо другому фильтру.

Как и текстовые фильтры, фильтры преобразований могут быть сценариями интерпретатора, вызывающими стандартные программы UNIX, или самостоятельными выполняемыми программами, поставляемыми сторонними производителями программного обеспечения.

Ниже приведен пример простейшего фильтра, преобразующего данные в формате **groff** (который не поддерживается принтером) в PostScript (который понятен принтеру):

```
#!/bin/sh
exec  grops
```

Из страницы справочного руководства **grops** можно почерпнуть сведения о том, что эта программа преобразует вывод из формата **groff** в PostScript. В данном случае данные до вывода на принтер просто подвергаются обработке посредством **grops**.

Фильтр необходимо сохранить в каталоге **/usr/local/libexec** в файле с именем **g2ps**, отражающим преобразование **groff** в PostScript. Если этот фильтр назначить опции **-t** команды **lpr**, то напечатать исходный файл в формате **groff** можно будет прямо из командной строки, воспользовавшись командой:

```
lpr  -t  myfile
```

где **myfile** — имя файла, выводимого на печать.

Далее в этой главе мы рассмотрим, как сконфигурировать систему печати с использованием этого фильтра в файле **/etc/printcap** (этот файл управляет поведением всей системы печати).

Настройки в файле /etc/printcap

Файл **/etc/printcap** представляет собой "клей", соединяющий все, о чем было рассказано ранее, в единую систему. В нем определены: имя и псевдонимы принтера, используемые фильтры преобразования, опции доступа к принтеру, местонахождение спула и многое другое.

Файл **/etc/printcap** изначально присутствует в системе, однако все опции в нем закомментированы. Поэтому при настройке комментарии с некоторых опций следует убрать, если потребуется, что-то изменить в них и/или добавить новые.

Формат файла **/etc/printcap** достаточно прост. Вот пример записи из него:

```
simba|lp|local line printer:\
#       :sh:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/simba:lf=/var/log/lpd-errs:
    :if=/usr/libexec/if-ps:
```

В первой строке задано имя принтера, за которым следуют псевдонимы. В данном случае принтер называется **simba**, псевдонимом является **lp** (это означает, что данный принтер используется по умолчанию), затем следует длинное описание принтера.

Вторая строка закомментирована. Если комментарий убрать, перед печатью данных будет печататься заглавная страница с именем пользователя, именем файла и т.д. Этим стоит воспользоваться, если на одном принтере печатают документы несколько пользователей — титульные страницы облегчат им поиск своих распечаток.

В третьей строке задано местонахождение принтера. **:lp** означает local printer — локальный принтер (в противоположность удаленному). Принтер присоединен к первому параллельному порту **/dev/lpt0** (LPT1 в DOS). **:sd** указывает на используемый принтером каталог спула — **/var/spool/lpd/simba** (обратитесь к разделу в этой главе, посвященному созданию и конфигурированию каталога спула). **:lf** означает log file, т.е. файл, куда заносятся ошибки, возникающие при печати.

В четвертой строке задан входной фильтр, используемый принтером. В данном случае это фильтр **if-ps** (который был создан ранее для преобразования текста в формат PostScript).

Установка фильтров преобразования

В этом же файле настраиваются и фильтры преобразования. Для этого нужно добавить соответствующую опцию (они перечислены ниже), за которой следует имя фильтра. Формат совпадает с форматом приведенной выше строки для входного фильтра **:if**. В табл. 16.3 перечислены опции различных фильтров:

Таблица 16.3 Опции фильтров в файле /etc/printcap

Фильтр	*Опция в файле /etc/printcap*
DVI	:df
FORTRAN	:rf
cifplot	:cf
plot	:gf
dittroff	:nf
troff	:rf
raster	:if

Помните, что эти фильтры не заданы жестко. Если вам не требуется фильтр для исходных кодов на Фортране, вы можете использовать эту опцию для любого другого фильтра. Вот пример того, как можно установить фильтр, преобразующий **groff** в PostScript (созданный ранее):

```
:rf=/usr/local/libexec/g2ps:
```

Обратите внимание, что после добавления этой строки нужно внести некоторые изменения в предшествующую ей строку **:if**. Необходимо добавить символ обратной косой черты, который экранирует перевод строки, поскольку **lpd** ожидает, что вся входная информация в этом файле расположена в одной строке.

Включение lpd

После того, как файл **/etc/printcap** сконфигурирован нужным образом, можно запустить демон **lpd**. Имея права пользователя **root**, его можно запустить командой **lpd**. Чтобы он запускался автоматически при каждой загрузке системы, в файл **/etc/rc,conf** необходимо добавить следующую строку:

```
lpd_enable="YES"
```

Теперь вы сможете печатать из командной строки, следуя инструкциям, представленным в следующем разделе. Если печать не работает корректно, обратитесь к разделу, посвященному решению проблем в конце этой главы.

Печать из командной строки

Команда **lpr** предназначена для посылки файла спулеру принтера. В простейшем случае она имеет вид

```
lpr file_name
```

где **file_name** — имя файла, выводимого на печать. Если необходимо напечатать несколько файлов, их можно указать в одной командной строке.

Принтер, на который посылаются данные, можно указать с помощью необязательной опции **-P**. Если она опущена, **lpr** использует принтер по умолчанию. Принтер по умолчанию определяется на основании перечисленных ниже условий. Первое условие, которое выполняется, и задает принтер.

- Если установлена переменная среды PRINTER, использует ее значение как принтер по умолчанию.
- Если установлена переменная среды LPDEST, **lpr** использует ее значение как принтер по умолчанию.
- Используется имя принтера **lp**, которое является псевдонимом одного из принтеров в файле **/etc/printcap**.
- Если значения переменных среды PRINTER и LPDEST не установлены и в файле **/etc/printcap** не содержится принтера с псевдонимом **lp**, запрос на печать не будет завершен успешно.

Команда **lpr** имеет и несколько других опций. Наиболее полезные из них перечислены в табл. 16.4:

Таблица 16.4 Опции команды lpr

Опция	*Описание*
-l	Использовать фильтр, который печатает коды управляющих символов и игнорирует переход на новую страницу.
-p	Вывод форматируется с помощью **pr**. Программа **pr** форматирует страницы по 66 строк на каждой с заголовком вверху, содержащим дату и время создания файла, номер страницы, а также пять пустых строк внизу.

Опция	Описание
-P	Посылает задание на указанный принтер.
-h	Отключает печать титульной страницы. Не дает никакого эффекта, если печать таких страниц отключена по умолчанию.
-m	Посылает электронное сообщение с уведомлением о завершении задания печати.
-r	Удаляет файл по завершении спулинга. Этой опцией не следует пользоваться, поскольку она удалит файл до того, как он будет успешно напечатан. В сочетании с опцией **-s** (описана в следующем пункте), она не удалит файл до завершения печати.
-s	Использовать символические ссылки. Вместо копирования файлов в каталог спула, эта опция позволяет создавать символические ссылки на существующие файлы. Она позволяет печатать файлы, слишком большие по размеру, чтобы разместить их в каталоге спула. При использовании этой опции следите за тем, чтобы файл не был изменен или удален до окончания печати.
-#n	Здесь **n** — число копий каждого файла.
-J job	Здесь **job** — имя задания, печатаемое на титульной странице. По умолчанию это имя первого указанного файла.
-T	Название, используемое программой **pr** как заголовок вверху страницы. По умолчанию это имя файла. Эта опция применяется только вместе с **-p**.
-i n	Здесь **n** — число столбцов, на которое выравнивается печатаемый вывод.
-w n	В данном случае **n** — ширина страницы в колонках. Применяется только вместе с опцией **-p**.

Печать из X-Window

Различные приложения X-Window используют разные методы печати. Здесь рассматривается пример браузера Netscape, предоставляющего возможность распечатать Web-страницу.

Для печати из Netscape необходимо выбрать меню File, а затем Print. На экране появится диалоговое окно Print, приведенное на рис. 16.1.

В разделе Print To: можно выбрать, куда следует выводить печатаемые данные — на принтер или в файл. Если выбран Printer, строка Print Command: ожидает ввода. Заметьте, что по умолчанию указана команда **lpr**, которой осуществляется печать из командной строки. При необходимости здесь можно указать дополнительные опции **lpr** (например, отправка электронного сообщения по окончании печати или распечатка нескольких копий). Подробные сведения приведены в разделе, посвященном **lpr** ранее в этой главе.

Большинство программ X-Window способно выводить информацию в файл. В этом случае создается файл в формате PostScript, который можно переслать прямо на PostScript-принтер или обработать с помощью GhostScript для печати на не-PostScript-принтере.

Netscape выводит данные в PostScript независимо от того, куда направлен вывод — команде **lpr** или в файл. Как было отмечено в разделе о конфигурировании, такая методика широко распространена. Как и большинство программ UNIX, Netscape не имеет драйверов принтера. Она просто выводит информацию в формате PostScript.

Рисунок 16.1
Диалоговое окно Print в Netscape. Обратите внимание, что при нажатии кнопки File в верхней части окна, по умолчанию будет задано имя ***netscape.ps****., поскольку Netscape выводит данные на печать в формате PostScript.*

Печать в StarOffice

Если в системе установлен пакет StarOffice, сконфигурировать печать в нем можно, дважды щелкнув по пиктограмме Printer Setup на рабочем столе. Диалоговое окно, которое появится на экране, показано на рис. 16.2.

Поскольку пакет StarOffice включает большое количество драйверов различных принтеров, нет необходимости использовать систему GhostScript для интерпретации PostScript-файлов. Достаточно просто выбрать нужный принтер из списка, щелкнуть на кнопке Add new printer, и он будет установлен как принтер по умолчанию. Для настройки принтера щелкните на кнопке Configuration.

Если принтер не установлен, StarOffice использует очередь Generic Printerin, при этом вывод в формате PostScript пересылается программе **lpr**.

Даже если в системе установлен GhostScript, предпочтительнее использовать драйвер (если он включен в пакет StarOffice), так как последний лучше поддерживает различные свойства принтера и доступ к нему возможен прямоиз приложений StarOffice, тем самым отпадает необходимость применять фильтры и GhostScript.

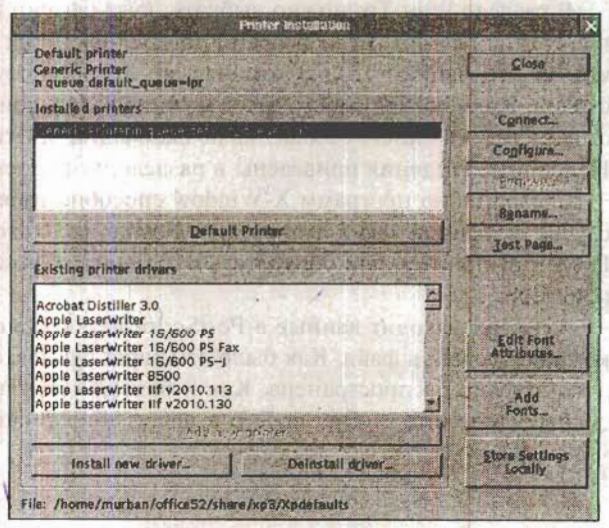


Рисунок 16.2
Диалоговое окно Printer Configuration в пакете StarOffice. Обратите внимание, что в пакет включены драйверы для многих моделей принтеров (в нижней части окна).

Проверка состояния заданий печати

Для проверки состояния заданий, находящихся в очереди, используется команда **lpq**. Вызываемая без аргументов, она сообщает состояние всех текущих заданий в очереди принтера по умолчанию. Поиск принтера по умолчанию происходит в следующем порядке: если установлена переменная среды PRINTER, ее значение рассматривается как принтер по умолчанию. Если она не установлена, принтером по умолчанию является принтер, имеющий псевдоним **lp** в файле **/etc/printcap**.

Аргументом **lpq** может быть номер задания, позволяющий получить информацию только о нем, или имя пользователя, тогда выводится информация только о его заданиях. Вывод **lpq** выглядит следующим образом:

```
bash$ lpq -P simba
simba is ready and printing
Rank    Owner Job  Files                          Total  Size
active  mike  5    /home/murban/sample.txt        2000   bytes
2nd     mike  6    /home/murban/sample1.txt       2500   bytes
3rd     jack  7    /home/jack/myfile.txt          3200   bytes
4th     jack  8    ...                            5500   bytes
```

Опция **-P** позволяет указать принтер. В данном случае **lpq** выводит состояние заданий очереди принтера **simba**, а не принтера по умолчанию. Все опции команды **lpq** объясняются далее.

Назначение информации, которую выводит **lpq**, вполне очевидно.

Первая строка сообщает о том, чем принтер занят в текущий момент. В данном случае он печатает. Если принтер остановился, в нем закончилась или помялась бумага, информация об этом будет отображена здесь же.

В очереди находится четыре задания: одно из них активно, а остальные пронумерованы в порядке их последующего вывода на печать. Обратите внимание на следующее. Столбец 3, **job**, содержит идентификатор задания, присваиваемый каждому из них. Если необходимо отменить печать (об этом далее), используется именно этот идентификатор. В столбце 4 указан печатаемый файл или файлы. Обратите внимание на четвертое задание, где вместо имени файла выведено троеточие. Это значит, что полный путь к файлу просто не поместился в отведенном для него столбце.

Еще несколько опций **lpq** перечислены в табл. 16.5.

Таблица 16.5 Опции программы lpq

Опция	*Описание*
-P name	**lpq** выводит информацию о состоянии очереди принтера с именем **name**, а не принтера по умолчанию.
-l	Эта опция заставляет **lpq** вывести подробную информацию о каждом файле в очереди, даже если она не помещается в одной строке. (В предыдущем примере вместо троеточия был бы выведен полный путь к файлу.)
-a	Выводит состояние локальных очередей всех принтеров.

Удаление заданий из очереди

Команда **lprm** используется для удаления заданий из очереди, а иногда и текущих печатаемых заданий. Если она запущена без аргументов, она удаляет текущее задание,

печатаемое на принтере по умолчанию и принадлежащее данному пользователю. Если текущее задание не принадлежит пользователю, запустившему **lprm**, команда не предпринимает никаких действий. Если команду запускает **root**, она удаляет текущее задание независимо от прав владения.

Запуск **lprm** с идентификатором задания в качестве аргумента удаляет это задание из очереди принтера по умолчанию (подразумевается, что это задание принадлежит пользователю). Обычные пользователи могут удалять только свои задания, **root** — любые. Например:

```
bash$ lprm 5
dfA001simba.samplenet.org   dequeued
cfA001simba.samplenet.org   dequeued
bash $
```

Первый файл, удаленный из очереди в примере, представляет собой непосредственно файл данных (точнее, его копию в каталоге спула), а второй — управляющий файл.

Отметьте, что системе требуется некоторое время после запуска **lprm** (до нескольких секунд). Может показаться, что утилита **lprm** "зависла", но это не так. Через несколько секунд система вновь выдаст приглашение командной строки. При попытке удаления задания, не принадлежащего пользователю, **lprm** выведет сообщение **Permission denied** (В доступе отказано).

lprm поддерживает несколько опций, изменяющих его поведение. Они перечислены в табл. 16.6.

Таблица 16.6 Опции команды lprm

Опции	*Описание*
-P name	Опция указывает имя принтера, из очереди которого следует удалить задание. Если опция не задана, принтером по умолчанию является значение переменной среды PRINTER или принтер с псевдонимом **lp**.
-	Символ дефиса указывает на удаление из очереди всех заданий, принадлежащих текущему пользователю. Если используется пользователем **root**, то удаляет все задания.
user_name	Удаляет все задания указанного пользователя. Этой опцией может воспользоваться только пользователь **root**, так как обычные пользователи могут удалять только свои задания.

ПРЕДУПРЕЖДЕНИЕ

Удаление текущего печатаемого задания не заставит принтер немедленно остановить печать. В зависимости от объема его оперативной памяти, в буфере принтера может находиться несколько страниц данных. А это значит, что часть документа (или даже весь документ небольшого размера) будет распечатана уже после удаления задания из очереди.

ПРЕДУПРЕЖДЕНИЕ

Удалить задания принтера можно лишь из той системы, откуда они были посланы (если к *принтеру обращается несколько машин в сети*).

Управление принтером

Принтерами и соответствующими им очередями управляет команда **lpc**. Системные администраторы пользуются ее для администрирования принтеров. Ограниченная часть ее функциональности доступна и обычным пользователям, включая вывод состояния очередей и перезапуск демона принтера, если он "завис". Кроме того, **lpc** включает и отключает принтеры и очереди, изменяет порядок заданий в очереди (файлы из хвоста очереди могут быть напечатаны первыми) и проверяет ее состояние.

Если программа **lpc** запущена без аргументов, она переходит в интерактивный режим и выводит приглашение. С аргументами **lpc** запускается, выполняет указанные действия и завершает работу. Вначале мы рассмотрим интерактивный режим работы **lpc**.

Программа lpc в интерактивном режиме

Запущенная без аргументов, команда **lpc** переходит в интерактивный режим и выводит следующее приглашение:

```
lpc>
```

Получить список поддерживаемых команд позволяет команда **?** или **help** (как всегда, все команды завершаются нажатием клавиши Enter). **help** с последующим именем команды выводит краткое (в одну строку) описание команды.

При вводе команд допускаются сокращения (наиболее короткие, но не приводящие к неоднозначности). Например, команду **status** можно сократить до **stat**, но не **sta**, поскольку существует команда **start** и в последнем случае **lpc** не сможет их различить.

Если введенной информации недостаточно, чтобы распознать существующую команду, **lpc** выведет сообщение **?Ambiguous command** (Неопределенная команда). Если указанной команды не существует, программа выведет сообщение **?Invalid command** (Неверная команда).

Состояние очереди

Для проверки состояния очереди используется команда **status**, которую, как было показано ранее, можно сократить до **stat**. Она требует аргумента. Им может быть или **all** (состояние очередей всех демонов) или имя принтера. Например, команда:

```
lpc> status lp
lp:
        queuing is enabled
        printing is enabled
        2 entries in spool area
        waiting for lp to become ready (offline?)
lpc>
```

выдает состояние принтера по умолчанию. В данном случае очередь разрешена, печать разрешена и в очереди находится два задания, ожидающих печати. Процесса печати не происходит по той причине, что принтер в данный момент недоступен.

Командой **status** могут пользоваться и системные администраторы, и обычные пользователи.

Запрет печати и останов демона

Команды **abort** и **stop** предназначены для запрета печати и останова демона, который обрабатывает спул принтера (запустить эти команды может только системный

администратор). Обе команды требуют аргумента: **all** (запрет печати всех очередей и останов всех демонов) или имя принтера (команда выполняется лишь для него).

Команда **abort** немедленно прерывает печать и останавливает демон, обрабатывающий спул. Например:

```
lpc> abort lp
lp:
        printing  disabled
        daemon  (pid  597)  killed
lpc>
```

Вот вывод последующей команды **status**:

```
lpc>  status  all
lp:
        queuing  is  enabled
        printing  is  disabled
        2  entries  in  spool  area
        printer  idle
lpc>
```

Здесь необходимо отметить несколько моментов:

- Обратите внимание, что в спуле находится два задания. Команда **abort** просто закрывает демон и запрещает принтер. Она не удаляет задания из спула. После перезапуска демона и разрешения принтера задания из спула будут распечатаны.
- Обратите внимание также на то, что постановка заданий в очередь разрешена. Это значит, что пользователи могут пересылать задания на печать. Они будут находиться в очереди, ожидая перезапуска демона и разрешения принтера. После чего и они, и задания, уже находившиеся в очереди, будут распечатаны.

Команда **stop** работает подобно команде **abort**, однако она ожидает, пока не будет распечатано текущее задание, и только после этого запрещает печать и останавливает демон. Как и **abort**, команда **stop** не отключает очередь. Таким образом, задания размещаются в ней. Когда принтер станет доступен, они будут распечатаны.

СОВЕТ

Если принтер необходимо отключить на длительный срок (например, для выполнения текущего обслуживания), его имя следует сделать псевдонимом другого принтера в файле **/etc/printcap**. Если это системный принтер по умолчанию, обязательно сделайте **lp** также псевдонимом другого принтера. Тогда все задания будут перенаправляться на другой принтер, не требуя от пользователей внесения изменений в конфигурацию. Разумеется, предпочтительнее перенаправить вывод на принтер такого же типа и расположенный где-то неподалеку. Кроме того, всем пользователям нужно направить уведомление о том, где они смогут найти свои распечатки.

Запрещение постановки заданий в очередь

Ни **abort**, ни **stop** не запрещают постановку новых заданий в очередь. Когда принтер отключен, а демон завершил работу по команде **abort**, ничто не препятствует пользователям посылать задания, а очереди — принимать их, ожидая, пока принтер вновь не станет доступен. Поэтому, если принтер должен быть отключен на продолжительное время, необходимо запретить постановку новых заданий в очередь. Для этого есть два способа.

Первый заключается в использовании команды **disable**. Она требует указания аргумента: **all** (запрет всех очередей в системе) или имя принтера (запрещается постановка заданий в его очередь). Например:

```
lpc> disable lp
lp:
        queuing disabled
lpc> status all
lp:
        queuing is disabled
        printing is enabled
        3 entries in spool area
        waiting for lp to become ready (offline?)
lpc>
```

В данном случае очередь запрещена, но печать — разрешена. То есть, принтер продолжит печать всех заданий, находящихся в очереди (если это возможно), но очередь не будет принимать новых.

Если попытаться распечатать файл на принтере, очередь которого не принимает заданий, система выдаст следующее сообщение:

```
bash$ lpr myfile.txt
lpr: Printer queue is disabled
bash$
```

Другой способ запрещения очереди заключается в применении команды **down**. Как и **disable**, она требует хотя бы одного аргумента, имеющего тот же смысл, что и раньше.

Фактически, команда **down** запускает команду **stop**, которая запрещает печать и останавливает демон по окончании печати текущего задания, а затем команду **disable**, отключающую очередь. Дополнительным аргументом команды является сообщение. Сообщение записывается в файл **status** в каталоге спула. Содержимое сообщения отображается в выводе команды **lpq**, сообщая пользователям, почему очередь не принимает их запросов. Например:

```
lpc> down lp Printer is down for maintenance.
lp:
        printer and queuing disabled
lpc>
```

Если после этой команды запустить **lpq**, на экране появится следующее сообщение:

```
bash$ lpq
Warning: lp is down: Printer is down for maintenance.
Warning: lp queue is turned off
Printer is down for maintenance.
Rank    Owner           Job    Files                          Total Size
bash$
```

Для разрешения очереди, которая была запрещена ранее, достаточно просто ввести команду **enable** с аргументом **all** (включение всех очередей) или имя принтера. Например:

```
lpc> enable lp
lp:
        queuing enabled
lpc> status lp
lp:
        queuing is enabled
        printing is disabled
```

```
        3 entries in spool area
        Printer is down for maintenance.
lpc>
```

Обратите внимание, что, хотя эта команда и разрешила постановку заданий в очередь, она не разрешила печать на принтер. Поэтому в текущий момент очередь принимает задания, но они не выводятся на печать. Если для останова очереди использовалась команда **down**, то для ее включения следует применить команду **up**. Она разрешит постановку заданий в очередь, перезапустит демон, управляющий спулом, и разрешит печать на принтер. Однако она не устранит заданного ранее сообщения состояния. Для его очистки необходимо воспользоваться командой **restart**, которая обсуждается в следующем разделе.

Перезапуск демона, обрабатывающего спул

Если требуется перезапустить демон, управляющий спулом, применяется команда **restart**. Если в данный момент не запущено ни одного демона, эта команда запускает одну копию демона. Если демон уже выполняется, эта команда завершает его работу и затем заново запускает его (обратите внимание на предупреждение ниже). Этой командой можно воспользоваться для перезапуска "зависшего" демона.

Команда **restart** имеет один обязательный аргумент: **all** (перезапуск всех демонов печати в системе) или имя принтера, демон которого перезапускается. Например:

```
lpc> restart lp
lp:
                daemon (pid 2102) killed
lp:
              daemon started
lpc>
```

ПРЕДУПРЕЖДЕНИЕ

В некоторых случаях, команда **restart** останавливает выполняющийся демон, но не запускает его снова (как должно быть). Из-за этого следует проверять вывод команды **lpq**. Если в нем присутствует сообщение: **Warning: no daemon present** (Предупреждение: отсутствует демон), это значит, что демон был остановлен, но не запущен снова. В этом случае достаточно еще раз выполнить команду **restart** в интерактивном режиме программы **lpc**, и проблема решится.

Очистка каталога очереди

Команда **clean** программы **lpc** очищает каталог очереди. Фактически, она удаляет из него все управляющие файлы (предотвращая, таким образом, печать). Как и большинство команд **lpc**, она имеет один обязательный аргумент: либо **all**, либо имя принтера, каталог спула которого требуется очистить.

Изменение приоритета заданий печати

Для изменения порядка заданий в очереди применяется команда **topq**.

Только системный администратор может изменить порядок заданий в очереди. Команда поддерживает следующий синтаксис: **topq printer_name num_job(s)**, где **num_job(s)** отвечают тем заданиям, которые требуется переместить на вершину очереди. Если указать несколько идентификаторов, она распечатает задания в приведенном порядке. Первое задание из списка станет первым в очереди, второе — вторым и т.д. Предположим, например, что очередь выглядит следующим образом:

```
Rank     Owner      Job   Files                          Total Size
1st      murban     8     myfile.txt                     151625  bytes
2nd      murban     9     cardlist.txt                   38311  bytes
3rd      murban     10    lions.txt                      1113  bytes
4th      murban     12    schedule.txt                   6599  bytes
```

Для печати заданий в обратном порядке необходимо воспользоваться командой **topq lp 12 10 9 8.**

```
lpc>  topq  lp  12  10  9  8
lp:
        moved   cfA008simba.samplenet.org
        moved   cfA009simba.samplenet.org
        moved   cfA010simba.samplenet.org
        moved   cfA012simba.samplenet.org
lpc>
```

Теперь очередь выглядит по-другому:

```
Rank     Owner      Job   Files                          Total  Size
1st      murban     12    schedule.txt                   6599  bytes
2nd      murban     10    lions.txt                      1113  bytes
3rd      murban     9     cardlist.txt                   38311  bytes
4th      murban     8     myfile.txt                     151625  bytes
```

Вместо списка идентификаторов заданий параметром **topq** может служить имя пользователя. В этом случае **topq** перемещает все задания указанного пользователя на вершину очереди.

Обратите внимание, что **topq** не прерывает текущее выполняемое задание. Все задания, размещенные на вершине очереди, будут напечатаны после того, как завершится текущий процесс печати. Если другие задания необходимо распечатать сразу, для удаления текущего процесса следует воспользоваться командой **lprm**.

Выход из lpc

Для выхода из программы **lpc** достаточно ввести команду **quit** или **exit** в приглашении **lpc>**. Управление будет вновь передано командному интерпретатору.

Использование программы lpc в неинтерактивном режиме

Программу **lpc** можно применять и в неинтерактивном режиме. В этом случае в командной строке необходимо указать набор аргументов: команду, которую требуется выполнить, и ее параметры.

Когда **lpc** запускается в неинтерактивном режиме, программа ожидает, что ее первым аргументом будет команда с последующим списком параметров. Например:

```
# lpc restart lp
lp:
        daemon  (pid  2280)  killed
lp:
        daemon  started
#
```

В неинтерактивном режиме доступны те же команды, что описаны ранее.

Управление доступом к lpc

Иногда другим пользователям необходимо передать полномочия управлять принтером с помощью **lpc**, не предоставляя при этом прав доступа **root**. Для этого существует группа **operator**.

Любой пользователь, включенный в группу **operator**, может полноценно использовать программу **lpc**, не имея при этом полного доступа к системе. Пользователь может изменять приоритет заданий печати, запускать и останавливать демоны и включать или отключать очереди.

Более подробные сведения о группах, а также о том, как добавить пользователей в группу, даны в главе 10.

Основы сетевой печати

Печать по сети во многом подобна локальной печати. Для настройки сетевого принтера в файле **/etc/printcap** используется следующая строка:

```
simba|lp|local line printer:\
        :lp=::rm=nova:rp=simba:sd=/var/spool/lpd/simba:lf=/var/log/lpd-errs:
```

Эта запись определяет подключение к сетевому принтеру **simba** на хосте **nova**. Отметьте, что имя локального принтера (в строке 1) не обязательно должно совпадать с удаленным. Оно может быть любым. Тем, на какой принтер пересылается задание, управляет, фактически, запись **rp** в строке 2. Файл хранится в локальном каталоге спула лишь в том случае, если на удаленной машине в спуле нет места. Когда оно освобождается, файл перемещается в каталог спула на удаленной машине.

Обратите внимание, что здесь не нужно указывать имя входного фильтра, все действия по фильтрации предпринимаются на удаленном хосте. Принтер на удаленной машине необходимо сконфигурировать, следуя инструкциям, изложенным ранее в этой главе.

Решение проблем

При использовании принтеров могут возникнуть определенные проблемы. Ниже приведен список наиболее распространенных из них, а также даны способы их решения.

Принтер не получает данных, задания находятся в очереди

Запустите **lpq** и убедитесь, что запущен демон спула. Если команда выдает сообщение **Warning: no daemon present**, необходимо воспользоваться **lpc** для перезапуска демона. Кроме того, запустив **lpc**, убедитесь, что принтер не запрещен.

Принтер сигнализирует о приеме данных, но не печатает

Это симптом попытки переслать не-PostScript-данные PostScript-принтеру. Проверьте фильтр и убедитесь, что он корректно преобразует текст в формат PostScript.

Вместо изображения или Web-страницы печатаются сотни страниц мусора

Это зачастую служит симптомом пересылки PostScript-данных принтеру, не поддерживающему этот язык. Не-PostScript-принтер пытается распечатать такой файл

как обычный текст. Проверьте работу программы GhostScript и обратитесь к разделу "Печать файлов в формате PostScript на не-PostScript-принтерах".

Принтер работает медленно

Установите принтер в режим запроса (polled mode) (предполагается, что принтер подключен к параллельному порту). Воспользуйтесь следующей командой:

```
lptcontrol -p
```

Чтобы принтер переходил в этот режим при каждой загрузке, добавьте эту команду в один их начальных сценариев. См. раздел в начале главы, посвященный утилите **lptcontrol**.

Вывод печатается "лесенкой"

Симптомы этой проблемы выглядят примерно так:

```
Line one of the file.
                    Line two of the file.
                                        Line three of the file.
```

Такой вывод будет продолжаться до тех пор, пока текст не дойдет до края страницы. Это достаточно распространенная проблема, причиной которой служит разница в том, как UNIX и DOS/Windows интерпретируют символ перевода строки (LF). Когда Windows переходит на новую строку, эта система пересылает два символа: возврат каретки и перевод строки. С другой стороны, UNIX пересылает только символ перевода строки, подразумевая и возврат каретки. Если принтер ожидает комбинации символов в стиле DOS, а получает только символ перевода строки, он перемещает бумагу, но не перемещает печатающую головку к началу строки. Существует несколько способов решения этой проблемы.

Первый заключается в том, чтобы ознакомиться с конфигурацией принтера и настроить то, как следует интерпретировать символ перевода строки. Обратитесь для этого к документации вашего принтера.

Второй способ состоит в создании фильтра, преобразующего символ LF в комбинацию CR и LF.

Если принтер понимает язык HP-PCL, для решения этой задачи рекомендуется использовать следующий фильтр:

```
#!/bin/sh
#
# hpif - Простой текстовый входной фильтр lpd для принтеров,
# поддерживающих язык HP-PCL.
# Установлен в каталоге /usr/local/libexec/hpif
#
# Просто копирует stdin в stdout. Игнорирует все аргументы
# фильтра.
# Указывает принтеру преобразовывать LF в CR+LF. По окончании
# возвращает страницу.

printf "\033&k2G" && cat && printf "\033&l0H" && exit 0
exit 2
```

О том, как установить текстовый фильтр, рассказано в соответствующем разделе ранее в этой главе.

Если принтер не поддерживает язык HP-PCL, для преобразования LF в CR LF можно воспользоваться командой **tr**. Для этого подойдет следующий фильтр:

```
#!/bin/sh
# Фильтр, исправляющий эффект "лесенки" на принтерах, не
# поддерживающих язык PCL.
/bin/cat | tr '\ 13' '\ 13\ 10' && exit 0
exit 2
```

Весь текст печатается в одной строке, поверх уже напечатанного текста

Эта проблема, очевидно, противоположна эффекту "лесенки". Она встречается довольно редко. Фактически, она свидетельствует о том, что символ LF интерпретируется принтером как CR. Печатающая головка возвращается к началу строки, но бумага не прокручивается вверх.

Для решения проблемы необходимо внести изменения в конфигурацию принтера, чтобы символы CR и LF обрабатывались корректно. Обратитесь к документации принтера.

Если вы не сможете решить проблему аппаратно, воспользуйтесь фильтром, применявшимся для решения проблемы "лесенки".

ГЛАВА 17

Конфигурирование ядра

- Роль ядра
- Зачем конфигурировать собственное ядро?
- Конфигурационные файлы ядра
- Создание конфигурационного файла ядра
- Компиляция и установка ядра
- Добавление файлов устройств в каталоге /dev
- Аварийное восстановление

Эта глава посвящена одной из самых пугающих для начинающего пользователя UNIX тем: конфигурированию и сборке ядро. В мире настольных операционных систем с этим никогда не приходится сталкиваться, однако это обязательная часть работы с системой, распространяемой в открытом коде и находящейся в постоянной разработке. Ведь новые устройства требуют и новой функциональности от ядра.

Конфигурирование ядра представляет собой центральную часть настройки FreeBSD и подобных ей систем. Хорошо настроенное ядро служит требованиям системы и пользователей на все 100%, не включает ненужный багаж, обычно присутствующий в неоптимизированном ядре, и работает значительно быстрее. Данная глава расширяет понимание процесса конфигурирования ядра.

Роль ядра

Ядро — главный выполняемый файл в системе. Оно запускается первым после загрузчиков и продолжает выполняться в течение всего сеанса работы системы. В его задачи входит наблюдение за всеми процессами в системе, поддержка протокола TCP/IP и других сетевых задач, управление доступом ко всем устройствам системы, распределение памяти и многое другое.

Каждая операционная система — от MS-DOS и Windows до самых высокотехнологичных мэйнфреймов — имеет ядро. В некоторых системах оно спрятано, в других нет. В Windows, например, это выполняемый файл в каталоге **C:\WINDOWS\SYSTEM**, в классической Mac OS оно вообще недоступно файловой системе. Во многих UNIX-системах его традиционным местоположением является корневой каталог файловой системы. Во FreeBSD ядро находится в каталоге **/boot**. Ядро по умолчанию ("основное" ядро) является частью каждого релиза операционной системы, и при выходе новой версии изменения ядра, как правило, связаны с добавлением драйверов новых устройств, которые оно поддерживает. Ядро отвечает за работу со всеми устройствами, подключенными к системе. Именно поэтому при инсталляции драйверов новых устройств в Windows, обычно, приходится перезагружать машину — ядро изменилось. Системе требуется перезапуск для того, чтобы воспользоваться новым ядром. Если драйвер нового устройства используется как расширение ядра, или модуль, перезагрузки не требуется.

Во FreeBSD используется *микроархитектура ядра* (*microkernel architecture*), т.е. ядро как таковое является сравнительно небольшим и модульным. Операционные системы Windows NT и Mach (ядро, на основе которого построена система Mac OS X) также представляют нам примеры микроядер, где новые устройства зачастую добавляются как модули, которые можно загружать или выгружать на этапе исполнения без перекомпиляции ядра. Linux и Windows 95/98 являются *монолитными* (*monolithic*) ядрами, где код ядра оптимизирован для более высокой производительности с минимальными контекстными переключениями. Такая архитектура упрощает поддержку кода ядра для разработчиков, но в большинстве случаев требует от администратора перекомпиляции ядра всякий раз, когда добавляется поддержка нового устройства.

Разумеется, такое разделение является не вполне точным, Linux, например, поддерживает модули, а ядро FreeBSD иногда требует перекомпиляции. Различие между микро- и монолитной архитектурой ядер является в определенной мере философским и связано с большим кругом вопросов, чем просто поддержка устройств. Фундаментальное отличие состоит в следующем: механизм микроядра позволяет переда-

вать вызовы, не принадлежащие основной системе, пользовательскому уровню обработки вместо их внутренней обработки. В этом случае ядро включает лишь самые необходимые элементы, которые требуют наивысшего статуса выполнения. Целью такой архитектуры является повышение устойчивости ядра на этапе исполнения, возможность управления процессами, обеспечение легкости в поддержке модулей устройств.

Хотя эта схема выглядит очень заманчивой, тем не менее ядро, где каждое устройство и опция поддерживаются как модули, загружаемые во время работы, остается недостижимым идеалом. По мере работы с системой FreeBSD вы рано или поздно достигнете момента, когда ядро потребуется перекомпилировать.

Зачем конфигурировать собственное ядро?

По умолчанию во FreeBSD устанавливается ядро **GENERIC**. Оно содержит поддержку ряда возможных конфигураций устройств, позволяя системе FreeBSD работать на самых разных типах машин. Природа аппаратного обеспечения машин с архитектурой x86 предполагает, что в такое ядро встроено множество драйверов. Операционная система, предназначенная для работы со строго определенным набором оборудования (например, IRIX для SGI или Mac OS X для Apple), может обойтись гораздо меньшим числом поддерживаемых устройств. Ядро **GENERIC** содержит различные опции распределения памяти и оптимизации среднего уровня, а также дополнительные элементы, поддерживающие ядро в модернизированном состоянии (насколько это возможно). Очевидно, что за универсальность приходится платить. Таким образом, для вашей конкретной системы ядро можно сконфигурировать более эффективно.

На этапе загрузки ядро проверяет все типы знакомых ему устройств. Об этом свидетельствует достаточно длительный вывод, который мы видим на экране. Это ядро пытается обнаружить десятки различных устройств, поддержка которых включена в ядро **GENERIC**. Нет ничего страшного, что ядро не находит их, но поиски замедляют загрузку. Устранив все ненужные устройства из ядра, можно значительно снизить время загрузки системы и уменьшить размер ядра (следовательно, и объем памяти, требующийся ему). На современных системах с сотнями мегабайт оперативной памяти об этом не приходится беспокоиться, но эти факторы весьма важны для машин, которые отвечают минимальным требованиям FreeBSD.

ПРИМЕЧАНИЕ

Даже ядро **GENERIC** позволяет отключить проверку всех известных ему устройств. Для этого предназначен конфигуратор загрузки (см. приложение В). Он позволяет явно устранить те устройства, которые вызывают конфликты прерываний или адресов памяти (или не присутствуют в системе). Это, естественно, не изменяет самого ядра и скомпилированных с ним элементов, но, по крайней мере, ускоряет загрузку системы.

При добавлении к системе таких устройств, как USB-периферия, звуковые карты, SCSI-контроллеры и т.д., пользоваться ими можно лишь после того, как их поддержка включена в ядро. То же касается и новых типов файловых систем (например, EXT2FS, см. главу 9). Многие устройства и файловые системы поддерживаются как модули, однако большинство компонентов требует включения в ядро. Перекомпиляция ядра дает также возможность произвести и другие настройки системы (напри-

мер, количество буферов памяти, специальные свойства управления памятью или имя ядра). Поэтому перекомпиляция ядра дает возможность значительно повысить производительность системы.

К примеру, симметричная многопроцессорность (SMP, Symmetric Multi-Processing) не присутствует в ядре **GENERIC**. Если в системе используется более одного процессора, необходимо собрать специальное ядро, чтобы ресурсы применялись эффективно.

Использование dmesg для сбора информации о запуске ядра

Чтобы исключить ненужные устройства из ядра, необходимо четко знать, какие именно устройства присутствуют в системе, чтобы случайно не отключить их поддержку. Как упоминалось ранее, ядро проверяет все известные ему устройства на этапе загрузки и печатает состояние каждой проверки. Эта информация не только выводится на экран при загрузке, но и записывается во внутренний буфер сообщений. Просмотреть его позволяет команда **dmesg**.

Вывод **dmesg**, как правило, достаточно велик, поэтому его следует перенаправить команде **less**, как показано в листинге 17.1.

Листинг 17.1 Вывод команды dmesg перенаправлен less для удобства чтения

```
# dmesg | less
Copyright (c) 1992-2001 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993,
1994
    The Regents of the University of California. All rights reserved.
FreeBSD 4.2-STABLE #1: Sun Mar 4 14:05:42 PST 2001
       btman@stripes.arclight.net:/usr/obj/usr/src/sys/STRIPES
Timecounter "i8254" frequency 1193182 Hz
CPU: Pentium III/Pentium III Xeon/Celeron (598.06-MHz 686-class CPU)
  Origin = "GenuineIntel" Id = 0x683 Stepping = 3
  Features=0x383f9ff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,SEP,MTRR,PGE,MCA,CMOV,
➥PAT,PSE36,MMX,FXSR,SSE>
real memory = 132907008 (129792K bytes)
config> di sn0
config> di lnc0
config> di ie0
config> di fe0
config> di ed0
config> di cs0
config> di bt0
config> di aic0
config> di aha0
config> di adv0
config> q
avail memory = 125018112 (122088K bytes)
Preloaded elf kernel "kernel" at 0xc044a000.
Preloaded userconfig_script "/boot/kernel.conf" at 0xc044a09c.
Pentium Pro MTRR support enabled
md0: Malloc disk
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pci0: <Intel model 1132 VGA-compatible display device> at 2.0 irq 11
pcib1: <PCI to PCI bridge (vendor=8086 device=244e)> at device 30.0
on pci0
pci1: <PCI bus> on pcib1
```

```
fxp0: <Intel PLC 10/100 Ethernet> port 0xde80-0xdebf mem 0xff8fe000-
↪0xff8fefff irq 11 at device 8.0 on pci1
fxp0: Ethernet address 00:d0:b7:c7:74:f1
fxp1: <Intel Pro 10/100B/100+ Ethernet> port 0xdf00-0xdf3f mem
↪0xff700000-0xff7f
ffff,0xff8ff000-0xff8fffff irq 11 at device 9.0 on pci1
fxp1: Ethernet address 00:d0:b7:bd:5d:13
isab0: <PCI to ISA bridge (vendor=8086 device=2440)> at device 31.0
↪on pci0
isa0: <ISA bus> on isab0
atapci0: <Intel ICH2 ATA100 controller> port 0xffa0-0xffaf at device
↪31.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
```

На самом деле вывод продолжается и содержит дальнейший поиск устройств, проверку файловых систем и запуск демонов. В нем присутствуют ошибки времени исполнения, связанные с загрузкой разных устройств. Но в данном случае нас интересуют только сообщения из блока, приведенного в листинге.

Первая часть загрузочных сообщений (после строк об авторских правах и информации о процессоре) содержит конфигурацию ядра, установленную в конфигураторе загрузки. Строка, начинающаяся с **config>**, представляет собой команду, запущенную в визуальном режиме, скорее всего (см. строки **di**), для удаления ненужных устройств. Такие устройства больше не появятся в выводе **dmesg**. Если же вы не удалили их с помощью конфигуратора, ядро будет искать их и далее.

Если в выводе присутствует сообщение not found, оно отвечает драйверу, который можно удалить из ядра. При конфигурировании ядра желательно держать окно **dmesg** открытым, чтобы случайно не удалить те устройства, которые система находит при загрузке.

Конфигурационные файлы ядра

На момент написания этой книги FreeBSD не поддерживала визуальной утилиты наподобие **linuxconf** в Red Hat Linux или диалогового процесса **make config**, присущего другим дистрибутивам Linux. Ядро FreeBSD конфигурируется с помощью текстовых файлов. Такой метод может показаться запутанным, однако он обеспечивает определенную гибкость, не присущую визуальным методам.

Визуальное средство конфигурирования обеспечивает интерактивную "обратную связь", позволяя включать/отключать устройства и опции по одной. Это плюс. Однако оно не позволяет поддерживать сразу несколько конфигураций, сравнивать их с помощью таких утилит, как **diff** и **grep**, и использовать основную конфигурацию **GENERIC**, нескольких предыдущих корректных конфигураций и файл **NOTES** в качестве директивы. Текстовый файл позволяет увидеть все опции устройств одновременно, тогда как интерактивная утилита зачастую оказывается чересчур сложной и запутанной, запрашивая включение/отключение каждой опции. Если только вы не помните степень важности всех отдельных опций и их последовательность, тяжело отследить те из них, которые необходимо выбрать. Методика FreeBSD позволяет настраивать **config**-файл, внося изменения из предыдущих версий или просто копируя "стандартную" конфигурацию другой системы с последующей сборкой ядра.

Если в системе FreeBSD установлен исходный код ядра, конфигурационные файлы находятся в каталоге **/sys/i386/conf**. Среди них находится и файл **GENERIC**.

ПРИМЕЧАНИЕ

Если вы работаете на платформе Alpha, а не x86, в имени предыдущего каталога необходимо заменить **i386** на **alpha**. Некоторые параметры, указанные в этой главе будут другими, однако сам подход останется прежним.

Конфигурационный файл GENERIC

Откройте файл **GENERIC** в текстовом редакторе. Просмотрите различные опции, не внося изменений (поскольку сам файл **GENERIC** изменять не следует). Вы увидите, что файл содержит поддержку самых разных типов устройств и машин. Вот, например, "обязательный" блок:

```
machine         i386
cpu             I386_CPU
cpu             I486_CPU
cpu             I586_CPU
cpu             I686_CPU
ident           GENERIC
maxusers        32
```

Под ним находятся "дополнительные" строки: элементы, включенные в ядро **GENERIC** по умолчанию, но не обязательные в каждом ядре. Они включают в себя полезные опции ядра, показанные в листинге 17.2.

Листинг 17.2 Встроенные опции ядра GENERIC

```
options     MATH_EMULATE        #Support for x87 emulation
options     INET                #InterNETworking
options     INET6               #IPv6 communications protocols
options     FFS                 #Berkeley Fast Filesystem
options     SOFTUPDATES         #Enable FFS soft updates support
options     MFS                 #Memory Filesystem
options     MD_ROOT             #MD is a potential root device
options     NFS                 #Network Filesystem
options     NFS_ROOT            #NFS usable as root device, NFS required
options     MSDOSFS             #MSDOS Filesystem
options     CD9660              #ISO 9660 Filesystem
options     DEVFS               #Device Filesystem
options     PROCFS              #Process filesystem
options     COMPAT_43           #Compatible with BSD 4.3 [KEEP THIS!]
options     SCSI_DELAY=15000    #Delay (in ms) before probing SCSI
options     UCONSOLE            #Allow users to grab the console
options     USERCONFIG          #boot -c editor
options     VISUAL_USERCONFIG   #visual boot -c editor
options     KTRACE              #ktrace(1) support
options     SYSVSHM             #SYSV-style shared memory
options     SYSVMSG             #SYSV-style message queues
options     SYSVSEM             #SYSV-style semaphores
options     P1003_1B            #Posix P1003_1B real-time extensions
options     _KPOSIX_PRIORITY_SCHEDULING
options     KBD_INSTALL_CDEV    # install a CDEV entry in /dev
```

Остальную часть файла занимают строки **device**. В них указаны все типы устройств, включенные в **GENERIC**. Большинство из них необходимо исключить из оптимизируемого ядра. Ниже приведен один из блоков строк **device**, содержащихся в файле. См. листинг 17.3.

Листинг 17.3 Некоторые драйверы устройств, встроенные в ядро GENERIC

```
# PCI Ethernet NICs that use the common MII bus controller code.
# NOTE: Be sure to keep the 'device miibus' line in order to use
# these NICs!
device      miibus      # MII bus support
device      dc          # DEC/Intel 21143 and various workalikes
device      fxp         # Intel EtherExpress PRO/100B (82557, 82558)
device      pcn         # AMD Am79C79x PCI 10/100 NICs
devicerl          # RealTek 8129/8139
devicesf          # Adaptec AIC-6915 (''Starfire'')
devicesis         # Silicon Integrated Systems SiS 900/SiS 7016
deviceste         # Sundance ST201 (D-Link DFE-550TX)
devicetl          # Texas Instruments ThunderLAN
devicetx          # SMC EtherPower II (83c170 ''EPIC'')
devicevr          # VIA Rhine, Rhine II
devicewb          # Winbond W89C840F
devicexl          # 3Com 3c90x (''Boomerang'', ''Cyclone'')
```

Подсказки ядру относительно устройств

ПРИМЕЧАНИЕ

Данный раздел посвящен FreeBSD 5.0, так как FreeBSD 4.x не поддерживает файлы, содержащие подсказки ядру относительно устройств.

FreeBSD использует *файлы*, содержащие *подсказки об устройствах* (*device hints*), которые позволяют абстрагировать атрибуты различных устройств, не требуя их статической компиляции с ядром. Ранее атрибуты задавались в конфигурационных файлах ядра следующим образом:

```
device          ata0    at isa? port IO_WD1 irq 14
device          ata1    at isa? port IO_WD2 irq 15
```

Теперь достаточно указать лишь строку:

```
device          ata
```

Это связано с тем, что атрибуты устройства **ata** установлены в файле параметров устройств **/boot/device.hints**, к которому ядро обращается во время загрузки. Вот строки из него, отвечающие данному примеру:

```
hint.ata.0.at="isa"
hint.ata.0.port="0x1F0"
hint.ata.0.irq="14"
hint.ata.1.at="isa"
hint.ata.1.port="0x170"
hint.ata.1.irq="15"
```

Эти атрибуты сообщают ядру на какой шине, по какому адресу и на каком прерывании (IRQ — interrupt request) находится устройство **ata**. Изменение этих атрибутов не требует перекомпиляции ядра. Но если вы предпочитаете скомпилировать атрибуты статически, достаточно включить следующую строку:

```
#To statically compile in device   wiring   instead   of /boot/device.hints
#hints         "GENERIC.hints"      #Default  places  to  look  for  devices.
```

Файл LINT

В каталоге **/sys/i386/conf** содержится файл под названием **LINT**, в котором описаны все возможные опции и устройства, доступные в данной системе. Это еще один

файл, который также не следует изменять. Просто откройте его в текстовом редакторе и просмотрите содержимое.

Все устройства, записи о которых содержатся в **GENERIC**, имеют соответствующие записи и в файле **LINT**. Причем в этом файле ничего не закомментировано (кроме настоящих комментариев). Теоретически, любую строку можно скопировать в специальный конфигурационный файл. Обычно в этом возникает необходимость лишь при добавлении необычного устройства или опции ядра. В этом случае нужно, в первую очередь, обратиться к файлу **LINT**.

Один из мифов о FreeBSD указывает на возможность собрать ядро на основании файла **LINT**, включая все возможные опции и драйверы. На самом деле это практически невозможно, поскольку многие опции ядра являются взаимоисключающими или неустойчивыми.

Обратите внимание, что во FreeBSD 5.0 файл **LINT** отсутствует, вместо него используется файл **NOTES**, находящийся в том же каталоге. Файл **NOTES** похож на **LINT**, но он содержит комментарии и параметры устройств. При желании файл **LINT** можно легко создать из **NOTES**, используя команду **make lint**. В результате будет создан конфигурационный файл, которым теоретически можно воспользоваться для сборки огромного ядра, содержащего все возможные опции и драйверы (скорее всего, оно не сможет даже скомпилироваться, не говоря уже о загрузке).

Создание конфигурационного файла ядра

Итак, мы подошли к ключевому моменту создания нового ядра. Файл **GENERIC** не следует изменять, поскольку он обновляется при каждой синхронизации исходного кода или обновлении системы. Чтобы изменения, внесенные вами, не были перезаписаны в будущем, необходимо создать его копию. По традиции имя конфигурационного файла должно состоять из одного слова, написанного прописными буквами. В данном примере ядро называется **CUSTOM**:

```
# cp GENERIC CUSTOM
```

Теперь файл **CUSTOM** можно изменять нужным вам образом. Первое, что нужно сделать, — это изменить все ссылки **GENERIC** на **CUSTOM**. Затем следует удалить лишние записи **cpu** и, возможно, изменить значение **maxusers**. Вот пример блока конфигурационного файла для сервера с процессором Pentium III:

```
machine         i386
cpu             I686_CPU
ident           CUSTOM
maxusers        64
```

ПРИМЕЧАНИЕ

Обратите внимание на значение опции **maxusers**: это не максимально возможное число пользователей, зарегистрированных в системе (последнее задается в строке **device pty <num>**). Эта переменная отвечает за более тонкие настройки: по ее значению вычисляются размеры внутренних таблиц, например, максимально возможное число процессов (оно равно **16*maxusers+20**). Значение этой опции примерно равно ожидаемому числу пользователей, однако не следует делать его меньшим 4. В этом случае допустимое число процессов будет превышено значительно раньше, чем можно ожидать.

Теперь приступим к настройке опций и устройств. Обратитесь к файлу **NOTES**, чтобы выяснить, что именно вы изменяете, и держите окно с выводом **dmesg** откры-

тым. Не удаляйте из конфигурационного файла драйверы устройств, которые система находит при загрузке! Не удаляйте строки, а размещайте в начале символ комментария #.

Если вы сомневаетесь в какой-либо опции или устройстве — не запрещайте ее. Ядро **GENERIC**, безусловно, содержит множество элементов, которые не требуются в конкретной системе, однако лучше оставить что-то лишнее, чем исключить необходимое, особенно если речь идет о производственном сервере.

Компиляция и установка ядра

Когда работа с конфигурационным файлом закончена, останутся сущие пустяки. Сборка ядра представляет собой простой процесс, требующий (в теории) лишь трех команд:

```
# cd /usr/src
# make buildkernel KERNCONF=CUSTOM
# make installkernel KERNCONF=CUSTOM
```

Аргумент **KERNCONF** указывает на используемый конфигурационный файл, если он опущен, подразумевается **GENERIC**. На первом шаге производится синтаксический анализ конфигурационного файла, настройка каталога для сборки, сборка всех зависимостей и непосредственно ядра, на втором — установка ядра в каталог **/boot** (при этом текущее ядро переименовывается в **/boot/kernel.old**). Чтобы воспользоваться новым ядром, необходимо перезагрузить машину. Можно объединить обе команды **make** в одну:

```
# make kernel KERNCONF=CUSTOM
```

ПРИМЕЧАНИЕ

У ядра установлен флажок **schg** (системно-неизменяемый). Это значит, что даже пользователь **root** не может удалить или перезаписать файл, не отключив предварительно этот флажок. Цель **make installtarget** пытается отключить его до установки нового ядра. Но если система работает на уровне **securelevel** 1 или выше (эта настройка защиты выбирается во время инсталляции или устанавливается в файле **/etc/rc.conf**, см. **man securelevel**), флажок текущего ядра нельзя отключить. Необходимо перезагрузиться в однопользовательском режиме (переход в него с помощью команды **shutdown** в этом случае не работает) и продолжить инсталляцию нового ядра.

Добавление файлов устройств в каталоге /dev

ПРИМЕЧАНИЕ

Данный раздел неприменим к FreeBSD 5.0, поскольку в этой версии используется интерфейс файловой системы DEVFS к системным устройствам и виртуальная файловая система **/dev** создается динамически во время загрузки системы.

Многие устройства, добавленные в конфигурацию ядра, могут отсутствовать в каталоге **/dev**. Последний включает лишь те устройства, которые или являлись частью основной системы, или были явно добавлены при последующем использовании системы. Файлы устройств не генерируются автоматически, поэтому их необходимо создать вручную для всех устройств, добавленных в ядро.

Сделать это достаточно просто. После перезагрузки системы с новым ядром необходимо перейти в каталог **/dev** и запустить сценарий **MAKEDEV**:

```
# ./MAKEDEV
```

Все устройства, поддерживаемые текущим ядром, будут созданы как "специальные файлы" в каталоге **/dev**, которыми можно воспользоваться для взаимодействия с новыми устройствами. Более эффективно прямое решение — создать файлы лишь новых устройств (например, в ядро был включен звуковой драйвер **snd0**). Для включения только этого устройства применяется следующая команда:

```
# ./MAKEDEV snd0
```

Аварийное восстановление

К сожалению, ошибки при компиляции достаточно широко распространены. Если вы пытаетесь скомпилировать ядро из исходных кодов, установленных при инсталляции системы, и процесс завершается ошибкой (обычно строкой "***** Error code 1**"), обычно эта ошибка — результат неустойчивой опции ядра или устройства, которую вы включили. Попытайтесь установить, к чему именно относится эта ошибка, просмотрев последние строки вывода компилятора. Если это не очевидно, посетите Web-сайт FreeBSD (**http://www.freebsd.org**) и произведите поиск в архивах рассылки по уникальным словам из сообщения, сгенерированного компилятором.

Если исходный код был обновлен после инсталляции системы (об этом процессе рассказано в главе 18), то, скорее всего, код находится в нестабильном состоянии (особенно в ветви -CURRENT или -STABLE) и найденные ошибки будут исправлены в ближайшие несколько дней. Поэтому желательно подписаться на соответствующий список рассылки: **freebsd-current** или **freebsd-stable** (подробности подписки — на Web-сайте FreeBSD).

Иногда события развиваются и по-другому: ядро компилируется, устанавливается и перегружается корректно, а затем при попытке загрузиться выдает предупреждение. Это может быть связано с проблемами монтирования файловых систем, перехватом ошибки ядра или другими режимами неудачной загрузки. Паниковать в этом случае не стоит, поскольку у вас всегда есть возможность вывести систему из подобного состояния.

Перегрузите машину. Нажмите соответствующую функциональную клавишу, чтобы миновать загрузчик. Затем в приглашении загрузчика (когда он в течение 10 секунд ожидает ввода) введите **/boot/kernel.old** и система загрузит предыдущее, работоспособное ядро.

ВНИМАНИЕ!

Помните, что при следующей попытке сборки и инсталляции ядра файл **/boot/kernel.old** будет перезаписан текущим (т.е. неработоспособным ядром). Чтобы избежать этого, сделайте копию ядра **kernel.old** с другим именем, например, **/boot/kernel.frank**. Этим ядром вы всегда сможете воспользоваться в случае неудачной загрузки.

Существует и другая проблема: система загружается корректно и полностью, однако такие системные утилиты, как **ps**, **top** и **w**, не работают. В большинстве случаев это означает, что исходный код ядра значительно новее, чем тот, из которого собрана система, и библиотека **libkvm** устарела. В этом случае необходимо перекомпилировать библиотеку **libkvm**. Тем не менее наилучшее решение — собирать ядро из того же исходного кода, из которого собрана вся система. Использование обновленного исходного кода ядра допустимо, когда вся система собирается заново командой **make world** (см. главу 18).

18 ГЛАВА

Поддержка FreeBSD

- Отслеживание исходного кода FreeBSD
- Что такое make world?
- На что следует обратить внимание перед сборкой всей системы
- Задачи, выполняемые до сборки всей системы
- Сборка системы из исходных файлов
- Использование *mergemaster* для проверки измененных конфигурационных файлов
- Перезагрузка системы после обновления

Любая операционная система развивается, и должна поддерживаться в актуальном состоянии. Такие элементы, как обновления защиты, исправления ошибок и поддержка новых технологий должны выполняться как можно быстрее. Пользователи не могут ждать выхода новой полной версии, поскольку для этого требуется от шести месяцев до трех лет. В мире, живущем по времени Internet (когда защита системы может быть взломана хакерами через несколько часов после того, как опубликована информация о дыре), поддержка операционной системы на современном уровне требует быстроты реакции, измеряемой минутами!

Никто из производителей операционных систем не отрицает этой необходимости. Microsoft выпускает периодические обновления в виде "сервис-пакетов" (service pack) для пользователей Windows NT/2000. Обновления для индивидуальных приложений Windows и Macintosh появляются в течение нескольких дней с момент обнаружения каких-либо проблем. Подобный механизм обновлений присутствует и в Linux. Система FreeBSD также позволяет осуществлять поддержку подобным способом. Но поскольку в ней пользователи имеют доступ к исходному коду, поддержка возможна и с помощью более удобных методов.

Основные версии FreeBSD (4.0, 5.0) выходят с интервалом один-два года, промежуточные (3.4, 4.1.1), как правило, через три-шесть месяцев. Но как бы часто ни появлялись обновления, с точки зрения ежедневных потребностей администратора, отвечающего за защиту системы, это случается слишком редко. Чтобы обновлять систему "в реальном времени", воспользуйтесь способами, которые обеспечивает FreeBSD (например, CVSup и **make world**).

Отслеживание исходного кода FreeBSD

Простейший и наиболее распространенный путь обновления заключается в ожидании новой официальной версии системы. Этот метод подразумевает несколько больше, чем просто получения компакт-диска (или его образа) и инсталляции новой версии с применением методов, описанных в главе 2. Однако это слишком медленное решение для наших дней. Чтобы идти в ногу со временем, необходимо постоянно следить за исходным кодом системы и приложений.

Дерево файлов с исходным кодом FreeBSD содержится в хранилище CVS (CVS repository), представленном на нескольких зеркальных серверах и поддерживаемое небольшим ядром сотрудников. В отличие от модели Linux (доступны все исходные файлы ядра, но при этом каждый отдельный дистрибутив имеет собственный набор выполняемых файлов и библиотек, индивидуальную структуру файловой системы и собственные правила использования исходного кода), FreeBSD структурирована таким образом, что вся система доступна любому пользователю в исходном коде. Механизмом, применяемым для работы со структурой исходный код и внесение исправлений в нее, является CVS.

Объяснение ветвей исходного кода STABLE и CURRENT

Исходный код FreeBSD находится в состоянии постоянного изменения. Исправляются ошибки, обновляются утилиты (и расширяются их возможности), добавляются новые свойства, структура системы подвергается реорганизации. Параллельно поддерживается несколько ветвей дерева исходного кода. В основном, в активной разработке находятся две различных ветви. Более старые ветви почти не подвергаются поддержке и разработке.

Хранилище кода можно рассматривать как дерево, его ствол растет, и на вершине часто появляются новые ветви. Самая верхняя из них называется CURRENT (текущая), а находящаяся под ней — STABLE (стабильная).

При появлении каждой новой версии (3.x, 4.x, 5.x и т.д.) она ответвляется от основного кода ядра, т.е. от "ствола" дерева. Это довольно распространенная модель разработки: она позволяет авторам заморозить определенный набор свойств так, чтобы его дальнейшее развитие не вызывало неустойчивости системы, поскольку новые свойства добавляются блоками.

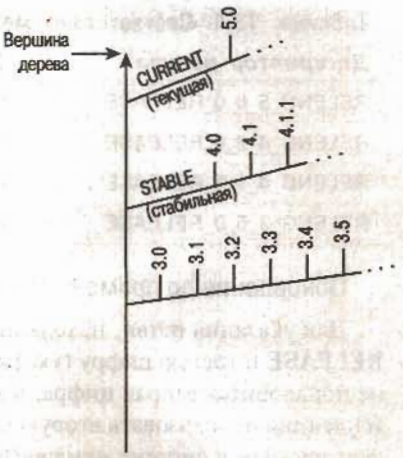


Рисунок 18.1. *Дерево исходного кода FreeBSD, показывающее взаимосвязь между ветвями CURRENT и STABLE.*

Важно помнить, что CURRENT — это не та версия, которую можно поставить на производственный сервер! CURRENT представляет собой наиболее современную версию кода, а значит — самую нестабильную. В этой версии тестируются самые новые свойства системы. Версия CURRENT предназначена для разработчиков и для тех, кому — без всякой альтернативы — немедленно нужны новые свойства, нужны еще до того, как версия будет объявлена STABLE (стабильной).

В определенный момент разработки версии CURRENT, обычно после первого релиза (например, 5.0), начинает считаться стабильной. После чего обозначения CURRENT и STABLE сдвигаются по дереву на одну ветвь вверх. CURRENT обозначает новую, самую верхнюю ветвь, а STABLE — бывшую ветвь CURRENT, дозревшую до использования на серверах. Новые релизы анонсируются в обеих ветвях в течение нескольких месяцев. Рано или поздно набор свойств в версии CURRENT развивается до требуемого уровня стабильности, а версия STABLE перестает удовлетворять текущим запросам пользователей. В этот момент дерево вырастает вновь, давая начало новому циклу.

Выбор цели обновления

Обновление системы можно произвести в любой момент, независимо от того, в какой точке дерева находится процесс разработки. Для этого есть две возможности: во-первых, можно выбрать версию, в которой состояние кода заморожено и содержится в статическом хранилище; во-вторых, можно просто синхронизировать исходный код посредством CVSup. (Это похоже на синхронизацию портов, о которой говорилось в главе 15.) После чего нужно собрать новую версию системы.

В этой главе обсуждается синхронизация кода одним из приведенных способов и сборка новой системы. Единственное практическое различие между двумя этими методами заключается в дескрипторе, используемом в конфигурации CVSup. Все дескрипторы начинаются с префикса RELENG — RELease ENGineer (Инженер релиза). Они указывают, что ветвь была официально начата инженером релиза из основной команды FreeBSD.

Обновление до релиза

Дескрипторы версии релиза содержат три цифры, разделенные подчерками (вместо точек), и суффикс **RELEASE**. В табл. 18.1 показано соответствие между дескрипторами релизов и номерами версий.

Таблица 18.1 Соответствие между дескрипторами релизов и номерами версий

Дескриптор релиза	*Номер версии*
RELENG_5_0_0_RELEASE	FreeBSD 5.0
RELENG_4_2_0_RELEASE	FreeBSD 4.2
RELENG_4_1_1_RELEASE	FreeBSD 4.1.1
RELENG_3_5_0_RELEASE	FreeBSD 3.5

Обновление до промежуточной версии

Для указания ветви, находящейся в процессе разработки, следует опустить суффикс **RELEASE** и третью цифру (указывающую на номер обновления релиза). Возможно, вам не понадобится вторая цифра, поскольку в современных релизах FreeBSD наблюдается тенденция использовать вторую цифру в качестве номера обновления релиза, что позволяет перейти к системе нумерования с двумя индексами. См. табл. 18.2.

СОВЕТ

Обратите внимание, что табл. 18.2 относится к лету 2001 года. Из-за динамической природы разработки ветвей подобная информация со временем устаревает. Узнать о текущем состоянии разработки можно узнать по адресу **http://www.freebsd.org/handbook/cvsup.html**.

Таблица 18.2 Дескрипторы ветвей и обозначение версий для промежуточных версий кода

Дескриптор ветви	*Обозначение версии*
RELENG_5	FreeBSD 5-CURRENT
RELENG_4_3	FreeBSD 4.3-STABLE
RELENG_4	FreeBSD 4-STABLE (Больше не поддерживается)

Что такое make world?

Сборка всей системы из файлов с исходным кодом представляет более деликатный подход по сравнению с инсталляцией с компакт-диска, но она полностью сочетается с методикой работы FreeBSD: сборка выполняемых файлов из исходного кода позволяет убедиться, что система полностью совместима с оборудованием и настройками. Этот процесс осуществляется в несколько этапов и содержит определенную долю риска. Однако хорошо организованная структура FreeBSD помогает уменьшить риск и воспользоваться преимуществами, которые предоставляет такая методика обновления системы.

Процесс сборки системы называется **make world**. Он обеспечивает необходимые меры предосторожности, которые обычно недоступны при установке скомпилированных выполняемых файлов с компакт-диска. Например, если что-то не так с исходным кодом, это станет ясным на этапе компиляции, тем самым будет исключена ситуация, когда инсталлированная система в некоторый момент оказывается непригодной к использованию. Релиз представляет собой "фотографию" кода в состоянии, когда он достаточно стабилен, а процесс синхронизации функционально совпадает с синхронизацией исходного кода любой ветви разработки. Процесс сборки системы имеет удобный интерфейс, что побуждает администраторов использовать наиболее

современные версии исходного кода. Именно здесь и проявляется реальное преимущество системы: самые современные обновления защиты и исправления ошибок всегда доступны любому из нас. Все, что требуется для обновления определенной части системы, — это найти файлы с исходным кодом, скомпилировать их и затем инсталлировать. Таким образом, вы можете всегда идти в ногу со временем. Обновление с помощью этого метода отдельных компонентов (вместе с вопросами сетевой защиты) подробно обсуждается в главе 29. Здесь мы уделим внимание сборке всей системы.

Сборка всей системы (**world**) состоит из четырех основных шагов: сборка и инсталляция приложений основной системы (кроме ядра), а затем сборка и инсталляция ядра. Прежде чем приступить к сборке всей системы, необходимо произвести некоторую подготовку и принять меры предосторожности.

На что следует обратить внимание перед сборкой системы

Риск, сопровождающий сборку всей системы, нельзя недооценивать. На производственном сервере такой метод обновления системы лучше не применять. Проще подождать выхода компакт-диска с полным релизом и следовать стандартной процедуре инсталляции. Итак, рассмотрим проблемы, которые могут возникнуть при сборке всей системы.

Прежде всего, **make world** требует, по крайней мере, одной перезагрузки системы. Если время непрерывной работы системы — критический параметр (допустим, она должна постоянно находиться в режиме on-line), вам больше подойдет обновление до актуальной стабильной версии с последующим апдейтами, выходящими в промежутке между полными релизами.

Группа FreeBSD (FreeBSD Group) не может дать гарантий, что команда **make world** не приведет к полному уничтожению системы (фактически, таких гарантий не имеет никакое обновление). Реальная проблема заключается в том, что частая сборка операционной системы повышает риск ее сбоя. Данный подход(**make world**) представляет собой наилучший способ следить за последними исправлениями, однако не исключено использование системы с новыми, непроверенными свойствами или измененным кодом. Более того, код может не скомпилироваться в силу простого невезения: скажем, для синхронизации вы выбрали как раз тот момент, когда в систему был включен нестабильный код. Помните, что ветви STABLE и CURRENT часто изменяются и не обладают статусом релизов (так как могут содержать не согласованные фрагменты различных подсистем). Промежуточный код, выпускаемый между релизами следует рассматривать как бета-версию. Сборку всей системы можно производить лишь в том случае, если вы используете FreeBSD-машину в качестве рабочей станции или небольшого сервера. Другой причиной использования **make world** может быть исправление критической ошибки или появление нового свойства.

Если вы следите за версией STABLE или CURRENT, необходимо подписаться на список рассылки **freebsd-stable@freebsd.org** или **freebsd-current@freebsd.org**, соответственно. Эти списки представляют собой форумы для всех тех, кто следит за обсуждением проблем, новыми исправлениями ошибок, появлением новых свойств, а также за возможными проблемами при синхронизации исходного кода. Кто-либо из разработчиков может поместить в список **freebsd-stable@freebsd.org** предупреждение о том, что не следует пытаться собирать систему (**make world**) в течение одной-двух недель, пока не будет

проверено недавно добавленное новое свойство. Если вы не будете следить за списком рассылки, а подобное свойство окажется менее стабильным, чем ожидали его разработчики, вам придется работать с неустойчивой и незащищенной системой, а ведь как раз этого вы и пытаетесь избежать, работая с исходным кодом.

ПРИМЕЧАНИЕ

Оба списка рассылки следуют стандарту **majordomo**. Подписаться на любой из них можно, послав электронное сообщение:

```
subscribe freebsd-stable
```

по адресу **majordomo@freebsd.org**. Не посылайте запросы по подписке на адреса **freebsd-stable** или **freebsd-current**! Это одна из самых распространенных ошибок в Internet! Два этих адреса предназначены для контроля трафика списка рассылки, а не для подписки.

Вы получите запрос подтверждения, который необходимо переслать обратно, чтобы вас включили в список рассылки. Чтобы отписаться, проделайте то же самое, заменив слово **subscribe** на **unsubscribe** в теле сообщения.

Важно отметить, что, если даже и не запускать процесс **make world** регулярно, следить за изменениями в исходном коде ядра весьма желательно. Все это необходимо для того, чтобы синхронизировать исходный код на вашей машине (все, что находится в каталоге **/usr/src**) с текущим состоянием ветви STABLE или CURRENT. Такое согласование абсолютно не влияет на работу операционной системы. Затем можно собрать абсолютно новую систему (не заменяя ее предыдущую версию) или же, поскольку процесс сборки является иерархическим (как и файловая система, содержащая файлы с исходным кодом), просто перейти в любую точку в **/usr/src** и собрать отдельный компонент. Именно так происходит обновление системы, вызванное рекомендациями по защите системы (они обсуждаются в главе 29).

Главное, что нужно знать, — процесс обновления FreeBSD необратим. Деинсталлировать новую версию и вернуться к предыдущей невозможно. Поэтому наилучшее, что можно предпринять — следовать определенным мерам предосторожности.

ОТСЛЕЖИВАНИЕ ОШИБОК И ОТЧЕТЫ О ПРОБЛЕМАХ

База данных ошибок, обнаруженных во FreeBSD, доступна в оперативном режиме. Состояние всех обнаруженных ошибок и запросов на создание новых свойств можно просмотреть по адресу **http://www.FreeBSD.org/cgi/query-pr-summary.cgi**. Можно также обратиться к странице **http://www.freebsd.org** и следовать ссылке Bug Reports (Отчеты об ошибках).

Вы также имеете возможность сообщить о найденных ошибках. Именно так разработчики и узнают о них — от пользователей. Для этого применяется утилита **send-pr**. Запустите эту команду, и она откроет текстовый редактор с готовым шаблоном, содержащим информацию о вашей системе и несколько полей для ввода данных об обнаруженной ошибке. После сохранения файла и выхода из редактора отчет пересылается прямо базе данных GNATS и просматривается разработчиками.

Конечно же, прежде чем отправлять отчет об ошибке, убедитесь, что о ней еще не сообщили. Для этого следует обратиться по приведенному выше URL и произвести поиск по отчетам. Кроме того, желательно быть подписанным на один из двух обсуждавшихся ранее списков рассылки: **freebsd-stable** или **freebsd-current**. Неплохо до отправки отчета поместить сообщение в список рассылки, пообщаться с другими пользователями, которые также могли столкнуться с этой ошибкой. Скорее всего, об ошибке уже известно и, возможно, что уже найден временный способ ее обойти.

Задачи, выполняемые до сборки всей системы

Итак, вы решили заново собрать свою систему. Если вы собираетесь синхронизовать исходный код с одной из промежуточных ветвей или же хотите обновить систему до версии RELEASE, необходимо произвести определенные действия до того, как приступать к этому процессу. Рекомендуется следовать плану, который описан далее.

Самое главное (и это справедливо для любого обновления) — произвести резервное копирование системы. Многие читатели, без сомнения, игнорируют это предостережение и переходят к следующему шагу. Лишь трезво оценив те небольшие усилия, которых требует резервное копирование, вы поймете, что оно того стоит. Представьте, проводя обычное периодическое обновление, вы вдруг неожиданно обнаруживаете, что файловая система уничтожена и данные всех пользователей безвозвратно потеряны.

Возможности резервного копирования включают в себя носители на магнитных лентах, оптические диски (CD-R, DVD-RAM, DVD-R и т.д.), дополнительный жесткий диск, монтируемый лишь для зеркального копирования текущего содержимого основного диска, или вторая машина, синхронизуемая с основной посредством NFS или CVSup. (Об этих методиках рассказано в главе 20).

Обратите внимание на список рассылки для выбранной вами ветви. Убедитесь, что вы выбрали удачный момент для обновления. Если это не так, а вам крайне требуется определенное исправление, обновите лишь нужный вам компонент.

Следующий шаг в подготовке к сборке системы — синхронизация исходного кода. Настройка для этого производится лишь один раз. Далее этот процесс автоматизируется.

Синхронизация локального дерева исходного кода с деревом STABLE, CURRENT или RELEASE

Существует несколько способов синхронизации исходного кода. Методы, предлагаемые разработчиками FreeBSD, включают в себя CTM, Anonymous CVS и CVSup. Каждый из них имеет свои достоинства. О каждом из них можно узнать подробно на Web-сайте FreeBSD в разделе Synchronizing Your Source (Синхронизация исходного кода). Здесь мы остановимся на объяснении метода CVSup, как наиболее удобного и эффективного из всех возможных.

Использование CVSup

Настройка **CVSup** уже обсуждалась в главе 15. Здесь потребуется тот же самый процесс, поэтому, если вы не настроили **CVSup** с помощью порта **cvsupit**, обратитесь к описанию в главе 15. Следует отметить лишь несколько различий.

1. Когда **cvsupit** запрашивает используемую ветвь исходного кода (в первом меню), необходимо выбрать ветвь, соответствующую системе. Если используется исходный код STABLE, необходимо выбрать ветвь STABLE нужной версии. Если, например, *используется версия* 4.3-RELEASE, *нужно* выбрать **RELENG_4_3**. Если же вам требуется самый новый (и наименее проверенный) код, выберите пункт ".", — вершину дерева исходного кода.
2. В следующем меню необходимо просто нажать клавишу Enter для выбора всех доступных компонентов исходного кода. Не рекомендуется выбирать отдельные компоненты, такая гибкость предназначена для экспертов, которым она действительно требуется.

3. Для набора портов и документации примите значения по умолчанию. Эти деревья желательно синхронизировать одновременно. Обновление второго набора можно отключить, закомментировав его в файле **/etc/cvsupfile**.

Утилита **cvsupit** предложит произвести обновление — ответьте положительно. По окончании добавьте эту команду в ежедневный файл **periodic**, как рассказано в главе 15. В этом случае ваш исходный код ядра всегда будет не старше одного дня.

Обращайте внимание на содержимое каждого обновления, высылаемого вам электронной почтой! Если вы следите за версией STABLE или CURRENT, из сообщений вы узнаете, что именно изменилось.

Использование CVSup для синхронизации исходного кода RELEASE

CVSup — это замечательный способ синхронизации исходного кода как с версией RELEASE, так и с любой промежуточной версией. Для этого необходимо воспользоваться дескриптором нужной вам версии (как было рассказано ранее в этой главе). Однако cvsupit не позволяет сделать этого автоматически.

Лучше всего запустить утилиту **cvsupit** для создания файла **/etc/cvsupfile**. Затем выйти из **cvsupit** и открыть файл **/etc/cvsupfile** в текстовом редакторе. Файл похож на пример, приведенный в листинге 18.1.

Листинг 18.1 Пример файла /etc/cvsupfile, созданного утилитой cvsupit

```
*default    host=cvsup8.FreeBSD.org
*default   base=/usr
*default   prefix=/usr
*default   release=cvs
*default   tag=RELENG_4
*default   delete  use-rel-suffix

src-all
*default  tag=.
ports-all
doc-all
```

Теперь строку с дескриптором версии можно изменить. Например, получить исходные коды версии 4.3-RELEASE позволяет следующая строка:

```
*default     tag=RELENG_4_3_0_RELEASE
```

Теперь, запустив обновление CVSup, вы получите "замороженные" исходные файлы выбранной версии (в отличие от постоянно изменяющегося исходного кода ветвей STABLE и CURRENT). Если вы обновите их через неделю — ничего не изменится. Теперь можно запустить команду **make world** и заменить дескриптор версии в файле **/etc/cvsupfile** его предыдущим значением. Далее следует возобновить ежедневную синхронизацию кода, чтобы в вашей системе всегда были представлены исправления и обновления кода. Это более удобный метод обновления системы на важном сервере.

Решение возможных проблем

Существуют ситуации, когда процесс обновления CVSup не завершается успехом, а поскольку он не имеет аналога в других операционных системах, зачастую трудно установить, когда процесс прошел как запланировано, а когда — нет. Рассмотрим несколько распространенных типов проблем и их решение.

- CVSup не соединяется с выбранным сервером (Connection refused — в соединении отказано).

Попытайтесь выбрать другой сервер CVSup. Чем выше его номер (например, **cvsup8.freebsd.org**), тем менее он будет загружен. Кроме того, попытайтесь запустить CVSup в то время дня, когда трафик минимален (например, ночью).

- CVSup соединяется, но ничего не происходит.

  Возможно, вы работаете за брандмауэром или за маршрутизатором NAT, маскирующим ваш IP-адрес. CVSup работает с сетевыми конфигурациями NAT, однако некорректные настройки вызывают проблемы. Убедитесь, что брандмауэр пропускает трафик CVSup. При этом используется порт сервера 5999.

- CVSup полностью удалил каталог /usr/src!

  Так происходит, если указан неверный дескриптор ветви. Сервер просто возвращает содержимое дерева CVS для указанной ветви. Если последняя задана несуществующим значением, вы не получите ничего. Убедитесь, что указан правильный дескриптор ветви.

Текстовый файл UPDATING

Когда исходный код обновлен, текущий файл **UPDATING** находится в каталоге **/usr/src**. Он представляет собой бюллетень, содержащий наиболее важную информацию об обновлении версии системы, гораздо более свежую, чем любое печатное или электронное издание (эта книга — увы! — не исключение). В файле **UPDATING** в обратном хронологическом порядке (с указанием даты) перечислены изменения в процессе сборки, о которых необходимо знать пользователям. Следует ознакомиться со всеми сведениями от начала файла и до той даты, когда последний раз выполнялась команда **make world**. Наверняка вы найдете там то, о чем здесь не рассказано.

Следует помнить также, что каталог **/etc** не обновляется процессом **make world** автоматически. Конфигурационные файлы не перезаписываются новыми. С одной стороны, это значит, что специальная конфигурация системы не будет утрачена, с другой — к ней потребуется добавить что-то новое или удалить устаревшее. Некоторые файлы придется объединить вручную. В файле **UPDATING**, как правило, содержится информация именно такого рода. К счастью, имеется довольно тонкая утилита под названием **mergemaster**, которая позволяет с легкостью объединить новые файлы с содержимым каталога **/etc**. О программе **mergemaster** будет рассказано далее, а вначале мы обсудим процесс **make world**.

Объединение файлов /etc/group и /etc/passwd с соответствующими новыми версиями

Большинство файлов в каталоге **/etc** не влияют на возможность сборки системы. Однако иногда добавляются новые службы, которые следует устанавливать с правами владения определенного пользователя и группы. Помните, что **make world** — это не приложение для инсталляции со встроенным обновлением конфигураций. Это процесс перекомпиляции и установки новой версии операционной системы поверх старой. Иначе говоря, он не станет создавать новых пользователей или групп. Избежать недоразумений позволяет добавление новых записей к уже существующим в файле **/etc/group**.

Новая версия этого файла находится в **/usr/src/etc/group**. Обычно она содержит около 20 строк. Откройте файл **/etc/group** в другом терминальном окне и проверьте, есть ли в **/usr/src/etc/group** записи, отсутствующие в **/etc/group**. Если да, просто скопируйте их. Идентификаторы групп новых записей, обычно, меньше 100.

Один из простых способов сравнения двух файлов — команда **diff**. Пример приведен в листинге 18.2.

Листинг 18.2 Сравнение файлов /etc/group и /usr/src/etc/group с помощью diff

```
# diff -c /etc/group /usr/src/etc/group | less
*** /etc/group    Wed May    2 09:57:10 2001
--- /usr/src/etc/group    Fri Aug 27 16:23:41 1999
***************
*** 8,24 ****
  bin:*:7:
  news:*:8:
  man:*:9:
-  wheel:*:10:root,frank,joe
  games:*:13:
  uucp:*:66:
  xten:*:67:xten
  dialer:*:68:
+ network:*:69:
- mysql:*:88:
- users:*:100:
  nogroup:*:65533:
  nobody:*:65534:
```

Группы, существующие только в файле **/etc/group**, показаны знаком -, а новые группы в **/usr/src/etc/group** помечены знаком +. В данном примере в файл **/etc/group** нужно скопировать группу **network**.

Важно, чтобы GID новой группы оставался таким же, как в новой версии файла **group**. В тех редких случаях, когда идентификатор группы не совпадает, необходимо заменить его в файле **/etc/group** на новое значение. Это может привести к тому, что существующие файлы в системе (владельцем которых была эта группа) не будут ассоциироваться ни с какой группой. Поиск этих файлов в системе можно осуществить с помощью команды **find**:

```
# find / -group <GID> -print
```

Изменить права доступа можно с помощью команды **chmod**, как показано в главе 10.

Еще менее вероятно, что несовпадения возникнут в пользовательской базе данных. Исходные файлы не включают **passwd**, только **/usr/src/etc/master.passwd**. Он содержит еще меньше строк, чем файл **group**. Сравните его с **/etc/master.passwd** и при необходимости добавьте новых пользователей командой **adduser**.

Объединение файла /etc/make.conf с его новой версией

Файлы **make.conf** являются глобальными конфигурационными файлами, которые управляют всеми действиями команды **make**, включая и **make world**. Новая инсталляция FreeBSD не содержит файла **/etc/make.conf**, но (аналогично файлам **/etc/rc.conf** и **/etc/defaults/rc.conf**) существует **/etc/defaults/make.conf**, в который включены все возможные *настройки по умолчанию. Обычно можно все оставить как есть. В некоторых случаях* работу системы можно ускорить, правильно настроив несколько из них.

В справочнике по FreeBSD (FreeBSD handbook) рекомендуют включить следующие строки (скопировав их из **/etc/defaults/make.conf** в **/etc/make.conf**):

```
CFLAGS= -O -pipe
NOPROFILE=      true        # Avoid compiling profiled libraries — Не
                            # компилировать профилированные библиотеки
```

Вы можете поэкспериментировать и с другими опциями, однако их описание выходит за пределы данной главы и книги в целом.

Сборка системы из исходных файлов

Итак, все готово к сборке системы. Этот процесс требует определенных шагов, о которых и пойдет речь далее.

Процесс **make world** состоит из четырех основных этапов компиляции, перечисленных в табл. 18.3. Их можно сократить до двух, если вы достаточно уверены в себе.

Таблица 18.3 Этапы, из которых состоит процесс make world

Набор команд из четырех шагов	*Набор команд из двух шагов*	*Значение*
make buildworld make installworld	make world	Компилирует и устанавливает все, кроме ядра
make buildkernel make installkernel	make kernel	Компилирует и устанавливает ядро

Командами **make world** и **make kernel** следует пользоваться лишь опытным администраторам, которые точно знают, какие действия при этом выполняются. Остальным пользователям рекомендуется придерживаться схемы из четырех команд.

Очистка каталога /usr/obj

Этот шаг не требуется в том случае, если команда **make world** выполняется в первый раз. Если это не так, из каталога **/usr/obj** необходимо удалить все файлы. Это ускорит процесс и предотвратит конфликты, возникающие, когда система сталкивается с файлами, которые нельзя перезаписать.

В каталоге **/usr/obj** хранятся объектные файлы (скомпилированные компоненты), используемые до сборки. Их удаление не всегда можно выполнить одной командой **rm -rf**. При сборке всей системы создаются файлы с установленным флагом **schg** (системно-неизменяемый), которые не может удалить даже пользователь **root**. Эта мера предосторожности направлена на устранение таких случайностей, как например, **rm -rf \.***. Чтобы корректно очистить каталог, выполните следующий набор команд:

```
# cd /usr/obj
# chflags -R noschg *
# rm -rf *
```

Эти команды не будут выполнены, если система работает на уровне защиты **securelevel** 1 или выше (эта настройка защиты выбирается во время инсталляции системы или устанавливается в файле **/etc/rc.conf**, см. **man securelevel** и главу 29). В таком случае система не позволит отключить флажки **schg**. Чтобы выполнить этот шаг, необходимо загрузить систему в однопользовательском режиме.

Запуск записи в log-файл

Вывод команды **make world** необходимо сохранить в log-файле. Если сборка произойдет некорректно, в нем можно будет найти необходимые данные и (если потребуется) направить их в список рассылки. Хотя ошибка, как правило, немедленно прерывает компиляцию, и соответствующие данные находятся в буфере прокрутки экрана, удобнее все же иметь файл с записью о полном процессе.

Для этого применяется команда **script**. Фактически, она запускает "интерпретатор внутри интерпретатора", перехватывая весь вывод в файл. Для выхода из **script** нужно воспользоваться командой **exit**.

```
# script ~/buildworld.out
Script started, output file is ~/buildworld.out
# make buildworld
...
# exit
Script done, output file is ~/buildworld.txt
```

Аргументом команды является файл, в который направляется вывод. В имени файла желательно отразить номер попытки сборки. Тогда для каждого процесса **make world** система сохранит отдельный log-файл.

ПРЕДУПРЕЖДЕНИЕ

Не размещайте log-файл в каталоге **/tmp**! Файлы из этого каталога удаляются при каждой загрузке системы, и когда новая система начнет работать, файл будет удален.

make buildworld

Перейдите в каталог **/usr/src**, еще раз проверьте все и начните выполнение первого этапа **make**:

```
# make buildworld
```

Процесс **buildworld** занимает один-два часа, в зависимости от оборудования. Его можно ускорить с помощью опции **-j**, запускающей одновременно несколько процессов:

```
# make -j4 buildworld
```

Это полезно даже в системе с одним процессором. На машине с несколькими процессорами можно добиться еще более высокой производительности (используйте значения порядка 10).

Процесс сборки рекурсивно проходит по каталогу **/usr/src** примерно в алфавитном порядке. Информацию о его продвижении можно получить из каталога **/usr/obj**. Когда вы увидите, что компилируется нечто вроде **/usr/src/usr.sbin**, это сигнал — вы близки к завершению.

Решение проблем

Подобно CVSup, процесс сборки полной операционной системы не имеет аналогов в настольных системах, поэтому, вполне вероятно, вы столкнетесь с некоторыми трудностями. Рассмотрим наиболее общие из них.

- Компилятор завершает работу с ошибкой "signal 11".

  Сигнал 11 сообщает об ошибке адресации памяти компилятора. Причиной, как правило, служит проблема с оборудованием. Проверьте, не завышена ли тактовая частота вашего процессора. Это часто вызывает проблемы, когда FreeBSD выполняет набор ресурсоемких операций при компиляции. Если это не так, проверьте оперативную память и другое оборудование.

- Компилятор завершает работу с несколькими строками "***** Error code 1 *****".

  Это основная ошибка, при которой процесс компиляции завершается неудачей в определенной точке. Процесс **buildworld** не может быть продолжен, если любая

из его частей завершилась неудачно, поэтому несколько последних строк вывода могут оказаться полезными при выяснении причин.

Прежде всего, убедитесь, что до компиляции из каталога **/usr/obj** удалены все файлы. Объектные файлы от предыдущих сборок могут не только увеличить время компиляции, но и вызвать различные проблемы. Очистите каталог **/usr/obj** и попытайтесь снова.

Если это не помогает, произведите поиск в архивах на сайте **http://www.freebsd.org** (в списке рассылки **stable/freebsd-stable** или **current/freebsd-current**) по значащим словам из вывода компилятора. Если вы ничего не найдете, поместите вопрос в соответствующий список рассылки, включив в него вывод команды **uname -a**, время последнего обновления CVSup и несколько значащих строк вывода компилятора.

Обновление ядра

Итак, кроме ядра, все собрано и готово к инсталляции. Пришло время собрать и установить новое ядро.

Обновление ядра GENERIC

Если система работает с ядром GENERIC (за сведениями о нем обратитесь к главе 17), процесс достаточно прост:

```
# make buildkernel
# make installkernel
```

или даже:

```
# make kernel
```

После этого новое ядро установлено в **/kernel** (**/boot/kernel** в FreeBSD 5.0), а старое переименовано в **/kernel.old** (**/boot/kernel.old** в FreeBSD 5.0).

Обновление специального ядра

Если система работает со специальным ядром, процесс его перекомпиляции выглядит несколько сложнее. Можно попытаться собрать новую версию специального ядра и загрузиться с ним, однако правильная (т.е. безопасная) методика предполагает предварительную сборку ядра GENERIC, которым можно воспользоваться для загрузки системы при возникновении проблем. Дополнительной мерой предосторожности служит также сохранение старого ядра.

В каталоге **/usr/src** запустите следующую команду:

```
# make buildkernel KERNCONF=GENERIC
```

Это первое резервное ядро — ядро GENERIC, скомпилированное из нового исходного кода. Предыдущее ядро **/boot/kernel** необходимо скопировать с именем **/boot/kernel.prev** (или любым другим именем, указывающим, что это ядро работоспособно и система работала с ним ранее). А затем установить новое ядро:

```
# make installkernel KERNCONF=GENERIC
```

Затем скопировать его как **kernel.GENERIC**:

```
# cp /boot/kernel /boot/kernel.GENERIC
```

И лишь после этого попытаться собрать специальное ядро из нового исходного кода:

```
# make buildkernel KERNCONF=CUSTOM
```

Естественно, слово **CUSTOM** необходимо заменить именем используемого специального ядра. Теперь его можно установить как **/boot/kernel**:

```
# make installkernel KERNCONF=CUSTOM
```

Итак, ядра, которые должны быть подготовлены, приведены в табл. 18.4.

Таблица 18.4 Новые ядра (FreeBSD 4.4)

Ядро	*Описание*
/kernel	Новое специальное ядро
/kernel.GENERIC	Новое ядро **GENERIC**
/kernel.old	Старое работоспособное ядро

После перезагрузки, если загрузка специального ядра закончится неудачно, попытайтесь загрузить новое ядро **GENERIC** (как описано в главе 11). Если и оно не сможет загрузиться корректно, вернитесь к исходному ядру.

Нештатные ситуации

Сборка и установка ядра представляет собой самую рискованную фазу процесса **make world**. Меры предосторожности, принимаемые системой для защиты ядра в течение этого процесса, еще больше запутывают процедуру и не слишком дружественны к пользователю. Вот несколько причин неудачной сборки и их решения.

- Компилятор неудачно завершает работу.

  Исходный код ядра может не компилироваться корректно по тем же причинам, что и вся система (об этом уже говорилось ранее). Если вы не сможете решить проблему, обратитесь к спискам рассылки.

- Система не позволяет установить новое ядро!

  Если система работает на уровне защиты **securelevel** 1 или выше, команда **make installkernel** не способна отключить флажок **schg** текущего ядра. Для этого необходимо перезагрузиться в однопользовательском режиме и корректно завершить каждый шаг процесса. Помните, что при загрузке в этом режиме (используя **boot -s** в приглашении загрузчика **loader**), никакие файловые системы не монтируются автоматически. Необходимо выполнить команду **mount -a** и лишь затем перейти в каталог **/usr/src** и запустить процесс **make installkernel**.

make installworld

По завершении сборки ядер необходимо перезагрузиться в однопользовательском режиме, предотвращающем запуск процессов (выполняемых пользователями или демонами), способных изменить обновляемые файлы. Изменения такого рода могут привести к серьезной неустойчивости системы. Однопользовательский режим позволяет также немного ускорить процесс.

Воспользуйтесь командой **reboot**. По достижении фазы загрузчика **loader** (когда происходит отсчет секунд) нажмите клавишу для перехода к приглашению и введите команду **boot -s**. Система загрузится, позволяя убедиться, что новое ядро работает. После этого можно продолжить инсталляцию.

ПРИМЕЧАНИЕ

Если система не загрузится (она может аварийно завершить работу, выдать сообщения об ошибках и т.д.), перезагрузитесь снова и попытайтесь запустить в однопользовательском режиме резервное ядро. Пытайтесь до тех пор, пока не достигнете ядра, которое использовалось ранее (оно должно загружаться без проблем, ведь единственное, что изменилось в системе, — это ядро). На этой стадии следует отложить процесс **make world** и разобраться с новым ядром. Скорее всего, неудачная загрузка ядра в такой фазе связана с временными проблемами в исходном коде, которые будут решены в ближайшем будущем.

В однопользовательском режиме вы по умолчанию обладаете правами пользователя **root**. Перейдите в каталог **/usr/src** и запустите финальную, наиболее важную часть процесса **make world**. Установка системных выполняемых файлов занимает меньше времени, чем их компиляция, но она содержит не меньше возможностей для ошибок. Обратите внимание, что теперь ошибки возникают в частично измененной системе. Обязательно сохраните вывод в log-файле.

```
# make installworld
```

По завершении этого шага бинарные файлы и ядро должны быть совместимы. Запустите такие утилиты, как **ps** и **top**, чтобы убедиться, что все работает должным образом. Если ядро и двоичные файлы собраны не из одних исходных кодов, утилиты будут выдавать ошибки. Если же ошибок нет, это значит, что инсталляция системы прошла успешно.

Возможные трудности

Если на этапе **make installworld** происходит что-то незапланированное, это потенциально опасно, поскольку новая система установлена не полностью, часть ее совместима с новым ядром, а часть — нет. К счастью, на этой стадии сбои происходят редко.

- Инсталляция успешно не завершается, докладывая о проблеме с правами владения или доступа.

Это одна из причин несинхронизации файлов **/etc/group** и **/etc/master.passwd** с их новыми версиями до установки системы. Убедитесь, что все новые пользователи и группы включены в эти файлы, а затем попытайтесь снова.

Использование mergemaster для проверки измененных конфигурационных файлов

Остался один шаг: включить в иерархию **/etc** (и другие области, например, **/var/log** и **/usr/share**) новые версии конфигурационных файлов. Как уже обсуждалось ранее, процесс **make world** не вносит изменений в **/etc**, чтобы специально настроенные конфигурации не были потеряны. В ранних версиях системы объединение файлов из каталога **/etc** с их новыми версиями представляло собой сложный процесс, зачастую сопровождавшийся ошибками. Теперь утилита **mergemaster**, стандартное средство FreeBSD, не только упрощает эту процедуру, но и делает ее безопаснее.

mergemaster принимает много различных предосторожностей, это надежная утилита. Тем не менее, поскольку всегда существует риск повредить конфигурационные файлы, необходимо создать резервную копию каталога **/etc**. Воспользуйтесь командой:

```
# cp -Rp /etc /etc.old
```

Обычно, **mergemaster** не требует никаких опций, утилита действует правильно по умолчанию. Для первого раза рекомендуется указать опцию **-v** (информативный режим вывода) и **-c** (контекстное сравнение **diff**, а не глобальное).

```
# mergemaster -cv
```

Вначале **mergemaster** создает временный корневой каталог и устанавливает в него все необходимые файлы из исходных кодов (их нужно не просто скопировать, а правильно объединить с уже существующими версиями). По умолчанию, используется каталог **/var/tmp/temproot**. Программа показывает листинг файлов, находящихся в **/etc** и отсутствующих во временном каталоге (обычно, это файлы, добавленные администратором, их не затрагивает работа утилиты). Затем она сравнивает все файлы из **/etc** и некоторых других каталогов с файлами из **/var/tmp/temproot**. При нахождении несовпадений на экране отображается вывод команды **diff**, при этом используется утилита постраничного просмотра, установленная в переменной **PAGER** (или **more** по умолчанию). При прокрутке к нижней части вывода **diff** программа запрашивает, какие действия произвести с новым файлом. См. листинг 18.3.

Листинг 18.3 Возможности по объединению файлов в mergemaster

```
***************
***  321,326  ****
---  327,333  ----
    kern_securelevel="-1"    #  range:  -1..3  ;  '-1'  is  the  most  insecure
    update_motd="YES"        #  update  version  info  in  /etc/motd  (or  NO)
    start_vinum=""           #  set  to  YES  to  start  vinum
+   unaligned_print="YES"    #  print  unaligned  access  warnings  on  the
alpha  (or  NO).

      ##################################################################
    ###  Define  source_rc_confs,  the  mechanism  used  by  /etc/rc.*  ##

    Use  'd'  to  delete  the  temporary  ./etc/defaults/rc.conf
    Use  'i'  to  install  the  temporary  ./etc/defaults/rc.conf
    Use  'm'  to  merge  the  old  and  new  versions
    Use  'v'  to  view  to  differences  between  the  old  and  new  versions
again

    Default  is  to  leave  the  temporary  file  to  deal  with  by  hand
```

Как легко заметить, по умолчанию утилита не производит никаких действий, оставляя файл в каталоге **/var/tmp/temproot** для последующего внесения изменений вручную. Таким образом, **mergemaster** — очень надежная утилита.

Если выбрать команду '**m**' для объединения двух файлов, программа перейдет в режим **sdiff**. В нем построчно показаны две версии измененного файла. Это позволяет выбирать информацию из старой (слева) или новой (справа) версии.

```
        ***  Type  h  at  the  sdiff  prompt  (%)  to  get  usage  help
pccard_beep="1"    #  pccard  beep  |  pccard_beep="2"      #  pccard  beep
%
```

Иногда строки выглядит одинаково, поскольку различие просто не поместилось на экране. Например, в строках с номером версии отличие, как правило, находится в конце строки. Для этого можно воспользоваться опцией **-w** устанавливающей ширину экрана:

```
# mergemaster -cv -w 120
```

О доступных командах **sdiff** позволяет узнать команда **'h'**, набранная в приглашении. По окончании редактирования файлов на экране вновь отображается **mergemaster**. Утилита позволяет повторно просмотреть изменения (и даже внести коррективы). Затем новый файл размещается в нужном каталоге.

И наконец, **mergemaster** запрашивает подтверждение на удаление файлов, оставшихся в каталоге **/var/tmp/temproot**. Если вы оставили несколько новых файлов, не объединив их со старыми версиями, выберите **"no"**. Тогда после выхода из программы вы сможете перейти в каталог **/var/tmp/temproot** и объединить оставшиеся файлы вручную.

Нештатные ситуации

Если утилита **mergemaster** выполняет неверные действия, они, к счастью, не деструктивны для системы, особенно, если создана резервная копия каталога **/etc**. Вот несколько распространенных ошибок.

- Я случайно перезаписал важный файл в **/etc** новой версией!

  Не беспокойтесь. Если вы создали резервную копию (например, **/etc.old**), выйдите из **mergemaster**, нажав Ctrl+C, и скопируйте резервный файл в каталог **/etc**.

- Mergemaster случайно удалил все файлы, остававшиеся в **/var/tmp/temproot**!

Запустите **mergemaster** снова и просто игнорируйте файлы, которые уже были заменены или объединены. В последнем запросе на удаление **/var/tmp/temproot** выберите **"no"**.

Перезагрузка системы после обновления

Теперь зададим себе ряд вопросов:

- Произвел ли я синхронизацию с последней версией исходного кода?
- Выполнил ли я **buildworld**?
- Скомпилировал ли я новое ядро?
- Загрузилось ли новое ядро?
- Выполнил ли я **installworld**?
- Работают ли такие утилиты, как **ps** и **top**?
- Правильно ли прошло объединение файлов из **/etc** с их новыми версиями?

Если ответом на все вопросы будет "да", вы готовы к перезагрузке системы.

```
# reboot
```

Когда система возвратится в многопользовательский режим, проверьте работу утилит **ps** и **top**. Все ли синхронизовано? В заключение, запустите **uname -a** и просмотрите номер версии:

```
# uname -a
```

FreeBSD stripes.somewhere.com 4.3-STABLE FreeBSD 4.3-STABLE #0: Wed Jan 31 18:45:43 PST 2001 frank@somewhere.com:/usr/src/sys/compile/CUSTOM i386

Если все произошло именно так, примите мои поздравления! Вы успешно выполнили процесс **make world**.

19 ГЛАВА

О жестких дисках и файловых системах

- Режимы доступа IDE/ATA
- Диски SCSI
- Геометрия жесткого диска
- Разбиение жесткого диска на разделы
- Создание дисковых меток
- Подготовка файловой системы к использованию

В этой главе мы поговорим о дисках. В отличие от Windows (где диску BIOS автоматически присваивается номер) или Macintosh (где новый диск просто появляется на рабочем столе), UNIX-подобные системы требуют более глубокого знания геометрии, разделов, режимов доступа и других особенностей работы с дисками. И FreeBSD не является исключением.

Как было рассказано в главе 9, иерархическая структура файловой системы UNIX обеспечивает более высокую гибкость при работе с файлами, чем традиционные настольные операционные системы. За эту гибкость приходится платить: для добавления нового диска в систему его необходимо правильно установить физически, правильно разбить на слайсы (slices) и разделы, правильно пометить их, а затем смонтировать в нужной точке. Это требует больше шагов, чем в настольных системах, однако стандартизация оборудования упрощает этот процесс. А с помощью утилиты **sysinstall** процессом установки нового жесткого диска управлять совсем несложно.

Режимы доступа IDE/ATA

Наиболее распространенный тип жесткого диска для ПК использует интерфейс IDE (Integrated Drive Electronics — устройство с интегрированной электроникой). Это название подразумевает, что микросхемы контроллера, управляющие диском, интегрированы с ним в одно устройство, не требующее отдельного адаптера (как, например, SCSI). Официально этот интерфейс называется ATA (Advanced Technology Attachment — высокотехнологичное устройство). Названия ATA и IDE зачастую взаимозаменяемы. В этой главе обсуждаются дисковые устройства IDE/ATA и SCSI. Скорее всего, вы используете IDE-диски. Они относительно недорогие и повсеместно распространены, однако им присущи определенные недостатки. Дополнительные затраты при покупке SCSI-устройств компенсируются их более надежной работой.

При выборе оборудования для системы пользователи чаще всего выбирают жесткие диски IDE (если только машина не будет использоваться как высокопроизводительный сервер). Интерфейс IDE обеспечивает достаточно высокую скорость и является стандартом, поддерживаемым всеми материнскими платами архитектуры x86. При выборе диска необходимо обратить внимание на режим доступа, поддерживаемый диском, материнской платой и системой FreeBSD.

Режимы PIO

Режимы PIO (Programmed I/O — программируемый ввод-вывод) представляют собой исходный стандарт передачи данных интерфейса ATA. Они существуют до сих пор, поскольку режим PIO является составной частью BIOS и не требует дополнительной поддержки от операционной системы. В настоящее время почти все системы поддерживают режимы DMA и Ultra DMA. Если не используется старое оборудование, режим PIO не применяется.

Пять стандартных режимов передачи данных PIO и соответствующие им скорости приведены в табл. 19.1.

Таблица 19.1 Режимы передачи данных PIO

Режим PIO	*Максимальная скорость передачи данных, Мб/с*	*Стандарт*
0	3,3	ATA
1	5,2	ATA

Режим PIO	Максимальная скорость передачи данных, Мб/с	Стандарт
2	8,3	ATA
3	11,1	ATA-2
4	16,7	ATA-2

ПРИМЕЧАНИЕ

Большинство современных материнских плат используют для работы с оборудованием шину PCI. Если вам приходится иметь дело со старыми системами, где основной шиной является ISA, максимальным возможным режимом будет PIO Mode 2. Любой более скоростной режим превысит возможности самой шины.

Режимы DMA

Главный недостаток интерфейса с режимом PIO заключается в том, что он требует значительных ресурсов процессора для обработки потока данных. Это было одной из причин использования SCSI-дисков, так как последние используют независимый контроллер, снижая тем самым загрузку процессора. В начале 90-х появились режимы DMA. Они перечислены в табл. 19.2. Эти режимы позволяют диску взаимодействовать с оперативной памятью напрямую, минуя процессор. Естественно, это требует изменений в электронике самого диска и поддержки операционной системы. В наши дни применение этих режимов стало поистине всеобъемлющим.

Таблица 19.2 Режимы передачи данных DMA

Режим DMA	Максимальная скорость передачи данных, Мб/с	Стандарт
Single Word Mode 0	2,1	ATA
Single Word Mode 1	4,2	ATA
Single Word Mode 2	8,3	ATA
Multiword Mode 0	4,2	ATA
Multiword Mode 1	13,3	ATA-2
Multiword Mode 2	16,7	ATA-2

В период расцвета простых режимов DMA вы, возможно, даже не слышали о них. Это связано с тем, что скорость передачи данных сравнима с режимами PIO. Хотя DMA-диски меньше использовали ресурсы процессора, это преимущество практически не проявлялось из-за плохой поддержки DMA в таких операционных системах, как Windows 95. А режим PIO уже был встроенным, поэтому производители операционных систем не спешили отказываться от него.

Режим Ultra DMA

Ситуация коренным образом изменилась, когда появились режимы Ultra DMA. Эти улучшенные режимы DMA теперь являются общим стандартом для индустрии. А по соотношению цена/производительность они даже обгоняют интерфейс SCSI, оставаясь при этом дешевыми.

Режимы Ultra DMA значительно повысили скорость передачи данных по сравнению со стандартными DMA. Это стало возможно благодаря тому, что в них для пе-

редачи данных используется передний и задний фронт сигнала. Это значит, что при той же частоте скорость передачи данных удваивается до 33 Мб/с. Скорости в режимах Ultra DMA показаны в табл. 19.3. Рост связан также с переходом на новый стандарт кабеля IDE: 80 контактов вместо 40.

Таблица 19.3 Режимы передачи данных Ultra DMA

Режим Ultra DMA	*Максимальная скорость передачи данных, Мб/с*	*Стандарт*
0	16,7	ATA/ATAPI-4
1	25,0	ATA/ATAPI-4
2	33,3	ATA/ATAPI-4
3	44,4	ATA/ATAPI-5
4	66,7	ATA/ATAPI-5
5	100,0	ATA/ATAPI-6

Любое оборудование, выпущенное после 1998 года, поддерживает режим UDMA, поэтому обратите внимание на следующие четыре компонента вашего ПК:

- Жесткий диск, поддерживающий режим Ultra DMA;
- 80-контактный кабель IDE;
- Материнская плата (или контроллер IDE), поддерживающая режим Ultra DMA;
- Поддержка режима Ultra DMA в операционной системе (или BIOS).

Если один из элементов отсутствует, система все равно сможет работать, однако передача данных будет происходить с меньшей скоростью, скорее всего, в каком-либо из режимов PIO, автоматически поддерживаемом BIOS. С материнскими платами и контроллерами, не поддерживающими UDMA, возможны определенные проблемы, такие как неустойчивость или "зависание" системы. Их можно решить или обновив BIOS (позволив контроллеру поддерживать UDMA), или запретив режим UDMA в самом диске, с помощью утилит, поставляемых его производителем.

FreeBSD полностью поддерживает режимы Ultra DMA. Проверить, все ли нужные для поддержки элементы присутствуют в системе, позволяет поиск устройств во время загрузки. Просмотреть сообщения можно, набрав команду **dmesg**:

```
ad0: 6194MB <HITACHI_DK239A-65> [13424/15/63] at ata0-master using
UDMA33
```

Если система не определила возможность использования UDMA и жесткий диск новее, чем остальные компоненты компьютера, следует найти утилиты управления диском (поставляемые производителем), загрузить с дискеты систему MS-DOS и запретить использование UDMA. В этом случае скорость передачи данных будет ниже, однако система не будет сбоить и зависать.

Диски SCSI

Если вам не требуются IDE-диски или вы собираетесь подключить к системе внешние диски, альтернативой может стать интерфейс SCSI. На первый взгляд режимы SCSI (приведенные в табл. 19.4) кажутся не менее запутанными, чем режимы IDE.

Таблица 19.4 Режимы передачи данных SCSI

Стандарт	Режим передачи данных	Скорость передачи данных, Мб/с	Максимальная длина кабеля, м	Число устройств на шине, шт	Число контактов соединительного кабеля, шт
SCSI-1	SCSI	5	6	8	50
SCSI-2	Fast SCSI	10	3	8	50
	Wide SCSI	10	6	16	68
	Fast Wide SCSI	20	3	16	68
SCSI-3	Ultra SCSI	20	1,5-3	4-8	50
	Wide Ultra SCSI	40	1.5-3	4-16	68
	Ultra2 SCSI	40	12	8	50
	Wide Ultra2 SCSI	80	12	16	68
	Ultra3 SCSI	160	12	16	68
	Ultra160 SCSI	160	12	16	68
	Ultra320 SCSI	320	12	16	68

Несмотря на сложные правила именования, интерфейс SCSI ведет себя достаточно предсказуемо и корректно. Каждый последующий стандарт в два раза быстрее предыдущего. Используется лишь два типа разъемов соединительных кабелей. Это значит, что даже самый новый SCSI-контроллер сумеет распознать SCSI-устройство, соответствующее более раннему стандарту, и сможет с ним работать.

Цепочка устройств SCSI может содержать не больше устройств, чем задано стандартом, обычно 8 или 16. Вспомните, что структура интерфейса IDE/ATA позволяет работать лишь с четырьмя устройствами (два контроллера по два устройства). Интересно, что порядок устройств в цепочке не имеет значения. Здесь нет таких понятий, как ведущее или ведомое устройство. Существует только контроллер, или хост-адаптер (он сам является SCSI-устройством и получает свой идентификатор), и остальные устройства.

SCSI имеет немало преимуществ также и при использовании внешних дисков. Интерфейс IDE/ATA не обеспечивает удобного способа подключения внешнего диска к внутренней шине. В некоторых случаях можно воспользоваться параллельным (принтерным) портом, например, для носителей Iomega ZIP, но это чрезвычайно медленный интерфейс, неприменимый для большинства задач. SCSI был единственным интерфейсом для внешних дисков, пока не появился стандарт FireWire (IEEE 1394).

Наибольшей проблемой применения SCSI-устройств является... цена. SCSI-диски обычно дороже аналогичных IDE-дисков в 1,5–2 раза. К этому следует добавить цену SCSI-адаптера, который не поставляется с материнской платой (только некоторые материнские платы сейчас поставляются со встроенным SCSI-адаптером, хотя, нужно отметить, число их растет). Их также нельзя назвать дешевыми.

Следует отметить, что со SCSI связано множество вопросов, выходящих за рамки этой книги: терминаторы, SCSI BIOS, настройка идентификаторов устройств (посредством джамперов). Представляет интерес и то, что происходит, когда в одной цепи присутствуют устройства разных стандартов. Более подробные сведения о SCSI-устройствах можно найти, например, на Web-сайте PC Guide по адресу **http://www.pcguide.com**.

Геометрия жесткого диска

Как администратор, вы будете сталкиваться с этими понятиями при каждом запуске утилиты **fdisk** (или программы с более дружественным интерфейсом). Одним из преимуществ FreeBSD является способность эффективно работать даже на старом оборудовании, непригодном для использования, скажем, с современными версиями Windows. Материнской платы с процессором Pentium 166 МГц вполне достаточно для поддержки FreeBSD. Но здесь есть ловушка: попытайтесь подключить новый 70-гигабайтный жесткий IDE-диск — и система обнаружит не более 8 Гб. Вот здесь-то и вступает в игру геометрия жесткого диска. Чтобы использовать весь объем диска, нужно предпринять определенные действия.

В те времена, когда физический размер диска делали настолько большим, насколько позволяла цена, несмотря на будущие сложности с масштабируемостью, емкость жестких дисков IDE вычислялась как функция от их физической геометрии. Четыре параметра — число головок, цилиндров, секторов и байт в секторе — полностью описывали диск и его емкость (см. рис. 19.1), то есть объем данных, которые он мог хранить. Жесткие диски и теперь имеют эту внутреннюю геометрию, однако сейчас она не так важна.

Рисунок 19.1
Геометрия жесткого диска

Диск состоит из нескольких пластин, вращающихся вокруг общей оси. Данные сохраняются с обеих сторон пластины и читаются одной или несколькими головками с каждой стороны. Таким образом, диск с четырьмя пластинами и с одной головкой на каждую сторону имеет в сумме 8 головок. Каждая пластина разделена на концентрические кольца — дорожки. Дорожки с одинаковым номером на всех поверхностях образуют цилиндр. Дорожки разделены на секторы (обычно на 64, из них используется 63). Каждый сектор хранит определенное число байтов (обычно, 512).

Каждый диск сопровождается информацией о его параметрах. Подсчитать размер диска можно их простым перемножением. Диск, который использовался в примере работы утилиты **fdisk** в главе 9, содержит следующую информацию:

```
# fdisk /dev/ad1
******* Working on device /dev/ad1 *******
parameters extracted from in-core disklabel are:
cylinders=1247 heads=255 sectors/track=63 (16065 blks/cyl)
Media sector size is 512
```

Умножение этих чисел (кроме значения блок/цилиндр) дает следующий результат:

```
1247 cylinders x 255 heads x 63 sectors/track x 512 bytes/sector =
10.2GB
```

Минутку! 255 головок? Чтобы это значило? Прежде всего, число должно быть четным. Однако, в любом случае, это означает 64 пластины — количество, физически невозможное для диска высотой в сантиметр. Здесь мы столкнулись с виртуальной геометрией диска, которая стала возможной благодаря двум замечательным решени-

ям, найденным индустрией жестких дисков. Они позволили обойти ограничение физических размеров, вызванное исходной разработкой интерфейса BIOS. Итак, рассмотрим эти решения (режимы LBA и Extended INT13).

Режим LBA и предел 528 Мб

Во времена флоппи-дисков никто не мог даже предположить, что кому-нибудь понадобится жесткий диск такого гигантского объема, как 528 Мб. Поэтому это число стало тем ограничением, которое накладывало количество бит, используемое BIOS ПК, на описание геометрических размеров диска. Значения приведены в табл. 19.5.

Таблица 19.5 Ограничения, накладываемые на геометрию жесткого диска

Параметр	*Количество бит*	*Максимальное значение*
Цилиндры	10	$2^{10}-1 = 1023$
Головки	8	$2^{8}-1 = 255$
Сектор/Дорожка	6	$2^{6}-1 = 63$

Таким образом, диск мог иметь до 255 головок и не более 1023 цилиндров. Поскольку количество байтов в секторе и число секторов на дорожку — величины почти фиксированные (площадь, на которой записана единица информации, определяется надежностью считывания). Кроме того, в устройстве можно было разместить не так много пластин и головок. Производители вскоре сошлись на стандарте ATA с 16 физическими головками, т.е. объем диска был ограничен 528 Мб. В любом случае, BIOS не мог адресовать более 1023 цилиндров.

К 1993 году стало ясно, что запросы дискового пространства вышли за допустимые пределы. Весь доступный объем могла занять одна-единственная видеоигра! Поэтому производители BIOS разработали схему под названием LBA (Logical Block Addressing — логическая адресация). Благодаря LBA стало возможно расширение адресации BIOS путем пересчета цилиндров с номерами выше 1024 на "виртуальные головки", используя, таким образом, преимущество в допустимом количестве головок (до 255). Например, диск с 1852 цилиндрами и лишь 16 физическими головками можно рассматривать как имеющий 463 цилиндра и 64 головки. Размер диска при этом не изменяется, но значения цилиндров и головок находятся в пределах, допустимых BIOS. Все стало хорошо... но не надолго.

Режимы Extended INT13 и предел 8.4 Гб

Такая схема представляла собой лишь временное решение, так как и режим LBA должен был переполниться, как только число цилиндров и головок достигнет максимальных значений. Максимально допустимый объем поднялся до 8,4 Гб. Однако прошло всего четыре года, и жесткие диски достигли и этого предела.

Решением стал интерфейс BIOS, названный Extended INT13 (Расширенное использование прерывания 13), который используется современным оборудованием и широко распространен в мире Windows (где программное обеспечение взаимодействует с диском через BIOS, что не требуется во FreeBSD). Этот новый интерфейс абстрагирует адресацию геометрии диска как 16-байтовый пакет дисковой адресации (Disk Addressing Packet), эффективно устраняя какие-либо ограничения размера (по крайней мере, на ближайшее время). Если ваше оборудование не очень старое и

поддерживает режим Extended INT13, вам не потребуется знание геометрии диска. Если воспользоваться какой-либо утилитой, отображающей параметры диска размером более 8,4 Гб, она выведет такие параметры, как 16383 цилиндра, 16 головок и 63 сектора. Подобная конфигурация, сообщает операционной системе, что нет смысла пытаться вычислить размер диска по его геометрии. Помните, что система FreeBSD сама заботиться обо всем, что связано с "геометрией". По крайней мере, до тех пор, пока диски IDE не достигнут 137 Гб и не будет исчерпано максимально возможное в стандарте ATA число цилиндров — 65536... (В настоящий момент и этот барьер преодолен. — Прим ред.)

Практические вопросы

Если вы устанавливаете FreeBSD на старом оборудовании, как, например, материнской плате эпохи LBA, то можете столкнуться с какими-либо из описанных выше ограничений. Вы не потеряете данных, но заметите такие проблемы, как, например, объем диска 8,4 Гб вместо 10 Гб. Теперь вы знаете, с чем это связано.

Конечно же, если вы используете жесткие диски SCSI или FireWire (IEEE 1394), то не столкнетесь с проблемами подобного рода. Эти интерфейсы были разработаны с учетом заведомо большей масштабируемости, чем IDE/ATA.

Разбиение жесткого диска на разделы

Добавление нового диска в систему требует четырех шагов: его установки, разбиения на разделы, разметки и монтирования. В зависимости от того, используется SCSI- или IDE-диск, процедура физической установки будет разной. В этом может помочь инструкция, которая обычно сопровождает диск.

В последующих примерах мы будем предполагать, что жесткий IDE-диск емкостью 40 Гб устанавливается как ведомое (slave) устройство первого контроллера. Имя устройства, как было рассказано в главе 9, — **/dev/ad1**. (SCSI диск в такой же конфигурации имел бы имя **/dev/da1**.) Вспомните, как обычно строится имя устройства. Расширения, следующие за исходным префиксом **ad1**, содержат дополнительную информацию о разделах. Так происходит обращение к различным частям диска, выделенным для использования, например, разными операционными системами.

Разделы BIOS (Slices)

Каждая операционная система позволяет разбить диск на разделы. Важно помнить, что FreeBSD имеет не один, а *два уровня разбиения*. На первом находятся разделы BIOS, адресуемые непосредственно BIOS оборудования ПК. Их может быть не более четырех, и другие операционные системы рассматривают их как "разделы" (partitions). Если разбиение диска выполнялось в системе Windows или Macintosh, каждый раздел является именно разделом BIOS.

В FreeBSD для разделов BIOS используется термин "слайсы" (slices). Об этом важно помнить. Когда же во FreeBSD речь идет о разделах, то подразумевается второй уровень разбиения, более глубокий, чем слайсы. Разделы второго уровня часто называют "разделами BSD".

Разделы BSD

Такие подразделы используются во FreeBSD, чтобы отделить друг от друга разные части системы: **/var**, **/usr**, **/home** и т.д. Каждый раздел имеет имя, состоящее из одной

буквы. В системе может присутствовать до восьми разделов BSD, как показано в табл. 19.6. При разбиении слайса на разделы BSD (это выполняется в процессе разметки, о которой речь пойдет далее), несколько первых букв зарезервированы для специальных целей.

Таблица 19.6 Разделы BSD и их назначение

Раздел	*Назначение*
a	Корневой раздел (/)
b	Раздел swap
c	Адресуется к слою целиком или всему диску в "опасном" режиме. ("Опасный" режим использовался в прошлом, чтобы обойти проблемы с геометрией диска. Он не используется во FreeBSD 4.4 и 5.0)
d	Для общего использования (не используется утилитой Disklabel Editor)
e	Для общего использования
f	Для общего использования
g	Для общего использования
h	Для общего использования

На рис. 19.2 показана структура полного имени устройства. Первая часть (**ad1**) задает основное имя устройства. Диски IDE/ATA имеют имена от **ad0** до **ad3** (четыре возможных устройства контроллера). Затем идет имя слайса, причем номер начинается с 1 (**s1**). И в заключение, имя раздела BSD (**e**).

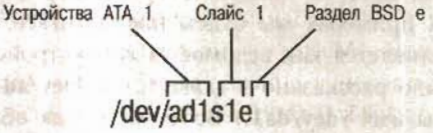


Рисунок 19.2
Расшифровка имени устройства

Дополнительные разделы других операционных систем

Некоторые операционные системы имеют структуру разделов, схожую с разделами во FreeBSD: например, дополнительный раздел DOS (Extended DOS Partition) позволяет создать на DOS/Windows-машине подразделы (логические диски). Функционально идентичный этому метод используется и в Linux. Подобные способы разбиения диска являются довольно неуклюжими по сравнению с решением, предложенным во FreeBSD. Поэтому возможности взаимодействия различных операционных систем усложнены. Разделы внутри дополнительного раздела DOS, например, рассматриваются во FreeBSD как дополнительные слои на том же уровне, что и обычные разделы DOS, начиная со слоя 5 (т.е. после четырех стандартных разделов DOS). Для монтирования второго логического диска внутри дополнительного раздела DOS жесткого диска, установленного как устройство **ad3**, необходимо использовать имя **/dev/ad3s6**, вместо стандартного имени раздела BSD внутри слоя.

Попытка продвинуться в обратном направлении будет выглядеть еще сложнее. Вероятно, вам не удастся смонтировать диск с разметкой FreeBSD в системе DOS/Windows, однако на Linux-машине это вполне возможно. Linux работает с разделами

(которые во FreeBSD называются слайсами) так же, как и FreeBSD: они именуются численно и последовательно, а не иерархически. Для монтирования слайса FreeBSD в Linux воспользуйтесь табл. 19.7, где приведено соответствие имен. В таблице описаны разделы ведомого устройства первого контроллера, на котором третий слайс/расширенный раздел DOS содержит файловую систему FreeBSD.

Таблица 19.7 Соответствие меток в Linux и FreeBSD

Метка Linux	*Метка FreeBSD*
/dev/hdb4	/dev/ad1s3a
/dev/hdb5	/dev/ad1s3b
/dev/hdb6	/dev/ad1s3e
/dev/hdb7	/dev/ad1s3f

Редактор слайсов (fdisk) в программе sysinstall

Как пользователь **root** запустите программу **/stand/sysinstall** и выберите пункт "Configure". Прокрутите экран до опции "Fdisk", отмеченной как "Slice (PC-Style partition) Editor". Выбор этой опции даст возможность получить список всех дисков, которые найдены в системе. Убедитесь, что в нем содержится **ad1** (новый диск, установленный как slave-устройство первого контроллера). Если нет, значит, диск установлен неправильно. Чтобы проверить, был ли диск вообще обнаружен при загрузке системы, просмотрите файл **/var/run/dmesg.boot**. Если устройство не обнаружено, необходимо проверить кабель и состояние джамперов на диске. Помните, что, если на первом контроллере раньше был только один диск, его джампер наверняка был в положении "Single". Необходимо убедиться, что теперь джампер старого диска установлен в положении "Master", а нового диска — в положении "Slave".

Если диск представлен в списке, выберите его (пробелом) и программа запустит визуальный редактор **fdisk** (fixed disk — жесткий диск). Такая утилита в той или иной форме имеется в любой операционной системе, поскольку каждой из них необходим инструмент для управления разделами BIOS (слайсами). Программу **fdisk** можно использовать прямо из командной строки, однако она имеет много опций и потенциальных ловушек, поэтому лучше воспользоваться визуальным интерфейсом, который обеспечивает программа **sysinstall**. В таких вопросах, как разбиение и форматирование диска, лучше подстраховаться!

В меню будут отображены все существующие разделы. Если это новый, неформатированный диск, список слоев будет пустым. Здесь вы можете воспользоваться достаточно гибкими возможностями. При желании можно создать разделы BIOS для других операционных систем или FreeBSD, в стиле Linux или DOS. Но, скорее всего, потребуется выбрать опцию A — "Use Entire Disk" (Использовать весь диск). Меню программы показано на рис. 19.3. О других опциях можно пока не думать, они добавлены для повышения гибкости программы и улучшения совместимости.

Диск содержит три элемента: небольшую область, "зарезервированную" для диспетчера загрузки, который будет установлен позже, один слой FreeBSD (**ad1s1**) и "оставшийся" участок дискового пространства (малой емкости), не включенный в основной слой.

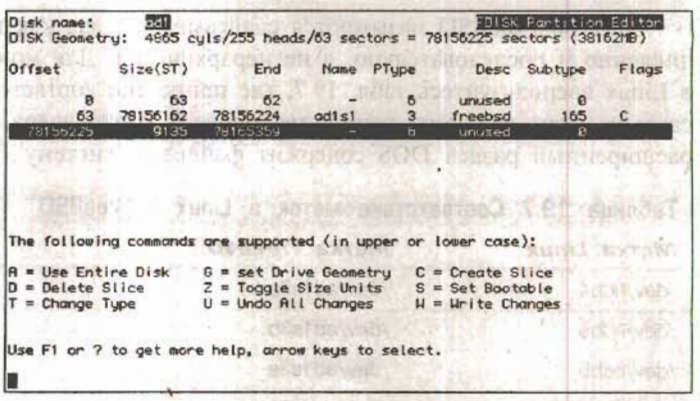


Рисунок 19.3
Slice Editor (после выбора опции "Use Entire Disk").

Однако в этой фазе еще никакие изменения не записаны непосредственно на сам диск. Чтобы зафиксировать разметку диска и сохранить изменения, используется опция **W**. Для выхода из Slice Editor воспользуйтесь опцией **Q**. Программа запросит, какой тип диспетчера загрузки следует установить. Если это второй диск, добавляемый к работающей системе, выберите None (Нет). Если же вы будете загружаться с этого диска, установите FreeBSD Boot Manager (BootMgr). Это может понадобиться, например, когда диск готовится к использованию на другой машине или для замены текущей системы. В ином случае выберите "Standard" — программа выполнит команду, эквивалентную команде DOS **fdisk /MBR**, которая удаляет из MBR диспетчеры загрузки, специфичные для операционных систем (как LILO Linux и BootMgr FreeBSD), и делает его доступным для других систем, например, Windows.

Создание дисковых меток

Итак, диск разбит на слайсы и подготовлен для работы с FreeBSD, но еще не отформатирован — его нельзя подключить к системе. Далее необходимо воспользоваться утилитой Disklabel Editor (Редактор дисковых меток).

Находясь в программе **sysinstall**, выберите Label (Метка), будет запущен Disklabel Editor. Утилита вновь попросит выбрать диск из списка установленных жестких дисков. Для выбора **ad1** используйте пробел. На экране появится меню программы Disklabel Editor — визуального интерфейса команды **disklabel**. В нем создаются и форматируются разделы BSD.

ПРИМЕЧАНИЕ

Disklabel Editor во FreeBSD не позволяет изменять размеры существующих разделов. Этого нельзя сделать, не переформатировав его с потерей всех данных, поэтому тщательно продумайте схему создания разделов до того, как сохранять изменения!

Полная разметка раздела FreeBSD

Если вы добавили в систему новый диск, перейдите к разделу "Adding a New Disk". Этот раздел демонстрирует, как воспользоваться утилитой Disklabel Editor для создания разделов BSD и точек монтирования в полной системе FreeBSD, включая корневой раздел, **/usr**, **/var** и раздел подкачки. Это особенно полезно при инсталляции FreeBSD на новый диск, например, при замене текущего диска новым, большим по объему.

Нажмите **A** и выберите опцию "Auto Defaults for all!" (Автоматические настройки для всех разделов). Программа автоматически вычисляет размер разделов, которые будут пригодными для большинства систем (см. рис. 19.4). Если FreeBSD-система будет использоваться как рабочая станция или небольшой сервер с несколькими пользователями, вы вполне можете воспользоваться этими настройками по умолчанию.

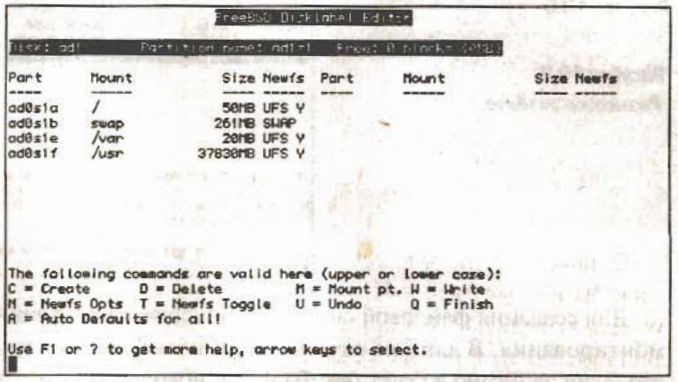


Рисунок 19.4
Утилита Disklabel Editor (после выбора опции "Auto Defaults for all!".

Для сервера настройки по умолчанию наверняка не подойдут. Почтовые сообщения и log-файлы размещаются в разделе **/var**. Если в системе присутствует 100 пользователей, хранящих свой почтовый спул на сервере, или Web-сервер генерирует log-файлы размером 100 Мб, раздел **/var** размером 20 Мб будет мгновенно исчерпан. Таким образом, для **/var** требуется больший объем дискового пространства. Все эти ситуации — замечательный пример того, насколько гибкой может быть работа утилиты Disklabel Editor.

Прежде всего, необходимо удалить оба раздела **/var** и **/usr**. Для этого выберите их и нажмите **D**. Прокрутите экран к верхней части, где поле "Free:" показывает пространство, доступное после удаления разделов. Теперь разделы нужного размера можно создать заново. (Помните, что ничего не будет записано на диск до тех пор, пока вы не нажмете клавишу **W**.)

Разместив курсор в строке "**Partition name: ad1s1**", нажмите "**C**", чтобы создать новый раздел. На экране появится диалоговое окно, запрашивающее объем нового раздела. Отведите **/var** 200 Мб, их должно быть вполне достаточно для среднего сервера. (Если есть возможность, можно отвести и больше.) Вместо числа, отображенного в окне, введите "**200m**". Затем выберите **FS** для создания файловой системы (а не раздела подкачки). Программа запросит точку монтирования — введите **/var**. После этого раздел будет создан и добавлен в список, а пункт "**Free:**" изменен соответствующим образом.

Теперь этот же процесс следует повторить для раздела "**/usr**", используя оставшееся дисковое пространство. В результате список разделов будет выглядеть как приведенный на рис. 19.5.

Добавление нового диска

На новом диске можно создать до восьми разделов BSD. Не используйте опцию **A**, если вы хотите просто добавить дисковое пространство к работающей системе. Вместо этого выберите "**Partition name: ad1s1**" и нажмите клавишу **C**, создав новый раздел. Если вы хотите создать на диске один раздел BSD большого объема (например, добавить 40 Гб к **/home**), примите значение по умолчанию, т.е. полный объем слайса. Можно поступить

и по-другому: указать размер одного раздела (например, **20G**), а из того, что осталось, создать другой раздел.

Рисунок 19.5
Разметка раздела.

Для создания файловой системы (а не области подкачки) выберите **FS** и укажите точку монтирования. В данном случае это не очень важно. Этот пункт указывает лишь на то, что будет записано в файл **/etc/fstab** для новой файловой системы. А это происходит лишь в случае инсталляции системы.

Таким способом можно создать необходимое число разделов, пока не будет использовано все доступное дисковое пространство. Чтобы удалить раздел, выберите его и нажмите клавишу **D**. Помните, что эти операции не затрагивают диск. Все изменения записываются лишь при нажатии клавиши **W**.

ПРИМЕЧАНИЕ

Если вы создадите более четырех разделов, все "зарезервированные" имена будут использованы, начиная с a и b; c будет пропущено, поскольку оно имеет специальное значение. Для разделов после d будет использована "пустая" метка **X**, и вы не сможете их корректно применить.

Сохранение изменений и форматирование диска

Форматирование дисков во FreeBSD осуществляется командой **newfs**. Эта утилита командной строки также запускается автоматически из **sysinstall**. Каждая файловая система из списка, для которой в столбце **Newfs** установлено значение **Y**, будет отформатирована после принятия изменений. Включить или отключить значение можно с помощью клавиши **T**.

Посмотрите внимательно на список разделов, создаваемых в редакторе. Их метки содержатся в столбце **Part**. Независимо от того, как вы создаете разделы, имена от **a** до **d** зарезервированы для специального использования, как показано в табл. 19.6. Первый создаваемый раздел имеет метку **e**. Это значит, что имена новых монтируемых разделов, добавляемых в систему, будут выглядеть так: **ad1s1e**, **ad1s1f** и т.д. Имена меток понадобятся при монтировании файловых систем.

По завершении разметки диска нажмите **W** для сохранения изменений. Диск будет размечен, после чего запустится процесс **newfs**, форматирующий каждый из созданных разделов. По окончании нажмите **Q** для выхода из программы **sysinstall**.

Подготовка файловой системы к использованию

Итак, почти все готово. Остается лишь смонтировать новые файловые системы. Кстати, их автоматическое монтирование при загрузке можно настроить в файле **/etc/fstab**. Подробное обсуждение возможностей команды **mount** и содержимого файла **/etc/fstab** можно найти в главе 9.

Здесь мы сталкиваемся с простейшим из возможных случаев: новые отформатированные разделы FreeBSD. Команду **mount** можно использовать наиболее прямолинейным способом.

Создайте каталог, который будет точкой монтирования, например, **/mnt/newdisk**. (Он может находиться в любом месте файловой системы.) Затем запустите команду **mount**:

```
# mount /dev/ad1s1e /mnt/newdisk
```

Для проверки того, что диск смонтирован, выполните команду **df**. То же самое можно сделать и для любого раздела, разместив точку монтирования в произвольном месте файловой системы. Для демонтирования применяется команда **umount**:

```
# umount /mnt/newdisk
```

Примите наши поздравления — вы овладели основами работы с жесткими дисками в FreeBSD!

20 ГЛАВА

Курс выживания для пользователей FreeBSD

- Переход к FreeBSD
- Чего делать не следует
- Тонкая настройка производительности
- Подготовка к худшему: резервные копии

Данная глава представляет собой краткое руководство для новых пользователей FreeBSD, обучающее тому, как быстро привести систему в устойчивое рабочее состояние. Она предназначена в первую очередь для администраторов, работавших с другими операционными системами. Зачастую у них складывается превратное представление о том, как функционирует FreeBSD. В этой главе будет развеяно большинство заблуждений.

Первые недели работы с новой операционной системой являются наиболее тревожными: некорректные конфигурации могут привести к дырам в защите, потере данных или всеобщей дезорганизации работы системы. Ключом к выживанию в новой среде является изучение возможных ловушек. Знание, что приняты самые необходимые меры предосторожности, сделает углубленное изучение общих принципов администрирования более спокойным и приятным.

Данная глава будет полезна и новичкам и опытным пользователям. В ней собрана информация о различиях философского характера как между FreeBSD и Windows, так и между FreeBSD и ее близким родственником — Linux. Кроме того, здесь показано, как обойти возможные проблемы, и даны советы, помогающие поддерживать работоспособность системы. В заключение, в этой главе рассматривается спектр неизбежных сбоев системы и средств, способных предотвратить их или, по крайней мере, облегчить восстановление.

Переход к FreeBSD

На момент написания этой книги приблизительно 15-20% всех UNIX-серверов с открытым кодом работало под управлением FreeBSD. На большинстве остальных серверов использовался Linux. Очень малый процент составляли машины под управлением NetBSD, OpenBSD и других малоизвестных версий. Если же рассматривать рынок серверов в более широком смысле, то Windows NT/2000 присутствует на 20-30% машин, тогда как доля UNIX-систем (как коммерческих версий, так и версий с открытым кодом) составляет около 65%. В любом случае, это означает, что пока FreeBSD используется на сравнительно небольшом числе машин. Но, как хорошо известно, доля рынка отнюдь не свидетельствует о качестве программного продукта.

Применение FreeBSD на серверах растет. Существует постоянный поток пользователей, переходящих с систем, работающих под Linux или под коммерческими версиями UNIX, к FreeBSD. Это объясняется ее устойчивостью и высокой производительностью. Очень многие предпочитают систему с открытым кодом, политика распространения которой коренным образом отличается от политики распространения Linux. Кроме того, ряд компаний предпочитают иметь FreeBSD на серверах с высокой загрузкой просто в силу того, что только эта система способна справится с поддержкой сильно загруженных Web-служб и обеспечить требуемую производительность и надежность. В конечном счете, уважение к FreeBSD в сообществе пользователей серверов растет. Растет и желание иметь на ответственных участках нечто устойчивое, с мощными средствами управления безопасностью, и не думать постоянно о совместимости стандартов.

Переход к FreeBSD от Windows NT/2000 или Linux означает отказ от двух привычных элементов: доступности программного обеспечения и укоренившихся методов.

Именно необходимость отказа от старых привычек являются тем порогом, о который "спотыкаются" опытные администраторы, сталкивающиеся с FreeBSD первый раз. Мы рассмотрим пути перехода к FreeBSD от этих двух систем и основные препятствия на этом пути.

Переход от Windows NT/2000

Windows NT/2000 представляет собой систему с графическим интерфейсом, которая не позволяет производить удаленное администрирование. Это значит, что для эффективной работы с Windows NT/2000 необходимо сидеть непосредственно за машиной. Удобный для удаленного доступа интерфейс командной строки — еще одно преимущество UNIX-систем.

СОВЕТ

Встроенные средства Windows NT/2000 и ПО от сторонних производителей значительно повышают возможности удаленного доступа к этой платформе. В принципе, удаленно управлять можно службами, содержащимися в панели управления Services. Кроме того, существуют различные методы удаленного доступа к командной строке DOS на серверной машине. Однако они не представляют собой полноценной функциональности машины, так как ограничены используемым механизмом управления. Возможность конфигурирования программного обеспечения, установленного на сервере, зачастую зависит от клиентских программ. О них рассказано в следующем разделе.

Приложения клиент-сервер

Большинство корпоративного программного обеспечения для серверов Windows разработано согласно следующей модели: на сервере физически устанавливается серверная часть этого ПО, а на рабочей станции — клиентская. Последняя используется для администрирования сервера. Среди пакетов, как правило, присутствует терминальный сервер, позволяющий непосредственно управлять сервером. В силу ориентации Windows на графический интерфейс, подобные средства не слишком удобны для удаленного администрирования.

Во FreeBSD, в принципе, нет специализированных приложений клиент-сервер. Некоторые из них используют интерфейс Web-браузера (например, серверы аудио-видеопотоков и т.д.), однако большинство задач администрирования системы решается с помощью удаленного доступа к терминалу. Telnet (или SSH) — доминирующие многоцелевые средства в мире UNIX. Технически, это один из самых эффективных методов для работы с сервером, так как в обе стороны пересылается лишь текст. Кроме того, это наиболее гибкий метод, способный предоставить доступ к операциям сервера, что не под силу ни одному графическому средству. Естественно, он предоставляет и больше возможностей, чтобы испортить работу системы. В этом-то и состоит проблема выбора: с одной стороны, простота, относительная надежность и прямая функциональность (с невозможностью удаленного доступа) программного обеспечения клиент-сервер Windows, с другой — гибкость и возможности терминального администрирования, сочетающиеся с определенным риском.

Защита

О модели защиты Windows NT/2000 написаны целые тома. Разумеется, данная книга не рассматривает эти вопросы так подробно. Но если вы в управлении пользо-

вателями, доменами и правами доступа привыкли к стилю Windows, то переход к FreeBSD потребует изучения некоторых новых элементов.

Пользователь **root** эквивалентен понятию Administrator. Это ни для кого не секрет. Однако понятие **root** значительно шире: это и возможности запуска служб с определенными привилегиями, и возможность предоставления необходимых прав другим администраторам (но не более, чем нужно для выполнения определенной работы). Итак, на первом этапе мы рассмотрим фундаментальные различия между моделями защиты Windows NT/2000 и FreeBSD.

- Во FreeBSD для указания прав владения и доступа используются в первую очередь биты User, Group и Other. В Windows NT/2000 для этой цели предназначены Списки контроля доступа (Access Control Lists). Во FreeBSD 5.0 применение ACL также возможно (см. главу 10), но в Windows NT/2000 — это единственная модель прав доступа.
- В Windows NT/2000 применяется концепция "доверительных путей" (trusted paths), разрешающая неявную аутентификацию пользователей, регистрирующихся в системе с указанных хостов. Например, Windows можно указать "доверять" пользователю, соединяющемуся с системой с некоторого компьютера (при определенных или любых обстоятельствах), исключая тем самым необходимость аутентификации. FreeBSD не поддерживает таких возможностей. Их можно добиться лишь применением системы Kerberos (об этом рассказано в главе 29).
- Имена учетных записей во FreeBSD могут включать до 16 символов и содержат только символы нижнего регистра и цифры. Использование пробелов и специальных символов не поддерживается.
- Windows NT/2000 имеет 27 различных пользовательских прав для изменения и запуска файла с различными привилегиями. FreeBSD различает процессы, запущенные пользователем **root**, отдельными пользователями и в режиме setuid (или set-user-ID — с установленным идентификатором пользователя). В последнем случае процесс имеет права доступа владельца файла.
- FreeBSD и UNIX не поддерживают иерархических групп. Windows NT/2000 позволяет размещать группы пользователей внутри других групп до произвольной глубины. Во FreeBSD это невозможно. Все группы существуют на одном уровне.
- Домены Windows не существуют в мире UNIX. Службы удаленной аутентификации (как, например, Domain Controller в Windows NT/2000) и домены защиты обрабатываются посредством систем NIS (yellowpages, "желтые страницы"). Подробные сведения о NIS содержатся в справочном руководстве **man nis**.
- Равноправный раздельный доступ к файлам, осуществляемый NetBIOS в Windows, в UNIX-системах реализуется с помощью NFS. Воспользовавшись пакетом Samba во FreeBSD можно настроить разделяемый доступ к файлам и принтерам в стиле Windows. О NFS рассказано в главе 31, Samba обсуждается в главе 32.

- Для аутентификации кода в загруженных пакетах программного обеспечения в Windows используется схема под названием Authenticode. В этой схеме загруженный пакет должен представить "сертификат", в результате проверки которого он принимается или отвергается пользователем. Таким образом, система имеет дополнительный слой защиты против потенциально опасного программного обеспечения. В UNIX нет подобного стандартизованного механизма, но система портов/пакетов FreeBSD обеспечивает централизованную проверку и аутентификацию, подходящую практически для всех приложений. Подробнее о портах и пакетах см. главу 15.

Windows NT/2000 провозглашают системой, которая значительно проще в настройке защиты, чем UNIX, однако опыт показывает, что это не так. Windows очень просто сделать незащищенной: небольшие ошибки в конфигурировании, предпринимаемом с благими намерениями. Хотя подобное возможно и во FreeBSD, тем не менее это бывает значительно реже. Это связано с тем, как структурирована система UNIX (модель, которая поощряет более высокие меры безопасности в ответ на возможные действия неопытного пользователя), а также с большим объемом полезной информации, посвященной защите UNIX-систем. Замечательным ресурсом является, например, сайт SANS (System Administration, Networking, and Security) Института системного администрирования, применения сетей и защиты — **http://www.sans.org**.

Эквиваленты программного обеспечения

Рассмотрим примеры программного обеспечения, используемого на серверах Windows, и его эквиваленты во FreeBSD (см. табл. 20.1).

Таблица 20.1 Программное обеспечение серверов Windows

Служба	*Windows NT/2000*	*FreeBSD*	*Доступен в*
Mail (SMTP)	Microsoft Exchange	Sendmail	Основной системе
Mail (POP)	Microsoft Exchange	**qpopper**	Портах
MAIL (IMAP)	Microsoft Exchange	IMAP-UW	Портах
News (NNTP)	Microsoft Exchange	**inn**	Портах
Ведение системных журналов	Microsoft SMS	syslogd	Основной системе
Web/HTTP	Microsoft IIS	Apache	Портах
FTP	Microsoft IIS	**ftpd**	Основной системе
Динамические Web-страницы	Cold Fusion, ASP	**PHP**	Портах
DNS	Microsoft Active Directory	BIND	Основной системе
Remote Access (Удаленный доступ)	Microsoft Terminal Server	Telnet, SSH	Основной системе
Directory Services (Службы каталогов)	Microsoft Active Directory	OpenLDAP	Портах

Следует отметить, что в мире Windows тесная интеграция между различными элементами программного обеспечения сервера зачастую приводит к тому, что на одной и той же машине удается использовать не более одной важной службы. Эту приводит к использованию набора отдельных машин: почтовый сервер, Web-сервер, DNS-сервер и т.д. Сообщество пользователей систем UNIX с открытым кодом не присоединяется к такой философии по двум причинам:

- UNIX-системы с открытым кодом часто используются именно из-за их низкой цены. Это свидетельствует о том, что покупка нескольких серверов выходит за пределы возможностей бюджета организации.
- Программное обеспечение в UNIX обычно более устойчивое и корректное, чем в Windows, поэтому "вредные" взаимодействия между службами или демонами чрезвычайно редки и потенциально менее опасны.

Вам необязательно следовать методу, предлагаемому UNIX. Тем не менее нужно отметить, что выполнять несколько важных служб на одной машине, работающей под UNIX, гораздо проще, чем на Windows-сервере.

Поддержка системы на современном уровне

Компания Microsoft поддерживает обновление системы посредством *сервисных пакетов* (*service packs*). Сервисный пакет — это большой набор изменений в программах и исправления двоичных файлов, которые появляются примерно раз в полгода. Быстрее появляются (например, когда бывают найдены дыры в защите) двоичные исправления (*hotfixes*), которые можно загрузить с Web-сайта Microsoft.

Метод, применяемый для поддержки FreeBSD в актуальном состоянии, может показаться более сложным, однако, когда вы поймете его, то увидите, что он проще и эффективнее. FreeBSD придерживается принципа, что система должна быть всегда доступна в исходном коде, за которым можно следить, поддерживая локальную копию всего дерева с исходным кодом и синхронизуя ее с любой требуемой частотой. Средство CVSup автоматизирует этот процесс и заботится о том, чтобы объем пересылаемой информации был минимален. При желании можно хранить дистрибутив релиза FreeBSD, который был использован для инсталляции системы, в исходном коде и периодически применять к нему исправления (patches). Это один из самых распространенных методов поддержки системы.

Каждый бюллетень защиты, рассылаемый FreeBSD Security Officer (лицо, поддерживающее бюллетень), содержит исправления, сделанные в исходном коде. Бюллетень сопровождается подробными инструкциями. Их остается лишь скопировать в командную строку и выполнить. Как правило, систему не требуется даже перезагружать.

Систему можно содержать на самом современном уровне, путем сборки из синхронизованного исходного кода, обновление которого можно производить как угодно часто. Этот процесс называется **make world**, его подробное описание дано в главе 18. Разумеется, этого не следует делать новичкам в мире FreeBSD, но по мере накопления опыта работы вы сможете обновлять систему до требуемого состояния. На первом этапе вам будет вполне достаточно информации, изложенной в главе 18, чтобы синхронизировать исходный код ветви RELEASE, вносить исправления, направленные на укрепление защиты, и поддерживать систему на должном уровне.

Переход от Linux

Естественно, Linux ближе к FreeBSD, чем Windows NT/2000. Обе эти системы следуют традициям UNIX, поставляются в исходном коде и их модели защиты практически идентичны. Для обеих платформ существует одно и то же программное обеспечение (особенно, если принять во внимание совместимость FreeBSD с форматом двоичных файлов Linux). Кроме того, на обеих платформах возникают сходные проблемы системного администрирования, поэтому и применяемые решения во многом совпадают.

Трудности перехода от Linux к FreeBSD носят больше технологический, чем философский характер, поэтому в этом разделе особенности администрирования рассматриваются глубже, чем в разделе, посвященном переходу от Windows.

Функциональные различия между Linux и FreeBSD:

- Размещение различных компонентов системы
- Разделы и типы файловых систем
- Порты и пакеты
- Синхронизация системы
- Файлы паролей
- Технические особенности

Размещение различных компонентов системы

О чем следует помнить во FreeBSD? Повторяйте это как заклинание: все, что устанавливает администратор, размещается в каталоге **/usr/local**. FreeBSD тщательно придерживается этой модели, предотвращая таким образом "загрязнение" основной системы (**/usr/bin**, **/var/lib** и т.д.). В табл. 20.2 приведено несколько примеров расположения различных компонентов системы. При этом следует помнить, что они могут отличаться в разных дистрибутивах Linux.

Таблица 20.2 Сравнение расположения некоторых ресурсов в Linux и FreeBSD

Ресурс	*Linux*	*FreeBSD*
Корневой каталог сервера Apache	/var/lib/apache	/usr/local/www
Конфигурационный файл Apache	/var/lib/apache/conf	/usr/local/etc/apache
Двоичные файлы	/var/lib/apache/sbin	/usr/local/sbin/
Выполняемый файл Sendmail	/usr/lib/sendmail	/usr/sbin/sendmail
Каталог базы данных MySQL	/var/lib/mysql	/var/db/mysql
Начальные сценарии запуска системы	/etc/rc.d	/etc/rc.*
Начальные сценарии запуска, установленные пользователем	/etc/rc.d	/usr/local/etc/rc.d
Двоичные файлы, установленные пользователем	/usr/bin	/usr/local/bin
Библиотеки, установленные пользователем	/usr/lib	/usr/local/lib
Файлы разделяемого доступа, установленные пользователем	/usr/share	/usr/local/share

Некоторые отличия связаны с разницей в работе RPM (широко распространенного в Linux менеджера пакетов) и метода портов/пакетов FreeBSD, применяемого для инсталляции программного обеспечения. Другие являются следствием разных схем построения основной системы (Linux наследует часть структуры System V, тогда как FreeBSD, естественно, подобна BSD). Подробное обсуждение работы портов и пакетов во FreeBSD приведено в главе 15.

Разделы и типы файловых систем

Файловая система Ext2FS, используемая в Linux, в разбиении диска на разделы напоминает DOS: существует четыре основных раздела BIOS, один из которых можно превратить в "дополнительный раздел", с содержащимися внутри него "логическими" разделами. Все они доступны пользователю на одном "уровне", хотя на самом деле представляют собой иерархическую структуру. Систему нельзя загрузить с дополнительного раздела (только с основного!), что ведет к потенциальной путанице.

Иерархическая структура разделов FreeBSD более явная: разделы BIOS, называемые слайсами (slice), подразделы, называемые разделами (или разделами BSD). Процессы загрузки обрабатывают различные уровни явно, раскрывая тайну, связанную с дополнительным и основным разделами. Настройка диска включает в себя создание слайсов, а затем разбиение диска. Этот процесс подробно описан в главе 19.

Исходной файловой системой FreeBSD является UFS/FFS. Диски Linux можно использовать во FreeBSD без изменений, однако ядро FreeBSD необходимо перекомпилировать с поддержкой Ext2FS. О том, как включить эту опцию и собрать новое ядро, рассказано в главе 17.

Порты и пакеты

Пользователи, работавшие с Linux, возможно, привыкли инсталлировать новое программное обеспечение с помощью утилиты RPM. Хотя это возможно и во FreeBSD, тем не менее делать так не рекомендуется (поскольку это может всерьез повредить системе, например, путем установки несовместимых библиотек в каталог **/usr/lib**). Предпочтительным методом инсталляции и поддержки программного обеспечения во FreeBSD являются средства работы с пакетами (**pkg_***) или порты.

Пакеты и порты обеспечивают централизованный механизм фильтрации программного обеспечения, написанного для Linux или других версий UNIX, с проверкой опций сборки и инсталляции. Такой подход обеспечивает корректную компиляцию программ во FreeBSD и установку в предназначенные для них каталоги. Порты и пакеты доступны, как правило, не с сайтов разработчиков приложений, а с централизованных сайтов. Их поддержкой занимаются специально уполномоченные лица (committers). О портах и пакетах подробно рассказано в главе 15.

Синхронизация системы

FreeBSD больше полагается на исходный код, чем Linux: поскольку существует один-единственный дистрибутив операционной системы, она доступна в исходном коде. В мире Linux нет ничего подобного. Здесь каждый дистрибутив имеет свою структуру и собственный набор программного обеспечения. Единственный компонент исходного кода, который гарантированно существует в любой Linux-системе, это каталог с исходным кодом ядра. Разные дистрибутивы придерживаются различ-

ных принципов автоматической загрузки файлов с исходным кодом, синхронизации и сборки.

Таким образом, для синхронизации локальной копии исходного кода с самой новой версией во FreeBSD достаточно воспользоваться утилитой CVSup, как описано в главе 18. Исправления защиты и обновления (hotfixes) (доступные в Linux как RPM, исправления исходных кодов или исправления двоичных файлов) распространяются во FreeBSD в двух формах: исправления исходного кода и обновления файлов в центральном дереве.

ПРЕДУПРЕЖДЕНИЕ

Обновление компонентов системы (а не системы целиком) представляет собой эффективный метод, который работает в большинстве случаев (особенно для небольших компонентов системы). Но чем "фундаментальнее" составной элемент системы, тем больше вероятность того, что для его корректной работы потребуется выполнить **make world** (см. главу 18). Такие важные составные части системы включают, как правило, основные библиотеки (в каталоге **/usr/lib**) и заголовочные файл (**/usr/include**). К менее рискованным областям относятся двоичные файлы в **/usr/bin** и **/usr/sbin**.

Файлы паролей

В Linux и во FreeBSD используются *теневые пароли* (*shadow passwords*). Это свойство защиты системы описано в главе 10. Во FreeBSD файл **/etc/master.passwd** содержит все данные пользователей, а **/etc/passwd** — лишь их поднабор (без реальных паролей). В Linux файл **/etc/shadow** не содержит никакой полезной информации, кроме имен пользователей и зашифрованных паролей. Скомбинировав файлы **/etc/passwd** и **/etc/shadow** из Linux-системы, можно получить полноценный файл **master.passwd** и установить его во FreeBSD. Таким способом можно сохранить всю регистрационную информацию пользователей. Для этого используется сценарий на языке Perl, приведенный в листинге 20.1 (он содержится на прилагаемом CD-диске под именем **mkpasswd.pl**).

Листинг 20.1 Пример сценария для преобразования базы данных пользователей Linux в формат FreeBSD

```
#!/usr/bin/perl
# Установите путь к интерпретатору bash на FreeBSD-машине (если он
# установлен), или же сделайте начальным командным интерпретатором /bin/csh
$newshell = "/bin/csh";

open (PASSWD,"/etc/passwd");
@passwd = <PASSWD>;                          # Прочесть файл /etc/passwd
close (PASSWD);
open (SHADOW,"/etc/shadow");
@shadow = <SHADOW>;                          # Прочесть файл /etc/shadow
close (SHADOW);

foreach $user (@shadow) {
   @sdata = split(/:/,$user);
   $passhash{@sdata[0]} = @sdata[1];         # Создать таблицу поиска по именам
                                             # пользователей
}
foreach $user (@passwd) {
   @pdata = split(/:/,$user);
   @pdata[6] = $newshell."\n" if (@pdata[6] eq "/bin/bash\n");
   @pdata[1] = $passhash{@pdata[0]};   # Заменить "x" зашифрованным паролем
```

```
    splice (@pdata,4,0,undef,"0","0");      # Объединить лишние пустые поля
    foreach (@pdata) {                      # Вывести окончательную строку
        print "$_";
        print ":" unless ($_ =~ /\n/);
    }
}
```

Сделайте сценарий выполняемым и запустите с правами пользователя **root** в Linux-системе, перенаправив вывод в файл:

```
% ./mkpasswd.pl > new.master.passwd
```

Перенесите файл **new.master.passwd** на FreeBSD-машину и поместите в **/etc** (**pwd_mkdb** не принимает данные из стандартного ввода или из файла в другом каталоге). Откройте его в текстовом редакторе, а в другом окне — файл **/etc/master.passwd** FreeBSD. Первый блок, примерно из 15 учетных записей, необходимо откорректировать (это **daemon**, **bin**, **news** и другие системные псевдопользователи, чьи идентификаторы в Linux и FreeBSD различны). Для этого скопируйте этот блок из **/etc/master.passwd** и замените соответствующий блок в **new.master.passwd**.

Создайте резервную копию файла **/etc/master.passwd**. В заключение, запустите **pwd_mkdb** для создания нового файла баз данных hash-паролей:

```
# pwd_mkdb -p /etc/new.master.passwd
```

Если команда завершится без ошибок, не выходите из системы. Необходимо проверить настройки, обладая правами **root**. Вы должны зарегистрироваться и открыть новый терминальный сеанс. Если вы не сможете этого сделать, запустите команду **pwd_mkdb**, указав в качестве параметра резервную копию файла, и базы данных вернутся к предыдущему состоянию.

Технические особенности

Между Linux и FreeBSD существует множество небольших отличий. Вот наиболее важные из них:

- FreeBSD не имеет файла **System.map**. В Linux эта таблица используется для хранения векторов перехода (jump vectors) к часто используемым символам ядра. Во FreeBSD эти символы встроены в ядро, поэтому думать о них не нужно.
- Командным интерпретатором по умолчанию во FreeBSD является **tcsh**, а не **bash**. Кстати, **tcsh** и **csh** являются жесткими ссылками друг на друга, т.е. попросту совпадают. Если требуется **bash**, его необходимо установить из портов или пакетов.
- Пакет двоичной совместимости с Linux (его желательно установить) создает дерево **/compat/linux**, которое содержит базовую структуру выполняемых файлов и библиотек, необходимых Linux-программам. Он также создает модуль ядра **linux.ko** (его состояние позволяет проверить команда **kldstat**). Эти структуры позволяют "прозрачно" обрабатывать выполняемые файлы Linux. Возможны определенные аномалии, однако большая часть процесса проходит гладко. Тем не менее следует устанавливать версии программ, предназначенных для FreeBSD, из портов и пакетов.

- По умолчанию FreeBSD не позволяет удаленно регистрироваться в системе с правами пользователя **root**. Изменить эту настройку позволяет файл **/etc/ttys** (см. главу 10).
- Начиная с версии 5.0 FreeBSD использует файловую систем устройств DEVFS вместо стандартного каталога **/dev**, заполненного "специальными" файлами. О том, как работать с DEVFS, рассказано в главе 17.
- Устройства шины PCI во FreeBSD нумеруются, начиная с нуля для интегрированных устройств. Для остальных устройств шины PCI номер растет по мере "удаления" от материнской платы. В Linux интегрированные устройства, как правило, имеют наиболее высокие номера, которые уменьшаются с "удалением" от материнской платы. Это значит, что на машине с двумя картами Ethernet устройства **eth0** и **eth1** в Linux будут во FreeBSD именоваться, соответственно, **fxp1** и **fxp0**.

За исключением приведенных выше пунктов, переход от Linux к FreeBSD проходит достаточно гладко.

Что "следует" и "не следует" делать

Каждая операционная система имеет многочисленные ловушки, в которые легко попадают непосвященные, особенно при новой инсталляции. Поэтому далее мы рассмотрим испытанные методы администрирования и типы наиболее часто совершаемых ошибок.

Итак, СЛЕДУЕТ:

- Использовать порты и пакеты для инсталляции программного обеспечения. Хотя можно получить исходный код определенной программы непосредственно от разработчика и скомпилировать его, используя сценарии, поверьте, это лишь "загрязнит" базовую систему. Если программа не распространяется как порт, пошлите сообщение с запросом в список рассылки **freebsd-ports@freebsd.org**. Кроме того, вы можете изучить процесс портирования и даже сами стать лицом, поддерживающим определенный порт.
- Следить за регулярными (ежедневными, еженедельными и ежемесячными) сообщениями. В них отображаются изменения в выполняемых файлах с битом **setuid**, использование файловой системы, отвергнутые почтовые хосты и другие элементы, способные эффективно указать на наличие вторжения.
- Изучить средство CVSup. Оно не просто позволяет синхронизовать исходный код системы и порты, но может служить и хорошим решением при зеркалировании (об этом рассказано далее в этой главе).
- Инсталлировать утилиту **sudo** (**/usr/ports/security/sudo**) и поощрять ее использование. Если вам приходится администрировать систему совместно с другими пользователями, предоставление доступа к **sudo** позволяет ограничить их потенциальную возможность нанести ущерб системе, одновременно позволяя им эффективно выполнять свои задачи.
- Использовать **ntpd** и **ntpdate** для синхронизации системных часов. Для этого достаточно добавить следующие строки к файлу **/etc/rc.conf**:

```
ntpdate_enable="YES"
ntpdate_flags="tick.usno.navy.mil"
xntpd_enable="YES"
```

Любой NTP-сервер подходит для использования утилитой **ntpdate**, которая выполняется во время загрузки системы и корректирует несоответствия во времени до того, как будут запущены другие службы. Для управления демоном **ntpd**, который синхронизирует часы с NTP-сервером, следует создать файл **/etc/ntp.conf**. Подробные сведения о создании этого файла имеются в **man ntp.conf**.

- Быть вежливым с теми, кто поддерживает систему FreeBSD. Их около 15 человек и вся ответственность за операционную систему целиком лежит на их плечах (в отличие от тысяч независимых разработчиков и распространителей Linux; о корпорации Microsoft я вообще молчу).
- Решить, какой модели защиты следует придерживаться. Доверяете ли вы всем своим пользователям? Если да, то большинство ресурсов системного уровня можно оставить видимыми. В этом случае, когда возникает несоответствие прав доступа, например, Web-приложениям требуется сохранять пароли в текстовом виде или записывать в каталог, правами на запись в который обладают все пользователи, беспокоиться не стоит. Если же нет, вам следует быть более осторожным. Принципы защиты системы в сетевом окружении подробно обсуждаются в главе 29.
- Подписаться на соответствующие списки рассылки по вашей версии FreeBSD. Если вы поддерживаете сервер, вам просто необходимо подписаться на **freebsd-security@freebsd.org**. Полезными могут оказаться также списки **freebsd-ports@freebsd.org** и **freebsd-questions@freebsd.org**. Если вы следите за исходным кодом версии STABLE или CURRENT, подпишитесь на **freebsd-stable@freebsd.org** или **freebsd-current@freebsd.org**.
- Следить за размером log-файлов. Файлы **httpd-access.log** и **httpd-error.log**, принадлежащие Apache, поглощают огромное количество дискового пространства, если через Web-сервер проходит заметный объем трафика. Воспользуйтесь встроенной утилитой Apache **rotatelogs** или запустите задание **cron**, периодически очищающее log-файлы. Если раздел **/var** недостаточно велик, почта, базы данных и рабочие файлы быстро заполнят весь его объем, и система не сможет корректно функционировать.
- С гордостью поднять флажок с "демоном".

НЕ СЛЕДУЕТ:

- Использовать экспериментальные средства на корпоративном сервере. На нем не следует проверять новые идеи, как, например, Soft Updates или модули ядра для неподдерживаемой файловой системы. Опробуйте их на отдельной машине. И лишь после того, как вы полностью разберетесь с ними, применяйте на сервере.
- Производить сканирование портов сервера при запущенном PortSentry или другом средстве выявления вторжения. Это средство блокирует получение па-

кетов с атакующего IP-адреса, и вы утратите связь. В этом случае вам придется найти новую точку сети, чтобы из нее достичь сервера и снять блокирование трафика с IP-адреса, попавшего в "черный" список.

- Использовать команду **shutdown** для перехода в однопользовательский режим (например, для замены ядра), если система работает на уровне защиты **securelevel** 1 или выше. Данная команда не переводит систему на этот уровень. В таких случаях необходимо перезагрузить машину и ввести **boot -s** в приглашении загрузчика. Для возврата в многопользовательский режим достаточно просто выйти из интерпретатора. Все это можно проделать только с физической консоли!
- Слишком увлекаться временем непрерывной работы системы. Помните, что система, которую не перезагружали в течение 350-дней, либо не использовалась в течение всего года, либо не обновлялась в соответствии с бюллетенями защиты. Этого следует избегать и своевременно обновлять систему.
- Забывать добавить себя (как пользователя) в группу **wheel** во время инсталляции! Иначе после инсталляции нового сервера вы не сможете получить прав пользователя **root** при удаленном доступе.
- Использовать пробелы в именах файлов. Это возможно (когда пробелы, как и любые специальные символы, экранированы символом обратной косой черты, например, **My\ Document.txt**), однако нет гарантий, что программное обеспечение будет работать с такими именами корректно (хотя в большинстве случаев, ошибок не возникает).
- Создавать сценарии командного интерпретатора или программы на языке Perl на локальной Windows-машине, а затем загружать в двоичном режиме по FTP. Двоичный режим не производит правильного перевода символов конца строки и, хотя сценарий может выглядеть корректным, символ возврата каретки в конце первой строки (например, **#!/usr/bin/perl**) не будет правильно распознан интерпретатором. Если наблюдаются проблемы с CGI-программированием и сценариями, которые по непонятным причинам не работают после загрузки на сервер, убедитесь, что загрузки происходит в ASCII-режиме. В любом случае предпочтительнее программировать на самом сервере. Для этого существует множество удобных средств.

Тонкая настройка производительности

В конфигурации по умолчанию FreeBSD оптимизирована для работы с устаревшим оборудованием таким образом, чтобы обеспечить максимум безопасности и надежности. Это значит, что при сильной загруженности производительность системы будет значительно ниже, чем у той же машины, управляемой Linux или Windows. Это вовсе не свидетельствует о том, что FreeBSD хуже как платформа. Просто ее нужно настроить для высокопроизводительной эксплуатации. Другие платформы, с которыми FreeBSD обычно сравнивают, по умолчанию настроены на максимальную производительность, что исключает возможность их запуска на устаревшем оборудовании. В большинстве случаев серверы FreeBSD после настройки демонстрируют очень высокую производительность.

Настройка производительности затрагивает несколько областей системы. Ряд настроек требует пересборки ядра (о чем было рассказано в главе 17), а некоторые — нет.

Настройки ядра

Если машина используется как высокопроизводительный сервер, ядро необходимо настроить для поддержки высокого уровня трафика. Ядро **GENERIC** предназначено для рабочей станции или слабо загруженного сервера, поэтому его требования к оборудованию минимальны. Высокие требования ядра к ресурсам могут слишком перегрузить систему, если она не способна удовлетворить им.

Прежде всего, необходимо увеличить значение опции **MAXUSERS** (по умолчанию оно равно 32). Эта настройка устанавливает не допустимое число пользователей, которые могут одновременно работать в системе (как можно было бы ошибочно предположить по ее названию), а важный внутренний параметр, на основе которого вычисляются другие значения (количество сетевых буферов памяти, или **NMBCLUSTERS** и т.д.). Для загруженных систем значение **MAXUSERS** следует установить равным 128 или 256 (и даже выше).

СОВЕТ

Значение опции **NMBCLUSTERS**, вычисляемое как **(MAXUSERS * 16) + 512**, управляет числом доступных сетевых буферов памяти (их можно просмотреть с помощью команды **netstat -m**). Если вывод показывает, что все они использованы, а увеличение значения **MAXUSERS** не изменяет ситуацию, опцию **NMBCLUSTERS** можно настроить в ядре независимо, установив для сильно загруженных систем значение, равное 16384 или 32768. Можно поступить и по-другому: установить значение соответствующей переменной на этапе запуска ядра. Для этого к файлу **/boot/loader.conf** необходимо добавить строку **kern.ipc.nmbclusters="16384"**.

Система Soft Updates и асинхронная запись

Одним из свойств, которое позволяет FreeBSD надежно работать на маломощных системах, является *синхронная запись* (*synchronous writes*). Этот термин означает, что система ожидает, пока завершится каждая операция записи метаданных в файловой системе. При таком подходе в случае аварийного завершения работы системы или внезапном отключении питания маловероятно, что файловая система будет повреждена. В то же время это значит, что система будет работать заметно медленнее, чем подобные платформы, поскольку в ней установлен искусственный "потолок" производительности. Впрочем, при желании, его можно легко устранить.

В главе 9 обсуждался механизм Soft Updates, методика, при которой дисковые операции записи выполняются на уровне, управляемом программным обеспечением. При этом ускоряется дисковый ввод-вывод (I/O) и сохраняется требуемый уровень надежности файловой системы при возможных сбоях.

Механизм Soft Updates встроен в ядро **GENERIC** и поэтому им всегда можно воспользоваться. Однако для этого его необходимо включить с помощью утилиты **tunefs**. Последняя представляет собой вспомогательную программу для поддержки файловой системы FFS, используемой в FreeBSD. Программу **tunefs** следует запускать в однопользовательском режиме, чтобы изменения вступили в силу, поскольку файловые системы, на которые она воздействует, не могут быть в этот момент смон-

тированы. Так как **tunefs** размещается в каталоге **/sbin**, она доступна в однопользовательском режиме без монтирования дополнительных файловых систем.

Перегрузите систему в однопользовательском режиме (введите **boot -s** в приглашении загрузчика). В командной строке введите следующее:

```
tunefs -n enable /
tunefs -n enable /usr
tunefs -n enable /var
```

Выполните эту команду и для других файловых систем на диске. Возможно, вам потребуется указать полный путь команды **/sbin/tunefs** вместо **tunefs**.

Другой способ обойти синхронную запись (если не используется Soft Updates) состоит в монтировании файловых систем в асинхронном режиме, когда операции записи не препятствуют выполнению системных функций. Преимуществом является выигрыш в скорости, недостатком — снижение надежности. Если система аварийно завершит работу в момент записи на диск, файловая система будет повреждена (в некоторых случаях непоправимо).

Если вы готовы пойти на риск, присущий асинхронному монтированию файловой системы, воспользуйтесь опцией **async** команды **mount** в файле **/etc/fstab**. Команда **mount**, выполняемая из командной строки, выглядит следующим образом:

```
# mount -o async /usr /dev/ad0s1f
```

Для автоматического монтирования в асинхронном режиме добавьте опцию **async** в четвертом столбце файла **/etc/fstab**, например:

```
/dev/ad0s1f        /usr        ufs        rw,async        2        2
```

Вопросы геометрии диска

Если требуется оптимизировать производительность диска во FreeBSD, имеет смысл расположить разделы в определенных областях диска. Поскольку данные с периферийной части диска пересылаются быстрее, чем из центральных областей (диск вращается с постоянной угловой скоростью, поэтому за один оборот с крайних дорожек можно прочесть больше данных, чем с близких к центру), нужно расположить активно используемые файловые системы ближе к краю.

Вот почему большое значение имеет порядок, в котором разделы записаны в таблице разделов. Первая запись начинается от края диска, а все последующие размещаются ближе к центру (на оптических носителях, например, компакт-дисках, хранение данных организовано по-другому: диск вращается с постоянной линейной скоростью, поэтому угловая скоростью зависит от точки, с которой в этот момент происходит чтение). Чаще всего обращение происходит к корневому разделу и разделу подкачки, поэтому они должны находиться ближе к краю. Кроме того, файловые системы меньшего размера предпочтительнее разместить как можно ближе к краю диска. Таким образом, оптимальное разбиение диска выглядит так: вначале располагается небольшой корневой раздел, за ним — раздел подкачки (вдвое больший, чем объем физической памяти). Далее следуют **/var**, **/usr** и **/home** (обычно, их размер растет именно в таком порядке).

СОВЕТ

В UNIX алгоритмы управления виртуальной памятью, как правило, оптимизированы таким образом, чтобы раздел подкачки был вдвое больше, чем объем физической памяти. Поэтому, например, при 256 Мб оперативной памяти следует создавать раздел подкачки в 512 Мб. Раздел меньшего размера приведет к замедлению, когда операции постраничной записи будут перекрывать друг друга.

Несколько настроек с помощью sysctl

Программа **sysctl** управляет несколькими переменными ядра (например, **kern.ipc.nmbclusters**, с которой мы уже сталкивались ранее). Их значения отвечают устойчивой работе системы на маломощном оборудовании с низкой производительностью. Настроить значения требуемых переменных можно в файле **/etc/sysctl.conf**, обработка которого происходит при загрузке системы. В инсталляции по умолчанию файл **/etc/sysctl.conf** отсутствует. Поэтому его необходимо создать и добавить следующие записи:

```
vfs.vmiodirenable=1
kern.ipc.maxsockbuf=2097152
kern.ipc.somaxconn=8192
kern.ipc.maxsockets=16424
kern.maxfiles=65536
kern.maxfilesperproc=32768
net.inet.tcp.rfc1323=1
net.inet.tcp.delayed_ack=0
net.inet.tcp.sendspace=65535
net.inet.tcp.recvspace=65535
net.inet.udp.recvspace=65535
net.inet.udp.maxdgram=57344
net.local.stream.recvspace=65535
net.local.stream.sendspace=65535
```

Отметьте, что значения этих переменных, в большинстве случаев, неочевидны и недокументированы. До того, как перезагрузить системы с новым файлом **/etc/sysctl.conf**, можно воспользоваться командой **sysctl** для просмотра значений по умолчанию:

```
# sysctl vfs.vmiodirenable
vfs.vmiodirenable: 0
```

Любое значение можно настроить и вручную, воспользовавшись ключом **-w**:

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

Полезные справочные руководства

Большое количество полезной информации содержится на странице **man tuning**. В этом документе собраны основные идеи по настройке системы, часть из которых была рассмотрена здесь, включая вопросы геометрии диска и то, как оптимизировать файловую систему в соответствии с требованиями к чтению и записи.

Страница **man tunefs** содержит подробное обсуждение команды **tunefs** для настройки не только механизма Soft Updates, но и других возможностей повышения производительности, которыми обладает файловая система FFS. И наконец, страница **man**

sysctl рассказывает о механизме переменных ядра и его использовании для изменения их значений на этапе исполнения.

Подготовка к худшему: резервные копии

Какие бы меры предосторожности вы ни принимали и каким бы опытным администратором ни были, рано или поздно вашу систему попробуют взломать, может произойти сбой оборудования или случится что-то еще, после чего система станет непригодной к использованию, а ее восстановление — невозможным. Что нужно сделать, чтобы свести к минимуму ущерб от подобной неприятности? Единственным средством может быть тщательная подготовка к тому, чтобы в один прекрасный момент придется устанавливать систему с нуля. Приготовьтесь к худшему, гласит закон Мэрфи, и оно никогда не случится. Предосторожность — это цена, которую приходится платить за безопасность.

Создание ключевых файлов

Чтобы созданная вами конфигурация не пострадала при какой-либо катастрофе в системе, необходимо сохранить ключевые файлы, которые позволят быстро настроить заново проинсталлированную систему. Их можно записать на компакт-диск, а при отсутствии таких возможностей — на дискету.

- **/etc/rc.conf**. Главный конфигурационный файл системы.
- **/etc/master.passwd**. Главная база данных пользователей. Все остальные базы данных можно генерировать на основе этого файла.
- **/etc/mail/myconfig.mc**. Основной конфигурационный файл системы **sendmail**, содержащий специальную конфигурацию.
- **/etc/fstab**. Важный файл для правильного воссоздания структуры диска.
- **/usr/local/etc/***. Отдельные конфигурационные файлы всех установленных программ.

Это наиболее важные файлы, идентифицирующие систему. В сумме они занимают несколько сотен килобайт, но именно они избавят вас от недель болезненных настроек, проб и ошибок при инсталляции и настройке системы.

Эти файлы можно запаковать в tar-архив и ежедневно высылать на определенный почтовый адрес (естественно, содержащийся *не* на FreeBSD-машине). Вот пример команды, необходимой для этого:

```
# tar cvfz - /etc/rc.conf /etc/master.passwd /etc/fstab /usr/local/etc |
uuencode seedfiles.tar.gz | mail -s "Seed Files" me@myaccount.com
```

Ее можно разместить в сценарии **/etc/periodic/daily** или **crontab** файле пользователя **root**.

Резервные копии

О важности создания резервных копий мы даже не будем говорить. Ни для кого не секрет, что регулярное создание резервных копий нудное, утомительное и дорогое занятие. Фактически, многие администраторы рассказывают о важности резервного копирования, не прибегая к нему сами.

Здесь мы рассмотрим различные возможности резервного копирования, чтобы в дальнейшем вам не помешало отсутствие необходимой информации.

Старым испытанным методом создания копий является встроенная в UNIX утилита **dump**. Ее выполнение определяется уровнями **dump**, заданными в файле **/etc/fstab** (см. главу 9). Программа позволяет создавать "инкрементные" резервные копии, когда в первый раз производится копирование всей системы, а затем копируются только измененные или новых файлов. Это позволяет сэкономить пространство на магнитной ленте и затрачиваемое время.

Чтобы воспользоваться этой возможностью, необходимо соответствующее устройство. Подойдет носитель на магнитной ленте (**/dev/rsa0** по умолчанию), внешний жесткий диск или любое другое устройство, которое можно монтировать и производить запись на него с помощью стандартных операций. Зачастую для таких устройств, как носители на магнитной ленте, в качестве интерфейса выбирался SCSI. Поэтому вам потребуется и SCSI-контроллер. Альтернативой могут послужить устройства FireWire (IEEE 1394). Кроме того, существуют носители на магнитной ленте с интерфейсом IDE/ATAPI. Прежде чем покупать устройство, убедитесь, что система FreeBSD поддерживает его! Для команды **dump** монтировать устройство не требуется, поскольку эта команда записывает информацию непосредственно на это устройство.

Следующая команда выполняет полное резервное копирование файловой системы **/home**:

```
# dump -0u -f /dev/nrsa0 /home
```

Опция **0** отвечает копированию на уровне 0 (т.е. копируются все файлы), а опция **u** указывает на запись в файл **/etc/dumpdates**. Это файл в текстовом формате, необходимый утилите **dump**, чтобы при "инкрементном" резервном копировании установить, какие файлы требуют сохранения.

ПРИМЕЧАНИЕ

Устройство **nrsa0** совпадает с **rsa0** за исключением того, что первое имя указывает на "неперематываемое" устройство. Это имя указывает программе, что по окончании записи ленту перематывать не нужно. Носители на магнитной ленте, в частности со SCSI-интерфейсом, используют имена устройства как опции, задающие утилите **dump** режим работы с лентой. См. табл. 20.3.

Таблица 20.3 Метаимена устройства в качестве опций управления

Метаимя устройства	*Действие*
/dev/sa0	Реальное имя устройства
/dev/rsa0	Указывает **dump** перемотать ленту к началу по завершении записи
/dev/nrsa0	Указывает **dump** не перематывать ленту к началу по завершении записи
/dev/ersa0	Указывает **dump** возвратить кассету из кассетоприемника по завершении записи

После этого можно производить "инкрементное" резервное копирование, используя тот же синтаксис команды, но более высокий номер уровня dump:

```
# dump -1u -f /dev/nrsa0 /home
```

Уровни **dump** позволяют копировать только те файлы, которые изменились с момента копирования на более низком уровне. Поэтому, если производится копирование на уровне 3, а затем на уровне 4, в последнем случае обрабатываются только те файлы, которые изменились с момента предыдущего копирования на уровне 3. Если затем произвести копирование на уровне 2, все файлы, обработанные на уровнях 3 и 4, будут скопированы снова. Чем ниже уровень, тем больше файлов будет скопировано. Уровень 0 отвечает глобальному резервному копированию, когда сохраняются все файлы, независимо от предыдущих сеансов резервного копирования.

Обычно применяют следующую процедуру: вначале создают резервную копию системы на уровне 0, затем в начале каждой недели — на уровне 1, используя новую ленту (или набор лент). Каждый день производится копирование более высокого уровня. Подробнее об этом см. **man dump**.

Восстановление

При восстановлении данных используется утилита **restore**. Она может выполняться в интерактивном или неинтерактивном режиме. Первый применяется для определенных файлов, например, если файл был случайно удален и его требуется восстановить. Вначале необходимо перейти в нужный каталог командой **cd**, а затем запустить программу **restore** в интерактивном режиме:

```
# restore -if /dev/nrsa0
```

В оболочке программы поддерживается набор команд: **cd**, **ls**, **pwd**, **add**, **delete** и др. Они позволяют перемещаться по структуре каталогов резервной копии, просматривать файлы и добавлять в список (командой **add**). Удалить файлы из списка (но не с ленты или диска) позволяет команда **delete**. Команда **extract** восстанавливает с ленты все файлы, находящиеся в списке на момент ее запуска. Полный синтаксис всех интерактивных команд программы **restore** содержится на странице справочного руководства **man restore**.

```
restore> add frank
restore> extract
```

Восстановить все файлы с ленты можно с помощью неинтерактивных опций:

```
# restore -rf /dev/nrsa0
```

На ленте часто содержится несколько сеансов резервного копирования, например для разных дней недели. Так, на одной ленте может содержатся полная еженедельная копия и шесть ежедневных инкрементных сеансов. Если требуется файл, отвечающий определенной копии, для его извлечения необходимо вначале правильно позиционировать ленту. Для это предназначена утилита **mt** (magnetic tape — магнитная лента).

После того, как кассета с лентой вставляется в привод, она автоматически перематывается к началу. Для перемотки, например, к третьему сеансу используется опция **fsf** (fast forward — быстрая перемотка вперед):

```
mt -f /dev/nrsa0 fsf 2
```

Параметр **fsf 2** указывает программе **mt** перемотать ленту на два сеанса вперед, т.е. к началу третьего. Для перехода ко второму сеансу следует вновь перемотать ленту к началу (командой **mt**) и воспользоваться опцией **fsf**:

```
mt -f /dev/nrsa0 rewind
mt -f /dev/nrsa0 fsf 1
```

С командами **dump** и **restore** связано гораздо больше особенностей, чем можно привести в этой книге. Они включают в себя: использование опции **-s** команды **restore** в неинтерактивном режиме для перехода к определенному сеансу копирования, дополнительные возможности утилиты **mt**, работа с несколькими лентами, удаленные устройства резервного копирования и многое другое. Хорошее руководство для начинающих, где рассказано о **dump** и **restore**, находится по адресу **http://www.nethamilton.net/dump.html**. Кроме того, обязательно ознакомьтесь со справочными руководствами команд **dump**, **restore** и **mt**.

Зеркальные сервера

Заниматься резервным копированием утомительно. Поэтому нет ничего удивительного в том, что постоянно создавать резервные копии могут лишь единицы. Кроме того, поддерживать резервные копии часто оказывается нецелесообразным и с финансовой точки зрения (особенно, в малом бизнесе). К счастью, существует альтернативное решение.

Добавив немного денег к той сумме, которая требуется для установки полноценной системы резервного копирования, можно создать зеркальный сервер — еще одну FreeBSD-машину, чьей единственной задачей будет ежедневная синхронизация файлов с основным сервером. Если с последним что-либо случиться, его можно будет мгновенно заменить зеркальным сервером. Для этого применяется уже известная нам утилита CVSup.

В идеале, второй сервер должен физически находиться где-нибудь в другом месте по отношению к первому. Система инкрементного резервного копирования CVSup позволяет разместить его в другом месте, даже если сетевая линия между машинами имеет узкую полосу пропускания. Рассмотрим, что требуется для настройки CVSup.

Предположим, что требуется создать зеркальные копии четырех каталогов: **/home**, **/usr/local/www**, **/var/mail** и **/etc**. В принципе, необходимо дублировать больше данных, но для примера этого будет вполне достаточно. Вначале необходимо настроить основной каталог, указывающий демону **cvsupd** (который устанавливается в каталог **/usr/local/sbin** как часть порта или пакета **cvsup**), какие данные пересылать клиентам, т.е. зеркальной машине.

Обладая полномочиями **root**, введите следующие команды:

```
# cd /usr/local/etc
# mkdir -p cvsup/sup
# cd cvsup/sup
# mkdir home www mail etc
```

Затем в каждом из четырех новых подкаталогов необходимо создать два файла: **.cvs** и **releases**. Эти файлы описывают "набор" для сервера **cvsupd** и включают его обслуживание. В подкаталоге **home** необходимо создать файл **home.cvs** со следующим содержимым:

```
upgrade home
rsymlink *
```

В файле **releases** разместите следующее:

```
home list=home.cvs prefix=/
```

Те же действия нужно проделать для остальных подкаталогов, изменяя лишь имя файла **.cvs** и соответствующую ссылку на него. После этого необходимо запустить сервер **cvsupd**:

```
# cvsupd -b /usr/local/etc/cvsup -C 1 -l /dev/stdout
```

Когда действия на основном сервере выполнены, переходим к зеркальной машине.

В каталоге **/etc** нужно создать файл **mirror-supfile** (или с любым другим именем). Его содержимое приведено на листинге 20.2, где **server1.hostname.com** соответствует имени хоста основного сервера.

Листинг 20.2 Пример конфигурационного файла CVSup (mirror-supfile)

```
*default host=server1.hostname.com
*default delete use-rel-suffix
*default compress
*default preserve

*default base=/usr
*default release=home
home

*default base=/usr/local
*default release=www
www

*default base=/var
*default release=mail
mail

*default base=/usr
*default release=etc
etc
```

Обратите внимание на последнюю запись: размещение зеркального каталога **/etc** происходит не в **/etc**, а в **/usr/etc**. Причины очевидны! Это лишний раз демонстрирует, какой гибкостью обладает CVSup при копировании набора каталогов в определенную точку файловой системы на зеркальном сервере.

После создания файла **mirror-supfile** все готово для первого сеанса. Для запуска процесса CVSup используется команда:

```
# cvsup -L 2 /etc/mirror-supfile
```

На экране появится обширный вывод, указывающий, какие файлы пересылаются и в каком режиме. Чтобы остановить сеанс можно воспользоваться комбинацией клавиш Ctrl+C. При этом CVSup корректно закончит работу и при следующем запуске начнет оттуда, где ее работа была прервана. Очевидно, что первый процесс пересылки будет длинным, ведь ему требуется полностью передать все сохраняемые данные с одного сервера на другой. Однако все последующие процессы будут быстрыми и эффективными.

Когда в каталог **/usr/etc** будут перенесены файлы с основного сервера, из этого каталога можно скопировать файл **master.passwd** и с помощью команды **pwd_mkdb** синхронизовать пользователей (как было показано выше в разделе "Переход от

Linux"). Если в системе имеется достаточно много пользователей или часто добавляются новые, подобную операцию можно производить автоматически. CVSup пропускает те файлы, для которых совпадает контрольная сумма и права владения. Если на основном сервере и зеркальной машине присутствуют разные учетные записи пользователей, информация о правах владения файлов может не совпасть! Если владельцем какого-то файла является некоторый пользователь, который отсутствует в базе данных второй системы, CVSup будет проводить копирование этого файла всякий раз при запуске процесса зеркалирования. Поэтому синхронизации баз данных пользователей позволяет ускорить процесс CVSup от 4–5 часов до 15 минут.

После того как процесс синхронизации серверов начнет выполняться гладко, его можно добавить в сценарий **/etc/periodic/daily** или файл **crontab** пользователя **root**. Подробный вывод процесса будет высылаться по электронной почте. Чтобы сократить его, нужно воспользоваться опцией **-L**.

На серверной стороне можно создать большее число наборов. Для этого достаточно создать требуемые каталоги внутри иерархии **/usr/local/etc/cvsup/sup** и добавить их в файл **mirror-supfile**. Процесс **cvsupd** перезапускать не нужно — он обработает новые наборы каталогов автоматически.

Важно помнить, что создание зеркальных серверов отнюдь не является полной заменой инкрементного резервного копирования. Это решение, подходящее для тех администраторов, кому не нужен аудит истории, контроль версий или внешние резервные архивы. Проще говоря, это не лучшее решение там, где FreeBSD используется как корпоративный сервер. Зеркалирование обеспечивает быстрое и простое решение на случай катастрофического сбоя системы. В серьезном бизнесе, к сожалению, не обойтись без резервных копий на ленточных носителях.

21 ГЛАВА

Знакомство с программированием на языке Perl

- Что такое Perl?
- Perl во FreeBSD
- Основы создания сценариев на Perl
- Простой сценарий на языке Perl
- Дополнительные возможности Perl
- Полезные ресурсы, посвященные Perl

Сегодня практически невозможно представить мир UNIX или Internet без языка Perl. Несмотря на некоторые структурные несоответствия и "причуды", Perl стал *де-факто* стандартным средством для быстрого написания гибких и многоцелевых программ. Код Perl не нужно компилировать, как программы на C, у него более свободный синтаксис, встроенная типизация данных и управление памятью. Таким образом, устранены многие недостатки традиционных языков программирования, что сделало Perl удобным средством даже для тех, кто не имеет опыта программирования на нем. Разумеется, из этого следует, что программы на Perl менее эффективны, чем на C или C++ (поэтому большинство компонентов FreeBSD написано на традиционных языках).

В этой главе читатель познакомится с основами языка Perl и сможет воспользоваться его достоинствами.

Что такое Perl?

Perl представляет собой интерпретируемый язык программирования, предназначенный, в первую очередь, для обработки текста. Он включен в состав FreeBSD и является основой для многих важных системных утилит. Его название означает "Practical Extraction and Report Language" — практический язык извлечений и отчетов (существует и несколько менее лестных расшифровок). Perl был создан Ларри Уоллом (Larry Wall) в качестве замены более ограниченного в возможностях языка **awk**. Ларри написал его, чтобы формировать отчеты. Он и по сей день является его главным архитектором и на момент написания этой книги работает над новой версией Perl 6.

О Perl написано множество книг. Его полезность беспримерна, применение — обширно, а структура — сложна, так что даже признанные гуру этого языка продолжают сталкиваться со скрытыми "причудами" или поражаться приемам, на которые способен Perl. В этой главе авторы даже не пытаются рассказать об этом языке программирования более или менее подробно, а рассматривают Perl в объеме, достаточном для написания простейших и понимания концепций более сложных (уже существующих) сценариев. Следуя по этому пути, читатель сможет изучить методы языка самостоятельно, на примерах, используя написанные на Perl утилиты FreeBSD или средства, найденные в Internet.

Perl во FreeBSD

Примерно десятая часть программ, составляющих основу инсталляции FreeBSD, написана на языке Perl. Остальные представляют собой двоичные файлы, скомпилированные из исходного кода на C, C++ или каком-либо другом популярном языке. Эти двоичные файлы, во-первых, не поддаются прочтению человеком, а во-вторых, не могут быть "декомпилированы", т.е. приведены к исходному состоянию. Код Perl не требует компиляции. Он существует в виде обычного текста, легко читаем и редактируем, выполняется *интерпретатором Perl* — **/usr/bin/perl**.

Мы видели в главе 13, как создаются сценарии командного интерпретатора **/bin/sh**. При этом имя интерпретатора обязательно указывается в первой строке сценария. Сценарии на Perl не являются исключением. Различие заключается в том, что сцена-

рии командного интерпретатора используются больше как "пакетные программы" (batch programs, как сказали бы в MS-DOS). В них последовательности команд запускаются так, как если бы они были введены в командной строке, используя простейшие методики замены переменных и управления потоком выполнения. А Perl представляет собой полнофункциональную среду программирования, позволяющую писать такие сложные программы, как Web-серверы или системы управления базами данных. Поскольку программы Perl являются интерпретируемыми, а не компилируемыми, они медленнее, чем такие же программы на C (особенно их запуск). Сильные качества Perl проявляются в обработке текста, поэтому идеальным его приложением являются, к примеру, такие сценарии, как **adduser**, где выполняются действия над строками и файлами, а не сложные математические вычисления.

В этом качестве Perl повсеместно используется во FreeBSD. Примерами утилит, написанных на нем, являются **adduser**, **pkg_version** и **sockstat**. Кроме того, он широко применяется и в сторонних приложениях, например, **majordomo** (это популярная утилита управления списками рассылки). Некоторые программы, разработанные как сценарии Perl, были затем переписаны на C из соображений скорости. Однако с программами на Perl вам придется сталкиваться не реже, чем со сценариями командного интерпретатора, поэтому важно знать, как они работают.

Сильные стороны Perl

Perl балансирует между простотой создания сценариев командного интерпретатора (высокоуровневый подход) и гибкостью программирования на C (низкоуровневый подход). В результате мы имеем язык, позволяющий использовать переменные разных типов, включая структуры данных, импортируемые объекты, ассоциативные массивы, многомерные массивы и обработчики ввода-вывода, не беспокоясь об их типизации, выделении и освобождении памяти, разыменовании указателей, создании прототипов функций и других обременительных задачах, связанных с программированием на C, равно как о компиляции двоичных файлов. Perl предоставляет чрезвычайную свободу, освобождая программиста от деталей низкоуровневого программирования. Именно высокоуровневые средства сделали Perl тем, чем он стал сейчас — предпочтительным языком CGI-программирования и Web-приложений серверной стороны. Эта сфера его применения стремительно выросла с ростом Internet и электронной коммерции, выросла в том числе и благодаря ему.

Специфической задачей Perl является выполнение операций над строками: поиск по шаблону, замена и перевод строк, регулярные выражения, а также структуры, облегчающие чтение и запись текстовых файлов с использованием естественных строковых массивов. В Perl нет необходимости объявлять переменную как принадлежащую строковому, целочисленному, символьному или какому-либо другому типу. Perl различает типы переменных по контексту их применения, выделяя для каждой из них необходимый объем памяти и выполняя операции над ней в соответствии с ее типом.

Perl предоставляет программистам одновременно простоту и мощь любого из широко используемых языков. Его синтаксис — чрезвычайно свободный и гибкий, он "не обращает внимания" на незначительные ошибки, сурово караемые в других языках. Кроме того, этот язык расширяем (в Internet можно найти тысячи библиотеч-

ных модулей), хорошо документирован и поддерживается на десятках самых разных платформ. Сценарий на Perl будет выполняться одинаково, где бы вы его ни запустили — в Linux, IRIX, Windows или во FreeBSD.

Слабые стороны Perl

Годы разработки с открытым кодом и инкрементные улучшения превратили Perl в нечто вроде бегемота. Его синтаксис едва ли можно назвать элегантным. Имея ряд преимуществ, Perl все же не является идеальной моделью реализации языка программирования.

Хотя свобода Perl позволяет программисту игнорировать многие типы ошибок, это зачастую имеет и обратный эффект, приводящий к разработке некорректного кода. Perl воспринимает синтаксис, повторяющий в некоторых случаях C, Бейсик, Фортран и некоторые другие языки. Хотя это наделило язык гибкостью, с другой стороны, это значит, что структура, лежащая в основе языка, слишком сложна, чтобы поддерживать некоторые важные аспекты узкоспециализированных языков. Во многом Perl — это "черная овца" среди языков программирования.

PYTHON И ДРУГИЕ ЯЗЫКИ, СЛЕДУЮЩИЕ ЗА PERL

После стремительного взлета Perl появилось немало проектов, стремившихся изничтожить его недостатки. Многие языки (REBOL, Java, Ruby и многие другие) находились в стадии разработки к тому моменту, когда Perl уже набрал "критическую массу" в мире серверного программирования. По пятам за ним следует язык, разработанный почти одновременно, — Python.

Python — язык еще более объектно-ориентированный, чем Perl, с жестким синтаксисом. Здесь не применяются скобки в стиле C, вместо них используется обязательная табуляция, которая значительно упрощает вложенность блоков управления потоком выполнения программы. Кроме того, Python поддерживает "точечную" иерархию объектов и строго структурированные модули (или "словари"), обеспечивая более прямой и расширяемый интерфейс к объектам, чем Perl.

Программы на Perl компилируются во внутренний код (который затем выполняется интерпретатором) при каждом запуске, а Python генерирует специальный код (файл **.pyc**) лишь при первом запуске программы, что сильно ускоряет все последующие. Частично такой подход является обязательным, поскольку все объектно-ориентированные программные среды являют собой обмен скорости на структуризацию. Python менее восприимчив к ошибкам, чем Perl, а программы на нем легче поддерживать. Однако ему недостает некоторых возможностей по обработке текста, то есть в той области, которая является самой сильной в Perl.

Хотя Python не является частью основной системы FreeBSD, его можно установить из набора портов **/usr/ports/lang/python**. Поскольку в прошлом Perl тоже инсталлировался из портов, а использование Python растет все быстрее, можно ожидать, что Python появится в **/usr/bin** уже достаточно скоро.

Основы создания сценариев на Perl

Сценарий Perl аналогичен сценариям командного интерпретатора, с которыми мы сталкивались в главе 13. Первая строка сообщает, какой интерпретатор следует использовать для обработки последующего содержимого файла:

```
#/usr/bin/perl
```

Оставшаяся часть сценария (как и в случае программирования на языке командного интерпретатора) состоит из операторов присвоения значений переменным, блоков управления потоком выполнения, циклов, системных вызовов, операций ввода-вывода и т.д. Например:

```
#!/usr/bin/perl
$string = "Hello world!";
$hostname = `hostname`;
if ($hostname eq "uncia") {
   print $string."\n";
   print `date`;
}
```

Обратите внимание на фигурные скобки в стиле C, выделяющие блок **if** вместо синтаксиса "**if/fi**" и "**case/esac**". Как и в C, каждый оператор заканчивается точкой с запятой (;), позволяя размещать в одной строке любое число операторов. Пустые символы между выражениями и операторами являются необязательными: "**$a = 1**" так же корректно, как и "**$a=1**". Синтаксис Perl ближе к C, чем к языку командного интерпретатора. Фактически, Perl можно рассматривать как гибрид этих двух языков, унаследовавший их лучшие черты.

Сценарий Perl нет необходимости компилировать, что чрезвычайно облегчает отладку: достаточно внести изменения в файл и запустить его снова, если вновь возникнут ошибки — снова отредактировать его и т.д. Интерпретатор Perl производит "компиляцию" на этапе исполнения. При этом он не сохраняет скомпилированный код. Поэтому Perl медленнее, чем полноценные программы на C, однако он вполне пригоден для сценариев, не требующих очень быстрого выполнения или супервысокой производительности (например, **adduser** или **majordomo**).

Чтобы запустить сценарий Perl, его необходимо предварительно сделать выполняемым посредством команды **chmod**:

```
# chmod +x myscript.pl
```

В результате будут установлены права доступа к файлу, равные **0755** (подробная информация о правах доступа — в главе 10):

```
-rwxr-xr-x  1 frank  frank  170 Jun 14  2000 myscript.pl*
```

Далее, чтобы запустить сценарий, необходимо указать префикс ./ поскольку каталог "." (т.е. текущий), скорее всего, не включен в пути:

```
# ./myscript.pl
Hello world!
Sat Apr 28 15:29:17 PDT 2001
```

Поскольку сценарии на языке Perl являются интерпретируемыми, их необязательно делать выполняемыми, можно даже не включать строку, указывающую интерпретатор. Хотя этот метод и применяется реже, но при желании сценарий можно запустить как аргумент самого интерпретатора **perl**:

```
# perl myscript.pl
```

Программа **perl** имеет несколько аргументов. Опция **-w** включает отображение предупреждений. Ей можно воспользоваться разными способами, например:

```
# perl -w myscript.pl
#!/usr/bin/perl -w
```

Переменные и операторы

Переменная в языке Perl является или *скаляром*, или *массивом*. Два этих вида переменных представляют собой два различных способа хранения данных. Эти данные

могут принимать форму числа (выраженного одним из нескольких способов), строки текста или других типов, называемых в Perl *"буквенными"* (*literals*). Имя скалярной переменной (наиболее распространенный вид) всегда начинается со знака доллара (**$**).

Замечательное свойство Perl состоит в том, что совершенно не нужно беспокоиться о том, к какому типу принадлежит число (целому, с плавающей точкой и т.д.). Кроме того, строку нет необходимости рассматривать как массив символов или указатель. Perl самостоятельно выполняет все необходимые операции. Более того, не нужно даже выполнять преобразование строк в числа и наоборот. Perl сам определит, что строка содержит только число и позволяет применять к ней математические операторы. Все остальное, начиная с первого нецифрового символа, будет отбрасываться, поэтому, например, **"123blah"** трактуется числовыми операторами как **"123"**.

Для изменения значений переменных используется набор операторов: математические (+, -, =), операторы увеличения в стиле языка C (++ и --), математические операторы с присваиванием (+=, -=, *=), возведение в степень (**), остаток от деления (%), операторы сравнения (>, <, =>, =<, == и !=), оператор конкатенации строк (.), оператор строкового повторения (x). Фактически, в Perl существует гораздо больше операторов, чем вам потребуется. В книгах, посвященных Perl, о них рассказано более подробно.

Вот несколько строк кода, демонстрирующих применение переменных, литералов (literals) и операторов:

```
$a = 5;
$a++;
$b = $a ** 2;
$c = "test" . $b;
print "$c";
```

В результате этот блок выводит значение **"test36"**. Если вам ясно, как это значение получено, то вы изучили основы языка Perl.

Скаляры, массивы и ассоциативные массивы

Переменные можно использовать по отдельности или в виде массивов определенного числа измерений. Мы уже сталкивались со скалярными переменными **$a**, **$b** и **$c** в предыдущем примере. Каждая из них содержит число или строку. Иногда возникает необходимость обрабатывать группы связанных между собой данных. Для этого применяются массивы:

```
@array1 = ("blah",5,12.7,$a);
@array2 = ($a, $b, $c);
```

К массивам применимы те же правила именования, что и к скалярам, за исключением того, что массивы обозначаются символом "at" (@), а не знаком доллара. Как видно из примеров, нет необходимости объявлять массив определенной длины или размещать в нем данные одного типа. Массивы могут содержать числа, строки, другие массивы и т.д.

Обратиться к элементу массива позволяют квадратные скобки. Третий элемент предыдущего массива **@array1** является скалярным значением и адресуется как **$array1[2]**. Помните, что нумерация элементов начинается с нуля!

Иногда элементы массива также имеют префикс @, а не $. Это значит, что адресуется определенный диапазон массива, например, **@array1[1,2]**. Это также массив, состоящий из двух элементов. Обозначение **@array1[2]** представляет собой диапазон из одного элемента, что совпадает со скалярным значением. Желательно, разумеется, использовать обозначение **$array1[2]**.

Существует несколько операторов для обработки массивов. Эти встроенные функции позволяют создавать массивы разными способами. Массивы часто называют "списками". В таком контексте к ним применимо понятие "стека" и набор операторов **push()**, **pop()**, **shift()** и **unshift()**. Они перечислены в табл. 21.1, вместе с результатами их действия на массив **@array1**.

Таблица 21.1 Операторы списка

Оператор	*Назначение*	*Синтаксис*	*Результат*
push()	Добавить значение к концу списка	push(@array1,"test");	@array1 = ("blah",5,12.7,6,"test")
pop()	Удалить значение с конца списка и возвратить его	$d = pop(@array1);	@array1 = ("blah",5,12.7,6), $d = "test"
unshift()	Добавить значение к началу списка	unshift(@array1,"test");	@array1 = ("test","blah",5,12.7,6)
shift()	Удалить значение из начала списка и возвратить его	$d = shift(@array1);	@array1 = ("blah",5, 12.7,6), $d = "test"

СОВЕТ

Следует отметить, что каждую из этих операций можно выполнить и несколько иначе, направив вывод оператора в новый (или тот же самый) массив. Например, оператор **@array3 = push(@array1, "test")** создает новый массив, оставляя исходный массив (**@array1**) без изменений.

Следующей полезной функцией при работе с массивами является **sort()**. Оператор **sort(@array1)** располагает элементы массива в лексикографическом порядке, рассматривая их как строки. Формат этой процедуры позволяет указать собственный алгоритм сортировки, что значительно расширяет возможности **sort()**. Например, если создать процедуру числового сравнения двух аргументов **numerically()**, сортировку всего массива можно осуществить командой **sort numerically (@array1)**.

Массивы особенно полезны при работе с реляционными данными — то ли с обычными текстовыми файлами наподобие **/etc/passwd**, то ли через интерфейс к реальным базам данных. Массив — это еще и способ доступа к отдельным строкам файла, прочитанным из стандартного входного потока. Далее мы рассмотрим, как это делается.

СОВЕТ

Чтобы узнать размер массива, нужно обратиться к последнему элементу в "скалярном контексте". Простейший способ состоит в присвоении списка скалярной переменной:

```
$size = @array1;
```

После этого значение **$size** равно 4.

Массив можно создать из скалярной строки посредством функции **split()**. Она разделяет строку на отдельные элементы, используя заданный символ-разделитель.

```
$mystring = "Test|my name|Interesting data|123";
@mydata = split(/\|/,$mystring);
```

Обратите внимание, что для указания символа-разделителя используются символы косой черты, а символ конвейера (|) экранирован символом обратной косой черты, чтобы отличить его от оператора выбора, о котором будет рассказано в разделе, посвященном регулярным выражениям. Массив **@mydata** теперь содержит в качестве элементов строки **"Test"**, **"my name"**, **"Interesting data"** и **"123"**.

Специальным типом массива является *ассоциативный массив* (*associative array*). Он аналогичен ассоциативной таблице (hash table), в которой данные сохраняются как пары ключ—значение. Префиксом ассоциативного массива является знак процента (%), но каждый элемент, естественно, представляет собой скаляр, поэтому при обращении к ним используется стандартный префикс **$**. Вот пример инициализации ассоциативного массива:

```
$assoc1{ key1}   = "value1";
$assoc1{ key2}   = "value2";
```

Теперь к массиву можно применить какой-либо из операторов обработки ассоциативных массивов:

```
@myvalues = values(%assoc1);
while ((@mykey,$myvalue) = each(%assoc1)) {
   print "$mykey -> $myvalue\ n";
}
```

Ассоциативные массивы особенно полезны в таких приложениях, как серверное CGI-программирование, когда все переменные HTML-формы пересылаются серверу и читаются программой в ассоциативный массив, ключами в котором служат имена полей формы.

Управление потоком

Полный набор структур управления потоком делает Perl полномасштабной программной средой, отличающейся от простого "пакетного" языка сценариев. Именно эти структуры позволяют применять итерации и создавать сложные ветвления в программах.

if/elsif/else

Наиболее распространенной управляющей структурой является блок **if**:

```
if ($a == 5) {
   print "It's 5\n";
} elsif ($a > 5) {
   print "Greater than 5\n";
} else {
   print "Must be less than 5\n";
}
```

ПРИМЕЧАНИЕ

Обратите внимание, что в условии (**$a == 5**) используется *оператор равенства* **==**, а не оператор присваивания **=**. Оператор **==** и другие операторы сравнения всегда применяются в ус-

ловных операторах. Если сравниваемые значения являются строками, а не числами, используются их строковые эквиваленты: **eq** вместо ==, **lt** вместо <, **ne** вместо != и т.д.

foreach

Другим широко распространенным оператором управления потоком в Perl является **foreach**. Он позволяет итерировать все элементы массива.

```
foreach $line (@buffer) {
   print $line;
}
```

Если имя необязательной переменной, которая принимает значения в цикле (**$line** в этом примере), опущено, то используется переменная по умолчанию **$_**, которая ссылается на текущий элемент. Если применяется несколько вложенных циклов **foreach**, переменные необходимо указывать, чтобы предотвратить путаницу.

for

Perl поддерживает стандартный цикл **for**, почти идентичный одноименному оператору в C. Назначение **for** состоит в итерации цикла определенное число раз, а не по элементам массива. Циклом **for** управляет итерационная переменная (как правило, не используемая больше нигде в сценарии), значение которой увеличивается автоматически при каждом проходе цикла. Аргументы оператора, как и в языке C, разделяются точкой с запятой: имя переменной, условие выхода из цикла и инкрементная операция:

```
for ($i; $i<100; $i++) {
   print "$i\n";
}
```

Этот пример печатает 100 строк, пронумерованных от 0 до 99. Первый аргумент устанавливает начальное значение переменной **$i**, второй — условие выхода из цикла (т.е. цикл выполняется до тех пор, пока это условие истинно), а третий указывает, что при каждом проходе цикла значение переменной должно увеличиваться на единицу.

while/until/do

И наконец, в Perl существует цикл **while**, действующий как упрощенная версия **for** без итерационной переменной. Его аргументом является условное выражение, значение которого определяется при каждом проходе цикла. Цикл выполняется до тех пор, пока условие истинно.

```
while ($i < 100) {
   $i += 5;
   $j++;
}
print "$j\n";
```

Цикл выполняется 20 раз и оператор **print** *выводит значение 20.*

Другим вариантом цикла подобного вида является **until**, смысл которого противоположен: он выполняется, пока условие ложно. Следующий цикл выполняет то же самое действие, что и приведенный выше:

```
until($i == 100) {
   $i += 5;
```

```
    $j++;
}
print "$j\n";
```

В некоторых случаях оказывается удобнее воспользоваться иным вариантом записи этих двух циклов: **do...while** или **do...until**. При такой записи цикл выполняется, как минимум, один раз поскольку значение условного оператора определяется в конце, а не в начале цикла:

```
do {
   $i += 5;
   $j++;
}  while ($i < 100);
print "$j\n";
```

Обратные кавычки (`) позволяют запустить любую команду в Perl-программе так же, как это делается в командной строке или в сценарии командного интерпретатора. Достаточно заключить команду в обратные кавычки и Perl выполнит ее, используя интерпретатор **/bin/sh** и ожидая окончания порожденного процесса. Вывод такой команды является возвращаемым значением, которое можно присвоить переменной, например:

```
$date = `date`;
```

Отметьте, что возвращаемая строка, как правило, завершается символом **\n**, поэтому для его удаления используется функция **chomp()** или вызываемая в следующем операторе, или окружающая исходное выражение, например:

```
chomp($date = `date`);
```

ПРЕДУПРЕЖДЕНИЕ

Важно понимать, что Perl не всегда "знает" пути поиска выполняемых файлов. Все может нормально работать на вашей машине, где разрабатывался сценарий, однако в другой системе он может и не выполниться корректно из-за невозможности найти необходимую системную команду. Поэтому желательно указывать полные пути:

```
@who = `/usr/bin/who`;
```

Аргументы командной строки

Программе Perl можно передать любое количество аргументов командной строки. Они разделяются пустым символом (если они не заключены в кавычки) и размещаются на этапе исполнения в массив **@ARGV**. Например:

```
# ./myscript.pl test "My String" 123
```

Таким образом, **@ARGV[0]** содержит значение **"test"**, **@ARGV[1]** — **"My String"** и **@ARGV[2]** — **"123"**. Это применимо и в CGI-программах, как мы увидим в главе 26. Если указать URL с аргументами, разделенными знаком + (как обычно принято в программах серверной стороны), массив **@ARGV** будет заполнен значениями по тому же принципу:

```
http://www.some-host.com/myscript.cgi?test+My%20String+123
```

Простой сценарий на языке Perl

Сценарий, приведенный в листинге 21.1, демонстрирует большинство из описанных выше методик, включая и некоторые новые. Он имеется и на прилагаемом компакт-диске в файле **simpledemo.pl**.

Листинг 21.1 Пример простого сценария на языке Perl (simpledemo.pl)

```
#!/usr/bin/perl

# <STDIN> представляет собой дескриптор ввода-вывода, указывающий на
# стандартный входной поток. В этом контексте он используется для чтения
# текста, вводимого пользователем в выдаваемом ему приглашении. Оператор
# chomp удаляет замыкающий символ новой строки/возврата каретки.
print "Please enter your name: ";
chomp ($name = <STDIN>);

srand;      # Инициализация генератора случайных чисел.
@namelist = ("Bob","Jane","Frank");
@colorlist = ("green","red","blue","yellow");
foreach $testname (@namelist) {
   # Выбрать случайный элемент массива позволяет оператор rand().
   $colors{$testname} = $colorlist[rand(@colorlist)];
}
while (($name,$color) = each(%colors)) {
   print "$name: $color\n";
   undef ($n);    # Функция undef() делает переменную неопределенной.
                  # Действует подобно NULL.
   do {
      $color = @colorlist[$n+1];
      $n++;
   } until ($color eq "blue");
}
```

Дополнительные возможности Perl

Мы познакомились с основами программирования на Perl, а сейчас перейдем к объяснению более сложных возможностей этого языка. Мы еще не сталкивались с доступом к файлам, функциями, модулями и даже с отличительной особенностью Perl — обработкой текста. К их рассмотрению мы сейчас и приступим.

Обработка текста

Регулярное выражение (их часто называют *regexp* — regular expression) — это способ указания шаблона для поиска в текстовом потоке. С их помощью можно не только производить простой поиск в строке, но и накладывать такие условия, как поиск в начале или конце строки, поиск групп определенных символов, строк заданной длины и т.д. Регулярные выражения составляют важную часть многих средств UNIX, включая, например, утилиту поиска и сравнения с шаблоном — **grep**, а также ее разновидности. Perl не только предоставляет ту же гибкость, которая присуща **grep**, но и включает ее в саму среду программирования. Именно этого и не хватает большинству языков программирования. Например, в C приходится копировать строки в память, производить посимвольное сравнение и т.д.

Регулярные выражения

Простейшим шаблоном регулярного выражения является строка текста. Для поиска в строке используется оператор =~, а само регулярное выражение заключается в символы косой черты:

```
if ($string =~ /abc123/) {   ...
```

Эту запись можно упростить, если строка текста уже получена, например, с помощью оператора чтения потока <>, который позволяют в цикле прочесть содержимое текстового файла, указанного как аргумент командной строки (об этом рассказано в следующем разделе). В этом случае строку содержит переменная **$_** (переменная "по умолчанию", с который мы познакомились в разделе с описанием оператора **foreach**), поэтому поиск можно производить неявно:

```
if (/abc123/) {   ...
```

Перейдем к более сложным шаблоном. Изменим предыдущий шаблон **"abc123"** так, чтобы он соответствовал только конструкции, находящейся в начале строки. Для этого используется символ ^: **/^abc123/**.

Существует символ и для поиска регулярного выражения в конце строки: **$**. В другом контексте он используется как префикс имени переменой. Используя два приведенных символа, можно создать шаблон, который соответствует строке, содержащей только конструкцию **"abc123"** и ничего больше:

```
if (/^abc123$/) {   ...
```

В этом заключаются основы создания шаблонов регулярных выражений. Можно также указать "класс" символов, например **[abc]**, включающий три буквы **a**, **b** или **c**. После любого символа, класса или группы можно использовать квантификаторы, указывающие на количество их вхождений в последовательность. Краткая сводка возможных шаблонов (далеко не полная) представлена в табл. 21.2.

Таблица 21.2 Синтаксис операторов регулярных выражений

	Текст
.	Любой одиночный символ
[abc123]	Любой из символов **"abc123"**
[^abc123]	Ни один из символов **"abc123"**
[a-g]	Все символы от **a** до **g** включительно
abc1|abc2	Оператор выбора: **abc1** или **abc2**
(abc123)	Группировка (для использования с квантификаторами, операторами выбора или обратными ссылками)
	Квантификаторы
?	*0 или 1 вхождение предыдущего текста*
*	0 или n вхождений предыдущего текста (n > 0)
+	1 или n вхождений предыдущего текста (n > 1)
*?	Заставляет производить поиск с квантификатором *, начиная с минимального соответствия

Квантификаторы	
{m}	Точно **m** вхождений предыдущего текста
{m,n}	От **n** до **m** вхождений предыдущего текста
{m,}	**m** или более вхождений предыдущего текста
Анкеры	
^	Анкер начала строки
$	Анкер конца строки
\b	Граница слова
\B	Не граница слова
Escape-коды	
\X	Экранирует (т.е. считает литералом, а не оператором) любой символ **X** (например, "."")
\r	Возврат каретки
\n	Новая строка
\f	Новая страница
\t	Табуляция
\d	Цифры (эквивалентно классу **[0-9]**)
\w	Символы, применяемые в словах (эквивалентно классу **[a-zA-Z0-9_]**)
\s	Пустой символ (эквивалентно классу **[\r\t\n\f]**)
\D	НЕ цифры
\W	НЕ слова
\S	НЕ пустой символ
\###	Символ с ASCII-кодом **###** (в восьмеричном представлении)
\cX	Символ Control+**X** (где **X** — любой символ)

ПРИМЕЧАНИЕ

При группировке в регулярном выражении наивысшим приоритетом обладают скобки, за ними следуют квантификаторы, анкеры и, наконец, оператор выбора.

Кроме того, к шаблону можно добавить различные опции после закрывающего символа косой черты. Они влияют на смысловую часть поиска. Например, **i** производит поиск без учета регистра: **/abc/i**.

Регулярные выражения могут быть сколь угодно сложными, отвечая решению самых разных задач в UNIX-среде.

Переводы

Конечно же, какой смысл имеет поиск без возможности замены? Perl поддерживает несколько встроенных операторов "перевода": оператор замены **s///**, оператор "транслитерации" **tr///** и функции обработки строк, например, **substr()**.

Для выполнения замен также используется оператор **=~**, но на этот раз он является оператором присваивания, а не сравнения. Его аргументами, фактически, явля-

ются оператор **s**, регулярное выражение, строка-заменитель и дополнительные опции. Все они разделяются символом косой черты:

```
$mystring =~ s/^test[0-9]/foo/g;
```

Вот более полезный пример, в котором угловые скобки преобразуются в соответствующие HTML-последовательности для отображения на странице:

```
$myhtml =~ s/</&lt;/g;
$myhtml =~ s/>/&gt;/g;
```

Последний символ **g** отвечает глобальной замене, когда все вхождения заменяются указанной строкой. Если он опущен, то заменяется только первое вхождение.

А что делать, если нужно сохранить число (**[0-9]**) из предыдущего примера? Именно для этого и предназначены группировки символов и обратные ссылки. Все, что в регулярном выражении помещено в скобки, можно повторить или сослаться на него по номеру. В регулярном выражении используется синтаксис **\#**, где **#** — номер группы символов, заключенной в скобки. Рассмотрим, например, регулярное выражение **/abc(123)(.*)\1\2/**, где **\1** интерпретируется как **123**, а **\2** — как **.***. Обратите внимание, что отсчет начинается не с 0, а с 1. В регулярном выражении можно использовать любое количество таких шаблонов.

Чтобы применить эти шаблоны при замене, за разделяющим символом косой черты на группы символов, заключенные в скобки, нужно ссылаться не как **\1** и **\2**, а как **$1** и **$2**. Эти специальные переменные, предназначенные только для чтения, остаются в памяти после нахождения совпадений, и ими можно воспользоваться в коде программы. Вот как выглядит предыдущий пример с сохранением числа:

```
$mystring =~ s/^test([0-9])/foo$1/g;
```

Итак, оператор замены обладает широкими возможностями, однако неудобен для таких функций, как, например, преобразование всех букв к одному регистру. Для решения подобных задач применяется оператор транслитерации **tr///**.

Оператор **tr** действует как упрощенная и ограниченная версия оператора **s**. Его аргументами являются две группы символов (не регулярные выражения, а просто группы символов или диапазон). Оператор преобразует символы первой из них в символы второй. Обратите внимание, что с командой **tr** используется оператор **=**, а не оператор соответствия **=~**. Например:

```
$mystring = "cat and dog";
$mystring = tr/abc/def/;
```

Теперь переменная **$mystring** содержит строку **fdt dnd dog**. Такое преобразование символов может принимать очень интересные формы, особенно, если новая строка короче старой. В этом случае оператор повторяет более короткий шаблон, чтобы получить одинаковое число соответствий. За последним символом косой черты можно указать опцию **d**, которая "выравнивает" шаблоны, удаляя лишние символы из *первого набора*.

Чаще всего, оператор **tr** применяется для преобразования строки к определенному регистру. Для этого в обеих частях оператора указывается диапазон символов:

```
$mystring = tr/A-Z/a-z/;
```

Таким образом, все символы строки **$mystring** преобразуются к нижнему регистру.

Одной из полезных функций обработки текста является **substr()**. О ее использовании подробно рассказано в руководстве по языку Perl. Эта функция возвращает подстроку заданной длины, находящуюся в исходной строке по указанному смещению. Например:

```
$mystring = "cat and dog";
$newstring = substr($mystring,0,3);
```

$newstring содержит строку "**cat**". Еще более удобной функциональности можно добиться от **substr()** в сочетании с **index()**. Последняя возвращает смещение, по которому в исходной строке находится указанная подстрока:

```
$mystring = "cat and dog";
$newstring = substr($mystring,index($mystring,"cat"),index($mystring,"dog"));
```

Переменной **$mystring** присваивается подстрока "**cat and** " (включая и замыкающий пробел).

Работа с файлами

Использование *файловых дескрипторов* (*filehandles*) позволяет открыть файл, прочесть его содержимое в массив, вывести данные в новый файл и т.д. Методы ввода-вывода применяются для ввода и вывода данных не только на консоль, а и в файлы.

Простейшим файловым дескриптором является оператор чтения потока (diamond operator) (хотя сам по себе он и не дескриптор). Это способ обращения к входному файлу (или набору файлов) из командной строки как файловому дескриптору (пока во входном потоке присутствуют строки). Этот оператор можно применить, например, в цикле:

```
while (<>) {
  print $_;
}
```

После этого необходимо запустить программу, указав в командной строке несколько аргументов — имен файлов:

```
# ./myscript.pl file1.txt file2.txt ...
```

Программа просто выведет содержимое всех файлов, как это делает команда **cat**. Это очень удобный и быстрый способ обращения к содержимому файла. Однако он значительно ограничен по сравнению с возможностями настоящих файловых дескрипторов, к рассмотрению которых мы сейчас перейдем.

По традиции имя файлового дескриптора состоит из прописных букв. Он создается командой **open()**. После этого он позволяет читать данные, выводить их и закрыть дескриптор. Вот пример того, как прочесть содержимое файла в массив:

```
open (FH,"/path/to/file1.txt");
@contents = <FH>;
close (FH);
```

Возможны ситуации, когда Perl не может открыть файл, например, если он не существует, права доступа к нему не позволяют этого или по иным причинам. Перехватить ошибки при открытии файла позволяет оператор **die**: если выполнение выражения закончится неудачей и управление перейдет к оператору **die**, он выведет аргумент (если он указан) в стандартный выходной поток и завершит работу сценария.

Вот наиболее распространенный способ использования этого оператора при открытии файлов:

```
open (FH,"/path/to/file1.txt") || die ("Can't open file1.txt!");
@contents = <FH>;
close (FH);
```

Запись в файлы представляет собой более сложную операцию, поскольку существует несколько способов выполнить ее. Важно помнить, что данные можно записать в любой поток, на который указывает дескриптор в командной строке. Они включают в себя операторы перенаправления > (перезаписать) или >> (добавить) и оператор конвейера |, выводящий данные на ввод другой программы. Сценарий может, например, вывести текст в электронное сообщение.

```
open (FH,">/path/to/file2.txt");
print FH $_ foreach (@contents);
close (FH);
```

```
open (MAIL,"| /sbin/sendmail -oi -t");
print MAIL "From: me\@somewhere.com\n";
print MAIL "To: you\@somewhereelse.com\n";
print MAIL "Subject: Check it out!\n\n";
print MAIL $_ foreach (@contents);
close (MAIL);
```

ПРЕДУПРЕЖДЕНИЕ

Помните, что символы @, используемые в текстовых строках (адресах электронной почты, например), необходимо экранировать символом обратной косой черты, чтобы Perl не рассматривал их как идентификаторы массивов. В противном случае сценарий завершит работу с ошибкой.

Файловый дескриптор является аргументом функции **print**. Важно понимать, что аргументом по умолчанию является встроенный файловый дескриптор **<STDOUT>** (стандартный выходной поток), если никакой другой не задан. Существует также и файловый дескриптор **<STDIN>**. Чтобы установить файловый дескриптор ввода/вывода по умолчанию, используется функция **select()**:

```
select (FH);
```

Теперь не нужно каждый раз использовать команду **print FH**. По окончании работы с потоком **FH** необходимо вновь объявить **STDOUT** потоком по умолчанию.

Для чтения листинга каталогов используются функции **opendir()** и **readdir()**. Чтобы открыть каталог и прочесть его содержимое в массив, можно воспользоваться следующим фрагментом кода:

```
opendir (DIR,"/path/to/dir");
@files = sort readdir (DIR);
closedir (DIR);
```

Овладев всеми изученными средствами, можно производить довольно интересные операции. Например, откроем файл **/etc/passwd**, прочитаем все записи, в которых идентификатор пользователя больше 1000, и выведем соответствующие имена учетных записей и полные имена пользователей.

```
#!/usr/bin/perl
open (PASSWD,"/etc/passwd") || die ("Can't open passwd file!");
@passwd = <PASSWD>;
close (PASSWD);
```

```
foreach (@passwd) {
   @userdata = split(/:/,$_);
   if (@userdata[2] > 1000) {
      print "@userdata[0]: @userdata[4]\n";
   }
}
```

Не правда ли, мы получили довольно удобное средство! Именно такие возможности и сделали Perl столь популярным языком программирования. Чтобы научиться писать программы, полезные для системного администрирования, требуется совсем немного усилий.

Функции

Perl имеет сотни встроенных функций, часть из которых мы уже видели. Они пригодны для решения большинства общих задач программирования. Эта функциональность может быть расширена путем подключения дополнительных модулей Perl. Однако со временем, когда вам понадобятся более сложные программы на Perl (особенно тогда, когда с серверной стороны используются целые наборы CGI-сценариев), вы сможете воспользоваться и собственными функциями (в Perl они называются *процедурами*).

Функции можно определять в любом месте сценария. Чтобы они работали, их даже не обязательно "декларировать".

Предположим, что функции требуется передать некоторое число аргументов, сумму которых она найдет. Для это применяется следующий синтаксис:

```
sub sum {
   $mysum += $_ foreach (@_);
   $mysum;      # Эта строка вычисляется как $mysum и устанавливает
                # значение, возвращаемое функцией
}
```

Тогда вызов функции будет выглядеть так (амперсант является префиксом имени функции):

```
$newsum = &sum(45,14,2134,89);
```

Переменная @_ указывает на список аргументов так же, как **@ARGV** представляет собой массив параметров, переданных программе в командной строке. Для определения функции более традиционного вида с определенным числом аргументов-переменных (подобный синтаксис используется в большинстве языков программирования), можно воспользоваться таким определением:

```
sub printname {
   ($name, $number, $passwd) = @_;
   print "$name/$number" if ($passwd);
}
```

Функции, как всегда, затрагивают тему глобального и локального пространства имен. В Perl нет локальных функций: все они определены глобально. Все переменные, определенные в функции, являются глобальными, если только не указано обратное (например, с помощью оператора **local()**). Массив @_ является локальным по определению. При каждом вызове функции создается новая копия ее аргументов. Используя функцию **local()**, то же самое можно проделать и с другими переменными, ограничив область их применения пределами функции:

```
sub sum {
   local($mysum);
   $mysum += $_ foreach (@_);
   $mysum;      # Эта строка вычисляется как $mysum и, таким образом,
                # устанавливает значение, возвращаемое функцией
}
```

Аналогичное действие выполняет и оператор **my**. На сегодняшний день он используется шире, чем другие методы. Для указания списка локальных переменных применяется следующий синтаксис:

```
my ($mysum, $name, $hash);
```

Если интерпретатор Perl запущен в "строгом режиме" (strict mode), он будет выдавать ошибки в случае, если локальные переменные не определены корректно внутри каждой функции, и не позволит воспользоваться переменными, которые не объявлены с помощью оператора **my**. Корректная работа с памятью не является обязательной, поскольку программы на Perl, как правило, выполняются недолго и интерпретатор осуществляет управление памятью самостоятельно. Тем не мене важно выработать хороший стиль программирования.

Модули Perl

Каждый хороший язык программирования имеет аналог разделяемых библиотек, и Perl также не исключение. Фактически, библиотеки Perl (называемые *модулями*) представляют собой блоки невыполняемого Perl-кода с расширением **.pm** (сокращение от Perl module — "модуль Perl"). Они "росли" по мере распространения в Internet так же, как это было, например, с набором портов FreeBSD. Порты и модули Perl взаимосвязаны. Об этом вскоре и пойдет речь.

.pm файл с кодом Perl (например, **mylib.pm**) можно разместить в том же каталоге, что и сценарий, а затем воспользоваться оператором **use** (расширение **.pm** следует опустить):

```
use mylib;
```

Во FreeBSD Perl устанавливается в каталог **/usr/lib/perl5**. Однако модули Perl в него не устанавливаются. Их можно инсталлировать при необходимости, и они будут размещены в каталоге **/usr/local/lib/perl5**. Он содержит два подкаталога: один из них предназначен для руководств **man** (его именем является номер текущей версии Perl), а второй — непосредственно для модулей (**site_perl**). В этом подкаталоге (на один уровень ниже) находятся различные каталоги, в которых сгруппированы установленные модули. Кроме того, здесь же присутствует каталог **i386-freebsd**, содержащий прекомпилированный код на языке C, который используется некоторыми модулями Perl для повышения производительности (например, в сложных математических алгоритмах).

Модули собраны в группы и именуются следующим образом: префикс, двойное двоеточие (::) и имя модуля. Такой формат используется в C++. Например, **Net::Telnet** это название модуля Perl, содержащего средства для работы с протоколом Telnet, а **Net::DNS** — функции для поиска имен серверов. Два этих модуля находятся в каталоге **/usr/local/lib/perl5/site_perl/5.005** в подкаталоге **Net** в файлах **Telnet.pm** и **DNS.pm**.

Этот каталог включен в пути поиска Perl. Для использования модуля в сценарии достаточно воспользоваться оператором **use**:

```
use Net::Telnet;
```

Теперь любой функцией из этого модуля можно пользоваться так же, как если бы она была определена в самом сценарии, то есть указав перед ее именем амперсант **&**.

Как же определить, какие функции присутствуют в модуле? Для этого существует утилита **perldoc**. Она работает подобно **man**, и, если модули установлены правильно (например, из набора портов), ознакомиться с документацией можно, указав в командной строке имя модуля. Вот примеры документации для модуля **Image::Size** (см. листинг 21.2):

Листинг 21.2 Пример документации для модуля Perl

```
# perldoc Image::Size
Image::Size(3) User Contributed Perl Documentation Image::Size(3)
NAME
        Image::Size - Image::Size — чтение размеров изображения (заданного
                      в одном из популярных форматов)
SYNOPSIS
                use Image::Size;
                # Получить размеры рисунка globe.gif
                ($globe_x, $globe_y) = imgsize("globe.gif");
                # Предположим, что в последующих примерах X=60 и Y=40
                use Image::Size 'html_imgsize';
                # Получить размеры в формате "HEIGHT=X WIDTH=Y" для создания
                # HTML-кода
                $size = html_imgsize("globe.gif");
                # $size == "HEIGHT=40 WIDTH=60"
                use Image::Size 'attr_imgsize';
                # Получить размеры как список, пригодный для передачи
                # процедурам в модуле CGI.pm
                @attrs = attr_imgsize("globe.gif");
                # @attrs == ('-HEIGHT', 40, '-WIDTH', 60)
                use Image::Size;
                # Получить размер внутреннего буфера в памяти
                ($buf_x, $buf_y) = imgsize($buf);
```

В документации такого типа, как правило, содержится полезный и точный прототип кода, который можно использовать в сценариях, а также полный список функций модуля.

Модули Perl и набор портов

Что же представляет собой "корректный" способ установки модулей Perl? Разумеется, это набор портов! Перейдите в каталог **/usr/ports** и просмотрите листинг. Обратите внимание на то, что многие порты имеют префикс **p5-**. Это и есть модули Perl, которые были превращены в соответствующие порты FreeBSD. (Существуют и пакеты с этими модулями.) Многие модули содержат скомпилированные компоненты на C, а также вспомогательные модули и документацию, поэтому необходимо, чтобы все было установлено корректно. Проследить за этим процессом и позволяют порты.

Например, **/usr/ports/net/p5-Net-Telnet** представляет собой порт модуля **Net::Telnet**. Такой же схеме именования подчиняются и другие порты модулей: дефис вместо двойного двоеточия в имени каталога. Некоторые категории портов содержат десятки модулей Perl. Все они добавлены в порты потому, что неоднократно подтвердили свою полезность. Такая распределенная модель позволяет Perl быть бесконечно расширяемым, оставаясь, в то же время, простым в своей исходной инсталляции.

Для установки модуля из соответствующего порта необходимо поступить так же, как при инсталляции любого другого порта: перейти в каталог командой **cd** и выполнить команды **make** и **make install**. Модули Perl имеют встроенную цель **make test**, которая позволяет определить, насколько хорошо модуль будет работать в системе. Эта цель неявно выполняется другими целями **make**.

Для проверки установленных модулей Perl используются команды **pkg_info** и **pkg_version**. Это проще, чем просматривать каталог **/usr/local/lib/perl5**. Аналогично работают и другие пакетные средства: **pkg_add** позволяет добавить модуль из tar-архива, а **pkg_update** — обновить его новой версией.

Полезные ресурсы, посвященные Perl

Поскольку Perl столь широко распространен, не удивительно, что существует множество ресурсов, посвященных ему. Рассмотрим некоторые централизованные и "официальные" источники информации о Perl.

Web-сайты

Если вам требуется руководство по Perl, в первую очередь следует обратиться к Internet. Он не настолько прямолинеен, как книга, однако почти на все вопросы здесь можно найти ответ.

www.perl.org

www.perl.org — это Web-сайт Perl Mongers (Продавцы Perl). Представляет собой независимый источник информации и пропагандистских рекомендаций. На нем перечислены списки рассылки, к которым можно присоединиться, и различные ресурсы, как, например, **www.perldoc.com**, централизованная база данных документации по Perl.

www.perl.com

"Официальный" Web-сайт Perl, поддерживаемый издательством O'Reilly. Здесь собраны новости, обучающие руководства и обсуждения различных способов применения Perl. Это, вероятно, наиболее полный сайт, посвященный текущим событиям в сфере Perl, но не самое лучшее справочное руководство по этому языку.

Книги

Каждому, кто программирует на Perl, рекомендуется иметь одну из справочных книг по Perl издательства O'Reilly. Они представляют собой легкие в использовании справочники, с которыми ничто не сравнится.

Книга с верблюдом ("Camel" Book)

Программирование на Perl (*Programming Perl*) — это полное справочное руководство по этому языку, написанное Ларри Уоллом (Larry Wall) (автором Perl), Томом Кристиансеном (Tom Christiansen) и Йоном Орвантом (Jon Orwant) (издательство O'Reilly, 2000). На обложке книги изображен верблюд — логотип Perl. Эта книга часто переиздается, чтобы отразить самые современные разработки в мире Perl, и считается "основной" книгой по этому языку. Одним из основных ее достоинств является список функций в алфавитном порядке.

Книга с ламой ("Llama" Book)

Книга *Изучение Perl* (*Learning Perl*), авторами которой являются Рэндал Л. Шварц (Randal L. Schwartz) и Том Феникс (Tom Phoenix) (издательство O'Reilly, 2001), представляет собой "сестру" книги с верблюдом. На обложке изображена лама. В книге достаточно подробно изложены основы Perl (примерно то, о чем было рассказано в этой главе). Книга написана в "обаятельном" стиле и охватывает достаточно много тем, необходимых для хорошего знакомства с языком Perl.

Словарь разработчика на Perl (Perl Developer's Dictionary)

Словарь разработчика на Perl (*Perl Developer's Dictionary*), написанный Клинтоном Пирсом (Clinton Pierce) (издательство Sams, 2001), представляет собой исчерпывающее руководство по всем функциям Perl. Это одна из наиболее полных книг по этому предмету на сегодняшний день. Она предназначена для программистов на Perl, которым больше требуется руководство по синтаксису и применению, чем учебник по Perl.

CPAN

Сеть *CPAN* (*Comprehensive Perl Archive Network, Обширная сеть архивов Perl*) содержит большинство модулей. Кроме того, в ней можно найти двоичные дистрибутивы, сценарии и другие средства, а также исходный код интерпретатора. Она полезна своим обширным списком модулей. Если вы захотите установить модуль, отсутствующий в наборе портов, его (а также полную документацию по нему), скорее всего, можно будет найти именно здесь.

Центральный сайт CPAN находится по адресу **http://www.perl.com/CPAN-local/**. Цель CPAN состоит в поддержке большого числа зеркальных серверов, расположенных по всему миру и обеспечивающих быстрый доступ к их ресурсам.

FreeBSD и работа в сети

- Основы компьютерных сетей
- Настройка основных сетевых служб
- Подсоединение к Internet с помощью PPP
- Настройка почтовых служб
- Конфигурирование Web-сервера
- Настройка FTP-сервера
- Конфигурирование Internet-шлюза
- Сетевая безопасность
- DNS-сервер
- Сетевая файловая система NFS
- Совместное использование ресурсов с Microsoft Windows
- DHCP-сервер

22 ГЛАВА

Основы компьютерных сетей

- Введение в сети
- Топологии сетей
- Сетевые компоненты
- Сетевые протоколы
- Протокол TCP/IP
- IP-адреса
- Подсети и сетевая маска
- Маршрутизация
- Имена компьютеров и доменные имена
- DHCP

Были времена, когда существовало два типа операционных систем — специально спроектированные для работы в сети и не поддерживающие сеть в принципе. К первой категории относились в основном UNIX-подобные операционные системы для мэйнфреймов (т.е. для больших машин), например, AIX, IRIX, Digital UNIX, BSDI, а также еще более специализированные системы типа Novell NetWare и VMS. Вторая категория включала настольные операционные системы пользовательского уровня, такие как Windows и Mac OS. Вообще говоря, эти два типа операционных систем имели мало общего, однако главное отличие состояло в том, что сетевые операционные системы были многопользовательскими, со встроенными в ядро сетевыми возможностями и удаленным доступом.

Сейчас картина в корне изменилась. В наши дни надежность и сетевые функции однопользовательских настольных систем превышают по возможностям многие старые операционные системы класса мэйнфреймов. Mac OS X (с архитектурой FreeBSD и ядром Mach) и Windows 2000/XP — вот примеры платформ, фундаментальным принципом которых является работа в сети. Они призваны заменить старые, несетевые системы. Стек сетевых транспортных протоколов TCP/IP, разработанный для Internet, встроен в ядро этих систем точно так же, как это было в мэйнфреймах и как это есть во FreeBSD.

Что касается FreeBSD, то это система, изначально спроектированная как сетевая платформа. И независимо от того, работает ли она на рабочей станции или на сервере, в сети она наверняка используется эффективно. А вот FreeBSD-компьютер, не подключенный к сети, работает, откровенно говоря, вполсилы.

В данной главе будут описаны принципы построения компьютерных сетей применительно к FreeBSD, мы также расскажем о ее роли в развитии Internet.

Введение в сети

Сеть — это способность двух и более компьютеров узнавать о существовании друг друга и обмениваться между собой данными. В век Internet сеть может предложить буквально все: и отправку электронной почты, и HTML-страницы с информационного сервера, и большие мультимедийные файлы (типа видеофильмов), и музыку в mp3-формате. Таким образом, сферы применения сетевого компьютера значительно шире, чем предполагает тот комплект программ, который имеется в распоряжении пользователя. Появление сетей изменило способы ведения бизнеса и породило новые виды развлечений. По сути, Internet стал движущей силой компьютерной революции.

В наши дни почти все действующие компьютерные сети построены на основе стандартного стека (набора) протоколов TCP/IP. Набор протоколов TCP/IP, структура которого детально рассматривается далее, прошел почти тридцатилетний путь развития. Универсальность и модульность данного стандарта способствовали его признанию и успешному распространению. Появление первой компьютерной сети ARPAnet (проект министерства обороны США), объединившей четыре компьютера в четырех разных западноамериканских университетах, датируется 1969 годом. Но протокол *TCP/IP начинает свою историю только с 1974 года, с того самого момента*, когда увидел свет документ под названием "Протокол для пакетной передачи данных по сети", написанный пионерами Internet — Винтом Серфом (Vint Cerf) и Бобом Каном (Bob Kahn). Вначале это была просто программа управления передачей данных. Позднее она превратилась в два протокола — TCP (Transmission Control Protocol —

протокол управления передачей) и IP (Internet Protocol — межсетевой протокол). Этот набор протоколов дал миру надежный способ отправки и приемки данных по сетям с пакетной коммутацией, чья полезность и пригодность находились в то время в процессе изучения. Затем, в 1977 году, в лаборатории Bell Labs были реализованы первые UNIX-системы, которые сыграли большую роль в популяризации этого стека протоколов. В его состав к этому времени вошли новые структурные уровни, представленные протоколами *UDP* (User Datagram Protocol, протокол пользовательских дейтаграм) и *ICMP* (Internet Control Message Protocol, протокол управления сообщениями в сети Internet).

Стек протоколов TCP/IP формирует базовую структуру Internet и определяет все ее атрибуты. Данный стандарт пока что является единственно возможным способом реализации глобальных сетей, чьи магистрали соединяют между собой города (WAN). И он же в большинстве случаев используется при построении локальных сетей (LAN), которые, в принципе, можно реализовать и на более простых протоколах. Так, многие предприятия и университеты используют высокоскоростные сети на базе Ethernet. Протокол TCP/IP — многоуровневое коммуникационное решение.

Модель архитектуры TCP/IP рассматривает обмен данными как совокупность четырех наборов взаимозависимых процессов. Эти процессы сгруппированы в следующие 4 уровня: уровень приложений, транспортный уровень (взаимодействие хост — хост), межсетевой уровень (Internet) и уровень доступа к сети. Примерами протоколов, поддерживаемых на уровне приложений, служат FTP, Telnet, NFS и DNS. Протоколы TCP и UDP соответствует транспортному уровню. Протоколы межсетевого уровня — это IP (Internet Protocol) и RIP (Route Information Protocol). Уровень доступа к передающей среде представлен драйверами устройств. Далее рассматривается, как все эти протоколы укладываются в понятия TCP/IP-структуры и как с ними оперирует FreeBSD.

Топологии сетей

Различные сети отличаются друг от друга структурой, логическим и физическим размещением устройств, механизмами передачи данных и т.д. — вариантов очень много. Есть несколько типичных способов подключения FreeBSD-компьютера к сети: это хост в университетской сети, это узел компьютерной сети корпорации и, наконец, домашний компьютер, подсоединенный к Internet по коммутируемой линии (альтернативный вариант — выделенная линия).

На рисунке 22.1 изображена топология типичной университетской сети, однако эта же диаграмма может описывать и работу корпоративной или домашней сети.

Любой из вариантов реализации подобной схемы подразумевает подключение к глобальной сети по линиям связи T1, T3, DSL (или же через модемные соединения). Пропускная способность глобальных соединений колеблется в широком диапазоне — от килобит в секунду для модемных соединений (33-56 Кбит/с) до гигабит в секунду для соединений по стандартам OC-48 и OC-192, применяемых телекоммуникационными операторами. Основные типы глобальных соединений и их характеристики приведены в таблице 22.1.

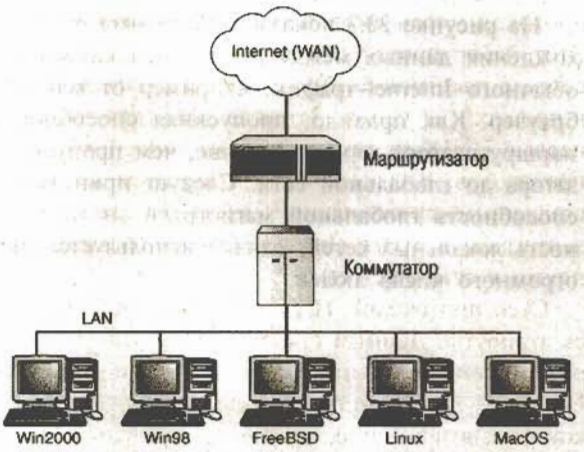


Рисунок 22.1
Топология сети с разнотипными компьютерами, включающей как локальный, так и глобальный участки, разделенные маршрутизатором.

Таблица 22.1 Типы последовательных глобальных соединений и соответствующие им скорости передачи данных.

Типы соединений	*Скорость передачи данных*	*Примечания*
Модем	До 56 Кбит/с	
ISDN	56 — 128 Кбит/с	
DSL	192 Кбит/с — 1,5 Мбит/с	
Кабельный модем	1 — 5 Мбит/с	
T1	1,5 Мбит/с	Соответствует 24 линиям DS0
T3	43,2 Мбит/с	Соответствует 28 линиям T1
OC-3	155 Мбит/с	Соответствует 100 линиям T1
OC-12	622 Мбит/с	Соответствует 4 линиям OC-3
OC-48	2,5 Гбит/с	Соответствует 4 линиям OC-12
OC-192	9,6 Гбит/с	Соответствует 4 линиям OC-48

Подобные соединения разработаны для передачи данных на большие расстояния. Кроме того, эти стандарты, благодаря их последовательной "природе", вполне могут быть реализованы на базе инфраструктуры телефонной сети, будь то обычные медные кабеля или современные оптико-волоконные каналы.

Другим типом сетевой топологии является модель сети Internet-провайдера, которая включает в себя магистральную сеть, подобную уже рассмотренной выше и, кроме того, множество модемных подключений к этой сети, осуществляемых конечными пользователями. Эта топология показана на рисунке 22.2.

Когда глобальный сетевой трафик проходит через DSU (Data Service Unit, устройство обработки данных) или CSU (Central Switching Unit, центральное устройство коммутации) и маршрутизатор, происходит его преобразование, дабы он соответствовал транспортным механизмам локальной сети, например Ethernet. Физическая реализация связи в локальной сети позволяет компьютерам, подключенным к сети, общаться между собой.

На рисунке 22.3 показана диаграмма пропускной способности сети на пути прохождения данных между двумя локальными сетями. Эта схема отражает передачу обычного Internet-трафик, например от веб-сайта до клиента, использующего веб-браузер. Как правило, пропускная способность канала на участке от клиента до маршрутизатора гораздо меньше, чем пропускная способность участка от маршрутизатора до глобальной сети. Следует принимать во внимание, что хотя пропускная способность глобальной магистрали значительно превосходит пропускную способность локальных сетей, однако используется она для одновременного обслуживания огромного числа людей.

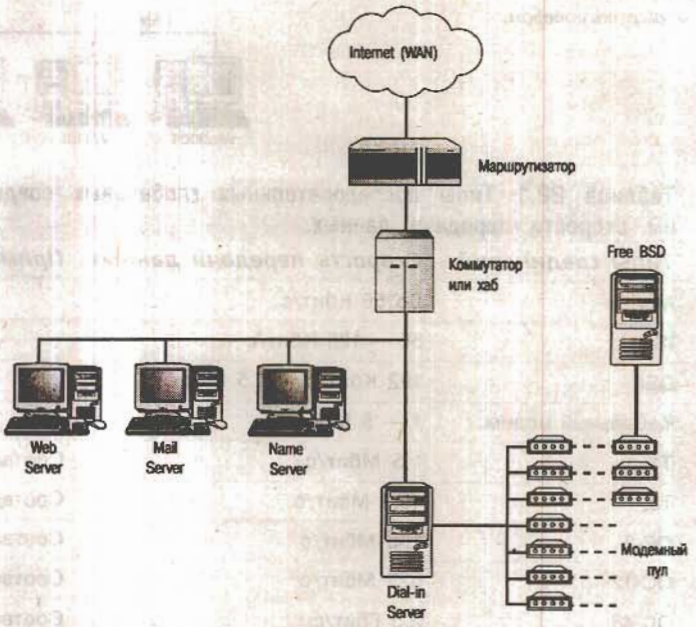


Рисунок 22.2
Сетевая топология, соответствующая модели Internet-провайдера: домашние пользователи соединяются с сетью провайдера посредством коммутируемой сети, а сеть провайдера подключена к глобальной сети Internet посредством скоростных каналов типа T1.

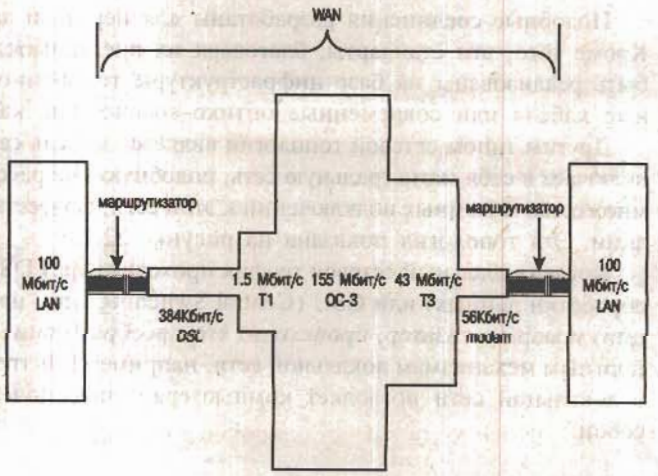


Рисунок 22.3
Диаграмма изменений пропускной способности на пути от сервера Internet-провайдера до локальной сети клиента через глобальные сети различной производительности.

Сетевые компоненты

В данном разделе рассмотрены различные сетевые аппаратные средства. Правильное понимание их назначения и функционирования поможет построить сеть наиболее оптимально. Начнем, пожалуй, с сетевых кабелей.

Кабели

Основное преимущество локальных Ethernet-сетей состоит в том, что при низкой стоимости они обеспечивают скорости передачи данных, достижимые только в глобальных магистралях: 10 Мбит/с в стандартном Ethernet, 100 Мбит/с в большинстве современных локальных сетей (Fast Ethernet) и даже 1 Гбит/с в сетях на базе самого нового стандарта Gigabit Ethernet. Эти скорости могут быть достигнуты с помощью дешевых кабелей и сравнительно недорого оборудования.

Обратной стороной дешевизны и скорости Ethernet-соединений является невозможность их использования на больших расстояниях (максимум в пределах соседних зданий). Это объясняется тем, что данные при передаче не усиливаются специальным оборудованием, а сигнал при передаче по кабелю затухает, особенно это касается Ethernet-кабелей, как тонкого, так и толстого. Сводная информация по кабелям для различных видов Ethernet приведена в таблице 22.2.

Таблица 22.2 Различные типы Ethernet-кабеля

Тип кабеля	*Название сети*	*Максимальная скорость передачи данных, Мбит/с*	*Кабель*	*Тип коннектора*	*Максимальная длина сегмента сети*
10Base-2	Тонкий Ethernet	10	Тонкий коаксиал	BNC	185 м
10Base-5	Толстый Ethernet	10	Толстый коаксиал	DB-15, DIX/AUI	500 м
10Base-T	Ethernet	10	UTP Категории 3, 5	RJ-45	100 м
100Base-TX	Fast Ethernet	100	UTP Категория 5	RJ-45	100 м
1000Base-T	Gigabit Ethernet	1 Гбит/с	UTP Категория 5	RJ-45	100 м

ПРИМЕЧАНИЕ

Некоторые широко используемые аббревиатуры

DIX: Digital, Intel, Xerox

AUI: Attachment Unit Interface — интерфейс устройства подключения

BNC: Bayonet Nail Concelman — штыревой разъем с байонетным замком, или British Naval Connector — британский военно-морской коннектор

UTP: Unshielded Twisted Pair — неэкранированная витая пара

Архитектура Ethernet ориентирована на шинную топологию. Это значит, что все узлы сети подключаются к единой магистрали. Максимальное количество узлов, которые могут быть подключены к одному магистральному кабелю, зависит от типа

используемого кабеля. На каждом конце магистрального сегмента должен быть установлен терминатор (сопротивлений в 50 Ом), имеются ограничения на число сегментов сети и их длину. Сети, построенные на толстом коаксиале, кроме того, весьма громоздки. В наши дни более распространенным является сети на витой паре, ввиду относительной дешевизны и простоты их создания. Сети Ethernet на витой паре строятся по топологии "звезда", при которой сетевая карта каждого компьютера соединена непосредственно с хабом. Более того, витая пара в отличие от коаксиала намного проще в монтаже благодаря большей гибкости кабеля и простым "телефонным" коннекторам. До недавних пор сетевые карты выпускались с разъемами для подключения разных типов кабеля, но теперь становится все труднее найти сетевую карту, рассчитанную на что-либо другое, а не на стандартный RJ-45 (разъем под витую пару).

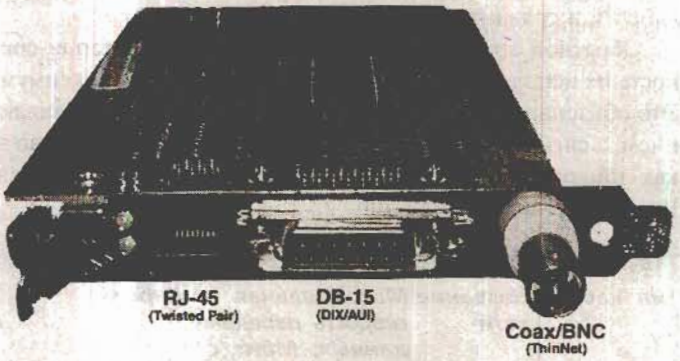


Рисунок 22.4
Сетевая карта с разъемами для трех типов кабеля: витой пары (RJ-45), толстого (DB-15) и тонкого (BNC) Ethernet.

Такая стандартизация связана со снижением стоимости построения локальных сетей на витой паре и ростом их популярности. Сегодня из BNC-коннекторов и терминаторов чаще создаются загадочные металлические скульптуры, а толстый коаксиал годится разве что для рукопашного боя.

Витая пара стоит недорого и может применяться практически везде. Сам кабель, коннекторы RJ-45 и обжимной инструмент доступны в любом магазине электронного оборудования. Информация о раскладке контактов витой пары и их правильном обжиме дана в следующем разделе.

Прямое и перекрестное соединения

Существует две схемы подключения разъема RJ-45 к кабелю — для получения *прямого* и *перекрестного кабеля* (*Straight-through* and *Crossover Cables*). Схемой разводки называется порядок подключения проводов кабеля к контактам разъема. В чем разница между прямым и перекрестным подключением? В прямом кабеле на обоих концах коннекторы подключены идентично, а в перекрестном — по-разному.

Устройства с коннекторами RJ-45 подразделяются на два типа: компьютерные и хабы (концентраторы). К компьютерным устройствам относятся компьютеры, маршрутизаторы и другие конечные устройства, а к хабам — концентраторы и коммутаторы. Чтобы подключить компьютер к хабу, нужен прямой кабель. Он же необходим для соединения компьютера с коммутатором, или маршрутизатора с хабом. Это соединения разнородных по своей природе устройств. А вот для подключения друг к другу однородных устройств, например для соединения между собой двух хабов (с целью увеличения числа портов) или хаба с коммутатором, а также для соединения двух

компьютеров необходим перекрестный кабель. Правило простое: прямой кабель нужен для соединения разнородных устройств, а перекрестный — для однородных.

Исключением из правила является только uplink-порт, имеющийся на хабах и специально предназначенный для подключения одного хаба к другому. Этот порт подключают к обычному порту другого хаба прямым кабелем. Это необходимо в больших сетях, где длинные кабели подключаются к многопортовым концентраторам или коммутаторам. Эти кабели уходят сквозь стены в другие офисы, и их нельзя подключать прямо к стандартным портам небольших хабов. Как правило, эти кабели должны быть согласованы с адаптером и оборудованы коротким перекрестным кабелем для подключения к небольшому хабу. В подобных случаях замена прямого соединительного кабеля на перекрестный, мягко говоря, не практична. Когда хабы обзавелись uplink-портами, необходимость подключения их друг к другу через обычные порты перекрестным кабелем отпала сама собой. Тем не менее помните, что соединение двух хабов посредством их uplink-портов по-прежнему требует перекрестного кабеля.

А как определить, какой из кабелей является прямым, а какой — перекрестным? Согните кабель так, чтобы его концы оказались рядом друг с другом. Если цветовая последовательность обжатых проводов кабеля в обоих разъемах совпадает, то это прямой кабель, в противном случае — перекрестный. Разводка прямого кабеля показана на рис. 22.5, перекрестного — на рис. 22.6.

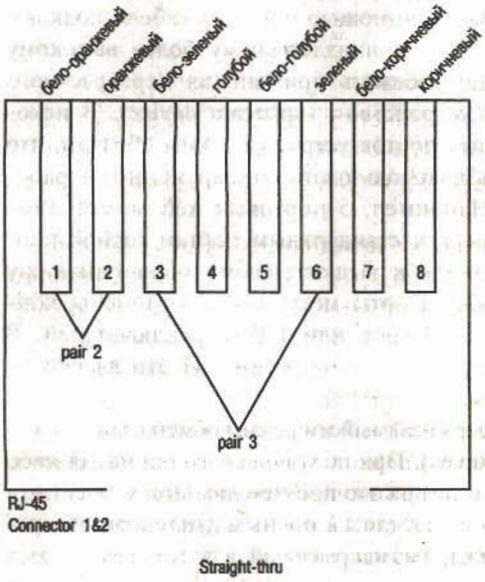


Рисунок 22.5 *Разводка прямого кабеля*

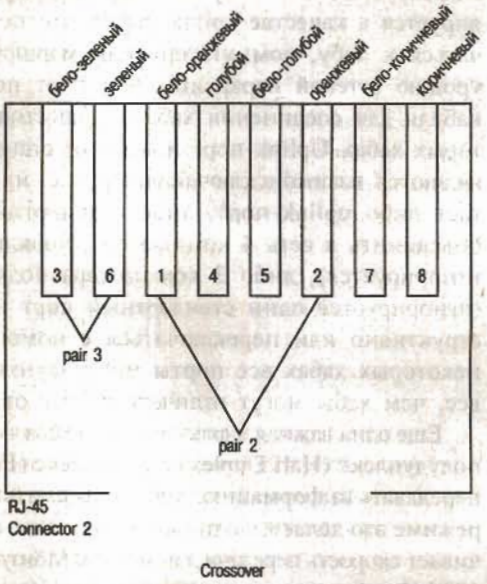


Рисунок 22.6 *Разводка перекрестного кабеля*

Хабы

Хаб (*концентратор*) — это устройство, которое служит центром для подключения нескольких узлов сети. Оно имеет несколько гнезд RJ-45, или портов, обычно от 4 до 24, к которым можно подключить соответствующее число кабелей. Стоимость хабов колеблется от $40 до нескольких сотен долларов, все зависит от их качества, числа

портов и способности взаимодействовать с сетевыми устройствами типа 10base-T и 100base-TX. Большинство хабов способны работать только на одной из скоростей (либо 10 Мбит/с, либо 100 Мбит/с). Хабы, которые могут определять скорость автоматически (N-Way), стоят значительно дороже. Выпускаются хабы самых разных размеров — от малюток величиной с ладонь, до 19-дюймовых устройств, монтируемых в стойки. В связи с большим значением хабов для нормальной работы сети рекомендуется подключать их к источникам бесперебойного питания. Существуют управляемые хабы: подключившись к ним по telnet, можно, используя командную строку, конфигурировать производительность каждого порта индивидуально. Естественно, что такие хабы относятся к разряду дорогостоящих.

ПРИМЕЧАНИЕ

Пару слов относительно терминологии. Английское слово hub обозначает концентраторы, к которым подключаются узлы с одинаковой сетевой архитектурой, а слово concentrator обозначает концентраторы, к которым можно подключать узлы с различными сетевыми архитектурами. Различие между двумя этими терминами, вообще говоря, весьма размыто, поэтому в обиходе концентраторы любых типов называют хабами.

Хаб является эффективным повторителем, т.е. весь сетевой трафик передается одновременно на все порты. Таким образом, компьютер, подключенный к данному концентратору, способен видеть входящий и исходящий трафик любого компьютера, подключенного к тому же хабу. Как правило, один из портов концентратора резервируется в качестве uplink-порта, что позволяет с помощью прямого кабеля подключаться к хабу, коммутатору или маршрутизатору, принадлежащему более высокому уровню сетевой иерархии. Этот порт помогает избежать применения перекрестного кабеля для соединения хаба с вышестоящим устройством (upstream device). В некоторых хабах Uplink-порт и один из стандартных портов устроены таким образом, что являются взаимоисключающими, т.е. их нельзя использовать одновременно — работает либо uplink-порт, либо стандартный. Например, 5-портовый хаб может либо объединить в сеть 4 компьютера, подключенных к стандартным портам (uplink-порт игнорируется), либо 3 компьютера подключаются к вышестоящему маршрутизатору (игнорируется один стандартный порт на хабе). Порты могут быть соединены конструктивно или переключаться с помощью тумблеров или DIP-переключателей. В некоторых хабах все порты могут функционировать одновременно. И это далеко не все, чем хабы могут отличаться друг от друга.

Еще одна важная характеристика хаба — поддерживаемый им режим обмена данными — полудуплекс (Half Duplex) или дуплекс (Full Duplex). При полудуплексе узел может либо передавать информацию, либо принимать ее, т.е. делать это поочередно. При дуплексном режиме это делается одновременно. Так, сеть Fast Ethernet с полным дуплексом обеспечивает скорость передачи данных 100 Мбит/с в каждом из направлений, а полудуплекс — дает 100 Мбит/с суммарно в обоих направлениях.

Коммутаторы

Внешне *коммутатор* (*switch*) выглядят почти так же, как и концентраторы, но портов у них поменьше, а стоят они гораздо дороже. Коммутаторы, как и концентраторы, оснащены несколькими портами — гнездами RJ-45. Они бывают всевозможных размеров и выпускаются одними и теми же компаниями, так что на полках магазинов их немудрено перепутать. Они даже кое в чем схожи по своим функциям

и принципам работы, так что вы можете строить сеть как на основе коммутатора, так и на основе концентратора. Однако различия между ними все же есть, и весьма существенные.

В хабе все порты используют одну внутреннюю шину. Иначе говоря, все компьютеры, подключенные к хабу, равно как и компьютеры, подключенные к другому хабу, работающему в паре с первым, относятся к так называемому *домену коллизий* (collision domain). Под этим термином подразумевается сетевое пространство, на которое будут распространяться конфликты в случае их возникновения. Пакет, посланный одному компьютеру, получат и все остальные компьютеры, входящие в домен. Когда пакет достигает компьютера, сетевая карта сравнивает адрес назначения пакета с адресом компьютера: если они совпадают, то данные принимаются, если нет, то отвергаются. В каждый момент времени передавать информацию может только один компьютер, в противном случае неизбежны коллизии. Они обычно возникают в том случае, когда два узла считают сеть свободной и начинают передачу информации практически одновременно. Сталкивающиеся между собой пакеты разрушаются, и их приходится передавать заново. Это снижает эффективность работы сети.

Коммутатор имеет куда более сложную структуру. Каждый порт коммутатора представляет собой отдельный домен коллизий, поэтому устройство, подключенное к порту, получает только ту часть трафика, которая предназначается ему. В состав коммутатора входит программное обеспечение, осуществляющее анализ заголовка каждого сетевого пакета с целью определения адреса назначения данного пакета. Затем данный пакет транслируется только на тот порт коммутатора, к которому подключено устройство с соответствующим сетевым адресом. Все это автоматически означает наличие у коммутатора ARP-кэша и динамически обновляемой таблицы маршрутизации. Это позволяет управляющим коммутатором программам определять, что и к какому порту подключено. Иногда подключение устройства к другому порту коммутатору, может привести к его зависанию. Потребуется выключить его и снова включить, чтобы была обновлена таблица маршрутизации. Выпускаются управляемые коммутаторы, допускающие удаленное (telnet) конфигурирование из командной строки. Понятно, что это отражается на их сложности и стоимости.

Важнейший аспект функционирования коммутатора заключается в эффективном использовании (с минимумом потерь) пропускной способности сети. Так, 8-портовый хаб типа 100base-TX вынужден распределять пропускную способность сети, равную 100 Мбит/с, между 8 портами. И когда устройства, подключенные к его портам, начинают одновременно передавать большой объем данных, в сети происходят многочисленные коллизии. Это влечет за собой повторные попытки передачи тех же данных на физическом уровне (самый нижний уровень сетевой модели OSI). В свою очередь, это резко снижает реальную скорость передачи данных по сравнению с теоретически достижимой — 100 Мбит/с. Данную проблему успешно решают коммутаторы. Так, если 8-портовый 100-мегабитный хаб обеспечивает скорость 100 Мбит/с на своей внутренней шине, то межпортовая шина 8-портового 100-мегабитного коммутатора имеет пропускную способность 800 Мбит/сек. Вот почему коммутаторы намного дороже, чем концентраторы с аналогичным числом портов.

Сеть, построенная на основе коммутаторов, выигрывает и в плане безопасности, так как компьютер в такой сети просто не способен "увидеть" не предназначенный ему трафик. Да, вы можете запустить программу-сниффер (**tcpdump** или EtherPeek),

вы можете установить вашу сетевую карту в promiscuous mode, дабы она принимала все пакеты, которые видит. Вы попытаетесь шпионить за всем трафиком внутри вашего домена коллизий. Но коммутаторы дают возможность разделить трафик так, чтобы это было невозможно. Еще один метод атаки — умышленная генерация каким-либо хостом агрессивного трафика (при этом сеть наводняется большим количеством ненужных пакетов, использующих всю ею пропускную способность) — малоэффективен в сети с коммутатором. Вот и получается, что коммутаторы стоят своих денег, ведь они, в отличие от концентраторов, существенно повышают производительность сети, ее надежность и безопасность.

В заключение остается добавить, что, как правило, у коммутаторов отсутствует uplink-порт, хотя некоторые из портов (один или два) могут быть оснащены специальным переключателем для перевода порта из режима, требующего перекрестный кабель, в режим работы с прямым кабелем. Основное правило: коммутатор является устройством типа хаба, поэтому для соединения двух коммутаторов используется перекрестный кабель, а для подключения коммутатора к uplink-порту хаба применяется прямой кабель.

Мосты

Мосты (*bridges*) — это устройства, работающие по тем же принципам, что и коммутаторы. Однако они выполняют и ряд функций. Для понимания функций мостов, необходимо понимать назначение 4-го уровня стека TCP/IP. Подробно эти вопросы мы рассмотрим немного позже, пока же можно считать, что мост — это разновидность коммутатора, соединяющего между собой различные сети так, что компьютер из одной сети способен общаться с компьютером из другой сети без помощи маршрутизаторов. Это очень полезно, в частности, при соединении двух сетей разной архитектуры, например Ethernet и Token Ring.

Маршрутизаторы

Маршрутизаторы (*routers*) — самые сложные и дорогостоящие из всех сетевых устройств. И хотя само понятие маршрутизации в деталях рассматривается в одном из следующих разделов данной главы, уже сейчас необходимо объяснить функции маршрутизаторов и то, как они применяются в сетевой топологии.

Маршрутизаторы оснащены операционными системами и хранят таблицы маршрутизации, которые содержат адреса устройств, по которым устройства сети могут быть найдены. Большинство сетей имеет только один маршрутизатор, определяющий номер этой сети (адрес подсети в терминологии TCP/IP). Он однозначно идентифицирует эту локальную сеть в масштабах глобальной сети. В принципе, можно установить любое количество маршрутизаторов в пределах локальной сети, разбивая ее тем самым на самостоятельные подсети более низкого уровня.

Маршрутизаторы отвечают за вычисление кратчайшего пути следования пакета от передатчика к приемнику, принадлежащих различным подсетям, общаясь с другими маршрутизаторами при помощи множества различных протоколов. При отправке пакета он путешествует от маршрутизатора к маршрутизатору, продвигаясь на все более высокие уровни сети до тех пор, пока не будет достигнут маршрутизатор искомой зоны. С этого момента начинается "спуск" пакета от маршрутизатора к маршрутизатору вплоть до хоста с искомым адресом. Пакет спускается вниз по уровням до тех пор, пока не достигнет искомой подсети, где местный локальный маршрутизатор направляет его

на искомый хост в данной подсети. Процесс маршрутизации — это то, на чем основана работа Internet. Более подробно маршрутизация будет рассмотрена в главе 28.

Большинство маршрутизаторов снабжены одним или двумя портами с разъемами типа RJ-45 или AUI, представляющими собой интерфейс к локальной сети. При помощи последовательных кабелей маршрутизаторы соединяются с устройствами обработки данных/обслуживания каналов DSU/CSU или другими высокоскоростными преобразователями, за которыми начинаются скоростные глобальные магистрали (T1, например). В целом маршрутизаторы сильно отличаются по своим характеристикам, как-то: физические габариты, сложность, число и тип портов, возможности управления и стоимость. А понимание их работы, пожалуй, самый важный аспект в понимании сетевых технологий.

Сетевые протоколы

Ранее уже упоминалось, что большая часть Internet-трафика передается с использованием стека протоколов TCP/IP, в который входит несколько различных протоколов. Стек означает набор; в сетевых технологиях это совокупность взаимосвязанных сетевых протоколов, функциональность которых полностью или частично соответствует всем уровням используемой сетевой модели. IP — Internet-протокол. На его основе построена передача большей части Internet-трафика. Этот протокол относится к сетевому уровню модели OSI (Open Systems Interconnection). Internet-протокол описывает формат и способ маршрутизации пакетов данных. Вторая составляющая стека TCP/IP — это TCP, потоковый протокол транспортного уровня, ориентированный на установление соединений и осуществляющий доставку пакетов данных. Существуют и другие важные протоколы, например ICMP, на основе которого работают утилиты **ping** и **traceroute**. Далее каждый из перечисленных здесь протоколов будет описан более детально.

TCP: Transmission Control Protocol

Главная отличительная особенность TCP от UDP — это высокая надежность первого и сравнительно низкая надежность второго. Так, TCP обладает множеством реализованных в нем механизмов, позволяющих с высокой степенью вероятности гарантировать, что трафик, передаваемый по этому протоколу, в целости и сохранности достигнет места назначения. Помимо этого, TCP обладает функциями по разбивке больших пакетов на подходящих размеров фрагменты с целью более эффективной передачи данных.

По сути, в TCP реализован двухступенчатый механизм, подразумевающий квитирование данных (подтверждение их успешной доставки). При посылке TCP-пакета передающий компьютер в течение некоторого времени ожидает получения подтверждения приема (Acknowledge — ACK). Причем для повышения скорости работы во время ожидания первого подтверждающего пакета продолжается передача последующих пакетов данных. Если же по истечении тайм-аута от принимающего компьютера не пришло ACK, то передающий компьютер вторично отправляет "потерявшийся" пакет с данными, считая, что отосланный пакет не дошел до места назначения. Данный алгоритм известен как повторная передача, и хотя подобное квитирование существенно повышает надежность передачи, оно же служит причиной снижения скорости работы сети (часть пропускной способности расходуется служебными пакетами, да и повторные передачи тоже отнимают время и ресурсы).

Еще одним фундаментальным аспектом в реализации TCP служит работа в режиме полного дуплекса, что означает одновременную передачу и прием данных в обоих направлениях с использованием однотипных пакетов. Кроме того, в TCP существует еще целый ряд интересных особенностей, таких как проверка контрольной суммы, обеспечивающая целостность принятых данных, автоматическая фрагментация пакетов и сборка фрагментов в единое целое, а также упорядочивание данных, пришедших в неправильной последовательности. Все эти функции в UDP просто не реализованы.

Таким образом, TCP активно используется тогда, когда целостность и надежность приема/передачи данных является критически важным параметром. Под эту категорию подпадает множество приложений, таких как электронная почта, HTTP-доступ в Internet, пересылка файлов по FTP и т.д.

UDP: User Datagram Protocol

UDP относится к разряду ненадежных протоколов. В нем отсутствуют механизмы квитирования и повторной передачи. Не говоря уже о том, что UDP-пакеты могут быть отосланы в широковещательном режиме, т.е. всем хостам в сети, вне зависимости от того, им ли адресованы эти данные.

В принципе, UDP-пакеты по пути к адресату могут легко затеряться на просторах сети, и нет никакой возможности отследить факт их успешной доставки по адресу. Поскольку UDP-пакеты не обладают специфическими номерами, передатчик просто "извергает" их из себя в сеть, а уж дойдут ли они до места назначения — это как бог даст. Другими словами, обеспечение надежности UDP-пакетов должно осуществляться теми приложениями, которые используют данный протокол. Так, система NFS, обеспечивающая доступ к сетевым дискам так, как будто они локальные, является приложением, работающим на основе UDP-пакетов, подробнее об этом рассказано в главе 31. Почему же приложение, разработанное для работы с файлами по сети, использует ненадежный протокол UDP? Причина такого выбора заключается в том, что в локальной сети NFS-ресурсы могут использоваться одновременно множеством хостов, причем каждый из них может исчезнуть из сети в любой момент без предупреждения. Чтобы не реализовывать достаточно сложную процедуру соединения и поддержки связи по протоколу TCP со всеми этими недолговечными хостами, было решено использовать UDP, а о вопросах надежности позаботится на прикладном уровне средствами NFS. Это возможно, поскольку в UDP-пакетах содержится информация о полноте передачи, так что система NFS может судить о целостности данных (нет ли потерявшихся или поврежденных пакетов).

В наши дни наибольшая польза от UDP состоит в работе с потоковыми медиаданными. Для таких сетевых приложений, как телеконференции, широковещательная музыка и потоковое видео, работающих в режиме реального времени, потеря пары-тройки пакетов не критична. Такие данные при передаче разбиваются на множество маленьких пакетов-фрагментов, которые передаются сплошным потоком и собираются в единое целое на компьютере-получателе с моментальным выводом результата на динамик и монитор компьютера. Если утеряна небольшая часть данных, это, как правило, проходит незамеченным для пользователя. Если же случается перегрузка сети в целом, то пакеты просто теряются на уровне маршрутизатора (он отслеживает свои собственные тайм-ауты и обладает буфером принимаемых/передаваемых данных). Затем поток возобновляется в реальном времени при условии, что

передающий хост исчез из сети не надолго. При таком подходе нет нужды повторно посылать все утерянные пакеты, это просто несущественно. Именно в этой области проявляются преимущества UDP и не имеют значения его недостатки.

ICMP: Internet Control Message Protocol

Некоторые относят ICMP к сетевому уровню, приравнивая его к таким протоколам, как IP и IPX; другие рассматривают его как транспортный протокол, подобный TCP и UDP. Правы обе стороны: одна часть ICMP принадлежит сетевому уровню, другая — транспортному. Дело в том, что ICMP-сообщения обрабатываются именно на сетевом уровне, но содержат IP-заголовок, что затрудняет четкую классификацию данного протокола по уровням сетевой модели.

ICMP был разработан в качестве диагностического протокола. Наиболее частыми его применениями являются утилиты **ping** и **traceroute**, которые возвращают информацию, извлекаемую из фундаментальной структуры ICMP. Работа этих утилит построена на способности ICMP запрашивать удаленный хост, дабы получить от него ответ, подтверждающий его существование в сети и работоспособность. Этот факт подтверждается так называемым эхо-ответом (echo-reply). Кроме того, ICMP может быть использован для передачи сообщений об ошибках протоколам транспортного уровня, например, передатчик UDP-пакета может получить сообщение о том, что порт недоступен (Port Unreachable), если UDP-пакет направлен на неактивный (закрытый) порт. В ICMP встроены 16 кодов возможных ошибок и ряд дополнительных функций.

TCP/IP

Как же различные сетевые компоненты и протоколы работают совместно друг с другом? Стек протоколов TCP/IP представляет собой совокупность следующих друг за другом протоколов разных уровней, а также их обработчиков, которые получают данные от выше- или нижележащего протокола, преобразуют полученные данные и передают их на вход следующего уровня. На рисунке 22.7 изображен один из двух возможных путей прохождения данных через стек протоколов — сверху вниз — начиная с высшего, пользовательского уровня (прикладная программа), далее через множество протоколов и так до самого нижнего, физического уровня (сетевой кабель).

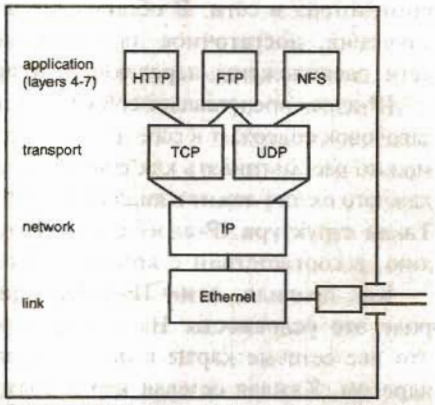


Рисунок 22.7 *Диаграмма уровней стека протоколов TCP/IP*

Прикладная программа передает свои данные, которые необходимо отослать по сети, на транспортный уровень. Они поступают на вход либо TCP-, либо UDP-обработчика, который в полученный пакет данных добавляет заголовок, содержащий информацию о способах декодирования этих данных на транспортном уровне принимающего хоста (это действие называется инкапсуляцией). Затем преобразованный пакет направляется на сетевой уровень, здесь IP-обработчик добавляет к пакету свой заголовок, содержащий информацию для анало-

гичного IP-обработчика на приемнике. Потом пакет данных достигает физического уровня, на котором драйвер сетевой карты инкапсулирует весь пакет (со всеми ранее созданными заголовками) в соответствующий стандартам данной сети физический сигнал. Он-то и передается сетевой картой в кабель.

Компьютер, получающий этот сигнал, пропускает прибывший пакет через стек протоколов снизу вверх (пакет проходит уровни в обратном порядке). Причем на каждом уровне анализируется соответствующий заголовок. Согласно информации, содержащейся в пакете, он подвергается ряду последовательных преобразований, передаваясь с уровня на уровень, вплоть до прикладной программы.

Различные сетевые устройства относятся к различным уровням стека протоколов. К примеру, мосты, соединяющие различные подсети, работают на физическом уровне. А коммутаторы и маршрутизаторы работают на сетевом уровне, где самой важной является информация об IP-адресах отправителя и получателя пакетов.

Детальные сведения о заголовках, которые добавляются к пакету на каждом из уровней, потребуются нам намного позднее, когда мы будем рассматривать, как эти механизмы применяются во FreeBSD. Достойно прочтения иллюстрированное руководство Ричарда Стивенса по стеку протоколов TCP/IP, издательство Addison-Wesley, 1994. В этом документе просто и понятно описаны все подробности работы протоколов, включая аспекты их взаимодействия между собой, а также внутреннюю реализацию и работу каждого протокола в отдельности.

IP-адреса

IP-адрес предназначен для однозначной идентификации компьютера в сети Internet, хотя это подразумевает нечто большее, нежели один адрес для каждого компьютера в сети. В общем случае, IP-адрес — это логическое описание места назначения, достаточное для того, чтобы маршрутизаторы могли определить, где в сети расположены передающая и принимающая пакеты машина.

IP-адрес представляет собой 32-битную строку, содержащуюся в IP-заголовке, причем заголовок содержит в себе два таких адреса: куда и откуда. Это 32-битное двоичное число можно рассматривать как совокупность четырех октетов по 8 бит каждый, т.е. значение каждого октета лежит в диапазоне от 0 до 255. Выглядит это примерно так: 111.112.113.114. Такая структура IP-адреса, кроме лучшей читаемости, обеспечивает должную иерархию, в соответствии с которой адреса делятся на классы (A, B и C).

Как правило, один IP-адрес привязан к одной сетевой карте, хотя в некотором роде это условность. На самом деле единственное ограничение заключается в том, что две сетевые карты в пределах одной сети не могут обладать одним и тем же IP-адресом. Каждая сетевая карта должна для нормальной работы иметь, как минимум, один IP-адрес, но при желании к одной сетевой карте можно привязать несколько IP-адресов. Впрочем, более распространенным является вариант с наличием на компьютере двух или нескольких сетевых карт, у каждой из которых свой IP-адрес. В этом случае компьютер становится шлюзом между подсетями (т.е. он представлен в обеих подсетях одновременно). К примеру, один адрес связан с Ethernet-картой, а другой — с картой беспроводной сети по стандарту 802.11.

Определить IP-адрес любой из сетевых карт на компьютере, равно как и прочих сетевых интерфейсов, можно с помощью утилиты **ifconfig**, типичный вывод которой показан в листинге 22.1. Опция **-a** выводит список всех сетевых устройств (интерфей-

сов) в системе. Можно также указать конкретный интерфейс (например, x10) для получения информации только по нему одному. Строка **inet** отображает искомый IP-адрес; в данном примере, к интерфейсу x10 привязаны несколько IP-адресов для одной сетевой карты.

Листинг 22.1 Типичный вывод утилиты ifconfig

```
# ifconfig -a
x10:   flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>   mtu   1500
         inet  64.41.131.102  netmask  0xffffff00  broadcast  64.41.131.255
         inet6  fe80::201:2ff:fe55:1256%x10  prefixlen  64  scopeid  0x1
         inet 209.154.215.246 netmask 0xffffffff broadcast 209.154.215.246
         ether   00:01:02:55:12:56
         media:  autoselect  (100baseTX)  status:  active
         supported  media:  autoselect  100baseTX  <full-duplex>  100baseTX
10baseT/UTP  <full-duplex>  10baseT/UTP  100baseTX  <hw-loopback>
lp0:   flags=8810<POINTOPOINT,SIMPLEX,MULTICAST>   mtu   1500
gif0:   flags=8010<POINTOPOINT,MULTICAST>   mtu   1280
gif1:   flags=8010<POINTOPOINT,MULTICAST>   mtu   1280
gif2:   flags=8010<POINTOPOINT,MULTICAST>   mtu   1280
gif3:   flags=8010<POINTOPOINT,MULTICAST>   mtu   1280
lo0:   flags=8049<UP,LOOPBACK,RUNNING,MULTICAST>   mtu   16384
         inet6  fe80::1%lo0  prefixlen  64  scopeid  0x7
         inet6  ::1  prefixlen  128
         inet  127.0.0.1  netmask  0xff000000
ppp0:   flags=8010<POINTOPOINT,MULTICAST>   mtu   1500
sl0:   flags=c010<POINTOPOINT,LINK2,MULTICAST>   mtu   552
faith0:   flags=8000<MULTICAST>   mtu   1500
```

Существует несколько специальных IP-адресов, которые очень важны. Во-первых, это адрес сети — адрес, в котором один или несколько последних байтов равны нулю, например 64.41.131.0. Этот адрес относится ко всей сети 64.41.131 и, как правило, используется при настройке маршрутизаторов. Еще более важным является адрес, у которого последний байт равен 255. Это так называемый широковещательный адрес. Например, адрес 64.41.131.255 относится ко всем компьютерам, составляющим сеть 64.41.131, а адрес 64.41.255.255 относится ко всем хостам, входящим в состав сети 64.41. Применение этих адресов будет подробно изложено при рассмотрении организационных принципов маршрутизации.

INTERNET-ПРОТОКОЛ ВЕРСИИ 6 (IPV6)

Возникает вопрос, а достаточен ли диапазон адресного пространства, обеспечиваемый 32-битной длиной адреса? На первый взгляд, 2^{32} — это приблизительно 4,3 млрд. возможных комбинаций, а сегодня в сети зарегистрировано около 30 млн. хостов. Однако число компьютеров, подключенных к Internet, растет с каждым годом все быстрее, и адресное пространство, вероятно, будет очень быстро исчерпано. Особенно если учесть, что на самом деле в сети доступно менее 4,3 млрд. адресов. Давайте подсчитаем. Существует только 256 сетей класса A, по 16,7 млн. потенциальных хостов в каждой. И те компании, которые являются владельцами зарегистрированных сетей класса A вряд ли используют все 16,7 млн. адресов внутри сети.

Многие предприятия решают эту проблему при помощи NAT (Network Address Translation — трансляция сетевых адресов) и IP-маскарадинга. Эти средства позволяют всю внутреннюю сеть представить единственным IP-адресом, хотя сама внутренняя сеть может при этом состоять из большого количества хостов. Однако при этом внешние компьютеры не могут непосредственно обращаться к внутренним компьютерам сети. И единственный способ избежать этого, обеспечив полноценное соединение между двумя компьютерами, — это назначить каждому компьютеру в сети IP-адрес в стандартном 32-битном виде.

Поэтому ряд независимых групп разработали новый стандарт IP-адресации, так называемого IPv6. Этот стандарт обеспечивает 128-битное адресное пространство, с адресами в четыре раза более длинными, чем в ныне действующем стандарте IPv4. В результате мы получим $3{,}4\times10^{38}$ возможных адресов. Новый стандарт включает также такие возможности, как встроенное шифрование данных и аутентификация, дифференциация соединений при помощи плавающих меток, полностью автоматическая настройка хостов и др.

Проблема в том, что новый стандарт очень медленно входит в обиход. Построение сетей — процесс дорогостоящий, а их перевод на новый стандарт означает замену всего устаревшего сетевого оборудования, неспособного работать с IPv6, и требует значительных средств. Пока очень немногие компании решились на такой шаг. Тем не менее FreeBSD предоставляет полную поддержку IPv6 и может обеспечить вам преимущества данного стандарта, даже если соответствующей инфраструктуры еще нет.

ARP и MAC-адреса

Хотя пользователь может работать с IP-адресами, задавая их в таких сетевых приложениях, как FTP-программы или браузеры, на самом деле эти адреса —искусственная конструкция, предназначенная для того, чтобы на уровне операционной системы связать хост и соответствующий ему реальный сетевой интерфейс. Естественно, что IP-адрес, назначенный тому или иному компьютеру, может быть в любой момент изменен на другой IP-адрес. Что же касается самого нижнего уровня стека протоколов TCP/IP, то на нем IP-адрес не играет никакой роли. Сетевая карта ничего не знает о том, какой логический адрес назначен ей операционной системой. А драйверы устройств работают на более высоком уровне, чем уровень, на котором происходит процесс коммуникации.

На нижнем сетевом уровне значение имеют только так называемые физические адреса, также известные как аппаратные адреса, или MAC-адреса (Media Access Controller). Каждое сетевое устройство имеет уникальный MAC-адрес, присваиваемый производителем. Теоретически, MAC-адрес — это что-то вроде отпечатков пальцев для сетевых карт. Это однозначный идентификатор каждой сетевой карты, а значит и компьютера, в котором она установлена, поскольку в мире не существует двух сетевых карт с одинаковыми MAC-адресами. На практике, однако, все не так просто. Не существует механизма маршрутизации, способного работать с MAC-адресами, но есть механизм, работающий с обычными IP-адресами. Давайте разберем способ взаимодействия этих различных типов адресов.

В листинге, отображающем результаты работы утилиты **ifconfig**, есть строка ether. В этой строке указан аппаратный адрес данной сетевой карты 00:01:02:55:12:56, характерный для карт от 3Com. Каждый сетевой адаптер имеет уникальный 48-разрядный адрес, байты которого разделены двоеточием. Адрес длиною в 6 байт обеспечивает доступное адресное пространство в 2 800 000 000 000 000 потенциально возможных адресов — значительно больше, чем те 30 млн. компьютеров, которые сегодня подключены к сети. Первые три байта адреса однозначно определяют производителя сетевой карты. Каждый зарегистрированный производитель имеет в своем распоряжении одну или несколько 3-байтных сигнатур, выделенных ему IEEE (Institute of Electrical and Electronics Engineers — Институт инженеров по электротехнике и электронике). А оставшиеся 3 байта — серийный номер адаптера. Так что на рынке может существовать до 16,7 млн. производителей сетевых карт, каждый из которых может выпустить до 16,7 млн. единиц продукции.

Теперь о том, как работает система адресации в целом. Когда пакет переходит на физический уровень стека протоколов, то сетевая карта (под управлением драйвера) должна отправить этот пакет в сеть, адресовав его на компьютер, имеющий IP-адрес, указанный в заголовке пакета. Естественно, что сетевой адаптер использует для отправки и адресации MAC-адреса, поскольку он ничего не знает про IP-адреса. Но откуда же тогда стеку протоколов известен MAC-адрес удаленного компьютера в сети, если пришедший пакет снабжен заголовком только с IP-адресом? Для перевода одного адреса в другой используется ARP.

ARP (Address Resolution Protocol) — протокол преобразования адресов, который служит для преобразования IP-адресов в MAC-адреса. Когда приложение посылает данные на компьютер с IP-адресом 10.5.6.100, пакет с этими данными перед отправкой попадает на физический уровень стека протоколов, где соответствующая программа-обработчик консультируется с ARP-кэшем, который представляет собой таблицу соответствия IP- и MAC-адресов, хранимую и обновляемую средствами операционной системы. Разумеется, если пакет на адрес 10.5.6.100 еще ни разу не отправлялся, то соответствующая запись в ARP-кэше отсутствует, и стеку протоколов еще только предстоит выяснить, какой же аппаратный адрес соответствует данному IP-адресу. Это делается посылкой специального широковещательного пакета запроса с вложенным в него искомым IP-адресом. Этот запрос адресован всем аппаратным адресам в локальной сети (путем указания адреса назначение 255 — ff:ff:ff:ff:ff:ff). Получают этот пакет все компьютеры в локальной сети, но отвечает только тот, чей IP-адрес совпал с запрашиваемым в пакете. Причем в заголовке ответного пакета содержится не только аппаратный адрес назначения, но и аппаратный адрес отправителя — с этого момента для IP-адреса 10.5.6.100 становится известен его MAC-эквивалент, и в ARP-кэш заносится соответствующая запись для использования ее в будущем. И в дальнейшем все пакеты, адресованные хосту 10.5.6.100, передаются на уже известный аппаратный адрес.

ПРИМЕЧАНИЕ

Как вы помните, коммутаторы ведут свою собственную, внутреннюю таблицу адресов ARP-кэша. Зная принцип работы протокола ARP, легко понять и то, как коммутаторы, используя ARP, определяют адрес хоста, подключенного к определенному порту. Именно благодаря внутреннему ARP-кэшу, коммутаторы оптимизируют сетевой трафик, посылая пакеты только на тот порт, к которому подключен хост назначения.

Команда arp

Протокол ARP прозрачен для прикладных программ и стека протоколов TCP/IP. Однако существуют ситуации, когда администратор сети должен вручную поработать с ARP-кэшем. Содержимое этого кэша можно проверить, введя команду **arp -a**:

```
# arp -a
w001.sjc-ca.dsl.cnc.net (64.41.131.1) at 0:0:c5:7c:7:f0 [ethernet]
w013.sjc-ca.dsl.cnc.net (64.41.131.13) at 0:30:65:a4:9a:5e [ethernet]
w063.sjc-ca.dsl.cnc.net (64.41.131.63) at ff:ff:ff:ff:ff:ff permanent
  ↪[ethernet]
```

Записи в ARP-кэше хранятся определенное время, после чего кэш автоматически обновляется. При этом выполняется серия запросов (это необходимо, поскольку IP-адреса, в отличие от аппаратных, могут меняться). Но иногда приходится проводить ручную очистку кэша, например, если на сетевой компьютер устанавливается новая се-

тевая карта, то в сети появляется новый аппаратный адрес, к которому привязан старый IP-адрес. В этом случае данный хост некоторое время будет недосягаем из-за несоответствия записей в кэше реальному положению дел. Удалять такие записи можно тотально для всего кэша, либо выборочно — по имени записи или по содержащемуся в ней IP-адресу:

```
# arp -d w001.sjc-ca.dsl.cnc.net
# arp -d 64.41.131.13
# arp -d -a
```

Подсети и сетевые маски

Сетевые маски — один из самых трудных моментов в понимании работы TCP/IP-сетей, с другой стороны — это самая элегантная часть конфигурации. Главная задача *сетевой маски* (*netmask*) — указать маршрутизатору, к какой из подсетей, внутренней (локальной) или внешней (глобальной), принадлежит IP-адрес получателя. Когда маршрутизатор получает пакет (не ARP-запрос, а обычный TCP/IP-пакет) и должен решить, что с ним делать, он принимает решение, сравнивая IP-адрес получателя, со своей сетевой маской.

Сетевая маска представляет собой 32-битную строку, похожую на IP-адрес, одно из ее типичных значений — 255.255.255.0. Как и для IP-адресов, для маски подсети используется десятичная нотация. Единицы в маске сети обозначают сетевую часть IP-адреса, нули соответствуют адресу хоста в IP-адресе. Предположим, что маршрутизатор отвечает за сеть с номером 64.41.131 и получает пакет с адресом назначения 64.41.131.45. Маршрутизатор сравнивает IP-адрес пакета со своей сетевой маски, выполняя логическую операцию and, а затем результат сравнивается с адресом сети, к которой относится маршрутизатор. Поскольку результат совпал с номером локальной сети, пакет направляется в локальную сеть. Если адрес пакета не совпадает с адресом локальной подсети, то пакет перенаправляется в глобальную сеть.

Аналогичный процесс имеет место на любом компьютера, использующем TCP/IP. Если результат проверки показал принадлежность получателя к внешнему миру (глобальной сети), то отправитель направит пакет на адрес указанного в его настройках *шлюза*, т.е. маршрутизатора заданного по умолчанию. Если же результат проверки показал принадлежность искомого IP-адреса к локальной сети, но соответствующая запись в кэше отсутствует, то формируется ARP-запрос.

Этот механизм делает возможной тонкую настройку локальной сети, а точнее ее разбиение на подсети. Предположим, организация закупила для использования диапазон адресов сети класса B 64.41.xxx.xxx (64.41 — номер данной подсети, .xxx.xxx — адрес хоста в пределах подсети). Главный маршрутизатор под именем R1 управляет всей этой подсетью, но предположим, что в ней существует еще и маршрутизатор R2, заведующий двумя подсетями класса C — 64.41.131.xxx и 64.41.132.xxx. Компьютеры в подсети 64.41.131.xxx могут быть подключены к тому же концентратору, что и компьютеры подсети 64.41.132 (см. рис. 22.8), но они не смогут непосредственно общаться между собой, если их сетевые маски имеют одинаковое значение — 255.255.255.0, типичное для сетей класса C. Но если установить на отправителе значение сетевой маски 255.255.0.0, типичное для сетей класса B, то отправитель из сети 64.41.131. окажется в той же подсети, что и получатель в сети 64.41.132, а значит, пакет пойдет напрямую от хоста к хосту. Тем не менее получатель не сможет ответить отправите-

лю непосредственно, так как сам он принадлежит к сети класса C, а не B (если быть еще более точным, то при значении сетевой маски равным 255.255.0.0, компьютер вообще не знает, что адрес 64.41.131.xxx принадлежит к подсети класса C, сетевая маска класса B заставляет рассматривать этот адрес как номер подсети 64.41 и искомый хост в ней с номером 131.xxx!)

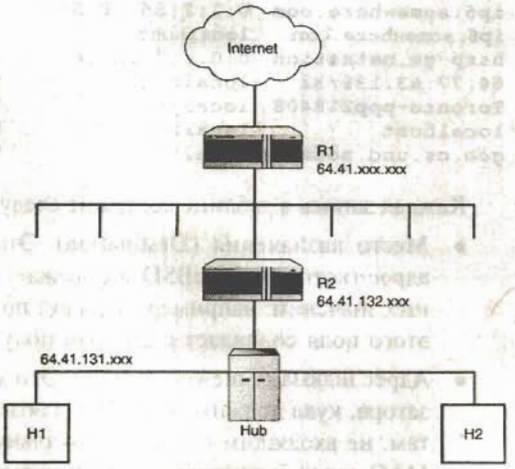


Рисунок 22.8
Сеть с подсетями демонстрирует путь пакета между двумя хостами, настройки которых не запрещают им общаться напрямую.

Причем значение сетевой маски может быть задано и в шестнадцатиричном виде типа 0xffffff00 (255.255.255.0) или 0xffffffc0 (255.255.255.192), последнее значение задает подсеть из 64 хостов с номерами от 1 до 64.

В сетях с заданными сетевыми масками существует еще одна запись, достойная упоминания, так называемая *CIDR* (*Classless Inter-Domain Routing — безклассовая междоменная маршрутизация*), представленная в форме номера подсети, слэша и количества бит, составляющих сетевую маску. Например, маска 255.255.255.0 в сети с номером 64.41.131 в нотации CIDR будет записана как 64.41.131/24, поскольку маска содержит 3 байта (24 бита), установленных в 1. А вот маска 255.255.255.192 в нотации CIDR будет представлена как 64.41.131/26, поскольку она содержит 26 битов, установленных в единицу. Такая форма записи сетевой маски встречается как в таблицах маршрутизации, так и в других конфигурационных записях.

Маршрутизация

Вообще говоря, конфигурирование маршрутизаторов — это один из главнейших навыков настройки сетей. Поэтому, чтобы успешно администрировать сети, очень важно правильно понимать работу маршрутизатора, в частности работу с таблицами маршрутизации при помощи утилиты **portsentry**, которая подробно описывается в главе 29.

Любой маршрутизатор работает с так называемой таблицей маршрутизации, представляющей набор правил, согласно которым пакеты с определенными IP-адресами перенаправляются на следующий маршрутизатор. Поскольку FreeBSD можно настроить в качестве маршрутизатора, давайте внимательно разберем таблицу маршрутизации, использовав для вывода информации утилиту **netstat -rn**:

```
# netstat -rn
Routing tables
```

```
Internet:
Destination        Gateway            Flags    Refs      Use   Netif  Expire
default            hsrp-gw.netnation. UGSc      126  1379327   xl0
64.41.53.101/32    localhost          UGScB       0        3   lo0
net-64-40-111.netn link#1             UC          0        0   xl0   =>
ip3.somewhere.com  0:50:ba:b3:98:13   UHLW        2   357107   xl0        735
ip4.somewhere.com  0:50:ba:b3:95:bb   UHLW        0     2272   xl0        500
ip6.somewhere.com  0:1:2:55:12:56     UHLW        0   118941   lo0   =>
ip6.somewhere.com  localhost          UGScB       0        0   lo0
hsrp-gw.netnation. 0:0:c:7:ac:1e      UHLW      119     8814   xl0        405
64.77.63.139/32    localhost          UGScB       0        0   lo0
Toronto-ppp218408. localhost          UGScB       0        0   lo0
localhost          localhost          UH         45 45210727   lo0
goo.cs.und.nodak.e localhost          UGScB       0        0   lo0
```

Каждая запись в таблице содержит следующую информацию:

- Место назначения (Destination). Это может быть IP-адрес, имя хоста, сетевой адрес (который FreeBSD отображает в формате CIDR) или несколько специальных значений, например, маршрут по умолчанию (default route). По сути, значение этого поля совпадает с адресом получателя, содержащимся в каждом пакете.
- Адрес шлюза (Gateway address). Это адрес вышестоящего по иерархии маршрутизатора, куда должны перенаправляться определенные пакеты, адресованные хостам, не входящим в данную локальную сеть. Это может быть как IP-адрес, так и MAC-адрес, если вышестоящий маршрутизатор охватывает ту локальную сеть (нижестоящий маршрутизатор расположен в одной из ее подсетей), к которой принадлежит отправитель. FreeBSD также отображает запись для шлюза link#1, который является маршрутом для основного сетевого интерфейса, откуда могут быть узнаны новые, ранее неизвестные маршруты (исходя из значения C флага).
- Флаги (flags). Служат для указания типа соответствующего им маршрута. Каждая буква представляет собой отдельный флаг. Как видно из данного примера, для каждой записи присутствует от 2 до 5 флагов. Для уточнения значений каждого из флагов необходимо свериться с **man netstat**.
- Сетевой интерфейс (Network Interface). Если адрес назначения совпадает с определенным правилом, этот пакет отправляется посредством указанного для данного правила интерфейса.

Система IP-маршрутизации выросла на базе стандартов первой сети ARPA (созданной министерством обороны США). С тех пор она постоянно совершенствуется, отвечая новым условиям в быстро растущей сети Internet. Большие телекоммуникационные концентраторы содержат совершенно невообразимые таблицы маршрутизации, в которых присутствует множество резервных ссылок на альтернативные маршруты от хоста к хосту, предусмотренных на случай, если оптимальные маршруты по каким-либо причинам недоступны. Ни один из маршрутизаторов не хранит полной информации обо всех возможных маршрутах в Internet, но каждый знает некоторое, ограниченное число важнейших маршрутов. Кроме того, каждый маршрутизатор хранит адреса вышестоящих маршрутизаторов, а те, в свою очередь, обладают сведениями о всех маршрутизаторах, отстоящих от них на один уровень вниз. Любой не локальный адрес перенаправляется на вышестоящий маршрутизатор. Так продолжается до тех пор, пока пакет не попадет на маршрутизатор, хранящий в своей таблице запись, содержащую маршрут к искомому IP-адресу. Иначе говоря, все

пакеты, адресованные во внешний мир, перенаправляются с надеждой, что рано или поздно, путешествуя с маршрутизатора на маршрутизатор, эти пакеты найдут своего адресата.

На своем пути вверх пакет достигает самого верхнего маршрутизатора, отвечающего за домен верхнего уровня, в который адресован пакет. Это можно гарантировать с большой степенью вероятности. Затем начинается путь пакета вниз по цепочке маршрутизаторов в нужном направлении. И вот тут-то очень важно, чтобы каждый из маршрутизаторов на пути пакета был правильно сконфигурирован. Иначе пакет, спустившись на уровень вниз, неожиданно обнаружит, что маршрутизатор, который должен бы знать его маршрут еще на один уровень вниз, почему-то его не знает. В таком случае неправильно сконфигурированный маршрутизатор вернет пакет вверх, а маршрутизаторы верхнего уровня попытаются вновь отправить его вниз, по тому же самому маршруту — возникнет кольцо маршрутизации (router loop):

```
traceroute to arclight.net (209.237.26.189), 30 hops max, 40 byte packets
 1 r2-72-core-van.netnation.com (10.10.4.253) 101.878 ms 135.377 ms 85.218 ms
 2 dis2-vancouver-atm1-0-0-33.in.bellnexxia.net (206.108.110.189) 132.023 ms
↪80.653 ms  81.686 ms
 3 core2-vancouver-pos11-1.in.bellnexxia.net (206.108.101.45) 102.365 ms
↪61.537 ms  68.561 ms
 4 core2-seattle-pos12-0.in.bellnexxia.net (206.108.102.209) 79.989 ms 109.389
↪ms  115.587 ms
 5 bx3-seattle-pos5-0.in.bellnexxia.net (206.108.102.202) 86.434 109.678 129.128
 6 sea1-bellnexxia-oc12.sea1.above.net (208.184.233.73) 91.201 67.287 79.369
 7 core2-core1-oc48.sea1.above.net (208.185.175.178) 74.219 79.480 93.121
 8 sjc2-sea1-oc48.sjc2.above.net (216.200.127.117) 188.692 212.627 181.123
 9 core1-sjc2-oc48.sjc1.above.net (208.184.102.25) 195.260 194.973 272.053
10 main1-core1-oc12.sjc1.above.net (208.185.175.246) 344.104 318.313
11 core1-sjc2-oc48.sjc1.above.net (208.184.102.25) 195.260 194.973 272.053
12 main1-core1-oc12.sjc1.above.net (208.185.175.246) 344.104 318.313
13 core1-sjc2-oc48.sjc1.above.net (208.184.102.25) 195.260 194.973 272.053
14 main1-core1-oc12.sjc1.above.net (208.185.175.246) 344.104 318.313
15 core1-sjc2-oc48.sjc1.above.net (208.184.102.25) 195.260 194.973 272.053
16 main1-core1-oc12.sjc1.above.net (208.185.175.246) 344.104 318.313
17 core1-sjc2-oc48.sjc1.above.net (208.184.102.25) 195.260 194.973 272.053
18 main1-core1-oc12.sjc1.above.net (208.185.175.246) 344.104 318.313
```

Ситуация с отбрасыванием пакета будет продолжаться до тех пор, пока нижестоящий маршрутизатор не исправит свою таблицу маршрутизации. К счастью, проблема зацикливания бывает временной, рано или поздно сбойный маршрутизатор построит новую таблицу маршрутизации, и работоспособность сети будет восстановлена.

Шлюзы и трансляция сетевых адресов (Gateways and Network Address Translation)

Термин *шлюз* часто используется как синоним *маршрутизатора*, и хотя это не является некорректным определением, необходимо внести ясность с точки зрения современных реалий.

При настройке стека протоколов на сетевом компьютере необходимо задать адрес шлюза, т.е. адрес вышестоящего маршрутизатора. Посмотрите на рисунок 22.8: шлюзы устройства H1 и H2 сконфигурированы так, что они могут связываться с хостами, чьи адреса не совпадают с сетевой маской данных устройств. Обычно в сетевой топологии такого типа роль шлюза выполняет устройство R2. Устройства H1 и H2 не всегда способны напрямую связаться с R1. А это бывает важно с точки

зрения аварийных ситуаций в сети. Если появится необходимость выключить R2, то маловероятно, что он будет пропускать через себя сетевой трафик; хотя и существуют устройства, способные в выключенном состоянии пропускать сквозь себя данные без обработки, но маршрутизаторы редко относятся к таковым.

Шлюз в традиционном смысле — это граничное устройство, через которое локальный трафик перенаправляется в глобальную сеть. Таким устройством обычно является маршрутизатор, выполняющий также функции трансляции сетевых адресов. NAT-демон, работающий на маршрутизаторе, формирует псевдонимы адресов локальной сети, преобразуя их в реально существующие адреса. NAT может быть реализован в вариантах: один — одному, многие — одному и один — многим.

Чтобы использовать FreeBSD в качестве NAT-маршрутизатора, необходимо сконфигурировать и запустить демон **natd**, который подробно описан в главе 28.

Имена хостов и доменов

На заре развития Internet все хосты, его составлявшие, адресовались исключительно в числовом виде, т.е. в виде IP-адресов. Когда сеть Internet выросла, число хостов в ней увеличилось, и стало просто невозможно удержать в голове все эти длинные числовые последовательности. Чтобы решить эту проблему, была создана система доменных имен, которая поставила в соответствие IP-адресам осмысленные текстовые имена. Ею мы и пользуемся сегодня.

DNS-записи хранятся в базе данных на многочисленных корневых серверах, которые поддерживаются компанией Network System Inc. Эти серверы обновляют свои базы благодаря считыванию информации с серверов, которые являются уполномоченными продавцами доменных имен. Причем каждый такой DNS-сервер хранит в своей базе данных запись о себе самом. FreeBSD использует демон BIND, который хранит свои DNS-записи в файле **/etc/namedb**. Подробное описание настройки этого демона приводится в главе 30.

Внутри каждого домена, каждый его компьютер обладает своим именем хоста, например **tiger** или **Pluto**, с соответствующей записью в базе доменного сервера. Сочетание этого имени и имени домена дает полное имя хоста в Internet, например **tiger.somethere.com**.

Очень важно понимать, что иерархическая структура доменных имен не имеет ничего общего со структурой IP-адресов. Так, например,домен **somethere.com** может иметь любой IP-адрес, главное, чтобы это доменное имя и соответствующий ему IP-адрес был правильно указан на сервере, контролирующем данную зону доменных имен. Например, HTTP-сервер с именем **www.somethere.com** может иметь адрес 64.41.131.45, а FTP-сервер с именем **ftp.somethere.com** — адрес 213.11.31.221. Кажущееся несоответствие адресов в одном домене не мешает нормальной работе, поскольку DNS полностью вне стека протоколов TCP/IP.

Однако доменные имена играют важную роль на прикладном уровне. Можно провести такую аналогию: IP-адреса — это логическая абстракция над реальными MAC-адресами, а доменные имена — это логическая абстракция над IP-адресами. И точно так, как ARP находит соответствие между IP- и MAC-адресами, DNS находит соответствие между доменными именами и IP-адресами. Итак, на прикладном уровне — доменные имена, на сетевом — IP-адреса, а на физическом — MAC-адреса.

DHCP

Ранее уже обсуждалось, как осуществляется привязка IP-адресов к сетевым Ethernet-интерфейсам, подробное рассмотрение этого вопроса ждет вас в 23 главе, а мы обратим внимание на DHCP (Dynamic Host Configuration Protocol, протокол динамического конфигурирования хостов). Этот протокол по мере надобности динамически назначает IP-адреса сетевому интерфейсу.

Если интерфейс сконфигурирован как пользователь DHCP, то при загрузке он посылает запрос на DHCP-сервер. Этот запрос (по сути, он является обратным ARP-запросом) содержит аппаратный адрес интерфейса. В ответ на него DHCP-сервер высылает временный IP-адрес и сетевую маску, назначенную данному интерфейсу на время его работы (значение адреса сервер извлекает из пула свободных адресов, при этом выдаваемому адресу назначается некий период использования). Подробное описание данного протокола приводится в главе 33.

23 ГЛАВА

Настройка основных сетевых служб

- Конфигурирование сетевой карты
- Настройка сетевых параметров при помощи sysinstall
- Настройка сетевых параметров вручную
- Создание IP-псевдонимов
- Настройка имен и IP-адресов в файле /etc/hosts
- Утилита ping
- Настройки DNS в файле /etc/resolv.conf
- Другие файлы конфигурации сети

В данной главе разговор пойдет о том, как настроить компьютер, подключенный к сети, чтобы он был способен корректно обмениваться данными с другими машинами.

Конфигурирование сетевой карты

Как уже говорилось в предыдущей главе, на компьютере могут быть установлены несколько сетевых карт одновременно, причем каждая из них может иметь несколько IP-адресов и соответствующих настроек. Да, конечно, по умолчанию подразумевается, что каждый компьютер имеет одну сетевую карту с одним IP-адресом, но это лишь частный случай. Поэтому в данной главе описано, как настроить сетевые карты и как связать с каждой из этих карт нужное число IP-адресов.

Вначале нужно удостовериться, что операционная система видит по крайней мере одну сетевую карту и нормально с ней работает. В таблице 23.1 перечислены сетевые карты, которые поддерживаются обычным ядром операционной системы FreeBSD. Это означает, что система автоматически опознает любые сетевые карты, относящиеся к типам, указанным в таблице 23.1 (PCI-карты, во всяком случае). В принципе, FreeBSD поддерживает и множество других разновидностей сетевых карт, но их драйверы не встроены в ядро. Следовательно, для настройки таких карт необходимо перекомпилировать ядро, включив в него поддержку требуемой карты (см. файл **/sys/i386/conf/NOTES**). Сам же процесс перекомпиляции ядра подробно описан в главе 17.

Таблица 23.1 Список сетевых карт, драйверы которых встроены в ядро FreeBSD по умолчанию.

	PCI Cards
de	Intel DC21x4x ("Tulip")
vx	3Com 3c590, 3c595 ("Vortex")
	PCI/MII Cards
fxp	Intel EtherExpress Pro/100B (82557, 82558)
tx	SMC 9432TX (83c170 "EPIC")
wx	Intel Gigabit Ethernet Card ("Wiseman")
dc	DEC/Intel 21143 and various workalikes
pcn	AMD Am79C79x PCI 10/100 NICs
rl	RealTek 8129/8139
sf	Adaptec AIC-6915 ("Starfire")
sis	Silicon Integrated Systems SiS 900/SiS 7016
ste	Sundance ST201 (D-Link DFE-550TX)
tl	Texas Instruments ThunderLAN
vr	VIA Rhine, Rhine II
wb	Winbond W89C840F
xl	3Com 3c90x ("Boomerang," "Cyclone")

	ISA Cards
ed	Novell NE1000/NE2000, 3Com 3c503, Western Digital/SMC 80xx
	ISA Cards
ex	Intel EtherExpress Pro/10 (82595)
ep	3Com 3c509
fe	Fujitsu MB86960A/MB86965A
ie	AT&T StarLAN 10 and EN100; 3Com 3c507; unknown NI5210; Intel EtherExpress
lnc	Lance/PCnet cards (Isolan, Novell NE2100, NE32-VL, AMD Am7990 and Am79C960)
cs	IBM Etherjet and other Crystal Semi CS89x0-based adapters
sn	SMC 9000
	PCMCIA Cards
wi	Lucent WaveLAN 802.11
an	Aironet 4500/4800 802.11
xe	Xircom/Intel EtherExpress Pro100/16

ПРИМЕЧАНИЯ

Полный список поддерживаемых сетевых карт может меняться от версии к версии. То оборудование, которое поддерживается конкретная версия операционной системы, обычно перечислено в файле **/sys/i386/conf/GENERIC** данной системы.

Большинство выпускаемых сегодня сетевых карт предназначены для PCI-шины, которая автоматически производит настройку устройств. Это значит, что отсутствует необходимость вручную настраивать такие параметры, как номер IRQ, DMA и адреса портов, назначенные данной сетевой карте (чего не скажешь об ISA-устройствах). А владельцам старых ISA-карт придется повозиться и вручную проделать ряд настроек.

Настройка сетевых параметров при помощи sysinstall

Наиболее простой способ настройки сетевой карты заключается в применении утилиты **sysinstall**. При первом же запуске этой утилиты она выводит окно, в котором можно визуально настроить все опции TCP/IP для данной сетевой карты.

Самый простой способ запуска **sysinstall**, если система загружена в стандартном многопользовательском режиме, — команда **/stand/sysinstall**. Далее необходимо в возникшем окне настройки выбрать пункт Configure из главного меню, а потом — опцию Media. Выберите пункт FTP и укажите любой FTP-сервер — появится диалоговое окно с вопросом: Running multi-user, assume that the network is already configured? (Запустить многопользовательский режим, предполагая, что сеть уже сконфигурирована?) Для входа в режим настройки сети выберите No.

Появится окно выбора настраиваемого сетевого интерфейса. В нем будет отражен список всех сетевых интерфейсов, обнаруженных FreeBSD на данном компью-

тере. Возможно, этот перечень будет содержать ряд, на первый взгляд, бессмысленных опций. Например, среди опций типа lp0 (порт принтера) и многочисленных опций типа PPP или SLIP обнаружатся что-то вроде gif0 и faith0. Они относятся к устройствам стандарта IPv6 и пока не нужны. Требуемые же опции, вероятно, будут перечислены в верхней части списка. В данном примере описана опция fxp1, соответствующая сетевой PCI-карте Intel EtherExpress Pro/100B.

Когда сетевая карта выбрана, система предложит сконфигурировать ее автоматически: сначала IPv6, а затем — DHCP. Откажитесь от этого предложения. Тогда на экране появится окно ручной настройки сети, показанное на рисунке 23.1.

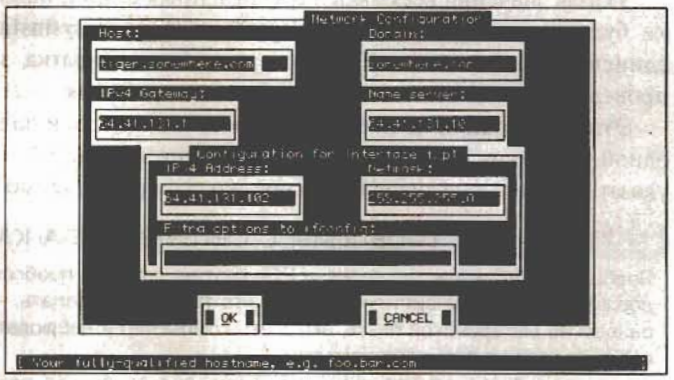


Рисунок 23.1
Окно ручной настройки сети в утилите sysinstall.

Каждый параметр в этом окне сопровождается кратким описанием в нижней части окна, здесь же приведены более подробные объяснения.

- ***Хост* (Host)**. Это поле содержит имя компьютера, являющееся частью полного доменного имени. К примеру, если полный адрес компьютера выглядит как **www.somewhere.com**, то в поле хост должно быть www.
- ***Домен (Domain)***. А здесь хранится оставшаяся часть имени, т.е. **somewhere.com**. Разумеется, имя это может быть и более сложным и включать в себя подсети, что-нибудь вроде **cslab.ivyleague.edu**.
- ***Шлюз IPv4 (IPv4 Gateway)***. Здесь указан IP-адрес маршрутизатора, играющего роль шлюза. В качестве оного следует выбирать ближайший маршрутизаторов более высокого уровня, именно он будет отвечать за переадресацию трафика с конкретного компьютера во внешний мир (и в другие подсети).
- ***Сервер имен (Name server)***. IP-адрес самого надежного сервера имен в настраиваемой подсети. В качестве сервера имен лучше всего использовать соответствующий сервер, предоставленный Интернет-провайдером.
- ***Адрес IPv4 (IPv4 Address)***. Здесь указывается IP-адрес, назначенный данной сетевой карте. Этот адрес должен входить в диапазон адресов той подсети, за которую отвечает маршрутизатор, записанный в поле IPv4 Gateway, поскольку именно маршрутизатор делит адреса на внешние и внутренние, сравнивая их с сетевой маской конкретной подсети.
- ***Сетевая маска (Netmask)***. Значение данного поля служит для определения принадлежности пакетов к локальной (внутренней) или глобальной (внешней) подсети. Для различных классов IP-адресов существуют значения маски по умол-

чанию: для класса C — 255.255.255.0, для класса B — 255.255.0.0 и для класса A — 255.0.0.0.

- ***Дополнительные опции.*** В большинстве случаев пользователь не должен специально что-то вносить в данное поле, за исключением случаев тонкой настройки, производимой администратором сети, например для повышения производительности. Любое значение, помещенное в это поле, автоматически добавляется в команду **ifconfig**, которая служит основой для работы утилиты **sysinstall**.

Указав значения всех вышеперечисленных опций, нажмите OK, и все новые настройки будут подхвачены сетевой картой на лету, а **sysinstall** попытается установить соединение с указанным FTP-сервером. Если попытка закончится неудачей, следует проверить правильность всех сделанных настроек.

Эти же операции по настройке сети выполняются и на этапе установки FreeBSD — на одной из ранних стадий установки появляется такое же окно настройки сети, где нужно указать правильные значения, особенно если речь идет об инсталляции системы по сети.

ПРОБЛЕМЫ С СЕТЕВЫМИ ISA-КАРТАМИ

Вообще-то, большинство сетевых PCI-карт не создают проблемных ситуаций. Если карта поддерживается ядром системы, то все, что требуется сделать, — это вставить сетевую карту в разъем на материнской плате, включить компьютер и наблюдать, как система сама определяет сетевую карту в процессе загрузки:

```
fxp0: <Intel PLC 10/100 Ethernet> port 0xde80-0xdebf mem 0xff8fe000-0xff8fefff irq 11
at device 8.0 on pci1
fxp0: Ethernet address 00:d0:b7:c7:74:f1
fxp1: <Intel Pro 10/100B/100+ Ethernet> port 0xdf00-0xdf3f mem 0xff700000-
0xff7fffff,0xff8ff000-0xff8fffff irq 11 at device 9.0 on pci1
fxp1: Ethernet address 00:d0:b7:bd:5d:13
```

Однако процесс настройки сетевых ISA-карт может оказаться весьма нетривиальным, а порой даже мучительным. Дело в том, что для ISA-карт требуется вручную указать значения номера прерывания и диапазон адресов памяти, причем выбранные значения не должны использоваться каким-либо другим устройством. Большинство карт настроены на использование адресного пространства памяти 0x300 или 0x280, а также IRQ 9 или 10. И если на одном компьютере должно быть более одной сетевой карты, то нужно указать каждой карте ресурс, который не используется другими устройствами. Эта настройка выполняется при помощи перемычек или DOS-утилиты, входящей в набор драйверов конкретной карты.

При начальной инсталляции FreeBSD установить причины конфликтов, вызванных сетевой картой, довольно легко — это делается при помощи все того же меню настройки. Удалите все ненужные драйверы, и если после этого конфликт не разрешиться, то придется настраивать каждую сетевую карту вручную. А для этого нужно загрузиться в режиме MS DOS и запустить утилиту, настраивающую данную сетевую карту. Утилита предоставит список значений ресурсов, которые могут быть использованы данным устройством. Из этих значений выберите те, которые не используются другой сетевой картой, и перезагрузитесь во FreeBSD.

Возможно, что вам понадобятся альтернативные записи для сетевой карты, описанные в главе 17. Они хранятся в файле **/boot/device.hints**, где система рассчитывает найти данное устройство. В этом файле можно указать тип вашей сетевой карты следующим образом:

```
hint.ed.0.at="isa"
hint.ed.0.port="0x300"
hint.ed.0.irq="10"
```

Достаточно изменить номер прерывания и диапазон адресов памяти, используемых настраиваемой картой, а затем перезагрузиться для их активации.

Настройка сетевых параметров вручную

Безусловно, использование утилиты **sysinstall** сильно облегчает настройку сети, но данная утилита недостаточна гибкая, чтобы решать некоторые задачи, возникающие при конфигурировании сети. В данном разделе описывается, как редактировать сетевые настройки из командной строки. Естественно, что все эти действия может выполнять только суперпользователь.

ПРЕДОСТЕРЕЖЕНИЕ

Целый ряд функций, обсуждаемых в этом разделе, таких как добавление IP-псевдонимов или модификация таблицы маршрутизации, недоступны из **sysinstall**. Более того, если настроить сетевую карту вручную (например, связать с ней несколько IP-адресов), а затем запустить **sysinstall**, то можно потерять настройки, сделанные из командной строки. Дело в том, что **sysinstall** подходит для решения основных задач настройки сети, но в более сложных ситуациях этой утилиты следует избегать.

Использование ifconfig

Команда **ifconfig** (сокр. от Interface Configurator) — многоцелевое средство для изменения сетевых настроек на лету (без перезагрузки). Главное применение данной команды заключается в привязке IP-адреса к сетевой карте (интерфейсу), но, как и многие другие команд UNIX, эта утилита имеет гораздо более широкую сферу применения. В этом разделе приводится описание наиболее часто используемых функций **ifconfig**. Более подробную информацию можно найти в man-страницах, посвященных этой команде.

Прежде всего, **ifconfig** поможет собрать информацию о текущем состоянии сетевых интерфейсов. Так, команда **ifconfig -a** позволяет определить: список всех интерфейсов, присутствующих в системе, а также их доступность в данный момент:

```
# ifconfig fxp1
fxp1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>  mtu  1500
        inet6 fe80::2d0:b7ff:febd:5d13%fxp1 prefixlen 64 scopeid 0x2
        inet 64.41.131.102 netmask 0xffffff00 broadcast 64.41.131.255
        ether 00:d0:b7:bd:5d:13
        media: autoselect (100baseTX <full-duplex>) status: active
        supported media: autoselect 100baseTX <full-duplex> 100baseTX
10baseT/UTP <full-duplex> 10baseT/UTP
```

ПРЕДОСТЕРЕЖЕНИЕ

Конфигурационные примеры, приводимые в этом разделе, на практике почти наверняка приведут к тому, что система при изменении параметров сетевой карты закроет все сетевые соединения. Другими словами, вряд ли стоит делать подобные настройки удаленно, используя telnet или ssh; сетевую карту нужно настраивать с консоли того компьютера, на котором эта карта установлена!

Так, например, для изменения IP-адреса, связанного с сетевой картой, нужно выполнить следующую команду:

```
# ifconfig fxp1 64.41.131.103
```

где 64.41.131.103 — новый адрес, а fxp1 — интерфейс, которому он назначается.

Такое задание IP-адреса, т.е. без указания сетевой маски, приводит к тому, что значение сетевой маски и широковещательного адреса система выставит по умолча-

нию, причем сделает это как для сети класса A, в чем можно убедиться, проверив новые настройки интерфейса fxp1:

```
inet 64.41.131.103 netmask 0xff000000 broadcast 64.255.255.255
```

Давайте заново повторим задание IP-адреса, но на этот раз для сети класса C. Для этого необходимо указать не только сам адрес, но и его сетевую маску. В команде **ifconfig** это достигается с помощью ключевого слова **network**, причем значение сетевой маски можно задавать в обычной форме, в шестнадцатиричной нотации или в виде символического имени, определенного в файле **/etc/networks**:

```
# ifconfig fxp1 64.41.131.103 netmask 255.255.255.0
# ifconfig fxp1 64.41.131.103 netmask 0xffffff00
# ifconfig fxp1 64.41.131.103 netmask your-netmask
```

В команде можно указывать также и значение широковещательного адреса, хотя это не является необходимым. Широковещательный адрес вычисляется автоматически по заданным значениям IP-адреса и сетевой маски. Однако если при настройке сетевую маску пропустить, то не будет работать обратный ARP, поскольку по умолчанию предполагается значение сетевой маски для сети класса A. При указании широковещательного адреса обязательно указывайте одновременно и маску:

```
# ifconfig fxp1 64.41.131.103 netmask 255.255.0.0 broadcast
➥64.41.255.255
```

Иногда бывает необходимо указать также значение MTU (Maximum Transmission Unit, максимальный размер пакета данных) для конкретного сетевого интерфейса. По умолчанию значение MTU для сетевых карт равно 1500 (пакет длиной в 1500 байтов). Иногда, это значение MTU приходится изменять для уменьшения задержек при работе в низкоскоростных сетях. Выбором оптимального размера пакетов обычно занят маршрутизатор, и его изменение требуется в исключительных случаях. Примером такой настройки может служить команда:

```
# ifconfig fxp1 mtu 536
```

И наконец, еще одним секретом использования **ifconfig** является применение ключевого слова **media**. Оно позволяет выбирать тип сетевой карты из списка возможных. Это очень удобно в тех случаях, когда речь идет о сетевых картах, имеющих множество интерфейсов (как, например, у карты, изображенной на рисунке 22.4 в предыдущей главе), и вы хотите иметь возможность переключаться между ними. Допустим, что в компьютере установлена сетевая карта 10/100 Ethernet, а сетевой концентратор автоматически определил скорость работы с такой картой в режиме полнодуплексной 100-мегабитной передачи, в то время как нужно ограничить эту скорость 10 Мбит, то решить эту задачу можно при помощи уже упомянутых ключевых слов **media** и **mediaopt**. Во-первых, нужно выяснить, что сообщает команда **ifconfig fxp1** о поддерживаемых типах среды передачи:

```
media: autoselect (100baseTX <full-duplex>) status: active
supported media: autoselect 100baseTX <full-duplex> 100baseTX
10baseT/UTP <full-duplex> 10baseT/UTP
```

Вывод команды **ifconfig** показывает, что в текущий момент сетевая карта автоматически настроена на максимально доступную для данной сети скорость (100baseTX <full-duplex>). Давайте перенастроим карту в режим работы 10baseT. Строка сообщает, что ключевой параметр для этого режима — 10baseT/UTP, к тому же доступен

режим полного дуплекса, который можно активизировать с помощью ключевого слова **mediaopt**. Все эти опции указываются в рамках листинга в угловых скобках, запись, данная ниже, предлагает эту же информацию в более наглядной форме:

```
autoselect
100baseTX  <full-duplex>
100baseTX
10baseT/UTP <full-duplex>
10baseT/UTP
```

Итак, для установки данного интерфейса в один из возможных режимов 10-мегабитной передачи (дуплекс или полудуплекс) применяются команды:

```
# ifconfig fxp1 media 10baseT/UTP
# ifconfig fxp1 media 10baseT/UTP mediaopt full-duplex
```

Ну а для обратной операции, т.е. для возврата к 100-мегабитной скорости работы, нужно выполнить команду:

```
# ifconfig fxp1 media autoselect
```

Помните, что режим полного дуплекса означает возможность одновременного приема и передачи данных, так что режим 10baseT с полным дуплексом означает полные 10 Мбит в обоих направлениях, в то время как режим 10baseT с полудуплексом обеспечивает лишь суммарную скорость приема-передачи в 10 Мбит, т.е. второй вариант работает значительно медленнее. Причем концентратор в принципе не понимает полного дуплекса, работать в таком более скоростном режиме умеет только коммутатор.

СОВЕТ

Для более углубленного изучения этих вопросов полистайте man-страницы руководства, набрав, к примеру, команду **man fxp**.

Команда route и настройка шлюза

До сих пор мы не упоминали о том, как задать маршрутизатор, являющийся шлюзом, или DNS-сервер. А ведь это обязательная часть сетевых настроек (в рассмотренной выше утилите **sysinstall** они определяются одновременно). Начнем с маршрутизатора.

Определить шлюз в **ifconfig** нельзя, поскольку этот адрес не привязан к какому-либо конкретному сетевому интерфейсу. Однако таблица маршрутизации FreeBSD, позволяющая системе выполнять функции полноценного аппаратного маршрутизатора (см. главу 28), содержит запись маршрута по умолчанию (default route). По этому маршруту уходят все пакеты, адресованные во внешний мир. Именно установка маршрута по умолчанию и является настройкой шлюза.

Освоение команды **route** — чертовски трудная задача, не менее важная, чем изучение **ifconfig**. Однако для начала достаточно рассмотреть лишь два ключевых слова этой команды — **add** и **delete**.

ПРИМЕЧАНИЕ

Пока настройки выполняются из той же локальной сети, к которой принадлежит и настраиваемый хост под управлением FreeBSD, можно безбоязненно менять все маршруты — включая и маршрут по умолчанию. Потери сетевого соединения (ввиду изменений настроек этого самого

соединения) не будет. Но если попытаться сделать это при помощи удаленной telnet- или ssh-сессии, то вполне вероятна внезапная потеря сетевого соединения. Это аналогично тому, что происходит при работе с **ifconfig**.

В любом случае, прежде чем приступать к настройке, нужно проверить результат работы команды **netstat -rn**. Эта команда отобразит маршрут по умолчанию в текущих настройках:

```
# netstat -rn
Routing tables
Internet:
Destination   Gateway          Flags      Refs  Use   Netif  Expire
default       64.41.131.1      UGSc       1     1     fxp0
...
```

Кажется, что адрес шлюза уже настроен. Возможны два варианта: либо сеть с имеющейся настройкой работает правильно, либо не работает вообще, причем второе вероятнее. Дело в том, что команда **route** не контролирует корректность настроек. Поэтому можно задать такой маршрут по умолчанию, который в принципе недостижим из данной подсети. Не говоря уж об установке по умолчанию фиктивного маршрута. Процесс установки маршрута по умолчанию состоит из двух обязательных шагов: удаления записи о существующем маршруте и добавления записи о новом, заведомо верном.

```
# route delete default
delete net default
# route add default 64.2.43.1
add net default: gateway 64.2.43.1
```

ПРИМЕЧАНИЕ

Команда **route** может вывести из себя кого угодно, поскольку в различных вариантах UNIX она воплощена по-разному. Причем различия эти едва различимы и относятся к синтаксису. Хотя функции, предоставляемые любой UNIX-платформой (будь то FreeBSD, Solaris, IRIX или Linux), очень похожи, способы получения этой функциональности могут быть очень разными. Чтобы получить наглядное представление о вышесказанном, проинсталлируйте сервис **portsentry** (из **/usr/ports/security**) и посмотрите содержимое файла **/usr/local/etc/portsentry.conf**. Этот конфигурационный файл, как правило, содержит не менее 9 различных способов (для 9 различных платформ) установки тупикового маршрута, так называемой черной дыры.

Имя хоста (hostname)

Установка имени компьютера (hostname) — задача простая. Она решается с помощью команды **hostname** с одним-единственным параметром — полным доменным именем компьютера:

```
# hostname tiger.somewhere.com
```

Та же команда позволяет узнать текущее имя компьютера, причем как в полной (с доменной частью), так и в краткой (только имя хоста) форме:

```
# hostname
tiger.somewhere.com
# hostname -s
tiger
```

Сетевые настройки в файле /etc/rc.conf

Итак, мы познакомились с инструментарием, позволяющим настраивать IP-адрес, сетевую маску, имя хоста и маршрут по умолчанию, а теперь посмотрим, как все эти параметры можно настроить путем редактирования конфигурационного файла **/etc/rc.conf** (конфигурирование системы описано в главе 11).

Настройки по умолчанию система хранит в файле **/etc/defaults/rc.conf**, который лучше не трогать, а вот настройки, сделанные вами, можно найти в файле **/etc/rc.conf**. Нужно учитывать, что установки по умолчанию пытаются охватить все возможные варианты конфигурации. Это значит, что в вашем конкретном случае их придется доработать, чтобы сделать работоспособными. Если при настройке Ethernet-карты вы пользовались утилитой **sysinstall**, то в результате ее работы в **/etc/rc.conf** были добавлены записи:

```
# -- sysinstall generated deltas -- #
ifconfig_fxp1="inet 64.41.131.102 netmask 255.255.255.0"
network_interfaces="fxp1 fxp0 lo0"
defaultrouter="64.41.131.1"
hostname="tiger.somewhere.com"
```

ПРИМЕЧАНИЕ

Порядок записей **/etc/rc.conf** не имеет значения, поскольку сценарий загрузки считывает все эти параметры одновременно, каждый параметр — в соответствующую переменную окружения.

Как можно видеть, параметры, записанные в конфигурационный файл утилитой **sysinstall**, включают в себя перечень всех сетевых интерфейсов, установленных в системе (строка **network_interfaces**); строки **ifconfig_xxx#,** описывающие каждый отдельно взятый интерфейс; и строки, содержащие имя хоста и маршрут по умолчанию (**hostname** и **defaultrouter**, соответственно). При загрузке системы все эти значения считываются в сценарий конфигурации на основании первого автоматического выполнения системой всех изученных в данной главе команд: **ifconfig**, **route,** и **hostname**.

Вы спросите, а как же DNS? Как уже говорилось, настройка DNS не настраивается при конфигурировании сети TCP/IP. DNS относится к прикладному уровню вспомогательных сервисов сети. Чтобы сеть корректно заработала, система доменных имен не нужна.

Работа с /etc/netstart

То, что система FreeBSD предоставляет пользователю сценарий **/etc/netstart**, — весьма любезно с ее стороны. Этот сценарий вовсе не является обязательной частью процесса загрузки, и его можно вообще удалить. Но учитывая те удобства, которые он дает, это было бы неразумно. По сути, во время загрузки все сетевые службы запускаются из этого сценария. Опытные системные администраторы помнят: в прошлом при добавлении каждого нового сетевого сервиса его приходилось вручную записывать в сценарии загрузки. Это была важнейшая часть конфигурирования. Что же касается сегодняшних версий FreeBSD, то в них все автоматически запускаемые при загрузке демоны четко разделены на пользовательские и системные: первые обитают в пользовательских загрузочных сценариях (это, к примеру, **/etc/netstart**), а вторые — в системных сценариях, в частности **/etc/rc.network**, выполняющихся при загрузке системы.

В функции сценария **/etc/netstart** входит отслеживание изменений в файле **/etc/rc.conf** и запуск соответствующих сетевых сервисов. Вначале считываются записи из **/etc/defaults/rc.conf**; если нужно, они заменяются соответствующими записями из **/etc/rc.conf.** После чего последовательно запускаются скрипты **/etc/rc.pccard** и **/etc/rc.network**. Следует помнить, что все записи в **/etc/defaults/rc.conf** и **/etc/rc.conf** доступны для сценария **rc.network** только во время загрузки системы и недоступны при запуске его вручную. Таким образом, сценарий **/etc/netstart**, по сути, является для **rc.network** оболочкой, позволяющей перезапускать сетевые сервисы в любое время.

Все рассмотренные нами команды — **ifconfig**, **route**, **hostname** — при загрузке запускаются из сценария **/etc/rc.network**, при этом учитываются различные условия и выполняется ряд проверок, что позволяет исключить использование неправильных синтаксических конструкций и неправомерных действий. Сценарий **/etc/netstart** — лучший способ ввести в действие текущие изменения в сетевых настройках. В любом случае это удобнее, чем ручной индивидуальный запуск каждой отдельной программы.

Однако есть здесь и подводные камни. Дело в том, что сценарий **/etc/netstart** осуществляет настройку маршрута по умолчанию при помощи команды **route**; но, как вы помните, нельзя добавить новый маршрут по умолчанию, не удалив старый. Беда в том, что сценарий **/etc/rc.network** (оболочкой которого является **/etc/netstart**) изначально предназначался для работы во время загрузки системы, когда параметры настройки, включая маршрут по умолчанию, еще не определены системой, а значит, и удалять командой **route delete default** ничего не нужно. Отсюда следует, что использование сценарий **/etc/netstart** предполагает два этапа, описанных в листинге 23.1:

Листинг 23.1 Перезапуск сетевых настроек при помощи /etc/netstart

```
# route delete default
delete net default
# /etc/netstart
Doing stage one network startup:
Doing initial network setup:.
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet6 fe80::2d0:b7ff:fec7:74f1%fxp0 prefixlen 64 scopeid
0x1
        inet 64.41.131.101 netmask 0xffffff00 broadcast
64.41.131.255
        ether 00:d0:b7:c7:74:f1
        media: autoselect (100baseTX <full-duplex>) status: active
        supported media: autoselect 100baseTX <full-duplex>
100baseTX 10baseT/UTP <full-duplex> 10baseT/UTP
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x9
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
add net default: gateway 64.41.131.1
Additional routing options: tcp extensions=NO TCP keepalive=YES.
Routing daemons:.
```

Все это отлично работает, если сценарий запускается локально. Но если вы управляете этим процессом откуда-то из локальной сети или Интернет (посредством telnet, например), вас поджидает коварная ловушка. Как только на удаленном хосте будет уничтожен маршрут по умолчанию, связь прервется. Таким образом, запустить вдогонку сценарий **/etc/netstart**, который бы переопределил новый маршрут по умол-

чанию, уже не удастся. Впрочем, есть одна хитрость, позволяющая избежать этой западни. Она заключается в том, что обе команды (первая удаляет маршрут по умолчанию, вторая задает его новое значение) пересылаются на настраиваемый компьютер одновременно. Это можно сделать, набрав обе команды, разделенные точкой с запятой (;), в одной командной строке. Конечно, это рискованно, но если все пройдет успешно, то удаленный хост выполнит обе команды, и сетевое соединение будет почти мгновенно восстановлено:

```
# route delete default; /etc/netstart
```

Создание IP-псевдонимов

Не существует причины, из-за которой нельзя назначить одной сетевой карте сразу несколько IP-адресов. При доставке пакета система ищет хост по его IP-адресу, запрашивая аппаратный адрес сетевой карты, к которой привязан искомый IP-адрес. А вот назначить один IP-адрес сразу нескольким интерфейсам нельзя. Одна сетевая карта может откликаться на несколько IP-адресов, но один и тот же IP-адрес не может быть присвоен нескольким MAC-адресам, иначе в сети возникнут ошибки и коллизии.

Способ, позволяющий закреплять несколько IP-адресов за одним аппаратным адресом (т.е. за одной сетевой картой) получил название IP-алиасинг (IP-aliasing). Причем, как и в случае с командой **route**, в разных UNIX-системах синтаксис назначения псевдонимов отличается. Во FreeBSD для этой цели служит команды **ifconfig** с ключевым словом **alias**, причем это слово должно быть самым последним параметром:

```
# ifconfig fxp1 64.41.131.103 netmask 255.255.255.255 alias
```

Обратите внимание на то, что здесь значение сетевой маски равно 255.255.255.255! Это обязательное условие при назначении такого IP-псевдонима, который принадлежит к той же локальной сети, что и основной IP-адрес настраиваемого интерфейса. (В противном случае нужно использовать обычное для данной подсети значение маски.) Что же все это значит с точки зрения работы TCP/IP? Сетевая маска со всеми битами, равными 1, обеспечивает состояние, при котором стек TCP/IP будет обрабатывать только те пакеты, в которых адрес назначения во всех битах совпадает с номером подсети. Иначе говоря, создается подсеть с одним-единственным адресом. Таким образом, все пакеты, направляемые как на этот адрес-псевдоним, так и от него, посылаются исключительно через маршрутизатор. Псевдоним считается не принадлежащим к той подсети, в которой находится компьютер с данной сетевой картой. Если же многочисленные псевдонимы будут иметь одинаковую сетевую маску, то их широковещательные адреса также будут одинаковыми, что приведет к сбоям в работе стека протоколов TCP/IP. Использование маски со всеми единицами — это способ обмануть **ifconfig** и соотнести с одним интерфейсом несколько IP-адресов.

Для того чтобы занести псевдонимы в файл **/etc/rc.conf** необходимо использовать запись вида ifconfig_xxx#_alias#, которая синтаксически работает подобно записи ifconfig_xxx#. Вот пример того, как могут выглядеть записи о псевдонимах в файле **/etc/rc.conf**:

```
ifconfig_fxp1="inet  64.41.131.131  netmask  255.255.255.0"
ifconfig_fxp1_alias0="inet  64.41.131.132  netmask  255.255.255.255"
ifconfig_fxp1_alias1="inet  64.41.131.133  netmask  255.255.255.255"
```

```
ifconfig_fxp1_alias2="inet 64.41.131.134 netmask 255.255.255.255"
ifconfig_fxp1_alias3="inet 64.41.132.161 netmask 255.255.255.0"
ifconfig_fxp1_alias4="inet 64.41.132.165 netmask 255.255.255.255"
ifconfig_fxp1_alias5="inet 64.41.132.166 netmask 255.255.255.255"
```

Настройка имен и IP-адресов в файле /etc/hosts

Для удобства пользователей необходимо преобразовывать имена хостов в соответствующие им IP-адреса. За решение этого вопроса отвечает служба DNS, но она пока не настроена. Для компьютеров локальной сети эту задачу можно решить при помощи файла **/etc/hosts**.

Файл **/etc/hosts** содержит записи, в каждой из которых указаны: IP-адрес, официальное имя хоста и другие имена-псевдонимы, разделенные пробелом или знаком табуляцией. Причем в момент разрешения имен, система, прежде чем формировать DNS-запрос, в первую очередь консультируется именно с записями в этом файле, другими словами, данный файл является в одно и тоже время как резервной копией сервера имен, так и перекрывающей его информацией о именах в тех случаях, когда сервер имен лишен корректных сведений. Ну а имена-псевдонимы используются как сокращенные имена для обозначения хостов сети.

Вот пример нескольких записей в **/etc/hosts**:

```
64.41.131.132       ns              ns.somewhere.com lion.somewhere.com
64.41.131.133       www2            www2.somewhere.com
64.41.132.165       www3            www3.somewhere.com
114.235.123.11      www.foobar.com  fred
```

При установлении сетевого сеанса связи с другим компьютером его можно вызывать либо по имени (вторая колонка), либо по псевдониму (третья колонка), а не по его IP-адресу (первая колонка).

Утилита ping

Сделав все необходимые сетевые настройки (с помощью **sysinstall** или **/etc/netstart**), необходимо убедиться, что сеть с настраиваемого компьютера видна и доступна. Сделать это можно с помощью программы **ping**, основанной на протоколе ICMP. Она посылает запрос на удаленный хост. Получив ответ или не дождавшись его, **ping** выдает статистику обмена пакетами между хостами: время пути в оба конца, процент потерь и т.п.

Использовать программу **ping** очень просто. В качестве параметра можно указывать как адрес удаленного компьютера, так и его имя (или псевдоним). Остановить ее работу можно, нажав Ctrl+C:

```
# ping fred
PING fred (114.235.123.11): 56 data bytes
64 bytes from 114.235.123.11: icmp_seq=0 ttl=243 time=485.344 ms
64 bytes from 114.235.123.11: icmp_seq=1 ttl=243 time=351.589 ms
^C
-- fred ping statistics --
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 351.589/418.466/485.344/66.877 ms
```

Здесь мы видим результаты работы программы **ping** при корректных настройках TCP/IP: хост по имени fred ответил на эхо-запрос. Но если ответ на запрос не пришел в течение определенного времени (тайм-аута), то удаленный компьютер считается недостижимым и программа **ping** выдает отчет о стопроцентной потере пакетов. Это может означать и неверные настройки хоста, и обрыв кабеля, и неправильную работу DNS (если имел место запрос по имени), и то, что удаленный компьютер выключен.

```
# ping 64.41.131.133
PING 64.41.131.133 (64.41.131.133): 56 data bytes
ping: sendto: Host is down
ping: sendto: Host is down
ping: sendto: Host is down
ping: sendto: Host is down
^C
-- 64.41.131.133 ping statistics --
10 packets transmitted, 0 packets received, 100% packet loss
```

Настройки DNS в файле /etc/resolv.conf

Чтобы работать с полноценным DNS-сервисом, необходимо иметь доступ как минимум к одному серверу имен, где бы он ни находился. Большинство серверов имен способны обрабатывать DNS-запросы, приходящие из любой части сети Интернет. Список таких серверов хранится в файле **/etc/resolv.conf**.

```
search somewhere.com
nameserver 64.41.131.132
nameserver 207.78.98.20
nameserver 64.40.111.102
```

Ключевое слово **nameserver** служит для задания адреса сервера имен, а порядок, в котором они перечисляются в файле, определяет последовательность обращения к ним — если в течение заданного времени первый сервер из списка не прислал ответа, то запрос направляется второму серверу и т.д.

Ключевое слово **search** определяет домен, в котором осуществляется поиск компьютера с указанным именем, т.е. краткое имя преобразовывается в полное. К примеру, по команде **nslookup www**, осуществляется поиск адреса компьютера с полным именем **www.somewhere.com**. Причем в строке **search** файла **/etc/resolv.conf** можно указать несколько доменов, в которых должен осуществляться поиск:

```
search somewhere.com foobar.com cslab.ivyleague.edu
```

Технически, файл **/etc/resolv.conf** — это лишь запасной вариант, обеспечивающий работу DNS, что-то вроде знаменитого файла **/etc/hosts**. На самом деле система должна запустить свой собственный DNS-демон. Все, что касается запуска сервера имен, будет рассмотрено в главе 30, целиком посвященной этой теме.

СОВЕТ

Файл **resolv.conf** можно редактировать в любое время и без помощи каких-либо специальных команд для обновления баз данных имен. DNS-запросы генерируются сетевыми прикладными программами, а таблица, содержащаяся в файле **resolv.conf**, каждый раз считывается заново. Поэтому можно открыть этот файл в любом текстовом редакторе, внести нужные изменения и сохранить изменения, после чего следующий DNS-запрос сформируется на основе обновленного файла **resolv.conf**.

Другие файлы конфигурации сети

Существует еще несколько ранее не упоминавшихся файлов конфигурации, которые достойны краткого описания. Кстати, большинство файлов конфигурации снабжены руководствами по их заполнению и настройке. Чтобы открыть такое руководство просто, выполните команду **man <filename>**, к примеру **man inetd.conf**.

/etc/networks: Файл содержит символические имена подсетей, которые используются, к примеру, в таблице маршрутизации.

/etc/hosts.allow: Это список правил безопасности, оговаривающий, с какими сервисами каких хостов разрешен сетевой обмен для данного компьютера. Можно также указать, какие действия следует предпринять в том случае, если обнаружена попытка установления запрещенного соединение.

/etc/inetd.conf: Демон **inetd**, или суперсервер, отвечает за прослушивание сетевых портов и запуск соответствующего конкретному порту демона, если обнаружена попытка установить соединение с данным портом. Все разрешенные к запуску демоны должны быть указаны в **/etc/inetd.conf** (прослушиваются только порты, указанные в этом файле), а сами соединения должны удовлетворять установленным правилам безопасности для данного порта. Более подробно данный вопрос обсуждается в главе 29.

/etc/services: База данных, упорядоченная по типам существующих (но не обязательно активных) IP-сервисов, с привязкой каждого сервиса к конкретному порту. Здесь хранится полезная справочная информация по сервисам и портам, как для программ, работающих с сетью, так и для системного администратора сети.

/etc/protocols: Содержит сведения о различных сетевых протоколах. Интересен в качестве эталонной информации об IP-протоколах.

/etc/rpc: Содержит сведения по таким сервисам, как NFS и NIS, их портам, а также информацию о различных программах, отвечающих за формирования отчетов по состоянию частей системы.

/etc/pam.conf: Pluggable Authentication Modules (подключаемые модули аутентификации) отвечают за работу многоступенчатых систем аутентификации пользователя (например, S/Key, Kerberos и др.). Более подробная информация дана в главе 29.

/etc/host.conf: Этот файл определяет порядок, в котором производится поиск именованных ресурсов, позволяет задать приоритеты этого поиска и т.д. К примеру, настройка по умолчанию содержит запись **hosts**, за которой следует запись **bind**. Это означает, что при разрешении имен, система вначале консультируется с файлом **hosts**, а затем обращается к серверу имен. В пятой версии FreeBSD этот файл был заменен файлом **/etc/nsswitch.conf**. Его формат несколько отличается от формата файла **hosts**, так что почитайте руководство **man nsswitch.conf**.

24 ГЛАВА

Подсоединение к Интернет с помощью PPP

- Выбор провайдера
- Информация, необходимая для настройки
- Сравнение kernel PPP и user PPP
- Конфигурирование kernel PPP
- Пользовательский PPP
- Решение проблем
- Последние замечания

PPP (Point to Point Protocol) — это протокол двухточечной связи; чаще всего он используется для подсоединения к Интернет с помощью модема. Операционная система FreeBSD включает PPP в двух формах — демон **pppd**, встроенный в ядро системы, и PPP, запускаемый как пользовательская программа. Оба эти компонента имеют свои преимущества и недостатки, которые мы и обсудим.

Выбор Интернет-провайдера

Чтобы подключиться к Интернет, необходимо заключить соглашение с поставщиком Интернет-услуг. В крупных городах найдется не один, а несколько Интернет-провайдеров, так что пользователь может выбирать. Вероятно, большинство ищет провайдера, предоставляющего за минимальную цену неограниченный по времени и трафику доступ в Интернет. Однако существуют и другие, не менее важные, критерии выбора провайдера:

- Каково отношение количества клиентов данного провайдера к числу принадлежащих ему телефонных линий, выделенных под Интернет? Ведь если клиентов в десятки раз больше, чем линий, линии будут постоянно заняты.
- Каково отношение пропускной способности внешнего канала провайдера к суммарной пропускной способности его входных линий?
- Поддерживает ли провайдер FreeBSD? Некоторые провайдеры не предоставляют возможности подключения к ним хостов под управлением FreeBSD (только Windows или Macintosh).

Информация, необходимая для настройки

Чтобы подключиться к провайдеру, необходимо предварительно выяснить ряд технических аспектов:

- Номер последовательного порта, к которому подключен модем на компьютере у клиента. DOS-обозначения портов соответствуют определенным UNIX-обозначениям тех же портов: com1 — cua0, com2 — cua1, com3 — cua2.
- IP-адреса DNS-серверов провайдера, что позволит преобразовывать доменные имена в соответствующие числовые значения IP-адресов.
- Телефон(ы) dialup-доступа к провайдеру, т.е. телефоны его входных линий.
- Имя пользователя и пароль (**login** и **password**), под которыми вы зарегистрированы у провайдера.
- Тип используемой провайдером аутентификации. Существует три варианта обмена информацией между провайдером и клиентом при установке соединения: стандартный запрос login в командной строке, *PAP* (Password Authentication Protocol) или *CHAP* (Challenge Handshake Protocol), описание каждого из них приводится далее.
- Какой IP-адрес выделяет провайдер клиенту — статический (если да, то какой именно) или динамический?

Очевидно, что всю эту информацию, за исключением первого пункта, должен предоставить клиенту провайдер.

Сравнение kernel PPP и user PPP

В операционной системе FreeBSD доступны два типа PPP — пользовательский и встроенный в ядро системы. Оба они имеют свои преимущества и недостатки. Так, пользовательский PPP легче настроить, чем встроенный в ядро, но при своей работе он использует так называемое туннельное устройство, что снижает скорость работы. PPP, встроенный в ядро, более эффективен, поскольку работает в качестве демона, но при этом необходимо, чтобы ядро было скомпилировано с опцией поддержки PPP.

Конфигурирование kernel PPP

Данный тип PPP поддерживается **pppd**-демоном, основные файлы конфигурации которого размещены в каталоге **/etc/ppp**. Кроме файлов конфигурации самого демона, нужно будет отредактировать еще и файл **/etc/resolv.conf**.

/etc/resolv.conf

Информация, содержащаяся в **/etc/resolv.conf** определяет способ преобразования системой имен хостов в IP-адреса. В этом файле используется ключевое слово **domain**. В строке **domain** указывается доменное имя по умолчанию, т.е. считается, что те имена хостов, которые представлены в виде неполного доменного имени, автоматически принадлежат к домену по умолчанию. Каждая следующая строка этого файла содержит ключевое слово **nameserver**, за которым следует IP-адрес DNS-сервера. Вот как примерно может выглядеть файл **/etc/resolv.conf**:

```
domain   samplenet.org
nameserver   111.111.11.1
nameserver   222.222.22.2
```

Данный файл может редактировать только пользователь **root**.

Файл /etc/ppp/options

Этот файл, как правило, содержит опции, с учетом которых выполняется демон **pppd**, причем при запуске демона, этот список считывается системой до того, как принимаются во внимание опции, заданные пользователем в командной строке (в команде запуска **pppd**). Конкретные опции, заданные в этом файле, на различных хостах могут отличаться, все зависит от типа аутентификации, используемого провайдером. Далее приведен пример списка опций для динамического IP-адреса и аутентификации для входа в сеть в виде текстового приглашения в командной оболочке. PAP- и CHAP-аутентификация будут рассмотрены в следующем разделе.

```
/dev/cuaa0   115200
crtscts
modem
connect   "/usr/sbin/chat   -f   /etc/ppp/opions/chat.script"
noipdefault
silent
domain   samplenet.org
defaultroute
```

Приведенные здесь опции означают следующее:

- **/dev/cuaa0 115200**: данная строка определяет последовательный порт, к которому подключен модем (в данном случае — com1) и максимальную скорость между модемом и портом, равную 115200 бит/с;
- **crtscts**: эта опция устанавливает аппаратный контроль за приемом/передачей данных; необходима для высокоскоростных соединений;
- **modem**: данная опция настраивает демон **pppd** так, чтобы он проверял наличие сигнала CD (Carrier Detect) от модема, после чего осуществляет открытие порта;
- **connect**: эта опция служит для задания программы дозвона и ее сценария; как правило, для автоматизации установки модемного соединения используется программа **chat**, сценарии которой подробно описаны далее;
- **noipdefault**: указывает, что IP-адрес назначается динамически;
- **silent**: указывает **pppd** на необходимость ожидать прихода LCP-пакетов (Line Control Protocol — протокол контроля за линией);
- **domain**: задает доменное имя провайдера, к которому осуществляется подключение, что иногда необходимо для аутентификации (доменное имя добавляется к имени хоста);
- **defaultroute**: при включении данной опции в таблицу маршрутизации на время работы ppp-соединения добавляется запись о данном маршруте, которая удаляется при закрытии соединения.

Если необходимо установить соединение со статическим IP-адресом, вместо опции **noipdefault** нужно указать что-нибудь вроде 333.333.333.33:444.444.444.44, где первое число соответствует IP-адресу, выделенному вам провайдером, а второе число (после двоеточия) является IP-адресом сервера провайдера, к которому и осуществляется подключение, т.е. это адрес шлюза. Если провайдер не сообщил вам адрес шлюза, то второе число можно не указывать (но двоеточие после первого адреса должно быть все равно).

СОВЕТ

На старых компьютерах (скажем, с 486-м процессором) с внешним модемом при использовании вышеприведенных опций возможно появление разных глюков. Это вызвано тем, что в таких системах для работы последовательного порта применяется устаревшая версия чипа UART, который не поддерживает скорость 115200 бит/с. Придется задать значение скорости 57600 бит/с или меньше.

Сценарий программы chat

Демон **pppd** не имеет встроенной функцией дозвона — для этой цели служит программа **chat**. Используя сценарий, состоящий из пар "послать запрос" — "дождаться ответа", эта программа позволяет автоматизировать общение с модемом. Другими словами, ее сценарий содержит последовательность символов, которую программа **chat** должна послать модему, и последовательность, которую модем должен прислать в ответ при успешном выполнении команды. Получив подтверждение от модема, программа **chat** отправляет на модем следующую последовательность символов (последовательности разделены пробелами или знаком табуляции, если же какая-то последовательность содержит пробелы, то ее необходимо заключать в кавычки).

Вот пример простого сценария. Он создается в обычном текстовом редакторе, но нужны права **root**:

```
ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDT5551212 CONNECT "" TIMEOUT
   10 ogin: foo TIMEOUT 5 sword: bar
```

Расшифровывается эта запись так: ABORT — указание прервать сеанс связи, если от модема будет получен ответ BUSY или NO CARRIER. Далее следует команда AT (сокращение от англ. Attention — внимание), в ответ на которую от модема ожидается OK. Затем модему посылается следующая команда — ATDT (Attention Dial Tone) и номер телефона. Она указывает модему, что нужно набрать номер (для импульсных АТС, применяется команда ATDP). От модема ожидается ответ CONNECT, по получении которого устанавливается тайм-аут в 10 секунд, в течение которого от модема должна поступить строка **ogin** (с учетом того, что от сервера провайдера может прийти ответ как **login**, так и **Login**). Это приглашение к регистрации. Программа **chat** отвечает foo (в реальном сценарии должен быть указан ваш **login**) и 5 секунд (TIMEOUT 5) ожидает приглашения на ввод пароля — **sword** (часть слова **password**). Получив его, программа отправляет ответ bar (здесь должен быть ваш пароль). Если провайдер после получения пароля автоматически переключается в режим приема-передачи (так называемый PPP-режим), то сценарий на этом и заканчивается. Если же нет, то сценарий должен в качестве завершающего штриха содержать команду, переключающую сервер провайдера в этот режим — ее, очевидно, сообщит вам провайдер.

СОВЕТ

Если вы хотите уточнить, какие приглашения посылает сервер провайдера, используйте программу типа **minicom**, обеспечивающую дозвон и подключение к серверу провайдера в интерактивном режиме.

ПРЕДОСТЕРЕЖЕНИЕ

Чтобы программу **chat** мог запустить рядовой пользователь, этот сценарий должен быть общедоступным. Но это небезопасно. В этом случае следует использовать другие способы аутентификации — PAP и CHAP. При обычной аутентификации можно разрешить запуск соединения с Интернет только определенным группам пользователей.

Запуск демона pppd

Когда сделаны все вышеописанные настройки в сценариях и файлах конфигурации, можно считать PPP готовым к работе. Для его запуска достаточно ввести в командной строке **pppd**. И если все пойдет успешно, то вскоре модем дозвонится по указанному номеру и установит соединение.

Чтобы разорвать соединение, можно применить либо команду **kill** с указанием **pid** процесса (его можно узнать с помощью команды **ps**), либо **killall pppd**.

PAP- и CHAP-аутентификация

Оба этих типа аутентификации не требуют использования оболочки (shell) и запускаются при установлении соединения. Кроме того, они обеспечивают более высокий уровень безопасности. Если необходимо предоставить обычным пользователям право устанавливать соединение с Интернет, то обычный сценарий **chat** для аутентификации из командной оболочки должен быть общедоступным. А это означает, что любой, имеющий право запускать командную оболочку, может узнать пароль досту-

па в Интернет. Файлы же **pap-secrets** и **chat-secrets** доступны только пользователю **root**. Конфигурация этих файлов не так тривиальна, как обычная аутентификация, поэтому имеет смысл рассмотреть их более подробно.

Начнем, пожалуй, с файла **/etc/ppp/options**. В этот файл нужно добавить по крайней мере еще одну строку, которая определяет используемый для аутентификации профиль. Эта новая строка называется строкой пользователя (**user line**) и задает имя профиля (список профилей хранится в отдельном файле), который нужно использовать для установки данного соединения. Выглядит она примерно так: **user foo**, где вместо foo должен быть ваш **login**. Кроме этого, может потребоваться указание одной или нескольких из нижеприведенных опций:

- **refuse-chap**: если задать эту опцию, то хост откажет клиенту в CHAP-аутентификации, даже если удаленный компьютер делает запрос именно на такой тип аутентификации;
- **refuse-pap**: отказ в PAP-аутентификации;
- **require-chap**: требование использовать именно CHAP-аутентификацию;
- **require-pap**: то же самое, но для PAP.

Если не применять ни одну из опций отказа (**chap-refuse**, **pap-refuse**) в вашем файле **/etc/ppp/options**, то хост примет любой тип аутентификации, предложенный удаленным сервером, если же применить обе эти опции, то ни один из этих типов аутентификации не пройдет, что приведет к невозможности установить соединение с сервером.

pap-secrets and chap-secrets

Файлы **/etc/ppp/chap-secrets** и **/etc/ppp/pap-secrets** содержат информацию для CHAP- и PAP-аутентификации, соответственно. Оба файла составляются в одном и том же формате: **username hostname password**, где **username** — это **login**, **hostname** — имя хоста, к которому осуществляется подключение, а **password** — пароль для входа в Интернет. Причем в поле **hostname** вместо имени можно указать звездочку, что означает возможность установить соединение с любым компьютером. Итак, запись в этих файлах может выглядеть следующим образом:

```
foo  *  bar,
```

где вместо foo будет ваш **login**, а вместо bar — реальный пароль.

ПРЕДОСТЕРЕЖЕНИЕ

Еще раз особо отметим, что право доступа к файлам pap-secrets и chap-secrets должен иметь только root, в противном случае вашим паролем для доступа в Интернет сможет воспользоваться кто угодно.

Соединения по требованию и постоянные соединения

Соединение по требованию (*dial-on-demand*) *устанавливается демоном* **pppd** автоматически, когда возникает необходимость переслать в Интернет исходящий трафик (т.е. при попытке выхода в Интернет начинается автоматический дозвон). *Постоянное соединение* (*persistent connection*) поддерживается 24 часа в сутки, а при разрыве соединения выполняется автоматическое его восстановление (даже если трафик отсутствует).

Соединение по требованию

Демон **pppd** постоянно следит за наличием трафика и при его появлении начинает дозвон, чтобы установить соединение. В файле опций (**/etc/ppp/options**) этому типу соединения соответствуют следующие строки:

- **demand**: включить режим соединения по требованию;
- **idle** *n*: разрыв соединения в случае его простоя, т.е. если в течение *n* секунд отсутствует исходящий или входящий трафик.

Если хост работает в таком режиме, имеет смысл создать сценарий, который будет запускать демон **pppd** при загрузке системы. Сделать это можно несколькими способами. Первый — добавить строку **pppd** в файл **/etc/rc.local**. Но гораздо эффективнее создать сценарий, включающий единственную строку — **pppd**, и поместить его в **/usr/local/etc/rc.d**. Назвать этот сценарий можно как угодно, но лучше просто **ppp**.

СОВЕТ

Некоторые проблемы может вызвать одновременная работа демона **pppd**, настроенного на режим dial-on-demand, и демона Fetchmail, настроенного на регулярную проверку почты. Почтовые запросы создают исходящий трафик, поэтому соединение все время открыто. Внеплановые попытки дозвона могут быть вызваны и DNS-запросами. Однако наиболее вероятным "зачинщиком" этих запросов является **sendmail**. Настройка почтового сервиса подробно рассматривается в главе 25.

Постоянное соединение

Демон **pppd** можно заставить поддерживать соединение круглосуточно. Это достигается указанием опции **persist** в файле **/etc/ppp/options**.

Установление и разрыв соединения

При установлении соединения демон **pppd** выполняет сценарий **/etc/ppp/ip-up**. При разрыве соединения он отрабатывает сценарий **/etc/ppp/ip-down**. Примером применения этой особенности работы **pppd** является использование лэптопов, с которыми можно работать в Интернет, находясь в самолете или поезде. Демон Fetchmail можно запускать из сценария **ip-up** при установлении соединения и останавливать его выполнение с помощью сценария **ip-down** при разрыве соединения. Таким образом, эта почтовая программа не будет формировать запросы на проверку почты в те моменты, когда соединение отсутствует.

Пользовательский PPP

Если по какой-либо причине использование kernel PPP невозможно или нежелательно, применяется user PPP. Он не запускается в качестве демона и не требует ядра, скомпилированного с включенной поддержкой PPP. Взамен этого ядро должно быть скомпилировано с поддержкой туннельного устройства, через которое и работает user PPP.

Итак, необходимо проверить наличие туннельного устройства (по умолчанию оно имеется в ядре, скомпилированном при установке системы). Если ядро перекомпилировалось вручную, то следует проверить файл конфигурации ядра **/sys/i386/conf** на предмет наличия строки **pseudo-device tun 1**. Если эта строка существует, то ядро скомпилировано с поддержкой туннельного устройства.

Файл /etc/ppp/ppp.conf

Это файл настройки user PPP. При установке PPP система создает образец файла **ppp.conf**, который можно отредактировать с учетом конкретного хоста и сети. В образце опции настройки изначально заданы на работу с PAP- или CHAP-аутентификацией, но, чтобы пользовательский PPP работал с реальным провайдером, придется изменить несколько строк в данном файле. В листинге 24.1 приведена распечатка файла-образца с добавленными объяснениями по каждой опции из тех, что требуют изменений. Разумеется, чтобы внести эти изменения, нужны права суперпользователя.

Листинг 24.1 Файл /etc/ppp/ppp.conf

```
#####################################################################
#  PPP  Sample  Configuration  File
#  Originally  written  by  Toshiharu  OHNO
#  Simplified  5/14/1999  by  wself@cdrom.com
#
#  See  /usr/share/examples/ppp/  for  some  examples
#
#  $FreeBSD: src/etc/ppp/ppp.conf,v  1.7 2001/02/22  23:28:12  brian Exp $
#####################################################################

default:
  ident  user-ppp  VERSION  (built  COMPILATIONDATE)

  #  Убедитесь,  что  в  пункте  "set  device"  указан  последовательный
  #  порт,  к  которому  подключен  модем.  (cuaa0  =  COM1,  cuaa1  =  COM2)
  #
  set  device  /dev/cuaa1

  set  log  Phase  Chat  LCP  IPCP  CCP  tun  command
  set  speed  115200
  set  dial  "ABORT  BUSY  ABORT  NO\\sCARRIER  TIMEOUT  5  \
             \"\"  AT  OK-AT-OK  ATE1Q0  OK  \\dATDT\\T  TIMEOUT  40  CONNECT"
  set  ifaddr  10.0.0.1/0  10.0.0.2/0  255.255.255.0  0.0.0.0
  set  timeout  180                  #  3  минуты  —  тайм-аут  по  умолчанию
  add  default  HISADDR              #  Добавить  маршрут  по  умолчанию
  enable  dns                        #  запрос  DNS-информации

papchap:
  #
  #  отредактируйте  следующие  три  строки,  внеся  в  них  значения,
  #  указанные  вашим  провайдером.
  #
  set  phone  PHONE_NUM
  set  authname  USERNAME
  set  authkey  PASSWORD
```

Ниже описывается значение каждой из опций в данном файле:

- **set device**: определяет последовательный порт, к которому подключен модем, перевод ДОС-эквивалентов в UNIX-совместимые устройства таков: com1 — **/dev/cuaa0**, com2 — **/dev/cuaa1** и т.д;
- **set log**: указывает тип событий, которые нуждаются в протоколировании; весьма полезно для отладки проблемного соединения (более подробно см. раздел, посвященный разрешению проблем);
- **set speed**: определяет скорость работы последовательного порта; значение по умолчанию, 115200, должно работать в большинстве ситуаций, однако если на

старых системах возникают проблемы, имеет смысл уменьшить это значение до 57600 или меньше;

- **set dial**: строка настройка модема, аналогична сценарию **chat**; этот сценарий по умолчанию в большинстве случаев вполне работоспособен, так что не стоит менять имеющиеся настройки;
- **set ifaddr**: первое число, указанное в данной опции (10.0.0.1/0), нужно заменить на реальный статический адрес, выделенный вам провайдером; если же вы работаете с динамическим IP-адресом, то оставьте все как есть; второе число (10.0.0.2/0) нужно заменить на реальный адрес шлюза, выданный провайдером; если провайдер не сообщал адрес шлюза, то ничего изменять не нужно; и наконец, третье число в данной опции (255.255.255.0) соответствует значению сетевой маски, которое не следует изменять, если провайдер специально не оговаривал этот момент;
- **set timeout**: полный аналог опции **idle** *n*, которая задает тайм-аут; если входящий и исходящий трафик отсутствуют в течение этого времени, то происходит разрыв соединения из-за его простоя;
- **set default HISADDR**: добавляет в таблицу маршрутизации маршрут по умолчанию, т.е. адрес шлюза провайдера; нужно помнить, что система узнает этот адрес из опции **ifaddr**, которую нужно настроить заблаговременно, в противном случае PPP не будет знать, какой маршрут нужно добавить;
- **enable DNS**: задает проверку правильности указанных в **/etc/resolv.conf** серверов имен провайдера — если они некорректны (т.е. недоступны), то PPP автоматически обновляет эту информацию, используя сервер провайдера;
- **set phone**: задает номер телефона дозвона к серверу провайдера;
- **set authname**: задает логин на сервере провайдера;
- **set authkey**: задает пароль доступа к серверу провайдера;

В том, случае, если работа с PAP- или CHAP-аутентификацией по какой-либо причине не подходит (требуется стандартная аутентификация), в файл **ppp.conf** нужно будет внести еще несколько изменений. Во-первых, удалите секцию **papchap**, включая ее опции **set phone**, **set authname** и set **authkey**, во-вторых, добавьте после опции **set dial** следующие строки:

```
Provider:
      set  phone  "5551212"
      set  login  "TIMEOUT  10  \  "\  "  \  "\  "  gin:  foo  sword:  bar"
```

где слова **foo** и **bar** нужно заменить на реальные логин и пароль. И аналогично тому, как это было с PPP, встроенным в ядро, на удаленный сервер нужно будет отправить команду, которая переведет его из командной оболочки в режим PPP — эту команду вам обычно сообщает провайдер, она дописывается после пароля. В любом случае, *данный сценарий подобен сценарию* **chat**: он состоит из пар "ожидаемый ответ" — "передаваемая строка", причем передача заданной строки символов на модем осуществляется только после получения от него ожидаемого ответа, т.е. после исполнения им предыдущей команды.

Запуск пользовательского PPP

При работе с PAP- или CHAP-аутентификацией запуск user PPP осуществляется из командной строки путем ввода команды **ppp -background papchap**. А при работе с аутентификацией в командной оболочке используйте для запуска PPP команду **ppp -background provider**. Данные команды запускают пользовательский PPP в фоновом режиме. Если все пойдет нормально, то вы услышите как дозванивается модем, а после установления соединения система выдаст сообщение PPP Enabled и вернется в командную оболочку.

Для разрыва соединения нужно при помощи команды **ps** выяснить его **pid** и воспользоваться командой **kill** или **killall ppp**.

Запуск user PPP рядовым пользователями

Если необходимо, чтобы устанавливать соединение могли рядовые пользователи, файл **/etc/ppp/ppp.conf** должен содержать строку **allow users**.

Соединение по требованию и постоянное соединение

Пользовательский PPP, как и PPP, встроенный в ядро, поддерживает различные варианты соединений.

Соединение по требованию

Чтобы установить такое соединение, в команде запуска пользовательского PPP существует опция **-auto**. Команда выглядит так: **ppp -auto papchap** или **ppp -auto provider** (в зависимости от типа аутентификации). Если же необходимо устанавливать соединение при каждой загрузке, то в файл **/etc/rc.conf** нужно добавить строку **ppp_enable="YES"**. Впрочем, данная строка приводит к установке в активное состояние ряда опций, которые могут подходить, а могут и не подходить для конкретно взятого компьютера в сети:

- режим соединения по требованию: устанавливает соединение при активизации исходящего или входящего сетевого трафика и разрывает его при отсутствии оного в течение времени, заданного опцией **timeout** в файле **/etc/ppp/ppp.conf**;
- подразумевается аутентификация при помощи PAP; если реально работает аутентификация в командной строке оболочки, то в **/etc/rc.conf** нужно добавить строку **ppp_profile="provider"**;
- активизируется *NAT* (*Network Address Translation*), открывающий доступ в Интернет для тех компьютеров в локальной сети, которым не назначены реальные IP-адреса; если же это неприемлемо, то в файл **/etc/rc.conf** нужно добавить строку **ppp_nat="NO"**.

Изменения в файле **/etc/rc.conf** вступают в силу только после перезагрузки системы.

Постоянное соединение

Чтобы такое соединение не разрывалось системой при простоях, необходимо удалить в файле **/etc/ppp/ppp.conf** строку **set timeout**. Запустить PPP в таком режиме можно из командной строки: **ppp -ddial papchap** или **ppp -ddial provider** для PAP- или CHAP-аутентификации, соответственно. Чтобы автоматически устанавливать такой режим при загрузке системы, файл **/etc/rc.conf** должен содержать две дополнительные опции: **ppp-enable="YES"** и **ppp_mode="ddial"**. Имеется ряд дополнительных опций:

- по умолчанию подразумевается аутентификация при помощи PAP; если реально работает стандартная аутентификация, то в **/etc/rc.conf** нужно добавить строку **ppp_profile="provider"**;
- активизируется NAT, открывающий доступ в Интернет для тех компьютеров в локальной сети, которым не назначены реальные IP-адреса; если это неприемлемо, то в файл **/etc/rc.conf** нужно добавить строку **ppp_nat="NO"**.

Любые изменения в **/etc/rc.conf** вступают в силу только после перезагрузки.

Запуск программ при установлении и разрыве соединения

Пользовательский PPP, как и kernel PPP, способен автоматически выполнять ряд команд в момент установления или разрыва соединения. Эти команды должны быть указаны в файлах **/etc/ppp/ppp.linkup** (для команд, запускаемых при установке соединения) или **/etc/ppp/ppp.linkdown** (при разрыве соединения). Синтаксис этих файлов примерно таков:

```
papchap:
        command_to_run_here
```

Опция **papchap** указывает на тип аутентификации, с которым нужно выполнять указанные команды. Очевидно, что при стандартной аутентификации **papchap** нужно заменить на **provider**.

Данная возможность позволяет автоматизировать процессы получения-доставки почты, она также полезна для пользователей переносных систем (подробнее об этом — в главе 25).

Решение проблем

Если модем не набирает номер:

- проверьте, правильно ли указан последовательный порт;
- в старых системах, попытайтесь вместо стандартной скорости порта (115200) указать скорость 57600 бит/с;
- тщательно проверьте сценарий работы с модемом; чтобы выяснить, какие строки инициализации (помимо стандартных) нужно послать на конкретный модем, возможно, придется обратиться к его документации;

Если модем пытается дозвониться, но не может установить соединение:

- проверьте, какой тип аутентификации используется; совпадает ли он с тем, что указал ваш провайдер;
- проверьте правильность логина и пароля;
- проверьте сценарий **chat** и документацию к модему, возможно, модему требуются дополнительные опции;
- возможно, что при соединении с конкретным провайдером не работает пользовательский PPP, а kernel PPP с теми же настройками работает без проблем;

Если соединение установлено, но попытки открыть любой сайт не увенчались успехом:

- проверьте правильность задания серверов имен в файле **/etc/resolv.conf**; кроме того, в файле **/etc/host.conf** опция **host** должна предшествовать опции **bind**.

Если все вышесказанное не помогло решить проблему, используйте пошаговую отладку процесса установки соединения. Для этого в файл **/etc/ppp/options** нужно добавить строку **debug**, что позволяет регистрировать события, происходящие при запуске kernel PPP, или изменить опцию **set log** в файле **/etc/ppp/ppp.conf** — для записи протокола событий user PPP. Сообщения от kernel PPP протоколируются в файле **/var/log/messages**, а сообщения от user PPP фиксируются в файле **/var/log/ppp.log**. Набрав в окне терминала команду **tail -f /var/log/messages** или **tail -f /var/log/ppp.log**, соответственно, можно наблюдать за процессом установления соединения в интерактивном режиме: при этом отображаются последние 10 строк указанных файлов и изменения, происходящие в них.

Последние замечания

Информации, приведенной в данной главе, вполне достаточно для настройки доступа в Интернет через провайдера, но не более того. Оба типа PPP достаточно сложны и имеют множество опций, которые здесь не рассматривались. Если вам потребуется более подробное описание PPP, обращайтесь к соответствующим страничкам руководства: **man pppd** (для kernel PPP) и **man ppp** (для user PPP). Кроме того, в **usr/share/examples/ppp** имеются примеры некоторых конфигурационных файлов с комментариями.

25 ГЛАВА

Настройка служб E-mail

- Общие сведения об SMTP
- Агенты пересылки почты (MTA) и пользовательские почтовые агенты (MUA)
- Настройка основных служб Sendmail
- Mail Relay
- Протокол POP3
- Настройка POP3-сервера qpopper
- Настройка IMAP-сервера IMAP-UW
- Электронная почта для отдельных рабочих станций
- Альтернативные почтовые серверы

Трудно представить себе Internet без электронной почты. Возможно, даже Web-страницы с их графикой и коммерческими возможностями не приносят столько пользы, как одно из самых старых приложений Internet — электронная почта. Она вездесуща и незаменима. Именно поэтому каждый UNIX-сервер по умолчанию дает возможность работать с электронной почтой, причем не просто посылать и получать почту на пользовательском уровне, а и организовать полноценный почтовый сервер. FreeBSD — не исключение.

Проблема заключается в том, что конфигурирование Sendmail, приложения SMTP-сервера, представляющего собой основу электронной почты в Internet, является одной из самых сложных задач администрирования. Тем не менее данная глава ознакомит читателя с системой Sendmail на уровне, достаточном для настройки защиты и высокой производительности системы.

Кроме того, здесь рассказано и о другом элементе электронной почты: протоколах POP и IMAP, служащих для получения сообщений клиентами. Впрочем, они гораздо проще, чем система Sendmail. Мы, начнем с наиболее сложного протокола.

Общие сведения об SMTP

SMTP — *Simple Mail Transfer Protocol (простой протокол передачи почты)* — один из самых ранних протоколов Internet. Он ровестник самой электронной почты, которая была создана в 1971 году (прошло всего два года с того момента, как была создана первая сеть ARPA). Ее автор — Рэй Томлинсон (Ray Tomlinson). Задачей SMTP была пересылка текстовых сообщений с одного хоста на другой. Схема пересылки почты отличается от работы сети Internet, где посылка отдельных пакетов осуществляется от одного маршрутизатора другому. SMTP-сервер отправителя взаимодействует непосредственно с SMTP-сервером получателя.

Протокол SMTP достаточно прост. Соединение устанавливается на 25-ом порту сервера. Затем следует автоматический код приветствия, несколько необязательных команд аутентификации и команд, устанавливающих тип операции, выполняемой сервером. Затем сервер запрашивает сообщение и получает его. И наконец, команда завершения закрывает соединение. SMTP используется именно для таких операций и большими возможностями не обладает. Для работы по SMTP не требуется даже специального программного обеспечения. Фактически, любую SMTP-операцию можно выполнить прямо из командной строки. Пример посылки сообщения приведен в листинге 25.1. Полужирным шрифтом выделен пользовательский ввод, производимый во время сеанса работы.

Листинг 25.1 Выполнение операции по SMTP

```
# telnet destination.com 25
Trying 64.41.134.166...
Connected to destination.com.
Escape character is '^]'.
220 destination.com ESMTP Sendmail 8.11.1/8.11.1; Wed, 16 May 2001 22:55:37
-0700 (PDT)
HELO stripes.sender.com
250 destination.com Hello w012.z064002043.sjc-ca.dsl.cnc.net [64.2.43.12],
pleased to meet you
MAIL From: frank@sender.com
```

```
250 2.1.0 frank@sender.com... Sender ok
RCPT To: bob@destination.com
250 2.1.5 bob@destination.com... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From: frank@sender.com
To: bob@destination.com
Subject: Testing, 123...

This is a test message.
.
250 2.0.0 f4H5uCu53501 Message accepted for delivery
QUIT
221 2.0.0 destination.com closing connection
Connection closed by foreign host.
```

Как легко заметить, взаимодействие между двумя серверами осуществляется посредством четырехсимвольных команд. По соглашению в командах применяются только буквы верхнего регистра. Аргументы команд включаются в заголовки сообщения. Сообщение, полученное пользователем по адресу **bob@destination.com**, в клиентской почтовой программе будет иметь следующие заголовки:

```
From: frank@sender.com
To: bob@destination.com
Subject: Testing, 123...
```

Заголовки отделяются от тела сообщения одной пустой строкой. Строка, содержащая только точку, обозначает конец тела сообщения. Когда SMTP-программа, получающая сообщение, считывает этот признак, она добавляет сообщение в почтовый файл получателя (**/var/mail/bob**, если он находится на FreeBSD-машине). Это, собственно, и все. Полнофункциональная SMTP-программа (типа Sendmail) пересылает сообщения с помощью нескольких дополнительных команд, повышающих производительность и эффективность.

Клиент взаимодействует с SMTP-сервером своего Internet-провайдера, а не прямо с сервером получателя. Это сделано для того, чтобы воспользоваться преимуществами постановки сообщений в очередь. Такое поведение SMTP-серверов на уровне приложений расширяет функциональность SMTP точно так же, как DNS, не являясь частью стека TCP/IP, делает это для приложений Internet. Постановка сообщений в очередь приводит к тому, что сообщения ожидают момента, когда станет доступным соединение с почтовым сервером получателя. Такая методика освобождает клиента (подключающегося, например, по коммутируемой линии) от необходимости следить за SMTP-сервером получателя, размещать сообщения в очереди и проверять соединение через определенные промежутки времени. Вместо этого клиент устанавливает одно соединение, пересылает все сообщения очереди SMTP-сервера и завершает сеанс. Отправкой сообщений занимается уже SMTP-сервер.

Агенты пересылки почты (MTA) и пользовательские почтовые агенты (MUA)

Путь сообщения от отправителя к адресату схематически показан на рис. 25.1. Независимо от того, подключен ли пользователь по коммутируемой линии или рабо-

тает непосредственно на консоли почтового сервера, путь сообщения одинаков: введенные данные поступают к пользовательскому почтовому агенту (Mail User Agent) (он же почтовый клиент), затем сообщение размещается в очереди SMTP-сервера. Как только устанавливается соединение с SMTP-сервером адресата, агент пересылки почты (Mail Transfer Agent, MTA) (например, Sendmail) извлекает сообщение из очереди и пересылает его. Поскольку данный процесс пересылает сообщение, ни источником, ни адресатом которого сам SMTP-сервер не является, этот этап называется *relaying (пересылка)*. На удаленной системе MTA помещает это сообщение в почтовый ящик адресата. Его можно прочесть с помощью почтового клиента (MUA) при работе на консоли или загрузить по протоколу POP3 (при подключении, например, по коммутируемой линии).

Как же Sendmail работает в роли отправителя и получателя?

Рисунок 25.1
Диаграмма пути, который проходит сообщение от одного пользователя к другому. Показано назначение MUA и MTA.

Распространенные агенты пересылки почты (MTA)

Вот несколько программ MTA, используемых в Internet:

- **Sendmail**. На сегодняшний день, самый распространенный MTA. Программа разработана Эриком Олманом (Eric Allman) в 1983 году и стала стандартом дефакто для различных платформ. Имеет и коммерческие и свободно распространяемые компоненты. Компания Sendmail, Inc. (**http://www.sendmail.com**) — полноценная корпорация, которая занимается поддержкой Sendmail для коммерческих версий UNIX и Linux. Проект Sendmail (**http://www.sendmail.org**) распространяет бесплатную версию этой программы, поддержка которой осуществляется на добровольной основе. Ресурсы доступны в открытом коде, традиционном для таких систем, как FreeBSD (Sendmail включен в нее по умолчанию)
- **Microsoft Exchange**. Эквивалент Sendmail для системы Windows, поддерживающий несколько типов пересылки сообщений (POP3, NNTP, LDAP), Exchange разработан на коммерческой основе и по традиции Microsoft распространяется в закрытом коде.

- **Postfix**. Эта программа, написанная Вьетсом Венемой (Wietse Venema) как альтернатива Sendmail, разработана с тем, чтобы стать более быстрой, более защищенной и более простой в работе, чем Sendmail.
- **Qmail**. Программа Qmail, еще одна альтернатива Sendmail, была написана Дэном Бернстайном (Dan Bernstein). Она быстро становится популярной, благодаря преимуществам в вопросах защиты и скорости. В отличие от агента Postfix (который имитирует взаимодействие Sendmail с системой, например, в размещении спула и конфигурационных файлов), Qmail использует собственную структуру и не пытается облегчить переход к ней Sendmail — пользователям.

  MTA, которые портированы во FreeBSD, можно найти в каталоге **/usr/ports/mail**.

Распространенные пользовательские почтовые агенты (MUA)

Пользовательских почтовых агентов (Mail User Agents) значительно больше, чем MTA (например, Microsoft Outlook/Outlook Express, Eudora, Apple Mail, Netscape Mail и т.д.). Ниже приведен список наиболее популярных MUA, включенных в состав FreeBSD.

- **Pine**. Официальный акроним от "Program for Internet News and E-mail". Программа разработана в Вашингтонском университете. Pine довольно широко используется, обеспечивая интуитивный интерфейс, множество современных функций обработки сообщений, составитель сообщений (message composer) на базе редактора **pico** (поставляемый в пакете с Pine).
- **Elm**. Программа, послужившая основой для Pine. Elm эволюционировала от древних почтовых программ **mail** и **mailx**, которые применялись для работы с электронной почтой в университетских и правительственных сетях, предшествовавших Internet. Примитивный по стандартам сегодняшнего дня клиент Elm все еще достаточно распространен.
- **Mutt**. Представляет собой гибрид различных MUA и включает свойства Elm, Pine и других программ. Mutt считается лучшим среди них. Он имеет функции, отсутствующие у многих клиентов, включая и те, что требуют графических MUA на клиентской стороне.

Обратите внимание, что перечисленные здесь почтовые агенты ориентированы на работу с командным интерпретатором, их можно запустить во время сеанса Telnet или SSH. Это значит, что эти клиенты являются текстовыми и не поддерживают изображения или HTML-форматирование (хотя их можно настроить таким образом, чтобы открывать подобные вложения во вспомогательных приложениях). Но главное достоинство состоит в том, что с помощью этих программ пользователь может проверить почту независимо от того, где он находится, — достаточно открыть сеанс Telnet или SSH. И еще! Ваша система никогда не пострадает от заполонивших Internet вирусов, предназначенных для Windows/Outlook.

Настройка основных служб Sendmail

В состав FreeBSD включена программа Sendmail, причем она уже сконфигурирована и готова к работе. Достаточно лишь активировать Sendmail (чтобы агент запускался во время загрузки), добавив в файл **/etc/rc.conf** следующую строку:

```
Sendmail_enable="YES"
```

После того как Sendmail запущена, можно посылать и получать сообщения от любого пользователя Internet. Для этого необходимо правильно настроить несколько элементов. В конфигурации по умолчанию Sendmail подходит для большинства серверов, поэтому мы в первую очередь рассмотрим, как эффективно управлять системой Sendmail.

Схема размещения файлов Sendmail

С работой Sendmail связаны три точки файловой системы:

- **/etc/mail**: Конфигурационные файлы Sendmail.
- **/var/mail**: Почтовые ящики пользователей.
- **/var/spool/mqueue**: Файлы очереди сообщений.

Выполняемый файл Sendmail расположен в **/usr/sbin/sendmail**, а log-файлы хранятся в **/var/log/maillog** (их обновление происходит ежедневно, автоматизированный запуск заданий обсуждается в главе 14).

ПРИМЕЧАНИЕ

В некоторых системах по историческим причинам выполняемый файл Sendmail размещен в **/usr/lib/sendmail**. Если вам приходится запускать программы, которым требуется именно такой путь, можно несколько нарушить структуру каталогов FreeBSD и создать символическую ссылку:

```
# ln -s /usr/sbin/sendmail /usr/lib/sendmail
```

Файлы почтовых ящиков (т.е. почтовый спул) в каталоге **/var/mail** представляют собой обычные текстовые файлы. Имя каждого из них совпадает с именем пользователя-владельца, а права доступа установлены как 600 (доступен для чтения и записи только владельцу). Новые сообщения добавляется к файлу почтового спула адресата. Кроме того, существуют специальные lock-файлы POP, имеющие нулевую длину и имя **.username.pop**. Они получают содержимое соответствующего файла почтового спула, пока открыто POP3-соединение. О работе POP3-сервера рассказано далее в этой главе.

Конфигурационные файлы

Ниже перечислены файлы из каталога **/etc/mail**, необходимые при работе Sendmail.

/etc/mail/sendmail.cf

Это основной конфигурационный файл Sendmail. Однако, в отличие от подобных файлов других программ, он не предназначен для редактирования. Изменения вносятся в основной конфигурационный файл — Master Config (**.mc**), из которого после компиляции получается **sendmail.cf**.

Файл **sendmail.cf** содержит опции, наборы правил и свойства. Его формат сложен и неудобен для чтения. Поэтому пытаться изменять что-то в **sendmail.cf** можно лишь в случае крайней необходимости.

/etc/mail/freebsd.mc

Это вышеупомянутый файл Master Config. Он содержит список свойств и опций, которые отменяют действие настроек по умолчанию в стандартном конфигурационном файле точно так, как **/etc/rc.conf** отменяет действие **/etc/defaults/rc.conf**. В данном случае формат опций гораздо сложнее. Команды заданы в формате макроязыка **m4** (за пределами конфигураций Sendmail используется редко). Файл **.mc** компилируется вместе с файлом **cf.m4**, размещенным в каталоге с исходным кодом Sendmail (в **/usr/share/sendmail** или **/usr/src/contrib/sendmail**). В результате компиляции создается конфигурационный файл **.cf**, который можно затем установить как новый файл **sendmail.cf**.

Раньше, чтобы создать файл **sendmail.cf**, необходимо было перейти в каталог исходного кода, определить, какой **.mc** лучше всего подходит для данной системы, внести изменения согласно документации, затем, пользуясь сложными командами компиляции языка **m4**, создать требуемый файл и лишь потом вручную установить его. На сегодняшний день ситуация заметно улучшилась, но интерфейс конфигурирования все еще нельзя назвать дружественным.

Каждая строка в языке **m4** содержит в некоторой позиции подстроку **dnl**. Последняя означает "delete through newline" — "удалить до конца строки". **dnl** отмечает конец строки (поэтому эта команда должна присутствовать в каждой строке). Чтобы закомментировать строку, следует расположить **dnl** в ее начале, а чтобы раскомментировать — в конце.

В каталоге **/etc/mail** имеется Makefile, который позволяет создать новый **.cf**-файл из **freebsd.mc** с помощью команды **make cf**. Чтобы установить выходной файл (**freebsd.cf**) как новый вариант **sendmail.cf**, достаточно воспользоваться командой **make install**.

```
# make cf
/usr/bin/m4 -D_CF_DIR_=/usr/share/sendmail/cf/ /usr/share/sendmail/cf/m4/
cf.m4   freebsd.mc > freebsd.cf
# make install
install -c -m 444 freebsd.cf /etc/mail/sendmail.cf
```

Файл **/etc/mail/Makefile** предназначен еще для нескольких целей, о которых рассказано далее.

/etc/mail/aliases

В каталоге **/var/mail** существует почтовый ящик каждого пользователя системы. Кроме того, можно воспользоваться псевдонимами, перенаправляющими входящую электронную почту на другие адреса (на локальной машине или где-либо в Internet) или в файлы или программы (с помощью конвейера). Примеры таких настроек по умолчанию содержатся в файле **/etc/mail/aliases**. Строка с псевдонимом содержит имя, двоеточие, пробел или символ табуляции и адрес назначения или конвейер:

```
tiger: bob@stripes.com
fsmith: frank
```

```
pager: "|/usr/local/bin/pageme"
dump: ">>/home/frank/dump2me"
mylist:include:/home/frank/list.txt
```

Это пример подключения внешнего файла псевдонимов. Список адресов читается из обычного текстового файла. Это простой способ поддержки списка рассылки: достаточно внести изменения в файл, и таблица псевдонимов всегда будет актуальна.

После внесения изменений в файл **/etc/mail/aliases**, необходимо заново собрать файл **aliases.db**, в котором хранится ассоциативная таблица псевдонимов для быстрого поиска (это похоже на файл **/etc/master.passwd**, о котором говорилось в главе 10). Для создания этого файла можно воспользоваться традиционной командой **newaliases** или командой **make aliases**:

```
# make aliases
/usr/sbin/sendmail -bi
/etc/mail/aliases: 22 aliases, longest 10 bytes, 213 bytes total
```

ПРИМЕЧАНИЕ

Глобальные псевдонимы — не единственный способ перенаправить почту с локального адреса на другой локальный или внешний адрес. Например, если пользователю требуется, чтобы вся входящая почта автоматически пересылалась на какие-либо внешние адреса, можно воспользоваться настройками в файле **/etc/mail/aliases**, однако это требует прав доступа **root**. Существует более удобный способ, если пользователь имеет полноценную учетную запись в системе.

Чтобы пересылать почту на другой адрес, достаточно создать в home-каталоге пользователя файл **.forward**, содержащий требуемый адрес. Выполнить это можно с помощью любого текстового редактора или просто команды **echo**:

```
# echo "frank@somewhereelse.com" > .forward
```

Чтобы остановить пересылку почты, следует удалить этот файл.

/etc/mail/access

База данных доступа **access** позволяет указать правила, применяемые к отдельным хостам, подсетям или группам адресов. Это отличная предосторожность против нежелательной рекламы. Допустимые правила включают **OK**, **REJECT**, **RELAY**, **DISCARD** или **550 <message>**. Содержимое файла **/etc/mail/access** по умолчанию показывает, как форматируется поле адреса/имени хоста:

```
cyberspammer.com          550 We don't accept mail from spammers
FREE.STEALTH.MAILER@      550 We don't accept mail from spammers
another.source.of.spam    REJECT
okay.cyberspammer.com     OK
128.32                    RELAY
64.2.43                   RELAY
RELAY
```

- Правило **OK** принимает сообщения от заданного хоста даже тогда, когда он не удовлетворяет другим проверкам, о которых рассказано далее.
- Правило **REJECT** отвергает соединения, устанавливаемые с указанного хоста.
- Правило **DISCARD** отбрасывает сообщения после их принятия, что позволяет отправителю думать, что все сообщения были успешно доставлены адресату.
- Правило **RELAY** включает пересылку сообщений указанному хосту, отменяя другие проверки (подобно правилу **OK**).

- Правило **550 <message>** задает сообщение "отказа", которое отображается отправителю, принадлежащему заданному хосту. Оно появляется во время SMTP-сеанса и включается в почтовое сообщение об ошибке, отсылаемое пользователю.

Когда в файл **/etc/mail/access** внесены изменения, необходимо заново сгенерировать **access.db**. Для этого используется команда **make maps**, которая обновляет файлы карт, измененные с момента предыдущего запуска **make maps**. Чтобы перезапустить главный процесс Sendmail с новой версией файла **access.db**, воспользуйтесь командой **make restart**:

```
# make maps
/usr/sbin/makemap hash access.db < access
# make restart
/bin/kill -HUP `head -1 /var/run/sendmail.pid`
```

/etc/mail/local-host-names

В этом файле (в прошлом он назывался **sendmail.cw**) указаны имена хостов, принадлежащих данному серверу. Это особенно важно, когда на одной машине размещено несколько доменов, для каждого из которых обеспечивается почтовая служба (виртуальный хостинг). Если не добавить доменное имя в файл **local-host-names file**, входящее сообщение вернется к отправителю с устрашающим сообщением: **MX list for <domain> loops back to myself; local configuration error message** (Список MX для домена <domain> замкнут на себя; ошибка локальной конфигурации).

В инсталляции по умолчанию этот файл отсутствует. Его нужно создать, если машина имеет несколько доменных или хостовых имен в DNS. Добавив имена хостов в этот файл, необходимо перезапустить сервер командой **make restart**.

```
# make restart
/bin/kill -HUP `head -1 /var/run/sendmail.pid`
```

/etc/mail/virtusertable

Файл **/etc/mail/virtusertable** также используется при виртуальном хостинге. Он позволяет поставить в соответствие почтовым адресам локальные учетные записи, другие адреса или сообщения об ошибках. Данный файл представляет собой гибрид базы данных доступа и файла псевдонимов.

Предположим, что на машине размещен второй домен, **mycave.org**. Требуется, чтобы письма на адрес **webmaster@mycave.org** пересылались локальному пользователю **bill**, на **info@mycave.org** — на удаленный адрес **anne@elsewhere.com**, а письма на все остальные адреса **@mycave.org** отвергались. Для решения этой задачи служит следующий набор правил (символом-разделителем служит табуляция):

```
webmaster@mycave.org        bill
info@mycave.org             anne@elsewhere.com
@mycave.org                 error:nouser User unknown
```

Порядок, в котором определены правила, не имеет значения. При создании ассоциативной таблицы **virtusertable.db** каждое правило имеет собственное значение для поиска. Как и для базы данных доступа, файл **virtusertable.db** необходимо обновить, а затем перезапустить сервер:

```
# make maps
/usr/sbin/makemap hash virtusertable.db < virtusertable
# make restart
/bin/kill -HUP `head -1 /var/run/sendmail.pid`
```

Подробное обсуждение виртуального хостинга и использования файла **virtusertable** можно найти на Web-сайте консорциума Sendmail по адресу **http://www.sendmail.org/virtual-hosting.html**.

ПРИМЕЧАНИЕ

Файл **/etc/mail/Makefile**, необходимый для сборки различных конфигурационных баз данных, добавлен во FreeBSD. Он не включен в версию, распространяемую разработчиками Sendmail. Поэтому другие операционные системы не обладают подобными удобствами. Чтобы поддерживать Sendmail в Linux или на другой платформе, необходимо знать все команды, содержащиеся в Makefile и выполнять их вручную.

Вопросы разрешения DNS-адресов

Для корректной работы сервера Sendmail необходимо, чтобы в таблицах DNS содержалась точная информация о машине, на которой он запущен. Если вы установили FreeBSD, присвоили машине произвольный IP-адрес и пытаетесь отправить сообщение, оно может вернуться назад с ошибкой: сервер не может правильно выполнить разрешение имени хоста. Многие SMTP-серверы, включая и Sendmail, сконфигурированные по умолчанию, не принимают сообщения от адресатов, не имеющих корректных доменных имен. Прежде чем запускать Sendmail, необходимо убедиться, что для данной машины правильно настроено обратное DNS-преобразование имен. Для этого следует запустить команду **nslookup** с IP-адресом в аргументе:

```
# nslookup stripes.somewhere.com
Server:   lion.somewhere.com
Address:   64.41.131.132

Name:      stripes.somewhere.com
Address:   64.41.131.102
```

Здесь приведен пример корректной настройки обратного DNS-преобразования. В этом случае система Sendmail сможет работать. Если же вывод гласит следующее:

```
# nslookup stripes.somewhere.com
Server:   lion.somewhere.com
Address:   64.41.131.132

*** lion.somewhere.com can't find stripes.somewhere.com: Non-existent host/
domain
```

это значит, что с доставкой почты (и отправлением, и получением) возможны проблемы, пока не будет настроено корректно обратное преобразование имен. Отметьте, что проверку необходимо осуществлять на удаленном DNS-сервере, поскольку локальный DNS-сервер иногда выводит информацию, известную только ему и не распространяющуюся на остальную часть Internet. Вопросы настройки DNS обсуждаются в главе 30.

Управление системой Sendmail

Сервер Sendmail работает следующим образом: запускается "главный" процесс, который прослушивает порт 25, ожидая входящих соединений, а также дополнитель-

ные процессы для обработки очередей, посылки сообщений удаленным адресатам и т.д. Главный процесс запускается при загрузке из **/etc/rc**. Его запуск и останов достаточно просты благодаря файлу **Makefile** и конфигурационным файлам ресурсов в каталоге **/etc**. Чтобы начать выполнение процесса, достаточно воспользоваться командой **make start**:

```
# make start
(. /etc/defaults/rc.conf; source_rc_confs;  if [ "${ sendmail_enable} " =
"YES" -a -r /etc/mail/sendmail.cf ]; then   /usr/sbin/sendmail ${
sendmail_flags} ;  fi  )
```

Поскольку команда отображает на экране информацию о своих действиях, видно, что она читает часть сведений из конфигурационных файлов ресурсов системного уровня, где указаны такие флажки, как **-q30m** (производить запуск очереди каждые тридцать минут) и **-bd** (запуск демона в фоновом режиме). Sendmail не запустится, если переменная **sendmail_enable** в файле **rc.conf** установлена значением **NO**.

Перезапуск и останов главного процесса также не вызывают сложностей:

```
# make restart
/bin/kill -HUP `head -1 /var/run/sendmail.pid`
# make stop
/bin/kill -TERM `head -1 /var/run/sendmail.pid`
```

Просмотреть состояние каждого процесса Sendmail можно с помощью команд **ps** и **grep**. Каждый процесс сообщает свое положение в очереди как аргумент напротив имени в таблице процессов. В следующем примере приведен главный процесс (**51248**) и процесс в середине очереди (**54150**):

```
51248   ??   Ss      0:00.17 sendmail: accepting connections (sendmail)
54150   ??   I       0:00.02 sendmail: ./f4GKwVW16827
mail.backstreetboys.com.: user open (sendmail)
```

Очередь сообщений

Сообщения, ожидающие отправки, находятся в каталоге **/var/spool/mqueue**. В конфигурации по умолчанию каждые 30 минут запускается новый процесс **sendmail -q**, который пытается доставить каждое сообщение из очереди. Так продолжается в течение пяти дней, после чего те сообщения, которые не удалось отправить адресатам, возвращаются отправителю с сообщением об ошибке.

Если в очереди имеются сообщения, их можно при желании просмотреть. В отличие от таких закрытых систем, как Microsoft Exchange, где файлы очередей хранятся в базе данных, не позволяющей просмотреть простыми средствами, какие файлы ожидают отправки, Sendmail имеет прозрачный интерфейс для доступа к ним. Все поставленные в очередь сообщения представляют собой обычные текстовые файлы, которые можно читать или редактировать с помощью стандартных текстовых редакторов. Это не только дает администратору полный контроль над работой почтовой системы, но и предоставляет ему некоторую возможность для злоупотребления своими полномочиями, поскольку **root** имеет возможность прочесть содержимое сообщений, ждущих отправки.

Одна из команд, включенных в Sendmail, **mailq**, выводит список сообщений, находящихся в очереди на текущий момент.

```
# mailq
                /var/spool/mqueue (2 requests)
----Q-ID---- --Size-- -----Q-Time----- -----------Sender/Recipient----------
f4H1Ahu36976    6246 Wed May 16 18:10 MAILER-DAEMON
                (Deferred: Operation timed out with mlists.acmecity.com.)
                                                <fred@acmecity.com>
f4GKwVW16827    706 Wed May 16 13:58 www
                (host map: lookup (hotamil.com): deferred)
                                               Bob bob@hotamil.com
```

Применяя **mailq**, можно следить за тем, какие виды ошибок чаще всего происходят при доставке почты. Короче говоря, это очень удобное средство диагностики. Кроме того, очередь позволяет исправить ошибки в сообщениях, готовящихся к отправке. Предположим, например, что в очереди присутствует сообщение, приведенное в примере вывода **mailq** ранее. В имени домена содержится опечатка. Можно подождать пять дней, пока Sendmail не оставит попытки отправить сообщение по адресу на хосте **hotamil.com** и не возвратит его, или исправить ошибку прямо в очереди.

Для этого перейдите в каталог **/var/spool/mqueue** и найдите файлы, идентификаторы которых совпадают с записями в выводе **mailq**. Это будут файлы **df4GKwVW16827** и **qf4GKwVW16827**. Первый содержит тело сообщения, а второй — его заголовки в промежуточном формате. Откройте файл заголовков (**qf4GKwVW16827**) в текстовом редакторе, замените все вхождения строки **hotamil.com** на **hotmail.com**, сохраните его и дождитесь следующего запуска очереди. Сообщение уйдет. Кстати, сообщения удаляются из очереди только после успешной доставки.

Запустите команду **sendmail -q -v**. Такой формат команды позволяет просмотреть, как Sendmail производит все операции по SMTP с удаленными системами. При обработке каждого сообщения Sendmail выводит на экран все сведения об операции, как это было показано в начале этой главы. Здесь можно увидеть все приветственные сообщения, которые администраторы программируют в MTA (их видят только почтовые агенты). Завершить работу программы можно по Ctrl+C в любой момент.

Mail Relay

Одно из последних усовершенствований Sendmail — это защита от спама (т.е. сообщений рекламного характера) посредством специальных правил против пересылки. Эти правила были доступны уже давно, но лишь в недавней версии Sendmail 8.9 пересылка сообщений от одного сервера к другому (relaying) была запрещена в конфигурации по умолчанию.

Для пользователя, подключающегося, например, по коммутируемой линии, SMTP-сервер пересылает сообщение удаленному адресату даже в том случае, когда сообщение не принадлежит и не предназначается никому на данной машине. Как видно из рис. 25.2, именно наличие такой функциональности и используется спаммерами для посылки сообщений тем же адресатам через тот же SMTP-сервер: он передает почтовое сообщение, как эстафетную палочку.

Пересылка (relaying) обычно разрешатся с помощью записи MX (MX record) — строки SMTP-сервера в базе данных DNS (находящейся или на одной машине с SMTP-сервером или на другом сервере в той же сети), которая говорит всем маши-

нам в сети, что SMTP-сервер (S1) является легальной системой обмена почтой (Mail eXchanger). Sendmail в конфигурации по умолчанию принимает почту только от тех адресатов, которые принадлежат локальному домену. Таким образом, пользователи за пределами сети не могут использовать S1 для пересылки. При попытке такого доступа, все сообщения возвращаются отправителям с ошибкой **Relaying denied error** (Эстафетная пересылка запрещена).

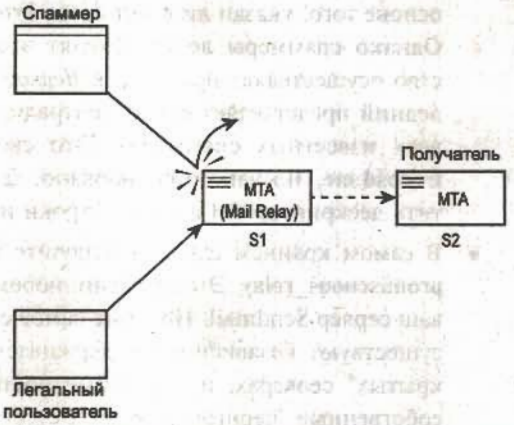


Рисунок 25.2
Работа сервера Mail Relay. Спаммеры и легальные пользователи, если они не являются локальными по отношению к S1, должны использовать S1 для пересылки сообщений серверу S2.

ПРИМЕЧАНИЕ

Определить, какой хост зарегистрирован в качестве Relay-сервера для определенного домена позволяет команда **host**:

```
# host somecompany.com
somecompany.com has address 164.199.3.78
somecompany.com mail is handled (pri=30) by mail-1.somecompany.com
```

После этого можно соединиться прямо с этим хостом и выполнить набор команд SMTP.

Однако есть проблема. Описанная методика подходит для Internet-провайдеров или корпоративных сетей, где записи DNS определены для всех хостов, а соответствующие записи MX указывают на SMTP Relay-серверы. Но как быть с отдельными хостами Internet, пользователи которых пытаются переслать почту? Каждый из таких пользователей при попытке послать сообщение через S1 столкнется с ошибкой **Relaying denied error**, если только S1 не настроен специальным образом для эстафетной пересылки. Существует несколько способов, однако использовать их не рекомендуется.

- Добавьте домены "доверенных" отправителей в файл **/etc/mail/relay-domains**, который отсутствует в конфигурации по умолчанию. Всем хостам в пределах указанного домена будет разрешено использовать ваш сервер для пересылки почты. После изменения файла необходимо перезапустить Sendmail. Такая методика проста и эффективна, но, если вы добавите достаточно большой домен, где, кроме легальных пользователей, обитают и спаммеры, все ее достоинства утратятся.
- Используйте базу данных доступа (**/etc/mail/access**). В ней можно указать правило **OK** или **RELAY** для каждого известного хоста или домена, с которого будет приниматься почта. Это хорошо работает для небольших сетей или не-

скольких удаленных хостов с легко определяемыми адресами. Для большого числа пользователей, имеющих динамические адреса, такая методика не подходит.

- Включите несколько из пяти или шести свойств пересылки, поддерживаемых Sendmail, в файле **/etc/mail/freebsd.mc**, а затем заново сгенерируйте **sendmail.cf** (как было показано ранее). Одно из свойств позволяет включить пересылку на основе того, указан ли в заголовке **From:** адрес вашего домена (**relay_local_from**). Однако спаммеры легко обходят это свойство. Другое дополнительное свойство осуществляет проверку в *Черном списке (Realtime Blackhole List, RBL)*. Последний представляет собой централизованную базу данных с информацией обо всех известных спаммерах. Это свойство по умолчанию включено в файл **freebsd.mc**, но закомментировано. Для его разрешения необходимо переместить дескриптор **dnl** в конец строки и заново собрать конфигурационный файл.
- В самом крайнем случае отключите проверку пересылки с помощью свойства **promiscuous_relay**. Это позволит любому пользователю посылать сообщения через ваш сервер Sendmail. Но то же самое сможет сделать и любой спаммер. В Internet существуют независимо поддерживаемые базы данных с записями о таких "открытых" серверах, и некоторые поставщики услуг Internet используют их как собственные "черные списки". Естественно, вам совсем не нужно, чтобы в них включили и ваш сервер! Если это произойдет, легальная исходящая почта также может оказаться блокированной.

Можно рекомендовать следующее общее правило: желательно, чтобы все ваши пользователи применяли SMTP-серверы, поддерживаемые их собственными ISP-провайдерами. Последние всегда имеют службы SMTP, открытые для клиентов. Поскольку заголовки (как, например, **From:**) находятся в теле сообщения и полностью контролируются почтовой программой-клиентом, ваш SMTP-сервер пользователям больше не потребуется.

Правила пересылки сообщений и доступные опции конфигурации можно найти по адресу **http://www.sendmail.org/tips/relaying.html**.

Протокол POP3

Мы изучили протокол SMTP и систему Sendmail, а также средства, необходимые для ее конфигурирования. Но SMTP — это лишь половина задачи. Чтобы сообщения могли быть доставлены от одного пользователя другому, необходим еще один процесс: загрузка почты с сервера по протоколу POP.

Необходимость в почтовом протоколе (Post Office Protocol) стала очевидной тогда, когда стало ясно, что в Internet будет работать множество пользователей, применяющих на своих компьютерах клиентские почтовые программы с графическим интерфейсом, а не текстовые клиенты, такие как Pine и Mutt, которые читают сообщения прямо из файла почтового ящика пользователя. Для использования таких MUA, как Microsoft Outlook или Eudora компании Qualcomm, клиенту необходимо подключиться к почтовому серверу, где хранится файл с его сообщениями (см. рис. 25.1),

определить, есть ли в нем новые сообщения и, если да, загрузить их. После этого программа-клиент отображает новые сообщения и удаляет их из файла на сервере.

В отличие от SMTP, POP3 (текущая версия POP, стандарт) требует аутентификации. Хотя защита не так важна при отправке почты, она становится весьма актуальной при ее получении. Действительно, кто угодно может бросить письмо в почтовый ящик, но только авторизованный адресат может получить его. POP3 почти не требует конфигурирования. Здесь мало элементов, которые могут быть настроены некорректно. Однако POP3-сервер, **qpopper**, не включен в основную инсталляцию системы FreeBSD. Для использования служб POP3 необходимо установить **qpopper** из набора портов (**/usr/ports/mail/qpopper**). Эта программа происходит от Berkeley POP-сервера, но на данный момент разрабатывается на коммерческой основе (однако свободно распространяется и в открытом коде) компанией Qualcomm, производящей почтовый клиент Eudora.

POP3-сервер запускается из суперсервера **inetd**. Последний прослушивает соединения по TCP и UDP и при открытии TCP-соединения на порту 110 ищет название службы в файле **/etc/services**, определяя, как обработать запрос. Затем он запускает процесс **qpopper** из **/usr/local/libexec/qpopper**. (Местоположение выполняемого файла в каталоге **libexec** показывает, что эта программа предназначена не для выполнения из командной строки, а для запуска другими программами.) Процесс **qpopper** обрабатывает операцию, аутентифицирует пользователя, открывает его почтовый ящик, определяет, какие сообщения нужно загрузить, и обслуживает их. Такая методика пригодна для большинства систем. Хотя **qpopper**, в действительности, имеет большой набор конфигурационных опций, большинство из них предназначено для повышения производительности сервера. Во многих случаях достаточно лишь инсталлировать порт и разрешить службу.

Важно отметить, что все операции в протоколе POP3 по умолчанию осуществляются в обычном тексте, а это приводит к потенциально возможному перехвату паролей и дырам в зашите. В версии 4.0 **qpopper** позволяет производить зашифрованные соединения с помощью библиотек SSL (Secure Sockets Layer), которые являются частью FreeBSD и используются в таких протоколах, как SSH (Secure Shell) и secure HTTP. Далее мы покажем, как можно воспользоваться этим достоинством **qpopper** для улучшения защиты системы.

Настройка POP3-сервера qpopper

Существует несколько уровней конфигурации POP3-сервера. Можно просто инсталлировать сервер и позволить всем использовать его; или запустить его в самостоятельном режиме, когда не требуется запуск демона **inetd**; или запустить его в "серверном режиме", улучшающем производительность; можно использовать зашифрованные сеансы посредством TLS/SSL. Эти опции не являются взаимоисключающими. Каждую из них мы рассмотрим далее.

Инсталляция и конфигурирование qpopper

Запустить службу POP3, не заботясь о производительности или защите, достаточно просто. Перейдите в каталог **qpopper** набора портов (**/usr/ports/qpopper**), соберите

приложение и установите его (см. главу 15). Можно установить пакет **qpopper** с помощью программы **sysinstall** или средств для работы с пакетами. После этого включите службу POP3 в суперсервере **inetd**. Для это добавьте строку в файл **/etc/inetd.conf**.

Откройте файл **inetd.conf** и найдите приведенные ниже строки, связанные с POP3. Добавьте строку или уберите комментарий с **pop3**, применив следующий синтаксис:

```
# пример записи для сервера pop3
#
#pop3 stream   tcp     nowait   root     /usr/local/libexec/popper     popper
pop3  stream   tcp     nowait   root     /usr/local/libexec/qpopper    qpopper -s
```

Перезапустите демон **inetd**, воспользовавшись командой **killall**:

```
# killall -HUP inetd
```

Чтобы проверить, доступна ли служба POP3, необходимо подключиться к порту 110 по Telnet. Для окончания сеанса используется команда **QUIT**.

```
# telnet localhost 110
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+OK Qpopper (version 4.0.2) at stripes.somewhere.com starting.
<4763.990313780@stripes.somewhere.com>
QUIT
+OK Pop server at stripes.somewhere.com signing off.
Connection closed by foreign host.
```

Если вы получили от сервера подобный ответ, значит, POP3-сервер работает правильно.

Запуск в автономном режиме (Standalone Mode)

При определенных настройках защиты суперсервер **inetd** не используется. Например, если при инсталляции системы выбирается уровень защиты Medium или High, в файле **/etc/rc.conf** добавляется строка, препятствующая запуску **inetd**:

```
inetd_enable="NO"
```

В защищенной таким образом системе включать **inetd** ради использования служб POP3 не следует — для этого существует режим standalone. Он недоступен по умолчанию, но его легко включить, внеся изменения в файл **Makefile**.

В каталоге **/usr/ports/mail/qpopper**, отредактируйте файл **Makefile**, добавив опцию **--enable-standalone** в строке **CONFIGURE_ARGS**:

```
CONFIGURE_ARGS=  --enable-apop=${PREFIX}/etc/qpopper/pop.auth \
                 --enable-nonauth-file=/etc/ftpusers \
                 --with-apopuid=pop --without-gdbm \
                 --enable-keep-temp-drop \
                 --enable-standalone
```

Затем выполните команды **make** и **make install** для сборки версии **qpopper**, способной запускаться в самостоятельном режиме. Она также устанавливается в каталог **/usr/local/libexec**. Перемещать ее не рекомендуется, поскольку это затрудняет дальнейшую деинсталляцию.

Отдельный процесс **qpopper** необходимо запустить из сценария при загрузке системы. Для этого создайте в каталоге **/usr/local/etc/rc.d** сценарий под названием **qpopper.sh** (см. главу 11). Убедитесь при этом, что служба **qpopper** отключена в **inetd**!

Запуск в режиме сервера (Server Mode)

Если пользователи системы обращаются к своим почтовым ящикам только по POP3 и не имеют доступа к командной строке или почтовым программам сервера, **qpopper** можно запустить в серверном режиме. Этот режим позволяет работать с почтовым ящиком напрямую. Такая методика более опасна, особенно, если пользователи могут обращаться к почтовому ящику и другими методами. Поэтому такой режим предназначен для систем, где, во-первых, доступ к почте возможен только по POP3 и, во-вторых, важную роль играет производительность.

Запустить **qpopper** в режиме сервера можно несколькими способами. Наиболее простой — воспользоваться опцией **-S** в файле **inetd.conf** или в командной строке, если сервер работает в самостоятельном режиме:

```
pop3    stream   tcp      nowait   root      /usr/local/libexec/qpopper
qpopper -s -S
```

В результате **qpopper** будет выполняться в режиме сервера для всех пользователей. Если требуется, такие настройки можно произвести для отдельного пользователя или группы. Управление группами лучше всего осуществлять посредством конфигурационного файла. Чтобы **qpopper** использовал такой файл, его необходимо создать в каталоге **/usr/local/etc/qpopper** (например, **qpopper.conf**), а затем указать в командной строке в **inetd.conf** с помощью опции **-f**:

```
pop3     stream   tcp      nowait   root      /usr/local/libexec/qpopper
qpopper -s -S -f /usr/local/etc/qpopper/qpopper.conf
```

В файле **qpopper.conf** нужно задать имя группы, для которой включен или отключен режим сервера, воспользовавшись ключевым словом **group-server-mode** или **group-no-server-mode**. Вот пример включения режима сервера для групп group1 и group2 и отключения для группы group3:

```
set group-server-mode=group1
set group-server-mode=group2
set group-no-server-mode=group3
```

Эта настройки можно произвести и для отдельных пользователей. Для этого в командной строке следует применить опцию **-u** (которая разрешает чтение конфигурационных файлов отдельных пользователей). После этого каждый пользователь, для которого устанавливается режим сервера, должен иметь в своем каталоге файл **.qpopper.options**, содержащий строку **set server-mode**. Если вы не хотите, чтобы пользователи могли изменять свои файлы **.qpopper.options**, воспользуйтесь опцией **-U** вместо **-u** и поместите соответствующие файлы в каталог **/var/mail** под именем **.<имя_пользователя>.qpopper.options**. Таким способом режим сервера можно применять индивидуально. Использовать опцию **-S**, задающую глобальный режим сервера, в этом случае не следует!

Включение шифрования по SSL

Поддержка TLS/SSL встроена в **qpopper** по умолчанию. Чтобы воспользоваться ей, необходимо создать и установить сертификаты защиты (security certificates). Вначале нужно создать каталог для них:

```
# mkdir -p -m665 /etc/mail/certs
# chown root:mail /etc/mail/certs
# chmod 660 /etc/mail/certs
```

Затем для генерирования запроса сертификата следует воспользоваться программой **openssl**. При этом нужно ввести корректную информацию о своей организации. Затем нужно установить право на чтение файла с личным ключом (**cert.pem**) только для пользователя **root**.

```
# openssl req -new -nodes -out req.pem -keyout /etc/mail/certs/cert.pem
...
# chmod 600 /etc/mail/certs/cert.pem
# chown root:0 /etc/mail/certs/cert.pem
```

После этого сертификат нужно зарегистрировать у уполномоченных представителей сертификации (Certifying Authority, CA), например, в VeriSign. Отправьте CA запрос на сертификацию в файле **eq.pem**, и вы получите подписанный сертификат (signed certificate). Конкатенируйте его к файлу **cert.pem**:

```
# cat signed_req.pem >> /etc/mail/certs/cert.pem
```

Теперь добавьте поддержку TLS/SSL в конфигурационном файле **/usr/local/etc/qpopper/qpopper.conf** и перезапустите сервер (если он работает в автономном режиме). После этого шифрование SSL будет поддерживаться. Любой клиент, поддерживающий SSL, сможет установить защищенное соединение.

```
set tls-support = stls
set tls-server-cert-file = /etc/mail/certs/cert.pem
```

При желании можно имитировать уполномоченных представителей сертификации (CA) самостоятельно, создав автосертификат (self-signed certificate), с которым система сможет работать. Однако клиенты с поддержкой SSL не доверяют таким сертификатам и при установлении соединения требуют подтверждения от пользователя.

Вначале необходимо создать тестовый личный ключ CA (не забудьте введенный пароль!), а затем — сертификат CA:

```
# openssl genrsa -des3 -out ca.key 1024
...
# openssl req -new -x509 -days 365 -key ca.key -out ca.crt
```

Теперь можно самостоятельно подписать сертификат по созданному ранее запросу (**req.pem**):

```
# openssl x509 -req -CA ca.crt -CAkey ca.key -days 365 -in req.pem -out
signed-req.pem -Cacreateserial
```

Это позволит запустить сервер **qpopper** с поддержкой SSL для проверки его функциональности. Для реальной работы потребуется сертификат, подписанный настоящими уполномоченными сертификации!

Официальный Web-сайт **qpopper** находится по адресу **http://www.eudora.com/qpopper**. Здесь можно найти множество полезной информации, включая документ в формате PDF (в разделе Documentation), в котором описаны все возможные опции конфигурирования **qpopper**. Этот документ создан с точки зрения системы Linux, поэтому многие имена каталогов отличаются от тех, что используются в инсталляции FreeBSD. Подробную информацию можно найти в **man qpopper**.

Настройка IMAP-сервера IMAP-UW

IMAP (Interactive Mail Access Protocol — Протокол интерактивного доступа к электронной почте) — представляет собой альтернативу POP, предпочитаемую некоторыми пользователями. Вероятно, на почтовом сервере этот протокол понадобится наравне с POP и SMTP. Фундаментальное различие между IMAP и POP заключается в том, что POP загружает каждое сообщение с сервера и сохраняет в почтовой программе, тогда как IMAP обращается к почте и управляет ей сугубо на сервере. Клиент IMAP, как серверная почтовая программа-клиент, может поддерживать несколько почтовых каталогов и пересылать сообщения между ними так же, как если бы они были локальными. Клиенту сообщения пересылаются только при запросе, и не удаляются до тех пор, пока пользователь явно не укажет сделать это. Такой подход обеспечивает гибкость таких текстовых MUA, как Mutt и Pine, в сочетании с удобством графического интерфейса почтовых программ Outlook и Eudora. Почти все почтовые клиенты поддерживают IMAP наравне с POP.

Одним из самых популярных IMAP-серверов является сервер IMAP-UW, созданный в Вашингтонском университете (той же группой, что разрабатывает Pine). Он имеется в наборе портов в каталоге **/usr/ports/mail/imap-uw** и в наборе пакетов (см. главу 15). Чтобы включить поддержку SSL, при сборке добавьте опцию **USE_SSL=YES** обеим командам: **make** и **make install**.

ПРИМЕЧАНИЕ

Порт IMAP-UW предупреждает о проблеме с защитой: известны ситуации, когда переполнение буфера в демоне позволяет пользователям IMAP получить доступ к интерпретатору с собственными правами. Это может оказаться проблемой и для вас. Если пользователям в любом случае предоставляется доступ к интерпретатору, это не беда. Если же нет, это серьезная дыра в защите. Можно поступить следующим образом: ожидать обновления, исправляющего проблему; подыскать другой IMAP-сервер; запретить IMAP, полностью полагаясь на POP.

Пакет IMAP-UW состоит из программы, тестирующей почтовый ящик **mboxtest**, и файлов, размещенных в каталоге **/usr/local/libexec**. Два из них, **ipop2d** и **ipop3d**, представляют собой POP-серверы (для POP2 и POP3, соответственно), которые не требуются, если в системе уже установлен **qpopper**. Однако эти демоны позволяют перенаправлять команды POP на IMAP-сервер, что позволяет перевести всех POP-клиентов на IMAP, если вы хотите ограничиться поддержкой только этого протокола.

Инсталляция IMAP-UW достаточно проста. Сама программа не требует конфигурационных файлов — все необходимые изменения вносятся в **/etc/inetd.conf** (поскольку IMAP-сервер работает через **inetd**, как и **qpopper**) и в **/etc/pam.conf** (дополнительно).

Закомментированная строка в **/etc/inetd.conf** содержит все необходимое для работы IMAP-UW. Достаточно устранить комментарий:

```
# пример записи для сервера imap4
#
imap4   stream   tcp    nowait   root    /usr/local/libexec/imapd      imapd
```

После этого необходимо перезапустить демон **inetd**:

```
# killall -HUP inetd
```

Демон IMAP прослушивает TCP-порт 143. Для проверки работы сервера нужно подключиться к этому порту по Telnet. Для окончания сессии следует нажать комбинацию клавиш Ctrl+] и ввести команду **quit**.

```
# telnet localhost 143
Trying 127.0.0.1...
Connected to localhost.somewhere.com.
Escape character is '^]'.
* OK [CAPABILITY IMAP4 IMAP4REV1 LOGIN-REFERRALS AUTH=LOGIN]
localhost.somewhere.com IMAP4rev1 2001.303 at Sun, 10 Jun 2001 11:21:26 -
0700 (PDT)
^]
telnet> quit
Connection closed.
```

Сценарий инсталляции предлагает внести изменения в записи **imap** в файле **/etc/pam.conf** (который управляет методами аутентификации), чтобы ограничить доступ. Для этого замените строку:

```
imap    auth    required    pam_unix.so    try_first_pass
```

следующими:

```
imap    auth    required    pam_unix.so
imap    account required    pam_unix.so    try_first_pass
imap    session required    pam_deny.so
```

То же самое можно сделать и с правилами **pop3**:

```
pop3    auth    required    pam_unix.so
pop3    account required    pam_unix.so    try_first_pass
pop3    session required    pam_deny.so
```

ПРИМЕЧАНИЕ

В файле **/var/log/messages** или в клиентском программном обеспечении IMAP можно заметить предупреждения о том, что почтовый ящик "уязвим", и что права доступа к **/var/mail** должны быть установлены как 1777. Такую проверку защиты производит сервер IMAP-UW (ее выполняет и Pine).

Почтовый каталог во FreeBSD имеет права доступа 775, т.е. все программы, которым требуется доступ для изменения или создания файлов в **/var/mail** должны иметь бит **setgid** для почтовой группы. Однако IMAP-UW не запускается как **setgid** и не может создавать в каталоге **/var/mail** lock-файлы, предотвращающие доставку почты в почтовый ящик при получении новых сообщений. Правильное решение заключается в изменении прав доступа к **/var/mail** на 1777. Это позволяет пользователям добавлять и удалять файлы, но в файлы могут вносить изменения только их владельцы. Это не идеальное решение, но в большинстве случаев оно подходит.

Шифрование SSL/TLS можно применять с IMAP так же, как и с протоколом POP3. При сборке программы из набора портов, выполните команды **make USE_SSL=YES** и **make install USE_SSL=YES** вместо **make** и **make install**. После этого можно создать сертификат, используя **cert**, или просто скопировать сертификат, созданный для **qpopper**, в **/usr/local/certs** (где IMAP-UW ищет этот файл). Если сертификат будет использоваться для обоих протоколов, можно изменить конфигурацию **qpopper** так, чтобы она указывала на тот же сертификат в **/usr/local/certs**.

Альтернативный способ защиты протоколов POP3 и IMAP — использование программы **stunnel**. О ней рассказано в главе 29.

Электронная почта для отдельных рабочих станций

Умение настроить полнофункциональный почтовый сервер — важный навык системного администрирования. Вам также потребуются знания о том, как вместо серверных средств использовать рабочую станцию с диспетчером окон Gnome или KDE и графическими приложениями.

Использование Fetchmail для получения почты с POP3- и IMAP-серверов

Большинство потребительских операционных систем содержат почтовые программы-клиенты (Outlook, Eudora, Netscape Messenger), имеющие собственные внутренние механизмы для проверки POP3- или IMAP-серверов. Каждая из них по-своему запрашивает почту и сохраняет ее присущим ей образом. Некоторые программы MUA такого типа существуют и во FreeBSD.

Кроме того, на рабочей станции можно использовать те же MUA, например, Mutt, Elm или Pine, что и на сервере. Эти программы работают напрямую с каталогом **/var/mail** и файлом почтового спула в нем, не имея собственного кэша для загруженных сообщений. Это позволяет разным программам одновременно работать с почтовым ящиком, производить поиск в нем с помощью средств командной строки и пользоваться другими гибкими средствами UNIX.

Чтобы получить сообщения с POP3- или IMAP-сервера, где они хранятся, на локальную рабочую станцию FreeBSD, требуется средство, способное установить POP3- или IMAP-соединение и доставить новые сообщение из удаленного почтового ящика в локальный. Этим средством является Fetchmail — небольшая, но гибкая утилита, написанная Эриком Рэймондом (Eric Raymond).

Fetchmail можно инсталлировать из набора портов (**/usr/ports/mail/fetchmail**) или пакетов. Его работой управляет файл **.fetchmailrc**, находящийся в home-каталоге пользователя. В нем указаны POP3- и IMAP-серверы, где хранится новая почта, интервалы времени, через которые ее нужно проверять, и другие опции. Fetchmail получает новую почту через указанные промежутки времени и пересылает ее на порт 25 рабочей станции, который в конфигурации по умолчанию прослушивает Sendmail.

Когда Fetchmail установлен, необходимо сконфигурировать файл **.fetchmailrc** в начальном каталоге. Это можно сделать двумя способами. Первый из них требует использования программы **fetchmailconf**. Это программа конфигурирования Fetchmail с графическим интерфейсом, написанная на Python. Она использует набор инструментальных средств TK (SDK для создания графических интерфейсов в X Window). Программа **fetchmailconf** требует, чтобы в системе были установлены Python и TCL/TK.

Второй способ предполагает создание файла **.fetchmailrc** вручную. Его формат достаточно прост. Назначение каждой опции объясняется в **man**-странице Fetchmail.

Настройка файла .fetchmailrc

Находясь в начальном каталоге, откройте в текстовом редакторе новый файл под названием **.fetchmailrc**. Он должен содержать три раздела. Важно, чтобы их порядок совпадал с тем, что приведен ниже. Отметьте также, что опции из предыдущего раздела нельзя использовать в следующем. Одной из распространенных причин ошибок в файлах **.fetchmailrc** является неверный порядок или местоположение опций.

Здесь приведен лишь пример настроек для получения почты с POP3-сервера. Как упоминалось ранее, страница справочного руководства по Fetchmail чрезвычайно обширная. На начальном этапе воспользуйтесь информацией из этого раздела, а по мере усложнения конфигурации обратитесь к справочному руководству.

Первый раздел файла **.fetchmailrc** содержит глобальные опции. Они применяются ко всем почтовым серверам и учетным записям пользователей, которые будут перечислены в конфигурационном файле далее. Некоторые из глобальных опций можно отменить с помощью специфических опций сервера или пользователя, но общее правило гласит, что все они действуют для всех серверов и учетных записей, проверяемых Fetchmail. Вот пример раздела глобальных настроек:

```
set daemon 600
set postmaster foobar
set logfile ./.fetchmail.log
```

Первая строка заставляет Fetchmail запуститься в режиме демона и проверять новую почту через каждые 600 секунд (10 минут). После запуска система проверяет новую почту и переходит в фоновый режим как демон. Если эта строка отсутствует, Fetchmail после запуска проверяет новую почту и завершает работу.

Во второй строке указан адрес отказа (fallback address). Вся почта, получаемая Fetchmail и не адресованная локальному пользователю, пересылается на эту учетную запись в локальной системе. Вероятно, здесь нужно указать ту же учетную запись, под которой запускается Fetchmail.

И наконец, третья опция определяет log-файл, в который Fetchmail записывает информацию о своих действиях. Здесь можно использовать и альтернативную строку **set syslog**, тогда для ведения журнала будет использоваться демон **syslogd**. Последний представляет собой демон системного уровня, который обрабатывает все системные события.

Это основные опции, применяемые в глобальном разделе.

Следующий раздел файла **.fetchmailrc** предназначен для серверов. В нем содержится информация обо всех почтовых серверах, на которых осуществляется проверка почты. Вот пример серверного раздела, в котором сконфигурирован один почтовый сервер:

```
poll mail.samplenet.org
proto pop3
no dns
```

В первой строке указана проверка новой почты на сервере **mail.samplenet.org** через интервал времени, заданный опцией **set daemon** в глобальном разделе. Альтернативная опция **skip mail.samplenet.org** пропускает этот сервер и не проверяет почту ни через определенный интервал времени, ни при запуске системы Fetchmail вручную.

Если применяется эта опция, проверка новой почты на данном сервере осуществляется лишь тогда, он указан в командной строке запуска Fetchmail.

Во второй строке указан протокол, используемый Fetchmail для соединения с сервером. В данном случае это POP3. Возможные опции: POP3, IMAP, APOP и KPOP.

Третья опция отключает поиск в DNS. Если вы подключены к Internet по коммутируемой линии, вам стоит включить эту строку.

В серверном разделе используется еще несколько опций. Узнать о них подробнее можно на странице справочного руководства **man fetchmail**.

Третий и заключительный раздел файла **.fetchmailrc** предназначен для пользовательских настроек. Он содержит информацию об учетных записях. Вот пример опций из этого раздела:

```
user foobar
pass secretword
fetchall
flush
```

Первая и вторая строки содержат имя пользователя и пароль, необходимые для доступа к почтовому серверу ISP-провайдера.

Третья строка указывает Fetchmail получать с сервера все сообщения, в том числе и те, что уже были прочитаны.

В заключение, в четвертой строке задана опция "очистки сервера" (flush) (все загруженные с сервера сообщения удаляются). Она не является обязательной, поскольку такая же установка применяется по умолчанию.

Другие опции позволяют, например, не удалять почту с сервера после ее загрузки на локальную машину. Подробные сведения о них приведены на странице справочного руководства Fetchmail.

Полностью конфигурационный файл Fetchmail выглядит следующим образом:

```
set daemon 600
set postmaster foobar
set logfile ./.fetchmail.log

poll mail.samplenet.org
proto pop3
no dns

user foobar
pass secretword
fetchall
flush
```

Он содержит следующие инструкции: "Проверять новую почту каждые 600 секунд. Посылать почту без адресата пользователю 'foobar'. Записывать все действия в файл **.fetchmail.log**. Проверять новую почту на сервере **mail.samplenet.org** каждые 10 минут по протоколу POP3 и не предпринимать попыток поиска в DNS. Для регистрации на сервере использовать имя **foobar** и пароль **secretword**, загрузить все сообщения и после этого удалить их с сервера".

ПРЕДУПРЕЖДЕНИЕ

Так как файл **.fetchmailrc** содержит имя пользователя и пароль, необходимые для регистрации на почтовом сервере, права доступа к нему должны быть установлены на чтение только вами. Таким образом, значение не должно быть выше, чем 600. Воспользуйтесь командой **chmod**

600 .fetchmailrc. Это предоставляет права на чтение и запись только владельцу файла. Если файл **.fetchmailrc** имеет более либеральные права доступа, Fetchmail не запустится.

Настройка Sendmail для отдельных рабочих станций

Для работы Fetchmail требуется, чтобы входящие сообщения принимались на 25 порту системой Sendmail (или ее эквивалентом). Кроме того, Sendmail полезен и для исходящей почты, поэтому не следует использовать рабочую станцию, не имеющую никакого MTA. Например, Mutt и подобные почтовые клиенты пересылают сообщения через Sendmail, не подключаясь при этом к внешнему SMTP-хосту.

Существует несколько способов улучшения производительности Sendmail и конфигурирования этой системы для использования на машине, не подключенной к Internet постоянно. Первый из них состоит в настройке "умного" почтового хоста, а второй — в автоматизации запуска почтового спула.

В файле **/etc/mail/freebsd.mc** содержатся следующие строки:

```
dnl Dialup users should uncomment and define this appropriately
dnl define(`SMART_HOST', `your.isp.mail.server')
```

Уберите комментарий со строки **define**, переместив дескриптор **dnl** в ее конец и замените **your.isp.mail.server** именем соответствующего почтового сервера. Соберите заново конфигурационный файл, воспользовавшись командами **make cf** и **make install** в каталоге **/etc/mail**.

В этом случае рабочая станция действует как любая настольная операционная система, где каждый почтовый клиент производит собственные SMTP-соединения с SMTP-сервером, поддерживаемым ISP-провайдером. Этот сервер пересылает почту в точку назначения. Так как в системе будут использоваться текстовые MUA, например, Mutt или Pine, которые пересылают почту Sendmail, не пытаясь установить SMTP-соединение с указанным SMTP-сервером, необходим централизованный способ для того, чтобы почтовые клиенты вели себя как коммерческие Windows-программы. Определение переменной **SMART_HOST** заставляет Sendmail перенаправить всю исходящую почту SMTP-серверу ISP-провайдера.

Цель этой настройки заключается в следующем. Как вы помните, большинство SMTP-серверов отвергает почту, приходящую с IP-адресов, неразрешаемых по обратному DNS-поиску, а многие сети ISP не содержат информацию для поиска по DNS для клиентов, подключающихся по коммутируемой линии. В этом случае рабочая станция FreeBSD не сможет отправлять почту. Но если ISP-провайдер поддерживает SMTP-сервер, который осуществляет релейную пересылку почты от клиентов (об этом свидетельствует то, что работают агенты MUA для Windows), определение переменной **SMART_HOST** обеспечит доставку сообщений.

Следующий элемент конфигурации, который нужно настроить, это очередь исходящих сообщений. По умолчанию Sendmail производит проверку очереди каждые 30 минут (для этого используется опция командной строки **-q30m**, установленная в файле **/etc/defaults/rc.conf**). Если вы подключаетесь по коммутируемой линии на короткие промежутки времени, этот интервал может оказаться слишком большим. Добавьте в файл **/etc/rc.conf** опцию, отменяющую настройку по умолчанию:

```
sendmail_flags="-bd -q10m"
```

Более эффективный способ состоит в том, чтобы оставить опцию **-q30m** без изменений и запускать проверку очереди при каждом подключении к Internet. Как вы помните, для этого можно воспользоваться командой **sendmail -q**, добавив необязательную опцию **-v**.

Запуск вручную может оказаться утомительным, поэтому команду **sendmail -q** можно, например, добавить в сценарий **ppp-up**, о котором было рассказано в главе 24.

Альтернативные почтовые серверы

Хотя Sendmail считается ведущим MTA, он также имеет и недостатки. Его критикуют за сложные конфигурационные файлы, главный конфигурационный файл в малоизвестном формате **m4**, большое число различных файлов времени исполнения (например, база данных доступа и файл **virtusertable**), относительно большой размер и невысокую скорость. Если вас устраивает то, как Sendmail работает в конфигурации по умолчанию (что обычно не требует дополнительных усилий по настройке), вам не стоит беспокоиться. Если же вам требуется более высокая скорость, меньшее потребление ресурсов, более удобное конфигурирование, существуют системы, способные заменить Sendmail. Все они, естественно, имеют свои достоинства и недостатки.

Postfix

Возможно, одной из самых распространенных альтернатив Sendmail является система Postfix, разработанная Вьетсом Венемой (Wietse Venema). Основными задачами этой разработки были повышение скорости и улучшение защиты. Структура этого агента похожа на Sendmail, что облегчает работу администраторов, использующих Sendmail и нуждающихся в повышении производительности.

По сообщениям, скорость работы Postfix высока (эта система быстрее, чем, например, Qmail, хотя подобные характеристики тяжело измерить эмпирически). Недостатком Postfix является сравнительная новизна этой системы, поскольку она не обладает дополнительными "зрелыми" возможностями Sendmail или Qmail. Домашняя страница Postfix расположена по адресу **http://www.postfix.org**.

Qmail

Дэн Бернстайт (Dan Bernstein) разработал систему Qmail как альтернативу Sendmail с точки зрения скорости и защиты. Этот сервер используется такими известными службами Internet, как Hotmail, Network Solutions и Yahoo! mail. Внутренняя структура Qmail создана с нуля, а не заимствована из Sendmail. Это значит, что конфигурация и структура файлов инсталляции Qmail не повторяют структуру Sendmail. Qmail хорошо подходит тем администраторам, которым требуется высокая скорость работы и надежная защита. Домашняя страница Qmail находится по адресу **http://www.qmail.org**.

Exim

Разработанная в Кембриджском университете, система Exim обладает следующими достоинствами: обилие документации, поддержка разработчиками и списки рас-

сылки. Эта система легко конфигурируется, во многом благодаря обширной документации. Нужно отметить, что Exim в прошлом имел проблемы с защитой и представляет собой плохую альтернативу Sendmail в том, что касается модульности и расширенных свойств. Однако это высокопроизводительный сервер, который может заметно ускорить работу SMTP-службы по сравнению с Sendmail. Домашняя страница Exim находится по адресу **http://www.exim.org**.

Smail

Более ранний MTA, Smail, отличается простой конфигурацией и высокой степенью защиты. Структура его конфигурационного файла значительно проще, чем в Sendmail (она близка к Exim), но синтаксис ближе к Sendmail. Следует отметить, что поддержка Smail постепенно прекращается.

26 ГЛАВА

Конфигурирование Web-сервера

- Общие сведения о протоколе HTTP
- Получение и инсталляция сервера Apache
- Конфигурирование сервера Apache
- Запуск и останов демона HTTP
- Основы контроля доступа к серверу
- Виртуальный хостинг
- Модули сервера Apache
- Серверные расширения
- Основы CGI

Сейчас трудно оценить, что сильнее загружает современный Internet, электронные сообщения или Web-приложения. Хотя электронная почта навсегда изменила способы обмена сообщениями, тем не менее по своей значимости ничто не может сравниться с рождением Web-пространства — новой среды для развлечения и коммерции.

Предположим, что при настройке сервера с FreeBSD требуется установить Web-сервер, а также обеспечить пересылку электронной почты и доступ пользователей к командному интерпретатору. В конфигурацию FreeBSD по умолчанию включена Sendmail — система, предоставляющая почтовые службы и не требующая никакой дополнительной конфигурации. С другой стороны, никакой эквивалент Web-сервера по умолчанию не устанавливается. Поэтому его необходимо инсталлировать самостоятельно из портов или пакетов. В 90% случаев в качестве Web-сервера используется Apache. Существует и несколько альтернативных серверов (например, Roxen или AOLserver), но Apache занял значительную долю рынка среди серверов Internet, поскольку был первой альтернативой серверу NCSA **httpd**.

В наши дни Apache пользуется гораздо большей популярностью, чем, допустим, Sendmail, и при этом практически не испытывает давления со стороны конкурентов. Проект Apache — один из лучших примеров философии программного обеспечения с открытым кодом в действии. Этот проект дал миру программное обеспечение, которое работает надежней, чем другие альтернативные серверы. Кроме того, оно лучше совместимо с утвержденными стандартами. Apache приспособлен к любым типам сред — начиная от небольших информационных Web-сайтов с низким трафиком и заканчивая полномасштабными серверами электронной коммерции с сотнями параллельных соединений. Плюс ко всему он поддерживает набор подключаемых модулей, обеспечивающих такие дополнительные возможности, как связь с базами данных и встроенная обработка сценариев на языке Perl.

В этой главе обсуждается, как настроить Apache для инсталляций разного типа.

Общие сведения о протоколе HTTP

HyperText Transfer Protocol (HTTP, протокол передачи гипертекста) — основа World Wide Web. Разработанный в 1993 году для поддержки обмена информацией в центре CERN (Швейцария), он представляет собой простейший протокол, не требующий аутентификации и включающий лишь несколько команд с клиентской стороны. Первоначально он был оптимизирован для пересылки небольших текстовых файлов. HTTP обеспечивает распространение связанных между собой информационных страниц, язык форматирования которых (понятный HTTP-браузерам) был разработан относительно недавно. Он называется HyperText Markup Language (HTML, язык разметки гипертекста).

В настоящее время использование HTTP вышло далеко за пределы его первоначальной спецификации. Сейчас этот протокол используется для пересылки больших двоичных файлов, изображений, анимации и других элементов HTML-страниц. На заре WWW это было редкостью (поддержка форматов изображений GIF и JPEG начала распространяться на различных платформах лишь в 1994 году). Отвечая велениям времени, спецификации HTTP были несколько изменены. Наиболее важным являет-

ся стандарт HTTP/1.1, поддерживающий такие свойства, как *конвейеры* (объединение нескольких одновременных запросов в один ответный поток). Большинство браузеров следует этому стандарту лишь частично.

В отличие от SMTP, FTP и других популярных протоколов, HTTP не поддерживает "сеансы". Когда клиент соединяется с сервером, HTTP производит несколько операций, а потом закрывает соединение. HTTP при соединении позволяет выполнить только один запрос. Кроме того, это означает, что данный протокол не поддерживает топологические свойства, ассоциированные с таким протоколом, как, например, SMTP: переключение (relaying), записи MX, очереди и т.д. Поэтому по мере увеличения числа одновременных соединений администратору HTTP-сервера приходится больше думать о полосе пропускания, ресурсах процессора и памяти, заботиться об их наиболее эффективном использовании.

Структура запроса в протоколе HTTP/1.0 чрезвычайно проста. Эмулировать операцию HTTP можно, подключившись к порту 80 на HTTP-сервере и выполнив запрос GET, который может содержать несколько строк следующего вида:

```
# telnet www.somewhere.com 80
Connected to www.somewhere.com.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Sun, 20 May 2001 22:45:55 GMT
Server: Apache/1.3.20 (Unix)
Content-Location: index.html
Vary: negotiate,accept-language,accept-charset
TCN: choice
Last-Modified: Fri, 31 Mar 2000 01:45:46 GMT
ETag: "6531f-54e-38e4034a;3a977613"
Accept-Ranges: bytes
Content-Length: 1358
Connection: close
Content-Type: text/html
Content-Language: en
Expires: Sun, 20 May 2001 22:45:55 GMT

<HTML>
<TITLE>test page</TITLE>
<BODY>
test
</BODY>
</HTML>
```

Чтобы получить только заголовки (а не полное сообщение), следует выполнить запрос HEAD. Ввод пустой строки (двойное нажатие на клавишу Enter) указывает на окончание многострочного запроса.

Первый блок строк в ответе представляет собой заголовок. По его содержимому легко определить, какой HTTP-сервер возвратил его, поскольку каждый сервер имеют уникальную сигнатуру в строке **Server:**. Остальные строки, особенно строки **Content-*:**, содержат информацию, которая помогает Web-браузеру правильно отформатировать страницу. Например, **Content-Length:** указывает браузеру, сколько данных осталось загрузить, а **Content-Type:** — как представить запрошенный файл (HTML, обычный текст, изображение GIF или JPEG и т.д.).

HTTP/1.0 позволяет включать в запрос дополнительные строки, например, наборы cookies, допустимые кодировки, предпочтительные языки и т.д. Кроме строки запроса (которая должна быть первой), порядок строк не имеет значения. Они являются дополнительными, обязательна лишь сама строка запроса. Запрос HTTP/1.1 практически совпадает с HTTP/1.0, за исключением того, что вторая строка (**Host:**) также является обязательной. Это дополнение к протоколу предназначено для поддержки виртуального хостинга (когда один Web-сервер может отзываться на несколько разных хостовых имен). Это значит, что клиент должен указать имя хоста, Web-содержимое которого он хочет получить. Поскольку Web-браузер определяет IP-адрес по имени хоста, указанного пользователем, и затем устанавливает HTTP-соединение с этим IP-адресом (о работе TCP/IP рассказано в главе 22), сервер ничего не знает об имени хоста, к которому пытается обратиться пользователь. Поэтому клиенту и необходимо указать заголовок **Host:**.

```
# telnet www.somewhere.com 80
Connected to www.somewhere.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.somewhere.com
```

Все современные браузеры, включая и текстовые (типа Lynx), поддерживают стандартный для HTTP/1.1 заголовок **Host:**. (Однако они не обязательно формируют свои запросы так, как HTTP/1.1. Например, Netscape Navigator поддерживает множество свойств HTTP/1.1, но посылает запросы HTTP/1.0.) Таким образом, виртуальный хостинг, основанный на информации из заголовка **Host:** (а не на IP-адресе и псевдонимах сетевого уровня), значительно упрощает настройку серверов. О виртуальном хостинге рассказано далее в этой главе.

Коды ответа и перенаправление

Хотя существует лишь несколько методов запроса (**GET**, **HEAD** и **POST**, а также несколько дополнительных методов в стандарте HTTP/1.1), имеется множество вариантов ответа, которые может возвратить сервер. Они являются трехзначными кодами и сгруппированы по значению первой цифры. В табл. 26.1 показан полный набор кодов HTTP-ответа и объяснены их значения (в основном в контексте сервера Apache и его свойств).

Таблица 26.1 Коды HTTP-ответа

Числовой код	*Название*	*Значение*
	2xx — Успешное выполнение	
200	OK	Стандартный код успешного выполнения
201	Created (Создано)	
202	Accepted (Принято)	
203	Partial Information (Частичная информация)	
204	No Content (Нет содержимого)	

Числовой код	Название	Значение
	3xx — Перенаправление	
300	Multiple Choices	Запросы **MultiViews** или **CheckSpelling** нашли несколько соответствий
301	Moved Permanently	Последний символ косой черты опущен
302	Moved Temporarily	Найдено перенаправление
304	Not Modified	Можно использовать кешированную копию
	4xx — Ошибка клиента	
400	Bad Request (Плохой запрос)	
401	Unauthorized	Для продолжения необходима аутентификация
403	Forbidden	Права доступа или конфигурация сервера запрещают доступ
404	Not Found	Файла не существует
	5xx — Ошибка сервера	
500	Internal Server Error	Ошибка CGI-программы
501	Not Implemented (Не реализовано)	
502	Bad Gateway (Плохой шлюз)	
503	Service Unavailable	Ресурсы, необходимые для обработки запроса, недоступны

Вы наверняка сталкивались с ошибками 404 и 403, и даже 500, если вам приходилось заниматься CGI-программированием. Не слишком понятным является код 304. Он никогда не выводится пользователю и предназначен для браузера. Тем не менее это один из самых распространенных кодов ответа корпоративного сервера, в чем можно убедиться, просмотрев log-файлы доступа (**/var/log/httpd-access.log**).

Когда клиент запрашивает файл, уже имеющийся в кэше (например, изображение GIF или HTML-страницу), он выполняет запрос **GET**, в поле **If-Modified-Since** которого установлена дата последней загрузки изображения. Это позволяет серверу проверить, изменился ли файл с этого момента. Если да, сервер пересылает файл (с кодом успешного выполнения 200), если нет, он возвращает код **304 (Not Modified)**, сообщающий браузеру, что можно использовать кешированный файл.

Другой код, часто используемый браузерами, но не известный пользователям, — это **301 (Moved Permanently)**. Чаще всего эта ошибка происходит в случае, когда запрашивается URL типа **http://some.host.com/Subdirectory**, где **Subdirectory** — имя каталога на сервере. Корректная форма указания URL содержит замыкающий символ косой черты **http://some.host.com/Subdirectory/**. Обратите внимание, что, если URL введен без последнего символа косой черты, браузер добавляет его автоматически. Это происходит, потому что при первом запросе он получает код **301**, перенаправляющий его на тот же URL, но уже заданный в правильном формате. URL в строке браузера обновляется, браузер посылает второй запрос и получает в ответ

требуемую страницу. Чтобы процесс выполнялся правильно, серверу необходимо знать имя хоста. Этой цели служит директива **ServerName** сервера Apache, назначение которой обсуждается далее в этой главе.

Более подробную информацию можно найти на Web-сайте Консорциума W3 (WWW Consortium) по адресу **www.w3.org/Protocols**. Исходный протокол HTTP/1.0 определяется спецификацией RFC 1945, а HTTP/1.1 — RFC 2068.

Получение и инсталляция сервера Apache

Название сервера Apache, как принято считать, происходит от "A Patchy Server" (Сервер исправлений), поскольку он "вырос" из набора исправлений, которыми дополнялся стандартный, но в начале 1995 года имеющий весьма ограниченные возможности NCSA **httpd**-сервер. Apache — яркий пример самого широко портируемого на сегодня программного обеспечения. Он работает на платформах под управлением самых разных операционных систем — от AIX до Windows, от BeOS до Mac OS X, и, конечно же, FreeBSD. Этот сервер представляет собой не просто одну из наиболее полных реализаций HTTP-сервера, но и рассматривается многими как приближение к идеалу программного обеспечения с открытым кодом. Еще бы! Свободно разрабатываемый и распространяемый продукт стал стандартом де-факто в той области рынка, где существуют большие коммерческие пакеты от ведущих компаний отрасли. Apache занимает 60% рынка (на момент написания этой книги) и эта доля растет.

Apache устанавливается из портов (**/usr/ports/www/apache13**) или пакетов. На сегодняшний день Apache 1.3 законченная версия кода, чьи былые ошибки и проблемы с защитой давно решены. Именно эта версия — основа популярности и высокой репутации Apache. Она способна удовлетворить самым разным запросам пользователей. Тем не менее в этой версии имеется несколько структурных "причуд" (например, модель порождения дочерних процессов, когда для обработки каждого нового запроса создается новый процесс Apache). Поэтому код версии Apache 2.0 переписан заново, он включает управление цепочками ядра, а также высокомодульную архитектуру, предназначенную для повышения производительности. Сейчас Apache 2.0 находится в стадии альфа-разработки, поэтому в этой главе рассказано об Apache 1.3, а не о новой версии, которая включена в порты в каталоге **/usr/ports/www/apache2** (может, вы захотите опробовать ее).

Достоинством сервера Apache для пользователей FreeBSD является то, что многие его компоненты были разработаны именно в среде FreeBSD, в частности, это модуль URL Rewrite (**mod_rewrite**), написанный Ральфом Энгельшалем (Ralf Engelschall), а также несколько опций настройки производительности. Из этого также следует, что Apache хорошо сочетается с файловой структурой FreeBSD.

Дистрибутив с исходным кодом Apache можно найти на многих зеркальных серверах по всему миру, если набор портов не позволяет обновить его с центрального сервера. Список серверов, а также набор документации по Apache имеется на сайте **http://httpd.apache.org**.

Схема расположения файлов сервера Apache

После инсталляции Apache корневой каталог сервера размещается в каталоге **/usr/local**. Этот новый каталог, **/usr/local/www**, содержит несколько подкаталогов, часть из которых является символическими ссылками. Схема расположения файлов Apache показана на рис. 26.1. Все компоненты Apache устанавливаются внутри иерархии **/usr/local** (в точном соответствии с идеологией FreeBSD), за исключением log-файлов (которые находятся в каталоге **/var/log** вместе с остальными log-файлами системы).

Фактически, несколько каталогов присутствуют лишь во временной структуре файловой системы, существующей на этапе инсталляции. Это предотвращает случайную замену сценарием установки существующих файлов на работающем сервере. Например, **/usr/local/www/data.default** представляет собой символическую ссылку на **/usr/local/share/doc/apache**, а **/usr/local/www/data** (где в конфигурации по умолчанию находится корневой каталог документов, т.е. каталог, где хранятся основные HTML-страницы сервера) — отдельную символическую ссылку на тот же каталог. При необходимости ее можно заменить. Аналогично, **/usr/local/www/cgi-bin** это символическая ссылка на **/usr/local/www/cgi-bin.default** (обычный каталог). Файлы, размещаемые в **cgi-bin**, в действительности попадают в **cgi-bin.default**, однако при необходимости **cgi-bin** можно также превратить в отдельный каталог.

Рисунок 26.1
Схема размещения файлов Apache.

Файлы в каталоге **/usr/local/etc/apache** также имеют **.default**-эквиваленты. Таким образом, вы всегда можете обратиться к файлам конфигурации по умолчанию, если требуется изменить что-то в конфигурации работающего сервера.

Как и в случае с Sendmail, схема расположения файлов значительно отличается от тех, которые применяются при инсталляции Apache на других платформах. Поэтому знание схемы FreeBSD вряд ли поможет вам при поддержке Apache на серверах, работающих под Linux, Solaris или Windows.

Один совет, который может оказаться полезным: обычно в любой конфигурации существует "корневой каталог сервера", как, например, **/usr/local/www** во FreeBSD, где находятся все компоненты, связанные с Apache. В Linux это, как правило, **/var/lib/apache**. В инсталляции подобного вида почти все, связанное с Apache, включая конфигурационные, двоичные и log-файлы, исходный код и сценарии сборки, находится в пределах одного каталога. Хотя такая схема не обладает достоинствами чет-

кой и последовательной структуризации, присущей FreeBSD, по крайней мере, все, что связано с инсталляцией, хранится в одном месте.

Конфигурирование сервера Apache

Как и каталог **/etc/mail** (с которым мы познакомились в главе 25), **/usr/local/etc/apache** (конфигурационный каталог Apache) содержит несколько файлов, часть из них мы сейчас рассмотрим:

- **httpd.conf**. Это главный файл конфигурации Apache. Сегодня все настройки собраны в одном файле, а не в трех отдельных файлах, как было раньше.
- **mime.types**. В этом файле содержится таблица, указывающая соответствия между расширением файла и типом MIME (заголовки **Content-Type**) в файлах, посылаемых Apache. Благодаря ей браузеры узнают, как обрабатывать загружаемые ими файлы.
- **magic**. Альтернативным типам MIME методом является magic — "магическая последовательность", когда тип файла определяется по шаблону, содержащемуся в самом файле, что делает обработку расширений необязательной. Этот же метод используется и командой **file**.
- **access.conf** и **srm.conf**. В этих файлах ранее содержались определенные настройки, которые теперь включены в **httpd.conf**. Если в этих файлах заданы какие-либо опции, сервер прочтет их, поэтому они и включены в конфигурационный каталог (для поддержки совместимости с прошлыми инсталляциями), тем не менее эти файлы не являются обязательными и их можно смело опустить.

Отметьте, что все эти файлы имеют **.default**-эквиваленты. Сразу после инсталляции они идентичны. После внесения изменений в основные файлы, которые необходимы для работы сервера, у вас всегда будут под рукой файлы с настройками по умолчанию. При обновлении версии Apache будут затронуты только файлы **.default**. Это позволяет обновить систему и добавить новые опции конфигурации, используя **diff** или какой-либо другой метод.

Использование httpd.conf

Файл **httpd.conf** достаточно длинный и подробный, но (в отличие от **/etc/mail/sendmail.cf**) он имеет понятный текстовый формат и массу комментариев. Каждая директива предполагает значение по умолчанию. Любую из них можно изменить, а содержащаяся в файле документация четко объясняет назначение каждой опции. Если что-либо будет сконфигурировано некорректно, то при попытке запуска Apache точно сообщит, в чем проблема.

Сразу же после инсталляции Apache готов к запуску и правильной обработке запросов. Однако в целях безопасности несколько директив потребуется установить. Найдите их в файле **httpd.conf** и измените соответствующим образом:

```
#
# Адрес, на который высылаются сообщения о проблемах с сервером. Он
# отображается на некоторых страницах, генерируемых сервером, например, в
# сообщении об ошибке.
```

```
#
ServerAdmin you@your.address
#
# Опция ServerName позволяет установить имя хоста, которое пересылается
# клиентам, если оно отличается от того, которое программа получила бы
# по умолчанию (например, использовать имя с приставкой www вместо
# реального имени хоста).
#
#ServerName new.host.name
```

Вторая из приведенных опций, **ServerName** используется при перенаправлениях с кодом **301** (о них было рассказано ранее). Если замыкающий символ косой черты опущен при запросе индекса или листинга каталога, Apache возвращает код **301 (Moved Permanently)** и URL, по которому должен обратиться браузер. Apache воссоздает URL перенаправления по информации, содержащейся в запросе клиента, который содержит только часть URL. Например, нужный URL **http://some.host.com/images/foo**, а в запросе указано **/images/foo**. Если происходит запрос по протоколу HTTP/1.1, то в нем содержится заголовок **Host:**, которым Apache может воспользоваться для построения полного URL. Именно для этого и применяется директива **ServerName**.

Версия протокола HTTP/1.1 влечет за собой несколько последствий, связанных с директивой **ServerName**. Если применяется виртуальный хостинг по именам (к его рассмотрению мы вскоре перейдем), Apache использует значение директивы **ServerName** для сравнения имени виртуального хоста с заголовком **Host:** запроса. Для каждого виртуального хоста следует указать значение **ServerName**. Даже если виртуальный хостинг не используется, желательно установить опцию **ServerName**. Это поможет Apache правильно конструировать URL-перенаправления.

Подробную документацию можно найти на Web-сайте Apache Group — **http://httpd.apache.org/docs/**.

Использование файлов .htaccess и заменяющих опций

Глобальные опции конфигурации содержатся в файле **httpd.conf**. Однако действие многих из них можно отменить, пользуясь настройками внутри контент-каталога. Перезапускать сервер при этом не нужно. Для этого в каталоге, к которому требуется применить другие настройки, размещается файл **.htaccess**, содержащий новые значения директив. При поступлении каждого запроса сервер вначале обращается к глобальной конфигурации, загруженной в память из файла **httpd.conf**, а затем к конфигурационному файлу из каждого каталога (**.htaccess**), проходя последовательно по иерархии каталогов к местонахождению запрашиваемого файла. Каждый последующий файл **.htaccess** может отменить предыдущие директивы, поскольку этот файл задает конфигурацию для своего каталога и подкаталогов в нем.

Можно ли использоваться файлы **.htaccess**, зависит от директивы **AllowOverride**. В файле **httpd.conf** опция **AllowOverride** установлена в значение **None** на уровне корневого каталога операционной системы и на уровне каталога **/usr/local/www/data**. Таким образом, по умолчанию файлы **.htaccess** игнорируются. Чтобы разрешить их использование, следует заменить значение **None** директивы **AllowOverride** (в блоке **/usr/local/www/data**) любым значением из табл. 26.2 или их комбинацией (например, **AllowOverride AuthConfig Limit**).

Таблица 26.2 Опции конфигурации AllowOverride

Директива	*Разрешает использование файлов .htaccess для отмены директив*
AllowOverride Options	Директива Options
AllowOverride FileInfo	Директивы типов файлов, например, **AddType** и **ErrorDocument**
AllowOverride AuthConfig	Директивы авторизации, например, **Require** и **Auth***
AllowOverride Limit	Директивы доступа к хосту, например, **Allow**, **Deny** и **Order**
AllowOverride All	Все перечисленные директивы

ПРИМЕЧАНИЕ

Директива **Options** — наиболее объемный и гибкий элемент Apache. Она управляет наиболее важными свойствами сервера, такими как **ExecCGI** (возможность запуска CGI-сценариев), **Includes** (серверные расширения) и **MultiViews** (гибкая договоренность о содержимом). Все эти свойства можно добавить или устранить из текущей конфигурации на любом уровне иерархической структуры каталогов. Полное описание директивы **Options** см. на **http://httpd.apache.org/docs/mod/core.html#options**.

После внесения соответствующих изменений в файл **httpd.conf**, необходимо перезапустить сервер (об этом сейчас пойдет речь). Теперь файлы **.htaccess** можно разместить в требуемых Web-каталогах и включить в них необходимые директивы (за более подробными сведениями обратитесь к документации **http://httpd.apache.org/docs/**).

Файл **httpd.conf** содержит закомментированный блок для управления каталогами пользователей (внутри **/home**). Если вы планируете поддерживать сервер, на котором пользователи смогут размещать свои страницы, и хотите быть уверенным, что браузеры, поддерживающие различные методы изменения данных по стандарту HTTP/1.1 (**DELETE**, **COPY** и **MOVE**), не смогут изменить их, следует включить этот блок:

```
#
# Управление доступом к каталогам UserDir. Ниже приведен пример сервера,
# где доступ к ним возможен в режиме только для чтения.
#
#<Directory /home/*/public_html>
#       AllowOverride FileInfo AuthConfig Limit
#       Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#       <Limit GET POST OPTIONS PROPFIND>
#            Order allow,deny
#            Allow from all
#       </Limit>
#       <LimitExcept GET POST OPTIONS PROPFIND>
#            Order deny,allow
#            Deny from all
#       </LimitExcept>
#</Directory>
```

Заметьте, что директива **AllowOverride** разрешает файлам **.htaccess** отменять опции **FileInfo**, **AuthConfig** и **Limit**, но директивы **Options** установлены на глобальном уровне.

ПРИМЕЧАНИЕ

В соответствии с историческим соглашением, каталоги с Web-документами пользователей называются **public_html**. Запрос типа **http://some.host.com/~user/** покажет документы,

находящиеся в **/home/user/public_html**. Также по традиции, если присутствует файл **index.html**, Apache возвратит этот файл, а не листинг каталога. (Серверы компании Microsoft используют для этого файл с именем **Default.htm**.) Число индексных файлов может быть любым. Директива **DirectoryIndex** позволяет указать все эти файлы. Apache попытается найти файлы с заданными именами в указанном порядке.

Запуск и останов демона HTTP

Процесс инсталляции Apache из набора портов запускает сервер автоматически. Для проверки следует найти **httpd** среди текущих процессов (с помощью утилит **ps** и **grep**):

```
# ps -waux | grep httpd
root        220   0.0   2.0   4436 2456   ??   Ss   Sat02PM   0:02.79
/usr/local/sbin/httpd
nobody      303   0.0   2.0   4496 2548   ??   I    Sat02PM   0:00.01
/usr/local/sbin/httpd
nobody      304   0.0   2.0   4460 2452   ??   I    Sat02PM   0:00.00
/usr/local/sbin/httpd
nobody      305   0.0   2.0   4460 2452   ??   I    Sat02PM   0:00.00
/usr/local/sbin/httpd
nobody      306   0.0   2.0   4460 2452   ??   I    Sat02PM   0:00.00
/usr/local/sbin/httpd
nobody      307   0.0   2.0   4460 2452   ??   I    Sat02PM   0:00.00
/usr/local/sbin/httpd
nobody 13963   0.0   2.0   4468 2468   ??   I    Sat10PM   0:00.00
/usr/local/sbin/httpd
```

Обратите внимание на то, что Apache использует модель порождения новых процессов, в которой один главный (master) процесс (владельцем которого является **root**) прослушивает порт 80 и при поступлении запроса порождает свою копию (ее владельцем является уже псевдопользователь **nobody**, не обладающий никакими правами доступа). Это дочерний процесс, обрабатывающий поступивший запрос. Таким образом, главный процесс никогда не обслуживает запросов, поскольку Web-сервер, обрабатывающий запросы от имени пользователя **root**, был бы очень опасным для безопасности системы, поскольку он обладал бы возможностью запускать программы с правами **root**. В такой ситуации малейшая "дыра" или ошибка в CGI-сценарии привела бы или к взлому системы, или просто к потере данных.

При такой модели порождения процессов, когда одновременно могут выполняться десятки процессов **httpd**, удаление и перезапуск каждого запущенного процесса, разумеется, нецелесообразен. Существует следующий способ: для останова сервера достаточно удалить главный процесс, если же сервер нужно перезапустить, главному процессу необходимо послать сигнал **HUP**. В последнем случае сервер завершит все дочерние процессы, перезапустит свой собственный процесс и заново запустит все порожденные процессы.

Хотя приведенная схема довольно проста, она требует использования таких утилит, как **ps**, **grep** и **kill**. К счастью, Apache поддерживает удобное средство, предназначенное для решения этой задачи: **apachectl**, которое установлено в каталоге **/usr/local/sbin** (включенном по умолчанию в пути поиска). С помощью **apachectl** запустить или остановить сервер Apache достаточно просто:

```
# apachectl start
/usr/local/sbin/apachectl start: httpd started
# apachectl stop
/usr/local/sbin/apachectl stop: httpd stopped
```

После внесения изменений в файлы внутри каталога **/usr/local/etc/apache** сервер необходимо перезапустить. Программа **apachectl** позволяет выполнить и эту операцию:

```
# apachectl restart
/usr/local/sbin/apachectl restart: httpd restarted
```

Команда **restart** утилиты **apachectl** представляет собой эквивалент посылки сигнала **kill -HUP** главному процессу **httpd**: все порожденные процессы завершаются (даже если они обслуживают файлы клиентов); в этом случае клиенты получают сигнал о внезапном обрыве соединения. Если нужно перезапустить Apache менее варварским способом (например, на сильно загруженном сервере, где очень важна целостность данных), можно произвести более "изящный" перезапуск, когда вместо сигнала **SIGHUP** используется **SIGUSR1**. В этом случае главный процесс перезапускается без завершения порожденных процессов. Иначе говоря, пересылка файлов клиентам не прерывается.

```
# apachectl graceful
/usr/local/sbin/apachectl graceful: httpd gracefully restarted
```

Утилита **apachectl** часто применяется вместе с командой **configtest**. Она заставляет Apache (независимо от того, запущен он в данный момент или нет) прочесть конфигурационные файлы из каталога **/usr/local/etc/apache** и сообщить о найденных ошибках, как это сделала бы команда **start**. Единственное исключение: сервер не запускается, даже если ошибки отсутствуют. Это не просто замечательное средство диагностики, но и обязательная часть команд **restart** и **graceful**: утилита **apachectl** применяет **configtest** для определения корректности конфигурации до перезапуска сервера. Если установки непригодны к использованию, продолжает выполняться текущий процесс. Это предотвращает непреднамеренное завершение работы сервера командами **restart** или **graceful**:

```
# apachectl graceful
/usr/local/sbin/apachectl graceful: configuration broken, ignoring restart
/usr/local/sbin/apachectl graceful: (run 'apachectl configtest' for
details)
```

СОВЕТ

Если в данный момент сервер Apache не запущен, команды **apachectl restart** или **apachectl graceful** запускают его.

Основы контроля доступа к серверу

Определенные части Web-сайта, разумеется, не должны быть общедоступными. Контроль доступа позволяет ограничить их просмотр лишь узким кругом пользователей на основе аутентификации по имени хоста/IP-адресу или паролю. Здесь рассматриваются оба метода.

Контроль доступа по адресу

Предположим, что доступ к определенному файлу, каталогу или набору файлов и каталогов должен быть ограничен списком фиксированных IP-адресов или имен хостов. Выполнить эту операцию можно на глобальном уровне (**httpd.conf**) или же (что гораздо эффективнее) воспользоваться файлом **.htaccess** в каталоге, содержащем элементы, которые требуется защитить (или на один уровень выше).

Прежде всего необходимо убедиться, что отмена глобальных опций настройками **.htaccess** разрешена для той части сайта, которую требуется защитить. Как было показано ранее, для разрешения директив контроля доступа (**Allow**, **Deny** и **Order**) на уровне **/usr/local/www/data** (или глубже в структуре каталогов, или в другом контейнере <**Directory**>), используется директива **AllowOverride Limit**. Поскольку все директивы Apache читаются одновременно, следует указать порядок чтения директив **Allow** и **Deny**, иначе они будут отменять друг друга самым неожиданным образом.

Чтобы закрыть каталог для всех, кроме пользователей с указанных адресов или имен хостов, в нем необходимо разместить файл **.htaccess** следующего содержания:

```
Order deny,allow
Deny from all
Allow from 64.41.131.102
Allow from stripes.somewhere.com
Allow from nowhere.com
Allow from 10.5.100
Allow from 10.67.22.211/255.255.255.0
```

Как легко видеть, допускается несколько различных форматов адреса. Это просто правила поиска совпадения: если хост соответствует шаблону имени или сети, ему открывается доступ. Аналогично, можно открыть каталог для всех, но закрыть его для определенных хостов:

```
Order allow,deny
Allow from all
Deny from 133t.hacker.com
Deny from 192.168
```

Если настройки происходят на глобальном уровне (в файле **httpd.conf**), их необходимо разместить внутри блока <**Directory**> или <**Location**>. Чтобы правила контроля доступа применялись к определенным файлам (где бы в системе они ни находились), следует воспользоваться директивами <**Files**> или <**FilesMatch**>, задающими список таких файлов.

```
<Files "*.jpg">
   Order deny,allow
   Deny from all
   Allow from 64.41.131.102
</Files>
```

FilesMatch позволяет указать имена файлов с помощью регулярных выражений. Ниже приведен пример фильтра, обрабатывающего файлы с расширением **.gif**, **.jpg**, **.jpeg** и **.png**:

```
<FilesMatch "\ .(gif|jpe?g|png)$">
   Order deny,allow
   Deny from all
   Allow from 64.41.131.102
</FilesMatch>
```

Отметьте, что применение спецификаций **<Directory>** и **<Location>** в файлах **.htaccess** не допускается.

Блок **<Limit>**, несмотря на свое название (limit — предел, ограничение), не является обязательным для приведенных выше примеров контроля доступа. Его назначение — перечислить методы доступа, подлежащие контролю. Например, можно контролировать доступ для методов **GET** и **POST**, оставив все остальные методы без ограничений. А блок **<LimitExcept>**, наоборот, позволяет указать только те методы, к которым *не* применяется контроль доступа. Пример этих директив был приведен ранее в закомментированном блоке, управляющем пользовательскими каталогами внутри иерархии **/home**.

Контроль доступа по паролю

Иногда непрактично или даже нежелательно ограничивать доступ по адресу. Определенная часть сайта может быть доступна зарегистрированным пользователям (например, заплатившим взнос), вместе с тем необходимо обеспечить доступ к другой части сайта с заранее непредсказуемых IP-адресов. В таком случае для контроля доступа лучше воспользоваться аутентификацией по паролю.

Apache хранит собственные базы данных имен и паролей пользователей, отличные от баз данных системного уровня **/etc/master.passwd** и связанных с ним файлов. Это абсолютно необходимо для защиты, поэтому не стоит даже пытаться объединить две эти базы данных. База данных паролей FreeBSD предназначена для аутентификации пользователей, обращающихся к командному интерпретатору и имеющих учетную запись в системе, а базы данных аутентификации Apache управляют доступом Web-пользователей к определенным областям Web-сайта. В некоторых случаях две функциональности накладываются друг на друга (например, при использовании сервера в локальной сети компании), тем не менее не стоит пытаться обойтись одной базой данных. Даже если вам это удастся, такая методика — хороший пример плохой защиты системы и неграмотного системного администрирования.

Для контроля доступа с помощью пароля необходимо воспользоваться файлом **.htaccess** (или эквивалентной конфигурацией в глобальном файле **httpd.conf**), включая использование блоков **<Directory>**, **<Location>** и **<Files>**, как это было сделано при контроле доступа по адресу. Однако содержимое файла или блока конфигурации будет существенно отличается от предыдущего, поскольку управление доступом осуществляется уже другим модулем Apache. Вот пример блока опций:

```
AuthType Basic
AuthName "Restricted Area"
AuthUserFile /usr/local/www/.htpasswd
Require valid-user
```

Это минимальное число директив, необходимых для аутентификации такого типа. Каждая из них (или альтернативная ей) является обязательной. Рассмотрим эти директивы по отдельности:

- **AuthType**. Она может иметь значение **Basic** или **Digest**. Нас интересует только **Basic**.
- **AuthName**. Это название защищаемой области. Оно отображается в окне аутентификации пользовательского браузера (например, "Enter username for

'Restricted Area' at www.somewhere.com" — "Введите пароль для 'Защищенной области' на сайте www.somewhere.com"), помогая пользователю определить, какое имя пользователя и пароль требуется ввести. Название может быть любым. Если оно содержит пробелы, его следует заключить в кавычки.

- **AuthUserFile**, **AuthDBUserFile** и **AuthDBMUserFile**. Одна из этих директива должна быть обязательно указана, поскольку она задает местонахождение базы данных имен пользователей/паролей для защищенной области. Каждая область может иметь отдельную базу данных, если нужно. Если это обычный текстовый файл, указывается директива **AuthUserFile**, если он имеет формат **db** или **dbm**, ускоряющий поиск (весьма желательно при большом количестве записей), — директива **AuthDBUserFile** или **AuthDBMUserFile**, соответственно. О том, как создать пользовательскую базу данных в формате текстового файла или **db/dbm**, рассказано далее.
- **AuthGroupFile**, **AuthDBGroupFile** и **AuthDBMGroupFile**. Эти директивы работают так же, как **AuthUserFile**. В них заданы группы пользователей из файла **AuthUserFile**, что позволяет ограничивать доступ не только отдельным пользователям, а и целым группам. Эта директива не является обязательной, но если она используется, файл **AuthUserFile** также необходим, чтобы пользователи распознавались, как принадлежащие к перечисленным группам.
- **Require**. Эта директива указывает Apache, как аутентифицировать пользователя. Аргументом может быть **valid-user** (имя пользователя из файла **AuthUserFile**; при этом требуется ввод корректного пароля), **user <username> <username>...** (список пользователей) или **group <groupname> <groupname>...** (список групп).

Добавление пользователей

Когда блок контроля доступа уже настроен и команда **apachectl configtest** сообщает о корректности конфигурации, в систему можно добавить учетные записи пользователей Apache. Вначале рассмотрим, как создать базу данных в виде обычного текстового файла, указываемого как аргумент директивы **AuthUserFile**. Для этого применяется команда **htpasswd**, которая в FreeBSD расположена в каталоге **/usr/local/bin** и поэтому автоматически включена в пути поиска:

```
# htpasswd -c /usr/local/www/.htpasswd frank
```

Ключ **-c** обязателен только при первом запуске. Он указывает команде **htpasswd**, что файл нужно создать, поскольку он еще не существует. Затем команда запрашивает пароль пользователя, который требуется ввести, как обычно, дважды. Если команда используется в сценарии, ключ **-b** позволяет указать пароль во втором аргументе:

```
# htpasswd -b /usr/local/www/.htpasswd joe Prld3L4ndz
```

Следующей полезной опцией является **-m**. Она указывает **htpasswd** использовать алгоритм шифрования MD5, а не системную процедуру **crypt()**, обеспечивая тем самым более серьезную защиту. О других опциях рассказано на странице справочного руководства **man apachectl**.

ПРЕДУПРЕЖДЕНИЕ

Файл с пользовательской базой данных может находиться в любом месте системы, где его может прочесть пользователь Web-сервера (**nobody**), и иметь любое имя. Использование имени **.htpasswd** является традиционным и защищает файл от просмотра в конфигурации сервера по умолчанию. Естественно, не следует размещать файл там, где посетители сайта смогут прочесть его с помощью Web-браузера, т.е. в каталоге **/usr/local/www/data** или в каком-либо из пользовательских каталогов **public_html**. Для него вполне подойдет такое недоступное по Web-протоколу место, как **/usr/local/www**. Разумеется, вам совсем не нужно, чтобы пользователи смогли загрузить базу данных паролей и попытались ее взломать!

Пользовательские базы данных в виде обычных текстовых файлов подходят для небольших списков пользователей. Когда число пользователей вырастает до десятков или сотен, время, необходимое для поиска пароля в базе, делает такой способ аутентификации неудобным или даже нефункциональным. Решение заключается в использовании настоящей базы данных в формате **db** или **dbm** и соответствующей директивы **AuthDBUserFile** или **AuthDBMUserFile**. Программа, предназначенная для работы с базами данных в таком формате, называется **/usr/local/bin/dbmmanage**:

```
# dbmmanage /usr/local/www/.htpwddb adduser frank
New password:
Re-type new password:
User frank added with password encrypted to NtMDxy6jwyW7A using crypt
```

В этом примере создается **db**-файл **/usr/local/www/.htpwddb**, для доступа к которому применяется директива **AuthDBUserFile**. При необходимости можно воспользоваться и **dbm**-эквивалентом. Подробную информацию об управлении записями пользователей можно почерпнуть из справочного руководства **man dbmmanage**.

Добавление групп

Перечислить группы в обычном текстовом файле (**AuthGroupFile**) очень просто. Каждая строка содержит имя группы, двоеточие и список пользователей в ней:

```
mygroup: frank joe alice
```

Файл группы должен находиться там же, где и пользовательский, и иметь сходное имя. Так как файл группы не содержит паролей, соображения безопасности здесь не столь существенны.

Файлами групп можно управлять и с помощью утилиты **dbmmanage**, но на практике к ней, как правило, прибегать не приходится (как в случае с **/etc/group**), поскольку групп, как правило, гораздо меньше, чем пользователей. Директиву **AuthGroupFile** допускается использовать в сочетании с **AuthDBUserFile**, если нужно объединить форматы. Вероятно, это простейший способ поддержки большого списка пользователей с небольшим списком групп.

Контроль доступа по адресу и паролю

При предоставлении доступа к закрытой области можно пользоваться обоими типами защиты: ограничениями адреса и аутентификацией по паролю. Для этого применяется директива **Satisfy**:

```
Allow from 64.41.131.102
Require valid-user
Satisfy all
```

Директива **Satisfy all** используется по умолчанию, если присутствуют обе директивы — **Allow** и **Require**. Это означает, что пользователь, во-первых, должен удовлетворять требованиям директивы **Allow**, во-вторых, он должен указать подходящие имя и пароль, и только после этого ему будет предоставлен доступ к закрытой области. Директива же **Satisfy any** говорит серверу, что пользователь должен либо указать корректный пароль, либо его адрес должен удовлетворять требованиям директивы **Allow**. Такая настройка применяется тогда, когда часть пользователей посещает Web-сайт с определенных адресов, а для всех остальных пользователей требуется аутентификация.

Виртуальный хостинг

Виртуальный хостинг представляет собой методику хранения содержимого Web-сайтов с разными именами доменов или хостов на одном сервере. Для обслуживания всех запросов достаточно одного Apache. Например, именам **www.mystore.com** и **www.frankspage.com** в DNS может соответствовать один и тот же IP-адрес, и Apache обслуживает оба этих сайтов (равно как и собственное имя хоста, которое задано директивой **ServerName**).

Как мы убедились, протокол HTTP/1.0 не указывает имя хоста. Поэтому ранее виртуальный хостинг был возможен лишь в том случае, когда каждому имени хоста был поставлен в соответствие отдельный IP-адрес (с последующим созданием IP-псевдонимов, указывающих на одну и ту же Ethernet-карту). Каждый виртуальный хост определялся по IP-адресу, и запрос, приходящий от Web-браузера, всегда получал в ответ страницу соответствующего Web-сайта. Недостатком такого подхода было то, что привязка больших блоков IP-адресов к одной и той же карте становилась громоздкой и приводила и к излишнему потреблению адресного IP-пространства.

С появлением версии протокола HTTP/1.1 данный процесс значительно упростился. Обязательный заголовок **Host:** указывает искомое имя хоста, поэтому виртуальные хосты, различаемые по имени, стали нормой в современном Internet. Клиенты, не поддерживающие заголовка **Host:** теперь чрезвычайно редки. Далее обсуждается исключительно такой вариант виртуального хостинга. Если вы заинтересованы в использовании виртуального хостинга на базе IP-адресов, обратитесь к документации, имеющейся на Web-сайте Apache.

Большая часть файла **httpd.conf** определяет сервер по умолчанию — глобальный набор определений, применяющихся ко всем запросам, получаемым сервером Apache. В сервере по умолчанию директива **ServerName** используется в первую очередь для конструирования URL-перенаправления с кодом **301**. Можно также воспользоваться небольшим набором директив, отменяющим глобальные настройки в том случае, когда заголовок **Host:** совпадает с определенным именем хоста. Такие наборы правил и представляют собой виртуальные хосты.

Предположим, что сервер называется **stripes.somewhere.com**. Его имя задано в главной директиве **ServerName**. Для настройки виртуального хостинга по именам следует воспользоваться директивой **NameVirtualHost** с аргументом * (этот символ-заместитель означает "все хосты"), за которой следует необходимое число различных блоков **<VirtualHost *>**:

```
NameVirtualHost *

<VirtualHost *>
   ServerName www.somewhere.com
   DocumentRoot /usr/local/www/data
   ServerAdmin webmaster@somewhere.com
   ErrorLog logs/www.somewhere.com-error_log
   CustomLog logs/www.somewhere.com-access_log common
</VirtualHost>

<VirtualHost *>
   ServerName www.frankspage.com
   ServerAlias frankspage.com
   DocumentRoot /home/frank/public_html
   ServerAdmin frank@frankspage.com
   ErrorLog logs/www.frankspage.com-error_log
   CustomLog logs/www.frankspage.com-access_log common
</VirtualHost>
```

Внутри контейнера **<VirtualHost>** директива **ServerName** определяет имя хоста. Директива **DocumentRoot** указывает, где находится корневой каталог файловой системы для приходящего запроса, а **ErrorLog** и **CustomLog** — альтернативные log-файлы для каждого виртуального хоста. **ServerAlias** позволяет перечислить псевдонимы виртуального хоста. В блоке **<VirtualHost>** можно включить и любые другие директивы.

Важно отметить, что при настройках, приведенных ранее, запрос к серверу по умолчанию (**stripes.somewhere.com**) или к любому другому имени хоста, соответствующему IP-адресу сервера, но не совпадающему ни с одним из блоков **<VirtualHost>**, обработан не будет.

Вот пример более корректной конфигурации:

```
NameVirtualHost *

<VirtualHost *>
   ServerName stripes.somewhere.com
</VirtualHost>

<VirtualHost *>
   ServerName www.somewhere.com
   ServerAlias *.somewhere.com
   DocumentRoot /usr/local/www/data
   ServerAdmin webmaster@somewhere.com
   ErrorLog logs/www.somewhere.com-error_log
   CustomLog logs/www.somewhere.com-access_log common
</VirtualHost>

<VirtualHost *>
   ServerName www.frankspage.com
   ServerAlias frankspage.com
   DocumentRoot /home/frank/public_html
   ServerAdmin frank@frankspage.com
   ErrorLog logs/www.frankspage.com-error_log
   CustomLog logs/www.frankspage.com-access_log common
</VirtualHost>
```

Виртуальные хосты можно создавать множеством способов: указывая различные IP-адреса и порты в блоках **<VirtualHost>**. Синтаксис таких методов можно уточнить по адресу **http://httpd.apache.org/docs/vhosts/**.

Модули сервера Apache

Одним из главных преимуществ сервера Apache является его модульная структура. Каждая конфигурационная директива является частью того или иного модуля. В ранних версиях Apache они включались или отключались на этапе компиляции, теперь же они доступны как динамические разделяемые объекты (dynamic shared objects, DSO), что позволяет загружать их во время исполнения.

Встроенные модули

Просмотреть модули, статически скомпилированные с Apache, позволяет команда **httpd -l**:

```
# httpd -l
Compiled-in modules:
   http_core.c
   mod_so.c
suexec: disabled; invalid wrapper /usr/local/sbin/suexec
```

Это значит, что с гарантией доступны лишь директивы модулей **Core** и **mod_so**. Модуль **Core** обеспечивает доступ к фундаментальным директивам Apache, без которых запустить его просто невозможно, а **mod_so** — за поддержку DSO, которая позволяет загрузить все остальные тридцать два модуля, поставляемые с Apache, и любые из дополнительно установленных модулей.

ПРИМЕЧАНИЕ

Сообщение о **suexec** можно игнорировать. Эта утилита позволяет выполнять процесс Apache с **setuid**, когда запросы запускаются от имени определенного пользователя (обычно для разных виртуальных хостов пользователи эти разные). Оболочка **suexec** по умолчанию не устанавливается. Ознакомиться с документацией можно по адресу **http://httpd.apache.org/docs/suexec.html**. В главе 29 рассказано о CGIwrap — альтернативе **suexec**.

Динамически загружаемые модули

Динамически загружаемые модули позволяют уменьшить потребление ресурсов выполняемыми процессами **httpd**, выгрузив ненужные модули. Это легко сделать на этапе исполнения, просто закомментировав их в файле **httpd.conf**. Если функциональность каких-либо модулей вдруг понадобится, их можно включить заново и перезапустить Apache. Полный список всех модулей сервера Apache и соответствующих им директив имеется на **http://httpd.apache.org/docs/mod/**.

Для включения модуля требуются две директивы: **LoadModule** (динамически подключает модуль к процессу **httpd**) и **AddModule** (включает директивы модуля в правильном порядке). Просмотрев файл **httpd.conf**, легко убедиться, что директивы **LoadModule** и **AddModule** указаны для всех поставляемых модулей Apache, которые установлены в каталоге **/usr/local/libexec/apache**. Порядок загрузки модулей важен. Если один модуль зависит от другого, последний должен быть подключен раньше. В конфигурации по умолчанию указан корректный порядок модулей.

Сторонние модули

Широкие возможности модульной структуры Apache проявляются, когда в игру вступают модули от сторонних поставщиков. Эти модули подключаются на этапе

исполнения, как и все остальные модули Apache, и обеспечивают дополнительную функциональность и директивы конфигурации. Обычно они позволяют Apache более эффективно обрабатывать определенные типы данных на сервере. Например, одним из очень популярных модулей является **mod_perl**, который встраивает в Apache интерпретатор языка Perl, что значительно ускоряет обработку CGI-сценариев на этом языке, поскольку отпадает необходимость запускать процесс командного интерпретатора и исполнять CGI-программу в нем. Не менее популярен и модуль **mod_php4**, который позволяет Apache обрабатывать динамические PHP-страницы (более удачный UNIX-аналог технологии ASP). В разделе **/usr/ports/www** набора портов включены еще 34 сторонних модуля Apache. Их имена имеют префикс **mod_**.

Создание модулей с помощью apxs

Раньше, когда поддержка DSO была, в лучшем случае, экспериментальной, подключение такого модуля требовало изменения файлов конфигурации исходного кода Apache и исходного кода модуля, а затем компиляции одного "гибридного" двоичного файла. Хотя технически это был "модульный" подход (все-таки это лучше, чем внесение изменений непосредственно в исходный код Apache), им было сложно пользоваться. Он требовал сборки нового выполняемого файла при появлении новой версии Apache или модуля, а также немалых усилий при работе с кодом.

Затем появилось средство **apxs**, APache eXtenSion (расширение Apache). Эта программа, находящаяся в **/usr/local/sbin**, предназначена для того, чтобы, используя поддержку DSO, уже скомпилированную в двоичном файле (статический модуль **mod_so**), обеспечить среду сборки, учитывающую все свойства двоичного файла **httpd**. Теперь для создания сторонних модулей исходный код Apache не нужен: **apxs** компилирует и преобразует модули из исходного кода или из уже скомпилированных объектных файлов в файлы **.so**, которые можно загрузить с помощью директивы **LoadModule**, как и любые другие модули.

Скорее всего, вам не придется работать с утилитой **apxs** непосредственно. Она используется на этапе компиляции средствами сборки портов. Все, что требуется для сборки стороннего модуля (как и любого другого порта), это перейти в его каталог в иерархии **/usr/ports/www** и запустить команду **make**. Все остальное сделает **apxs**. Если появится новая версия Apache, ее можно будет собрать и инсталлировать независимо — сторонние модули будут продолжать работать. И наоборот, при обновлении модуля его достаточно заново собрать и установить. В этом проявляются преимущества модульной системы.

mod_perl

Один из самых популярных сторонних модулей — **mod_perl**. Поскольку язык Perl чаще всего используется при CGI-программировании, запуск сценариев Perl означает дополнительную работу для сервера Apache. Когда приходит HTTP-запрос на запуск CGI-программы на Perl, Apache требуется запустить процесс командного интерпретатора и выполнить сценарий в нем, возвращая затем результат клиенту.

С помощью модуля **mod_perl** интерпретатор языка Perl встраивается прямо в Apache. Специальные директивы определяют некоторые типы файлов как выполняемые сценарии на Perl, которые Apache может запускать самостоятельно. Более того, каждый

сценарий, запускаемый Apache, сохраняется в памяти в скомпилированном виде для дальнейшего использования. Это значительно ускоряет запуск CGI-программ, а значит, и ответ сервера.

Более подробную информацию о **mod_perl** можно найти по адресу **http://perl.apache.org**.

mod_python

С ростом популярности языка Python как наследника Perl в серверном программировании появился модуль **mod_python**, обеспечивающий Apache ту же функциональность по отношению к Python, какую **mod_perl** обеспечивает по отношению к Perl. Интерпретатор Python доступен Apache непосредственно, позволяя серверу самостоятельно запускать CGI-программы на этом языке.

Домашняя страница **mod_python** находится по адресу **www.modpython.org**.

mod_php

PHP, чрезвычайно популярное средство для создания динамических страниц в UNIX, также является частью Apache благодаря модулям **mod_php3** и **mod_php4**. Директивы этих модулей позволяют Apache естественным образом обрабатывать **php**-файлы, запуская их для генерирования окончательной формы страницы перед пересылкой ее клиенту. После сборки и инсталляции модуля **mod_php4** файл **httpd.conf** автоматически обновляется, чтобы обеспечить поддержку **php**- и **phps**-файлов.

Более детальная информация о PHP и **mod_php4** имеется на Web-сайте **www.php.net**.

Существует множество других модулей, расширяющих функциональность Apache. Модульность Apache была окончательно закреплена лишь недавно, а широкое распространение сторонних модулей в сочетании с набором портов FreeBSD послужили началом новой эры в администрировании HTTP-сервера.

Серверные расширения

Одно из самых популярных свойств Apache — это возможность обрабатывать серверные расширения, которые включены в HTML-файлы и обрабатываются сервером перед отправкой их браузеру. *Серверные расширения* (*server-side includes*) позволяют производить различные операции, начиная от вывода на страницу переменных среды до импорта HTML-фрагментов или запуска серверных-программ при каждом обращении к странице.

Серверные расширения встраиваются в так называемый *анализируемый HTML* (*parsed HTML*). Последний представляет собой обычный HTML-код, который сервер Apache анализирует при запросе страницы, находит серверные расширения и обрабатывает каждое из них перед отправкой документа клиенту. Каждое расширение имеет следующий формат:

```
<!--#command attribute=value attribute=value ... -->
```

В общем случае, командой **command** может быть **include**, **exec**, **config**, **echo** и т.д. Пары **attribute=value** позволяют установить (используя расширения) или прочесть (используя переменные среды) значения переменных.

Чтобы разрешить использование серверных расширений, в Apache необходимо убрать комментарии со следующих двух строк в файле **httpd.conf** (или добавить их в соответствующий файл **.htaccess**):

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

Затем опцию **Includes** нужно добавить в блок **<Directory>** или **<Location>** (или файл **.htaccess**), который управляет соответствующей областью Web-сайта:

```
Options +Includes
```

Директива **AddType** добавляет новый класс файлов к типу MIME, расширяя базовый набор, который хранится в файле **/usr/local/etc/apache/mime.types**, а **AddHandler** присваивает определенному расширению соответствующий обработчик (внутренний метод обработки файла, запрашиваемого пользователем). В данном случае используется расширение **shtml** (традиционное расширение HTML-файлов, которые перед обработкой подвергаются синтаксическому анализу).

ПРЕДУПРЕЖДЕНИЕ

Если не считать наличия серверных расширений, единственное различие между **html**- и **shtml**-файлами — это разные расширения. Оба типа являются обычными HTML-файлами. Если в **html**-файл были включены серверные операции, следует изменить его расширение на **.shtml**. Серверные расширения в **html**- или **htm**-файлах обрабатываться не будут.

Вот несколько полезных примеров:

- **<!--#echo var="HTTP_USER_AGENT"-->** — Выводит строку, идентифицирующую браузер пользователя на странице. Подробное обсуждение доступных переменных среды приведено далее в разделе, посвященном CGI-программированию.
- **<!--#set var="e-mail" value="me@somewhere.com"-->** — Устанавливает переменную **e-mail** значением **me@somewhere.com** для оставшейся части страницы.
- **<!--#include virtual="/toolbar.html"-->** — Вставляет на страницу содержимое файла **/toolbar.html**. Серверные расширения обрабатываются рекурсивно, поэтому файл, подключенный таким образом, может содержать и собственные расширения. (Будьте бдительны!)
- **<!--#config timefmt="%A %B %d, %Y"-->** — Устанавливает формат для всех серверных расширений, выводящих дату и время. Строка формата совпадает с той, что используется командой **date**. См. **man date**.
- **<!--#flastmod file="index.shtml"-->** — Выводит дату и время последней модификации файла **index.shtml**. Допустимый формат приведен в предыдущем примере **config timefmt**.
- **<!--#exec cgi="/cgi-bin/counter"-->** — Запускает CGI-сценарий **counter** и отображает его вывод.

Если в **shtml**-файле размещены подобные директивы, но они почему-то не работают, просмотрите исходный код страницы. Эти директивы не должны содержаться в HTML-коде, отправляемом клиенту. Если они там присутствуют, следовательно, они не были обработаны сервером. Скорее всего, конфигурация Apache не настро-

ена для поддержки серверных расширений. Убедитесь, что директивы **AddType** и **AddHandler** заданы корректно и относятся к используемой части Web-сайта.

СОВЕТ

Если имя файла не указано, то по умолчанию клиенту отображается файл **index.html**. Директива **DirectoryIndex** позволяет изменить или расширить список значений:

```
DirectoryIndex index.php index.php3 index.html index.shtml
```

Apache пытается найти любой из файлов в порядке перечисления и, если ни один из них не найден (а директива **Options Indexes** включена), отображает листинг файлов каталога.

Вместо этого можно воспользоваться следующими директивами:

```
Options +MultiViews
DirectoryIndex index INDEX
```

Директива **MultiViews** сообщает Apache о поиске файла, имя которого (без расширения) совпадает с указанным, а **DirectoryIndex** сообщает, что обслуживанию подлежит файл вида **index.*** (с любым расширением). Обычно поиск файлов происходит в алфавитном порядке. Таким образом, сервер будет обрабатывать файлы **index.htm**, **index.HTM**, **INDEX.HTML** и т.д.

Полное руководство по серверным расширениям, где изложено все: от условного управления потоком выполнения до запуска внешних CGI-программ, — можно найти на сайте **http://httpd.apache.org/docs/howto/ssi.html**.

Основы CGI

Обычные Web-сайты состоят из статических HTML-страниц. Расширить их функциональность позволяют серверные расширения. Однако реальные Web-приложения (коммерческие сайты, доски объявлений, базы данных и многие другие) требуют определенной программной среды для обработки пользовательского ввода и управления запрашиваемым выводом. Стандартом этой среды служит CGI, Common Gateway Interface (Общий шлюзовый интерфейс).

CGI представляет собой протокол-посредник, своего рода слой Web-сервера, который позволяет получать данные от пользователя в виде HTML-форм и передавать их в стандартном формате любой программе на сервере — независимо от того, на каком языке она написана. CGI-программа может быть сценарием на языке командного интерпретатора или Perl, скомпилированным двоичным файлом, написанным на C, или любым другим файлом, который может быть выполнен пользователем сервера Apache (**nobody**). Вывод программы Apache перенаправляет прямо Web-браузеру. Это значит, что можно написать CGI-программу, которая читает переменные из HTML-формы, обрабатывает их, открывает конвейер с программой Sendmail, высылает значения переменных вам (администратору) и выдает ответ пользователю в HTML-формате. Далее мы увидим, как решить подобную задачу с помощью сценария на Perl.

Включение CGI в Apache

Существует два способа использования CGI-программ в Apache. Первый, наиболее корректный, состоит во включении в **httpd.conf** директивы **ScriptAlias**, определяющей некоторый каталог как содержащий только CGI-программы и отображающей виртуальный путь к нему (видимый Web-браузеру):

```
ScriptAlias /cgi-bin/ "/usr/local/www/cgi-bin/"
```

Эта строка, которая в файле **httpd.conf** включена по умолчанию, сообщает Apache, что **cgi-bin** — это каталог, предназначенный для CGI, и все файлы, находящиеся в нем, являются CGI-программами. Если в нем разместить что-либо другое и попробовать к нему обратиться, сервер возвратит ошибку с кодом **500** (Server Error), т.е. ошибку при запуске CGI. Количество директив **ScriptAlias** может быть любым. Путь к CGI-каталогу в файловой системе (например, **/usr/local/www/cgi-bin/**) совсем не должен быть доступным для Web-запросов. Псевдоним может указывать на любую точку файловой системы, доступную для чтения пользователю **nobody**. Это предотвращает возможность просмотра содержимого каталога посетителями Web-сайта. К тому же, они никогда не работают с CGI-программами напрямую. Последние вызываются как ссылки или для обработки форм.

ПРИМЕЧАНИЕ

Обратите внимание, что в пути **/cgi-bin/** указан замыкающий символ косой черты. Это еще одна предосторожность, которая предотвращает неавторизованный доступ к листингу каталога. Имя CGI-программы в Apache образуется путем добавления непосредственно имени программы (например, **script.cgi**) к виртуальному пути директивы **ScriptAlias** (например, **/cgi-bin/**), поэтому символ косой черты необходим для правильного воссоздания пути (в данном случае, **/cgi-bin/script.cgi**). Перенаправления с кодом **301** здесь не поддерживаются.

Другой способ включения CGI-программ состоит в использовании директивы **Options** для добавления опции **ExecCGI** к области сервера, заданной в блоке **<Directory>** или **<Location>**. Это полезно для разрешения запуска CGI-программ в пользовательских каталогах **public_html**. При этом используется расширение файлов (например, **.cgi**) и бит права на выполнение. В следующем примере разрешен запуск всех выполняемых пользовательских **cgi**-файлов, независимо от их местоположения:

```
<Directory /home/*/public_html>
   Options +ExecCGI
   AddHandler cgi-script .cgi
</Directory>
```

Если CGI-файл (находящийся в каталоге **ScriptAlias** или назначаемый обработчику по расширению) по каким-либо причинам не может быть запущен, пользователю пересылается сообщение с кодом **500** (Server Error). Более подробные диагностические сообщения содержатся в log-файле Apache. Вот пример записи об ошибке при неудачном запросе сценария **blah** на языке Perl в каталоге **/usr/local/www/cgi-bin**:

```
# tail /var/log/httpd-error.log
syntax error at /usr/local/www/cgi-bin/blah line 3, at EOF
Execution of /usr/local/www/cgi-bin/blah aborted due to compilation
errors.
[Tue May 22 22:06:26 2001] [error] [client 64.2.43.44] Premature end of
script headers: /usr/local/www/cgi-bin/blah
```

Две первых строки вывода принадлежат интерпретатору Perl. Они имеют тот же вид, как если бы сценарий был запущен из командной строки. В третьей строке сервер Apache сообщает о попытке запуска сценария, который завершил работу до того, как вывести какой-либо HTTP-заголовок, например, **Content-type:** (является обязательным для корректного CGI-сценария). Здесь следует убедиться, что программа, предназначенная для выполнения через CGI-интерфейс, написана корректно.

Написание CGI-программ

Формат, в котором CGI-переменные пересылаются серверной программе, представляет собой закодированную в URL текстовую строку, в которой переменные и значения разделяются знаком равенства (=), а сами переменные — символом амперсанта (&). Строка поступает на стандартный ввод сценария (**STDIN**).

Самым распространенным языком CGI-программирования является Perl, поэтому два этих термина ошибочно используются как синонимы. Их не следует путать: язык Perl, кроме Web-программирования, предназначен для решения многих других задач, а CGI обеспечивает интерфейс ко всем языкам, в том числе и к не существующим на данный момент. В силу превалирования Perl в CGI-программировании, мы рассмотрим пример сценария на этом языке.

Основными достоинствами Perl (с которыми мы познакомились в главе 21) являются обработка текста и простота разработки кода. Поэтому он и является идеальным кандидатом для чтения переменных HTML-формы (имя пользователя, адрес электронной почты, почтовый адрес, комментарии и т.д.) и преобразования их в формат, пригодный для дальнейшего использования. Обычно CGI-программы на Perl вызываются в результате обработки HTML-форм:

```
<FORM NAME="myform" METHOD="POST" ACTION="/cgi-bin/post2me">
```

Когда пользователь отсылает форму, все переменные передаются с помощью CGI-интерфейса сценарию **post2me**. Последний представляет собой программу на языке Perl, чьей задачей является их обработка. Первое, что делает сценарий, — это чтение переменных из стандартного входного потока и запись в ассоциативный массив (для упрощения дальнейшей работы с ними):

```
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+/ /;
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
        $value =~ s/~!/ ~!/g;
        $FORM{$name} = $value;
}
```

Вначале производится проверка запроса, на предмет безопасности. Она предотвращает пересылку программе злонамеренного ввода. В частности, если CGI-программа выводит пользователю HTML-файл на основе данных, введенных в форму, пользователь может включить определенный HTML-код, который, вызывая серверные расширения, сможет запустить определенную программу на сервере. Поскольку все программы по http-запросам запускаются от имени пользователя **nobody** (не обладающего необходимыми правами доступа), это не вызывает большой тревоги. Как бы там ни было, это в какой-то мере дыра в защите, забывать о которой нельзя. Поэтому блок кода запрещает потенциально опасные HTML-тэги, вставляя в соответствующих местах пробелы и дефисы.

Когда сценарий прочитает весь ввод, каждая переменная формы становится доступна как ключ ассоциативного массива **%FORM**. Например, содержимое поля **e-mail** доступно теперь как значение **$FORM{'e-mail'}**.

Теперь сценарий должен вывести корректный HTML-заголовок. Здесь существует две возможности: вывести HTML-код в стандартный выходной поток, создавая новую страницу прямо из сценария, или перенаправить пользователя на другой URL, пока сценарий выполняет свою работу. Первый способ осуществляется с помощью заголовка **Content-type:**, который может включать любой корректный тип MIME (браузер самостоятельно решает, как обрабатывать его), за которым следуют два символа новой строки, сигнализирующие об окончании блока заголовка:

```
print "Content-type: text/html\n\n";
```

Все, что выведено сценарием после заголовка, браузер будет визуализовать как HTML-код. Чтобы отобразить обычный текст, можно воспользоваться типом **text/plain**.

Второй метод (перенаправление) выполняется с помощью заголовка **Location:** и URL-перенаправления:

```
print "Location: http://www.somewhereelse.com/path/to/file.html\n\n";
```

Все, что будет выведено после этого заголовка, браузер уже не воспримет, поскольку к этому моменту он перейдет к новому URL.

CGI-программам доступны также и переменные среды, а HTTP-соединение несет немало интересной информации. Перечислим некоторые из них: **HTTP_REFERER** (URL ссылающегося документа), **HTTP_USER_AGENT** (браузер пользователя), **REMOTE_HOST** (имя хоста пользователя) и т.д. Просмотреть все значения позволяет сценарий **printenv**, включенный в инсталляцию Apache по умолчанию (он находится в каталоге **/usr/local/www/cgi-bin**). Для доступа к нему достаточно указать URL **http://www.somewhere.com/cgi-bin/printenv**, подставив, естественно, имя хоста или IP-адрес FreeBSD-машины. В Perl к переменным среды можно обратиться как к ключам ассоциативного массива **%ENV**, например, **$ENV{ 'REMOTE_HOST'}**.

Рассмотрим пример несложной CGI-программы на Perl, которая читает из HTML-формы три переменных (**name**, **e-mail** и **comments**), высылает их администратору по электронной почте и выдает сообщение пользователю. Этот сценарий, приведенный в листинге 26.1, содержится на прилагаемом компакт-диске под именем **sendcomments.cgi**.

Листинг 26.1 Пример CGI-программы на языке Perl

```
#!/usr/bin/perl

read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
      ($name, $value) = split(/=/, $pair);
      $value =~ tr/+/ /;
      $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
      $value =~ s/~!/ ~!/g;
      $FORM{$name} = $value;
}

print "Content-type: text/html\n\n";

open (MAIL,"| /usr/sbin/sendmail -oi -t");
print MAIL "From: $FORM{'name'} <$FORM{'e-mail'}>\n";
print MAIL "To: you\@your.hostname.com\n";
```

```
print MAIL "Subject: Form output\n\n";
print MAIL "$FORM{'name'}, from $ENV{'REMOTE_HOST'} ($ENV{'REMOTE_ADDR'}),
has sent you the following comment:\n\n";
print MAIL "$FORM{'comment'}\n";
close (MAIL);

print qq^<HTML>\n<HEAD>\n<TITLE>Thank you!</TITLE></HEAD>\n^;
print qq^<BODY><H3>Thank you!</H3>\nThanks for your comments!</H3>\n</
BODY>\n</HTML>^;
```

Чтобы настроить сценарий для своих нужд, следует заменить фиктивный заголовок **To:** реальным адресом электронной почты (символ @ необходимо обязательно экранировать символом обратной косой черты). Сам по себе сценарий не выполняет никаких особенных действий, но изучив саму методику, вы убедитесь, что принципы, изложенные здесь, применимы ко все серверным программам — от небольших обработчиков форм до сложных баз данных и коммерческих систем.

CGI-программу можно вызывать не только из HTML или методом **POST**. Можно указать URL сценария CGI, если ему не требуется передавать переменные в ассоциативный массив (наподобие **%FORM**, показанного в примере). URL выглядит следующим образом:

```
http://www.somewhere.com/cgi-bin/sysinfo?frank+3
```

По этому URL вызывается программа **sysinfo** из каталога **/cgi-bin/**. Все, что находится после вопросительного знака, называется *строкой запроса* (*query string*). Ее содержимое доступно сценарию как массив **@ARGV**. Аргументы разделяются знаком плюс (+). В программе **sysinfo** строка **frank** будет элементом **$ARGV[0]**, а число **3** — **$ARGV[1]**.

СОВЕТ

Обратиться ко всей строке запроса в Perl позволяет переменная среды **QUERY_STRING**, или **$ENV{ 'QUERY_STRING'}**.

27 ГЛАВА

Конфигурирование FTP-сервера

- Общие сведения о FTP
- Обзор структуры каталогов FTP
- Настройка FTP-сервера
- Управление FTP-доступом
- Разрешение анонимного доступа по FTP
- Виртуальный хостинг
- Использование альтернативных FTP-серверов

В состав FreeBSD входит FTP-сервер, который при желании можно заменить другим, более пригодным для ваших нужд. Встроенный сервер достаточно сложен и обладает хорошей защитой. Он позволяет пересылать файлы на FreeBSD-машину или с нее, не требуя никаких дополнительных настроек. Конфигурация FTP-сервера, применяемая по умолчанию, достаточно проста. Чтобы воспользоваться дополнительными преимуществами сервера, необходимо знать некоторые тонкости работы протокола FTP. К ним мы сейчас и перейдем.

Общие сведения о FTP

Протоколы FTP и HTTP могут показаться очень похожими. Оба этих протокола позволяют пересылать файлы, поддерживают аутентификацию пользователей и широко используются при работе в Сети: двоичные и мультимедийные файлы можно загрузить с Web-страниц как по протоколу **http://**, так и по **ftp://**. На самом деле эти протоколы были разработаны с разной целью, поэтому наборы поддерживаемых ими свойств значительно отличаются. В табл. 27.1 показаны основные различия в этих протоколах.

Таблица 27.1 Сравнение функциональности протоколов FTP и HTTP

Свойство	*FTP*	*HTTP*
Встроенная аутентификация пользователей	Да	Нет
Изначально разработан для пересылки	Больших двоичных файлов	Небольших текстовых файлов
Модель соединения	Двустороннее (Dual connection)	Одностороннее (Single connection)
Предназначен в первую очередь для загрузки/выгрузки файлов	Оба действия	Загрузки
Поддерживает тип содержимого (заголовки MIME)	Нет	Да
Поддерживает операции с файловой системой (**mkdir**, **rm**, **rename** и т.д.)	Да	Нет

Наибольшее различие между FTP и HTTP заключается в следующем: FTP требует установления сеанса связи. Иначе говоря, между клиентом и сервером устанавливается полноценное двустороннее соединение, команды пересылаются в обе стороны и в конце клиент прерывает сеанс со своей стороны. (Вспомните, что HTTP — это протокол, который не поддерживает сеансы, т.е. одиночный запрос, за которым следует одиночный или "конвейерный" ответ.) FTP не ограничивается одним соединением. Фактически, это протокол с двумя соединениями: одно предназначено для двусторонней пересылки команд и статусных сообщений (control connection — управляющее соединение), а второе — непосредственно для пересылки данных (data connection — соединение для передачи данных). Схема FTP-соединения представлена на рис. 27.1.

Рисунок 27.1
Сеанс FTP с установленным управляющим соединение и соединением для передачи данных.

FTP-клиент (команда **ftp**, включенная в состав FreeBSD) открывает управляющее соединение с FTP-сервером на 21-ом порту. Соединение остается открытым на протяжении всего сеанса. Когда пользователь вводит команду (например, **ls** или **get picture1.gif**), клиент и сервер договариваются о паре TCP-портов, между которыми открывается соединение для передачи данных, которое существует только на время передачи листинга или файла, а затем закрывается. Отдельное соединение для передачи данных открывается при каждой пересылке. Полный сеанс, типичный для работы с FTP, приведен в листинге 27.1.

Листинг 27.1 FTP-сеанс из командной строки

```
# ftp spots.somewhere.com
Connected to spots.somewhere.com.
220 spots.somewhere.com FTP server (Version 6.00LS) ready.
Name (spots.somewhere.com:frank):
331 Password required for frank.
Password:
230 User frank logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd mydir
250 CWD command successful.
ftp> ls
150 Opening ASCII mode data connection for '/bin/ls'.
total 484
-rw-r--r--  1 frank   frank     43175 Apr   8 01:14 addresses.txt
-rw-r--r--  1 frank   frank    100523 Apr   8 01:14 contents.html
-rw-r--r--  1 frank   frank     37864 Apr   8 01:14 directions
-rw-r--r--  1 frank   frank     37308 Apr   8 01:14 lk_logo.gif
-rw-r--r--  1 frank   frank     52427 Apr   8 01:12 picture1.gif
-rw-r--r--  1 frank   frank     18648 Apr  24 13:04 picture2.jpg
-rw-r--r--  1 frank   frank    175325 Apr   8 01:14 resume.html
226 Transfer complete.
ftp> get picture1.gif
local: picture1.gif remote: picture1.gif
150 Opening BINARY mode data connection for 'picture1.gif' (52427 bytes).
100% |***********************************************| 52427      00:00 ETA
226 Transfer complete.
52427 bytes received in 4.99 seconds (10.25 KB/s)
ftp> quit
221 Goodbye.
```

Первая часть соединения проводит аутентификацию имени пользователя и пароля так же, как это происходит при Telnet-соединении. В отличие от аутентификации пользователя HTTP, которая, фактически, входит в функции Apache и не является частью протокола HTTP, аутентификация пользователей FTP использует информацию об учетных записях пользователей на машине, работающей как FTP-сервер. В

этом смысле FTP-соединение очень похоже на Telnet. Чтобы зарегистрироваться в системе по протоколу FTP, пользователю необходимо или ввести корректную информацию о своей учетной записи (имя и пароль) или же воспользоваться анонимным FTP (anonymous FTP), если сервер допускает его использование.

FTP поддерживает набор команд, во многом похожих на команды интерпретатора: **ls**, **cd**, **mkdir**, **pwd** и т.д. Эти команды помогают перемещаться по структуре каталогов и изменять удаленные файлы так же, как если бы они находились на локальной машине. Для перехода к каталогу на локальной машине используется команда **lcd**. Она полезна тогда, когда файл необходимо загрузить не в тот каталог, из которого изначально был запущен FTP-клиент. Пересылкой файлов управляют команды **put** (загрузить на сервер), **get** (загрузить с сервера), **mput** (загрузить на сервер несколько файлов) и **mget** (загрузить с сервера несколько файлов). Эти пользовательские команды переводятся в команды клиента (например, **RETR**, **STOR**, **CWD** и **LIST**), которые понимает FTP-сервер, отвечающий с помощью трехзначных кодов наподобие тех, что применяются в HTTP. Значения кодов ответа знать не обязательно. Полный список клиентских команд FTP приведен на странице справочного руководства **man ftpd**.

ПРИМЕЧАНИЕ

Пересылка файлов по FTP осуществляется в одном из двух режимов: ASCII (обычный текст, где все данные пересылаются как буквенно-цифровые символы, а символы конца строки переводятся в соответствии с клиентской платформой, т.е. CR/LF для DOS/Windows, CR для Macintosh и LF для UNIX); или Binary (двоичный) (поток двоичных данных). Некоторые FTP-клиенты автоматически определяют, какой режим является подходящим, и переключаются в него до начала пересылки. Другие клиенты требуют явного выбора режима ASCII или Binary до пересылки файлов. Для переключения режимов используются команды **bin** и **asc**.

Очень важно, особенно при пересылке с одной платформы на другую таких файлов, как HTML-страницы или сценарии на языке Perl, использовать режим ASCII. В этом случае символы конца строки будут преобразовываться правильно. Двоичный режим приведет к тому, что сценарии Perl, загруженные на сервер с не-UNIX-клиента, просто не будут выполняться. Однако для изображений, выполняемых файлов и других типов двоичных данных этот режим является обязательным. Пересылка двоичных файлов в режиме ASCII приведет к их искажению.

FTP-сервером, используемым во FreeBSD по умолчанию, является стандартный BSD-демон **ftpd**. Он запускается из суперсервера **inetd**, как **telnetd** и **Qpopper**, и не может работать как отдельный демон. И дело не только в том, что ему недостает нескольких свойств, присущих его альтернативам (например, **WU-FTPD** или **ProFTPD**). Основная причина — многочисленные дыры в защите, неизбежные в сложном программном обеспечении. Альтернативные демоны рассматриваются в заключительной части этой главы. Здесь же мы рассмотрим стандартный FTP-сервер FreeBSD.

Обзор структуры каталогов FTP

Если не разрешена анонимная регистрация по FTP, схема расположения файлов на FTP-сервере достаточно проста и интегрирована в систему, как и большинство основных служб. В каталоге **/etc** находится несколько конфигурационных файлов, часть из которых используется как ресурсы, необходимые и другим службам систем-

ного уровня. Начальные каталоги пользователей также являются частью схемы расположения файлов FTP-сервера, поскольку каждый зарегистрировавшийся пользователь попадает непосредственно в свой каталог.

Если разрешен анонимный доступ к FTP, то существует корневой каталог сервера FTP (как и для Apache), который создается на этапе конфигурирования. По умолчанию он расположен в **/var/ftp** и включает несколько подкаталогов, которые позволяют регистрироваться пользователям, не имеющим учетных записей в системе.

Аутентифицированный и анонимный FTP-доступ

Когда пользователь, имеющий в системе учетную запись, регистрируется в системе по FTP с корректным именем и паролем, сервер предоставляет ему доступ к начальному каталогу и всем файлам в нем. Для проверки можно воспользоваться командой **ls**. Таким образом, каждый пользователь попадает в свою точку иерархии FTP-сервера. Анонимный FTP-доступ позволяет регистрироваться в системе пользователям, не имеющим учетных записей. Пользователь такого типа открывает соединение, вводит в качестве имени **anonymous** (или **ftp**) и любую текстовую строку (обычно адрес своей электронной почты, хотя это и не обязательно) в качестве пароля. В этом случае он попадает в "общедоступную" область FTP: **/var/ftp**, начальный каталог пользователя **ftp** (этого пользователя также требуется создать при разрешении анонимного доступа).

Между обычными пользователями и зарегистрировавшимися анонимно существует фундаментальное различие. Для анонимных пользователей с помощью утилиты **chroot** установлено, что корневым каталогом сервера является каталог **/var/ftp**. Поэтому им недоступно (даже для просмотра) все, что находится за пределами **/var/ftp**. Пользователь, зарегистрировавшийся под своей учетной записью, может с помощью команды **cd /usr/local** перейти к любой части системы и обратиться к файлам с теми же правами доступа, что и в терминальной сессии. Анонимный пользователь, введя команду **cd /pub**, на самом деле попадет в **/var/ftp/pub**.

Обычных пользователей, для которых также следует заменить корневой каталог, можно добавить в файл **/etc/ftpchroot**.

Настройка FTP-сервера

Службы FTP требуют достаточно большого числа конфигурационных файлов, часть из которых находится в каталоге **/etc**, а часть — в **/var/ftp/etc**. Причина такого деления заключается в том, что часть файлов должна быть доступна анонимным пользователям FTP, которым недоступно ничего за пределами иерархии **/var/ftp**. Рассмотрим несколько файлов из каталога **/etc**, которые глобально влияют на работу FTP-сервера.

- **/etc/ftpusers**. "Черный список" пользователей, которым запрещен доступ по FTP. Для предотвращения регистрации пользователей по FTP достаточно добавить их имена в этот файл.
- **/etc/ftpchroot**. Пользователи, для которых изменен корневой каталог (наподобие того, как это сделано для анонимных пользователей), в результате чего им доступен лишь их начальный каталог.

- **/etc/ftphosts**. Позволяет конфигурировать виртуальные хосты, как это делалось для Apache (см. главу 26).
- **/etc/ftpwelcome**. Приветственное сообщение. Содержимое этого файла отображается каждому пользователю, устанавливающему соединение, до приглашения ввода имени и пароля.
- **/etc/ftpmotd**. Второе приветственное сообщение (Message Of The Day — "Сообщение дня"), которое отображается обычным пользователям после регистрации в системе.
- **/etc/shells**. Мы сталкивались с этим файлом в главе 12. Его задачей является проверка того, что каждый пользователь, регистрирующийся в системе, имеет доступ к корректному командному интерпретатору. Это предотвращает регистрацию пользователей посредством таких учетных записей, как **bin**, **tty** и **nobody**, для которых в этом файле командные интерпретаторы не указаны.

Кроме конфигурационных файлов в каталоге **/etc**, существует набор дополнительных файлов, управляющих анонимным FTP-доступом. Однако они включают в себя не только конфигурационные файлы. Поскольку для анонимных пользователей **/var/ftp** представляет собой корневой каталог /, им недоступны средства из таких каталогов, как **/bin**, или файлы из **/etc**. FTP-сервер использует несколько системных утилит, в частности **/bin/ls** и **/bin/date** для создания листинга файлов и отправки его клиенту. Эти средства должны быть доступны и анонимным пользователям, поэтому и существует каталог **/var/ftp/bin**.

Дерево **/var/ftp** содержит приведенные ниже файлы и каталоги. Любой анонимный пользователь видит эти файлы, но они не являются "опасными" (так, например, файлы **/var/ftp/etc** не содержат паролей).

- **/var/ftp/bin**. Этот каталог содержит выполняемые файлы **ls** и **date**. Они необходимы FTP-серверу, чтобы генерировать листинги, а системные программы (при правильной настройке сервера) **/bin/ls** и **/bin/date** недоступны. Такое поведение применяется по умолчанию.
- **/var/ftp/etc/passwd**, **/var/ftp/etc/group**. Как и утилиты в **/var/ftp/bin**, эти файлы являются копиями файлов по умолчанию **/etc/passwd** и **/etc/group**. Их задача — преобразование идентификаторов в имена пользователей и групп при выводе листингов каталогов. Поскольку анонимный FTP-доступ ограничен каталогом **/var/ftp**, наличие этих файлов необходимо для отображения прав владения различными файлами в **/var/ftp**. Обычно файлы в общедоступной области принадлежат пользователю **root** или другой системной учетной записи, поскольку их, как правило, размещает администратор. Имена других пользователей, поскольку они отсутствуют в **/var/ftp/etc/passwd**, не будут отображаться, в результате чего анонимным пользователям будут выводиться только числовые идентификаторы.
- **/var/ftp/etc/ftpmotd**. Этот файл аналогичен **/etc/ftpmotd**, но отображается он анонимным пользователям. С их точки зрения, это и есть файл **/etc/ftpmotd**.
- **/var/ftp/pub**. Видимый анонимным пользователям как **/pub**, этот каталог содержит все загружаемые файлы. Иерархия внутри **/pub** оставлена на усмотрение системного администратора.

- **/var/ftp/incoming**. Этот необязательный каталог имеет права на запись с установленным битом устойчивости (sticky bit) (**1777**). Это значит, что анонимные пользователи могут загружать в него файлы. Помните, что это может быть опасным: этим каталогом могут воспользоваться для обмена пиратским программным обеспечением или файлами MP3. Используйте эту опцию лишь в случае крайней необходимости!

Заключительный конфигурационный файл, связанный с конфигурацией FTP-сервера называется **/etc/inetd.conf**. Как отмечалось ранее, демон **ftpd** запускается из суперсервера **inetd** и не может работать, если **inetd** не запущен. Вначале необходимо воспользоваться командой **ps** и определить, обслуживает ли демон запросы:

```
# ps -waux | grep inetd
root      1640   0.0   0.6   1048   780   ??   Ss    Thu08PM    0:00.15
inetd -Ww
```

Если среди выполняемых процессов **inetd** отсутствует, возможно, он отключен в файле **/etc/rc.conf**. Поищите строку **inetd_enable="NO"** и уберите ее. Затем откройте файл **/etc/inetd.conf** и убедитесь, что служба **ftpd** разрешена:

```
ftp     stream   tcp        nowait   root      /usr/libexec/ftpd      ftpd -l
```

Если эта строка закомментирована, уберите комментарий и перезапустите **inetd** (командой **killall -HUP inetd**). Проверьте конфигурацию, попытавшись соединиться с сервером (**ftp localhost**). Если система выдает приглашение для регистрации, значит все в порядке. Если нет, пройдите предыдущие шаги заново. Если потребуется, перезапустите процесс **inetd**.

Управление FTP-доступом

FTP не принадлежит к тем службам, использование которых стоит разрешать с легкостью. Хотя для пользователей и может оказаться важным иметь доступ для загрузки файлов на сервер (например, Web-страниц), следует помнить, что для вас это потенциальный источник проблем с защитой. В первую очередь потому, что используется текстовый механизм, когда все данные (включая и пароли) пересылаются незашифрованными, а следовательно, доступными для прослушивания с помощью соответствующего программного обеспечения. Как мы увидим в главе 29, большинство текстовых служб заменено их защищенными эквивалентами: Telnet на SSH, HTTP на Secure HTTP, а POP3 и IMAP — аналогичными программами со встроенным шифрованием. Однако FTP изначально незащищен и, хотя было предложено несколько решений (например, **sftp** и **sftpd** Брайана Веллингтона), незащищенный FTP-доступ продолжает широко использоваться, плохо поддается замене и фактически является обязательным требованием к полнофункциональному серверу. О том, как защитить его, рассказано в главе 29. Кроме того, при разрешении доступа по FTP необходимо принять специальные меры предосторожности.

Помня об этом, в первую очередь нужно блокировать возможность подключения по FTP определенных пользователей. Добиться этого можно несколькими способами. Два наиболее удобных предполагают использование файлов **/etc/ftpusers** и **/etc/shells**. Третий способ, **/var/run/nologin**, глобально управляет тем, принимает ли сервер соединения.

Файл /etc/ftpusers

Простейший способ запретить отдельному пользователю или целой группе возможность соединения с FTP-сервером — добавить его имя в файл **/etc/ftpusers**, который имеется в инсталляции FreeBSD по умолчанию и включает имена системных псевдопользователей (**operator**, **bin**, **tty** и т.д.). Эти пользователи имеют пустые пароли, а **ftpd** не позволяет никому соединиться с ним, используя пустой пароль. Добавление имен в файл **/etc/ftpusers** обеспечивает дополнительный уровень защиты.

В этот файл можно добавить любые нужные имена. Поскольку **ftpd** перечитывает все конфигурационные файлы при открытии каждого нового соединения, перезапускать какие-либо процессы нет необходимости. Попытайтесь соединиться с FTP-сервером как запрещенный пользователь, вы должны получить следующий ответ:

```
# ftp localhost
Connected to localhost.somewhere.com.
220 stripes.somewhere.com FTP server (Version 6.00LS) ready.
Name (localhost:frank):
530 User frank access denied.
ftp: Login failed.
ftp>
```

ПРИМЕЧАНИЕ

Обратите внимание, что сообщение **access denied** (в доступе отказано) появляется сразу после ввода имени пользователя — пароль даже не запрашивается. Это предотвращает пересылку паролей по сети и является дополнительной мерой предосторожности на случай, если кто-то прослушивает сеть.

В файл **/etc/ftpusers** можно добавлять и группы. Перед их именем следует указать символ **@**, например, **@users**. Если пользователь является членом этой группы, ему будет отказано в доступе.

Файл /etc/shells

После сверки имени пользователя с файлом **/etc/ftpusers** демон **ftpd** проверяет, какой командный интерпретатор назначен пользователю и присутствует ли он в файле **/etc/shells**. Если нет, пользователь опять-таки получит сообщение **access denied**. Предотвратить регистрацию в системе с помощью терминальной программы или FTP можно, заменив начальный командный интерпретатор пользователя на **/sbin/nologin**. Эта утилита рассматривалась в главе 12. Она просто выводит сообщение **account not available** (учетная запись недоступна) и прекращает работу.

Файл /var/run/nologin

Для полного отключения регистрации по FTP без изменения файла **/etc/inetd.conf** или других конфигурационных файлов нужно разместить файл с названием **nologin** в каталоге **/var/run**. Если **ftpd** находит этот файл, он отвечает на все попытки соединения следующим образом:

```
# ftp localhost
Connected to localhost.somewhere.com.
530 System not available.
ftp>
```

Для создания файла (нулевой длины) можно воспользоваться командой **touch /var/run/nologin**. Чтобы разрешить работу FTP-сервера, достаточно просто удалить этот файл (**rm /var/run/nologin**).

Разрешение анонимного доступа по FTP

По умолчанию анонимный доступ по FTP запрещен. Разрешить его можно с помощью утилиты **sysinstall**. Выполните команду **/stand/sysinstall**, затем перейдите к разделу Configure и Networking. Перейдите к опции Anon FTP и нажмите клавишу пробел. На экране появится меню Anonymous FTP Configuration, показанное на рис. 27.2.

Опции по умолчанию, как правило, подходят для большинства FreeBSD-систем. Поля **UID**, **Group** и **Comment** управляют тем, как создается новый пользователь **ftp**. Анонимный доступ по FTP работает по следующей схеме: пользователь **ftp** трактуется как обычный пользователь, добавленный к файлу **/etc/ftpchroot**, в результате чего регистрация с учетной записью **ftp** (или ее псевдонимом **anonymous**) направляет пользователя в каталог **/var/ftp**.

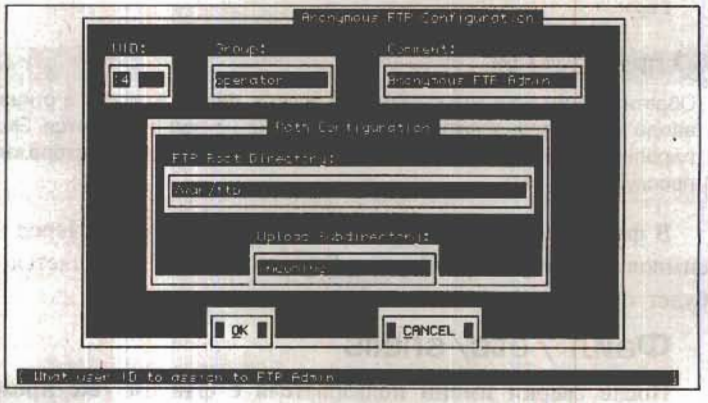


Рисунок 27.2
Опции конфигурации анонимного доступа по FTP.

Остальные поля можно изменить так, как требуется в конкретной системе (например, если UID, равный 14, уже занят или для загрузки файлов на сервер используется не **incoming**, а другой каталог). В результате будет создан пользователь **ftp** и набор всех нужных подкаталогов в дереве **/var/ftp**.

Утилита **/stand/sysinstall** не позволяет запретить анонимный доступ по FTP после того, как он был включен, однако для этого существует несколько других способов:

- Удалить дерево **/var/ftp**.
- Удалить пользователя **ftp**.
- Добавить пользователя **ftp** в файл **/etc/ftpusers** (вероятно, это самый простой и корректный метод).

Аналогично, загрузку файлов в каталог **incoming** можно запретить, удалив его или установив права доступа **755** (права доступа по умолчанию, когда записывать что-либо в каталог может только его владелец — пользователь **root**). Заново разрешить ее можно, изменив права доступа на **1777**: **chmod 1777 /var/ftp/incoming**.

Виртуальный хостинг

Если к машине "привязано" несколько IP-адресов, каждый из них можно сделать отдельным FTP-сервером. Механизмом виртуального хостинга управляет файл **/etc/ftphosts**, который не существует в системе FreeBSD по умолчанию (его необходимо создать).

Каждый виртуальный хост определен в своей строке, причем поля указывают на альтернативные конфигурационные файлы каждого хоста, разделенные пустым символом. Различные поля, их назначение и установки по умолчанию (используемые сервером в отсутствие файла **/etc/ftphosts**) описаны в табл. 27.2.

Таблица 27.2 Поля таблицы виртуальных хостов в файле /etc/ftphosts

Поле	*Описание*	*Значение по умолчанию*
Hostname	Имя хоста или IP-адрес виртуального хоста. Отметьте, что FTP не имеет эквивалента заголовку **Host:**, используемому в HTTP/1.1, поэтому виртуальные хосты FTP определяются только по IP-адресу машины, к которой подключен клиент. в этом поле указано имя хоста, **ftpd** все равно использует Даже если соответствующий ему IP-адрес. Помните об этом при добавлении виртуальных хостов.	Нет
User	Пользователь, начальный каталог которого используется для анонимного доступа по FTP к виртуальному хосту. Анонимные пользователи попадают в этот каталог как в корневой, поэтому в нем должны присутствовать эквиваленты подкаталогов **/var/ftp/etc** и **/var/ftp/bin**.	ftp
Statfile	Файл статистики, в котором содержится информация обо всех пересылках по FTP для виртуального хоста.	/var/log/ftpd
Welcome	Приветственное сообщение, отображаемое при попытке установления соединения с FTP-сервером.	/etc/ftpwelcome
MOTD	Сообщение дня, отображаемое после успешной регистрации.	/etc/ftpmotd

Ниже приведено несколько примеров виртуальных хостов в файле **/etc/ftphosts**. Если поле оставлено пустым или содержит дефис (**-**), используется значение по умолчанию.

```
64.41.131.106       frank   /var/log/ftpd-frank    /home/frank/welcome
/home/frank/welcome2
ftp2.somewhere.com  ftp2    -                      /etc/ftpd2welcome    -
64.41.131.107       ftp3    /var/log/ftpd-3        -                    -
```

Использование альтернативных FTP-серверов

Существует множество альтернативных пакетов FTP-серверов. Два наиболее популярных из них: WU-FTP Вашингтонского университета и сервер ProFTPD с широкими возможностями конфигурирования.

WU-FTPD

Изначально разработанный в Вашингтонском университете (Washington University) для использования с одной из самых популярных служб распространения дистрибутивов WUarchive, WU-FTPD приобрел широкую популярность и стал самым часто используемым FTP-сервером в мире. Он используется по умолчанию в Linux и многих коммерческих версиях UNIX. Его конфигурация несколько отличается от демона **ftpd**, включенного по умолчанию во FreeBSD. Он имеет несколько свойств, отсутствующих у **ftpd** (например, средство проверки конфигурации, возможность оперативного сжатия и упаковки, а также ограничения доступа или объема пересылки по дате и времени). WU-FTPD может понадобиться для поддержки совместимости с другими системами, отличными от FreeBSD.

WU-FTPD распространяется в виде пакета и порта (**/usr/ports/ftp/wu-ftpd**). После инсталляции переключиться на него с демона **ftpd** можно, закомментировав строку **ftp** в файле **/etc/inetd.conf** и заменив ее на другую:

```
ftp     stream   tcp     nowait   root     /usr/local/libexec/ftpd   ftpd -l
#ftp    stream   tcp     nowait   root     /usr/libexec/ftpd         ftpd -l
```

СОВЕТ

WU-FTPD устанавливается как **/usr/local/libexec/ftpd**, а страницу его справочного руководства нельзя получить напрямую, так как команда **man ftpd** выдает страницу демона по умолчанию. Для просмотра нужной страницы следует воспользоваться опцией **-M**, указывающей альтернативный путь к **man**-странице: **man -M /usr/local/man ftpd**

Более подробную информацию о WU-FTPD можно найти на Web-сайте **http://www.wu-ftpd.org**.

ProFTPD

ProFTPD был разработан с целью создать FTP-сервер, которым можно было бы управлять с помощью конфигурационных файлов, повторяющих структуру Apache. Конфигурационный файл сервера содержит иерархические блоки, похожие на содержимое файла **httpd.conf** (с которым мы познакомились в главе 26), а директивы по своему стилю похожи на директивы Apache. В результате получился сервер, похожий на Apache, с большими возможностями по ограничению доступа. Он обладает высокой степенью конфигурируемости и особенно нравится тем администраторам, которые знакомы с Apache. Его можно установить как пакет или порт (**/usr/ports/ftp/proftpd**).

Единственное отличие ProFTPD от других серверов состоит в том, что его, как и Apache, можно запускать в самостоятельном режиме, не прибегая к демону **inetd**. Чтобы запустить его из **inetd**, достаточно заменить строку **ftp** по умолчанию новой, указывающей на **/usr/local/libexec/proftpd**:

```
ftp     stream   tcp     nowait   root     /usr/local/libexec/proftpd   proftpd
#ftp    stream   tcp     nowait   root     /usr/libexec/ftpd      ftpd -l
```

Домашняя страница ProFTPD находится по адресу **http://www.proftpd.org** и содержит подробную информацию о его возможностях и директивах конфигурации.

28 ГЛАВА

Конфигурирование шлюза Internet

- Что такое маршрутизатор?
- Конфигурирование шлюза NAT в ОС FreeBSD
- Включение поддержки NAT
- Конфигурирование клиентов для использования нового шлюза
- Конфигурирование беспроводного доступа к Internet
- Маршрутизация между тремя и более сетями
- Динамическая маршрутизация

Маршрутизация и шлюзы рассматривались в главе 22, как и концепция NAT (network address translation — преобразование сетевых адресов). В главе 23 также затрагивались вопросы маршрутизации, но в основном в контексте конфигурирования хоста для использования в качестве маршрутизатора. В этой главе описывается конфигурирование FreeBSD для работы в качестве маршрутизатора или шлюза. Также рассматривается настройка преобразования сетевых адресов. Мы начнем с краткого определения рассматриваемых понятий.

Что такое маршрутизатор?

Как и предполагает название, *маршрутизатор* — это сетевое устройство, определяющее маршрут пересылки дейтаграмм по сети к месту назначения. Это очень простое определение. Говоря точнее, маршрутизатор соединяет две сети и определяет, как дейтаграммы проходят из одной сети в другую.

ПРИМЕЧАНИЕ

Дейтаграмма — это пакет цифровой информации. Кроме передаваемых данных, датаграмма содержит информацию об адресе места назначения. Не все дейтаграммы проходят по одному и тому же маршруту, даже если у них одно место назначения и один источник.

Чтобы лучше понять функции маршрутизатора, рассмотрим два рисунка. На рис. 28.1 представлены две сети компании, офисы которой находятся в Нью-Йорке и Дании.

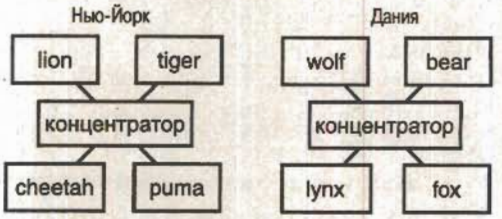


Рисунок 28.1.
Пример глобальной сети компании. Сеть включает несколько хостов в ее Нью-Йоркском офисе и несколько хостов в датском офисе

В сети, представленной на рис 28.1, все хосты, **lion**, **cheetah**, **tiger** и **puma**, "знают" о существовании друг друга, и могут взаимодействовать в Нью-Йоркском офисе. В датском офисе хосты **wolf**, **bear**, **fox** и **lynx** также "знают" о существовании друг друга и могут взаимодействовать. Проблема в том, что эти сети изолированы. Хостам в Нью-Йорке ничего не известно о хостах в датском офисе, и наоборот. Нет способа передачи сетевой информации между этими двумя офисами. Чтобы можно было передавать данные между офисами, необходимо установить пару маршрутизаторов. На рис. 28.2 представлена та же сеть, но теперь в каждом офисе установлен маршрутизатор.

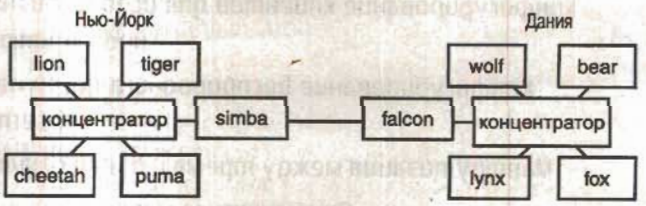


Рисунок 28.2.
Та же сеть, но в офисах установлены маршрутизаторы, связанные друг с другом

После установки маршрутизатора необходимо сделать одно изменение на хостах сети — сообщить каждому из них о существовании маршрутизатора в офисе. (Подробнее о том, как сообщить хосту о существовании маршрутизатора, см. в главе 23.)

Итак, что же изменилось в сети? Хостам в Нью-Йоркском офисе по-прежнему ничего не известно о хостах в датском офисе, а те, в свою очередь, не "знают" о хостах в Нью-Йорке. Но появилось существенное различие. Теперь хосты в каждой из сетей "знают" о существовании в офисе маршрутизатора. Если хосты используют этот маршрутизатор в качестве стандартного, исходящие пакеты, предназначенные неизвестному хосту, будут просто посылаться маршрутизатору. Дальнейшее направление сетевой информации к месту назначения — забота маршрутизатора.

Предположим, например, что хосту **cheetah** необходимо подключиться по сети к хосту **lynx**. Как уже было сказано, хосту **cheetah** не известно о существовании хоста **lynx**. Поэтому при необходимости отправить информацию на хост **lynx** хост **cheetah** просто передает ее на хост **simba**, который является маршрутизатором Нью-Йоркского офиса. Хост **simba** "знает" о существовании хоста **lynx**, а также то, что направить информацию на хост **lynx** можно путем пересылки ее на маршрутизатор в Дании (хост **falcon**). Маршрутизатор в Дании затем направит информацию на хост **lynx**.

Этот пример сильно упрощен. В реальной ситуации (особенно при посылке информации на другой континент), будет задействовано несколько маршрутизаторов, передающих информацию друг другу, пока она не достигнет места назначения. Если провести аналогию, маршрутизаторы можно рассматривать как службы управления воздушным движением, а информацию — как самолеты. Самолет по пути из Нью-Йорка в Данию будет вестись несколькими диспетчерскими службами, пока не приземлится в месте назначения.

Хорошо. Ну, и зачем же маршрутизатор? Почему просто не сообщить хостам в Нью-Йорке о хостах в Дании, и наоборот, чтобы они могли взаимодействовать друг с другом непосредственно? Маршрутизаторы используют по двум причинам.

- **Простота сопровождения.** Эта проблема не кажется существенной при работе с восемью системами. Но представьте себе сеть масштаба Internet. При отсутствии маршрутизаторов каждая подключенная к сети система должна "знать", как связываться с любой системой в сети. Очевидно, что при работе с миллионами систем сопровождение такого решения стало бы настоящим кошмаром.
- **Сокращение нагрузки на каналы связи.** Если бы два офиса были просто соединены одной большой сетью, то передаваемую по сети информацию пришлось бы посылать всем компьютерам. Если, например, хост **lion** посылает информацию хосту **cheetah** (оба хоста — в Нью-Йоркском офисе), она посылалась бы по кабелю через Атлантический океан в датский офис. И это при том, что хостам в Дании эта информация вовсе не нужна, поскольку предназначена не им. Представьте себе нагрузку на каналы связи, если при обмене информацией между двумя хостами по сети Internet, она посылалась бы каждому хосту, подключенному к Internet. Кроме того, трансокеанские выделенные линии достаточно дороги (оплата зависит от количества переданной информации). Это означает, что было бы расточительно использовать линию связи без необходимости. Маршрутизатор действует как "дверь", удерживающая в локальной сети информацию, предназначенную для локальных хостов. Только информация, предназначенная удаленному хосту, будет посылаться за пределы локальной сети. При этом повышается безопасность, поскольку локальная информация не рассылается по Internet.

Что такое шлюз?

В общесетевой терминологии *шлюз* — это маршрутизатор, позволяющий клиентам локальной сети обращаться к внешнему миру, — отсюда и название: "шлюз". Поэтому специалисты по сетям считают термины "стандартный маршрутизатор" и "шлюз" взаимозаменяемыми. В нашем примере из предыдущего раздела маршрутизаторы **simba** и **falcon** можно назвать шлюзами.

Если подходить с технической точки зрения, только что данное определение шлюза неверно. В соответствии с техническим определением, шлюз — это маршрутизатор, который может пересылать пакеты между сетями двух различных типов. Однако этого определения уже практически никто не придерживается; не будем и мы. В этой главе будет использоваться общепринятое определение, а термины "стандартный маршрутизатор" и "шлюз" — как взаимозаменяемые.

Очень часто шлюз применяют, чтобы дать возможность использовать одно подключение к Internet несколькими хостами.

Что такое NAT?

Термин NAT — сокращение от Network Address Translation (*преобразование сетевых адресов*) — обозначает способ подключения к Internet нескольких хостов с использованием одного IP-адреса. Тонкости этого процесса выходят за рамки книги, но по сути процесс NAT организуется путем подключения шлюза NAT к соответствующей сети. Когда внутренние хосты хотят послать или получить информацию из Internet, их запрос проходит через шлюз NAT. Шлюз NAT "скрывает" внутренний IP-адрес и посылает все запросы подключенных к нему хостов в Internet как исходящие от одного IP-адреса (а именно, IP-адреса, принадлежащего шлюзу NAT). Ответы приходят также на один IP-адрес (принадлежащий шлюзу NAT). Шлюз NAT затем направляет данные соответствующему внутреннему хосту, которому вовсе не обязательно иметь зарегистрированный IP-адрес. Этот метод дает два основных преимущества:

- Экономятся IP-адреса. Пул IP-адресов — ограниченный ресурс. И нет причины тратить IP-адреса, когда без этого можно обойтись. Использование NAT позволяет не регистрировать IP-адрес для каждой из имеющихся в систем.
- Для домашних пользователей и небольших организаций этот метод позволяет совместно использовать одно подключение к Internet нескольким компьютерам, т.е. отпадает необходимость покупать дополнительные учетные записи у поставщика услуг Internet. Можно также совместно использовать единственный модем и одну телефонную линию, так что не придется устанавливать дополнительные телефонные линии, если необходим одновременный доступ к Internet нескольким компьютерам.

В этой главе мы рассмотрим несколько способов конфигурирования различных типов маршрутизирующих служб в ОС FreeBSD. Начнем с описания совместного использования одного модема для подключения к Internet нескольких компьютеров дома или в небольшом офисе.

Конфигурирование шлюза NAT в ОС FreeBSD

В этом сценарии обычно имеется единственное подключение к Internet, через модем или непосредственно. Есть несколько систем, которым необходим доступ к Internet. Пример представлен на рис. 28.3.

Разновидность этого варианта — учебный класс, в котором необходимо обеспечить беспроводный доступ к Internet для портативных компьютеров студентов. Эта разновидность конфигурации будет рассмотрена позже, поскольку для этого тоже требуется установка шлюза NAT.

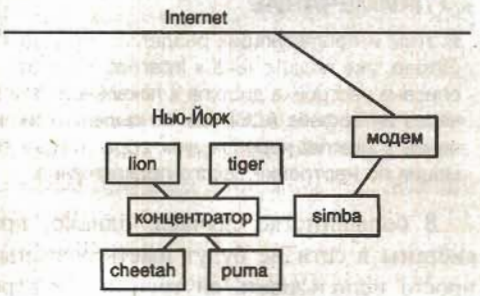


Рисунок 28.3. *Простой вариант шлюза. Хост* **simba** *подключен к Internet через модем. Идея состоит в том, чтобы обеспечить совместное использование этого модема хостами* ***lion***, ***tiger***, ***cheetah*** *и* ***puma***, *а также доступ для них в Internet через хост* ***simba***

В конфигурации такого типа система имеет два сетевых интерфейса. Например, **ppp0** — модемный интерфейс, связывающий компьютер с Internet. Вторым обычно является интерфейс Ethernet (например, **ed0**). Этот интерфейс связывает компьютер с внутренней сетью. Шлюз функционирует как "дверь" между этими двумя интерфейсами, передавая между ними пакеты. Он связывает внутреннюю сеть с Internet. Поступившая из Internet по интерфейсу **ppp0** информация будет передаваться на интерфейс **ed0** для пересылки соответствующему хосту сети. При поступлении информации, адресованной хосту в Internet, от хоста в локальной сети через интерфейс **ed0**, она будет передана интерфейсу **ppp0** для пересылки по сети Internet. Но прежде чем это станет возможным, необходимо включить перенаправление пакетов, чтобы информация могла переходить с одного интерфейса на другой.

Включение перенаправления пакетов

Для функционирования системы в качестве шлюза, необходимо, чтобы она могла перенаправлять пакеты с одного сетевого интерфейса на другой. Система будет обрабатывать входящую и исходящую информацию Internet для других компьютеров сети, так что при поступлении пакета, адресованного не ей, он перенаправляется адресату. По умолчанию (и в соответствии со стандартами Internet) перенаправление пакетов отключено, поэтому ОС FreeBSD будет отбрасывать все получаемые пакеты, не предназначенные для соответствующего хоста.

Перенаправление пакетов можно включить одним из двух способов. Например, с помощью программы **Sysinstall**. Однако проще сделать это вручную, поскольку для включения перенаправления пакетов необходимо добавить всего одну строку в один файл. Мы рассмотрим именно этот метод.

Откройте файл **/etc/rc.conf** в любом текстовом редакторе и добавьте следующую строку:

```
gateway_enable="YES"
```

После перезапуска системы перенаправление пакетов будет включено, и система сможет передавать пакеты с одного сетевого интерфейса на другой.

Если все системы в сети имеют реальные статические IP-адреса, больше ничего делать не надо. Теперь можно конфигурировать другие системы в сети так, чтобы они использовали хост **simba** в качестве шлюза. Всю информацию, которую хосты не

знают, как пересылать адресату, они будут передавать системе, сконфигурированной в качестве стандартного шлюза. О дальнейшей обработке этой информации позаботится стандартный шлюз.

ПРИМЕЧАНИЕ

В этом и последующих разделах предполагается, что система, конфигурируемая в качестве шлюза, уже подключена к Internet. Если это еще не сделано, обратитесь к главе 23, в которой описана настройка доступа к локальной сети (в этой главе также описано подключение к Internet через интерфейс ADSL или по выделенной линии). Если необходимо сконфигурировать подключение к Internet через модем, обратитесь к главе 24, где представлена дополнительная информация по настройке такого подключения.

В большинстве случаев, однако, при конфигурировании шлюза Internet другие системы в сети не будут иметь реальных зарегистрированных IP-адресов. Они будут просто использовать внутренние IP-адреса. Для обеспечения доступа этих систем к Internet необходимо включить преобразование адресов (NAT).

Включение NAT

При использовании протокола PPP для подключения к Internet по коммутируемой линии способ включения преобразования адресов будет зависеть от выбранной реализации протокола — пользовательской (User PPP) или на уровне ядра (Kernel PPP). Если подключение к Internet еще не установлено, я рекомендую использовать в качестве базы для организации NAT реализацию User PPP, поскольку ее проще сконфигурировать для поддержки преобразования адресов.

Если используется реализация PPP на уровне ядра или необходимо настроить преобразование адресов для подключения к Internet не по протоколу PPP, придется применять более сложную процедуру.

Процедуру настройки NAT мы разделим на две части. Можете прочитать только ту часть, которая относится к используемой конфигурации

Использование реализации PPP пользовательского уровня

Программа реализации протокола PPP пользовательского уровня в ОС FreeBSD имеет встроенную возможность трансляции адресов NAT, поэтому включить ее очень легко. Для включения трансляции адресов достаточно указать опцию **-nat** при вызове **ppp**. Просто добавьте ее к остальным опциям, используемым для запуска протокола PPP (подробнее см. в главе 24).

Останется только сконфигурировать клиенты с ОС Windows, Macintosh и др. для использования нового шлюза. Эта задача будет подробно рассмотрена в разделе "Конфигурирование клиентов" этой главы.

Использование реализации PPP на уровне ядра или подключение к Internet по выделенной линии

При использовании реализации протокола PPP на уровне ядра или при наличии подключения к Internet по выделенной линии (ADSL, кабель, T1, OC3) конфигурирование преобразования адресов несколько сложнее. В этом случае придется использовать демон NAT (**natd**), для работы которого требуется брандмауэр. Чтобы включить

поддержку брандмауэра, потребуется построить новое ядро. Это, однако, сделать несложно. Можно просто добавить следующие две строки в файл конфигурации ядра:

```
options  IPFIREWALL
options   IPDIVERT
```

Имеется ряд других опций, которые можно добавить в файл конфигурации ядра системы с поддержкой брандмауэра, например, обеспечивающих журнализацию. Детальное описание конфигурирования брандмауэров выходит за рамки этой главы, но, при необходимости построения нового ядра с поддержкой брандмауэра обратитесь к разделу "Конфигурирование брандмауэра" главы 29. В этом случае не придется создавать еще одно ядро, если в дальнейшем вы решите использовать какую-либо опцию брандмауэра, не включенную в ядро первоначально.

Если процесс построения ядра не вполне понятен, обратитесь к главе 17, где он описан подробно.

Создав новое ядро, необходимо включить демон **natd**.

Конфигурирование и включение демона natd

Демон **natd** можно включить либо путем соответствующего конфигурирования сетевых установок в программе Sysinstall, либо вручную, редактируя файл **/etc/rc.conf**. При наличии уже работающей базовой сетевой конфигурации проще сделать необходимые изменения вручную, а не проходить через все шаги программы, так что именно это метод и будет описан.

Откройте файл **/etc/rc.conf** в любом текстовом редакторе и добавьте следующие строки:

```
natd_enable="YES"
natd_interface="ppp0"
```

При задании значения параметра **natd_interface** в предыдущем примере предполагается, что имеется подключение к Internet через модем, и что ему соответствует интерфейс **ppp0**. При наличии подключения по выделенной линии через устройство Ethernet (например по линии ADSL, кабелю, линии T1 или OC3) необходимо заменить **ppp0** соответствующим сетевым интерфейсом, по которому осуществляется подключение к внешнему миру.

У демона **natd** есть еще несколько опций, управляющих функциями типа журнализации. Если вас интересуют другие опции, обратитесь к странице справочного руководства по демону **natd**.

Не выходите из редактора и не сохраняйте пока измененный файл **rc.conf**, поскольку необходимо добавить еще одну опцию. Речь идет о настройке брандмауэра.

Включение и конфигурирование брандмауэра

Необходимо добавить в файл **/etc/rc.conf** следующую стоку для обеспечения поддержки брандмауэра:

```
firewall_enable="YES"
```

Есть разные способы сконфигурировать правила брандмауэра — подробно это описано в главе 29. Ниже кратко описан метод, который я считаю наиболее подходящим.

Кроме предыдущей строки, добавьте в файл **/etc/rc.conf** следующую:

```
firewall_type="/usr/local/etc/firewall.conf"
```

Сохраните изменения, сделанные в файле **/etc/rc.conf**, и выйдите из редактора. Теперь можно создать файл **/usr/local/etc/firewall.conf**, в который поместить правила брандмауэра (кроме стандартного правила, запрещающего все, что не разрешено явно, — подробнее см. в главе 29).

Если вы хотите пропускать все пакеты, следующих правил в файле **/usr/local/etc/firewall.conf** вполне достаточно:

```
add divert natd all from any to any via ed0
add allow all from any to any
```

Результат будет таким же, как если бы брандмауэра вообще не было, поскольку он пропускает все пакеты, ничего не запрещая.

Возможность для пользователей просматривать используемые системой правила брандмауэра может угрожать защите системы, поэтому права доступа к файлу нужно установить так, чтобы читать его мог только пользователь **root**. Используйте для установки соответствующих прав доступа команду **chmod 600 /usr/local/etc/firewall.conf**.

ПРЕДУПРЕЖДЕНИЕ

Ранее описанные правила брандмауэра — очень опасны, поскольку разрешают прием пакетов любого типа от какого угодно источника и передают их системе, использующей соответствующий шлюз. Это угрожает защите сети. Поэтому не рекомендуется пропускать всю информацию. Обратитесь к разделу "Конфигурирование брандмауэра" в главе 29, где представлена информация о конфигурировании брандмауэра для блокирования потенциально опасных типов пакетов.

Выполнив описанные выше шаги, перезагрузите систему, чтобы изменения ядра были учтены и чтобы брандмауэр и демон **natd** запустились. Теперь шлюз сконфигурирован. Осталось только проинформировать клиентов о наличии шлюза. Конфигурирование типичных клиентов рассматривается в следующем разделе. Конфигурирование клиентов других типов описано в документации соответствующей операционной системы.

Конфигурирование клиентов для использования нового шлюза

Процедуры конфигурирования клиентов для использования шлюза — различны и зависят от типа используемой на клиенте операционной системы. Мы рассмотрим конфигурирование клиентов Windows, Macintosh, FreeBSD и Linux.

Учтите: представленные далее инструкции по конфигурированию предполагают, что базовые сетевые службы уже сконфигурированы, и описывают только настройку системы для использования определенного шлюза. Если базовые сетевые службы в системе еще не сконфигурированы, инструкции о том, как это сделать, можно найти в документации системы.

Конфигурирование клиентов Windows 95/98

Чтобы сконфигурировать использование шлюза на клиенте Windows 95 или 98, дважды щелкните на пиктограмме My Computer, затем щелкните дважды на Control Panel и, наконец, щелкните дважды на пиктограмме Network. Откроется диалоговое окно, показанное на рис. 28.4.

В этом диалоговом окне щелкните на строке TCP/IP, а затем — на кнопке Properties. Откроется диалоговое окно конфигурации стека протоколов TCP/IP. Щелкнув на вкладке Gateway, вы увидите диалоговое окно, представленное на рис. 28.5.

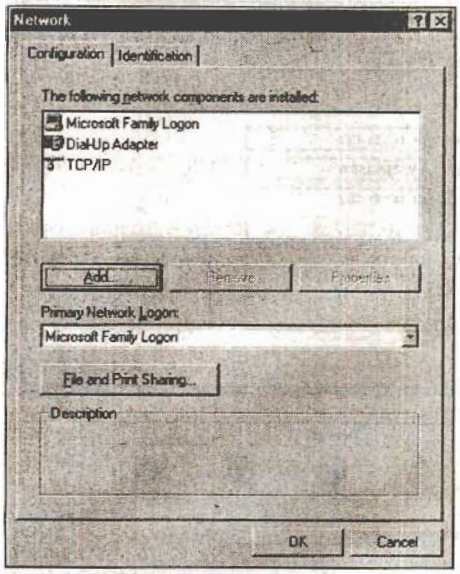


Рисунок 28.4. *Диалоговое окно Network Configuration в Windows 98. В Windows 95 и Windows ME оно может выглядеть иначе*

Рисунок 28.5. *Вкладка конфигурации шлюза Gateway и соответствующее диалоговое окно в Windows 98*

Введите IP-адрес шлюза в поле New gateway, как показано на рисунке. (В этом случае предполагается, что IP-адрес шлюза — **64.41.131.72**.) Затем щелкните на кнопке Add, и адрес шлюза покажется в списке установленных шлюзов (Installed gateways). Щелкните на кнопке OK для сохранения изменений. Щелкните на кнопке OK еще раз, чтобы выйти из диалогового окна конфигурации сети. В Windows 95 или 98 придется перезапустить компьютер, чтобы изменения были учтены, — об этом сообщит соответствующее диалоговое окно. При использовании Windows ME перезагрузка не понадобится.

После перезагрузки должна появиться возможность доступа к Internet с Windows-системы.

Конфигурирование клиентов Mac OS и Mac OS X

Клиенты Macintosh потенциально проще настроить для использования шлюза FreeBSD, чем Windows-системы, поскольку все параметры конфигурации TCP/IP задаются в одном окне и не нужна перезагрузка.

Если вы используете классическую операционную систему Mac OS (вплоть до версии 9), откройте панель управления TCP/IP, как показано на рис. 28.6, и выберите Manually (вручную) из выпадающего меню Configure:, если эта опция еще не выбрана. Введите IP-адрес шлюза в поле Router address:. При закрытии окна будет выдан запрос о необходимости сохранения изменений. После сохранения новые установки применяются немедленно.

В Mac OS X откройте окно System Preferences и выберите панель Network, показанную на рис. 28.7. Как и в Mac OS 9, выберите опцию Manually из меню Configure:

и укажите IP-адрес шлюза в поле Router:. Щелкните на кнопке Save для записи изменений. Новые параметры сети сразу же вступают в силу.

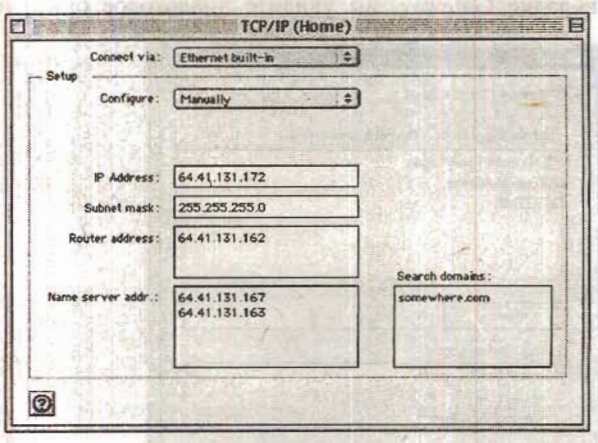


Рисунок 28.6. *Конфигурирование параметров TCP/IP в Mac OS 9*

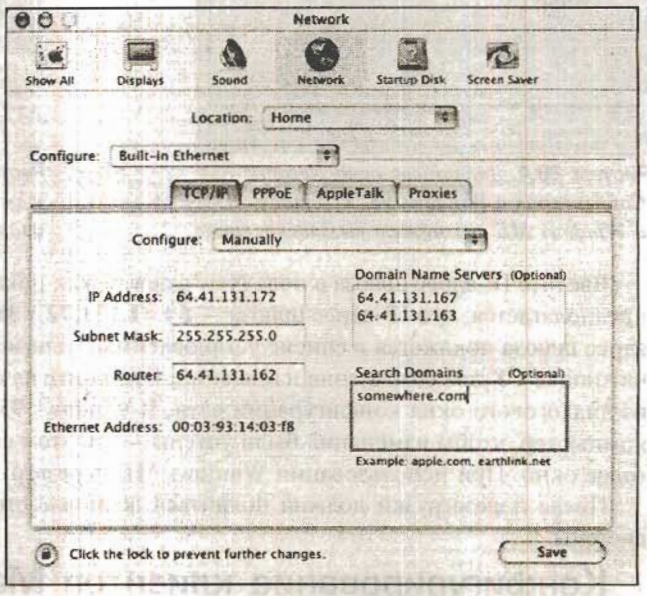


Рисунок 28.7. *Конфигурирование параметров TCP/IP в Mac OS X*

В обеих версиях операционных систем вы можете сконфигурировать встроенную карту Ethernet, карту беспроводной связи AirPort или другое сетевое устройство. Не забудьте повторить этот процесс для всех задействованных устройств.

Конфигурирование клиентов FreeBSD

Чтобы сконфигурировать стандартный шлюз в ОС FreeBSD, добавьте следующую строку в файл **/etc/rc.conf**:

```
defaultrouter="64.41.131.162"
```

Конечно, придется заменить IP-адрес в предыдущем примере IP-адресом реальной системы с ОС FreeBSD, работающей в качестве шлюза.

Изменение не будет учтено, пока система не перезагрузится.

Если вы не хотите перезапускать систему, можете добавить стандартный маршрут вручную с помощью следующей команды, выполненной от имени пользователя **root**:

```
route add default 64.41.131.162
```

Конфигурирование клиентов Linux

Из-за недостаточной стандартизации в Linux процедура конфигурирования Linux-системы для работы с шлюзом на базе ОС FreeBSD будет зависеть от используемого дистрибутива. Мы не будем описывать здесь все дистрибутивы, а только наиболее популярные. Если ваш дистрибутив не описан, обратитесь за инструкциями к документации.

ПРЕДУПРЕЖДЕНИЕ

По той же причине недостаточной стандартизации дистрибутивов Linux примеры этого раздела могут у вас не сработать. Обратитесь к документации по используемому дистрибутиву Linux, если при конфигурировании используемого шлюза возникнут проблемы.

Дистрибутив Red Hat Linux

Сетевая конфигурация Red Hat Linux задается в файле **/etc/sysconfig/network**. Чтобы использовать новый шлюз на базе ОС FreeBSD, в этом файле должна быть следующая строка:

```
GATEWAY=64.41.131.162
```

Замените IP-адрес в предыдущем примере IP-адресом интерфейса шлюза в системе с ОС FreeBSD, работающей в качестве шлюза.

Убедитесь, что такой строки еще нет, прежде чем добавлять новую, потому что две строки **GATEWAY** могут вызвать конфликт.

Систему Red Hat Linux придется перезапустить, чтобы изменения были учтены.

Если вы не хотите перезапускать систему, можете добавить стандартный маршрут вручную с помощью следующей команды, выполненной от имени пользователя **root**:

```
route add default gw 64.41.131.162
```

Дистрибутив Slackware Linux

В дистрибутиве Slackware Linux начальная часть настройки сети управляется файлом **/etc/rc.inet1**. В этом файле выполняется базовая настройка сети, включая маршрутизацию. Чтобы использовался новый шлюз FreeBSD, в файле должна присутствовать следующая строка:

```
GATEWAY="64.41.131.162"
```

Замените IP-адрес в предыдущем примере IP-адресом интерфейса шлюза в системе с ОС FreeBSD, работающей в качестве шлюза.

Убедитесь, что такой строки еще нет, прежде чем добавлять новую, потому что две строки **GATEWAY** могут вызвать конфликт.

Систему Slackware Linux придется перезапустить, чтобы изменения были учтены.

Если вы не хотите перезапускать систему, можете добавить стандартный маршрут вручную с помощью следующей команды, выполненной от имени пользователя **root**:

```
route add default gw 64.41.131.162
```

Конфигурирование беспроводного доступа к Internet

Беспроводный доступ к сети стал в последнее время очень популярен, поскольку не требуется ни прокладка кабелей, ни установка сетевых разъемов. Любой ПК, имеющий беспроводную сетевую карту, может работать с сетью. Многие переносные компьютеры уже имеют встроенные средства беспроводного доступа к сети.

Одно из наиболее типичных применений этой технологии — совместное использование сети Internet в университетских аудиториях. Студент может работать с переносным компьютером за столом и иметь доступ к Internet через центральный сервер.

Конфигурирование ОС FreeBSD для работы в качестве беспроводного шлюза практически идентично уже описанному конфигурированию обычного шлюза. Единственное отличие в том, что в системе FreeBSD должна быть установлена карта беспроводного доступа.

В табл. 28.1 представлен список беспроводных сетевых интерфейсов, поддерживаемых в настоящее время в ОС FreeBSD, а также соответствующие устройства, которые должны быть указаны в файле конфигурации ядра.

Таблица 28.1. Беспроводные сетевые интерфейсы, поддерживаемые в настоящее время в ОС FreeBSD

Беспроводный интерфейс	*Модули устройств ядра*
Aironet 4500/4800 802.11	device an
карты на базе AMD Am79C930	device awi
Xircom CNU/Netware Airsurfer	device cnw
Lucent WaveLan 802.11	device wi
Lucent WaveLan (только ISA)	device wl

За исключением устройств **cnw** и **wl**, поддержка этих карт включена в стандартное ядро GENERIC. Так что, если не используется беспроводная сетевая карта, требующая устройства **cnw** или **wl**, все уже готово для беспроводного подключения (если только не было создано специализированное ядро, из которого удалена поддержка соответствующих устройств).

Если все же необходимо построить новое ядро для поддержки используемой беспроводной сетевой карты, обратитесь к главе 17, где этот процесс описан детально.

После того, как беспроводная сетевая карта заработает, проделайте все ранее описанные в этой главе процедуры конфигурирования шлюза и клиентов.

Маршрутизация между тремя и более сетями

Во всех рассмотренных ситуациях конфигурировался маршрутизатор с единственным маршрутом. Например, наш шлюз **simba** имеет единственное подключение к сети Internet и обслуживает только одну сеть. На рис. 28.8 представлена соответствующая диаграмма.

В этом простом примере хосты **lion**, **tiger**, **cheetah** и **puma** используют хост **simba** в качестве стандартного шлюза для взаимодействия с системами в Internet. Аналогично, у

хоста **simba** тоже есть стандартный маршрутизатор — маршрутизатор поставщика услуг Internet. При коммутируемом соединении по PPP это незаметно, поскольку стандартный маршрут добавляется автоматически при установке соединения и автоматически же удаляется по его завершении. Об этом не надо беспокоиться, но нужно учитывать, что у шлюза тоже есть стандартный маршрутизатор, используемый точно так же, как клиенты используют ваш маршрутизатор.

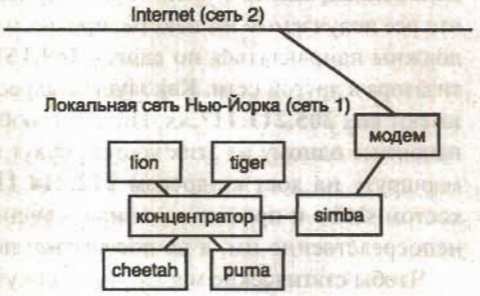


Рисунок 28.8.
Шлюз simba перенаправляет пакеты только между двумя сетями — локальной сетью и Internet

Однако иногда встречается более сложная конфигурация, вроде показанной на рис. 28.9. В этой ситуации хост **simba** действует в качестве стандартного маршрутизатора для своей сети, а также является стандартным маршрутизатором для хоста **falcon** — стандартного маршрутизатора другой сети.

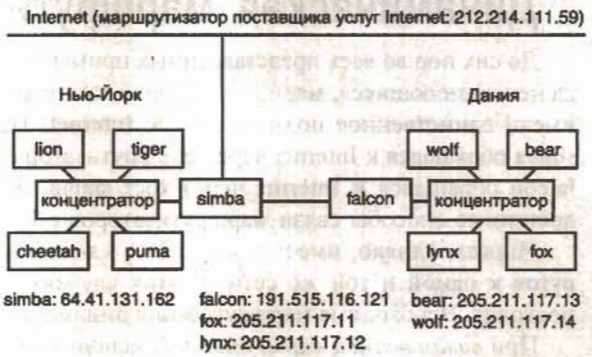


Рисунок 28.9.
Более сложная конфигурация шлюза. Хост simba является стандартным маршрутизатором как своей сети, так и хоста falcon, который выполняет функции стандартного маршрутизатора для другой сети.

В этом примере, помимо обслуживания клиентов собственной сети, хост **simba** фактически выступает в роли поставщика услуг Internet для сети, обслуживаемой маршрутизатором **falcon**. Это означает, что хост **simba** теперь перенаправляет пакеты между тремя сетями. Чтобы добиться этого, необходимо добавить еще один маршрут на хосте **simba**, с информацией о том, какие пакеты необходимо перенаправлять в сеть, обслуживаемую хостом **falcon**, и как их туда направить.

Сеть, обслуживаемая хостом **falcon**, владеет блоком адресов класса C. Обратите внимание, что все адреса в сети имеют вид **205.211.117.xx**. Учтите также, что хост **falcon** имеет IP-адрес ***169.151.116.121***. Чтобы хост **simba** перенаправлял пакеты для сети, обслуживаемой хостом **falcon**, необходимо добавить следующие строки в файл **/etc/rc.conf**:

```
defaultroute="212.214.111.59"
static_routes="falcon"
falcon="-net   205.211.117.0   169.151.116.121"
```

В данном случае IP-адрес **212.214.111.59** — это стандартный маршрутизатор для хоста **simba**. Другими словами, любые получаемые хостом **simba** пакеты, предназначенные неизвестным ему хостам, будут посылаться этому маршрутизатору. Хост с адресом **212.214.111.59** — маршрутизатор поставщика услуг Internet, обслуживающего хост **simba**.

Мы также добавили так называемый *статический маршрут*. Это статическая, или неизменная, запись в таблице маршрутизации. В данном случае мы сообщаем хосту **simba**, что все получаемые им пакеты, предназначенные хостам с базовым адресом **205.211.117.0**, должны направляться по адресу **169.151.116.121**. Это IP-адрес хоста **falcon** — маршрутизатора в другой сети. Как видите, адреса всех хостов сети, обслуживаемой хостом **falcon**, имеют вид **205.211.117.xx**. Поэтому любые пакеты, получаемые хостом **simba** и предназначенные одному из этих хостов, будут направляться хосту **falcon**, а не по стандартному маршруту на хост с адресом **212.214.111.59**. Кроме того, любые пакеты, полученные хостом **simba** и предназначенные клиентам в его собственной сети, будут направляться непосредственно им, а не посылаться по стандартному маршруту.

Чтобы статические маршруты были учтены, необходимо перезагрузить систему. Перезагрузка рекомендуется, но если не хотите, можете добавить статические маршруты вручную, чтобы они были учтены немедленно. В предыдущем примере это можно сделать с помощью следующих команд, выполненных от имени пользователя **root**:

```
route  add  -net  205.211.117.0  169.151.116.121
```

Динамическая маршрутизация

До сих пор во всех представленных примерах использовались статические, т.е. никогда не изменяющиеся, маршруты. Дело в том, что во всех этих примерах маршрутизаторы имели единственное подключение к Internet. Например, в предыдущем разделе, хост **simba** обращался к Internet через маршрутизатор своего поставщика услуг Internet. А хост **falcon** обращался к Internet через хост **simba**. В обоих случаях это были единственно доступные способы связи маршрутизаторов с Internet.

Иногда, однако, имеется несколько каналов выхода в Internet или несколько маршрутов к одной и той же сети. В этих случаях статическая маршрутизация не вполне подходит. Необходима маршрутизация динамическая.

При *динамической маршрутизации* используется специальный демон и протокол маршрутизации, позволяющий выявлять новые маршруты и динамически добавлять их в таблицу маршрутизации. Кроме того, при динамической маршрутизации из таблицы также автоматически удаляются маршруты, которые более недействительны.

Для ОС FreeBSD имеется несколько демонов маршрутизации. В базовую систему FreeBSD включается демон **routed**. Это сравнительно старая программа, использующая весьма старый протокол маршрутизации RIP (Routing Information Protocol — протокол передачи маршрутной информации). Протокол RIP имеет ряд проблем с защитой, поэтому для динамической маршрутизации есть более достойные демоны, чем **routed**. В набор портированных приложений ОС FreeBSD включены также демоны **gated** и **zebra** — оба находятся в подкаталоге **net** дерева портированных пакетов. Подробнее об установке программного обеспечения из набора портированных пакетов FreeBSD см. в главе 15.

Описание конфигурирования демонов маршрутизации выходит за рамки данной книги. Дополнительную информацию можно найти на странице справочного руководства и в документации по выбранному демону маршрутизации.

Хорошая новость состоит в том, что вам скорее всего вообще не понадобится запускать демон маршрутизации. Как уже было сказано, демон маршрутизации нужен только при наличии нескольких маршрутов к одной сети (например, при наличии нескольких подключений к Internet). В противном случае рассмотренных выше записей статической маршрутизации вполне достаточно.

Маршрутизация в масштабе предприятия и демилитаризованная зона (DMZ)

Разновидностью описанного подхода к маршрутизации между несколькими сетями является концепция демилитаризованной зоны (DMZ), часто применяемая в производственной среде.

Проблема в том, что некоторые системы (например, Web-серверы) за шлюзом должны иметь реальные IP-адреса, но все остальные системы должны использовать NAT. Обычно для этого используется шлюз с тремя сетевыми интерфейсами. Один интерфейс — связь с Internet, второй — для систем, использующих преобразование адресов NAT, а третий — для систем, которые не должны использовать NAT. Интерфейс, обслуживающий системы с реальными IP-адресами, называют *демилитаризованной зоной* (Demilitarized Zone — *DMZ*). Подробнее о настройке зоны DMZ см. в главе 29.

29 ГЛАВА

Защита сети

- Модели защиты
- Правила задания паролей
- Проблемы со службами, передающими информацию в явном виде
- Защита терминальной информации (OpenSSH)
- Защита служб электронной почты (POP3 и IMAP)
- Защита FTP-сервера
- Защита сервера Apache
- Профили защиты системы и защита ядра (securelevel)
- Использование брандмауэра
- Предотвращение вторжений и взломов
- Атаки на службы (DOS)
- Физическая защита
- Другие источники информации о защите

Из всех тем, близких сердцу системного администратора, наибольший интерес вызывает защита. Защита — определенно, наиболее важная часть работы системного администратора, независимо от того, управляет ли он сервером Windows, коммерческой UNIX-системой или ОС FreeBSD. Не случайно по теории защиты, практике защиты, взлому защиты, философии защиты и т.д. написано больше книг, чем по любой другой теме администрирования. Это очень сложная тема. Не стоит и надеяться раскрыть ее полностью в одной главе, но защита настолько важна для успешной работы системы в сети, что описание ОС FreeBSD будет явно недостаточным без обсуждения защиты.

Сегодня сеть Internet — не слишком дружественное место для серверов. Изобилие "хакерских" ("rootkit") средств и опубликованных сценариев атак в сочетании с бесчисленными личностями, не находящими ничего лучше, чем деструктивные увлечения, создает атмосферу, в которой администратор должен всегда предполагать худшее и готовиться к нему. Надо иметь в виду, что система испытывается на наличие слабых мест в защите круглые сутки, и с каждым опубликованным методом взлома ситуация еще более ухудшается. Единственная защита — поддержка самой современной версии системы, выполнение рекомендаций по защите и знание реальных опасностей и зон максимального риска.

Угрозы защите сетевого сервера можно разделить на три основные категории.

- **Получение доступа от имени пользователя root.** Атакующий использует информацию, передаваемую в незашифрованном виде, или известные ошибки в программном обеспечении сервера (чаще всего — "переполнения буфера") для получения доступа к системе от имени суперпользователя. Затем он устанавливает собственные программы, скрывающие его присутствие от средств контроля системы (таких как **last** и **ps**), и может похищать важные для системы данные, служащие основой для дальнейших взломов.
- **Нарушение конфиденциальности.** Если информация, передаваемая по сети с вашей системы и на нее, не шифруется (не скрывается), атакующий может просматривать любую ее часть, включая пароли (что позволяет получить доступ от имени пользователя **root**) и конфиденциальные или секретные данные любого пользователя.
- **Сбой служб.** Атакующий использует силовые методы, вроде переполнения сервера большими объемами вполне законной информации, что лишает его возможности обслуживать обычных клиентов и может привести к сбою системы.

Эта глава поможет вам разработать правила защиты для системы FreeBSD, работающей в качестве сервера или рабочей станции, с учетом факторов риска. Реальность подтверждает, что совершенной модели защиты не существует, и только нечеловеческими усилиями можно поддерживать защиту системы на таком уровне, что никакая атака ей не страшна. "Совершенная защита" — миф. Лучшее, что можно сделать, — прикрыть области наибольшего риска благодаря знанию их природы и способов защиты.

Модели защиты

Есть несколько моделей защиты, которые вы можете принять для своей системы, — в зависимости от условий каждая из них может оказаться наиболее подходящей. Выбранная модель определяет, насколько тщательно придется выполнять определенные административные обязанности, например, внедрять правила использования паролей, открывать доступ к службам, шифровать передаваемую информацию и т.д.

- **"Я доверяю всем в Internet".** Хотя эту модель практически никогда нельзя рекомендовать, именно ее придерживаются администраторы любительских серверов (и дорого за это платят). Часто встречающиеся в университетской среде, особенно среди эксплуатируемых многие годы (с того времени, когда сеть Internet еще не была переполнена взломщиками), системы этого типа предлагают много открытых служб, не требуют шифрования при удаленном доступе, имеют нестрогие правила задания паролей и ведения учетных записей, являясь тем самым привлекательными целями для атак взломщиков.
- **"Я доверяю всем в локальной сети".** Типичная для сетей предприятий эта модель обычно используется, когда сервер защищен от остальной части Internet брандмауэром, а внутренняя сеть используется исключительно сотрудниками. Враждебно настроенные пользователи во внутренней сети встречаются редко, поэтому можно себе позволить не шифровать передаваемую информацию и даже отключать защиту учетных записей. Если брандмауэр надежен, сервер в такой среде может иметь самую слабую защиту и вполне приемлемо работать.
- **"Я доверяю локальным пользователям".** Несколько более параноидальная, чем две предыдущие, эта модель диктует строгие правила сетевой защиты: пользователи не принимаются, пока для них не будут сделаны новые учетные записи, передаваемая сетевыми службами информация шифруется (шифрование либо требуется, либо настоятельно рекомендуется), ненужные службы отключаются, и используются устойчивые к взлому пароли. Однако локальным пользователям разрешается доступ к внутренним службам и просмотр конфиденциальной информации (например, зашифрованных паролей). Идея в том, что если уж пользователям выделены учетные записи, они могут использовать систему в полном объеме, а если им нельзя доверять в такой степени, то это является основанием для удаления из системы. Эта модель подходит для любительских систем, обслуживающих пользователей из группы "низкого риска" (например, Web-сайт фанатов футбольного клуба или служба электронной почты для жильцов дома), или для высокоуровневых коммерческих серверов Internet, где лишь немногие пользователи имеют учетные записи.
- **"Я доверяю только себе и другим администраторам".** Наиболее параноидальная из всех эта модель не только предполагает такую же серьезную сетевую защиту, как и предыдущая, но и строгую защиту для локальных пользователей. Обычным пользователям запрещен доступ к файлам конфигурации системы и серверным программам благодаря специально сконфигурированным правам доступа. Администратор должен внимательно следить за каждым пользователем, чтобы гарантировать отсутствие неавторизованных действий, причем часто принимаются специальные меры (используются специализированные командные интерпретаторы, изменяется начальный каталог с помощью команды **chroot** и отключаются определенные команды), ограничивающие доступ пользователя к ресурсам системы. Эта модель подходит для высокоуровневых серверов, поддерживающих службы электронной почты или Web-хостинга для сотен или тысяч пользователей с неопределяемыми или неизвестными целями.

Выбрав подходящую модель защиты сети и пользователей, необходимо решить, какие области риска предполагаются этой моделью и что можно сделать, чтобы уменьшить вероятность взлома в этих областях. Наиболее типичными областями риска являются:

- небезопасные (слабые) пароли;
- службы, пересылающие информацию в явном (незашифрованном) виде;
- ненужные и легко взламываемые службы;
- открытая пересылка сообщений по протоколу SMTP;
- неограниченный сетевой доступ;
- устаревшее или уязвимое программное обеспечение.

Каждое из этих узких мест — потенциальная проблема в ОС FreeBSD в стандартной конфигурации. В этой главе описано, как решать каждую из них и представлены необходимые средства поддержки системы, предотвращающие взломы.

ПРИМЕЧАНИЕ

Открытая пересылка сообщений по протоколу SMTP скорее мешает быть добропорядочным членом сообщества систем Internet, чем является проблемой защиты. Ее детальное обсуждение представлено в главе 25.

Правила задания паролей

Если пользователи имеют небезопасные пароли, все остальные принимаемые меры защиты могут оказаться неэффективными. Вероятно, наиболее ответственная задача администратора системы FreeBSD — установить правила задания паролей, требующие от пользователей применять пароли, которые сложно подобрать или взломать (или хотя бы побуждающие их к этому).

Многим пользователям необходимость ввода паролей кажется неудобством, а использование строгих правил задания паролей — двойным неудобством. Неконтролируемый пользователь будет задавать в качестве пароля свое регистрационное имя, номер телефона, имя хоста, слово типа "password" или строку символов, которую удобно набирать, например строку из повторяющихся букв или цифр. Если пароль устареет, первое, что попытается сделать пользователь в ответ на запрос нового пароля, — ввести старый пароль. Одна из аксиом защиты, однако, гласит, что "удобство и безопасность — взаимоисключающи", т.е. повышать одно можно только за счет снижения другого. Большее удобство означает меньшую безопасность. Этот факт обойти трудно.

При выборе пользователем пароля с помощью стандартной программы **passwd** или сценария, вызывающего те же подпрограммы, что и **passwd**, выполняется несколько несложных проверок. По умолчанию пароли должны состоять не менее чем из шести символов, но это, похоже, единственная встроенная мера против выбора пользователями слабых паролей. Давайте рассмотрим несколько методов, позволяющих существенно повысить уровень защиты при выборе паролей.

Обеспечение безопасности паролей с помощью программы Crack

Идеальный пароль состоит не менее чем из восьми символов (чем длиннее, тем лучше) и содержит прописные и строчные буквы, цифры и знаки препинания или "метасимволы". Необходимо с помощью программы **passwd** проверять, все ли пользователи следуют этим принципам при задании пароля.

Напомню, что единственное ограничение, налагаемое стандартной программой **passwd**, — длина пароля должна быть не менее шести символов. Сообществом разработчиков FreeBSD прилагаются усилия по внедрению в программу **passwd** дополнительных проверок слабых паролей, прежде всего запрещающих пользователям выбирать небезопасные пароли. На момент написания этой главы, однако, лучший способ убедиться, что пользователи не используют легко подбираемые пароли, — периодически пытаться подобрать их самому. Это делается с помощью программы **Crack**, находящейся в наборе портированных приложений в каталоге **/usr/ports/security/crack**. Хотя эта программа может показаться "хакерской", она предназначена для контроля защиты системными администраторами. Программа Crack позволяет выполнять "атаки по словарю" (попытки подобрать в качестве пароля одно из английских слов), а также выявлять ряд других "удобных" паролей: повторяющиеся строки, регистрационное имя пользователя, группы цифр и т.д. Цель — показать, кто из пользователей применяет небезопасные пароли, чтобы можно было связаться с ними непосредственно и потребовать соблюдения установленных правил задания паролей.

После создания и установки портированной программы **Crack** (подробнее об установке дополнительного программного обеспечения см. в главе 15), появляется новый каталог, **/usr/local/crack**, с правами доступа, позволяющими просматривать содержимое или запускать находящиеся в нем программы только пользователю root. Для того чтобы проверить наличие слабых паролей в базе данных пользователей системы, перейдите в каталог **/usr/local/crack** и запустите программу **Crack** следующим образом (обратите внимание, что первая буква — прописная):

```
# ./Crack -fmt bsd /etc/master.passwd
```

Программа **Crack** создаст ряд утилит и скомпилирует несколько словарей. Затем она применит свои проверки к файлу **/etc/master.passwd**, посылая результаты в рабочие файлы, которые можно анализировать с помощью программы **Reporter** следующим образом

```
./Reporter -quiet
---- passwords cracked as of Sun Jan 14 12:17:41 EST 2001 ----
979693112:Guessed frank [frank] Frank Jones [/etc/master.passwd /bin/tcsh]
979693187:Guessed joe [password] Joe User [/etc/master.passwd /usr/local/
  ↪bin/bash]
---- done ----
```

Будет сообщено только о пользователях, пароли которых были подобраны. В представленном выше примере взломанные пароли показаны в первых квадратных скобках. Пароль пользователя Frank — **frank**, а пароль пользователя Joe — **password**. Оба эти пароля — крайне слабые, так как без особых усилий могут быть угаданы атакующим. Теперь можно связаться с этими пользователями и напомнить им о правилах задания паролей, потребовав изменить пароли на более безопасные.

После завершения работы программы **Crack**, удалите утилиты времени выполнения и файлы результатов с помощью следующих двух команд:

```
# make tidy
# rm run/F-merged
```

ПРИМЕЧАНИЕ

Поскольку ОС FreeBSD продолжает развиваться, весьма вероятно, что в программу **passwd** будет включена поддержка автоматической проверки сложности паролей, аналогичная выполня-

емой вручную с помощью программы Crack. Библиотеки, используемые программой Crack, доступны в каталоге **/usr/ports/security/cracklib**. В системах типа Linux на основе библиотеки **cracklib** был разработан встраиваемый модуль аутентификации (pluggable authentication module — PAM) — механизм, поддерживаемый также и в ОС FreeBSD (см. страницу справочного руководства **man pam**), но на момент написания этой главы ОС FreeBSD не поддерживает полную интеграцию **cracklib** в систему модулей PAM. Сейчас же авантюристы, желающие поработать с исходным кодом и экспериментальным программным обеспечением и заинтересованные во включении возможностей библиотеки **cracklib** в программу **passwd**, могут узнать, как это можно сделать, по адресу **http://www.kearneys.ca/~brent/FreeBSD/passwd42.html**.

Устаревание паролей

По умолчанию пароли в ОС FreeBSD не устаревают. Однако одним из стандартных правил обеспечения безопасности паролей является требование частой из смены, с заданным администратором сроком действия.

Чтобы сделать это в ОС FreeBSD, необходимо изменить файл **/etc/login.conf**. Этот файл обеспечивает централизованный способ управления возможностями и поведением системы (например, допустимым количеством одновременных процессов, максимально допустимым размером процесса, максимальным количеством одновременно открытых файлов, особенностями работы командных интерпретаторов и многими другими параметрами, описанными на странице справочного руководства **man login.conf**). Каждое из этих свойств может быть привязано к "классу" пользователей, который можно задавать с помощью команды **chfn** (как было описано ранее). Обычно пользователи не связаны с конкретным классом, поэтому значения стандартного класса **default** применяются для всех:

```
default:\
        :passwd_format=md5:\
        :copyright=/etc/COPYRIGHT:\
        :welcome=/etc/motd:\
        :setenv=MAIL=/var/mail/$,BLOCKSIZE=K,FTP_PASSIVE_MODE=YES:\
        :path=/sbin /bin /usr/sbin /usr/bin /usr/games /usr/local/
        ↪sbin /usr/local/bin /usr/X11R6/bin ~/bin:\
        :nologin=/var/run/nologin:\
        :cputime=unlimited:\
        :datasize=unlimited:\
        :stacksize=unlimited:\
        :memorylocked=unlimited:\
        :memoryuse=unlimited:\
        :filesize=unlimited:\
        :coredumpsize=unlimited:\
        :openfiles=unlimited:\
        :maxproc=unlimited:\
        :sbsize=unlimited:\
        :priority=0:\
        :ignoretime@:\
        :umask=022:
```

Символы обратной косой (\) "маскируют" переводы строк, позволяя задавать все эти свойства по одному в строке, что упрощает понимание файла.

Для установки даты устаревания пароля необходимо добавить в класс **default** дополнительную строку, задающую свойство **passwordtime**. Ее можно включить в блок в любом месте, но проще всего — сверху, сразу после имени класса:

```
default:\
        :passwordtime=90d:\
        :passwd_format=md5:\
        :copyright=/etc/COPYRIGHT:\
        :welcome=/etc/motd:\
```

В этом примере устанавливается устаревание паролей через 90 дней. Можно использовать и другие форматы задания времени, например **2y** (2 года), **6w** (6 недель) или **24h** (24 часа). Теперь, поскольку файл **/etc/login.conf** представляет собой базу данных, которую необходимо скомпилировать в хеш-таблицу (как и таблицы в каталоге **/etc/mail**, рассматривавшиеся в главе 25), надо вызвать программу **cap_mkdb** для генерации хеш-таблицы с учетом изменений:

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

С этого момента, если со времени последнего изменения пароля пользователем прошло более 90 дней, процедура регистрации потребует от пользователя выбрать новый пароль. Учтите, что если свойство **passwordtime** установлено, программа **passwd** записывает время последнего изменения пароля в шестое поле файла **/etc/master.passwd**:

```
frank:$1$LXZkCuzD$7Oa8LyRf5jYOb.XrXiB3d.:1060:100::999066364:0::/home/frank:/
➥bin/tcsh
```

СОВЕТ

Можно использовать файл **login.conf** и для изменения стандартной минимальной длины пароля. Это делается путем задания значения свойства **minpasswordlen**:

```
:minpasswordlen=8:\
```

Тем самым минимальная длина пароля устанавливается равной 8 символам.

Задание первоначальных паролей

Кажется удобным устанавливать простой начальный пароль для каждого нового пользователя, например **Temp123** или **ChangeThis**. Однако это несет серьезную угрозу защите, особенно если для всех вновь добавляемых пользователей задается один и тот же пароль.

Эту угрозу можно существенно уменьшить, генерируя для каждого пользователя пароль случайным образом. При этом можно использовать любую схему генерации паролей (например, по первым буквам слов в названиях песен), но выполнение таких действий вручную вам быстро надоест. Есть хороший способ генерировать уникальные и неподбираемые пароли — с помощью программы **md5** и нескольких случайных нажатий на клавиши:

```
# md5 -s "asdsad"
MD5 ("asdsad") = b5b037a78522671b89a2c1b21d9b80c6
```

Затем можно взять первые семь или восемь символов из этой строки (например, **b5b037a7**) в качестве пароля нового пользователя, проинструктировав пользователя о том, как с помощью программы **passwd** заменить пароль другим, проще запоминаемым. Имеет смысл реализовать эту схему в виде небольшого сценария на языке Perl, выполняющего для генерации нового пароля хеширование с помощью алгоритма MD5 результата выполнения функции **rand()**.

Одноразовые пароли на базе системы S/Key

Если вы действительно серьезно озабочены безопасностью паролей, можете сделать то же, что и правительственные структуры и другие организации, использующие максимальную защиту: назначать пользователям одноразовые пароли, перенося часть бремени обеспечения защиты со своих плеч на плечи пользователей. Одноразовые пароли генерируются ключевой программой (**key**), разновидности которой существуют для всех основных платформ — есть даже не зависящая от платформы реализация на языке Java, см. **http://www.cs.umd.edu/~harry/jotp/src.html**. (В ОС FreeBSD для вычисления ключей используется алгоритм MD4.)

Одноразовые пароли хорошо подходят для систем, в которых пользователи не обязаны использовать систему SSH вместо Telnet (систему SSH мы рассмотрим далее). Поскольку новый пароль должен генерироваться пользователем с помощью ключевой программы на локальной системе, которой передается выданная сервером фраза "запроса" ("challenge" phrase) и собственный секретный пароль пользователя, никогда не пересылаемый по сети (за исключением процедуры начальной установки ключа), перехват проходящих по сети пакетов никогда не даст никакой полезной информации. Однажды использованный пароль нельзя использовать повторно. Пользователь может передавать одноразовый пароль в виде обычного текста.

ПРИМЕЧАНИЕ

Система S/Key не менее ценное средство для озабоченного защитой пользователя, чем требование соблюдать принципы защиты для администратора. Многие части системы S/Key, например программа **keyinit**, должны поддерживаться пользователем. Если пользователь хочет обеспечить конфиденциальность паролей, он может использовать одноразовые пароли, даже если в этом нет особой необходимости.

Предположим, необходимо сделать так, чтобы пользователь Frank не мог зарегистрироваться с удаленного хоста с обычным паролем UNIX, а был вынужден использовать одноразовые пароли S/Key. Зарегистрировавшись на сервере (предпочтительно по безопасному каналу, например, с помощью SSH), он должен воспользоваться программой **keyinit** для настройки аутентификации системы S/Key:

```
# keyinit
Adding frank:
Reminder-Only use this method if you are directly connected.
If you are using telnet or rlogin exit with no password and use
   ↪keyinit -s.
Enter secret password:
Again secret password:

ID frank s/key is 99 st28077
COL APT HELM TAB DRY TRIM
```

ПРИМЕЧАНИЕ

Если пользователь Frank подключен по небезопасному каналу (например, при простом соединении с помощью Telnet), он должен использовать вызов **keyinit -s**. При указании опции **-s** программа **keyinit** использует генерацию ключа по ходу работы. Пользователь Frank вводит свой пароль для генерации секретного ключа (который используется исключительно для вычисления одноразовых паролей системы S/Key и не должен совпадать с его паролем в UNIX) и передает его по сети на сервер. Если подключение выполняется по небезопасному каналу, секретный пароль можно перехватить, что делает все дальнейшие действия по защите бесполезными. Опция **-s** требует от пользователя Frank использовать программу **key** локально, на его

собственной машине с ОС Windows, Macintosh или UNIX для генерации пароля, который затем надо будет ввести в программе **keyinit** в ответ на приглашение **s/key access password:** (которое выдается только при указании опции **-s**).

После того как пользователь Frank применил программу **keyinit** для настройки своего механизма S/Key, в результате чего для соответствующего регистрационного имени появилась запись в файле **/etc/skeykeys**, необходимо создать файл **/etc/skey.access** (если он еще не существует) и добавить в него следующую строку:

```
deny user frank
```

Файл **/etc/skey.access** сообщает ОС FreeBSD, при каких условиях удаленный пользователь может использовать обычный пароль UNIX, а когда обязан использовать одноразовый пароль системы S/Key. Каждая строка в файле **skey.access** задает правило, начинающееся с ключевого слова **permit** (разрешающего регистрацию как с паролем S/Key, так и с паролем UNIX) или **deny** (требующего ввода пароля системы S/Key), за которым следует любое количество условий. Эти условия, полностью описанные на странице справочного руководства **man skey.access**, могут задавать конкретных пользователей, группы, имена удаленных хостов или сетей или терминалы, с которых выполняется регистрация. Строка в представленном выше примере указывает, что пользователь Frank при попытке регистрации может вводить только пароль системы S/Key, а не обычный пароль UNIX.

При следующей попытке пользователя Frank подключиться к системе с помощью Telnet приглашение регистрации для него будет таким:

```
# telnet stripes.somewhere.com
Trying  64.41.131.102...
Connected to stripes.somewhere.com.
Escape character is '^]'.

FreeBSD/i386 (stripes.somewhere.com) (ttyp2)

login: frank
s/key 99 st28077
Password:
```

Теперь пользователь Frank должен вызвать программу **key** (или ее эквивалент) на локальной машине для получения необходимого пароля. Программе надо передать информацию запроса, представленную сервером: номер итерации (99 в нашем случае; это означает, что осталось 99 регистраций, прежде чем придется запускать программу **keyinit** снова) и строку "затравки" (в нашем примере, **st28077** — ту же, что и в примере вызова программы **keyinit**). Эти числа в сочетании с секретным паролем пользователя Frank используются для генерации пароля S/Key, состоящего из шести коротких английских слов в верхнем регистре:

```
# key 99 st28077
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:
COL  APT  HELM  TAB  DRY  TRIM
```

Теперь пользователь Frank может ввести эту строку в качестве пароля, и ему будет предоставлен доступ. Если атакующий перехватит эти слова, ему это ничего не даст — они передаются по сети только один раз (в ответ на приглашение регистрации). При следующем запросе системы у пользователя Frank пароля S/Key номер итерации будет 98, и пароль будет уже другим. Номер итерации уменьшается, пока не станет

нулевым. В этом случае пользователь Frank должен запустить программу **keyinit** снова, чтобы номер итерации опять стал равен 99, в противном случае доступа он не получит, и придется вмешаться администратору. (Например, чтобы изменить для пользователя правило в файле **/etc/skey.access**, заменив ключевое слово **deny** на **permit** пока пользователь не переинициализирует систему S/Key.)

СОВЕТ

При желании пользователь Frank может сгенерировать сразу несколько ключей, вызвав программу **key** с опцией **-n**. Затем он может распечатать эти ключи и взять их с собой, если это покажется ему более удобным или безопасным:

```
# key -n 5 50 st28077
Reminder - Do not use this program while logged in via telnet or rlogin.
Enter secret password:
41: SHOT YOU BIEN GIN JUDD AS
42: CHOW AVIS DOES EMIT FLAM WORK
43: DOCK ATE ANN WAS JOCK OAT
44: WALE AWL ELK LETS AWK WALE
45: GIFT BERT ROD GRIN YANG EAST
```

Запрос пароля S/Key также выдается при вызове команды **su**, что помогает вообще предотвратить передачу фактического пароля пользователя **root** по сети.

Чтобы отключить поддержку паролей S/Key для пользователя, удалите запись этого пользователя из файла **/etc/skeykeys**, а также любое упоминание о нем из файла **/etc/skey.access**.

Система Kerberos

Еще один достойный упоминания метод аутентификации — система Kerberos. Разрабатывавшаяся как централизованная система управления регистрацией для кластера Athena в МТИ система Kerberos обеспечивает аутентификацию пользователей центральным сервером сети с выдачей "мандатов" ("tickets") для работы со службами Telnet, FTP, POP3 и NFS без необходимости регистрироваться каждый раз. Если хосты в сети, между которыми передается информация, поддерживают систему Kerberos и подписаны на главном сервере, все проблемы аутентификации берет на себя система Kerberos. ОС FreeBSD позволяет настроить главный сервер Kerberos, на услуги которого подписываются все остальные хосты сети, или просто поддерживать работу с системой Kerberos в сети, где она уже работает.

До недавнего времени система Kerberos была действительно полезна только в коммерческой среде или в самом МТИ. Она упрощает выполнение типичных задач в больших локальных сетях, типичных для университетов, или иерархических сетях предприятий. Помимо этих, остается не так уж много ситуаций, когда система Kerberos действительно эффективна с учетом сегодняшних сетевых тенденций. Это не столько мера защиты, сколько способ избавиться от ненужных действий в средах, где используется исключительно ОС UNIX. Однако поскольку многие сети предприятий используют систему Kerberos для обеспечения централизованных, шифрованных служб регистрации и поскольку модель защиты Windows 2000 тесно связана с системой Kerberos (хотя и с модифицированной ее версией), она снова становится важной и широко распространенной. Может потребоваться интегрировать с ней машину с ОС FreeBSD.

Если необходимо обеспечить поддержку службы Kerberos в ОС FreeBSD, это можно сделать. При добавлении строки **kerberos_server_enable="YES"** в файл **/etc/rc.conf**

будет запускаться служба, управляющая главным сервером Kerberos. Если же необходимо работать с уже существующим главным сервером, можно убрать символ комментария с соответствующих строк в файле **/etc/pam.conf** для включения централизованной аутентификации служб, которые ее поддерживают, например:

```
login auth sufficient pam_kerberosIV.so try_first_pass
```

Дополнительную информацию по системе Kerberos можно найти на странице справочного руководства **man kerberos** и в оперативном руководстве по ОС FreeBSD (**http://www.freebsd.org/handbook**).

Проблемы со службами, передающими информацию в явном виде

Может показаться, что передача информации с клиентской машины на сервер (в приложениях типа Telnet, электронной почты и HTTP) вполне безопасна — пароли ваши скрыты (по крайней мере их не видно), и вся информация передается небольшими пакетами, проходящими по сети вместе с миллионами других пакетов, так что мысль, будто кто-то прослушивает ваш канал передачи, кажется смешной. Это можно делать только из того же сегмента локальной сети или из локальной сети на другом конце соединения, или в сети одного из недальновидных поставщиков услуг Internet по ходу передачи, причем маскировка и оборудование для этого должны быть лучше, чем у Джеймса Бонда. И даже если кому-то удастся организовать "прослушивание", придется следить за множеством подобных сеансов, прежде чем удастся обнаружить что-нибудь полезное. Ну и кому настолько нечего делать, чтобы заниматься такими вещами?

Да кому угодно. Безопасность служб, передающих информацию в явном виде (приложений, не шифрующих или не скрывающих данные транзакций), — миф. Что еще важнее, принцип "безопасность вследствие неочевидности" ("security through obscurity") — в данном случае представление о том, что каналы связи или службы безопасны, поскольку вам непонятно, кому они могут быть интересны, — не работает. Ложность этой аксиомы была многократно доказана за последние годы. Не поддавайтесь искушению игнорировать защиту служб, передающих информацию в явном виде, в угоду слепой вере, что никому не придет в голову взламывать вашу систему. Всегда надо предполагать наихудшее, например, что вся передаваемая вами информация постоянно перехватывается с враждебными целями.

Использование утилиты tcpdump для контроля передаваемой информации

Продемонстрировать существенный риск использования в сети служб, передающих информацию в явном виде, можно с помощью *анализатора протокола* — программы, следящей за всеми пакетами в сегменте сети и выдающей те из них, которые представляют для вас интерес. Анализатор протокола переводит карту Ethernet в "неразборчивый" режим ("promiscuous mode"), в котором она принимает все пакеты, проходящие по сети, а не отвергает пакеты, адресованные другому интерфейсу (как описывалось в главе 22). Затем он применяет к просматриваемой информации разнообразные фильтры с широкими возможностями конфигурирования. Именно так злоумышленник может извлечь только нужные пакеты из потока информации и перехватить сетевые транзакции.

Встроенный анализатор протокола ОС FreeBSD — программа **tcpdump**, находящаяся в каталоге **/usr/sbin**. Его использование для обычных пользователей запрещено — они получат сообщение об ошибке прав доступа (Permission denied) к устройству **/dev/bpf0** (фильтр пакетов Беркли). Эту программу может использовать только администратор (**root**). Программа **tcpdump** предназначена не столько для регистрации реальных данных в просматриваемых пакетах TCP/IP (для этих целей имеет смысл использовать программу EtherPeek от WildPackets), а скорее, как и система **Crack**, для проверки защиты — оценки объема информации, передаваемой в систему и из системы в незашифрованном виде, и отслеживания подозрительных действий в сети, возможно, связанных с враждебными намерениями.

ПРИМЕЧАНИЕ

Программу **tcpdump** можно использовать для скрытого слежения за действиями пользователей, независимо от их направленности. Предупреждение в файле конфигурации ядра **GENERIC** напоминает о необходимости учитывать этические аспекты, использования анализатора протокола. Все, естественно, зависит от того, в какого рода системе вы работаете, но использование анализаторов протокола аналогично прослушиванию телефонных переговоров или использованию скрытых видеокамер слежения. Поэтому использовать программу **tcpdump** стоит только в тех случаях, когда вас не смущало бы использование для сбора информации средств прослушивания и скрытых видеокамер.

Файл конфигурации программы **tcpdump** — весьма сложный и позволяет делать множество полезных и разнообразных вещей. Пока, однако, нас интересует только возможность продемонстрировать небезопасность передачи информации по протоколу TCP/IP в явном виде. Давайте настроим простой контролирующий фильтр для порта Telnet (TCP-порт 23):

```
# tcpdump -x port 23
tcpdump: listening on fxp0
20:14:19.076941 w044.z064002043.sjc-ca.dsl.cnc.net.54109 >
w012.z064002043.sjc-ca.dsl.cnc.net.telnet: S 1972342903:1972342903(0)
win 32768 <mss 1460,nop,wscale 0,nop,nop,timestamp 465710 0> (DF)
[tos 0x10]
                        4510 003c e44b 4000 4006 8024 4002 2b2c
                        4002 2b0c d35d 0017 758f 9077 0000 0000
                        a002 8000 001b 0000 0204 05b4 0103 0300
                        0101 080a 0007 1b2e 0000 0000
                        20:14:19.077050 w012.z064002043.sjc-
ca.dsl.cnc.net.telnet > w044.z064002043.sjc-ca.dsl.cnc.net.54109: S
1734674412:1734674412(0) ack 1972342904 win 17520 <mss 1460> (DF)
                        4500 002c c9c2 4000 4006 9acd 4002 2b0c
                        4002 2b2c 0017 d35d 6765 07ec 758f 9078
                        6012 4470 349c 0000 0204 05b4
...
20:14:19.677472 w044.z064002043.sjc-ca.dsl.cnc.net.54109 >
w012.z064002043.sjc-ca.dsl.cnc.net.telnet: . ack 195 win 33580 (DF)
[tos 0x10]
                        4510 0028 e458 4000 4006 802b 4002 2b2c
                        4002 2b0c d35d 0017 758f 910a 6765 08af
                        5010 832c 0c49 0000 5555 5555 5555
^C
134 packets received by filter
0 packets dropped by kernel
```

Поскольку программа **tcpdump** не создавалась для "подслушивания", результат (при указании опции **-x**) выдается в шестнадцатеричном формате, который можно преоб-

разовать в обычный, понятный человеку текст, с помощью соответствующего редактора. Эти данные никак не зашифрованы. Требуются минимальные усилия, чтобы получить полное содержимое пакетов, посылаемых с использованием протокола Telnet. Еще проще это сделать с помощью средств, автоматически декодирующих пакеты, например с помощью программы EtherPeek. Если эти пакеты содержат регистрационную информацию пользователя, атакующий с помощью программы **tcpdump** или EtherPeek сможет ее узнать и зарегистрироваться в вашей системе. Это касается и протоколов POP3, IMAP, FTP и HTTP, а также почти всех небольших сетевых служб вроде Finger и Syslog. Особенно важно это учитывать при передаче паролей или любых других конфиденциальных данных. Проблема эта связана с обеспечением конфиденциальности не меньше, чем с обеспечением защиты.

К счастью, есть способ решить эти проблемы. Существуют шифрованные варианты реализации основных протоколов передачи данных, многие из которых входят в стандартную установку ОС FreeBSD. Надо только знать, что они существуют, внедрить их и убедить пользователей в необходимости их применять.

Защита терминальной информации (OpenSSH)

Передача терминальной информации, обычно выполняемая приложениями Telnet или **rlogin**, вероятно, наиболее рискованный тип передачи информации в явном виде, но и самый простой для защиты. В состав ОС FreeBSD входит полный пакет SSH (Secure Shell — защищенный командный интерпретатор), разработанный для замены служб Telnet и **rlogin**, и позволяющий пользователям установить полностью зашифрованный канал обмена информацией с сервером, защищающий пароли регистрации и операции командной строки от попыток перехвата. Речь идет о пакете OpenSSH, первоначально разработанном для ОС OpenBSD, а теперь входящем в состав ОС FreeBSD.

Служба SSH работает на порту 22 как отдельный демон, порождающий новые процессы **sshd** (аналогично серверу Apache) для новых подключений. Чтобы включить поддержку сервера SSH, добавьте следующую строку в файл **/etc/rc.conf** (если ее там еще нет) и перезагрузите систему (или просто введите команду **sshd**):

```
sshd_enable="YES"
```

Клиент SSH заменяет программу Telnet. Нужно просто использовать команду **ssh** вместо **telnet** в командной строке:

```
#  ssh  stripes.somewhere.com
```

Программа **Ssh** сама запросит пароль, предполагая, что имя пользователя удаленной машины совпадает с именем локального. Другое имя пользователя можно задать несколькими способами:

```
# ssh stripes.somewhere.com -l frank
#  ssh  frank@stripes.somewhere.com
```

Программа ***ssh*** *устанавливает шифруемое подключение и передает* регистрационную информацию на сервер безопасным образом. С этого момента она работает как обычная реализация Telnet, — с точки зрения пользователя нет никакой разницы. Именно так пользователь операционных систем FreeBSD, Linux, UNIX или Mac OS X должен подключаться к вашей системе.

Пользователям настольных клиентских операционных систем типа Windows или классической Mac OS придется еще немного потрудиться. На этих платформах нет клиентских программ SSH для командной строки, но есть ряд отличных графических программ эмуляции терминала, включающих возможности как Telnet, так и SSH. У пользователей Windows есть программы SecureCRT (от Van Dyke, **http://www.vandyke.com**) и SSH (от SSH Communications Security, **http://www.ssh.com**), а пользователи Mac могут использовать программы NiftyTelnet/SSH или MacSSH. Программы для Windows — обычно коммерческие, а программы для Mac чаще распространяются свободно.

Необходимо убедить пользователей перейти на использование протокола SSH вместо Telnet. Можно сообщить им, опубликовав правила использования серверов, что, учитывая необходимость защиты, рекомендуется использовать протокол SSH. Однако это не гарантирует, что они будут его использовать, и администратор по-прежнему должен учитывать угрозу перехвата атакующим подключения одного из пользователей, решившего протокол SSH не использовать. Более трудоемкий, но результативный подход состоит в полном отключении службы Telnet и требовании использовать вместо нее службу SSH. Чтобы отключить службу Telnet, закомментируйте строку **telnetd** в файле **/etc/inetd.conf**:

```
#telnet stream tcp nowait root /usr/libexec/telnetd telnetd
```

Затем перезапустите процесс **inetd**:

```
# killall -HUP inetd
```

ПРИМЕЧАНИЕ

Протокол SSH имеет две популярные разновидности: SSH1 и SSH2. Операционная система FreeBSD поддерживает обе, но протокол SSH1 хуже продуман и потенциальная вероятность выявления в нем уязвимых мест защиты выше, чем в SSH2. Можно отключить поддержку SSH1, добавив следующую строку в файл **/etc/ssh/ssh_config**:

```
Protocol 2
```

Учтите, однако, что не все клиентские реализации SSH в полном объеме поддерживают протокол SSH2. Не отключайте протокол SSH1, если не уверены, что это можно делать.

Защита служб электронной почты (POP3 и IMAP)

Вероятно, еще более рискованной с точки зрения похищения паролей, чем использование службы Telnet, является поддержка текстовых протоколов POP3 и IMAP (причем их и защитить сложнее). Если почтовые клиенты пользователей настроены для подключения к серверу, например, раз в пять минут, для проверки наличия новых сообщений, легко перехватываемые передачи регистрационного имени и пароля происходят при каждом таком подключении, что повышает вероятность перехвата пароля (это связано с предсказуемой периодичностью посылки этими службами конфиденциальных данных). Если вы внедрили на сервере протокол SSH вместо Telnet, в ваших же интересах обеспечить защиту такого же уровня для служб электронной почты.

В главе 25 описывалась защита программы **qpopper** путем использования встроенных в ОС FreeBSD средств SSL (Secure Sockets Layer — уровень защищенных сокетов) для шифрования подключений по протоколу POP3. Такой же метод можно при-

менить и для протокола IMAP с помощью пакета программ IMAP-UW. Включить поддержку SSL в системе IMAP-UW можно путем получения сертификата в бюро сертификации, как было описано в главе 25. Если у сайта уже есть сертификат, полученный для другой службы (например, **qpopper**), можно использовать его и для системы IMAP-UW. Подробнее об этом см. в документации по системе IMAP-UW по адресу **http://www.washington.edu/imap/**.

Альтернативный способ шифровать протоколы POP3 и IMAP, централизованно управляя сертификатами SSL и не опираясь на встроенную поддержку SSL каждой из служб, — использование программы **stunnel**. Эта программа доступна в наборе портированных приложений (в каталоге **/usr/ports/security/stunnel**) и позволяет настроить универсальный тоннель SSL для любой службы в системе. Чаще всего, ее используют для шифрования информации, передаваемой по протоколам POP3 и IMAP. Если установить эту программу из набора портированных приложений, стандартный сценарий запуска (**/usr/local/etc/rc.d/stunnel.sh.sample**) запускает процесс прослушивания на портах 993 (для протокола IMAP) и 995 (для протокола POP3), которые являются общепринятыми портами для защищенных версий соответствующих протоколов, как указано в файле **/etc/services**.

ПРИМЕЧАНИЕ

Не забудьте переименовать файл **stunnel.sh.sample** в **stunnel.sh**, как было описано в главе 11. Суффикс **.sample** задается, чтобы администратор обязательно просмотрел содержимое сценария и убедился в корректности путей к файлам сертификатов (с расширением **.pem**).

При использовании программы **stunnel** все равно необходимо получить сертификат, как и в случае с **qpopper** и **IMAP-UW**. Сертификат для программы **stunnel** надо поместить в файл **/usr/local/etc/stunnel.pem**. После этого ваши клиенты протоколов POP3 и IMAP должны получить возможность подключаться к соответствующим портам (993 вместо 143 для IMAP и 995 вместо 110 для POP3) для установки безопасного подключения.

Проблема в том, что не все клиенты электронной почты поддерживают шифрование на базе SSL для протоколов POP3 и IMAP. Некоторые популярные клиенты, например Microsoft Outlook, поддерживают, другие — не поддерживают или поддерживают шифрование только одного из протоколов или эта поддержка неполна и необязательна. Требование использовать шифрование на базе SSL может означать требование сменить программу работы с электронной почтой, а это многим не нравится. Учтите также, что программа **stunnel** не заменяет протоколы POP3 или IMAP — она дополняет, добавляя общую поддержку SSL, любую выбранную службу. Это означает, что обычные службы POP3 и IMAP должны быть включены, — их нельзя удалить из файла **/etc/inetd.conf**. Если необходимо обязать клиентов подключаться только по безопасным каналам, придется использовать систему IPFW (как будет описано далее в этой главе), чтобы запретить подключения к соответствующим портам с любых хостов, кроме **localhost**.

Защита FTP-сервера

Служба FTP, описанная в главе 27, — еще одна служба с передачей информации в явном виде. Она выполняет аутентификацию по паролю и поэтому может быть взломана злоумышленником, перехватывающим пакеты. FTP подвергает систему почти такому же риску, как и служба Telnet, поскольку часто используется, например, для

выгрузки на сервер Web-страниц, но не регулярно, с предсказуемыми интервалами, как POP3 или IMAP. Это делает ее менее рискованной, но не снимает проблемы защиты.

К счастью, защищенный вариант FTP обеспечивается протоколом SSH. Если в системе включена поддержка протокола SSH (как это сделать, было описано выше), защищенный вариант FTP уже доступен. Зашифрованные сеансы FTP работают по каналу SSH: клиент SSH устанавливает терминальное соединение, запускает серверную программу **/usr/libexec/sftp-server** и открывает необходимые подключения к клиенту по шифруемым каналам. Затем защищенный клиент FTP работает аналогично обычной программе FTP, прозрачно для пользователя.

В ОС FreeBSD работу клиентской части безопасного сеанса FTP обеспечивает встроенная программа **sftp**, входящая в состав пакета OpenSSH. В ОС Windows реализацию защищенного клиента FTP, работающего с ОС FreeBSD, предлагает пакет SSH Communications Security. Ранее упоминавшиеся клиенты Mac OS также поддерживают возможности защиты FTP.

Альтернативный способ передавать файлы по защищенному каналу состоит в использовании программы **scp**. Она позволяет копировать файлы на удаленный сервер и с удаленного сервера, используя аутентификацию учетной записи, и во многом аналогична программе **rcp** (подробнее см. на страницах справочного руководства **man rcp** и **man scp**), но передает данные по зашифрованному тоннелю SSH. Некоторые клиенты SSH, например NiftyTelnet/SSH для компьютеров Mac, поддерживают этот метод передачи файлов.

Чтобы переслать с помощью программы **scp** файл с локальной машины на удаленный сервер SSH, используется команда следующего вида:

```
# scp file.txt stripes:
frank@stripes's password:
file.txt                100%  |*****************************|   511   00:00
```

После имени удаленного хоста указывается двоеточие (**:**), причем удаленный хост может быть как исходным, так и целевым в операции копирования. Можно также задавать полные имена файлов, определяющие их точное местонахождение. Это простой способ быстро и безопасно передать файлы, если не требуются разнообразные возможности настоящего FTP-клиента.

Защита сервера Apache

Наконец мы добрались до протокола HTTP. Безопасность протокола HTTP жизненно важна для электронной торговли, когда защищать надо номера кредитных карточек и счетов клиента, а не регистрационные имена и пароли. Это не менее существенно, особенно если бизнес зависит от уверенности клиентов в соблюдении конфиденциальности передаваемой информации.

Защита протокола HTTP была одним из первых масштабных применений системы SSL, и хотя эта система защиты сегодня принята практически для всех популярных служб, интеграция ее в сервер Apache по-прежнему имеет признаки неорганизованности, столь свойственные SSL в первые годы развития. Существуют две различные, никак не связанные между собой версии сервера Apache с поддержкой SSL, разрабатываемых одновременно: Apache-SSL и **mod_ssl**. Обе они включают библиотеки и средства OpenSSL системы FreeBSD, но поддерживаются разными группами и решают различные проблемы.

ПРИМЕЧАНИЕ

Защищенная и обычная текстовая версия протокола HTTP предназначены для совместной работы. Обслуживать все запросы HTTP через систему SSL — не очень хорошая и редко применяемая идея. Частично это связано с производительностью (скорость работы Web-сайта с большим количеством обращений резко снизится, что связано с дополнительными вычислениями при шифровании каждой страницы и изображения, независимо от наличия в них секретной информации), а частично — с неудобством и непривычностью такого способа взаимодействия. Большинство общедоступных данных Web не надо шифровать. Но переключение в защищенный режим при входе клиента на страницу покупки или в форму для сбора информации убеждает пользователей в том, что они вошли в более защищенную зону. Помните: вы должны удовлетворить ожидания пользователей и обеспечить конфиденциальность, а не только предоставить данные.

Система Apache-SSL

"Официальная" защищенная реализация сервера Apache, система Apache-SSL поддерживается собственно группой Apache Group, и имеет более ограниченный набор возможностей по сравнению с **mod_ssl**. Эту реализацию используют скорее благодаря стабильности и производительности, чем из-за поддержки расширенных возможностей. Разработка системы Apache-SSL сейчас ведется не особенно активно в основном из-за строго контролируемого набора возможностей и отсутствия широко известных ошибок.

Двоичный модуль системы Apache-SSL называется **httpsd**, а не **httpd**. Идея в том, чтобы запускать стандартный демон **httpd** для обслуживания обычных запросов HTTP к порту 80, и демон **httpsd** для обслуживания зашифрованных запросов к порту 443. Конечно, это означает, что необходимо установить и обычную версию Apache, без поддержки SSL.

Систему Apache-SSL можно установить из набора портированных приложений (каталог **/usr/ports/www/apache13-ssl**), а ее официальный Web-сайт находится по адресу **http://www.apache-ssl.org**.

Сервер Apache с модулем mod_ssl

Более полная и активно развивающаяся реализация протокола SSL для HTTP по сравнению с Apache-SSL, — **mod_ssl** — представляет собой стандартный модуль Apache, подключающий библиотеку OpenSSL к серверу Apache с использованием современной модульной архитектуры сервера. Эта реализация более прозрачна, чем Apache-SSL, включает намного больше возможностей и более универсальную модель конфигурирования. Например, при установке портированного набора **apache13-modssl** устанавливается один выполняемый файл, **httpd**, как и для обычной версии **apache13**, но файл конфигурации специально настроен так, чтобы при необходимости включить поддержку протокола SSL:

```
<IfDefine SSL>
Listen 80
Listen 443
</IfDefine>
```

К серверу Apache с модулем **mod_ssl** можно добавлять и другие модули, например **mod_perl**, **mod_php** и т.п., из каталога **/usr/ports/www**. Основные отличия модуля **mod_ssl** — богатый и полный набор возможностей, а также простота конфигурирования, хотя он и не настолько быстр и надежен, как система Apache-SSL. Хотя однозначных статистических данных об этом нет.

Сервер Apache с модулем **mod_ssl** можно установить из каталога **/usr/ports/www/apache13-modssl**. Официальный Web-сайт проекта — **http://www.modssl.org**.

Эксплуатация защищенного Web-сервера

Независимо от выбора реализации, Apache-SSL или Apache с модулем **mod_ssl**, поддерживать Web-сервер с защитой станет сложнее, чем без нее. Как уже было сказано, при использовании системы Apache-SSL предполагается, что одновременно работает обычный демон **httpd** для обслуживания обычных подключений по HTTP, передающих информацию в явном виде, и демон **httpsd** для зашифрованных запросов к порту 443. Это означает, что есть отдельная программа, **httpsdctl**, для управления системой Apache-SSL, работающая аналогично программе **apachectl** для обычного сервера Apache, а также имеется файл **httpsd.conf** в каталоге **/usr/local/etc/apache**, помимо файла **httpd.conf**. При установке сервера Apache с модулем **mod_ssl** никаких дополнительных компонентов не будет — все функциональные возможности включены в стандартный набор файлов, рассмотренный в главе 26.

При установке портированных приложений (как **apache13-ssl**, так и **apache13-modssl**) устанавливаются полные деревья каталогов Apache, включая пиктограммы, примеры HTML-страниц, динамические модули и файлы конфигурации. Поэтому необходимо быть внимательным при обновлении параллельно установленных систем Apache-SSL и обычного сервера Apache. Версия с модулем **mod_ssl** представляет собой единый набор каталогов, заменяющий стандартную реализацию сервера Apache, поэтому сопровождение ее намного проще.

Сертификаты OpenSSL гораздо чаще читаются Web-браузером, чем любой другой защищенной службой, поэтому необходимо специально убедиться, что эти сертификаты соответствуют действительности. Если имя хоста в сертификате не соответствует имени хоста сервера или если сертификат не заверен общепризнанным сертификационным бюро (другими словами, если вы сами его заверили), пользователь получит диалоговое окно, показывающее полную информацию о вашем сертификате и запрашивающее подтверждения его приема браузером. Естественно, это может произвести неприятное впечатление на пользователя и лишить его уверенности в безопасности вашего сайта. Даже если сертификат правильно отражает информацию о сайте, пользователь может просмотреть сертификат с помощью средств информирования о защите (Security Information) своего браузера, так что все поля, заданные при генерации запроса на получение сертификата будут видны любой заинтересованной стороне. Помните об этом — информацию сертификата весьма сложно изменить после того, как он заверен.

Плохо написанные сценарии CGI

Передача информации в явном виде — не единственная потенциальная дыра в защите сервера Apache. Не меньше надо беспокоиться о возможности нежелательных действий пользовательских сценариев в системе (как преднамеренных, так и случайных) и уничтожения ими файлов, особенно если модель защиты такова, что локальные пользователи могут не пользоваться доверием. Поскольку многие файлы в корневом каталоге документов сервера Apache принадлежат пользователю **nobody** (в частности, динамически создаваемые вашими же программами, выполняющимися на сервере) и поскольку от имени того же пользователя **nobody** выполняются также и CGI-программы всех пользователей, CGI-программа легко может удалить или изменить файл на сервере, принадлежащий пользователю **nobody**.

Это не так уж невероятно, как может показаться. Достаточно, чтобы CGI-программа удаляла ненужные собственные файлы пользователя, но по ошибке предваряла имена файлов не тем именем каталога. То же самое может произойти с выда-

ющей данные в файл программой, которая может повредить данные другого пользователя. Даже самые заслуженные разработчики CGI-программ раньше попадались в подобную ловушку. Опасность значительно возрастает, если в системе есть злонамеренный пользователь, намеренно создающий деструктивный сценарий для выполнения сервером Apache от имени пользователя **nobody**.

Решение этой проблемы — запускать сервер Apache в программе -"обертке", которая перехватывает CGI-программы пользователей, проверяет их безопасность (убеждаясь в соответствии прав доступа) и запускает их от имени соответствующих пользователей, а не от имени **nobody**. Для этого традиционно использовалась программа **suexec**, поставляемая в составе сервера Apache. Но более простым и гибким решением является другая система, CGIWrap, созданная Натаном Ньюлингером (Nathan Neulinger).

Повышение безопасности сценариев CGI с помощью системы CGIwrap

Система CGIWrap, доступная в наборе портированных приложений в каталоге **/usr/ports/www/cgiwrap**, обеспечивает двойное преимущество: защищает пользователей и корневой каталог документов сервера от угрозы плохо написанных или враждебных CGI-сценариев и позволяет при этом пользовательским CGI-программам записывать файлы, которые сами пользователи смогут изменять или удалять в командном интерпретаторе. Если задуматься, намного больше смысла в выполнении CGI-программы пользователем, которому она принадлежит, а не непривилегированным пользователем **nobody**. В идеальном мире, где программы всегда совершенны и безошибочны, и никто не пытается намеренно уничтожить чужие файлы, именно так и должны работать все Web-серверы.

Однако наш мир не идеален — CGI-программы пишутся с ошибками, и кругом полно хакеров. Пользователю достаточно создать CGI-программу, принадлежащую **root**, и поместить ее в свой каталог, ожидая, пока она не будет выполнена от имени владельца, суперпользователя, и вся ее разрушительная мощь не обрушится на выбранные в системе файлы.

Система CGIWrap защищает сервер от подобных проблем. Это решение, конечно, — не полное и не идеальное, но оно снимает подавляющее большинство угроз защите, связанных с пользовательскими сценариями CGI, настолько, что администратор может направить свои усилия по защите на другие области. Проверяя перед выполнением все пользовательские программы CGI на предмет защиты и запуская каждую программу от имени владельца, система переносит все неотвратимые угрозы, которые несут некорректно написанные сценарии CGI, с сервера на владельца сценария.

При установке системы CGIWrap из каталога портированных приложений программа **cgiwrap** (скомпилированная двоичная программа) попадает в каталог **cgi-bin** верхнего уровня, **/usr/local/www/cgi-bin**. CGIWrap не работает в качестве оболочки сервера Apache, как система **suexec**, а должна вызываться пользователями непосредственно, путем указания URL следующего вида:

```
http://www.somewhere.com/cgi-bin/cgiwrap/frank/myscript.cgi
```

Или в директиве включения на стороне сервера:

```
<!--#include virtual="/cgi-bin/cgiwrap/frank/myscript.cgi"-->
```

В результате выполняется программа **myscript.cgi** из каталога **/home/frank/public_html/cgi-bin**. Все CGI-программы пользователя должны устанавливаться в его

каталоге **public_html/cgi-bin**. Если этот каталог отсутствует, его необходимо создать. CGI-программы из других каталогов не будут обрабатываться системой CGIWrap, поэтому важно отключить возможность выполнения CGI за пределами дерева каталогов **DocumentRoot**-сервера, убедившись, что в блоках, определяющих каталоги пользователей, нет директивы **Options +ExecCGI**.

Официальная Web-страница системы CGIWrap с дополнительной информацией находится по адресу **http://cgiwrap.unixtools.org/**.

Профили защиты системы и защита ядра (securelevel)

Ядро ОС FreeBSD может работать с пятью различными уровнями защиты, от -1 до 3, которые задаются с помощью опции **kern_securelevel** в файле **/etc/rc.conf**. Каждый из этих уровней соответствует профилю, управляющему параметрами вроде возможности замены ядра на диске, загрузки и выгрузки модулей ядра, установки и изменения определенных прав доступа к файлам и флагов, монтирования файловых систем по требованию, а также отключения или изменения утилит типа IPFW (встроенный брандмауэр, который мы вскоре рассмотрим). Как было описано в главе 17, значение опции **securelevel** по ходу работы можно только увеличить — для его снижения необходима перезагрузка. Дополнительную информацию по защите ядра можно найти на странице справочного руководства **man securelevel**.

В ОС FreeBSD есть также и другой набор профилей многоуровневой сетевой защиты. Вы видели его в ходе установки и можете увидеть снова в программе **/stand/sysinstall**, выбрав пункты меню "Configure", а затем "Security". Полученное меню позволяет выбрать один из четырех общесистемных профилей защиты: Low, Medium, High и Extreme. Эти профили управляют возможностью запуска служб типа **sendmail**, **sshd** и **inetd** и примерно соответствуют уровням защиты ядра. В табл. 29.1 для каждого профиля представлены устанавливаемые им опции в файле **/etc/rc.conf**:

Таблица 29.1. Общесистемные профили защиты

Имя профиля	*Установки в файле /etc/rc.conf*
Low	sendmail_enable="YES"
	sshd_enable="YES"
	portmap_enable="YES"
	inetd_enable="YES"
Medium	sendmail_enable="YES"
	sshd_enable="YES"
	inetd_enable="YES"
High	kern_securelevel="1"
	kern_securelevel_enable="YES"
	sendmail_enable="YES"
	sshd_enable="YES"
	portmap_enable="NO"

Имя профиля	*Установки в файле /etc/rc.conf*
	nfs_server_enable="NO"
	inetd_enable="NO"
Extreme	kern_securelevel="2"
	kern_securelevel_enable="YES"
	sendmail_enable="NO"
	sshd_enable="NO"
	portmap_enable="NO"
	nfs_server_enable="NO"
	inetd_enable="NO"

Как и можно было ожидать, профиль "Extreme" ограничивает систему почти до уровня бесполезности. Уровень защиты ядра **securelevel** устанавливается равным 2, вследствие чего ядро нельзя изменять (с помощью модулей) или устанавливать новое без перезагрузки в однопользовательский режим, а единственный способ смонтировать или демонтировать файловую систему — явно, с помощью команд **mount** и **umount** (неявное монтирование по требованию, обеспечиваемое демоном **amd**, не допускается). Кроме того, демон **inetd**, система **sendmail**, демон **sshd**, сервер NFS и другие службы отключены.

Профиль "High" — менее ограничительный. Он задает ядру уровень защиты **securelevel** 1, тем самым упрощая монтирование файловых систем, но ядро тоже нельзя изменять. Демоны **sendmail** и **sshd** включены, но все остальные службы, упомянутые в профиле "Extreme", отключены.

Конечно, поскольку профили защиты работают исключительно путем установки опций в файле **/etc/rc.conf**, вы можете при необходимости комбинировать представленные в табл. 29.1 установки, создавая профиль защиты, соответствующий выбранной модели защиты системы. Не следует включать службу без необходимости. Если она не используется по назначению, то может оставаться бесполезной и безвредной; однако в ее защите может быть обнаружено уязвимое место, открывающее систему для взлома. Предохраняйтесь, по возможности.

Использование брандмауэра

Никто не может отрицать, что *брандмауэры*, или машины, работающие в качестве маршрутизаторов с фильтрами, становятся все более важной и даже незаменимой частью сети, подключенной к Internet. Легкодоступные средства взлома, применяемые изнывающими от безделья "юными хакерами" ("script kiddies"), требуют, помимо отказа от определенных служб и постоянного слежения за публикациями, посвященными защите, дополнительного уровня защиты. Необходим универсальный щит на уровне ядра, предотвращающий вообще доступ к определенным портам системы, с определенных хостов или по некоторым протоколам. Брандмауэры, в особенности брандмауэр IPFW, входящий в состав ОС FreeBSD, удовлетворяют эту потребность.

Брандмауэр может предотвратить подавляющее большинство случайных атак, пропуская только пакеты, которые администратор счел допустимыми. Однако еще одна аксиома сетевой защиты гласит, что даже самый дорогой и надежный брандмауэр становится бесполезным в случае неправильного конфигурирования. Причина неэф-

фективности брандмауэров — результат неправильного конфигурирования; а не их качество. Постоянно обновляемые и хорошо продуманные правила защиты не заменит никакой "компетентный" брандмауэр, поэтому не обольщайтесь, что, купив более дорогой брандмауэр, сможете решить все проблемы защиты. Именно такое представление приводит к столь многочисленным сегодня взломам защиты в Internet.

С помощью брандмауэра можно делать две вещи. Во-первых, фильтровать пакеты по заданным критериям, отвергая нежелательную информацию на уровне ядра (до того, как она достигнет критических служб системы). Во-вторых, можно вести учет, сохраняя статистическую информацию об использовании системы и следя за тем, какие объемы информации и откуда поступают. Система IPFW делает и то, и другое; ее можно запускать непосредственно на своей машине с ОС FreeBSD или использовать на системе, работающей как шлюз, защищающий несколько хостов локальной сети. На рис. 29.1 представлен как раз последний случай, когда машина с ОС FreeBSD и тремя сетевыми картами работает в качестве шлюзового маршрутизатора (см. главу 28), передающего пакеты между внутренней локальной сетью (LAN), "демилитаризованной зоной " (DMZ) и внешними сетями (WAN).

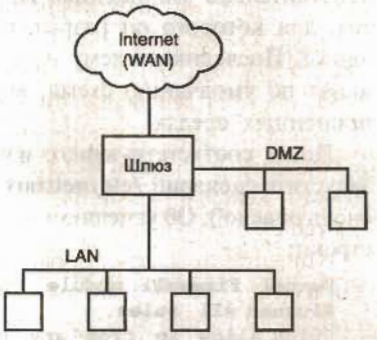


Рисунок 29.1.
Диаграмма шлюзового маршрутизатора, обеспечивающего функции брандмауэра. Показаны интерфейсы LAN, DMZ и WAN

ПРИМЕЧАНИЕ

"Демилитаризованная зона" — это сеть, непосредственно доступная для информации из глобальной сети, а не из локальной. Особенно полезная в случае преобразования IP-адресов в локальной сети (как было показано в главе 22) демилитаризованная зона представляет собой сеть для предприятия или поставщика услуг Internet, в которую помещаются машины с реальными адресами, например Web-серверы, шлюзы электронной почты и другие хосты, которые должны быть доступны из общей сети Internet.

Зона DMZ может быть защищена правилами брандмауэра на шлюзовом маршрутизаторе, а может и не защищаться. Защита обычно желательна, но в некоторых специальных случаях требуется, чтобы демилитаризованная зона была освобождена от общих правил брандмауэра локальной сети. Это зависит от конфигурации и назначения сети.

Включение поддержки брандмауэра

Система IPFW не поддерживается в ядре **GENERIC**. Имеется ряд опций, которые необходимо скомпилировать в специализированном ядре для ее поддержки, как было описано в главе 17: Это опции **IPFIREWALL**, **IPFIREWALL_VERBOSE** и **IPFIREWALL_VERBOSE_LIMIT=10**. Однако для использования системы IPFW вовсе не обязательно создавать новое ядро — она доступна в виде модуля ядра, автоматически загружаемого сценарием **/etc/rc.network**. Чтобы включить поддержку брандмауэра без перестройки ядра, добавьте следующие строки в файл **/etc/rc.conf**:

```
firewall_enable="YES"
firewall_type="open"
```

Если не указать, что брандмауэр должен быть открытого типа (**firewall_type="open"**), система IPFW запустится с единственным стандартным правилом с индексом 65535 (максимальная защита), запрещающим передачу любых IP-пакетов. Другими словами, после перезагрузки с включением системы IPFW стандартной конфигурации, машина будет полностью отрезана от сети, и, чтобы исправить это, необходим доступ к физической консоли.

ПРЕДУПРЕЖДЕНИЕ

Очень опасно экспериментировать с системой IPFW при отсутствии доступа к консоли, поскольку можно сделать машину недоступной. Пока вы не ознакомитесь с системой IPFW настолько, что будете точно знать последствия своих действий, всегда убеждайтесь в наличии доступа к консоли на случай, если что-то пойдет не так.

Установка типа брандмауэра **open** изменит стандартные правила так, что все IP-пакеты по умолчанию пропускаются (разрешаются), а не блокируются. Помните, что действительно защищенная система должна запрещать доступ ко всем хостам, кроме тех, для которого он разрешен явно, а не разрешать все, кроме нескольких исключений. Последнюю схему нельзя сделать полностью безопасной. Однако ограничивающая по умолчанию схема, вероятно, избыточна для систем, работающих в уже защищенных средах.

После соответствующего изменения файла **/etc/rc.conf** перезагрузите систему или запустите сценарий **/etc/netstart** с физической консоли (не с удаленного терминала — это очень опасно!). Об успешном включении брандмауэра будут свидетельствовать следующие строки:

```
Kernel firewall module loaded
Flushed all rules.
00100 allow ip from any to any via lo0
00200 deny ip from any to 127.0.0.0/8
00300 deny ip from 127.0.0.0/8 to any
65000 allow ip from any to any
Firewall rules loaded, starting divert daemons:.
```

Чтобы убедиться, что модуль системы IPFW автоматически загружен, можно воспользоваться программой **kldstat**:

```
# kldstat
Id  Refs  Address     Size    Name
 1     3  0xc0100000  355be4  kernel
 2     1  0xc0eee000  6000    ipfw.ko
 3     1  0xc0f19000  12000   linux.ko
```

Теперь доступна в полном объеме команда **ipfw**, позволяющая устанавливать правила пропуска пакетов и просматривать учетную информацию, накопленную системой IPFW.

ПРЕДУПРЕЖДЕНИЕ

При использовании системы NATD для совместного использования подключения к Internet, как описано в главе 28, все равно необходимо строить специализированное ядро, поскольку для системы NATD требуется включение в ядре опции **IPDIVERT**. Подробнее об этом см. в главе 28.

Конфигурирование системы IPFW

Для добавления или удаления правил системы фильтрования и учета пакетов ядра используется команда **ipfw**. Правила строятся по синтаксису, отдаленно напоминающему обычный английский язык; они состоят из действия (например, **deny**), протокола, к которому применяется действие (например, **tcp**), и спецификации адреса, включающей конструкции **from** и **to**. Например, правило, блокирующее пакеты TCP, которые поступают с хоста **badhost.com**, будет иметь вид **deny tcp from badhost.com to any**, а добавить его к брандмауэру ядра можно так:

```
# ipfw add deny tcp from badhost.com to any
```

Возможны и определенные вариации этой конструкции. Вместо адреса можно задавать маску сети — либо в виде шаблона маски (например, **255.255.255.0**), либо с помощью битовой маски CIDR (например, **/24**). Можно также блокировать отдельные порты, а не всю систему, т.е. предотвратить доступ хостов с непредсказуемыми IP-адресами к определенным службам сервера. Такое правило может выглядеть, например, так:

```
# ipfw add deny all from evil.isp.com/16 to www.somewhere.com 80
```

Можно исключать хосты из предшествующих правил, указывая их отдельно в правилах **allow**, например, так:

```
# ipfw add allow all from goodhost.evil.isp.com to
➥www.somewhere.com 80
```

Учтите, что правила системы IPFW проверяются в порядке задания. Каждое правило имеет индекс, обычно на 100 больший, чем у предшествующего, причем индекс можно задавать явно, после ключевого слова **add** в команде **ipfw**. Индексы позволяют задать порядок применения правил. Текущий набор правил можно получить с помощью команды **ipfw -a list**:

```
# ipfw -a list
00100     0      0  allow ip from any to any via lo0
00200     0      0  deny ip from any to 127.0.0.0/8
00300     0      0  deny ip from 127.0.0.0/8 to any
00400     0      0  deny tcp from badhost.com to any
00500     0      0 ipfw add deny all from evil.isp.com/16 to
  ➥www.somewhere.com 80
00600     0      0 ipfw add allow all from goodhost.evil.isp.com to
  ➥www.somewhere.com 80
65000  1214  79688  allow ip from any to any
65535     1     40  deny ip from any to any
```

Индекс выдается в первом столбце. Правило с индексом 65535, как уже было сказано, — это стандартное запрещающее правило, которое отвергает все пакеты, не разрешенные предыдущими правилами. Поскольку был задан открытый тип брандмауэра, перед ним помещено разрешающее передачу любых пакетов правило **allow**, но тоже с достаточно большим индексом, так что оно должно проверяться после всех остальных правил, добавленных при обычном использовании.

Во втором и третьем столбцах представлена статистика использования, показывающая количество пакетов и байтов, соответствовавших каждому правилу. Так можно понять, эффективны ли заданные правила.

В файле **/etc/rc.conf** можно задавать различные типы брандмауэров. Каждое ключевое слово имеет особое значение, представленное в табл. 29.2. Точные определе-

ния этих профилей можно получить, разобрав код сценария командного интерпретатора **/etc/rc.firewall**.

Таблица 29.2. Доступные типы брандмауэров

Ключевое слово	*Значение*
open	Разрешает прохождение всех входящих и исходящих пакетов
closed	Запрещает все пакеты IP, кроме проходящих через интерфейс закольцовывания (lo0)
client	Устанавливает правила для защиты только данной машины
simple	Устанавливает правила, предназначенные для защиты всей сети
unknown	Не загружает никакие правила, кроме стандартного запрещающего с индексом 65535
<имя файла>	Загружает правила из файла с указанным именем

Для ваших целей может подойти "готовый" профиль IPFW, например **client** или **simple**. Вероятно, однако, что при развитии системы потребуется специализированная конфигурация. Специализированный набор правил необходимо поместить в выбранный файл конфигурации; например **/etc/firewall.conf**. Укажите необходимый набор правил в этом файле, не задавая саму команду **ipfw**:

```
add deny tcp from badhost.com to any
add deny all from evil.isp.com/16 to www.somewhere.com 80
add allow all from goodhost.evil.isp.com to www.somewhere.com 80
add 65000 allow all from any to any
```

Теперь измените тип брандмауэра (значение **firewall_type**) в файле **/etc/rc.conf**:

```
firewall_type="/etc/firewall.conf"
```

При следующей перезагрузке или запуске файла **/etc/netstart** правила из файла **/etc/firewall.conf** будут загружены с индексами 100, 200, 300 и т.д. Правило **allow all** с индексом 65000 задает стандартное поведение — пропускать, а не отвергать пакеты. Если хотите, задайте и его.

Дополнительную информацию о системе IPFW можно найти на странице справочного руководства **man ipfw** и в руководстве FreeBSD Handbook по адресу **http://www.freebsd.org/handbook**.

Предотвращение вторжений и взломов

Брандмауэры, правила задания паролей и шифрование являются существенной защитой системы от неавторизованного доступа. Но всего этого не достаточно для защиты от действительно целеустремленного хакера, у которого есть "rootkit" или другое средство, разработанное для использования одного из известных уязвимых мест в той или иной службе. Имеется множество средств защиты, выходящих далеко за рамки возможностей простого брандмауэра. Они динамически блокируют подозрительные хосты, следят за вторжениями и избирательно контролируют доступ каждого хоста к отдельным службам. Давайте рассмотрим некоторые из этих средств.

Использование PortSentry

PortSentry, предлагаемый компанией Psionic Software, — это демон, контролирующий входящие сетевые пакеты, прослушивающий указанные порты и выявляющий пакеты, которые могут свидетельствовать о возможности сканирования портов — предварительной атаки, в ходе которой взломщик ищет поддерживаемые системой службы, чтобы попытаться их взломать. Выявив такие пакеты, демон PortSentry блокирует доступ соответствующего хоста к системе, перенаправляя его обращения в "черную дыру" или добавляя для него правило системы IPFW, так что все дальнейшие попытки подключения с этого хоста отвергаются. Демон PortSentry контролирует как TCP-, так и UDP-пакеты; он динамически строит таблицу запрещенных подключений, используемую как антитело против вируса, и реагирует на подозрительные действия превентивно, блокируя их прежде, чем они смогут нанести вред.

Являясь программой с открытым исходным кодом, PortSentry может быть установлена из каталога портированных приложений (**/usr/ports/security/portsentry**). Двоичная программа **portsentry** устанавливается в каталог **/usr/local/bin**, а в качестве конфигурационного используется файл **/usr/local/etc/portsentry.conf**. Откройте этот файл в любом текстовом редакторе — для правильной работы демона PortSentry его необходимо отредактировать.

Прежде всего необходимо решить, какой набор портов должен контролировать демон PortSentry. Имеется три готовых набора в виде пар, начинающихся с **TCP_PORTS** и **UDP_PORTS**. Первый набор — достаточно большой; он предназначен для реализации очень строгих правил, по которым любой подозрительный удаленный доступ к порту автоматически вызывает срабатывание блокирующего правила. Второй набор — более либеральный, а третий — минимальный, задающий слежение только за портами, к которым удаленный пользователь не должен подключаться, если только он не выполняет сканирование портов или не пытается организовать атаку. Второй, либеральный набор ("если вы хотите знать"), используется по умолчанию. Чтобы перейти на любой из остальных двух наборов, закомментируйте этот набор и уберите комментарий с того набора, который необходимо задействовать. При необходимости можно создать собственный набор. Убедитесь, что в соответствующий список не входят базовые службы! Например, порт 143 указан во всех трех готовых наборах, но 143 — это порт для протокола IMAP. Если вы поддерживаете службы IMAP, не забудьте удалить порт 143 из списка. В противном случае оставьте этот порт в списке — это предотвратит попытки воспользоваться недостатками в защите протокола IMAP.

Затем необходимо выбрать метод блокировки подозрительных хостов. Это можно делать одним из двух способов: с помощью системы IPFW (если она задействована, как было описано ранее в данной главе) или с помощью маршрутов в никуда (black-hole routes) (если система IPFW не используется). Выбор метода состоит в снятии комментария с одной строки, начинающейся ключевым словом **KILL_ROUTE** и задающей команду системы, которую демон PortSentry должен использовать для блокировки подозрительного хоста.

Для системы, в которой работает IPFW, раскомментируйте строку, содержащую вызов **/sbin/ipfw**, например:

```
# Те, кто использует ОС FreeBSD (и другие совместимые с ней
# системы), могут также воспользоваться встроенными функциями
# брандмауэра.
KILL_ROUTE="/sbin/ipfw add 1 deny all from $TARGET$:255.255.255.255 to any"
```

Если система IPFW не задействована, используйте метод маршрутов в никуда, который, несмотря на разные мнения по этому поводу, не менее действенен, чем использование системы IPFW:

```
# FreeBSD (не очень хорошо протестирован.)
KILL_ROUTE="route add -net $TARGET$ -netmask 255.255.255.255 127.0.0.1 -
↪blackhole"
```

После установки метода блокирования демон PortSentry готов к работе. Однако на момент написания этой главы в состав портированного приложения PortSentry не входит сценарий автоматического запуска. Можете использовать сценарий, представленный в листинге 29.1 (доступен на поставляемом с книгой компакт-диске в файле **portsentry.sh**). Проверьте, установлено ли для него право на выполнение и скопируйте его в каталог **/usr/local/etc/rc.d**, чтобы демон PortSentry запускался при каждой перезагрузке системы.

Листинг 29.1. Пример файла запуска PortSentry.

```
#!/bin/sh

PORTSENTRY="/usr/local/bin/portsentry"

case "$1" in
    start)
        ${PORTSENTRY} -tcp && echo " Starting PortSentry TCP mode..."
        ${PORTSENTRY} -udp && echo " Starting PortSentry UDP mode..."
        ;;
    stop)
        killall `basename ${PORTSENTRY}`
        ;;
    *)
        echo ""
        echo "Usage: `basename $0` { start | stop } "
        echo ""
        ;;
esac
```

СОВЕТ

При работающем демоне PortSentry с помощью команды sockstat можно узнать, какие порты он прослушивает:

```
# sockstat
USER    COMMAND  PID   FD PROTO    LOCAL ADDRESS   FOREIGN ADDRESS
root    portsent 2432   0 udp4     *:1             *:*
root    portsent2432    1 udp4     *:7             *:*
root    portsent2432    2 udp4     *:9             *:*
root    portsent2432    3 udp4     *:69            *:*
root    portsent2432    4 udp4     *:161           *:*
root    portsent2432    5 udp4     *:162           *:*
...
```

При каждом выявлении попытки атаки соответствующий хост и порты, которые он сканировал, что вызвало срабатывание системы, регистрируются в файле **/usr/local/etc/portsentry.blocked.tcp** для атак с использованием протокола TCP и в файле **/usr/local/etc/portsentry.blocked.udp** для атак с помощью протокола UDP. Так, демон PortSentry следит за тем, чтобы уже заблокированные не блокировались повторно. Эти файлы также позволяют понять, действия каких хостов были перехвачены системой выявления атак. Эти файлы автоматически очищаются при каждом запуске де-

мона PortSentry. Учтите, что при перезагрузке исчезают как дополнительные правила системы IPFW, так и записи в таблице маршрутизации, добавленные по ходу работы, так что однажды заблокированный хост может снова получить доступ к машине после перезагрузки, по крайней мере до следующей попытки просканировать ее порты.

СОВЕТ

При использовании метода **IPFW KILL_ROUTE** текущие блокирующие правила можно получить с помощью команды **ipfw -a list**:

```
# ipfw -a list
00001     1     44  deny  ip  from  209.237.26.165  to  any
```

При использовании маршрутов в никуда контролировать работу можно с помощью команды **netstat -rn**:

```
# netstat -rn
Routing tables

Internet:
Destination          Gateway          Flags     Refs    Use     Netif  Expire
...
209.237.26.165/32  127.0.0.1  UGScB       0 0     lo0
```

Флаг **B** отмечает маршрут в никуда ("black-hole" route), по которому пакеты просто отвергаются.

СОВЕТ

При каждом выявлении атаки в файл **/var/log/messages** добавляется строка, показывающая, что предпринял в ответ на нее демон PortSentry:

```
Jun 2 23:50:56 stripes portsentry[2430]: attackalert: Connect from host:
209.237.26.165/209.237.26.165 to TCP port: 1
Jun 2 23:50:56 stripes portsentry[2430]: attackalert: Host 209.237.26.165
has been blocked via wrappers with string: "ALL: 209.237.26.165"
Jun 2 23:50:56 stripes portsentry[2430]: attackalert: Host 209.237.26.165
has been blocked via dropped route using command: "/sbin/ipfw add 1 deny
all from 209.237.26.165:255.255.255.255 to any"
```

Еще одно средство от создателей демона PortSentry — программа Logcheck (доступная в каталоге **/usr/ports/security/logcheck**) — позволяет анализировать этот и другие журнальные файлы и посылать администратору ежедневный отчет о любых необычных событиях или выявленных атаках.

ПРЕДУПРЕЖДЕНИЕ

Как и в случае системы IPFW, будьте внимательны при тестировании демона PortSentry. Очень легко настроить его так, чтобы система, с помощью которой вы управляете своей машиной с ОС FreeBSD, оказалась заблокированной, например, при попытке подключиться с помощью Telnet к порту 1, чтобы узнать, что получится. В результате ваш хост окажется заблокированным, и все остальные попытки подключения по завершении времени ожидания окончатся неудачей. Придется подключаться с другого хоста или с физической консоли, чтобы удалить добавленное по ошибке правило с помощью команды **ipfw delete 1** либо **route delete <IP-адрес>/32**.

Если необходимо защитить определенные хосты (например, ваши собственные машины) от блокирования демоном PortSentry, добавьте их IP-адреса в файл **/usr/local/etc/portsentry.ignore**.

Использование файла /etc/hosts.allow

Файл **/etc/hosts.allow** позволяет блокировать доступ определенных хостов к указанным службам системы. Его можно использовать как неавтоматическую версию

демона PortSentry. Можно задать блок правил для службы так, что каждое правило применяется к определенному IP-адресу или набору адресов и либо разрешает, либо запрещает соответствующим хостам доступ к службе. Вот пример блока правил из стандартного файла **/etc/hosts.allow**, в который добавлено несколько дополнительных строк, иллюстрирующих допустимые синтаксические конструкции:

```
sendmail  : localhost : allow
sendmail  : .nice.guy.example.com : allow
sendmail  : .evil.cracker.example.com : deny
sendmail  : 231.21.15.0/255.255.255.0 : deny
sendmail  : 12.124.231. : deny
sendmail  : ALL : allow
```

Поскольку работа системы IPFW уже рассмотрена, формат этих правил понять несложно. Правило состоит как минимум из трех полей: служба (задаваемая именем соответствующего процесса), имя или IP-адрес обращающегося хоста и предпринимаемое действие (или несколько действий, если полей — четыре и более). Первые два столбца могут содержать списки (несколько записей, разделенных пробелами), а столбец хостов может задаваться во множестве форматов: значение, начинающееся точкой, позволяет задать целую подсеть системы DNS, а заканчивающееся точкой — подсеть на базе IP-адреса. Указав IP-адрес и после косой черты маску сети, можно задать сеть. Блок должен заканчиваться "стандартным" правилом, определяющим, должен ли быть разрешен или запрещен доступ к службе во всех остальных случаях. В общем случае, если уж служба поддерживается, значит, это действительно нужно, поэтому стандартное правило будет "разрешающим".

С помощью файла **/etc/hosts.allow** можно делать вещи и поинтереснее, задавая другие действия, помимо **"allow"** и **"deny"**. Можно сделать так, чтобы при неавторизованном доступе к службе администратору посылалось сообщение электронной почты или запускалась программа, выполняющая ту или иную встречную проверку уязвимости соответствующего удаленного хоста (хотя это, вероятно, не лучшая идея). При каждом срабатывании правила можно выполнить любую команду командного интерпретатора. Стандартное правило для службы **fingerd** показывает пример такой конфигурации:

```
fingerd  : ALL \
         : spawn (echo Finger. | \
           /usr/bin/mail -s "tcpd\ : %u@%h[%a] fingered me!" root) & \
         : deny
```

Коды **%u**, **%h** и **%a** и дополнительные опции конфигурации описаны на страницах справочного руководства **man 5 hosts_access** и **man hosts_options**.

Использование системы Tripwire

Особняком от непосредственного блокирования подозрительных хостов стоит задача выявления вторжения. Достаточно коварный хакер сможет обойти защиту, независимо от того, насколько тщательно она организована. Если это произошло, необходимы средства, позволяющие понять, насколько далеко зашло вторжение и не *нанес ли он вред системе. Если в систему вторглись*, хотелось бы узнать об этом сразу и оценить реальный ущерб от вторжения.

Популярным средством для решения этой задачи является система Tripwire, доступная в каталоге **/usr/ports/security/tripwire**. Это средство вычисляет запись аутентичности для всех программ в системе, сравнивая каждую из них ежедневно с контрольной записью аутентичности, созданной при первом запуске сразу после

установки. При выявлении любых отличий (например, если случайно изменился размер выполняемого файла **sshd** или произошло изменение его содержимого или метаданных, система Tripwire уведомит об этом администратора по электронной почте. Это позволяет оценить, была ли взломана система.

При первоначальной установке система Tripwire строит начальную базу данных "отпечатков" программ в ходе выполнения фазы **make install** и записывает ее в файл в каталоге **/var/adm/tcheck**. Однако с этим связана потенциальная угроза защите — атакующий, получив доступ к системе и обнаружив наличие базы данных Tripwire, просто изменит ее, так что система не сможет обнаружить его присутствие. Вот почему имеет смысл хранить базу данных не на машине, а на другой системе либо, что наиболее удобно, на дискете.

ОС FreeBSD предлагает простой способ создать такую архивную дискету. Необходимо только поместить дискету в дисковод перед началом фазы **make install**, а затем задать присвоение значения переменной **TRIPWIRE_FLOPPY=YES** в командной строке **make install**:

```
# make install TRIPWIRE_FLOPPY=YES
...
###  Фаза 3: Создание базы данных с информацией о файлах
###
### Предупреждение: база данных помещается в файл
###                 ./databases/tw.db_stripes.somewhere.com.
###                 Не забудьте перенести этот файл и файл
###                 конфигурации на безопасный носитель!
###
###                 (Tripwire ищет этот файл в каталоге
###                 '/var/adm/tcheck/databases'.)
# готовим дискету
/dev/rfd0c: 2880 sectors in 80 cylinders of 2 tracks, 18 sectors
        1.4MB in 5 cyl groups (16 c/g, 0.28MB/g, 32 i/g)
super-block backups (for fsck -b #) at:
32, 632, 1184, 1784, 2336
mount /dev/fd0c /mnt
# переносим базу данных на дискету
# Не забудьте вынуть дискету и защитить ее от записи.
```

Теперь на дискете находится копия начальной базы данных системы Tripwire, а также утилиты **tripwire**, **twcheck** и **gunzip** и копия файла **tw.config** — все, что необходимо для восстановления начальной базы данных с дискеты.

После этого можно организовать запуск программы **tripwire** каждую ночь как периодической задачи, — как это сделать, описано в главе 14. Запущенная без аргументов программа **tripwire** работает в режиме проверки согласованности, сканируя все файлы, указанные в файле **/var/adm/tw.config**, в поисках отличий от базы данных "отпечатков" файлов. Выявив несогласованность, программа сообщает о ней, как в следующем примере, в котором изменилось время модификации файла **/usr/sbin/sshd**:

```
# tripwire
...
### Phase 3: Creating file information database
### Phase 4: Searching for inconsistencies
###
###            Total files scanned:          16803
###                  Files added:            0
```

```
###                Files deleted:          0
###                Files changed:          14321
###
###           After applying rules:
###                Changes discarded:      14320
###                Changes remaining:      1
###
changed:  -r-xr-xr-x  root  197940  (null)  /usr/sbin/sshd
###  Phase 5: Generating observed/expected pairs for changed files
###
### Attr     Observed (what it is)     Expected (what it should be)
### ======= ======================== ============================
/usr/sbin/sshd
  st_mtime:  Sun Jun  3 00:55:51 2001  Sat Apr 28 21:17:19 2001
  st_ctime:  Sun Jun  3 00:55:51 2001  Sat Apr 28 21:17:19 2001
```

Может оказаться, что вы ожидаете отличия этого файла от записанного в базе данных системы Tripwire. Например, после установки обновленной версии демона **sshd**. После любого обновления файлов, контролируемых системой Tripwire, необходимо обновить и базу данных Tripwire, чтобы в ней была отражена новая информация. Это делается с помощью опции **-update**:

```
# tripwire -update /usr/sbin/sshd
```

В результате в текущем каталоге будет создан подкаталог базы данных, содержащий новый файл базы данных (который необходимо перенести в **/var/adm/tcheck/databases**) и резервную копию старой базы данных. Эти базы данных не сжаты; может потребоваться сжать их с помощью программы **gzip** и скопировать на дискету Tripwire:

```
# gzip databases/tw.db_stripes.somewhere.com
# mount /dev/fd0 /floppy
# cp databases/tw.db_stripes.somewhere.com /floppy
# umount /floppy
```

ОТДЕЛЬНОЕ ХРАНЕНИЕ ДАННЫХ

Идея хранения базы данных системы Tripwire на дискете связана с принципом пространственного промежутка (air gap) — принципом защиты, состоящим в том, что никакой автоматический процесс не может перенести данные с одной стороны этого промежутка на другую. Любая система, в которой данные передаются программно из точки А в точку Б, потенциально подвержена вторжению талантливого взломщика. Имея достаточно времени и приложив определенные усилия, тот, кто намерен заполучить ваши данные, сможет получить доступ к архивам, даже если используются сложные схемы типа второго "скрытого" жесткого диска, монтирующегося автоматически ночью для выполнения резервного копирования. Это все равно автоматизированная процедура, поэтому она уязвима для желающего взломать ее защиту.

Использование промежуточного пространства — это "последнее" средство, дающее гарантированную защиту за счет менее удобной процедуры. Если данные хранятся физически отдельно от машины, подключенной к сети, никакой хакер не сможет их заполучить. Именно так устроены сети медицинских и государственных учреждений: критические базы данных хранятся на машинах, не подключенных к сети. Они отделены пространством и поэтому защищены (по крайней мере настолько, насколько можно доверять администратору, имеющему физический доступ к системам).

Если секретные данные хранятся на дискете, компакт-диске CD-R или на другом съемном носителе отдельно, значит, до ваших "ключей от города" не смогут добраться, даже если вся сеть будет взломана. Конечно, если вы не забыли дискету в дисководе.

Если кажется, что система взломана

Независимо от вашей внимательности и количества принятых превентивных мер, невозможно быть на 100 процентов уверенным в защите системы. Абсолютная защита — настолько же недостижимый идеал, как и абсолютный нуль температуры или полный вакуум. И поскольку остается хоть минимальный шанс наличия бреши в защите системы, в случае любых сомнений приходится готовиться к худшему.

Хотя системы Tripwire и PortSentry могут существенно облегчить администратору работу по предотвращению и выявлению вторжения, он все равно должен следить за изменениями в поведении системы. Следите за результатами выполнения команды **top**, контролируйте загрузку системы в течение длительных периодов времени — не становится ли она больше. Если — да, исследуйте, с чем это может быть связано. Не игнорируйте странные изменения поведения, например необычно сформатированные приглашения регистрации или непривычные результаты выполнения стандартных команд. Периодически просматривайте каталоги **/tmp** и **/var/tmp**, обращая внимание на выполняемые файлы, установленные биты **setuid** или слишком большие файлы, и регулярно очищайте эти каталоги. Просматривайте журнальные файлы системы в каталоге **/var/log**; обращайте внимание на подозрительные сообщения, например, с длинными строками или мусорными символами — это почти наверняка результат попыток взлома путем переполнения буферов. Короче говоря, постоянно следите за всеми необычными явлениями. Такая постоянная подозрительность — иногда единственный способ заметить, что система ведет себя не так, как должна.

ПРИМЕЧАНИЕ

Распространенное место поиска свидетельств деятельности взломщиков — каталог **/dev**. Именно там часто помещают средства перехвата пакетов взломщики, использующие готовые средства из серии "rootkits" или сценарии. Однако, поскольку теперь ОС FreeBSD использует динамически генерируемую файловую систему DEVFS для специальных файлов устройств, этот каталог уже не вызывает такого беспокойства, но все равно не помешает за ним следить.

Если вы действительно подозреваете, что произошел взлом, и особенно если нашли свидетельства этого, надо предполагать, что ущерб — больше, чем кажется. Наиболее частой ошибкой администратора, обнаружившего свидетельства взлома защиты, является отключение ряда служб и предположение, что тем самым атака отбита. Часто так и бывает; однако если на все инциденты подобного рода реагировать именно так, недалеко и до катастрофы. Атакующему достаточно установить тот или иной "черный ход" в системе, который позволит ему вернуться и нанести более существенный ущерб, чем первоначально.

Если есть подозрение взлома системы, необходимо выполнить несколько действий.

1. Немедленно отключите систему от сети. Независимо от того, какие черные ходы установил атакующий, он ничего не сможет сделать, если система не подключена к сети. Это не даст взломщику, обнаружившему, что он выявлен, замести следы, стерев весь жесткий диск.
2. Проверьте, нет ли новых записей в таблицах **/var/cron/tabs** и **/etc/crontab**; проверьте также очередь **atq** на наличие заданий, оставленных взломщиком для выполнения в его отсутствие. Система может быть отключена от сети, но задания **cron** все равно будут работать, и взломщик может "добить" вашу систему таким способом, если не удалить все подозрительные задания.

3. Не пытайтесь связаться с атакующим или дать ему знать, что вы его выявили. Хотя вы и вывели свою систему из-под удара, атакующий скроется, если поймет, что вы пытаетесь его выследить. Да и правоохранительным органам будет сложнее после этого его найти. Пусть он думает, что вы просто отключили машину для восстановления.
4. Соберите вместе журнальные файлы из каталога **/var/log** и из любых других каталогов, в которых их могут создавать используемые программы, а затем поищите в них записи, которые могут показать, откуда проник атакующий и как он получил доступ. Если в системе поддерживаются службы, для которых недавно вышли предупреждения об уязвимых местах в защите, и вы не обновили эти службы, чтобы устранить их уязвимые места, — почти наверняка через эти службы атакующий и пробрался в систему.
5. Соберите всю полезную информацию, как указано на Web-сайте Национального центра защиты инфраструктуры (National Infrastructure Protection Center — NIPC): **http://www.nipc.gov** (или с сайта аналогичной национальной структуры по борьбе с компьютерными преступлениями, если вы живете не в США), и заполните заявление об инциденте. Получив от вас достаточно информации, ФБР сможет быстро выявить вторгшегося. Большая часть попыток взлома выполняется "вандалами-любителями" ("script kiddies") — случайными лицами, применяющими готовые, созданные другими средства, использующие ряд известных уязвимых мест. Взломщиков этого типа обычно легко находят и наказывают.
6. Создайте резервные копии важных данных — Web-документов, файлов конфигурации, начальных каталогов и всего содержимого каталога **/usr/local** — и переустановите операционную систему. Для большей надежности очистите предварительно жесткий диск, переустановите ОС FreeBSD "с нуля" и восстановите локальные данные. Используйте ежедневные отчеты системы Tripwire, чтобы понять, что необходимо пересоздать, и помните о "черных ходах", которые могут быть установлены вместе с остальными программами из каталога **/usr/local**.
7. Замените все службы самыми новыми версиями, просмотрев все соответствующие руководства по защите, прежде чем снова подключать систему к сети. Будьте особенно осторожны первые несколько дней после включения системы — могут быть повторные попытки взлома. Особенно внимательно в этот период просматривайте журнальные файлы; чем больше доказательств вы соберете, тем легче будет работать сотрудникам центра NIPC и ФБР.

Атаки на службы (DoS)

Хотя технически эта проблема и не связана с защитой, но такая разновидность враждебной сетевой деятельности в последнее время требует все больше внимания системных администраторов. Речь идет об атаках на службы (Denial of Service или DoS).

Атаки на службы не связаны с взломом защиты системы или нарушением конфиденциальности. Это простая передача большого потока запросов, вследствие чего по сети посылается так много пакетов, что в этом море теряются пакеты законных пользователей служб. Часто целью является остановка сервера, не справляющегося с огромным потоком данных и количеством запросов. От атак такого рода гораздо

сложнее защищаться, чем от прямых попыток взлома, которые можно отсечь с помощью систем IPFW, PortSentry и других описанных ранее средств. Атаки на службы нельзя предотвратить; можно только попытаться уменьшить их влияние, поскольку атаки задействуют только вполне допустимые пакеты, не отличающиеся от потока действительно нужных для работы данных. Проблема только в том, что их слишком много.

Иногда атака на службы оказывается исходящей из определенного источника, и ее можно заблокировать путем добавления правила брандмауэра, запрещающего прием пакетов с этого источника. Однако во многих последних атаках настоящий источник скрывался. Так, атаки путем посылки широковещательных сообщений **ping** (ICMP) выглядят исходящими из определенного адреса, который на самом деле является адресом жертвы, подвергающейся главному удару. Распределенные атаки на службы, или атаки DDoS, являются еще более изощренными, вовлекая сотни и даже тысячи взломанных настольных компьютеров, ненамеренно участвующих в атаке, так что выявить настоящего виновника практически невозможно.

Определенные опции конфигурации различных серверных программ и ядра помогут системе остаться в рабочем состоянии в ходе атаки на службы. Сейчас мы рассмотрим некоторые из возможных мер. Помните, однако, что они могут только повысить шансы системы на выживание в ходе атаки на службы, но не могут нейтрализовать саму атаку или гарантировать, что атакующий не повторит попытки с большей интенсивностью, пока не добьется своего.

Ограничение количества порождаемых серверных процессов

Многие атаки DoS направлены на службы типа Apache, Sendmail и другие, работающие путем "порождения" нового процесса для обработки каждого поступающего запроса. Если атакующий пошлет достаточно много запросов, в системе будет порождено столько процессов, что ресурсы процессора и памяти скоро исчерпаются и станет невозможной и стабильная работа системы. Ущерб от атак, провоцирующих порождение процессов, можно уменьшить, проверив, все ли подобные службы имеют встроенные ограничения на количество одновременно работающих процессов. Эти ограничения могут сказаться на способности сервера выполнять законные запросы в ходе обычной работы, но именно этот компромисс может спасти систему при атаке на службы.

Сервер Apache поддерживает директиву MaxClients со стандартным значением 150, не позволяющую обслуживать одновременно больше указанного количества запросов. Однако сложность в том, что продвинутый взломщик может вызывать интенсивно нагружающий процессор сценарий CGI до тех пор, пока сервер Apache не станет создавать процессы быстрее, чем они смогут завершаться, что приведет к сбою сервера гораздо быстрее, чем многочисленные запросы статических HTML-страниц. Часто единственным способом восстановления в этой ситуации, если система все же позволит зарегистрироваться с помощью Telnet или SSH, является остановка сервера Apache (**apachectl stop**) до завершения атаки. К счастью, большинство атак на службу HTTP можно проследить вплоть до конкретного IP-адреса клиента, заблокировав его с помощью правил системы IPFW или директивы **deny from** самого сервера Apache. Если это не поможет, можно уменьшить значение MaxClients, чтобы даже при его достижении клиенты не влияли на стабильность работы сервера.

Подобная возможность есть и в системе Sendmail: директива **MaxDaemonChildren**, отключенная по умолчанию. Ее можно включить, убрав перед ней символ комментария непосредственно в файле **/etc/mail/sendmail.cf** и перезапустив сервер (**make restart**). Кроме того, в системе Sendmail имеется встроенный "тормоз", предотвращающий запуск новых процессов при загрузке системы выше 12 процентов. Однако этот механизм срабатывает слишком медленно и не позволяет своевременно отреагировать на быстро развивающуюся атаку, так что может потребоваться явно ограничить количество порожденных процессов, поддерживаемых системой Sendmail в определенный момент времени.

Потенциальным общим решением проблемы атак, провоцирующих порождение процессов, которое позволит защититься даже от тех, что исходят из самой системы (от недовольного или продажного пользователя, например), является изменение файла **/etc/login.conf** и установка в нем ограничений на использование пользователем процессора, оперативной памяти и количество одновременно открытых файлов. Создайте класс для пользователя, от имени которого работает соответствующая служба — **nobody** в случае сервера Apache, или отдельного локального пользователя, ресурсы которого необходимо ограничить, и отнесите пользователя к данному классу с помощью команды **chfn**. Вот пример такого класса в файле **login.conf**:

```
baduser:\
        :cputime=30m:\
        :openfiles=24:\
        :maxproc=32:\
        :memoryuse=16m:\
        :tc=default:
```

Затем выполните команду **cap_mkdb /etc/login.conf** для включения этого нового класса в базу данных и установки ограничений для всех входящих в него пользователей.

Защита от атак с плацдарма

Атака "с плацдарма" использует для достижения своих целей ресурсы сети жертвы. В то время как для атак грубой силой или путем порождения процессов необходимо использование больших вычислительных мощностей, при атаке с плацдарма атакующему достаточно послать хитро составленный набор запросов, чтобы сама инфраструктура сети жертвы стала наихудшим ее врагом. Например, атака путем широковещательной рассылки запроса **ping** (или атака "smurf") связана с посылкой атакующим обычного запроса **ping** на широковещательный адрес вашей сети (что распространяет запросы на все хосты сети), адрес отправителя в котором подделан и является адресом другого хоста — несчастной жертвы, страдающей при такой атаке намного больше вас, поскольку каждый хост вашей сети в ответ переполняет жертву ответами **ping**. Эффект умножения при атаке такого рода может обернуться для жертвы катастрофой, а организатора атаки будет очень сложно (или даже невозможно) выявить. Другая разновидность атаки с плацдарма запускает UDP-пакет между портами службы **echo** двух серверов, вызывая их вхождение в бесконечную "эхо-войну", остановить которую можно только отключением порта службы **echo** (что и сделано в ОС FreeBSD по умолчанию).

Атаки с плацдарма лучше всего предотвращать на уровне основного маршрутизатора сети. Атаки типа smurf можно предотвратить, сконфигурировав маршрутизатор так, чтобы он не отвечал на широковещательные запросы **ping**. Если в качестве мар-

шрутизатора используется машина с ОС FreeBSD, установка **icmp_bmcastecho="NO"** в файле **/etc/defaults/rc.conf** предотвращает ответы на такого рода запросы, которые практически никогда не используются с хорошей целью.

По умолчанию в стандартном ядре **GENERIC** скомпилирована опция **ICMP_BANDLIM**, ограничивающая частоту ответов на ICMP-сообщения об ошибках (еще один типичный вариант атаки с плацдарма), ограничивая тем самым эффективность подобных атак.

Физическая защита

Даже если принять все возможные меры сетевой защиты, остается куда более существенная проблема защиты физической, от непосредственного доступа. Самая защищенная система в мире легко будет взломана, если неавторизованное лицо может получить физический доступ к серверу, поскольку никакая программная защита не спасет от взломщика с отверткой.

Безопасное размещение (secure co-location facilities) жизненно важно для коммерческого или любого высокоуровневого сервера Internet. Необходимы закрытые серверные стойки в закрытых для посторонних комнатах, причем открывать стойки и иметь доступ к машинам должны иметь право только сотрудники организации, обеспечивающей это безопасное размещение. Ваша система может иметь промышленное исполнение с закрытой передней панелью и средствами защиты BIOS, уведомляющими вас в случае снятия панели.

Любой человек, получивший физический доступ к машине, может перезагрузить ее в однопользовательский режим, в стандартной конфигурации которого пароль пользователя **root** не запрашивается. Это можно изменить, потребовав ввода пароля путем указания в файле **/etc/ttys**, что консоль "небезопасна", т.е. что вы не можете гарантировать, что с нее обращается только авторизованный персонал:

```
console  none                 unknown  off  insecure
```

Однако это не помешает злоумышленнику загрузиться с дискеты или CD-ROM и взломать систему. Другие подключенные к машине устройства, например модемы или беспроводные сети, также могут использоваться для получения несанкционированного доступа, поэтому их не должно быть на машине, физический доступ к которой вы пытаетесь ограничить. Вывод: при отсутствии гарантированной физической защиты полная защита недостижима.

Другие источники информации о защите

В этой главе рассмотрен ряд общих проблем защиты в применении к ОС FreeBSD. Однако тема сетевой защиты — обширна, и проблемы добавляются с каждым днем, поскольку все больше злонамеренных пользователей пытается найти способы остановить базовые службы сети Internet.

Есть ряд отличных источников информации по защите (как специфической для ОС FreeBSD, так и общей), с которыми имеет смысл ознакомиться.

Страница справочного руководства man security

Составленная Мэтью Диллоном (Matthew Dillon), страница справочного руководства **man security** содержит описание общих проблем защиты и правильной практики администрирования, а также различные советы по предотвращению взломов и

атак на службы. Эта страница является базой для нескольких сетевых ресурсов, включая часть руководства FreeBSD Handbook.

Списки рассылки

Подпишитесь на список рассылки **freebsd-security@freebsd.org**. Для этого пошлите сообщение по адресу **majordomo@freebsd.org** с текстом **subscribe freebsd-security** в теле сообщения. Именно в этом списке рассылки обсуждаются наиболее актуальные проблемы защиты. Администратор должен быть в курсе последних достижений, чтобы прикрыть слабое место в защите сразу же после его обнаружения.

Еще один полезный список рассылки по защите, посвященный проблемам защиты UNIX вообще, — Bugtraq. В этот список приходят рекомендации по всем основным проблемам, возникающим в защите Internet, иногда даже раньше, чем станет понятно их влияние на ОС FreeBSD. Список рассылки Bugtraq поддерживается на сайте **http://www.securityfocus.com**, где можно подписаться или выполнить поиск в архивах.

Руководства по защите ОС FreeBSD

Руководства по защите (security advisories) рассылаются ответственным за защиту ОС FreeBSD (FreeBSD Security Officer) через списки рассылки **freebsd-announce@freebsd.org** и **freebsd-security@freebsd.org** и предупреждают о новых выявленных уязвимых местах. Каждое такое руководство также архивируется в каталоге **http://www.freebsd.org/security/**.

Руководство содержит полное описание сути и влияния обнаруженного уязвимого места: находится ли оно в части базовой системы FreeBSD или в одной из портированных программ, специфична ли эта проблема для ОС FreeBSD, как обойти или решить проблему. Поскольку обсуждение сути проблемы до обнародования ее решения было бы приглашением для взломщиков к началу атак, руководства выходят только после того, как найдено решение проблемы. Это еще одна веская причина подписаться на список рассылки **freebsd-security@freebsd.org**, поскольку там уязвимое место обсуждается еще до выпуска руководства по его устранению.

Для устранения уязвимых мест в портированных приложениях или пакетах обычно достаточно обновить дерево каталогов и перестроить уязвимое приложение (см. главу 15). Исправление же в базовой системе FreeBSD, однако, обычно вносится в соответствующее дерево исходного кода (**-STABLE** или **-CURRENT**). Чтобы использовать его, необходимо перестроить соответствующую часть системы после обновления исходных кодов. Если исправление затрагивает достаточно фундаментальную часть системного кода, для обеспечения защиты системы может потребоваться полная перестройка, **make world**. Инструкции о том, как это сделать, представлены в главе 18.

Web-ресурсы

Страница информации о защите FreeBSD, **http://www.freebsd.org/security/**, содержит ресурсы и ссылки, полезные для администратора ОС FreeBSD или разработчика. Тут же представлен архив руководств по защите, а также различные советы и рекомендации по устранению факторов риска.

Документ FreeBSD Security How-To (**http://people.freebsd.org/~jkb/howto.html**) предлагает описание различных методов защиты системы FreeBSD. В нем рассмотрены многие темы, затронутые в данной главе, и множество тем, которые здесь даже не упомянуты.

Сайт CERT (**http://www.cert.org**) — наиболее известный сайт по защите в Internet — содержит информацию об уязвимых местах в защите всех операционных систем и считается авторитетным источником предупреждений об уязвимых местах и информации о восстановлении. На сайте CERT также принимаются отчеты об инцидентах. Вы можете сообщить на сайт о взломе, и вас свяжут с соответствующими органами для поимки взломщика.

SecurityFocus — сайт, на котором поддерживается список рассылки Bugtraq. Это сайт новостей о защите, посвященных широкому спектру тем: от систем выявления вторжения до защиты от вирусов. Там также есть многочисленные статьи по эффективным приемам защиты и ответственному отношению к администрированию системы. Там не так уж много специфического материала по ОС FreeBSD, но большая часть информации применима к любой платформе. Адрес сайта — **http://www.securityfocus.com**.

Книги

В файле **/etc/rc.firewall** рекомендуются две книги: "Брандмауэры и защита Internet" (*Firewalls & Internet Security*) Уильяма Чесвика (William R. Cheswick) и Стивена Беллоуина (Steven M. Bellowin) по общим проблемам защиты сетей, и "Построение брандмауэров Internet" (*Building Internet Firewalls, 2nd Edition*) Брента Чэпмена (Brent Chapman) и Элизабет Цвики (Elizabeth Zwicky) как более полное описание теории и практики использования брандмауэров.

Дополнительные книги и статьи по защите указаны и оценены с точки зрения полезности на сайте SecurityFocus в разделе "Library" ("Библиотека").

30 ГЛАВА

Сервер доменных имен

- Введение в систему BIND
- Подключение демона сервера имен
- Файл конфигурации системы BIND (named.conf)
- Создание файла зоны
- Конфигурирование кэширующего сервера имен

В предыдущих главах, в частности в главах 22 и 23, рассмотрено использование системы доменных имен (DNS) для поиска хостов в Internet по имени; при этом не требуется запоминание IP-адресов каждым пользователем. Был описан процесс настройки машины с ОС FreeBSD для получения информации о доменных именах с указанного сервера. Теперь мы подошли к проблеме конфигурирования машины с ОС FreeBSD для работы в качестве сервера доменных имен, предоставляющего информацию об именах как для своих процессов, так и для клиентских машин, которым она необходима.

Конфигурирование службы доменных имен — одна из наиболее сложных задач сетевого администрирования. Если настройка любого количества Web-серверов, даже на всех подключенных к сети компьютерах, — сравнительно простая задача, то установка сервера имен обычно делается только один раз, каким-нибудь "гуру", чьи деяния вскоре становятся легендой среди сетевых администраторов. В результате получается такая конфигурация службы DNS, которую сложно не то что сопровождать, но даже понять. Администраторов, знающих все детали настройки службы DNS, намного меньше, чем тех, кто ничего в этом не понимает. В этой главе мы не пытаемся дать исчерпывающее описание службы DNS. Об этом написаны целые книги, причем достаточно толстые. Цель этой главы — научить вас настраивать FreeBSD для работы в качестве сервера имен в нескольких типичных конфигурациях.

Программных реализаций службы DNS много. Но для операционных систем UNIX, на которых работает подавляющее большинство серверов имен в Internet, обычно выбирают систему BIND, распространяемую консорциумом Internet Software Consortium (ISC).

Введение в систему BIND

Система BIND (сокращение от *Berkeley Internet Name Domain*) состоит из основной программы-демона (**named**), набора библиотек разрешения имен, позволяющих выполнять поиск имен, и ряда административных средств. Система BIND входит в состав ОС FreeBSD, хотя соответствующая служба и не включена по умолчанию.

Структура системы доменных имен

Система доменных имен (DNS) — это иерархический протокол, работающий по сети Internet аналогично протоколам маршрутизации. Несколько "корневых серверов", географически распределенных по всей сети Internet для обеспечения надежности и избыточности, поддерживается корпорацией Network Solutions, Inc. и другими организациями. Каждое доменное имя (например, **somewhere.com**) строится в обратном порядке, начиная с корневой зоны, причем суффикс домена — **com**, **org**, **net** и т.д. — определяет верхний уровень иерархии, непосредственно после точки (.), обозначающей корневую зону. После каждого из суффиксов (которые обычно называют доменами верхнего уровня — top-level domains или TLD) следуют имена доменов, определяемые обычно не корневыми серверами, а отдельными хостами DNS в сети Internet. Например, служба DNS домена **somewhere.com** будет администрироваться сервером в его же сети (например, сервером **ns1.somewhere.com**). Этот сервер является "авторитетным" хостом DNS для данного домена. Центральный реестр, также поддерживаемый корпорацией Network Solutions, содержит записи обо всех таких доменах, так что корневые серверы могут предоставлять авторитетную информацию о соответствии их имен. Эти "записи хостов" задаются при настройке нового сервера

имен: в корпорацию Network Solutions или одному из непосредственно связанных с ней регистраторов посылается соответствующая форма.

Когда клиент обращается с запросом к службе DNS, он запрашивает сервер имен, сконфигурированный в ходе настройки стека протоколов TCP/IP, — обычно это сервер в той же сети, как показано на рис. 30.1. Если сервер не может ответить на запрос, он передает его серверу DNS более высокого уровня, если он доступен. Если же такого сервера нет, запрос направляется непосредственно корневым серверам.

Корневые серверы сами не поддерживают никаких авторитетных данных службы DNS. Они содержат только записи хостов (мы будем называть их записями **NS**), которые указывают на авторитетные серверы имен в каждом домене. Корневые серверы возвращают запрашивающему ответ DNS, содержащий ссылку на авторитетный сервер имен домена, к которому обращался клиент. Этот сервер, в свою очередь, возвращает запрашиваемые данные DNS локальному серверу DNS, который передает ее клиенту.

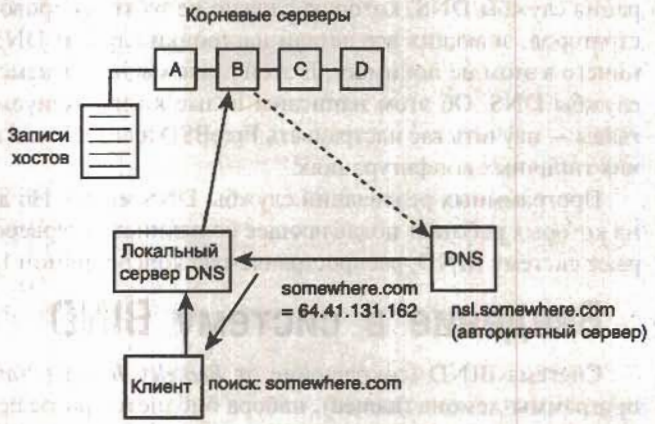


Рисунок 30.1. *Диаграмма поиска в системе DNS, показывающая путь от клиента на локальный сервер DNS, корневые серверы и авторитетный сервер имен*

Локальный сервер DNS может сохранить результат поиска в своем кэше на период времени, указанный авторитетным сервером имен. Это ускоряет обработку запросов, позволяя локальным клиентам получать непосредственные ответы на свои запросы DNS от локального сервера, без передачи информации по Internet. Это, однако, означает, что изменения в записях DNS на авторитетном сервере DNS домена не будут доступны клиенту до завершения периода устаревания. До этого времени доступная клиенту информация DNS "заморожена" и потенциально неверна. Мы еще вернемся к проблеме кэширования информации и устаревания данных далее в этой главе.

Зоны

Система BIND позволяет определять *зоны* — логические группы IP-адресов и имен хостов, существующие на определенном уровне иерархии имен DNS. Например, **com.** — это зона в корневой зоне (.), а **somewhere.com.** — это зона, существующая в зоне **com.** Обратите внимание, что имена зон всегда завершаются точкой, обозначающей корневую зону, — это является существенным при создании собственных файлов зон! *Поддомены* — это зоны, управляемые отдельными серверами доменных имен; например, **cluster.somewhere.com.** — это зона, в которую может входить несколько машин. Фактически она может иметь столько дальнейших подразделов, сколько захочет сконфигурировать администратор. Эти зоны обеспечивают *прямой поиск имен* в системе DNS, или сопоставление именам IP-адресов.

Обратный поиск имен в системе DNS, или сопоставление IP-адресам имен хостов, также осуществляется по зонам. Поскольку система DNS и IP-адреса ведут свое начало от проекта ARPAnet (подробнее см. в главе 22) и не претерпели с тех пор принципиальных структурных изменений, имя зоны обратного поиска DNS имеет вид **CCC.BBB.AAA.in-addr.arpa**. Это имя состоит из IP-адреса, записанного в обратном порядке, с добавлением суффикса **.in-addr.arpa**. Например, сеть **64.41.131.*** будет определяться зоной **131.41.64.in-addr.arpa**.

Каждая зона, которой управляет ваш сервер имен, должна быть определена в *файле зоны* — форматированном наборе определений, сопоставляющих имена хостов зоны IP-адресам (или наоборот). Файл зоны также содержит параметры, определяющие характеристики зоны, в частности период устаревания кэша. Файлы зоны — наиболее важная часть конфигурации службы DNS; они будут подробно рассмотрены далее в этой главе.

Файлы и программы системы BIND

Поскольку система BIND встроена в ОС FreeBSD, не придется беспокоиться по поводу ее установки или проверки наличия необходимых файлов в соответствующих местах. Не помешает, однако, знать, какие программы задействованы в работе сервера имен и где придется изменять конфигурацию.

- **/usr/sbin/named**. Собственно демон сервера имен. Он прослушивает порт 53, ожидая поступления запросов поиска системы DNS.
- **/usr/sbin/ndc**. Программа управления демоном сервера имен. Она используется для запуска, останова, перезагрузки и контроля сервера имен.
- **/etc/namedb**. Все файлы конфигурации и файлы текущего состояния системы BIND, включая файлы определения зон, находятся в этом каталоге (или в созданных администратором подкаталогах).
- **/etc/namedb/named.conf**. Основной файл конфигурации системы BIND. Из него система BIND "узнает", какими доменами управлять и как работать с каждым из них.

Включение демона сервера имен

Включение системы BIND в ОС FreeBSD — самая простая часть ее конфигурирования. Необходимо только, как и для любой другой рассмотренной ранее встроенной службы FreeBSD, добавить следующую строку в файл **/etc/rc.conf**:

```
named_enable="YES"
```

Если эта опция установлена, демон **named** автоматически запускается при загрузке операционной системы. Для запуска демона без перезагрузки используется программа **ndc**:

```
# ndc start
new pid is 12717
```

Теперь сервер должен работать. Но это только самое начало процесса конфигурирования системы BIND.

ПРИМЕЧАНИЕ

Для обеспечения максимальной производительности добавьте в файл **/etc/resolv.conf** строку **nameserver**, ссылающуюся на адрес закольцовывания (**127.0.0.1**) как на основной сервер

имен. Это обеспечит наиболее быстрый способ поиска имен в системе DNS на локальной машине. Убедитесь, что перед дополнительными серверами имен имеется следующая строка:

```
search somewhere.com
nameserver 127.0.0.1
nameserver 64.41.131.167
```

Запуск системы BIND в "песочнице"

Как утверждалось в главе 27, иногда имеет смысл (из соображений защиты) запускать определенные службы в среде, которую называют "песочницей", — отдельной структуре каталогов, не содержащей ничего лишнего, так что извне кажется, будто именно она и представляет всю файловую систему. В случае службы FTP эта структура называлась "тюрьмой измененного корневого каталога" ("**chroot** jail"). Фактический корневой каталог файловой системы изменялся так, что сервер и создаваемые им процессы не могли выйти за пределы отведенной им структуры каталогов выше определенной точки. Система BIND поддерживает такую же возможность, но в большинстве документации эту среду называют "песочницей" ("sandbox"), а не "тюрьмой". Принцип организации среды, однако, — тот же.

Типичное конфигурирование "песочницы" состоит в создании подкаталога **sandbox** в каталоге **/etc/namedb** и запуске демона **named** так, что он ограничивает себя только этим каталогом. Поэтому, если демон **named** будет взломан (что вполне возможно, поскольку уязвимые места в версиях системы BIND находят и по сей день), ущерб будет нанесен только соответствующему каталогу. Создайте каталог **/etc/named/sandbox** и измените его владельца и права доступа так, чтобы он принадлежал непривилегированному пользователю и группе **bind**:

```
# chown -R bind:bind /etc/namedb/sandbox
# chmod -R 750 /etc/namedb/sandbox
```

Затем создайте подкаталоги **/etc** и **/var/run** в каталоге **sandbox**. Скопируйте файл **/etc/localtime** в каталог **/etc/namedb/sandbox/etc**. Сервер будет записывать во время выполнения файлы в каталог **/var/run**, и ему необходим файл **localtime** для обработки последовательных значений из файлов зон и для правильной регистрации дат.

```
# mkdir /etc/namedb/sandbox/etc
# cp /etc/localtime /etc/namedb/sandbox/etc
# mkdir -p /etc/namedb/sandbox/var/run
```

Наконец, добавьте следующую строку в файл **/etc/rc.conf**:

```
named_flags="-u bind -g bind -t /etc/namedb/sandbox"
```

Учтите: если для управления демоном **named** используется программа **ndc** (как будет показано в этой главе) и демон работает в "песочнице", придется использовать опцию **-c**. Команды будут иметь следующий вид:

```
# ndc -c /etc/namedb/sandbox/var/run/ndc start
```

Учтите также, что для записи демоном **named** журнала в файл, последний должен быть в каталоге **sandbox**.

Файл конфигурации системы BIND (named.conf)

Чтобы от сервера имен был толк, убедитесь, находится ли он в нужном месте по отношению к клиентам и остальной части сети, с учетом ее топологии, и правильно ли

он сконфигурирован для взаимодействия с другими серверами имен. Неправильная конфигурация сервера имен может привести к постоянному обмену информацией между ним и корневыми серверами в безуспешных попытках согласовать данные. Файл **/etc/namedb/named.conf** необходимо правильно построить, а для этого — хорошо понимать.

К счастью, система BIND версии 8 (включается в ОС FreeBSD в момент написания этой главы; вскоре, вероятно, ее заменит BIND версии 9) существенно упрощает многие таинственные моменты конфигурирования сервера имен, обычно присутствовавшие в первых версиях системы (до версии 4 включительно). Система BIND 8, сразу заменившая BIND 4, добавила множество возможностей конфигурирования, одновременно избавив от большого количества избыточных или плохо продуманных элементов. В листинге 30.1 представлен простой файл **named.conf**, позволяющий понять используемый синтаксис и структуру файла конфигурации.

Листинг 30.1. Пример файла /etc/namedb/named.conf

```
/*
 *   Пример файла конфигурации системы BIND 8
 */

logging {
    category lame-servers { null; };
    category cname { null; };
};

options {
    directory "/etc/namedb";
};

zone "somewhere.com" {
    type master;
    file "somewhere.com";
};

zone "131.41.64.in-addr.arpa" {
    type master;
    file "131.41.64.in-addr.arpa";
};

zone "elsewhere.com" {
    type slave;
    file "slave/elsewhere.com";
    masters { 113.125.2.145; };
};

zone "." {
    type hint;
    file "named.boot";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "localhost.rev";
};

zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.INT" {
    type master;
    file "localhost.rev";
};
```

Как видно из листинга, файл конфигурации состоит из нескольких блоков (или "операторов") с синтаксисом в стиле языка C, включающих подоператоры в фигур-

ных скобках ({}). Все поддерживаемые операторы перечислены и описаны на странице справочного руководства **named.conf**. Чаще всего используются операторы **options**, **controls**, **logging** и **zone**.

Комментарии в файле **named.conf** также задаются в стиле языка C; однострочные комментарии начинаются двумя косыми (//), а блочные комментарии имеют синтаксис: /* комментарий */. Поддерживаются также комментарии в стиле командного интерпретатора (начинающиеся символом #).

Файл **named.conf**, входящий в стандартную поставку ОС FreeBSD, включает как операторы, представленные в листинге 30.1, так и ряд других. В нем также много справочной информации в виде комментариев, описывающих использование каждого оператора.

При любом изменении в файле **named.conf** или каком-либо из файлов зон (которые будут рассмотрены далее в этой главе) необходимо перезапустить демон **named** с помощью программы **ndc**:

```
# ndc reload
Reload initiated.
```

Использование перенаправляющего сервера

Возвращаясь к представленной на рис. 30.1 топологии, заметим, что *перенаправляющим сервером* (forwarder) называют сервер имен в сети "верхнего" уровня — сети, более близкой к корневым серверам, позволяющей запрашивать из нее информацию системы DNS. (Учтите, что доступ к серверу имен может быть ограничен, как будет показано в дальнейшем.) В качестве перенаправляющего сервера для системы FreeBSD, работающей как сервер имен домашней сети, лучше всего использовать сервер имен поставщика услуг Internet. Если ваша машина не может сама ответить на запрос DNS-информации, она запрашивает ее у корневых серверов (указанных в файле **/etc/namedb/named.root**). Это требует времени и передачи дополнительных пакетов по сети.

На самом деле не обязательно всем получать авторитетные ответы. В большинстве случаев вполне допустимо работать с неавторитетной DNS-информацией, например, поступающей от кэширующего сервера имен:

```
# nslookup www.freebsd.org ns.somewhere.com
Server: ns.somewhere.com
Address: 64.41.131.172

Non-authoritative answer:
Name: freefall.freebsd.org
Address: 216.136.204.21
Aliases: www.freebsd.org
```

Конфигурирование системы FreeBSD для обращения к одному или нескольким перенаправляющим серверам позволяет ей воспользоваться кэшем сервера имен верхнего уровня, а не извлекать авторитетные данные в ответ на каждый запрос. Недостатком (как уже было сказано) является "замороженность" данных в кэше на период, продолжительность которого задается в файле зон авторитетного сервера (этот период часто составляет несколько дней). На рис. 30.2 представлен путь DNS-запроса при использовании перенаправляющего сервера. Основную работу выполняет перенаправляющий сервер, строящий свой кэш по результатам собственных запросов и при выполнении запросов своих клиентов. Использующему его серверу DNS нижнего уровня необходимо только запросить перенаправляющий сервер, сокращая время выполнения запроса и объем передаваемой информации.

СОВЕТ

Увеличение объема передаваемой DNS-информации, часто вызываемое неправильной конфигурацией серверов имен, не компенсируется наличием кэширующих серверов имен. Эти перенаправляющие серверы предотвращают избыточную передачу ненужных запросов по низкоскоростным каналам связи, например, в случаях, когда хост DNS, работающий в качестве авторитетного сервера имен для домена, подключен по низкоскоростному каналу DSL или вообще по коммутируемой линии. Без кэширующих серверов имен каждый хост в Internet был бы вынужден посылать запросы DNS непосредственно ему. При наличии перенаправляющих серверов нагрузка на канал связи целевого хоста существенно уменьшается. Использование перенаправляющих серверов позволяет быть добропорядочным гражданином сетевого сообщества, а также ускорить и повысить эффективность собственных запросов.

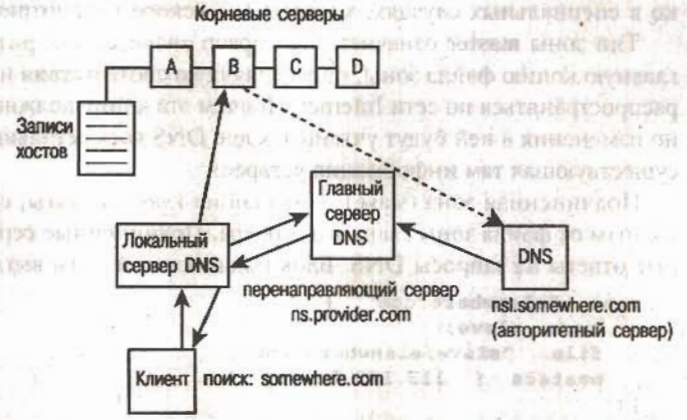


Рисунок 30.2. *Диаграмма поиска в системе DNS с использованием перенаправляющего сервера, который предоставляет кэшированную неавторизованную информацию DNS локальному серверу DNS и клиенту*

Чтобы обеспечить использование перенаправляющего сервера, замените адрес **127.0.0.1** в блоке **forwarders** оператора **options** IP-адресом сервера имен верхнего уровня (**127.0.0.1**, адрес локального хоста, тут не поможет) и уберите символы комментариев **/*** и ***/** в начале и конце блока.

```
/*
        forwarders {
                127.0.0.1;
        };
*/
```

Можно задавать сколько угодно перенаправляющих серверов — демон **named** обратится к каждому из них, прежде чем сдаться и запросить корневые серверы. Можно также убрать комментарий с оператора **forward only** в блоке **options**, чтобы демон **named** при обработке всех запросов обращался только к перенаправляющим серверам. Демон обращается за DNS-информацией к перенаправляющим серверам, пока не окажется, что они недоступны или не работают — тогда выполняются полные запросы к корневым серверам. Если установлена опция **forward only**, запросов к корневым серверам не будет.

Конфигурации главного и подчиненного сервера

Каждый блок зоны определяет домен или поддомен, который будет администрироваться вашей машиной с ОС FreeBSD. Чаще всего блок зоны определяет домен вида **somewhere.com**:

```
zone "somewhere.com" {
   type master;
   file  "somewhere.com";
};
```

Абсолютно необходимо в этом блоке только имя домена или поддомена (без завершающей точки, которая обычно указывается в имени зоны), оператор **type**, определяющий тип конфигурации для домена (главная или подчиненная), и имя файла, содержащего информацию зоны. Давайте разберемся, как действует сервер имен, будучи главным или подчиненным в определенной зоне.

Подоператор **type** может задавать значения **master**, **slave**, **stub**, **forward** или **hint**. Чаще всего используются значения **master** (главный) и **slave** (подчиненный); остальные — только в специальных случаях, которые мы вскоре рассмотрим.

Тип зоны **master** означает, что сервер является авторитетным для зоны — содержит главную копию файла зоны, определяющую соответствия имен и адресов, которые будут распространяться по сети Internet. Причем эта копия должна изменяться только вручную, но изменения в ней будут учтены в кэше DNS всех остальных серверов имен, как только существующая там информация устареет.

Подчиненная зона (**slave**) — это копия главной зоны; ее файл зоны является производным от файла зоны главного сервера. Подчиненные серверы могут давать авторитетные ответы на запросы DNS. Блок подчиненной зоны выглядит следующим образом:

```
zone  "elsewhere.com"  {
   type  slave;
   file  "slave/elsewhere.com";
   masters  {  113.125.2.145;  };
};
```

Подоператор **masters** содержит список главных серверов (перечисленных через точку с запятой), с которых берется информация зон. Эти серверы могут быть как авторитетными серверами главной зоны, так и неавторитетными подчиненными. Оператор **file** в подчиненной зоне задает имя автоматически генерируемого файла зоны, который демон **named** создаст после получения информации от главного сервера. В нашем примере файл **elsewhere.com** создается в подкаталоге **/etc/namedb/slave**. Такую структуру каталогов можно использовать для отделения авторитетных (главных) файлов зон от автоматически генерируемых (подчиненных).

СОВЕТ

При настройке подчиненной зоны на сервере выполните команду **ndc reload**, чтобы система BIND выполнила перенос зоны и создала новый файл зоны в заданном оператором **file** месте. В некоторых случаях файл зоны подчиненного сервера не обновляется сразу после изменения главного файла зоны, даже если выполнена команда **ndc reload**. В этом случае просто удалите файл подчиненной зоны и повторно выполните команду **ndc reload** для пересоздания файла и переноса данных зоны.

Другие типы зон

Как описано в справочном руководстве по файлу **named.conf**, поддерживается еще три типа блоков зоны.

- **stub**. Зоны типа **stub** работают так же, как и подчиненные, но переносят с главного сервера только записи NS — записи, задающие местонахождение авторитетной DNS-информации для указанного домена.

- **forward**. Зоны типа **forward** можно использовать для перенаправления запросов зоны другому серверу или группе серверов. В операторе **zone** этого типа можно задавать блок **forwarders**, действующий аналогично такому же глобальному оператору в блоке **options**. Это позволяет переопределять глобальный список перенаправляющих серверов.
- **hint**. Зона типа **hint** реально используется в единственном случае — задает начальный список потенциальных корневых серверов, который хранится в файле **/etc/namedb/named.root**. На самом деле, это не авторитетный список корневых серверов, а просто список "подсказок" системе BIND, задающий серверы, на которых имеется действительный в настоящее время список корневых серверов, часто изменяющийся в зависимости от особенностей функционирования сети.

Ограничение доступа к службе DNS

Система BIND позволяет ограничить доступ с помощью списков контроля доступа (Access Control List — ACL). Список ACL указывается в операторе **acl**, задающем имя списка и содержащем критерии включения хостов в список. Эти критерии могут иметь различные формы, а также быть рекурсивными; другими словами, список ACL может содержать другие списки ACL. Следующие списки контроля доступа являются встроенными.

- **any**. Разрешает доступ всем хостам.
- **none**. Запрещает доступ всем хостам.
- **localhost**. Разрешает доступ с IP-адресов всех интерфейсов системы.
- **localnets**. Разрешает доступ с любого хоста сетей, к которым подключена система.

Элементами списка контроля доступа также могут быть IP-адреса (например, **111.112.113.114**), адреса сетей в формате CIDR (например, **111.112.113/24**), адреса с отрицанием (**!111.112.113.114**, что означает "любой хост кроме **111.112.113.114**") или оператор **key** (используется для защищенных транзакций и в этой главе не рассматривается). Составляя список элементов ACL, более специфичные элементы лучше указывать перед элементами, имеющими более широкий охват, поскольку список проверяется в порядке задания элементов. Если хост соответствует элементу списка, доступ с него будет разрешен, независимо от того, есть ли далее для него запрещающий элемент. Все исключения надо задавать в начале списка.

Поскольку файл **named.conf** читается последовательно, список контроля доступа должен задаваться в файле до того, как на него будут указываться ссылки в других операторах. Лучше всего вынести все операторы **acl** в начало файла **named.conf**, до блока **options**. Вот пример простого списка контроля доступа:

```
acl  "my_list"  {
   localhost;
   localnets;
   another_list;
   !132.112.14.124;
   132.112.14/24;
};
```

После задания списка контроля доступа его можно использовать далее вместе с другими элементами списка адресов в операторе **zone** (чтобы управлять доступом к

запросам для соответствующей зоны) или в глобальном операторе **options**. Поддерживаются следующие операторы контроля доступа:

- **allow-query** { элементы_списка_адресов; ... };. Задает хосты, которым разрешено выполнять обычные запросы DNS. Оператор **allow-query** можно также задавать в операторе **zone** — в этом случае он переопределяет оператор **allow-query** блока **options**. Если этот оператор не указан, запросы разрешены со всех хостов.
- **allow-transfer** { элементы_списка_адресов; ... };. Задает подчиненные серверы, которым разрешено переносить информацию зоны с данного сервера. Оператор **allow-transfer** можно также задавать в операторе **zone** — в этом случае он переопределяет оператор **allow-transfer** блока **options**. Если этот оператор не указан, перенос информации зоны разрешен со всех хостов.
- **allow-recursion** { элементы_списка_адресов; ... };. Задает хосты, которым разрешено выполнять рекурсивные запросы через данный сервер. Если этот оператор не указан, рекурсивные запросы разрешены со всех хостов.
- **blackhole** { элементы_списка_адресов; ... };. Задает список адресов машин, от которых сервер не будет принимать запросы и которые он не будет использовать при разрешении запросов. На запросы с этих адресов ответов не будет.

Например, можно глобально ограничить запросы, разрешив запрашивать только членам ранее заданного списка контроля доступа **my_list**. Более того, можно ограничить запросы к домену **somewhere.com**, разрешив их делать только членам списка **my_list** и дополнительной сети, а также ограничить перенос информации зоны, разрешив его двум указанным подчиненным серверам. Для этого используется следующая частичная конфигурация:

```
options {
   directory   "/etc/namedb";
      allow-query  {  my_list;  };
};
zone  "somewhere.com"  {
   type  master;
   file   "somewhere.com";
      allow-query  {  my_list;  64.2.43/24;  };
      allow-transfer  {  64.2.43.167;  123.15.221.3;  };
};
```

Создание файла зоны

В файле зоны задаются соответствия имен хостов и IP-адресов в пределах домена или зоны. Именно в этом файле чаще всего делают ошибки при конфигурировании системы BIND, поэтому мы опишем его достаточно подробно.

Формат файла зоны (который часто называют *главным файлом зоны* — Master Zone File, *или просто, главным файлом* — Master File) весьма сложен и строг, хотя и допускает определенные послабления. "Живой" файл зоны в только что установленной системе BIND вполне понятен. Он помогает разобраться, какие виды примитивов (директив) разрешены и как они задаются, а также содержит допустимые и часто используемые сокращенные варианты директив.

Сначала давайте рассмотрим пример файла зоны, представленный в листинге 30.2, и разберем его компоненты. Обратите внимание, что в файлах зон точки с запятой (;) являются символами начала комментария.

Листинг 30.2. Пример файла зоны для домена somewhere.com.

```
$TTL  3600
   somewhere.com.   IN   SOA   stripes.somewhere.com.   root.somewhere.com.   (
                              20010610          ;  Serial
                              10800             ;  Refresh
                              3600              ;  Retry
                              604800            ;  Expire
                              86400   )         ;  Minimum  TTL

   ; Серверы DNS
   @                          IN  NS     stripes.somewhere.com.
   @                          IN  NS     spots.somewhere.com.

   ; Имена машин
               localhost IN  A     127.0.0.1
               ns1       IN  A     64.41.131.162
               ns2       IN  A     64.41.131.163
               mail      IN  A     64.41.131.167
               @         IN  A     64.41.131.162

               ; Псевдонимы
               www             IN  CNAME   @
               ftp             IN  CNAME   www.somewhere.com.

               ; Запись MX
               @               IN  MX  10  mail.somewhere.com.
```

Хотя это файл может показаться непонятным и не имеющим определенного формата, на самом деле, он имеет четкую структуру. Листинг 30.2 состоит из шести основных элементов:

- директива **$TTL**;
- запись **SOA** (Start-of-Authority);
- блок записей **NS** (серверов имен);
- блок записей **A** (адресов);
- блок записей **CNAME** (каноническое имя — Canonical Name), определяющих псевдонимы;
- запись **MX** (Mail Exchanger).

За исключением этих директив, каждый элемент файла — это запись ресурса (Resource Record — RR), определяющая свойства имени в зоне по отношению к определенной "точке отсчета" ("origin"). Рассмотрим все представленные элементы подробно.

Директивы

Формат файла зоны допускает использование нескольких различных базовых директив. Они задают глобальные установки для всего файла. Каждая такая директива задается прописными буквами и начинается с символа доллара ($). Директивы могут задаваться в любом месте файла зоны, причем каждая последующая директива переопределяет одноименные предыдущие.

Из всех имеющихся директив на практике используется только **$ORIGIN**.

- **$ORIGIN**. Синтаксис: **$ORIGIN <имя домена> [<комментарий>]**

  Значение **$ORIGIN** будет добавляться к любому неуточненному имени в записи. Примером неуточненного имени является **www**; если имя **www** указано в записи в файле зоны, а директива **$ORIGIN** имеет значение **somewhere.com.**, запись будет определена как **www.somewhere.com.** (с завершающей точкой).

  Если директива **$ORIGIN** в файле зоны не задана, предполагается значение, совпадающее с именем зоны, заданным в операторе **zone**, который ссылается на файл зоны. Если имя домена в директиве не имеет завершающей точки, оно не является "абсолютным", т.е. будет добавляться к любой предшествующей строке **$ORIGIN**. Например, следующие конструкции

```
$ORIGIN  com.
$ORIGIN  somewhere
www                    IN CNAME  stripes
```

  эквивалентны одной:

```
www.somewhere.com.  IN CNAME  stripes.somewhere.com.
```

- **$INCLUDE**. Синтаксис: **$INCLUDE <имя файла> [<исходный домен>] [<комментарий>]**

  Директива **$INCLUDE** импортирует файл с указанным именем и обрабатывает его так, как если бы содержимое входило в файл зоны в месте указания директивы **$INCLUDE**. При обработке содержимого включенного файла директиве **$ORIGIN** устанавливается значение **<исходный домен>**, если оно задано.

- **$TTL**. Синтаксис: **$TTL <стандартное время жизни> [<комментарий>]**

  Директива **$TTL** устанавливает стандартное время жизни (Time-to-Live — TTL) для любой записи, в которой значение времени жизни не задано. Эта директива используется только для записей типа *отрицательного кэширования*, когда система BIND запоминает факт несуществования записи на определенный период (т.е. время жизни). В записи ресурса (Resource Record — RR), поле TTL идет перед столбцом класса. В записях, представленных в листинге 30.2, оно пустое (поле **IN** задает класс). Эти записи наследуют значение директивы **$TTL** (3600 секунд), указанное в начале листинга.

- **$GENERATE**. Синтаксис: **$GENERATE <диапазон> <левая часть> <тип> <правая часть> [<комментарий>]**

  Директива **$GENERATE** используется для создания набора записей, отличающихся *итератором*, или значением шага. Другими словами, можно автоматически задать большой список записей, имена и IP-адреса которых строятся по определенной формуле.

- **<диапазон>**. Может иметь две формы: **начало-конец** или **начало-конец/шаг**. При использовании первой формы шаг устанавливается равным 1.

- **<левая часть>**. Описывает, чем будут отличаться создаваемые записи левой части. Любой одиночный символ **$** в левой части заменяется значением итератора. Если **<левая часть>** представляет собой неабсолютное имя, к ней добавляется текущее значение директивы **$ORIGIN**. Если в левой части ничего не надо изменять, используется символ-заместитель **0**.

- **<тип>**. **PTR**, **CNAME** или **NS**.

- **<правая часть>**. То же, что и **<левая часть>**, но задает правую часть записи.

Следующие директивы **$GENERATE**

```
$ORIGIN 0.0.192.in-addr.arpa.
$GENERATE 1-2   0 NS    ns$.somewhere.com.
$GENERATE 1-127 $ CNAME $.0
```

раскрываются системой BIND в такой список:

```
0.0.0.192.in-addr.arpa.  NS        ns1.somewhere.com.
0.0.0.192.in-addr.arpa.  NS        ns2.somewhere.com.
1.0.0.192.in-addr.arpa.   CNAME   1.0.0.0.192.in-addr.arpa.
2.0.0.192.in-addr.arpa.   CNAME   2.0.0.0.192.in-addr.arpa.
...
127.0.0.192.in-addr.arpa.    CNAME   127.0.0.0.192.in-addr.arpa.
```

Записи SOA

Теперь мы добрались до первой записи ресурсов (Resource Record) в файле зоны. Это запись Start-of-Authority (**SOA**). Она является наиболее важной частью файла зоны, и именно ее чаще всего приходится исправлять. Основная причина — стиль ее задания не соответствует интуитивно предполагаемому. Придется рассмотреть эту запись по частям.

В листинге 30.2. представлен следующий пример записи SOA:

```
somewhere.com.  IN SOA stripes.somewhere.com.  root.somewhere.com. (
                       2001061000       ; Серийный номер
                       10800            ; Обновление
                       3600             ; Повторные попытки
                       604800           ; Устаревание
                       86400 )          ; Минимальное время жизни
```

Запись **SOA**, как и все записи, имеет следующую форму:

```
<имя>  [<время жизни>]  [<класс>]  <тип>  <значение>
```

Если указанное **<имя>** не абсолютно, добавляется текущее значение директивы **$ORIGIN**. В нашем случае использовано абсолютное имя (оно завершается точкой).

Поле **<время жизни>** не задано, поскольку в представленном файле имеется глобальная директива **$TTL**, которая его задает. Значение **IN** (сокращение от "Internet") — это класс записи. Система BIND поддерживает и другие классы, но они нас в данном случае не интересуют. Если значение **in** указано в операторе **zone** в файле **named.conf**, это поле — избыточное, и его тоже можно не указывать.

Поле **<значение>** для большинства записей — достаточно простое: IP-адрес, имя хоста или символическое имя. Запись **SOA**, однако, намного сложнее. Она начинается с имени авторитетного сервера имен в зоне (**stripes.somewhere.com**). Следующий фрагмент информации — адрес электронной почты администратора домена, в котором знак @ заменен точкой. Заканчивается адрес тоже точкой. В нашем примере адрес администратора (**root@somewhere.com**) задан как **root.somewhere.com.** и будет использоваться системой BIND для посылки ему уведомлений о состоянии.

ПРИМЕЧАНИЕ

Причина замены в адресе электронной почты администратора знака @ точкой состоит в том, что символ @ имеет в файле зоны специальное значение. Это сокращенная ссылка на текущее значение директивы **$ORIGIN**, и система BIND заменяет ее полным именем домена.

Учтите, что для успешного использования символ @ должен представлять полное имя; нельзя использовать его в качестве фрагмента, в виде **www.@** или других подобных конструкций.

После адреса администратора идет блок установок в круглых скобках. Круглые скобки задают блок, в котором переводы строк игнорируются, поэтому набор значений можно сформатировать так, чтобы он был понятнее. В блоке задаются числа, определяющие обработку данных зоны в сети Internet.

- **Серийный номер**. Серийный номер позволяет системе BIND следить за давностью файлов зон. При каждом обновлении файла зоны необходимо увеличивать серийный номер, чтобы сообщать системе BIND, что необходимо обновить информацию на основе данного файла. Обычно принято использовать для этого числа формат **ГГГГММДДNN**; последние две цифры предназначены для нескольких версий, созданных в течение одного дня. При внесении изменения обновите серийный номер, указав в нем дату изменения.

ПРИМЕЧАНИЕ

При выполнении команды **ndc reload** для обновления информации в главном файле зоны, система BIND автоматически определит, что данные необходимо обновить, независимо от значения серийного номера. Однако поддержка корректного серийного номера — хорошая привычка. Она обеспечит правильную работу в случае обновления данных подчиненной зоны вручную.

- **Обновление**. Это число задает (в секундах) частоту, с которой подчиненные серверы должны проверять изменение данных зоны на главном сервере. Если серийный номер главного файла изменился с момента последнего переноса информации зоны, будет выполнен новый перенос.
- **Повторные попытки**. Если не удается связаться с главным сервером, подчиненные серверы будут пытаться связаться повторно с интервалами, задаваемыми этим числом (в секундах).
- **Устаревание**. Если не удается связаться с главным сервером в течение этого времени (задается в секундах), подчиненные серверы отбрасывают все свои данные для соответствующей зоны.
- **Минимальное время жизни**. Это число задает (в секундах), сколько времени хранить в кэше "негативные " ответы (показывающие отсутствие определенных данных).

Записи NS

В листинге 30.2 представлены следующие примеры записей серверов имен (Name Server — NS):

```
; Серверы  DNS
@         IN  NS stripes.somewhere.com.
@         IN  NS spots.somewhere.com.
```

Эти записи задают серверы DNS, которым разрешается давать авторитетные ответы на запросы в отношении данной зоны. Учтите, что символ @ обозначает текущее значение директивы **$ORIGIN (somewhere.com.)** и что имена обоих серверов завершаются точками (показывающими, что они — абсолютные).

Можно создавать записи **NS** для поддоменов, или зон в текущей зоне. Например, пусть имеется кластер машин в зоне **cluster.somewhere.com.**, DNS-информация которого обслуживается отдельным сервером имен (**ns.cluster.somewhere.com**). Запись NS для этого

сервера, разрешающая внешним хостам запрашивать у него информацию об IP-адресах машин кластера, будет иметь вид:

```
cluster IN NS ns.cluster.somewhere.com.
```

Учтите, поскольку после имени **cluster** не указана завершающая точка, система BIND добавит к имени значение директивы **$ORIGIN** (**somewhere.com.**).

Записи адреса

Запись адреса (A) используется для связи имени хоста и IP-адреса. В нашем примере имеются следующие записи:

```
; Имена машин
localhost       IN  A       127.0.0.1
stripes         IN  A       64.41.131.162
spots           IN  A       64.41.131.163
mail            IN  A       64.41.131.167
@               IN  A       64.41.131.162
```

Значение этих записей вполне понятно. Неуточненные имена в левой части дополняются текущим значением директивы **$ORIGIN**, так что имени **mail.somewhere.com** будет соответствовать адрес **64.41.131.167**. Символ **@** заменяется так, что запрос адреса **somewhere.com** вернет значение **64.41.131.162**.

Имена, определенные в записях **A**, называют *каноническими*, в отличие от *псевдонимов* (которые задаются записями **CNAME**).

Записи CNAME

Записи канонического имени (Canonical Name — **CNAME**) используются для создания псевдонимов. В примере, представленном в листинге 30.2, имеем:

```
; Псевдонимы
www             IN  CNAME       @
ftp             IN  CNAME       www.somewhere.com.
```

Псевдоним **www** (который дополняется значением **$ORIGIN** до **www.somewhere.com.**) указывает на каноническое имя **@**, или **somewhere.com.** (IP-адрес которого задан в блоке **A**).

Записи **CNAME** часто используются, поскольку не привязываются непосредственно к IP-адресам. Можно использовать записи **CNAME**, например, для ссылки на имена в другой зоне, которая может управляться соответствующим владельцем. Записи **CNAME** также позволяют сократить количество изменений в файле зоны в случае изменения IP-адресов.

Записи MX

Записи Mail Exchanger (**MX**) определяют, какие хосты должны использоваться для доставки электронной почты адресатам в данной зоне. Система Sendmail (и другие агенты передачи почты) ищет запись **MX** с максимальным приоритетом в зоне и подключается к соответствующему хосту. Хосты **MX** домена можно найти с помощью команды **host**:

```
# host somecompany.com
somecompany.com has address 207.114.98.18
somecompany.com mail is handled (pri=100) by mail.uu.net
somecompany.com mail is handled (pri=5) by mx-1.somecompany.com
```

Сначала используются записи с меньшими значениями приоритета, а затем делаются попытки подключиться к хостам с большим приоритетом, пока не удастся ус-

пешно установить соединение по протоколу SMTP. Приоритет задается как часть поля значения в записи, как в нашем примере:

```
; Запись MX
@              IN  MX      10   mail.somewhere.com.
```

ПРЕДУПРЕЖДЕНИЕ

Запись **MX** не может ссылаться на IP-адрес. Кроме того, многие агенты передачи почты и почтовые клиенты будут сообщать об ошибках, если выявят запись **MX**, ссылающуюся на псевдоним (заданный записью **CNAME**). Убедитесь, что записи **MX** ссылаются на имена, заданные отдельно в записях **A**!

Записи PTR

Записи указателей (**PTR**) по смыслу противоположны записям **A** или **CNAME**, и используются в файлах обратного просмотра зон DNS (например, **131.41.64.in-addr.arpa**) для сопоставления IP-адресам доменных имен. Пример файла обратного просмотра зоны DNS представлен в листинге 30.3.

Поскольку обратный поиск в DNS возвращает только одно имя, несколько записей **PTR** для одного IP-адреса бесполезны и только сбивают с толку. Поэтому файлы обратного просмотра обычно короче, чем соответствующие файлы прямого просмотра.

Файлы обратного просмотра зоны DNS

Файл обратного просмотра зоны DNS, задающий зону в виде **131.41.64.in-addr.arpa**, используется для сопоставления имен IP-адресам. Файлы этого типа указываются в файле **named.conf**, и соответствующая информация распространяется подчиненным серверам точно так же, как и файлы прямого просмотра зон DNS. В листинге 30.3 представлен пример файла обратного просмотра зоны DNS.

Листинг 30.3. Пример файла обратного просмотра DNS для зоны 131.41.64.in-addr.arpa

```
$TTL  3600
131.41.64.in-addr.arpa. IN SOA stripes.somewhere.com. root.somewhere.com. (
                        2001061000   ; Серийный номер
                        10800        ; Обновление
                        3600         ; Повторные попытки
                        604800       ; Устаревание
                        86400  )     ; Минимальное время жизни

@       IN NS    stripes.somewhere.com.
@       IN  NS   spots.somewhere.com.

162     IN  PTR  stripes.somewhere.com.
163     IN  PTR  spots.somewhere.com.
167     IN  PTR  mail.somewhere.com.
```

Обратите внимание, что вместо записей **A** или **CNAMEN** используются записи **PTR**. Имя, на которое указывает запись **PTR**, — действительное каноническое имя для соответствующего IP-адреса. В файле обратного просмотра не нужна запись **MX**, поскольку записи **MX** нельзя непосредственно связывать с IP-адресами.

Создание файла зоны localhost

Для правильной работы необходимо создать специальный файл для зоны **localhost** (**0.0.127.in-addr.arpa**). В каталоге **/etc/namedb** имеется сценарий командного интерпретатора, помогающий создать такой файл. Этот сценарий называется **make-localhost**; он читает шаблон (**PROTO.localhost.rev**) и заполняет его введенной администратором информацией. Поскольку для сценария не установлено право на выполнение, необходимо запускать его командой **sh**:

```
# sh make-localhost
Enter your domain name: somewhere.com
```

Получающийся в результате файл, **localhost.rev**, должен выглядеть примерно так, как в листинге 30.4 (комментарии переведены на русский — прим. научн. ред.).

Листинг 30.4. Автоматически сгенерированный файл зоны localhost.rev

```
;     From: @(#)localhost.rev 5.1 (Berkeley) 6/30/90
;  $FreeBSD: src/etc/namedb/PROTO.localhost.rev,v 1.6 2000/01/10
15:31:40 peter Exp $
;
; Этот файл автоматически создается сценарием 'make-localhost'
; в каталоге /etc/namedb.
;
$TTL 3600
@     IN    SOA   stripes.somewhere.com.  root.stripes.somewhere.com.  (
                        20010612    ; Серийный номер
                        3600        ; Обновление
                        900         ; Повторные попытки
                        3600000     ; Устаревание
                        3600 )      ; Минимальное время жизни
      IN    NS    stripes.somewhere.com.
1     IN    PTR    localhost.somewhere.com.
```

Не забудьте создать этот файл, прежде чем начнете эксплуатировать службу DNS!

Конфигурирование кэширующего сервера имен

Вполне можно запустить сервер имен, не являющийся авторитетным ни для одной из зон. Такой сервер называют *кэширующим сервером имен*, а его задача состоит исключительно в выполнении запросов DNS по требованиям клиентов и запоминании результатов для дальнейшего использования. Его роль противоположна роли чисто перенаправляющего сервера, конфигурирование которого было рассмотрено ранее: тогда как перенаправляющий сервер имен передает все поступающие ему запросы на обработку другому серверу, кэширующий сервер обрабатывает все поступившие запросы сам и сохраняет результаты в кэше. Другие серверы могут использовать этот сервер как перенаправляющий, используя результаты выполненной им работы.

Чтобы сконфигурировать кэширующий сервер имен, просто не указывайте в файле **named.conf** никакие операторы **zone**, кроме тех, что необходимы для работы самого сервера. Такая конфигурация представлена в листинге 30.5.

Листинг 30.5. Пример файла /etc/namedb/named.conf

```
/*
 * Пример конфигурации системы BIND 8
 */
logging {
    category lame-servers { null; } ;
    category cname { null; } ;
};
options {
    directory "/etc/namedb";
};
zone "." {
    type hint;
    file "named.boot";
};
zone "0.0.127.in-addr.arpa" in {
    type master;
    file "localhost.rev";
};
zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.INT" {
    type master;
    file "localhost.rev";
};
```

Поскольку кэширующий сервер имен является просто "урезанным вариантом" полнофункционального сервера имен, со временем можно расширить функциональные возможности сервера, добавляя дополнительные зоны. Между этими двумя "режимами" нет принципиальных отличий.

Конфигурация службы DNS будет изменяться со временем при развитии сети. Конфигурацию придется изменять при каждом добавлении нового хоста к сети или изменении имени хоста, а также при каждом добавлении или удалении авторитетного сервера имен для зоны. Имеет смысл потренироваться в администрировании службы DNS, чтобы в дальнейшем подобные задачи не представляли трудности.

ГЛАВА 31

Сетевая файловая система (NFS)

- Основы NFS
- Конфигурирование сервера NFS
- Конфигурирование клиента NFS
- Демон автоматического монтирования (amd)

Сетевая файловая система NFS (Network Filesystem) обеспечивает совместное использование файлов в ОС UNIX. В операционных системах Windows и Mac OS есть свои механизмы совместного использования файлов, позволяющие подключенным к сети компьютерам обращаться к файлам на удаленных машинах в локальной сети так, как если бы они находились на их собственном диске. Система NFS обеспечивает те же преимущества и, кроме того, ряд возможностей, отсутствующих в других протоколах совместного использования файлов.

В этой главе описано конфигурирование компьютера под управлением ОС FreeBSD для работы в качестве клиента и/или сервера NFS, т.е. для совместного использования файлов с другими компьютерами, работающими под управлением ОС UNIX, как в локальной сети, так и в Internet.

Основы NFS

ОС Windows использует для совместного доступа к файлам протокол NetBIOS, а Mac OS — протокол AppleTalk. Оба эти протокола — двухточечные; каждая система сообщает остальным о своем присутствии в сети широковещательной рассылкой, и все машины могут динамически монтировать предоставленные в совместный доступ каталоги друг друга. Система NFS немного отличается тем, что использует протокол типа клиент-сервер, явно выделяя серверы, предоставляющие в совместный доступ ресурсы, которые могут быть смонтированы определенными удаленными клиентами NFS. Эта модель создавалась для централизации на предприятии или в университетской сети, а не для двухточечных передач, которые в основном используются в других протоколах. Однако при соответствующем конфигурировании сетевая файловая система NFS позволяет получить не только все, что обеспечивают другие протоколы, но и намного больше, за исключением выявления серверов.

Под ОС Windows и Mac OS встроенные протоколы совместного использования файлов на компьютере рассылают информацию о предоставленных для общего доступа собственных ресурсах (или "общих ресурсах" — "shares") и делают запросы по поводу других имеющихся в сети общих ресурсов. Эти "выявляющие" запросы, посылаемые каждой машиной в сети и вызывающие быстрые ответы всех остальных машин, обуславливают весьма "разговорчивую" сетевую среду. Система NFS не имеет соответствующего механизма выявления. Как будет показано, каждый клиент NFS должен знать, где искать каждый из серверов, и монтировать ресурсы явно. Однако это означает, что объем передаваемой по сети информации уменьшается за счет отсутствия соответствующих запросов выявления ресурсов и ответов на них.

Структура NFS, однако, имеет преимущества. Сервер явно управляет тем, какие клиенты могут к нему подключаться в зависимости от имени хоста или IP-адреса или с помощью централизованной службы регистрации, например NIS или Kerberos. Еще одно полезное свойство системы NFS состоит в том, что, поскольку она не зависит от широковещательной рассылки в локальной сети, использующейся для выявления серверов, ее можно использовать по сети Internet точно так же, как и в локальной сети. Клиент в Бостоне может при необходимости смонтировать ресурс сервера в Сан-Франциско. Протоколы же NetBIOS и AppleTalk могут работать только в "доменах" или "зонах" в локальной сети.

Что касается ОС FreeBSD, NFS для нее — такая же файловая система, как и любая другая. Общий ресурс NFS можно смонтировать по сети так же, как и дискету или новый

раздел жесткого диска. Эта операция рассматривалась в главе 9. Общие ресурсы могут даже монтироваться автоматически при обращении к ним, если клиентская машина настроена соответствующим образом. Процедура монтирования общего ресурса NFS будет описана в этой главе позже.

Клиент-серверная структура NFS спроектирована так, чтобы можно было централизовать ресурсы в сети. Например, на предприятии можно создавать начальные каталоги для всех сотрудников на центральном сервере UNIX, и любая другая система в сети, поддерживающая работу с NFS, сможет смонтировать эти начальные каталоги и работать с ними удаленно, — не придется создавать копию каждого начального каталога на всех машинах. На рис. 31.1 показано применение такого сетевого решения. То же самое можно сделать для каталогов проектов (в среде разработки программного обеспечения) или совместно используемых приложений, которые устанавливаются централизованно (например, в кластере рабочих станций университета). Монтируемые ресурсы NFS можно использовать совместно со службой имен NIS (централизованное управление регистрацией) для аутентификации пользователей во всей сети. В этом случае принадлежность и права доступа ко всем файлам смонтированного общего ресурса будут такими же, как и на самом сервере NFS. Можно даже устанавливать ОС FreeBSD по протоколу NFS, если смонтировать установочный CD-ROM на сервере NFS и указать его программе Sysinstall.

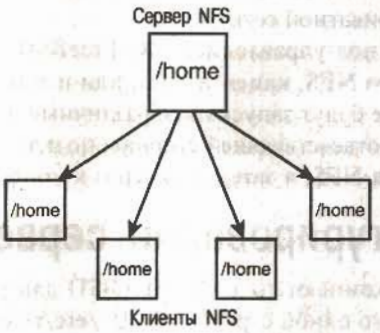


Рисунок 31.1.
Сеть предприятия с централизованными, монтируемыми по NFS начальными каталогами пользователей

Интересно, что служба NFS в основном базируется на протоколе UDP, а не на TCP. Как обсуждалось в главе 22, протокол UDP не имеет ничего, подобного средствам обеспечения надежности передачи или поддержки подключения протокола TCP, и, по сути, ненадежен. Хотя и кажется странным, что ориентированный на работу с файлами сетевой протокол использует в качестве транспортного механизма протокол UDP, выбрасывая дейтаграммы с сервера на клиент без всяких гарантий их целостности, фактически протокол NFS — один из лучших примеров правильного использования транспортного протокола UDP. Постоянные подключения, как в случае TCP, на самом деле не нужны, особенно в локальной сети. Клиенты NFS появляются и исчезают по мере запуска и останова машин в сети, и подключения TCP будут только мешать потокам данных файлов и каталогов, пересылаемых между конечными точками. Смонтированный по NFS ресурс может оставаться невостребованным целыми днями, а затем неожиданно его могут снова начать использовать. Протокол UDP обеспечивает немедленную доступность общих ресурсов без издержек, связанных с организацией подключения или ожиданием, а если хост отключается от сети, на сервере это не сказывается.

Целостность данных обеспечивается самим программным обеспечением системы NFS, передающим контрольные суммы, разбивающим данные на пакеты и устанав-

ливающим последовательность пакетов, и вообще, выполняющим все действия, обычно возлагаемые на протокол TCP. Однако клиенты системы NFS, работающие поверх протокола TCP, существуют, и сервер NFS их также поддерживает. Поддержка NFS поверх протокола TCP в основном предназначена для монтирования ресурсов по глобальным сетям, где потеря пакетов намного вероятнее, а преимущества постоянных подключений TCP существеннее, чем важные в локальной сети преимущества простого протокола UDP.

ПРИМЕЧАНИЕ

Один из недостатков использования протокола UDP состоит в том, что при попытке связаться с неподключенным сервером NFS ждать результатов подключения можно вечно. Отсутствие установленного подключения в стиле TCP означает, что для выявления не отвечающего сервера необходимы более примитивные методы. Далее в этой главе будет рассмотрено, как избежать длительного периода ожидания подключения по NFS.

Протокол NFS не содержит встроенных механизмов защиты или шифрования данных, поэтому использовать его для подключения через Internet можно только при работе с файлами, содержимое которых не является секретным. Важные или секретные данные никогда нельзя передавать по протоколу NFS через глобальную сеть. Это можно делать только в локальной сети, защищенной брандмауэром или через шифруемый тоннель виртуальной приватной сети (VPN).

Компьютер под управлением ОС FreeBSD можно сконфигурировать для работы в качестве сервера NFS, клиента NFS, или и того, и другого. В зависимости от конфигурации в системе будут запускаться различные небольшие группы процессов, управляющие работой соответствующей стороны подключения NFS. Сначала мы рассмотрим настройку сервера NFS, а затем перейдем к конфигурированию клиента.

Конфигурирование сервера NFS

Настройка компьютера с ОС FreeBSD для работы в качестве сервера NFS требует добавления всего одной строки в файл **/etc/rc.conf**:

```
nfs_server_enable="YES"
```

Убедитесь также, что параметр **portmap_enable** имеет значение **"YES"** (как и было по умолчанию, если вы не изменили его). Демон **portmap** необходим для функционирования системы NFS, поскольку NFS-серверу требуется механизм сообщения клиентам о том, к какому порту подключаться. Службы UDP работают с единственным "подключением" к порту, поэтому, хотя начальные соединения с сервером NFS выполняются через порт 2049, они сопоставляются службами RPC (удаленные вызовы процедур) с новыми, неиспользуемыми портами сервера. Поддержку служб RPC и предоставляет демон **portmap**. Учтите, что службы **portmap** весьма небезопасны и подвержены частным взломам, поэтому не стоит предоставлять доступ к серверу NFS из Internet, не защитив его брандмауэром. Подробнее о брандмауэрах см. в главе 29.

После установки опции **nfs_server_enable** и перезагрузки, ОС FreeBSD запускает два вида процессов-демонов (точнее, — три, с учетом **portmap**) после прочтения содержимого файла **/etc/exports**, задающего ресурсы, предоставляемые в общий доступ через NFS. Давайте рассмотрим каждый из этих демонов и разберемся в их назначении и конфигурировании.

Демон NFS (nfsd)

Демон сервера NFS, эквивалентный по назначению демонам **sshd** или **httpd**, называется **nfsd**. Несколько процессов **nfsd** запускаются при инициализации сети (по умолчанию — четыре); эти процессы должны обслуживать подключения клиентов NFS, каждый процесс — для одного клиента. В стандартной конфигурации к серверу NFS одновременно может подключаться не более четырех клиентов. Это количество можно изменить с помощью параметров **nfs_server_flags** в файле **/etc/rc.d**:

```
nfs_server_flags="-u -t -n 4"
```

Опции **-n** можно установить любое значение; установите его равным предполагаемому максимальному количеству одновременно подключающихся к серверу клиентов NFS. Опции **-t** и **-u** заставляют процессы **nfsd** обслуживать клиентов, подключающихся как по протоколу TCP, так и по UDP. Имеются и другие опции настройки. Например, указав **-h 64.41.131.102** среди других опций, можно привязать серверы NFS к интерфейсу с адресом **64.41.131.102**. (Вместо IP-адреса можно использовать имя хоста.) Это может потребоваться при наличии нескольких сетевых интерфейсов на сервере. Механизм реализации протокола UDP устроен так, что если не указать явно демону **nfsd**, запросы к какому IP-адресу он должен обслуживать, то его ответы не обязательно будут поступать с того адреса, с которым связывался клиент. Можно указать любое необходимое количество адресов с помощью нескольких опций **-h**. Например, пусть имеется сервер, работающий в качестве шлюза с двумя сетевыми картами — одна для "внутренней", а вторая — для "внешней" сети. С "внешней" картой связаны два IP-адреса, и необходимо обслуживать общие ресурсы NFS только для внутренних клиентов. Для этого можно настроить сервер NFS следующим образом:

```
nfs_server_flags="-u -t -n 4 -h 64.41.131.102 -h 64.41.131.116"
```

Затем можно заблокировать доступ к службе NFS по внешнему интерфейсу с помощью следующего правила системы IPFW:

```
ipfw add deny udp from any to 64.41.131.10 nfsd
```

Подробнее о конфигурировании брандмауэра на базе системы IPFW см. в главе 29.

Демон монтирования NFS (mountd)

Если **nfsd** — программа, обеспечивающая подключение для каждого отдельного монтирования ресурса NFS, то вторая программа службы отвечает за прослушивание сети в ожидании новых запросов клиентов NFS (поступающих по протоколу TCP к порту 2049). Это программа **mountd**, демон монтирования. Она запускается автоматически вместе с процессами **nfsd** в ходе процедуры начальной настройки сети при загрузке, если параметр **nfs_server_enable** имеет значение **"YES"**.

Демон **mountd** принимает запросы на подключение к NFS и передает их процессам **nfsd**. Он также следит за общими ресурсами NFS, указанными в файле **/etc/exports**; как будет вскоре продемонстрировано, для перезапуска служб NFS после изменения файла **/etc/exports** необходимо послать сигнал **HUP** демону **mountd**.

Полезных опций запуска демона **mountd** немного. Стандартные установки в файле **/etc/defaults/rc.conf**, которые можно переопределить в файле **/etc/rc.conf**, имеют следующий вид:

```
mountd_flags="-r"
```

Опция **-r** позволяет демону **mountd** обслуживать обычные файлы, а не только каталоги, для совместимости с некоторыми бездисковыми рабочими станциями, загружающимися с помощью службы NFS. Опция **-l** позволяет регистрировать все запросы монтирования NFS, а опция **-n** обеспечивает возможность монтирования общих ресурсов NFS на удаленные системы вроде ПК с ОС Windows, не поддерживающие модель владения и прав доступа ОС UNIX.

Задание общих ресурсов в файле /etc/exports

В файле **/etc/exports** перечислены каталоги, которые должны быть предоставлены для общего доступа через NFS, а также пользователи и системы, имеющие право доступа к ним. Если файл **/etc/exports** не существует или недоступен для чтения при запуске сети, процессы **nfsd** и **mountd** не запускаются.

Полный формат файла **/etc/exports** описывается на странице справочного руководства **man exports**. Строка экспорта состоит из одного или нескольких имен каталогов, которые экспортируются (предоставляются для общего доступа), опций экспорта и необязательного списка хостов (задаваемых IP-адресом, именем сети, сетевой группой или по имени), которым разрешается использовать соответствующие каталоги. Например, следующая строка предоставляет в общий доступ каталог **/home** и все его подкаталоги для любого подключившегося хоста:

```
/home  -alldirs
```

Учтите, что опция **-alldirs** может указываться, только если общий ресурс является точкой монтирования физической файловой системы (например, **/usr** или **/home**). В противном случае при ее указании доступ к ресурсу не будет предоставляться.

Общий ресурс, доступ к которому только для чтения могут получить три указанных хоста, задается следующим образом:

```
/usr2 -ro -alldirs stripes.somewhere.com spots.somewhere.com 64.41.131.165
```

СОВЕТ

Можно создавать сетевые группы хостов, указывая их в файле **/etc/netgroup**. Группа задается следующим образом:

```
имя_группы (хост, пользователь, домен) (хост, пользователь, домен) ...
```

Например, для создания группы **desktops**, содержащей три определенных хоста (с именами **sol**, **luna** и **terra**), необходимо добавить следующую строку:

```
desktops  (sol,,)  (luna,,)  (terra,,)
```

Сетевая группа на базе имен пользователей будет иметь такой вид:

```
developers  (,frank,)  (,bob,)  (,alice,)
```

Затем можно использовать любую из этих имен сетевых групп из файла **netgroup** вместо имен хостов в файле **/etc/exports**, выделяя общий ресурс NFS только членам соответствующей группы.

Принадлежность файлов общего ресурса NFS устанавливается на основе числовых идентификаторов владельцев файлов и каталогов. Если имена и идентификаторы пользователей на сервере и на клиентских машинах совпадают (например, если учетные записи машин синхронизированы с помощью служб NIS или Kerberos), права доступа также будут совпадать. Однако если идентификатор пользователя 1045 на сервере соответствовал регистрационному имени **bill**, а на клиенте идентификатор пользователя 1045 соответствует регистрационному имени **john**, владельцем файлов

общего ресурса будет Джон, а не Билл, как "считает" сервер. При экспортировании общего ресурса, содержащего файлы многих различных пользователей, убедитесь в наличии инфраструктуры, обеспечивающей согласованное сопоставление идентификаторов и имен пользователей на всех машинах сети.

Опции **-maproot=<имя пользователя>** или **-maproot=<идентификатор пользователя>** позволяют сопоставить права владения так, что пользователь с соответствующим именем или идентификатором на клиентской машине получит полные права доступа пользователя **root** на общем ресурсе. Например, следующая запись предоставляет всю файловую систему сервера NFS любому хосту в сети **64.41.131** так, что пользователь **frank** на клиентах будет иметь полный доступ на чтение и запись ко всем файлам:

```
/  -maproot=frank  -network  64.41.131  -mask  255.255.255.0
```

После внесения изменений в файл **/etc/exports** необходимо перезапустить процесс **mountd**. Для этого следует указать идентификатор соответствующего процесса, содержащийся в файле **/var/run/mountd.pid**:

```
#  kill  -HUP  `cat  /var/run/mountd.pid`
```

ПРИМЕЧАНИЕ

Нельзя задавать несколько строк экспорта для точек монтирования в одном разделе или физической файловой системе. Это позволяет предотвратить проблемы в случае конфликта прав доступа к различным общим ресурсам в пределах одной файловой системы. Клиенты NFS не могут отдельно обращаться к файловой системе, смонтированной в каталог уже смонтированного общего ресурса, причем одни и те же права доступа должны применяться для всех общих ресурсов в одной физической файловой системе. Следующая конфигурация недопустима:

```
/home/frank  64.41.131.102
/home/joe  64.41.131.102
```

А вот так правильно:

```
/home/frank  /home/joe  64.41.131.102
```

Для получения списка всех имеющихся общих ресурсов и прав доступа к ним можно использовать программу **showmount**. Вот как можно проверить, правильно ли файл **/etc/exports** настроен:

```
# showmount -e
Exports  list  on  localhost:
/usr                          Everyone
/home/frank                   64.41.131.102
/home/joe                     64.41.131.102
/                             64.41.131.0
```

Запуск служб NFS без перезагрузки

Наиболее простой способ запуска служб NFS — перезагрузка системы. Однако если необходимо запустить эти службы, а перезагрузка нежелательна, выполните следующие команды от имени пользователя **root** (команду **portmap** можно не вводить, если соответствующий демон уже работает):

```
# portmap
# nfsd  -u  -t  -n  4
# mountd  -r
```

Затем используйте команду **showmount -e**, чтобы проверить, правильно ли экспортированы общие ресурсы NFS.

Конфигурирование клиента NFS

Если компьютер под управлением ОС FreeBSD будет монтировать общие ресурсы NFS с других серверов, необходимо сконфигурировать его в качестве клиента. С технической точки зрения это совсем не обязательно — можно монтировать общий ресурс NFS примитивным способом безо всяких предварительных настроек. Однако конфигурирование системы в качестве клиента NFS дает дополнительные возможности и гарантирует скорость и надежность работы.

Для настройки клиентской машины NFS включите следующую строку в файл **/etc/rc.conf**:

```
nfs_client_enable="YES"
```

Эта установка включает демон ввода/вывода NFS, **nfsiod**, помогающий ускорить выполнение клиентских запросов NFS и настраивающий несколько параметров ядра, чтобы уменьшить время доступа. Все это делается автоматически сценарием **/etc/rc.network** во время загрузки, наряду с установками сервера NFS (рассмотренными выше).

Демон ввода/вывода NFS (nfsiod)

Демон **nfsiod** не обязателен для функционирования клиента NFS, но помогает ускорить работу. Он позволяет выполнять операции чтения и записи NFS асинхронно, так что "опережающее чтение" и "запись с задержкой " выполняются в фоновом режиме и не приходится ждать завершения каждого последовательного шага процесса. Как и в случае с демоном **nfsd**, должно работать столько процессов **nfsiod**, сколько общих ресурсов NFS смонтировано на клиентской машине. Количество процессов **nfsiod** (по умолчанию — 4), можно задать с помощью параметра **nfs_client_flags** в файле **/etc/rc.conf**:

```
nfs_client_flags="-n 4"
```

Больше никаких опций у демона **nfsiod** нет. Для запуска его без перезагрузки машины выполните команду:

```
# nfsiod -n 4
```

Монтирование удаленных файловых систем

Монтирование общего ресурса NFS выполняется с помощью команды **mount_nfs**, являющейся сокращенным вариантом стандартной команды **mount -t nfs** (как было описано в главе 9). Как правило, этой команде передается два аргумента: имя хоста и имя общего ресурса в виде скомбинированной строки, и локальная точка монтирования:

```
# mount_nfs spots:/home /home2
```

При успешном монтировании на экран ничего не выдается. Успешность монтирования можно проверить с помощью команды **df**:

```
# df
Filesystem     1K-blocks      Used     Avail Capacity  Mounted on
/dev/ad0s1a       992239     54353    858507       6%  /
/dev/ad0s1f     26704179   4872963  19694882      20%  /home
/dev/ad0s1e      9924475   1642343   7488174      18%  /usr
procfs                 4         4         0     100%  /proc
spots:/home      9924475   1642343   7488174      18%  /home2
```

Если перейти в каталог **/home2**, там будут все подкаталоги каталога **/home** на сервере NFS, причем принадлежность каждого файла устанавливается на основе

соответствия идентификаторов владельца, как было описано ранее. Файловая система останется смонтированной, пока не будет явно демонтирована с помощью команды **umount**:

```
# umount /home2
```

ПРЕДУПРЕЖДЕНИЕ

Не забудьте выйти из всех смонтированных по NFS каталогов, прежде чем пытаться демонтировать их с помощью **umount**! При попытке демонтировать используемую файловую систему вы получите сообщение об ошибке **"device busy"**.

Общие ресурсы NFS можно монтировать множеством различных способов; соответствующие опции описаны на странице справочного руководства **man mount_nfs**. Чаще всего используются опция **-T**, требующая использования транспортного протокола TCP вместо UDP (полезно при монтировании по глобальным сетям), и опции **-s** и **-x <секунд>**, благодаря которым точка монтирования устаревает и исчезает в случае сбоя или по истечении указанного периода времени (так называемое "мягкое" монтирование).

```
# mount_nfs -s -x 60 spots:/home /home2
```

СОВЕТ

Еще одна полезная опция — **-i**, обеспечивающая поддержку прерывания работы. Обычно если смонтирован общий ресурс NFS и соответствующий сервер не отвечает или недоступен, любые обращения к файловым системам (команды работы с общими файлами, например **ls**) могут зависать так, что даже нажатие комбинации клавиш Ctrl+C их не остановит. Опция **-i** позволяет смонтировать ресурс так, что нажатие Ctrl+C (сигнал прекращения работы) приводит к сбою команды, возвращая управление пользователю.

Как и для других типов файловых систем, можно добавить описание монтируемых ресурсов NFS в файл **/etc/fstab** для настройки заданных точек монтирования, что упрощает процесс монтирования. Поместите любые опции, передаваемые команде **mount_nfs**, в столбец Options, разделив их запятыми:

```
# Device      Mountpoint  FStype      Options          Dump  Pass#
spots:/home   /home2      nfs         rw,-T,-i,noauto  0     0
```

При наличии такой записи можно монтировать файловую систему NFS с помощью простой команды **mount**:

```
# mount /home2
```

Автоматическое монтирование удаленных файловых систем при загрузке системы

Все файловые системы, описанные в файле **/etc/fstab**, автоматически монтируются при загрузке, как было описано в главе 9, если только для них не указана опция **noauto**. Можно задать автоматическое монтирование общих ресурсов NFS при загрузке, добавив их описания в файл **/etc/fstab**, как было показано в предыдущем разделе. Однако надо учесть ряд обстоятельств.

Наиболее существенно то, что система NFS имеет исключительно продолжительный стандартный период ожидания, а фаза монтирования файловых систем при запуске — синхронный процесс, блокирующий дальнейшее продолжение работы. Если сервер или серверы NFS не удается найти (например, если соответствующий компьютер выключен или подключение клиентской машины к сети неправильно сконфи-

гурировано), процесс загрузки может остановиться на недопустимо продолжительное время, прежде чем попытки монтирования будут прекращены как неудавшиеся и процедура загрузки будет продолжена.

Эту проблему можно решить, поместив опцию **noauto** в файл **/etc/fstab**, как было показано в примере выше. Однако это означает, что придется монтировать каждый общий ресурс NFS вручную после полной загрузки системы. Есть более удобный способ справиться с этой проблемой — задать опцию **-b**.

```
# Device        Mountpoint  FStype      Options      Dump  Pass#
spots:/home     /home2      nfs         rw,-b        0     0
```

Задание опции **-b** приводит к тому, что команда **mount** пытается связаться с сервером и, если это сделать не удается, запускает порожденный процесс, выполняющий повторные попытки подключиться, пока продолжается процесс загрузки. Точно также при попытке монтирования общего ресурса из командной строки с опцией **-b**, запускается порожденный фоновый процесс, а вы сразу же получаете приглашение командного интерпретатора. Вот какой результат получается при попытке смонтировать общий ресурс, указанный в предыдущем примере строки из файла **/etc/fstab**, после выполнения попыток монтирования в течение 60 секунд:

```
# mount /home2
NFS Portmap: RPC: Port mapper failure - RPC: Unable to send
mount_nfs: Cannot immediately mount spots:/home, backgrounding
```

Это можно перевести следующим образом:

```
NFS Portmap: RPC: Сбой программы сопоставления портов - RPC; не могу
послать
mount_nfs: не могу немедленно смонтировать spots:/home, перехожу в
фоновый режим
```

Фоновый процесс **mount_nfs** будет продолжать попытки смонтировать общий ресурс, пока успешно его не смонтирует. Этот метод особенно удобен в вычислительных кластерах или лабораториях, где смонтированные ресурсы NFS облегчают работу, но не обязательны для правильного функционирования. Примером может служить кластер, в котором монтируемый по NFS ресурс содержит популярные пользовательские программы или игры, но все критически важные системные функции обеспечиваются программами на локальных дисках систем.

Демон автоматического монтирования (amd)

Демон автоматического монтирования, **amd**, делает работу с общими ресурсами NFS еще удобнее. Этот демон позволяет монтировать общие ресурсы NFS (на самом деле все типы файловых систем) динамически, при переходе в каталог, в который должен быть смонтирован ресурс, не вводя вообще никаких команд монтирования.

ОС FreeBSD обеспечивает простой способ настройки демона **amd**. Добавьте следующую строку в файл **/etc/rc.conf**:

```
amd_enable="YES"
```

При загрузке системы с таким параметром демон **amd** запускается с опциями, задаваемыми параметром **amd_flags**, стандартное значение которого обеспечивает автоматическое монтирование по имени всего содержимого каталогов **/host** или **/net**,

которые автоматически создаются демоном **amd**. В таком режиме демон запускается и вручную, с помощью следующей команды:

```
# amd -a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map
```

При работающем демоне **amd** перейдите с помощью команды **cd** в каталог **/host** и просмотрите его содержимое. Каталог пуст.

```
# cd /host
# ls
#
```

Однако попытайтесь получить каталог по имени, как если бы там уже был подкаталог, имя которого совпадает с именем одного из известных серверов NFS в сети:

```
# ls stripes
home
```

Итак, кажется, в каталоге **/host** действительно есть каталог **stripes**, а в нем — подкаталог **home**, содержащий то же, что и общий ресурс **stripes:/home**. Вы только что автоматически смонтировали этот общий ресурс в каталог **/host**, просто просмотрев его по имени. Функционально **/host/stripes/home** не отличается от точки монтирования **/home2**, созданной вручную в одном из предыдущих примеров. Команда **df** позволяет в этом убедиться:

```
# df
Filesystem          1K-blocks     Used    Avail  Capacity  Mounted  on
stripes:/home         9924475  1642345  7488172     18%
/.amd_mnt/stripes/host/home
```

ПРИМЕЧАНИЕ

Обратите внимание, что общие ресурсы NFS, смонтированные таким образом, фактически монтируются в подкаталог **.amd_mnt** корневого каталога. Этот каталог реально не используется — это лишь сокращение, применяемое демоном **amd** для отслеживания ресурсов.

Чтобы задать постоянное местонахождение автоматически монтируемого ресурса NFS, создайте символическую связь с соответствующим подкаталогом **/host** или **/net**:

```
# ln -s /host/stripes/home /home2
```

С этого момента при переходе в каталог **/home2** общий ресурс **stripes:/home** будет автоматически монтироваться, и вы получите доступ к соответствующим файлам. Неиспользуемый общий ресурс автоматически демонтируется.

СОВЕТ

Можно создавать намного более сложные карты монтирования для демона **amd**, обеспечивающие непосредственное монтирование файловых систем в определенные каталоги, задавая записи в файле **/etc/amd.conf**. Этот файл не существует после стандартной установки ОС FreeBSD; подробнее о его формате и предоставляемых возможностях см. на странице справочного руководства **man amd.conf**.

32 ГЛАВА

Совместное использование файлов и принтеров с Microsoft Windows

- Введение в систему Samba
- Особенности работы протоколов SMB/CIFS
- Установка и конфигурирование системы Samba
- Другие компоненты системы Samba
- Дальнейшее развитие системы Samba
- Файловая система smbfs

Система NFS — отличное решение проблемы совместного использования файлов UNIX-машинами, когда права доступа к файлам UNIX и метаданные файлов (такие как время последнего изменения) должны сохраняться при обращении с различных машин. Однако система NFS мало распространена в большинстве пользовательских операционных систем. ОС Windows и классические версии Mac OS поддерживают ее только с помощью приложений сторонних производителей. Кроме того, в системе NFS нет встроенного механизма "выявления", позволяющего клиентам просматривать списки доступных серверов.

Большинство компьютеров в локальных сетях чаще всего работают под управлением Windows. Поэтому при включении компьютера с ОС FreeBSD в существующую сеть для полномасштабного взаимодействия с этими клиентами и файл-серверами под управлением Windows необходимо, чтобы ОС FreeBSD поддерживала те же методы совместного использования файлов, что и Windows. Этот метод — *протокол SMB*, Server Message Block, и *общая сетевая файловая система*, Common Internet File System (*CIFS*), которая его постепенно вытесняет.

Средства совместного использования файлов по протоколам SMB/CIFS не встроены в ОС FreeBSD. Однако дополнительный пакет, который называется *Samba*, предоставляет машине под управлением ОС FreeBSD возможность работать в качестве файл-сервера Windows и участвовать в тех же процессах совместного использования файлов, что и реальные клиенты Windows.

Введение в систему Samba

Система Samba — это некоммерческий проект с открытым исходным кодом, начатый Эндрю Триджеллом (Andrew Tridgell), а сейчас разрабатываемый совместными усилиями сообщества разработчиков UNIX. Система Samba обеспечивает UNIX-машине (например, с ОС FreeBSD) возможность воспользоваться всеми преимуществами совместного доступа к файлам Windows, включая появление машины в списке ресурсов сети, защиту подключений на основе доменов NT и регистрации пользователей, а также поддержку сетевых служб печати. Имеется также ряд инструментальных средств, обеспечивающих выполнение многих административных функций сервера Windows NT/2000. После добавления портированной реализации файловой системы **smbfs**, которую мы рассмотрим в конце главы (она позволяет ОС FreeBSD работать в качестве клиента файловых ресурсов Windows), получается полный набор программного обеспечения, дающий компьютеру с ОС FreeBSD в сетевой среде Windows те же функциональные возможности, что и настоящей Windows-машине.

Официальный Web-сайт системы Samba находится по адресу **http://www.samba.org**, — там можно выбрать любой из географически близких зеркальных сайтов.

Особенности работы протоколов SMB/CIFS

Протокол SMB, ведущий свое начало от документов, опубликованных в 1985 году компанией IBM, а затем получивший дальнейшее развитие усилиями компаний Microsoft и Intel, — это обобщенная система совместного использования всех видов системных ресурсов в локальной сети. Речь идет о таких ресурсах как файлы, принтеры, последовательные порты и программные абстракции вроде именованных каналов. Это протокол работает в стиле клиент-сервер, хотя построенный на его основе общий доступ к файлам

в Windows на первый взгляд имеет двухточечную организацию. Протокол SMB — фундаментальная часть многих операционных систем, включая MS-DOS, Windows, OS/2 и Linux, — хотя сегодня он в основном используется в среде Windows и навязывается компанией Microsoft.

Команды протоколов SMB и CIFS посылаются поверх базовых сетевых протоколов, таких как IPX, NetBEUI, Banyan VINES и DECnet. Эти протоколы работают на "сетевом" уровне стека — на том же уровне, что и протокол IP (как было показано в главе 22), и поэтому не ограничены только транспортными протоколами стека TCP/IP. Однако чаще всего в качестве транспортного протокола SMB используется реализация NetBIOS (Network Basic Input/Output System — базовая сетевая система ввода/вывода, описанная в рабочих документах RFC 1001 и 1002) поверх IP, работающая с компонентами как TCP, так и UDP. Именно такое семейство протоколов используется при реализации совместного доступа к файлам в Windows.

Просмотр

Преимущество протокола SMB перед другими протоколами, вроде NFS, — поддержка автоматического выявления серверов, или *просмотра* (browsing). В среде Windows при открытии окна Мое сетевое окружение (Network Neighborhood) в нем отображаются имена всех доступных в локальной сети серверов протокола SMB. Этот список строится динамически путем периодической посылки каждой машиной широковещательных пакетов, запрашивающих "главный браузер" ("master browser") сети (компьютер с полным списком локальных и удаленных хостов SMB) и анонсирующих ее присутствие в сети. Все остальные машины в сети строят свои "списки просмотра" ("browse list") на основе данных этой широковещательной рассылки.

Имя каждой машины, представленное в окне программы просмотра сети (см. рис. 32.1), — это ее "имя NetBIOS", т.е. идентификатор, длина которого в ОС Windows должна быть не более 15 символов. Хотя Windows заставляет вводить имя NetBIOS в верхнем регистре, в окне просмотра сети оно выдается в нижнем регистре, с прописной буквы. В других операционных системах (таких как FreeBSD) имя NetBIOS совпадает с именем хоста, усеченным при необходимости до 15 символов.

Имена NetBIOS обрабатываются своеобразной службой имен — неким подобием службы имен DNS, но сопоставляет выдаваемые NetBIOS-имена машин конкретным машинам на основе других критериев, а не только IP-адреса (поскольку протокол NetBIOS не ограничен только средой IP). Компонент сервера имен системы Samba отделен от фактического сервера данных SMB, как будет показано ниже.

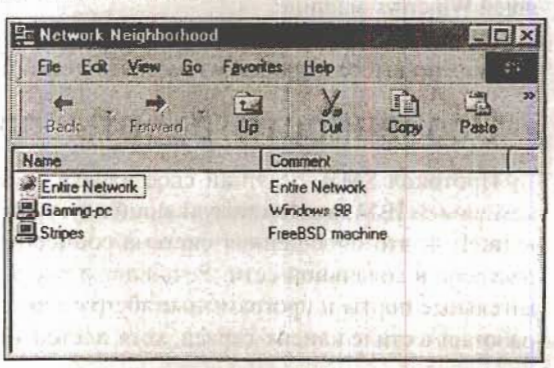


Рисунок 32.1. *Окно просмотра сети Windows, показывающее компьютер под управлением ОС FreeBSD, на котором работает система Samba*

Один из недостатков протокола NetBIOS — ориентация только на локальные сети LAN; для пакетов NetBIOS используется широковещательная рассылка, поэтому они не перенаправляются маршрутизаторами. Для связи зон совместного использования ресурсов Windows в различных сетях существует протокол WINS (Windows Internet Name Service — служба имен Internet для Windows), устраняющий отчасти эту проблему.

Защита, рабочие группы и домены

Доступ к общим ресурсам SMB контролируется на нескольких уровнях. Самый верхний уровень ограничивает доступ для просмотра содержимого любого из файл-серверов на базе IP-адреса или парольной аутентификации. Помимо этого для каждого отдельного общего ресурса (каталога, принтера или любого другого ресурса) также имеются свои права доступа и необязательные ограничения допустимых хостов/паролей. Наконец, в пределах общего ресурса к отдельным файлам ограничивается доступ (исходя из того, какой аутентифицированный пользователь или хост получил доступ к общему ресурсу).

Парольная аутентификация пользователей может осуществляться распределенным способом (каждым предоставляющим ресурс хостом) или централизованно (центральным сервером сетевой регистрации). В этом и состоит различие "рабочих групп" и "доменов" в Windows. *Рабочая группа* — это набор машин, согласованно появляющихся в окнах просмотра сети друг друга, но самостоятельно аутентифицирующих пользователей и обеспечивающих защиту. *Домен* — это группа машин, защита которых обеспечивается центральным сервером, на котором должны быть зарегистрированы все машины домена.

Система Samba позволяет ограничивать доступ на всех этих уровнях, и выполняет функции "главного браузера" (в контексте рабочей группы) или контроллера домена (централизованной службы регистрации в среде домена). Вскоре мы рассмотрим, как это делается.

Совместное использование файлов и принтеров с клиентами Macintosh с помощью AppleTalk

Система Samba обеспечивает совместный доступ к ресурсам на базе протоколов SMB/CIFS для взаимодействия с клиентами Windows. Но для работы с клиентами Macintosh придется использовать AppleTalk — другой набор протоколов для локальной сети, работающий поверх собственного транспортного уровня или поверх протокола IP. Поскольку во многих организациях и университетских сетях зоны совместного использования файлов и управления принтерами на базе протоколов AppleTalk достаточно велики, имеет смысл поддержать и этот набор протоколов в системе FreeBSD.

Пакет программ, обеспечивающих поддержку протоколов AppleTalk на UNIX-платформах, называется **netatalk**; он входит в набор портированных приложений. Это еще один успешный проект сообщества разработчиков программ с открытым исходным кодом, созданный в среде SourceForge и продолжающий постоянно совершенствоваться и развиваться.

Для установки поддержки протоколов AppleTalk соберите пакет **netatalk** из каталога портированных приложений **/usr/ports/net/netatalk**. Пакет **netatalk-asun** можно проигнорировать, поскольку сделанные в нем Эдрианом Саном (Adrian Sun) расширения уже включены в базовый пакет **netatalk**, в любом случае содержащий более новую версию. Не устанавливайте оба портированных пакета сразу, потому что система AppleTalk вообще не будет работать.

Для корректной работы пакета **netatalk** необходимо включить опцию **NETATALK** в конфигурации ядра. Подробнее о перестройке ядра см. в главе 17. Пакет **netatalk** включает несколько конфигурационных файлов — по одному для каждого используемого демона; в том числе — стандартные дистрибутивные конфигурации в файлах с расширением **.dist**. Все демоны будут работать и в стандартной конфигурации, хотя скорее всего придется изменить некоторые конфигурационные файлы для получения необходимых свойств. Каждый компьютер Macintosh в сети увидит вашу машину в зоне AppleShare в окне Chooser (или в окне Connect to Server в среде Mac OS X), как показано на рис. 32.2.

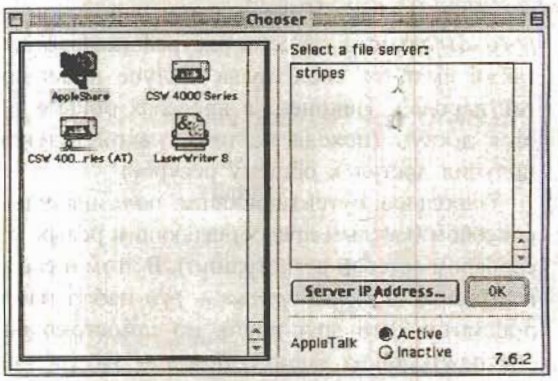


Рисунок 32.2.
*Окно Mac OS Chooser, показывающее файл-сервер на базе пакета **netatalk**, доступный в зоне AppleShare*

Подробные описания различных инструментальных средств пакета **netatalk** и ссылки на документацию (в большой степени Linux-ориентированную, но тем не менее полезную) можно найти на официальном Web-сайте **netatalk** по адресу **http://netatalk.sourceforge.net/**.

Установка и конфигурирование системы Samba

Система Samba доступна среди портированных приложений в каталоге **/usr/ports/net/samba** или в виде пакетов. Установка пакета или портированного приложения описана в главе 15.

После установки пакета Samba в системе появится много новых компонентов: выполняемые файлы демонов (в каталоге **/usr/local/sbin**), средства администрирования (в каталоге **/usr/local/bin**), документация и примеры (в каталоге **/usr/local/share**) и файлы конфигурации, устанавливаемые в каталоге **/usr/local/etc**. Некоторые из поддерживаемых файлов конфигурации не создаются при стандартной установке; их придется создавать с нуля, если необходимы предоставляемые ими функциональные возможности. Есть еще кодовые страницы (в каталоге **/usr/local/etc/codepages**), сопоставляющие наборы символов Windows и UNIX.

В состав пакета входит единственный конфигурационный файл — **smb.conf.default**, который для работы необходимо переименовать в **smb.conf**. Сценарий запуска **/usr/local/etc/rc.d/samba.sh.sample** также необходимо переименовать в **samba.sh**. При самом простом способе запуска системы Samba необходимо только отредактировать файл **smb.conf**, изменив строку рабочей группы в соответствии с именем рабочей группы или домена, в который должна входить машина:

```
# workgroup = Имя домена или рабочей группы NT, например, REDHAT4
  workgroup = MYGROUP
```

Теперь при перезагрузке система Samba будет запускаться автоматически. Для запуска системы вручную, выполните сценарий **samba.sh** с параметром **start**:

```
# /usr/local/etc/rc.d/samba.sh start
  Samba#
```

ПРИМЕЧАНИЕ

Обратите внимание на отсутствие перевода строки в результатах выполнения сценария после имени службы, **Samba**. Этот "косметический" недостаток существует потому, что в ходе начальной загрузки службы, упомянутые в каталоге **/usr/local/etc/rc.d**, запускаются последовательно и результаты сценариев намеренно выдаются с помощью команды **echo** в ту же строку. Возможность запускать службы вручную — просто удобное побочное свойство сценариев в каталоге **rc.d**, а не основное их назначение.

Демоны smbd и nmbd

Если сценарий **samba.sh** выполнен успешно, в системе можно обнаружить два новых процесса: **smbd** и **nmbd**:

```
# ps -waux | grep mbd
root  3855 0.0 1.5 2368 1816 ?? Is 2:43PM 0:00.00
   ↪/usr/local/sbin/smbd -D
root  3857 0.0 1.2 1940 1496 ?? Ss 2:43PM 0:00.02
   ↪/usr/local/sbin/nmbd -D
```

Демон **smbd** — это фактический сервер данных, процесс, обрабатывающий запросы SMB/CIFS от подключенных клиентов Windows, — запросы на передачу файлов, задания печати, листинги и т.д. В отличие от системы NFS, протокол SMB не требует запуска отдельного процесса для каждого из одновременных подключений; главный процесс **smbd** порождает новую собственную копию для каждого нового клиентского сеанса, которая и обрабатывает все запросы клиента в ходе сеанса. Опция **-D** указывает, что процесс **smbd** должен работать как отдельный демон, ожидающий запросы по TCP к порту 139.

Параллельно с демоном **smbd** работает **nmbd**, сервер имен NetBIOS. Именно этот процесс позволяет клиентам Windows видеть машину с ОС FreeBSD в окне просмотра сети, как было представлено на рис. 32.1. Он также отвечает на клиентские обращения к определенному хосту NetBIOS, если хост указан по имени. Если клиент Windows использует строку вида **<имя>** для подключения к определенному серверу по имени, то он посылает широковещательный запрос IP-адреса сервера с соответствующим именем NetBIOS. Задача посылки ответа с IP-адресом запрошенного хоста, чтобы клиент мог послать запрос SMB непосредственно ему, возлагается на демон **nmbd**. Он в чем-то аналогичен службе DNS (в том смысле, что сопоставляет общеизвестное имя прямому адресу), а также имеет много общего с реализацией протокола ARP (поскольку работает в локальной сети путем рассылки широковещательных запросов, а не обращается к централизованному серверу имен).

Файл smb.conf и система SWAT

Основной файл конфигурации системы Samba — **/usr/local/etc/smb.conf**, в котором можно устанавливать десятки различных опций и задавать общие ресурсы с нестандартными параметрами. В этом файле строки, начинающиеся символами # или ;, являются комментариями; в файле **smb.conf.default** (это пример файла конфигурации)

принято использовать символ # в качестве собственно комментариев, а символ ; — для отключения или включения строк конфигурации.

Каждая опция весьма неплохо описана в комментариях в файле **smb.conf.default**. Однако разобраться в содержимом этого файла непросто, поскольку опций для установки там очень много (они описаны на странице справочного руководства **man smb.conf**), причем между ними есть множество тонких различий. Имеется, однако, и альтернативный метод формирования файла **smb.conf**, если особенности работы сети позволяют его использовать. Речь идет о системе *SWAT, Samba Web Administration Tool* (средство администрирования системы Samba через Web), представленной на рис. 32.3.

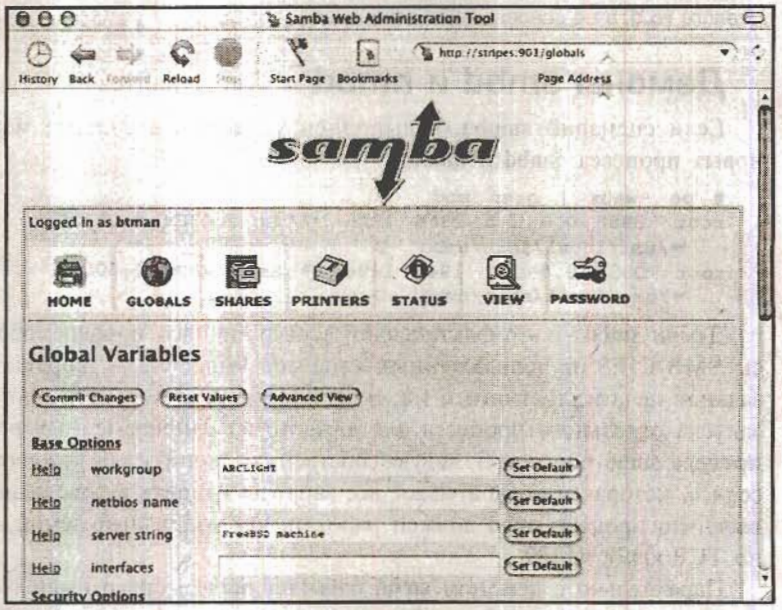


Рисунок 32.3. *SWAT, средство администрирования системы Samba через Web*

Система SWAT входит в состав портированного пакета Samba и позволяет конфигурировать Samba с помощью графического пользовательского интерфейса через Web-браузер. В результате существенно упрощается работа с файлом **smb.conf** и снижается вероятность ошибок в нем. Недостаток же этой системы, органически присущий всем Web приложениям, — существенная угроза защите. Система SWAT аутентифицирует пользователей с помощью базы данных пользователей системы, хранящейся в файле **/etc/master.passwd**. Пароли аутентификации посылаются по сети с клиентской машины на конфигурируемый сервер Samba в явном виде, и, если только вы не выполняете конфигурирование с локальной машины (**localhost**, с помощью X-Windows), это крайне рискованно с точки зрения защиты. Риск можно снизить несколькими способами, но ни один из них не обеспечивает полной безопасности. Используя систему SWAT, руководствуйтесь следующими принципами.

- Обращайтесь к системе SWAT только с локального хоста (**localhost**). Это предотвращает пересылку информации по сети.
- Работайте только под защитой брандмауэра, запрещающего передачу информации извне. По умолчанию файл **smb.conf** принадлежит пользователю **root**,

поэтому браузер должен регистрироваться в системе SWAT, передавая пароль пользователя **root**, который посылается по сети в явном виде (не шифруется) вместе с каждым HTTP-запросом к системе SWAT. Никогда не делайте этого в сети, потенциально открытой для враждебно настроенного любителя подслушивания.

- Создайте "фиктивного" пользователя (например, **smbowner**) и сделайте его владельцем файла **smb.conf** (с помощью команды **chown**). Работая с системой SWAT, регистрируйтесь в качестве этого пользователя, а не как пользователь **root**. Не используйте это имя пользователя для решения других задач на сервере и не давайте пользователю никаких привилегий, доступа к командному интерпретатору и не создавайте ему начальный каталог. Учтите, что если пароль этого пользователя будет перехвачен, то вторгшийся злоумышленник сможет изменить конфигурацию системы Samba. Но он не сможет сделать ничего другого.

Если описанные угрозы защите для вас приемлемы и вы можете использовать один из представленных выше методов доступа к системе SWAT, включите ее поддержку, добавив в соответствующем месте следующую строку в файл **/etc/services** (**901** — рекомендованный порт, но подойдет любой свободный порт TCP):

```
swat   901/tcp
```

Затем добавьте следующую строку в файл **/etc/inetd.conf**:

```
swat stream tcp nowait root /usr/local/sbin/swat swat
```

Наконец, перезапустите демон **inetd**:

```
# killall -HUP inetd
```

Теперь можно обращаться к системе SWAT по адресу URL **http://stripes.somewhere.com:901**, подставив в него имя хоста сервера Samba или просто **localhost**. Будет выдан запрос имени пользователя и пароля. Укажите выбранное для доступа к SWAT имя пользователя (**root**, если организация защиты это позволяет или имя специально созданного для этой цели пользователя).

Система SWAT позволяет обращаться и изменять предоставляемые для общего доступа ресурсы и принтеры, а также глобальные настройки; можно также узнать текущее состояние сервера и управлять пользователями системы. Если система SWAT обнаруживает, что зарегистрированный пользователь имеет полный доступ к файлу **smb.conf**, в интерфейсе будут доступны все семь кнопок действий, представленных на рис. 32.3. В противном случае, доступны будут только четыре из них, позволяющие получить информацию о состоянии сервера Samba, но изменить конфигурацию вы не сможете.

Поскольку система SWAT работает как пользовательский интерфейс к файлу **smb.conf** и в качестве средства администрирования особой ценности может и не представлять, в оставшейся части главы мы сосредоточимся на непосредственном конфигурировании сервера Samba путем редактирования файла **smb.conf**, а не на соответствующих средствах интерфейса SWAT.

Предоставление каталогов для общего доступа

Немало примеров конфигурирования каталога для общего доступа можно найти в файле **smb.conf.default**. Чтобы задействовать их, выполните соответствующие изменения в файле **smb.conf**, а затем остановите и снова запустите сервер Samba:

```
# /usr/local/etc/rc.d/samba.sh stop
# /usr/local/etc/rc.d/samba.sh start
```

Примеры общих ресурсов представлены в файле **smb.conf** после строки ===== **Share Definitions** =====. Имя каждого общего ресурса указывается в квадратных скобках, а следующие за ним строки, вплоть до нового имени в квадратных скобках, задают конфигурацию ресурса. Файл **smb.conf** начинается с блока **[global]**, позволяющего задать глобальные параметры. Каждый из остальных блоков определяет общий ресурс, настройки которого имеют приоритет над ранее определенными глобальными параметрами примерно так же, как и в файле конфигурации сервера Apache, **httpd.conf** (рассматривался в главе 26).

СОВЕТ

Страница справочного руководства **man smb.conf** содержит подробное описание возможных параметров конфигурации. Однако может оказаться более удобным просматривать эту страницу в формате HTML, с выделенными заголовками и примерами, что облегчает чтение, а также с контекстными гиперссылками. Версия страницы в формате HTNL доступна в сети по адресу **http://samba.org/samba/docs/man/ smb.conf.5.html**.

Чтобы предоставить для общего доступа обычный общедоступный каталог, определите блок общего ресурса следующим образом:

```
[my-public]
   comment = Общедоступные файлы
   path = /usr/local/share/samba-stuff
   public = yes
   writeable = yes
   printable = no
   write list = @staff
```

При наличии такого общего ресурса клиент будет видеть ресурс **my-public** на верхнем уровне списка общих ресурсов сервера. Однако пока пользователь не будет аутентифицирован и не окажется членом UNIX-группы **staff**, файлы ресурса будут доступны только для чтения. Уберите строку **write list**, чтобы сделать ресурс доступным для записи всем пользователям. Учтите, что установка **writeable = yes** эквивалентна установке **read only = no**.

По умолчанию определяется и включается общий ресурс **[homes]**; это специальный встроенный ресурс, обеспечивающий доступ к начальному каталогу каждого пользователя на сервере Samba при подключении клиента Windows от имени соответствующего пользователя. (Защиту на уровне пользователя и на уровне общего ресурса мы рассмотрим немного позже.)

```
[homes]
   comment = Начальные каталоги
   browseable = no
   writeable = yes
```

Поскольку этот ресурс установлен как непросматриваемый (**browseable = no**), начальные каталоги, не принадлежащие пользователю клиента, не показываются. Если клиент подключается от имени действительного пользователя, имеющего начальный каталог на сервере Samba, этот начальный каталог (имя которого в ОС UNIX обычно совпадает с именем пользователя) появится как один из доступных общих ресурсов. Начальные каталоги других пользователей не будут видны.

Совместный доступ к принтерам

Как и **[homes]**, **[printers]** — специальный общий ресурс, немного отличающийся от обычных общих ресурсов. В ОС FreeBSD все подключенные принтеры, которые определены в файле **/etc/printcap**, доступны пользователям системы Samba. Настройка поддержки локальных принтеров в ОС FreeBSD в файле **/etc/printcap** описана в главе 16.

По умолчанию, общий ресурс **[printers]** настроен следующим образом:

```
[printers]
   comment = Все принтеры
   path = /var/spool/samba
   browseable = no
# Установите public = yes, чтобы позволить печатать пользователю 'guest'
   guest ok = no
   writeable = no
   printable = yes
```

Как указано во встроенном комментарии, принтеры можно сделать общедоступными, так что любой пользователь сети сможет их использовать. Для этого необходимо задать *гостевого пользователя*, что мы рассмотрим далее. Учтите, что установка **public** — это синоним **guest ok**, так что для обеспечения возможности печати для всех пользователей можно просто заменить значение в строке **guest ok** на **yes**, а не добавлять строку **public = yes**.

В системе Samba версии 2.2.0 и более поздних, поддерживаются вызовы удаленных процедур печати (printing RPCs) Windows 2000/NT, так что можно устанавливать необходимые драйверы печати клиентам, у которых их нет. Описание этой возможности см. на Web-сайте Samba.

Управление доступом

При управлении доступом в системе Samba используется ряд абстрактных понятий, которые многим весьма сложно понять. Имеется несколько схем аутентификации пользователей Windows: LAN Manager (LANMAN), Windows NT/2000, Windows 95/98/Me и Windows for Workgroups, причем каждая немного отличается с точки зрения использования шифрования, регистрационных имен и процесса установки соединения.

В системе Samba имеется два популярных способа управления доступом: на уровне пользователей и на уровне общих ресурсов. Стандартным является управление доступом на уровне пользователей, задаваемое следующей опцией **security**:

```
# Режим защиты. Большинство предпочитает защиту на уровне
# пользователей. Подробнее см. в файле security_level.txt.
   security = user
```

Управление доступом на уровне пользователей

При управлении доступом *на уровне пользователей* клиент предоставляет серверу при начальной установке соединения пару имя пользователя/пароль. Сервер определяет, принимать ли клиента, на основе пары имя пользователя/пароль и идентификации клиентской машины. Если клиент принимается, ему доступны все общие ресурсы.

Правильная настройка защиты на уровне пользователей может оказаться непростым делом. Имя пользователя Windows, определяемое в начале сеанса Windows путем локальной регистрации или регистрации на контроллере домена, должно быть известно на сервере Samba в качестве имени обычного пользователя UNIX (или быть

сопоставлено имени пользователя UNIX). Например, если Windows-пользователь Harris регистрируется на своей Windows-машине, открывает окно Network Neighborhood (Мое сетевое окружение) и пытается подключиться к серверу Samba, он не получит доступа (при этом ему будет выдано диалоговое окно с запросом пароля для общего ресурса по имени **\\STRIPES\IPC$**), если только на UNIX-машине нет пользователя **harris**.

ПРИМЕЧАНИЕ

В Windows NT/2000 диалоговое окно ввода пароля позволяет, помимо пароля, ввести и имя пользователя. Но в Windows 95/98/Me мы получаем только запрос пароля, а имя пользователя автоматически строится на основе регистрационного имени.

Пользователи Samba должны существовать в базе данных паролей в файле **/usr/local/private/smbpasswd**, который похож на файл **/etc/master.passwd** тем, что зашифрованные пароли хранятся в нем для каждого локального пользователя UNIX. При установке системы Samba информация о пользователях из файла **/etc/master.passwd** преобразуется в формат Samba и помещается в файл **/usr/local/private/smbpasswd**, так что пароли LANMAN и Windows NT (оба задаются для совместимости) устанавливаются равными строке из 16 символов **X**, что показывает невозможность регистрации пользователя.

Чтобы разрешить регистрацию пользователя, необходимо установить допустимое значение пароля. Это делается с помощью программы **smbpasswd**. Она работает аналогично программе **passwd**, запрашивая старый пароль Samba, а затем требуя ввести новый пароль дважды, если только вы не работаете от имени пользователя **root** (тогда вводить старый пароль не придется и, кроме своего, можно будет изменить пароль любого пользователя).

```
# smbpasswd harris
New SMB password:
Retype new SMB password:
Password changed for user harris.
```

По умолчанию первоначальные версии Windows 95 и NT не использовали зашифрованные пароли. Они передавали пароли по сети в явном виде, как в ОС UNIX. Обновленные версии Windows 95 и Windows NT, начиная с service pack 3, используют по умолчанию зашифрованную передачу паролей, что можно изменить только путем редактирования системного реестра. Чтобы правильно интегрироваться с последними версиями Windows NT/2000, а также с Windows 98/Me, необходимо включить шифрование паролей в системе Samba. Это делается путем включения параметра шифрования паролей. (Прочтите документ **/usr/local/share/doc/samba/textdocs/ENCRYPTION.txt**, детально описывающий механизм шифрования паролей. См. также файлы **Win95.txt** и **WinNT.txt** в том же каталоге.)

```
# Вам может понадобиться включить шифрование паролей. Прочтите,
# пожалуйста, файлы ENCRYPTION.txt, Win95.txt и WinNT.txt в
# документации системы Samba. Не включайте эту опцию, не прочитав
# указанные документы.
   encrypt passwords = yes
```

Как описано в этих документах, зашифрованные пароли, хотя и предполагают усиление защиты, на самом деле ухудшают защиту на стороне сервера. Зашифрованные пароли Samba, хранящиеся в файле **/usr/local/private/smbpasswd**, не отличаются от паролей, пересылаемых по сети, и по защищенности эквивалентны паролям, хра-

нящимся в явном виде. Вторгшийся в систему злоумышленник, получивший доступ к файлу **smbpasswd**, может немедленно зарегистрироваться от имени любого пользователя (включая пользователя **root**) через систему Samba, тогда как зашифрованные пароли в файле **/etc/master.passwd** еще надо взломать одним из методов подбора. Файл **smbpasswd** надо беречь не меньше, чем **master.passwd**, если не больше!

СОВЕТ

Можно сопоставлять нескольких пользователей Windows с одним пользователем UNIX, что позволяет давать группам пользователей Windows те же права, что и у конкретного пользователя UNIX. Для этого необходимо создать файл, содержащий такие сопоставления, например **/usr/local/etc/smbusers.map**. Каждое сопоставление в этом файле задается отдельной строкой следующего формата:

```
<пользователь unix> = <пользователь win-1> [<пользователь win-2> ...]
```

Затем подключите этот файл, добавив в любом месте раздела **[global]** файла **smb.conf** строку:

```
username map = /usr/local/etc/smbusers.map
```

Каждый получивший такой псевдоним пользователь Windows должен регистрироваться с паролем того пользователя UNIX, с которым он сопоставлен.

Управление доступом на уровне общих ресурсов

При управлении доступом *на уровне общих ресурсов* клиент может подключаться к серверу Samba без всякой аутентификации по имени/паролю, и получит список всех общих ресурсов. Клиенту может быть отказано в доступе, только если его IP-адрес не указан в файле **smb.conf** (в строке **hosts allow**). Однако для каждого общего ресурса выполняется отдельная аутентификация пользователя с использованием системы паролей, рассмотренной при описании управления доступом на уровне пользователей. Общий ресурс, доступ к которому открыт для всех (**public = yes**), может использоваться беспрепятственно, но ресурс, предназначенный конкретному пользователю (например, начальный каталог из блока **[homes]**) защищен той же парой имя пользователя/пароль, что и при использовании описанного выше управления защитой на уровне пользователей.

Подробнее об организации защиты на уровне пользователей и общих ресурсов см. в файле документации **/usr/local/share/doc/samba/textdocs/security_level.txt**.

Гостевой пользователь

К некоторым службам Samba, в частности к службе печати, имеет смысл предоставлять доступ любому пользователю сети, независимо от аутентификации. Для этого используется *гостевая учетная запись* непривилегированного пользователя, имеющего доступ как пользователь UNIX только к предоставляемой для общего доступа службе. Учтите, что гостевые пользователи предназначены в основном для серверов Samba, работающих с защитой на уровне ресурсов, поскольку доступ гостевому пользователю предоставляется или запрещается для каждого ресурса отдельно.

Чтобы разрешить работу гостевому пользователю, раскомментируйте строку **guest account** в файле **smb.conf**:

```
# Раскомментируйте эту строку, если хотите использовать учетную
# запись guest. Соответствующего пользователя надо добавить в файл
# /etc/passwd, иначе используется пользователь "nobody".
  guest account = pcguest
```

Теперь необходимо добавить в систему учетную запись **pcguest** (или с любым другим выбранным для этой цели регистрационным именем) с помощью команды **adduser**.

Настройте предоставляемые пользователю ресурсы с помощью команды **chfn**. Создание пользователя **ftp** (созданный программой **sysinstall**, если система поддерживает анонимный доступ к FTP-серверу) может служить образцом при создании учетной записи гостевого пользователя системы Samba. Подробнее о настройке FTP-сервера см. в главе 27.

После создания этой учетной записи любой пользователь Windows, подключающийся к серверу Samba, будет иметь полный доступ к любому общему ресурсу, у которого параметр **guest ok** или **public** имеет значение **yes**. Запрос аутентификации для такого ресурса выдаваться не будет.

Можно указать параметр **guest only = yes**, показывающий, что к службе разрешены только гостевые подключения.

Журнальные файлы системы Samba

Система Samba создает различные журнальные файлы в каталоге **/var/log** — по одному для каждого типа службы и каждого подключенного клиента. Имена этих файлов имеют вид **log.<служба>**:

```
# ls -l /var/log/log.*
-rw-r--r--  1  root wheel  468   Jun  9  12:42  /var/log/log.gaming-pc
-rw-r--r--  1  root wheel  2343  Jun  9  14:49  /var/log/log.nmb
-rw-r--r--  1  root wheel  1606  Jun  9  14:44  /var/log/log.smb
```

Файлы **log.nmb** и **log.smb** содержат информацию о состоянии и ошибках серверов **nmbd** и **smbd**, соответственно. Кроме того, при возникновении ошибки на любом из подключающихся клиентских хостов (например, ошибок аутентификации), эти сообщения выдаются в файл **log.<имя клиента>**. Учтите, что в результате может создаваться множество файлов, переполняющих каталог **/var/log**. Можно перейти на комбинированный формат журнала, закомментировав строку **log file** в файле **smb.conf**:

```
#  Эта  строка  требует  от  системы  Samba  использовать  отдельный
#  журнальный  файл  для  каждой  подключающейся  машины
;     log  file  =  /var/log/log.%m
```

Если эта строка закомментирована, ошибки протокола SMB для отдельных хостов будут регистрироваться в файле **log.smb**.

Часто имеет смысл изменить и строку **max log size**, стандартно устанавливающую размер журнала равным 50 Кбайт. Это позволяет задать другой максимальный размер журнальных файлов. Когда журнальный файл Samba достигает указанного размера, он переименовывается путем добавления к имени файла суффикса **.old**, и соответствующий журнальный файл создается заново при добавлении новой записи. Файл **.old** перезаписывается при следующем заполнении журнального файла.

```
#  Задает  размер  журнальных  файлов  (в  Кбайтах).
   max  log  size  =  50
```

Переменные системы Samba

Параметры конфигурации в файле **smb.conf** необязательно задавать буквально — имеется ряд переменных подстановки, так что некоторые опции могут определяться динамически, в зависимости от особенностей подключения. Например, можно использовать имя пользователя клиента в значении опции с помощью переменной **%u**. Это позволяет устанавливать параметры вида **path = /usr/local/share/user-files/%u**. В

этом случае пользователь **harris** получит **/usr/local/share/user-files/harris** в качестве значения параметра **path**.

Вот некоторые из наиболее часто используемых переменных:

- **%u** — имя пользователя клиента;
- **%g** — имя основной группы пользователя **%u**;
- **%S** — имя текущей службы, если имеется;
- **%H** — начальный каталог пользователя, задаваемого переменной **%u**;
- **%h** — Internet-имя хоста, на котором работает сервер Samba;
- **%M** — Internet-имя клиентской машины;
- **%L** — имя NetBIOS сервера Samba. Эта переменная позволяет изменять конфигурацию в зависимости от того, какой клиент обратился к серверу; у сервера, таким образом, может быть "раздвоение личности";
- **%m** — имя NetBIOS клиентской машины;
- **%I** — IP-адрес клиентской машины;
- **%T** — текущая дата и время;
- **%$(envvar)** — значение переменной среды **envvar**.

Полный список переменных подстановки можно найти на странице справочного руководства **man smb.conf**.

Другие компоненты системы Samba

Пакет Samba включает ряд дополнительных инструментальных средств. Все они перечислены на странице справочного руководства **man samba**; мы рассмотрим их здесь кратко. Подробное описание каждого представлено на отдельной странице справочного руководства.

- **smbclient**. Простой клиент в стиле FTP, позволяющий подключаться к удаленным общим ресурсам SMB и печатать на удаленных принтерах Windows.
- **testparm**. Программа проверки синтаксиса файла конфигурации, определяющая корректность файла **smb.conf**; аналог команды **apachectl configtest** сервера Apache.
- **testprns**. Проверяет корректность работы принтеров, перечисленных в файле **/etc/printcap**, с системой Samba.
- **smbstatus**. Выдает список текущих подключений к серверу Samba. Система SWAT, как было сказано, включает страницу, выдающую эту же информацию в сформатированном виде через Web-браузер.
- **nmblookup**. Позволяет выполнять запросы имен NetBIOS, аналогичные выполняемым хостами Windows, непосредственно подключающимися к общему ресурсу SMB, по имени.
- **make_smbcodepage**. Средство создания определений новых кодовых страниц SMB для системы Samba.

Эти программы дополняют рассмотренные выше демоны **smbd, nmbd** и утилиту **smbpasswd**. Для этих программ также имеются отдельные страницы справочного руководства.

Дальнейшее развитие системы Samba

На момент написания этой главы последней версией системы Samba в портированных приложениях была версия 2.0.9; система Samba 2.2.0, следующая принципиально новая версия, только что вышла, и, когда вы будете читать эту главу, именно она или более новая версия, вероятно, и будет официальной в каталоге **/usr/ports/net/samba**.

Версия 2.2.0 включает много дополнительных сетевых возможностей Windows, включая новые возможности в Windows 2000, и повышает уровень взаимодействия файл-серверов Windows и UNIX. Вот лишь некоторые из новых возможностей.

- Автоматическая загрузка драйверов принтеров Windows NT/2000 с сервера Samba, если они отсутствуют на клиентской машине.
- Унификация списков контроля доступа (Access Control Lists — ACLs) Windows NT/2000 и UNIX, и удаленное управление этими списками с Windows-машин.
- Встроенная аутентификация регистрации Windows NT/2000.
- Поддержка распределенной файловой системы Microsoft (Microsoft Distributed File System — DFS) и возможность работы в качестве сервера DFS для Windows-клиентов.

Подробнее об этих и других новых возможностях можно прочесть по адресу **http://www.samba.org**.

Файловая система smbfs

Совместное использование файлов по протоколу SMB может быть двусторонним. Система Samba позволяет настроить машину с ОС FreeBSD для работы только в качестве сервера протокола SMB, но есть способ настроить и клиентскую часть, чтобы можно было монтировать удаленные общие ресурсы SMB, как любую файловую систему. Речь идет о файловой системе **smbfs**, доступной среди портированных приложений в каталоге **/usr/ports/net/smbfs**.

Реализация файловой системы **smbfs** для ОС Linux существовала уже некоторое время; но реализация в ОС FreeBSD — новая и привязана к платформе, поскольку поддержку файловой системы необходимо интегрировать в ядро, а ядра — наименее совместимые части операционных систем. Реализация **smbfs** для ОС FreeBSD включает модуль ядра **smbfs.ko** в каталоге **/modules** и программу **mount_smbfs** в каталоге **/sbin**, работающую аналогично другим программам **mount_***, рассмотренным в главе 9. Наиболее подробная документация по файловой системе **smbfs** представлена на странице справочного руководства **man mount_smbfs**.

После установки необходимо переименовать файл **/usr/local/etc/nsmb.conf.sample** в **nsmb.conf**, а файл **/usr/local/etc/rc.d/smbfs.sh.sample** — в **smbfs.sh**. Первый файл устанавливает стандартные значения для определенных хостов SMB и глобальные свойства всех монтируемых файловых систем **smbfs**, а второй — это сценарий монтирования общих ресурсов **smbfs**, указанных в файле **/etc/fstab**, в ходе начальной загрузки.

Чтобы смонтировать файловую систему SMB с помощью **smbfs**, используйте команду **mount_smbfs** с рядом простых опций. Опция **-I** задает имя хоста или IP-адрес, а два оставшихся аргумента — имя удаленного общего ресурса (в формате **//<пользователь>@<имя NetBIOS>/<имя ресурса >**) и локальная точка монтирования. Так,

для монтирования общего ресурса **public** с Windows-машины **gaming-pc** в локальный каталог **/smb/public** используется следующая команда:

```
# mount_smbfs -I 64.41.131.139 //guest@gaming-pc/public /smb/public
```

Будет выдан запрос пароля. Используйте пустой пароль, если ресурс доступен для доступа всем; укажите соответствующий пароль, если ресурс доступен только для чтения или защищен паролем.

ПРИМЕЧАНИЕ

Модуль ядра **smbfs.ko** при необходимости загружается автоматически программой **mount_smbfs**. Если хотите, можете загружать его при загрузке системы, добавив следующую строку в файл **/boot/loader.conf**:

```
smbfs_load="YES"
```

Однако скорее всего это не понадобится.

Чтобы добавить общий ресурс SMB в файл **/etc/fstab**, используйте строку следующего вида:

```
//guest@gaming-pc/public /smb/public smbfs rw,noauto 0 0
```

Сценарий **/usr/local/etc/rc.d/smbfs.sh** будет монтировать этот общий ресурс при загрузке системы FreeBSD.

ГЛАВА 33

Протокол DHCP

- Как работает протокол DHCP
- Преимущества использования DHCP
- Конфигурирование ядра для поддержки протокола DHCP
- Включение поддержки DHCP
- Программа dhclient
- Демон сервера DHCP
- Файл конфигурации демона dhcpd

Протокол DHCP (сокращение от Dynamic Host Configuration Protocol, *протокол динамического конфигурирования хоста*) — это сетевой протокол, позволяющий клиентам получать свои IP-адреса, информацию о сервере имен, шлюзе, и ряд других параметров сетевой конфигурации с сервера, на котором работает служба DHCP.

Как работает протокол DHCP

Подробное описание работы протокола DHCP выходит за рамки книги, но мы кратко рассмотрим наиболее важные моменты.

В ОС FreeBSD имеется клиентская программа **dhclient**, позволяющая компьютеру с ОС FreeBSD работать в качестве клиента DHCP. При загрузке системы, сконфигурированной для использования протокола DHCP, запускается программа **dhclient**, посылающая широковещательные запросы к порту 68. Эти запросы посылаются по протоколу UDP.

Если в сети имеется сервер DHCP, он будет прослушивать порт 68 в ожидании таких запросов. Получив запрос, касающийся конфигурационной информации, он проверяет наличие в своей базе данных свободного IP-адреса, который можно выделить клиенту. Затем он посылает в ответ всю запрошенную клиентом конфигурационную информацию на порт 67, тоже по протоколу UDP. Выделенный клиенту IP-адрес удаляется из пула доступных адресов, так что он не будет выделен другому клиенту, запрашивающему конфигурационную информацию по протоколу DHCP.

Клиент, поддерживающий протокол DHCP, будет прослушивать порт 67 в ожидании конфигурационной информации от сервера. Получив такую информацию, он конфигурируется в соответствии с ней.

Аренда IP-адреса

Назначенный сервером DHCP IP-адрес клиент не получает навсегда. Адрес просто арендуется. Срок аренды сконфигурирован на сервере DHCP. Информация о том, сколько будет действительна аренда, посылается клиенту вместе с конфигурационной информацией.

Аренда IP-адресов служит двум целям:

- Если не удается связаться с сервером DHCP, клиент просматривает локальную базу данных в поисках действительной аренды. Если таковая найдена, клиент может нормально работать, даже если сервер DHCP в настоящее время отключен.
- Неиспользуемые IP-адреса автоматически возвращаются в пул свободных по истечении срока аренды. Это помогает экономить IP-адреса. Например, сотрудник, пришедший из дочернего офиса, может подключить свой переносной компьютер в сеть организации и получить IP-адрес, т.е. сможет использовать сеть. Со временем этот IP-адрес устареет и будет возвращен в пул свободных для использования другими машинами. Таким образом, IP-адреса не тратятся напрасно на уже не существующие системы.

Преимущества использования DHCP

В некоторых ситуациях использование протокола DHCP дает ряд преимуществ по сравнению с простым присвоением каждой системе собственного статического IP-адреса. Вот некоторые из этих преимуществ.

- **Простота сопровождения.** Сервер DHCP автоматически следит за используемыми и свободными IP-адресами. Это снимает с системного администратора задачу поддержки списка IP-адресов, которые могут выделяться новым клиентам, а также пополнения этого списка при удалении клиента из сети навсегда. Все это выполняется автоматически сервером DHCP.
- **Простота установки новых клиентов.** При установке новых клиентов администратору (или пользователю) не надо беспокоиться о настройке конфигурации сети. Можно просто попросить нового клиента использовать для конфигурирования протокол DHCP, и все необходимые параметры конфигурации сети будут установлены автоматически.
- **Простота использования для мобильных пользователей.** Если пользователи часто бывают в дочерних офисах, протокол DHCP существенно упрощает им жизнь. При использовании протокола DHCP они могут подключить свой переносной компьютер к сети в дочернем офисе, и вся информация о сети будет сконфигурирована автоматически. В следующем дочернем офисе, куда они приедут, можно будет сделать то же самое. Таким образом, пользователям не придется переконфигурировать установки сети в каждом офисе, где им придется работать. Это также упрощает жизнь администраторам сетей, поскольку им не придется беспокоиться о выделении свободных IP-адресов для таких приходящих пользователей.
- **Сохранение IP-адресов.** Это существенно, особенно при наличии мобильных пользователей, периодически посещающих дочерние офисы и подключающие свои переносные компьютеры к сетям. Использование DHCP позволяет автоматически освобождать соответствующие IP-адреса после отъезда мобильных пользователей. Таким образом, не приходится напрасно выделять постоянные IP-адреса системам, лишь изредка подключаемым к сети.
- **Устранение проблем, вызываемых конфликтами IP-адресов.** Достаточно пользователю сделать одну опечатку при настройке системы и возникнут разнообразные проблемы в сети, если введенный IP-адрес конфликтует с другой системой, особенно если он совпадет с IP-адресом, выделенным серверу. Использование протокола DHCP устраняет эти проблемы, поскольку IP-адреса выделяются автоматически, а используемые адреса отслеживаются, так что один адрес не может быть выделен нескольким системам.

Конечно, в небольшой сети из 10 или 15 систем при отсутствии или нечастом появлении посетителей, которым необходимо включиться в сеть, все эти проблемы не так уж существенны, и поддержку протокола DHCP, вероятно, устанавливать не стоит. Однако использование протокола DHCP, определенно, стоит рассмотреть при наличии нескольких сотен или тысяч клиентов в сети, или если предполагается, что небольшая сеть разрастется до таких размеров со временем. Если предполагается подобный рост, имеет смысл начать конфигурирование протокола DHCP сразу, даже если в сети пока только 10 или 15 клиентов. Гораздо проще конфигурировать DHCP при наличии небольшого количества клиентов, чем переводить на использование DHCP существующую сеть из нескольких сотен систем.

Конфигурирование ядра для поддержки протокола DHCP

Чтобы можно было сконфигурировать ОС FreeBSD для работы в качестве клиента в сети с поддержкой протокола DHCP, необходимо наличие в ядре устройства Berkeley Packet Filter (фильтр пакетов Беркли). Оно устанавливается по умолчанию в стандартном ядре **GENERIC**, так что если не строилось специализированное ядро, из которого это устройство удалено, ничего делать не придется.

Для того чтобы проверить наличие поддержки устройства Berkeley Packet Filter в ядре, поищите следующую строку в файле конфигурации ядра:

```
device  bpf
```

При наличии такой строки в файле конфигурации ядра, ничего в нем изменять не надо. Если же этой строки нет, необходимо добавить ее и перестроить ядро. Подробные инструкции по перестройке ядра представлены в главе 17.

ПРЕДУПРЕЖДЕНИЕ

Использование Berkeley Packet Filter несет небольшую угрозу защите. Устройство **bpf** обеспечивает возможность работы анализаторов пакетов. Анализаторы пакетов могут использоваться во враждебных целях, поскольку способны перехватывать пароли и другую секретную информацию, пересылаемую по сети. Хотя анализаторы пакетов может запускать только пользователь **root**, эту возможность надо учитывать при работе в среде, где защита имеет принципиальное значение. Но даже с учетом этого, в очень большой сети преимущества использования протокола DHCP, вероятно, перевесят очень небольшой риск, которому подвергается защита при использовании устройства **bpf**.

Включение поддержки DHCP

Убедившись, что поддержка протокола DHCP сконфигурирована в ядре, ее можно включить одним из двух способов. Первый способ — с помощью программы **sysinstall**. Второй — путем редактирования конфигурационных файлов вручную. Начнем мы с описания использования программы **sysinstall**.

Включение DHCP с помощью программы sysinstall

При первоначальной установке ОС FreeBSD просто ответьте **yes** на задаваемый после установки запрос, следует ли использовать протокол DHCP для конфигурирования сети. Если ОС FreeBSD уже установлена, выполните следующие шаги для включения поддержки DHCP с помощью программы **sysinstall**.

1. От имени пользователя **root** введите **sysinstall** в командной строке для запуска программы **sysinstall**.
2. В главном меню выберите опцию Configure. Учтите, что в программе **sysinstall** мышь использовать нельзя. Для выбора необходимо использовать клавиши со стрелками и *клавишу Enter*.
3. В меню Configuration выберите опцию Networking. В результате будет выдано меню Network Services Menu (см. рис. 33.1).
4. Выберите опцию Interfaces с помощью клавиш со стрелками, а затем нажмите клавишу пробел.

5. Выберите сетевой интерфейс, который необходимо сконфигурировать, из следующего меню (см. рис. 33.2).

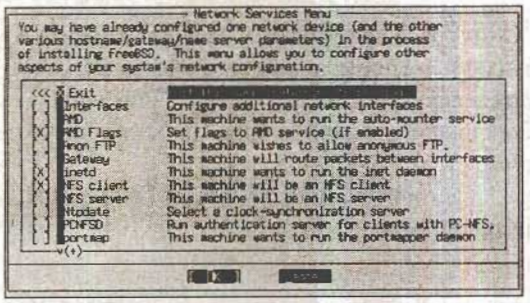


Рисунок 33.1. *Меню Network Services Menu в программе sysinstall*

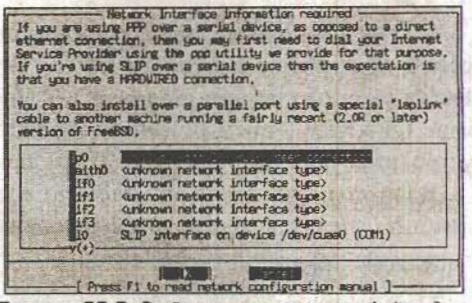


Рисунок 33.2. *Выбор сетевого интерфейса для конфигурирования. Многие из перечисленных устройств — псевдоустройства, а не реальные сетевые устройства*

6. Ответьте No на вопрос о необходимости сконфигурировать для интерфейса поддержку протокола IPv6.
7. Ответьте Yes на вопрос о необходимости сконфигурировать интерфейс с помощью протокола DHCP.

При утвердительном ответе на вопрос о необходимости конфигурирования интерфейса с помощью DHCP будет запущена программа **dhclient**, которая пошлет по сети широковещательный запрос о конфигурационной информации. Если она успешно получает конфигурационную информацию от сервера DHCP, то автоматически заполнит значения полей на следующем экране, где необходимо ввести значения параметров сетевой конфигурации. Затем можно выбрать кнопку OK, чтобы принять значения, полученные по протоколу DHCP.

Если это не первоначальная установка (вы вносите изменения в уже существующую систему), необходимо выйти из программы **sysinstall** и перезагрузить систему, прежде чем новые установки DHCP будут учтены.

Конфигурирование поддержки протокола DHCP вручную

Поддержку протокола DHCP можно сконфигурировать и вручную, редактируя файл **/etc/rc.conf**. Для этого необходимо добавить в этот файл строку **ifconfig** для устройства, которое будет конфигурироваться с помощью DHCP. Например, если используется устройство Ethernet **ed0**, в файл **/etc/rc.conf** необходимо добавить следующую строку:

```
ifconfig_ed0="DHCP"
```

Если необходимо сконфигурировать с помощью DHCP сетевой интерфейс, отличный от **ed0**, замените в данной строке имя устройства именем конфигурируемого интерфейса.

Если в файле **/etc/rc.conf** уже существует соответствующая строка для конфигурируемого устройства, и она содержит другое значение, то устройство, скорее всего, настроено на использование статического IP-адреса. В этом случае удалите существующую строку и замените ее указанной выше или закомментируйте существующую строку, поместив перед ней символ #, а затем добавьте указанную выше строку.

После внесения этого изменения в файл **/etc/rc.conf** перезагрузите систему, чтобы новые установки DHCP были учтены.

Программа dhclient

Программа **dhclient** — это клиентская часть протокола DHCP. Она автоматически запускается при каждой загрузке системы, если есть хоть один сетевой интерфейс, конфигурируемый с помощью DHCP.

Стандартное поведение программы **dhclient** при необходимости можно изменить с помощью флагов, задаваемых специальной опцией в файле **/etc/rc.conf**.

Если необходимо изменить стандартное поведение программы **dhclient**, добавьте следующую строку в файл **/etc/rc.conf** сразу после строки **ifconfig**, задающей использование протокола DHCP для конфигурирования интерфейса:

```
dhcp_flags="флаги"
```

где **флаги** — это список опций для программы **dhclient**. В табл. 33.1 представлены допустимые опции.

Таблица 33.1. Опции команды dhclient

Опция	*Действие*
-d	Программа **dhclient** остается в приоритетном режиме, а не переходит в фоновый после получения конфигурационной информации. Эта опция должна использоваться только для отладки и обычно не задается в файле **/etc/rc.conf**.
-cf ***имя_файла***	По умолчанию программа **dhclient** читает специфическую для операционной системы конфигурационную информацию из файла **/sbin/dhclient-script**. Опция **-cf** позволяет задать другой файл конфигурации для программы **dhclient**. Имя файла должно быть полным, включая путь.
-lf ***имя_файла***	По умолчанию программа **dhclient** хранит информацию об арендованных адресах в файле **/var/db/dhclient.leases**. Опция **-lf** позволяет указать программе **dhclient** другой файл для этих целей. Имя файла должно быть полным, включая путь.
-pf ***имя_файла***	По умолчанию программа **dhclient** хранит свой идентификатор процесса (PID) в файле **/var/run/dhclient.pid**. Опция **-pf** позволяет изменить этот файл, указав полное имя файла для хранения соответствующей информации.
-q	Программа **dhclient** работает в немногословном режиме. В результате выдается меньше сообщений о ходе работы.
-1	Программа **dhclient** только один раз пытается получить в аренду IP-адрес. Если это не удается, программа **dhclient** завершается с кодом возврата 2.

Сценарий /sbin/dhclient-script

Это файл конфигурации программы **dhclient**, специфический для используемой операционной системы. В этом файле не стоит делать никаких изменений; лучше его вообще не трогать, если вы не уверены в правильности своих действий.

Файл /etc/dhclient.conf

Это файл конфигурации программы **dhclient**, в котором можно задавать различные опции, определяющие ее поведение. Файл этот необходим для работы программы **dhclient**, хотя по умолчанию содержит только комментарии. Программа **dhclient** имеет вполне разумные стандартные значения опций, подходящие для большинства пользователей, поэтому обычно в этом файле ничего не изменяют.

Однако не мешает знать некоторые из опций, задаваемых в этом файле. В табл. 33.2 представлены основные опции.

Таблица 33.2. Некоторые опции файла dhclient.conf

Опция	*Действие*
timeout *n*	С ее помощью можно задать количество секунд, в течение которых программа **dhclient** должна ожидать ответа при попытке связаться с сервером DHCP, прежде чем признать сервер недоступным. Стандартное значение — 60 секунд.
retry *n*	Позволяет задать количество секунд, по прошествии которых программа **dhclient** повторит попытку связаться с сервером DHCP, если первый запрос не удался. По умолчанию программа ожидает пять минут.
select-timeout *n*	В некоторых сетях может быть более одного сервера DHCP. В этом случае клиент может получить несколько вариантов конфигурационной информации. Первый полученный вариант — не всегда лучший (например, второй вариант может содержать IP-адрес, совпадающий с текущим адресом клиента, и будет предпочтительнее варианта, предлагающего другой IP-адрес). Параметр *n* здесь задает количество секунд, в течение которых программа **dhclient** должна ожидать предложений от других серверов после получения первого ответа.
reboot *n*	При запуске программа **dhclient** пытается получить тот же IP-адрес, что использовался клиентом в прошлый раз. Если такой адрес получить не удастся, она примет другой IP-адрес. Параметр *n* задает, сколько секунд программа **dhclient** будет ожидать, прежде чем принять решение, что получить тот же IP-адрес, что и в прошлый раз, не удалось. Стандартное значение — 10 секунд.
request *опция*	Клиент будет запрашивать информацию об указанной опции у DHCP-сервера. Информацию об имеющихся опциях можно найти на странице справочного руководства **dhcp-options**.
require *опция*	Клиент будет запрашивать информацию об указанной опции у DHCP-сервера. Если требуемая информация не предоставлена, клиент отклоняет предлагаемый вариант. Информацию об имеющихся опциях можно найти на странице справочного руководства **dhcp-options**.
default *опция значение*	Если сервер DHCP не предоставляет информацию об указанной опции, для нее будет использоваться данное *значение*. Информацию об имеющихся опциях можно найти на странице справочного руководства **dhcp-options**.

Опция	*Действие*
supersede ***опция значение***	Для указанной опции будет всегда использоваться такое ***значение***, даже если сервер DHCP пришлет другое. Информацию об имеющихся опциях можно найти на странице справочного руководства **dhcp-options**.
reject ***адрес***	Любые варианты, присланные с сервера DHCP с указанным IP-адресом, будут отклонены.

Опций, которые можно использовать в файле конфигурации **/etc/dhclient.conf**, намного больше. Дополнительную информацию о поддерживаемых опциях можно найти на странице справочного руководства **dhclient.conf**. Более подробную информацию об опциях протокола DHCP, которые можно запросить или потребовать от сервера DHCP, ищите на странице справочного руководства **dhcp-options**.

Как уже было сказано, большинству пользователей не придется изменять файл **/etc/dhclient.conf**, поскольку стандартные значения опций программы **dhclient** обычно вполне подходят.

Демон сервера DHCP

В составе ОС FreeBSD не поставляется программное обеспечение сервера DHCP. Однако в наборе портированных приложений FreeBSD имеется свободно распространяемая реализация сервера DHCP. Эта реализация — пакет **isc-dhcp3** — предлагается в подкаталоге **net** дерева портированных приложений. Подробнее об установке портированных приложений см. в главе 15.

Кроме того, есть также программа **dhcpconf**, тоже входящая в набор портированных приложений FreeBSD в подкаталоге **net** и помогающая создать конфигурационные файлы, необходимые для запуска сервера DHCP в ОС FreeBSD.

При установке пакета **isc-dhcp3** будет создан файл запуска в каталоге **/usr/local/etc/rc.d** с именем **isc-dhcpd.sh.sample**. При следующей перезагрузке системы этот файл автоматически запустит сервер DHCP. Стандартный файл запуска подходит для простой конфигурации сервера DHCP. Описание различных опций демона **dhcpd** можно найти на странице справочного руководства **dhcpd**.

Имеет смысл переименовать файл запуска в каталоге **/usr/local/etc/rc.d**, дав ему имя, которое проще запомнить и набирать. Например, можно назвать файл **dhcpd**. Если файл переименован именно так, работой сервера DHCP можно управлять с помощью следующих команд, выполняемых от имени пользователя **root**.

Следующая команда запускает сервер DHCP:

```
/usr/local/etc/rc.d/dhcpd start
```

Следующая команда остановит работающий сервер DHCP:

```
/usr/local/etc/rc.d/dhcpd stop
```

Эта команда остановит, а затем снова запустит работающий сервер DHCP:

```
/usr/local/etc/rc.d/dhcp restart
```

Перед первым запуском сервера DHCP необходимо создать для него файл конфигурации.

Файл конфигурации dhcpd

Файл конфигурации для сервера DHCP, **dhcpd**, находится в каталоге **/usr/local/etc**. При установке демона **dhcpd** создается пример файла конфигурации в этом каталоге — файл **dhcpd.conf.sample**.

Имеется два способа конфигурирования файла **dhcpd.conf**: вручную и с помощью программы **dhcpconf**. Если вы никогда ранее не конфигурировали демон **dhcpd**, я рекомендую установить и использовать программу **dhcpconf**. Это даст вам возможность изучить файл **dhcp.conf**, созданный программой **dhcpconf**, и понять, как он устроен.

Программа dhcpconf

Программа **dhcpconf** предлагает меню и диалоговые окна, помогающие настроить базовую конфигурацию сервера DHCP. После установки ее можно запустить из командной строки, введя команду **dhcpconf** от имени пользователя **root**. На экран будет выдано диалоговое окно About, содержащее краткую информацию о программе. Нажмите клавишу Enter для перехода к главному экрану. На рис. 33.3 представлен главный экран программы **dhcpconf**.

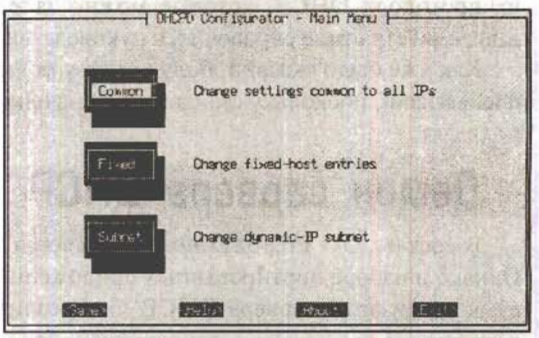


Рисунок 33.3. *Главное меню программы* **dhcpconf**

Учтите, что в этой программе нельзя использовать мышь. Для перемещения между полями используйте клавиши со стрелками, клавишу Enter и клавишу Tab.

Первая опция меню (Common) задает параметры, применяющиеся для всех выделенных IP-адресов. Это стандартная конфигурационная информация, которая будет посылаться клиентам. Нажмите клавишу Enter, чтобы задать ее. На рис. 33.4 представлено диалоговое окно Common.

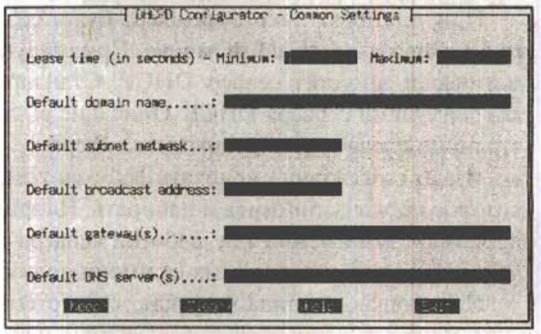


Рисунок 33.4. *Опции, общие для всех IP-адресов, и стандартные опции*

Большинство представленных в окне опций самоочевидны, кроме первой, задающей минимальный и максимальный период аренды.

Минимальный период аренды одновременно является и стандартным. Он определяет, сколько секунд арендованный адрес можно использовать. Максимальный период — это максимальное время передачи сервером IP-адреса в аренду. Клиент может запросить более длительную аренду, чем стандартное минимальное значение, но сервер никогда не предоставит в аренду IP-адрес на период, больший определенного здесь максимального.

Стандартный минимальный период аренды составляет 600 секунд (10 минут), а стандартный максимальный период аренды — 7200 секунд (120 минут). Клиент дол-

жен обновить аренду, прежде чем закончится максимальный период, иначе аренда будет потеряна, и его IP-адрес помещен в пул доступных адресов.

Введя необходимые значения, с помощью клавиши табуляции, перейдите на кнопку <Keep> и нажмите клавишу Enter. На экране снова появится главное окно.

Записи фиксированных хостов

Иногда требуется зарезервировать для некоторых систем статические IP-адреса (IP-адреса, которые никогда не будут изменяться). Обычно это делается для серверов. Это позволяет сделать опция DHCP Configurator. На рис. 33.5 показано диалоговое окно конфигурирования записей фиксированных хостов (Fixed-host Entries).

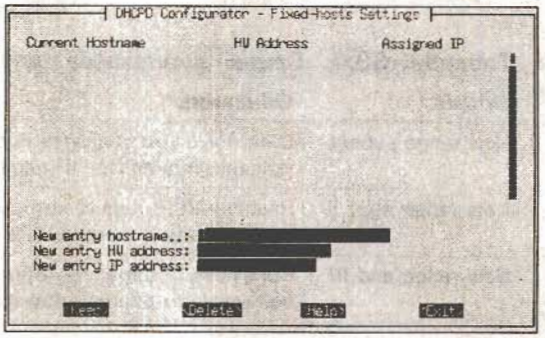


Рисунок 33.5. *Конфигурирование информации для систем, требующих статических IP-адресов*

В табл. 33.3 описано назначение опций этого диалогового окна.

Таблица 33.3. Опции конфигурации фиксированных хостов

Опция	*Описание*
New entry hostname	Это имя хоста системы, конфигурируемой для использования статического IP-адреса. Указывать надо только имя хоста, без имени домена, поскольку информация о домене уже была указана в общих (Common) опциях.
New entry HW address	Здесь необходимо ввести аппаратный адрес сетевой карты хоста, для которого конфигурируется статический IP-адрес.
New entry IP address	Это статический IP-адрес, который будет использовать конфигурируемый хост.

По завершении ввода соответствующей информации нажмите клавишу Enter, чтобы добавить хост в список. При необходимости можно ввести информацию и о других хостах, или с помощью клавиши табуляции перейти на кнопку <Keep>, чтобы возвратиться в главное меню. Учтите, что после заполнения последнего поля необходимо нажать клавишу Enter. Если, не нажав Enter, перейти на кнопку <Keep>, запись будет потеряна.

Диапазоны для подсетей

Наконец, необходимо задать диапазоны IP-адресов, выделяемые различным подсетям. Для этого выберите опцию Subnet главного меню. На рис. 33.6 представлено диалоговое окно конфигурирования подсетей.

В табл. 33.4 описано назначение соответствующих опций.

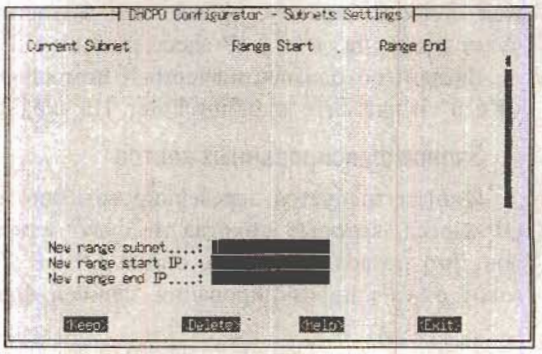


Рисунок 33.6.
Диалоговое окно Subnet, по виду и особенностям работы очень похожее на диалоговое окно Fixed-host entries, описанное в предыдущем разделе

Таблица 33.4. Опции диапазонов для подсетей

Опция	*Описание*
New range subnet	С ее помощью задается подсеть, для которой будет использоваться указанный диапазон IP-адресов.
New range start IP	Начальный IP-адрес, который будет доступен в подсети клиентам DHCP, получающим динамические IP-адреса.
New range end IP	Конечный IP-адрес, который будет доступен в подсети клиентам DHCP, получающим динамические IP-адреса.

Как и при конфигурировании хостов с фиксированными адресами, после ввода диапазона IP-адресов необходимо нажать клавишу Enter, чтобы запись была добавлена в список и изменения учтены. После этого можно добавлять следующие записи.

При необходимости использовать в подсети несколько блоков IP-адресов создайте новую запись, указав в ней другой диапазон IP-адресов. Обе записи будут учтены, и оба диапазона IP-адресов доступны для подсети.

Создав все необходимые записи, перейдите с помощью клавиши табуляции на кнопку <Keep> и нажмите клавишу Enter, чтобы возвратиться в главное меню.

СОВЕТ

Если упоминаемые в этой главе термины, например подсеть, вам не понятны, информацию о них можно найти в главе 22.

Сохранение информации и выход из программы

В главном меню программы DHCP Configurator перейдите с помощью клавиши табуляции на кнопку <Save> и нажмите клавишу Enter. Программа создаст конфигурационный файл **/etc/dhcpd.conf** и завершит работу.

Инструкции, представленные в этом разделе, позволяют настроить и запустить сервер DHCP. Дополнительную информацию о тонкостях конфигурирования можно найти на страницах справочного руководства по демону **dhcpd** и файлу **dhcpd.conf**, где описаны все опции, которые можно задать в конфигурационном файле **dhcpd.conf**.

5 ЧАСТЬ

X-Window

Дополнительные возможности настройки системы X-Window

34 ГЛАВА

Дополнительные возможности настройки системы X-Window

- Обновление версии 3.3.6 до 4.x
- Использование SuperProbe
- Конфигурирование системы X-Window с помощью сценария xf86config
- Опции в файле XF86Config
- Проверка настройки X-Window
- Файл .xinitrc с личными настройками
- Работа со шрифтами
- Использование удаленных клиентов X-Window
- xdm

По умолчанию в системе FreeBSD устанавливается графическая среда XFree86 версии 3.3.6. На данный момент эта версия признана устаревшей. В данной главе обсуждается обновление системы XFree86 до версии 4.1 и конфигурирование последней.

Обновление версии 3.3.6 до 4.x

На момент написания книги последней версией XFree86 была 4.1. Если вы решите использовать XFree86 4.1 вместо XFree86 3.3.6, необходимо установить пакет 4.1 с компакт-диска, прилагаемого к этой книге. Пакет находится в каталоге **packages/x11** и называется **XFree86-4.1.0_4.tgz**. Полное обсуждение установки пакетов см. в главе 15, а здесь будет вкратце рассказано, как установить XFree86 4.1 из командного интерпретатора. Как **root** выполните следующий набор команд:

```
# mount /cdrom
# cd /cdrom/packages/x11
# pkg_add -v XFree86-4.1.0_4.tgz
```

Пакет достаточно велик и его установка требует некоторого времени. По завершении установки несколько элементов необходимо сконфигурировать. Первое, что следует сделать, — это обновить файл **/etc/make.conf**, отразив в нем то, что система XFree86 4 установлена вместо XFree86 3. Для это откройте файл **/etc/make.conf** в текстовом редакторе (если файл отсутствует, создайте его) и добавьте следующую строку:

```
XFREE86_VERSION= 4
```

Затем нужно установить программу **Xwrapper**. Можно, конечно, запускать выполняемый файл **XFree86** от имени пользователя **root**, однако такая методика представляет собой серьезную угрозу защите системы. Обойти подобные проблемы позволяет программа **Xwrapper**.

Программу **Xwrapper** необходимо установить из дерева портов FreeBSD. Для этого вам потребуется соединение с Internet. Программа **Xwrapper** находится в каталоге **/usr/ports/x11/wrapper**. Обсуждение установки программного обеспечения из портов приведено в главе 15. Здесь лишь кратко показана процедура установки **Xwrapper**. Выполните следующие команды:

```
# cd /usr/ports/x11/wrapper
# make
(здесь последует несколько сообщений)
# make install
(еще несколько строк)
#
```

И наконец, необходимо удалить старую версию файла **/etc/XF86Config**, чтобы избежать проблем с новой конфигурацией. Воспользуйтесь командой **rm /etc/XF86Config**. Теперь можно приступить к конфигурированию XFree86 4.1.

Использование SuperProbe

Программа **SuperProbe** проверяет видеооборудование и пытается определить тип видеокарты, установленной в системе, объем оперативной памяти на ней и т.д. **SuperProbe** не позволяет определить все типы оборудования, а информация о неко-

торых известных ей картах иногда является неполной. Тем не менее она очень полезна в ситуациях, когда точно неизвестно, какая видеокарта присутствует в системе. Эта информация необходима для инсталляции системы X-Window.

ПРЕДУПРЕЖДЕНИЕ

Может случиться, что **SuperProbe** приведет к "зависанию" системы. Поэтому убедитесь, что все документы сохранены, и лишь затем запускайте эту программу. Запустить **SuperProbe** может только пользователь **root**.

Введите в командной строке **SuperProbe**, чтобы запустить программу в ее основной форме (об опциях, изменяющих поведение **SuperProbe**, будет рассказано далее). Программа выдаст предупреждение о возможном "зависании" системы и подождет в течение пяти секунд до того, как начать тестирование оборудования. Для выхода из программы достаточно нажать Ctrl+C и вернуться к командной строке. **SuperProbe** проверяет видеооборудование и сообщает о результатах. Вывод программы выглядит примерно так:

```
First video: Super-VGA
        Chipset: Matrox (chipset unknown) (PCI Probed)
                 Signature data: 50 (please report)
        RAMDAC:  Generic 8-bit pseudo-color DAC
                 (with 6-bit wide lookup tables (or in 6-bit mode))
```

Запишите информацию, выведенную **SuperProbe**, чтобы воспользоваться ею в дальнейшем при конфигурировании X-Window.

Поведение **SuperProbe** можно изменить. Одна из наиболее распространенных опций — **-verbose**. При этом программа выводит более подробную информацию о своих действиях. Обратитесь к **man**-странице программы, где рассказано обо всех доступных опциях.

Конфигурирование системы X-Window с помощью сценария xf86config

Если вы не хотите пользоваться графическим средством конфигурации, существует альтернатива — сценарий командного интерпретатора, выполняющий те же функции. Эта программа задает ряд вопросов и на основе ваших ответов настраивает систему X-Window.

ПРЕДУПРЕЖДЕНИЕ

Ошибки при конфигурировании настроек видеокарты могут физически повредить оборудование. Хотя большинство мониторов обладает встроенной защитой и отключается при попытке установить частоту развертки выше, чем возможно, некоторые мониторы пытаются работать в указанном режиме, что может вызвать их повреждению.

ПРЕДУПРЕЖДЕНИЕ

Программа **xf86config** перезаписывает существующий файл **Xf86Config**. Если в системе уже присутствует работоспособный файл **XF86Config**, вначале создайте его резервную копию, а потом экспериментируйте с новой конфигурацией. Файл находится в каталоге **/etc/X11/XF86Config**. Создайте копию с именем, например, **/etc/X11/XF86Config.bak**. Таким образом, если новая конфигурация окажется неудачной, вы всегда сможете вернуться к предыдущим настройкам.

Естественно, для запуска программы необходимы права **root**. Введите в командной строке **xf86config** и нажмите Enter. На экране появится следующее сообщение:

```
Эта программа создает файл XF86Config, основываясь на выборе в пунктах
меню.

Файл XF86Config обычно находится в каталоге /usr/X11R6/etc/X11 или /etc/
X11. Пример XF86Config включен в пакет XFree86. Он содержит конфигурацию
для стандартной карты VGA и монитора с разрешением 640x480. Программа
запрашивает путь к файлу, когда она готова записать информацию в него.

Вы можете воспользоваться примером файла XF86Config и внести изменения в
конфигурацию или же создать его с помощью этой программы.

До того как продолжить работу с этой программой, узнайте, какая
видеокарта используется в системе, какой чипсет установлен на ней, а
также объем оперативной видеопамяти. В этом может помочь SuperProbe.

Для продолжения нажмите клавишу Enter, для окончания Ctrl-C.
```

Первое устройство, с которого начнется настройка, — мышь.

ПРЕДУПРЕЖДЕНИЕ

Дважды проверяйте значения, вводимые в **xf86config**. Программа не обращает внимания на ошибки и опечатки. Если введено неверное значение, то после нажатия клавиши Enter уже нет возможности вернуться и исправить его. Необходимо прервать программу по Ctrl+C и запустить ее заново.

Настройка мыши

Экран настройки мыши в **xf86config** выглядит следующим образом:

```
Вначале укажите протокол мыши. Выберите его из списка:

1.  Microsoft compatible (2-button protocol)
2.  Mouse Systems (3-button protocol)
3.  Bus Mouse
4.  PS/2 Mouse
5.  Logitech Mouse (serial, old type, Logitech protocol)
6.  Logitech MouseMan (Microsoft compatible)
7.  MM Series
8.  MM HitTablet
9.  Microsoft IntelliMouse

Если у вас двухкнопочная мышь, скорее всего, она принадлежит к типу 1,
а если трехкнопочная, возможно, она поддерживает протоколы 1 и 2.
Существует два разновидности последнего типа: мышь с переключателем для
выбора протокола и мышь, принадлежащая протоколу 1 по умолчанию и
требующая нажатия (и удержания) клавиши при загрузке для переключения на
протокол 2. Некоторые мыши можно переключить на протокол 2, послав
специальную последовательность сигналов на последовательный порт (см.
опции ClearDTR/ClearRTS).

Введите номер протокола:
```

Введите число, соответствующее типу вашей мыши и нажмите Enter. Вот несколько советов по определению типа мыши:

- Если мышь имеет 9-контактный разъем, это последовательная мышь. Ей соответствует номера 1, 2, 5 или 6. Большинство таких устройств работают по протоколу 1 или 2, и только новые мыши Logitech поддерживают протокол 1 или 6. Протокол 5 предназначен для устаревших устройств Logitech.

- Если мышь имеет небольшой круглый разъем, это мышь PS/2. Любая такая мышь работает по протоколу 4. Даже, если это мышь Microsoft, но с разъемом PS/2, необходимо выбрать протокол 4. Пункт "мышь Microsoft" отвечает только последовательному устройству.
- Если вы устанавливаете FreeBSD на ноутбуке, встроенное устройство, скорее всего, работает по протоколу 4 (как правило, используется порт PS/2).

После выбора типа мыши и нажатия Enter, программа спросит следующее:

```
Если мышь имеет только две кнопки, рекомендуется включить эмуляцию трех
кнопок.

Ответьте на следующий вопрос 'y' или 'n'.
Включить эмуляцию трех кнопок?
```

Важно отметить, что включить эмуляцию трех кнопок не просто рекомендуется, сделать это нужно обязательно, если у вас двухкнопочная мышь. В системе X-Window используются все три кнопки. Поэтому ответьте **y**, если на вашей мыши их только две. В этом случае одновременное нажатие обеих кнопок эмулирует нажатие средней кнопки мыши.

Затем программа спросит, на каком устройстве находится мышь:

```
Введите полное имя устройства, к которому подключена мышь, например,
/dev/tty00. Чтобы воспользоваться устройством по умолчанию, /dev/mouse,
нажмите Enter.

Mouse device:
```

Если вы не создали ссылку, связывающую реальное устройство мыши с **/dev/mouse**, его необходимо ввести здесь. В табл. 34.1 показаны возможные устройства в зависимости от типа мыши.

Таблица 34.1 Имена устройств для различных типов мыши

Тип мыши	*Устройство мыши*
Мышь PS/2 (или лэптоп)	/dev/psm0
Последовательная мышь на порту COM 1	/dev/cuaa0
Последовательная мышь на порту COM 2	/dev/cuaa1
Последовательная мышь на порту COM 3	/dev/cuaa2
Последовательная мышь на порту COM 4	/dev/cuaa3
Bus mouse	/dev/mse0

Введите соответствующее устройство и нажмите клавишу Enter.

Выбор клавиатуры

Далее программа предложит выбрать тип используемой клавиатуры:

```
Выберите один из следующих типов, наиболее подходящий для вашей
клавиатуры. Если ничего не подходит, выберите 1 (Generic 101-key PC)

   1   Generic 101-key PC
   2   Generic 102-key (Intl) PC
   3   Generic 104-key PC
   4   Generic 105-key (Intl) PC
```

```
  5   Dell 101-key PC
  6   Everex STEPnote
  7   Keytronic FlexPro
  8   Microsoft Natural
  9   Northgate OmniKey 101
 10   Winbook Model XP5
 11   Japanese 106-key
 12   PC-98xx Series
 13   Brazilian ABNT2
 14   HP Internet
 15   Logitech iTouch
 16   Logitech Cordless Desktop Pro
 17   Compaq Internet
 18   Microsoft Natural Pro

Введите номер клавиатуры из списка.
```

Обратите внимание, что все типы клавиатур не помещаются на экране. По нажатию клавиши Enter можно перейти к следующему экрану. Когда список заканчивается, по нажатию Enter он прокручивается сначала.

Если вы используете клавиатуру с американской раскладкой, выберите 1 и нажмите Enter. Программа запросит следующее:

```
Введите вариант имени для раскладки 'us' или примите значение по
умолчанию, нажав клавишу Enter
```

Здесь нет необходимости ничего изменять, поэтому просто нажмите Enter и перейдите к следующему вопросу.

```
Ответьте 'y' или 'n'.
Выбрать дополнительные опции XKB (переключатель группы, индикатор группы
и т.д.)?
```

Выберите здесь **n**, если вам не требуется менять раскладку клавиш. Если же вам это нужно (или чтобы просто просмотреть возможности), нажмите **y**. За нажатием **y** последует несколько меню, в которых можно, к примеру, превратить клавишу Caps Lock в Ctrl (это полезно для профессионалов в Emacs) или поменять их местами. Введите номер нужной опции и нажмите Enter. Чтобы выйти из меню, не предпринимая никаких действий, просто нажмите Enter без номера.

Настройка монитора

В этом разделе настраиваются различные параметры монитора, включая частоту горизонтальной и вертикальной развертки. Вначале программа выдает сообщение:

```
В следующем разделе настраиваются частоты развертки монитора:
Вначале необходимо выбрать спецификации монитора. Двумя основными
параметрами являются частота вертикальной и горизонтальной развертки.

Допустимые диапазоны частот указаны в инструкции к монитору. Если вы
сомневаетесь, обратитесь к списку мониторов в файле /usr/X11R6/lib/X11/
doc/Monitors.

Нажмите Enter для продолжения или Ctrl-C для выхода
```

Нажмите клавишу Enter и перейдите к следующему меню, где будет запрошена частота горизонтальной развертки:

```
Укажите диапазон допустимых частот горизонтальной развертки. Необходимо
указать или один из диапазонов, определенных для стандартных типов
мониторов, или самостоятельно задать диапазон.
```

```
ОЧЕНЬ ВАЖНО, чтобы частота горизонтальной развертки выбранного вами типа
монитора не превышала его возможности. Если сомневаетесь, выбирайте более
консервативные значения.

 частота горизонтальной развертки в Кгц; типы мониторов и их режимы
 1  31.5; Standard VGA, 640x480 @ 60 Hz
 2  31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
 3  31.5, 35.5; 8514 Compatible, 1024x768 @ 87 Hz interlaced (no 800x600)
 4  31.5, 35.15, 35.5; Super VGA, 1024x768 @ 87 Hz interlaced, 800x600 @
    56 Hz
 5  31.5 - 37.9; Extended Super VGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
 6  31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
 7  31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
 8  31.5 - 64.3; Monitor that can do 1280x1024 @ 60 Hz
 9  31.5 - 79.0; Monitor that can do 1280x1024 @ 74 Hz
10  31.5 - 82.0; Monitor that can do 1280x1024 @ 76 Hz
11  Enter your own horizontal sync range

Выберите нужный пункт (1-11):
```

ПРЕДУПРЕЖДЕНИЕ

Несколько следующих вопросов связаны с настройками, которые могут потенциально повредить монитор, если они сконфигурированы некорректно. Убедитесь, что выбранные диапазоны не выше, чем поддерживает монитор.

Если вы точно знаете, какой диапазон частот горизонтальной развертки поддерживает ваш монитор, выберите пункт 11 и введите его вручную. Если точный диапазон вам неизвестен, обратитесь к файлу **/usr/X11R6/lib/X11/doc/Monitors**.

К сожалению, файл не имеет удобного для чтения формата, однако в нем имеется нужная вам информация. Файл содержит конфигурацию для нескольких типов мониторов. Вот пример записи из него:

```
Section "Monitor"
    Identifier "ELSA GDM-17E40"
    VendorName "ELSA GmbH"
    ModelName "GDM-17E40"
    BandWidth 135
    HorizSync 29-82
    VertRefresh 50-150
    ModeLine "640x480" 25 640 664 760 800 480 491 493 525
    ModeLine "640x480" 31 640 664 704 832 480 489 492 520
    ModeLine "800x600x32" 45 800 820 904 964 600 601 604 621
    ModeLine "1024x768x16" 78.7 1024 1044 1140 1264 768 770 773 796
    ModeLine "1152x875" 135 1152 1416 1456 1664 875 875 877 906
    ModeLine "1152x900" 135 1152 1400 1440 1648  900 901 905 935
    ModeLine "1280x1024i" 80 1280 1296 1512 1568 1024 1025 1037 1165
interlace
    ModeLine "1280x1024" 110 1280 1328 1512 1712 1024 1025 1028 1054
    ModeLine "1280x1024" 135 1280 1312 1456 1712 1024 1027 1030 1064
EndSection
```

Строки с командами **Section** и **EndSection** обозначают начало и конец определения параметров монитора.

В данный момент нас интересуют лишь пять строк из этого раздела.

Три первых из них: **Identifier**, **VendorName** и **ModelName**. Они идентифицируют тип монитора, для которого предназначены эти настройки. Отыщите здесь название производителя и монитора.

Еще две нужные строки: **HorizSync** и **VertRefresh**. Если вы найдете раздел, посвященный вашему монитору, запишите эти значения для дальнейшего использования. Выберите в меню пункт 11 и введите их.

Если вы не нашли информации ни в руководстве по монитору, ни в файле **Monitors**, выберите одну из записей, где приведены такие значения частот развертки, которые монитор с гарантией поддерживает.

После выбора частоты горизонтальной развертки из списка или ввода специальных значений и нажатия Enter программа запрашивает частоту вертикальной развертки:

```
Укажите диапазон допустимых частот горизонтальной развертки. Необходимо
указать или один из диапазонов, определенных для стандартных типов
мониторов, или самостоятельно задать диапазон. Для режимов чересстрочной
развертки значение частоты выше (например, 87 Гц, а не 43 Гц).

1   50-70
2   50-90
3   50-100
4   40-150
5   Введите свой диапазон частот вертикальной развертки

Введите нужный пункт:
```

Здесь тоже важно выбрать такой диапазон частот, который поддерживается монитором, иначе это может повредить его. Завершите выбор нажатием Enter.

```
Введите строку с идентификацией монитора: название производителя и
название модели. По нажатию Enter введите названия по умолчанию.

В строках можно использовать пробелы.
Введите строку с идентификацией монитора:
```

Эта информация будет внесена в файл **XF86Config**, генерируемый программой после ввода всех значений. Нажав клавишу Enter, можно принять значения по умолчанию.

Настройка видеокарты

Затем на экране появится информация о настройке видеокарты. Нажмите Enter и перейдите к списку видеокарт, который выглядит следующим образом:

```
 0   2 the Max MAXColor S3 Trio64V+          S3 Trio64V+
 1   2-the-Max MAXColor 6000                 ET6000
 2   3DLabs Oxygen GMX                       PERMEDIA 2
 3   928Movie                                S3 928
 4   AGX (generic)                           AGX-014/15/16
 5   ALG-5434                                CL-GD5434
 6   AOpen PA2010                            Voodo Banshee
 7   ASUS 3Dexplorer                         RIVA128
 8   ASUS PCI-AV264CT                        ati
 9   ASUS PCI-V264CT                         ati
10   ASUS Video Magic PCI V864               S3 864
11   ASUS Video Magic PCI VT64               S3 Trio64
12   AT25                                    Alliance AT3D
13   AT3D                                    Alliance AT3D
14   ATI 3D Pro Turbo                        ati
15   ATI 3D Pro Turbo PC2TV                  ati
16   ATI 3D Xpression                        ati
17   ATI 3D Xpression+                       ati
```

```
Выберите номер требуемой карты.
Для перехода к следующей странице нажмите Enter, для продолжения
конфигурирования — q.
```

В базе данных содержится больше видеокарт, чем может поместиться на экране. Для перехода к следующей странице нажмите клавишу Enter. По достижении последней страницы нажатие Enter приводит вас к началу списка.

Если видеокарта отсутствует в этом списке, просто нажмите q. Не выбирайте модель, которая как-будто похожа на вашу, — это может привести к некорректной работе. Модели, имеющие похожие имена аппаратно не всегда реализованы одинаково. Если же вы нашли свою видеокарту в списке, введите ее номер и нажмите Enter. На экран будет выведена информация о сделанном выборе, например:

```
Определение выбранной видеокарты:

Identifier: Matrox Millennium G400
Chipset:      mgag400
Driver:       mga
Do NOT probe clocks or use any Clocks line.

Нажмите Enter для продолжения или Ctrl-C для выхода.
```

Нажатие Enter позволяет перейти к следующему вопросу, нажатие Ctrl+C — завершить работу программы без сохранения конфигурации. В последнем случае необходимо будет заново запустить **xf86config** и начать настройку сначала.

Далее необходимо ввести некоторую информацию о видеокарте, начиная с объема оперативной памяти на ней:

```
Теперь необходимо ввести информацию о видеокарте. Она будет
использоваться в разделе "Device" файла XF86Config.

Необходимо указать, сколько видеопамяти установлено на карте. Лучше
использовать то значение, которое автоматически определено X-сервером,
который вы собираетесь использовать. Если вы столкнетесь с проблемой, что
сервер не поддерживает имеющийся объем памяти (например, карта ATI Mach64
ограничена 1024 Кб при использовании сервера SVGA), укажите максимальное
число, поддерживаемое сервером.

Сколько видеопамяти установлено на карте:

  1   256K
  2   512K
  3   1024K
  4   2048K
  5   4096K
  6   Other

Введите номер нужного пункта:
```

Если только вы не пользуетесь старой видеокартой, то, скорее всего, на ней установлено больше видеопамяти, чем в предлагаемых опциях, поэтому выберите 6 и введите нужный объем вручную. В этом случае необходимо ввести значение в килобайтах. Помните, что в двоичной математике килобайт равен 1024, а не 1000 байт. В табл. 34.2 приведен пересчет значений из Мб в Кб.

Таблица 34.2 Пересчет значений объема видеопамяти из Мб в Кб

Объем в Мб	*Объем в Кб*
8	8192
16	16384
32	32768
64	65536
128	131072

В пункте **Amount of memory in Kbytes** (Объем памяти в Кб) введите требуемое значение и нажмите Enter.

```
Введите строку с параметрами видеокарты: идентификатор, название
производителя и название модели. По нажатию Enter примите значения по
умолчанию (взятые из определения карты).

Ваша карта определена как Matrox Millennium G400.

В строках можно использовать пробелы.
Введите строку с идентификацией карты:
```

Если вы выбрали карту из списка, здесь будет предложена строка с ее идентификацией/описанием. Выберите значение по умолчанию. Если карта в списке отсутствует, введите ее описание и нажмите Enter.

В зависимости от того, выбрано определение из базы данных или нет, на экране появится разный набор вопросов.

Выбор видеокарты из базы данных

Если вы выбрали описание карты из списка, на экране появится меню с текущим списком видеорежимов. Вот пример того, как он может выглядеть:

```
Для каждой глубины цвета определены режимы (экранное разрешение в
пикселях). Разрешение, с которым сервер будет запускаться по умолчанию,
приведено в списке первым среди поддерживаемых монитором и видеокартой.
Текущие настройки:

"640x480" "800x600" "1024x768" "1280x1024" for 8-bit
"640x480" "800x600" "1024x768" "1280x1024" for 16-bit
"640x480" "800x600" "1024x768" "1280x1024" for 24-bit

Режимы, которые не поддерживаются из-за ограничений монитора или чипсета
видеокарты, пропускаются сервером автоматически.

  1   Изменить режимы для 8-битной глубины цвета (256 цветов)
  2   Изменить режимы для 16-битной глубины цвета (32/64 тысячи цветов)
  3   Изменить режимы для 24-битной глубины цвета (16 миллионов цветов)
  4   Продолжить.

Введите нужный пункт:
```

Скорее всего, вам потребуется внести изменения в эти настройки, поскольку по умолчанию X-Window запускается в режиме с разрешением 640×480, которым не слишком удобно пользоваться. Если у вас 17-дюймовый монитор, вполне подойдет режим 1024×768 или даже выше. Для 19- или 21-дюймового монитора необходим режим более высокого разрешения, чем 1024×768. Если же используется 15-дюймовый монитор, режим 1024×768, может подойти, но лучше заменить его на 800×600. Для 14-

дюймового монитора режим 1024×768 не подойдет, его необходимо заменить на 800×600 или даже на 640×480.

Здесь же необходимо выбрать режим глубины цвета. Как правило, чем выше, тем лучше. Если объем видеопамяти невелик, то придется найти компромиссное решение между глубиной цвета и разрешением. В большинстве случаев разница между 16-битным и 24-битным цветом почти незаметна. Однако разница между 8- и 16-битным цветом очень велика. Разумеется, здесь следует руководствоваться своими вкусами, тем не менее вот несколько рекомендаций:

- Если установленное разрешение не позволяет использовать более чем 8-битный цвет (256 цветов), лучше уменьшить разрешение так, чтобы цвет был 16-битным.
- Если нужное вам разрешение поддерживает 16-, но не поддерживает 24-битный цвет, в большинстве случаев (если только вы не работаете с высококачественной графикой) уменьшать его не стоит, поскольку вы не заметите разницы между цветовыми режимами.

После того как вы выбрали глубину цвета, которая будет использоваться по умолчанию, измените режим разрешения для нее. Введите номер строки и нажмите Enter. Например, если по умолчанию вы планируете использовать 24-битный цвет, выберите 3 в приведенном выше примере.

Выберите режим из следующего списка:

```
1   "640x400"
2   "640x480"
3   "800x600"
4   "1024x768"
5   "1280x1024"
6   "320x200"
7   "320x240"
8   "400x300"
9   "1152x864"
a   "1600x1200"
b   "1800x1400"
c   "512x384"

Введите цифры, отвечающие выбранным режимам.
Например, 432 отвечает режимам "1024x768" "800x600" "640x480", с режимом
по умолчанию 1024x768.

Введите режим
```

Выберите номер, соответствующий режиму, в котором X-Window будет запускаться по умолчанию. Например, если нужно использовать 1024×768, выберите **4**.

Если вы хотите переключаться между разными режимами, можно указать сразу несколько цифр. Первая из них отвечает режиму по умолчанию, а остальные будут использоваться при выполнении команд, изменяющих текущий режим. Например, если по умолчанию используется 1024×768, затем — 800×600 и, в заключение, 640×480, введите **432**.

Большинство пользователей не изменяет режим экрана. Если вы не применяли эту функцию в системе Windows или Macintosh, скорее всего, она не понадобится вам и во FreeBSD. Поэтому можете ограничиться здесь одним режимом. После вы-

бора режима разрешения для заданной глубины цвета введите Enter. Затем программа запросит информацию о виртуальных экранах:

```
Можно использовать несколько виртуальных экранов (рабочих столов), когда
экран больше, чем физическая область монитора. При этом "видимое окно"
перемещается при приближении мыши к краю экрана. Если виртуальный рабочий
стол не требуется, вы не сможете использовать здесь большие режимы. Для
каждой глубины цвета возможны виртуальные экраны разного размера

Ответьте 'y' или 'n'.
Использовать виртуальный экран, больший, чем физическая область?
```

Когда виртуальный экран больше физических размеров, некоторые его части будут находиться за пределами видимой области. Для его просмотра необходимо прокручивать экран с помощью мыши.

С моей точки зрения виртуальные экраны не очень удобны в работе. Пользуйтесь ими, если вы вынуждены работать с низким разрешением (640×480 или меньше). Но даже в такой ситуации от них не слишком много пользы, поэтому я рекомендую ответить **n** (нет).

Если вам требуется виртуальный экран, программа запросит его разрешение. Выберите требуемые размеры и нажмите Enter.

Если вам не требуется виртуальный экран, программа вернется к предыдущей конфигурации. На экране будет следующее:

```
Для каждой глубины цвета определены режимы (экранное разрешение в
точках). Разрешение, с которым сервер будет запускаться по умолчанию,
приведено в списке первым среди поддерживаемых монитором и видеокартой.
Текущие настройки:

"640x480" "800x600" "1024x768" "1280x1024" for 8-bit
"640x480" "800x600" "1024x768" "1280x1024" for 16-bit
"1024x768" for 24-bit

Режимы, которые не поддерживаются из-за ограничений монитора или чипсета
видеокарты, пропускаются сервером автоматически.

  1   Изменить режимы для 8-битной глубины цвета (256 цветов)
  2   Изменить режимы для 16-битной глубины цвета (32/64 тыс. цветов)
  3   Изменить режимы для 24-битной глубины цвета (16 млн. цветов)
  4   Продолжить.

Введите нужный пункт:
```

Обратите внимание, что строка с 24-битным цветом изменилась — теперь включен только режим 1024×768.

Если вы не планируете использовать несколько режимов с разной глубиной цвета, остальные настройки можно не изменять. В ином случае введите номер нужного цветового режима и сконфигурируйте разрешение для него, как в предыдущем случае.

По окончании настройки всех режимов введите **4** и перейдите к следующему меню. Программа запросит требуемый режим глубины цвета:

```
Укажите режим глубины цвета, используемый по умолчанию:

  1   1 bit ... — 1 бит (монохром)
  2   4 bits ... — 4 бита (16 цветов)
  3   8 bits ... — 8 бит (256 цветов)
  4   16 bits ... — 16 бит (65536 цветов)
```

```
   5   24 bits ... — 24 бита (16 миллионов цветов)

Enter ... — Введите номер режима.
```

Введите номер режима и нажмите Enter.

Сохранение конфигурации в файле

После выбора режима глубины цвета программа запросит подтверждение на сохранение изменений:

```
Программа внесет изменения в файл XF86Config. Убедитесь, что это не
перезапишет вашу предыдущую конфигурацию.

Внести изменения в файл /etc/X11/XF86Config?
```

Выберите **y** и запишите изменения в файл **XF86Config**. Если файл **XF86Config** уже существует, он будет перезаписан новыми настройками.

После нажатия **y** программа **xf86config** выведет следующее сообщение:

```
Произведена запись в файл. Проверьте его до запуска команды 'startx'.
Файл XF86Config должен находиться в одном из каталогов, где сервер
производит поиск (например, /etc/X11). В графическом режиме переключить
разрешение можно с помощью нажатия комбинации клавиш ctrl, alt и '+'.
Одновременное нажатие ctrl, alt и backspace немедленно завершает работу
сервера (воспользуйтесь этой комбинацией, если монитор не работает в
определенном режиме).

Сведения о последующем конфигурировании находятся в файле /usr/X11R6/lib/
X11/doc/README.Config.
```

После этого управление вернется к командной строке. Вы можете пропустить следующий раздел и перейти сразу же к разделу "Проверка настройки X-Window".

Если видеокарта отсутствует в списке базы данных

Если карта не была выбрана из списка, программа вначале задаст вопрос о режиме глубины цвета:

```
Укажите режим глубины цвета, используемый по умолчанию:

   1   1 бит (монохром)
   2   4 бита (16 цветов)
   3   4 бит (256 цветов)
   4   16 бит (65536 цветов)
   5   24 бита (16 миллионов цветов)

Введите номер режима.
```

Выберите нужный режим и нажмите Enter.

Далее система запросит подтверждение на запись изменений:

```
Программа внесет изменения в файл XF86Config. Убедитесь, что это не
перезапишет вашу предыдущую конфигурацию.

Внести изменения в файл /etc/X11/XF86Config?
```

Выберите **y** и запишите изменения в файл XF86Config. Если файл **XF86Config** уже существует, он будет перезаписан новыми настройками. После нажатия **y** программа **xf86config** выведет следующее сообщение:

```
Произведена запись в файл. Проверьте его до запуска команды 'startx'.
Файл XF86Config должен находиться в одном из каталогов, где сервер
производит поиск (например, /etc/X11). В графическом режиме переключить
разрешение можно с помощью нажатия комбинации клавиш ctrl, alt и '+'.
```

```
Одновременное нажатие ctrl, alt и backspace немедленно завершает работу
сервера (воспользуйтесь этой комбинацией, если монитор не работает в
определенном режиме).

Сведения о последующем конфигурировании находятся в файле /usr/X11R6/lib/
X11/doc/README.Config.
```

По умолчанию система X-Window запускается в режиме с разрешением 640x480. Чтобы заменить его более высоким, необходимо вручную изменить настройки в файле **XF86Config**. О его содержимом рассказано в следующем разделе.

Опции в файле XF86Config

Как и большинство аспектов настройки FreeBSD, конфигурацией системы X-Window управляет файл в обычном текстовом формате. Основной конфигурационный файл X-Window расположен в каталоге **/etc/X11** и называется **XF86Config**. Этот файл можно создать и/или изменить с помощью **xf86cfg** (конфигуратор X-Window с графическим интерфейсом) или **xf86config** (средство настройки X-Window с текстовым интерфейсом). Эти программы значительно упростили настройку X-Window по сравнению с редактированием файла вручную.

Однако есть несколько ситуаций, когда приходится вносить изменения в конфигурацию вручную. Например, если видеокарта отсутствует в списке. Или же когда система X-Window была настроена с помощью **xf86config**, а затем необходимо внести изменения в настройки мыши (например, изменить скорость). Кроме того, когда требуется внести лишь несколько незначительных изменений, желательно сразу внести их в файл, а не заново производить весь процесс настройки в **xf86config**.

До того как вносить изменения в файл **XF86Config**, создайте его резервную копию, чтобы легко устранить нежелательные изменения. Скопируйте **/etc/X11/XF86Config**, например, как **/etc/X11/XF86Config.bak**. После этого откройте его в текстовом редакторе (см. главу 7, где содержится информация о редакторах, включенных в состав FreeBSD).

СОВЕТ

Помните, что FreeBSD (и другие операционные системы, подобные UNIX) учитывают регистр. Поэтому не путайте текстовый конфигуратор (**xf86config**) с реальным файлом конфигурации (**XF86Config**). Так как регистр учитывается, это два разных файла.

Синтаксис файла XF86Config

Файл **XF86Config** разбит на несколько разделов. Каждый из них относится к определенному устройству или опциям конфигурации. Каждый раздел начинается с ключевого слова **Section**, за которым дано его название в кавычках. На окончание раздела указывает ключевое слово **EndSection**. Тело раздела задано в формате, пригодном для чтения. Комментарии начинаются с символа диеза (#) и простираются до конца строки. Вот пример этого раздела из файла **XF86Config**:

```
Section "Module"
# Загрузка модуля расширения DBE.
    Load        "dbe"    # Расширение двойного буфера
```

```
# Загрузка различных модулей расширения и запрещение инициализации
# расширения XFree86-DGA в модуле.
      SubSection   "extmod"
        Option      "omit xfree86-dga"     # не инициализировать расширение
                                           # DGA extension
      EndSubSection

# Загрузка модулей шрифтов Type1 и FreeType
      Load           "type1"
      Load           "freetype"

# Загрузка модуля GLX
#      Load          "glx"

EndSection
```

Комментарии, предшествующие этому разделу, не показаны. В файле **XF86Config** они объясняют назначение каждого раздела. В данном примере раздел **Module** загружает динамические модули, необходимые серверу для поддержки различных элементов. Например, часть команд после комментария **# This loads the Type1 and FreeType font modules** (Загрузка модулей шрифтов Type1 и FreeType) загружает модули для поддержки различных типов шрифтов. В данном случае в первой строке **Load** включена поддержка шрифтов Adobe Type 1, а во второй — модуль **freetype**, распространяемый свободно и необходимый X-Window для использования шрифтов TrueType.

Кроме основных разделов, существуют и вложенные подразделы. Они начинаются с ключевого слова **SubSection**, за которым следует ее название в кавычках, а заканчиваются словом **EndSubSection**. Как и основные разделы, они также предназначены для чтения.

Далее рассмотрены разделы и подразделы из файла **XF86Config**.

Раздел "Modules"

В этом разделе включается динамическая загрузка модулей, необходимых для поддержки различных компонентов, например, типов шрифтов. Динамические модули не являются частью двоичного файла X-сервера, а загружаются в момент запуска X-Window. Преимущество их в том, что они загружаются лишь в случае необходимости. Таким образом, ненужные модули не занимают память и другие системные ресурсы. Например, если в вашей системе нет шрифтов TrueType, не имеет смысла включать эту поддержку в X. Поэтому строку **Load "freetype"** можно закомментировать.

Названия загружаемых модулей следуют за ключевым словом **Load** в кавычках. Вот пример строки, включающей поддержку шрифтов TrueType:

```
Load "freetype"
```

Раздел "Files"

Этот раздел в X-Window аналогичен переменной среды PATH. В нем указано, где система может найти различные файлы.

Единственная часть этого раздела, которую, возможно, следует изменить, — это команды, указывающие X-Window, где находятся шрифты. Каждый каталог шрифта начинается с ключевого слова **FontPath**, за которым следует название каталога в

кавычках. Например, следующая команда задает местоположение шрифтов Adobe Type 1:

```
FontPath    "/usr/X11R6/lib/X11/fonts/Type1/"
```

В этот же каталог следует устанавливать загруженные или купленные шрифты Adobe Type 1. Подробнее о шрифтах в X-Window рассказано далее в этой главе.

При добавлении новых шрифтов название каталога не играет роли. Например, шрифты Adobe Type 1 можно установить в каталог **Type1**. Этому соглашению следуют лишь для того, чтобы легче понять, какой тип шрифтов находится в каталоге.

Раздел "ServerFlags"

Раздел содержит глобальные опции, управляющие поведением X-Window. Некоторые из них представлены в конфигурационном файле, генерируемом программой **xf86config**, и включают сведения об их назначении. Все опции по умолчанию закомментированы.

Все опции этого раздела начинаются с ключевого слова **Option**, за которым в кавычках следует опция. Так, например, следующая строка отключает возможность завершения работы X-сервера комбинацией клавиш Ctrl+Alt+Backspace:

```
Option "DontZap"
```

(Эту опцию не следует отключать до тех пор, пока вы не проверите работу системы X-Window и не убедитесь в корректности конфигурации.)

В следующих разделах рассказано о некоторых часто используемых опциях.

Option "NoTrapSignals"

Если опция включена, X-Window не сможет завершить работу корректно при возникновении ошибки. Вместо этого система запишет файл с дампом ядра. Это может привести к неработоспособности консоли после некорректного выхода из X-Window.

Эту опцию лучше оставить отключенной, если только вы не сталкиваетесь с постоянными сбоями X-сервера. В последнем случае файл с дампом ядра содержит информацию, необходимую для устранения неисправностей. Этот файл необходимо отправить проекту Xfree86 вместе с отчетом об ошибке.

Option "DontZap"

Включение этой опции запрещает останов X-сервера по комбинации клавиш Ctrl+Alt+Backspace. Убирать комментарий из этой строки не стоит до тех пор, пока вы не протестируйте X-Window и не убедитесь, что система работает корректно. Но и после этого нет смысла включать эту опцию. Единственной причиной могут послужить программы, которые используют эту комбинацию клавиш.

Option "DontZoom"

Включение этой опции запрещает переключение видеорежимов с помощью комбинаций клавиш Ctrl+Alt+keypad+ и Ctrl+Alt+keypad-. По умолчанию система позволяет переходит от одного режима к другому, если они настроены.

Единственной причиной для включения опции могут послужить программы, в которых используется эта комбинация клавиш.

Option "DisableVidModeExtension"

Включение этой опции предотвращает внесение каких-либо изменений в настройку программой **xvidtune**. Если система многопользовательская, желательно включить эту опцию, чтобы пользователи не могли применять **xvidtune**, поскольку неверные настройки могут повредить монитор. Обратите внимание, что даже если эта строка включена, программу **xvidtune** все равно можно запустить. Однако внести какие-либо изменения в систему с ее помощью нельзя.

Большинство опций из этого раздела, установленных программой **xf86config**, желательно не изменять.

Есть несколько опций, которые могут понадобиться, хотя и не входят в конфигурацию, настраиваемую по умолчанию **xf86config**. Здесь обсуждаются лишь основные опции. Если вы хотите добавить какую-либо из опций, добавьте соответствующую строку в файл **XF86Config**.

Option "AllowMouseOpenFail"

По умолчанию, если X-сервер не может открыть устройство мыши или другое координатно-указательное устройство, он не запускается и выдает сообщение об ошибке. Добавление этой строки позволяет запустить сервер даже при отсутствии мыши.

Если только вы не пытаетесь использовать X-Window без мыши или другого координатно-указательного устройства, нет причин изменять эту опцию.

Option "BlankTime" "n"

Эта опция заставляет экран погаснуть через **n** минут. Если значение не указано, по умолчанию это происходит через 10 минут.

Option "StandbyTime "n"

Эта опция заставляет монитор перейти в режим ожидания (standby) после **n** минут. Если значение не указано, по умолчанию это происходит через 20 минут.

Эта опция поддерживается не всеми видеодрайверами и работает только для мониторов, поддерживающих DPMS. Чтобы применить ее, в разделе файла **XF86Config**, посвященном монитору (он обсуждается далее в этой главе), необходимо специально указать поддержку DPMS.

Option "SuspendTime" "n"

Эта опция заставляет монитор перейти в режим отключения (suspend mode) через **n** минут. Если значение не указано, по умолчанию это происходит через 30 минут.

Эта опция поддерживается не всеми видеодрайверами и работает только для мониторов, поддерживающих DPMS. Чтобы применить ее, в разделе файла **XF86Config**, посвященном монитору, необходимо указать поддержку DPMS.

Option "OffTime" "n"

Эта опция отключает монитор через **n** минут. Если значение не указано, по умолчанию это происходит через 40 минут.

Эта опция поддерживается не всеми видеодрайверами и работает только для мониторов, поддерживающих DPMS. Чтобы применить ее, в разделе файла **XF86Config**, посвященном монитору, необходимо указать поддержку DPMS.

Option "NoPM"

Эта опция отключает определенные события, связанные с управлением питанием. По умолчанию управление питанием разрешено в тех системах, которые его поддерживают. Включать эту опцию следует лишь в том случае, когда вы сталкиваетесь со странными проблемами, которые могут быть связаны со средствами управления питанием.

Есть еще несколько опций, которые можно включить в этом разделе, но они используются реже. Если они потребуются вам, ознакомьтесь со **man**-страницей **XF86Config**.

Раздел "InputDevice"

Этот раздел предназначен для настройки устройств ввода. В конфигурационном файле может быть несколько разделов **InputDevice**. Обычно их не менее двух: для клавиатуры и мыши (или другого координатно-указательного устройства).

Разделам **InputDevice** можно поставить в соответствие несколько ключевых слов.

Первым ключевым словом является **Identifier**. За ним следует имя, идентифицирующее устройство. Системе X-Window не важно, как называется устройство, но лучше давать описательные имена.

Вторым ключевым словом является **Driver**. За ним следует имя драйвера в кавычках. Наиболее распространены драйверы **"keyboard"** и **"mouse"**; иногда применяется драйвер **"microtouch"**, предназначенный для поддержки экранов "touch screen".

Заключительная часть раздела **InputDevice** состоит из опций устройства.

В следующих разделах мы рассмотрим настройки для наиболее распространенных устройств: клавиатуры и мыши.

Клавиатура

Вот пример первой части раздела **InputDevice** с настройками клавиатуры:

```
Section "InputDevice"
      Identifier   "Keyboard1"
      Driver         "Keyboard"
```

Option "Protocol"

Если эта опция опущена или закомментирована, используется значение по умолчанию **"standard"**. Вероятно, изменять ее нет необходимости.

Option "AutoRepeat" "x y"

Эта опция управляет автоповтором; **x** представляет собой задержку в миллисекундах, после которой ввод клавиши начнет повторяться; **y** представляет собой количество повторов в секунду.

По умолчанию используются значения 500 миллисекунд и 30 раз в секунду.

Option "XkbRules" "xfree86"

Эта опция управляет тем, как интерпретируются различные аспекты клавиатуры. В большинстве случаев следует оставить значение **"xfree86"**. Если вы работаете на платформе PC-98 японского производства, установите значение **"xfree98"**.

Option "XkbModel" "pc104"

Если используется клавиатура Windows со 104 клавишами, опция имеет значение **"pc104"**, если со 101 клавишей — **"pc101"**. Эти значения применяются и на лэптопах. Хотя обычно клавиатура последних содержит меньшее число клавиш, дополнительные клавиши эмулируются.

Option "XkbLayout" "us"

Обычно эта опция имеет значение **"us"**. Если вы работаете на японской платформе PC-98, измените ее значение на **"nec/jp"**.

Option "XkbOptions" "ctrl:swapcaps"

Если эта опция не закомментирована, клавиша Caps Lock становится Ctrl, а левая клавиша Ctrl — Caps Lock. Такими установками пользуются профессионалы Emacs (в Emacs большинство команд связано с клавишей Ctrl) или же те пользователи, на UNIX-клавиатуре которых клавиша Ctrl размещена на месте клавиши Caps Lock.

Еще несколько опций управляют поведением клавиатуры, но используются гораздо реже.

Мышь

Вот пример первой части раздела **InputDevice** с настройками мыши:

```
Section "InputDevice"
# Идентификатор и драйвер
    Identifier   "Mousel"
    Driver        "mouse"
```

В следующих разделах описано несколько часто используемых опций.

Option "Protocol" "*protocol*"

Здесь ***protocol*** указывает тип мыши, используемый в системе. Эта опция обязательна, поскольку при ее отсутствии X-сервер не работает.

В большинстве случаев здесь можно указать **Auto,** и X-сервер определит тип мыши автоматически. Доступно и несколько других опций. Вот типы возможных протоколов: **Auto**, **Microsoft**, **MouseSystems**, **MMSeries**, **Logitech**, **MouseMan**, **MMHitTab**, **GlidePoint**, **IntelliMouse**, **ThinkingMouse**, **AceCad**, **PS/2**, **ImPS/2**, **ExplorerPS/2**, **ThinkingMousePS/2**, **MouseManPlusPS/2**, **GlidePointPS/2**, **NetMousePS/2**, **NetScrollPS/2**, **BusMouse**, **SysMouse**, **WSMouse**, **USB**, **Xqueue**.

Несколько советов по выбору подходящего протокола:

- Протокол **Logitech** используется только устаревшими последовательными мышами производства Logitech. Если у вас более новая последовательная мышь Logitech, необходимо использовать протокол Microsoft или MouseMan.
- Для мыши PS/2 используется протокол PS/2, независимо от того, кто является ее производителем.
- На лэптопе или ноутбуке встроенное координатно-указательное устройство, обычно работает по протоколу PS/2.

Вначале воспользуйтесь опцией **Auto** и изменяйте ее лишь в том случае, когда мышь не определяется или не работает корректно.

Option "Device" "*devicename*"

Эта опция указывает, на каком устройстве установлена мышь. Опция является обязательной, без нее X-сервер не работает.

В табл. 34.3 показаны возможные устройства, где может быть расположена мышь.

Таблица 34.3 Имена устройств для различных типов мыши

Тип мыши	*Устройство мыши*
Мышь PS/2 (или лэптоп)	/dev/psm0
Последовательная мышь на порту COM 1	/dev/cuaa0
Последовательная мышь на порту COM 2	/dev/cuaa1
Последовательная мышь на порту COM 3	/dev/cuaa2
Последовательная мышь на порту COM 4	/dev/cuaa3
Bus mouse	/dev/mse0

devicename необходимо заменить именем устройства мыши.

Option "Buttons" "n"

В большинстве случаев число кнопок мыши определяется автоматически. Их можно также указать с помощью данной опции.

n равно числу кнопок мыши. Распространены опции 2 и 3, хотя поддерживается значение до 5. Если у вас есть мышь с пятью кнопками, вы сможете пользоваться ими всеми.

Option "Emulate3Buttons"

Эта опция позволяет мыши с двумя кнопками эмулировать трехкнопочную. Она оказывается полезной в системе с двухкнопочной мышью, поскольку в системе X-Window используются все три кнопки.

После включения опции средняя кнопка мыши эмулируется одновременным нажатием обеих кнопок.

Option "Emulate3Timeout" "n"

Если установлена опция **Emulate3Buttons**, данная настройка задает число миллисекунд, которое должно пройти между нажатием левой и правой кнопки мыши, чтобы система X-Window рассматривала их как отдельные, а не эмулируемое нажатие средней кнопки. Другими словами, обе кнопки необходимо нажать в течение этого интервала времени, чтобы это действие воспринималось системой как нажатие средней кнопки. **n** — число миллисекунд.

Если эта строка отсутствует, а опция **Emulate3Buttons** включена, **Emulate3Timeout** по умолчанию имеет значение 50 миллисекунд.

Эти опции используются при конфигурировании мыши наиболее часто.

Раздел "Monitor"

Здесь устанавливаются частоты горизонтальной и вертикальной развертки монитора. Раздел включает ключевое слово **Identifier**, за которым в кавычках следует имя, идентифицирующее монитор. Само имя не имеет значения.

Два других ключевых слова также являются обязательными. Они перечислены далее.

HorizSync

Частота горизонтальной развертки в Кгц, поддерживаемая монитором. Ее можно указать несколькими способами:

- **Как диапазон**. Обычно так конфигурируют мониторы, поддерживающие несколько частотных режимов (на сегодняшний день такими являются почти все мониторы). Например, **HorizSync 44-76** используется для монитора, диапазон допустимых частот которого простирается от 44 до 76 Кгц.
- **Одно значение**. Оно используется, если монитор поддерживает только одну частоту.
- **Список частот**. Если монитор поддерживает несколько фиксированных частот, их можно перечислить в списке, разделенном запятыми, например, **"HorizSync 31.5, 35.2"**.
- **Как несколько диапазонов**. Если монитор поддерживает несколько диапазонов частот, их можно указать через запятую, например, **"HorizSync 15-25, 30-50"**.

VertRefresh

Это ключевое слово также должно быть указано в этом разделе. Оно задает значения частот вертикальной развертки, которые поддерживает монитор и имеет такой же формат, как и **HorizSync**.

ПРЕДУПРЕЖДЕНИЕ

Использование значений в **HorizSync** и **VertRefresh**, которые находятся за пределами диапазонов частот, поддерживаемых монитором, может повредить монитор. Поэтому с ними необходимо быть чрезвычайно осторожным. Обратитесь к руководству по монитора или документации на Web-сайте производителя.

В разделе **Monitor** можно указать еще несколько опций, но они используются реже. Полный список поддерживаемых опций имеется в **man**-странице **XF86Config**.

Раздел "Device"

В этом разделе устанавливаются параметры графического адаптера. Как и большинство других разделов, он начинается с ключевого слова **Identifier**, за которым в кавычках следует имя устройства.

Ключевое слово **Driver** с именем драйвера является обязательным. Например, для видеокарты NVIDIA TNT2 строка выглядит следующим образом:

```
Driver "nv"
```

Если ваша видеокарта поддерживается, настроить графическое устройство гораздо проще с помощью утилиты **xf86config** (о которой было рассказано ранее), чем делать это, редактируя файл **XF86Config**. Но если по какой-либо причине вам необ-

ходимо сделать это вручную, полный список поддерживаемых карт и драйверов для них можно найти по адресу **www.xfree86.org/current/Status.html**.

Одно из часто используемых ключевых слов в этом разделе — **VideoRam**, задающее объем видеопамяти. Если, например, карта содержит 16 Мб RAM, строка должна выглядеть следующим образом:

VideoRam 16384

Объем видеопамяти указывается в килобайтах. Помните, что в двоичной математике один килобайт равен 1024 байтам. Преобразование мегабайтов в килобайты для различных значений приведено в табл. 34.2.

В этом разделе используется еще несколько опций графических устройств, но они не нужны для большинства видеокарт. Полный список возможных опций приведен на странице справочного руководства по **XF86Config**.

Кроме того, посетите Web-страницу **http://www.xfree86.org/4.0.2/index.html** и проверьте, нет ли на ней специальных замечаний, касающихся вашей видеокарты и специальных опций, требующихся ей.

ПРИМЕЧАНИЕ

Если вы не можете найти драйвер для видеокарты, еще не все потеряно. Попытайтесь включить поддержку адаптера с помощью драйвера Vesa. Он представляет собой общий стандарт спецификаций режимов супер-VGA. Этот драйвер не позволит полностью поддерживать свойства видеоускорителя или все возможные режимы разрешения и глубины цвета, однако позволит работать с X-Window, имея видеоадаптер, не поддерживаемый на текущий момент. Если вы поступите таким образом, периодически проверяйте базу данных видеокарт, поскольку она постоянно пополняется новыми устройствами.

Раздел "Screen"

В разделе **Screen** содержатся настройки дисплея (т.е. комбинации монитора и графической карты), необходимые для работы X-сервера. Он начинается с ключевого слова **Identifier**, за которым в кавычках следует имя дисплея. В файле **XF86Config** может содержаться несколько таких разделов. Первый из них будет использоваться, если в разделе **ServerLayout** не установлены другие опции.

Приведенные ниже ключевые слова являются обязательными в этом разделе.

Device "devicename"

Это имя графического устройства, которое используется при конфигурировании дисплея. **devicename** (имя устройства) представляет собой имя, заданное ключевым словом **Identifier** в разделе с опциями графического устройства.

Monitor "monitorname"

Имя монитора, который используется при конфигурировании дисплея. **Monitorname** (имя монитора) задано в разделе опций монитора (ключевым словом **Identifier**).

DefaultDepth "n"

Режим глубины цвета, который используется по умолчанию. Допустимые значения **n** включают 8, 16 и 24 (в предположении, что их поддерживает видеокарта).

О других допустимых опциях можно узнать на странице справочного руководства **XF86Config**.

После настройки раздела дисплея необходимо настроить подраздел(ы) с его опциями.

Подраздел "Display"

В подразделе **display** настраиваются режимы разрешения и порядок, в котором их можно переключать для каждого установленного режима глубины цвета. В файле должен быть включен хотя бы один подраздел для глубины цвета по умолчанию, иначе X-сервер не сможет запуститься. Для каждого цветового режима используется один подраздел **display**.

В первой строке подраздела находится ключевое слово **Depth**, за которым следует значение режима, опции которого он содержит. Вот пример того, как может выглядеть его начало:

```
Subsection "Display"
     Depth          24
```

Третьей обязательной строкой является строка с режимами разрешения. Она имеет следующий формат: ключевое слово **Modes**, за которым следует список поддерживаемых разрешений. Первое из них будет использоваться по умолчанию, а перейти к остальным позволяют комбинации клавиш Ctrl+Alt+keypad+ и Ctrl+Alt+keypad-.

Например:

```
Modes "1024x768" "800x600" "640x480"
```

В этом примере для данного режима глубины цвета по умолчанию используется разрешение 1024x768. Нажатие Ctrl+Alt+keypad+ или Ctrl+Alt+keypad- позволяет перейти к 800x600 и 640x480.

Система X-Window позволяет работать с виртуальным дисплеем, разрешение которого больше реального. Например, разрешение физического экрана может быть равным 800x600, а виртуального 1024x768. Об этом рассказано в предыдущем разделе этой главы, где речь шла о настройке X-Window с помощью программы **xf86config**.

Приведенные ниже опции управляют поведением виртуального дисплея.

Опция Virtual resolution

Задает разрешение виртуального дисплея, например **"Virtual 1024x768"**. Если разрешение реального экрана меньше, отображается лишь часть рабочего стола. Чтобы просмотреть часть за пределами экрана, необходимо мышью прокрутить "видимое окно" (подведя ее курсор к краю экрана).

О других возможных опциях в подразделе дисплея можно узнать из **man**-страницы **XF86Config**.

Раздел "ServerLayOut"

Раздел **ServerLayout** является дополнительным. Если он отсутствует, используются первые из перечисленных экранов, клавиатур и устройств мыши. Если раздел **ServerLayout** включен в файл конфигурации, в нем указано, какие устройства должен использовать X-сервер.

Обязательным является ключевое слово **Identifier**, за которым следует имя в кавычках. Вот несколько опций, которые обычно заданы в разделе **ServerLayout**:

Screen "screenname"

Здесь задана конфигурация экрана, который используется для этого устройства. **screenname** отвечает имени, указанном в ключевом слове **Identifier** для раздела дисплея в **ServerLayout**.

InputDevice "keyboardname" "CoreKeyboard"

Эта команда указывает, какая клавиатура должна использоваться для этого раздела **ServerLayout**. **keyboardname** это имя, указанное в ключевом слове идентификатор в разделе с опциями клавиатуры.

Опция **CoreKeyboard** делает эту клавиатуру клавиатурой по умолчанию.

InputDevice "mousename" "CorePointer"

Эта опция аналогична опции для клавиатуры. Она указывает устройство мыши, используемое с **ServerLayout**. Опция **CorePointer** делает это устройство мыши устройством по умолчанию.

Строки опций можно включить и в раздел **ServerLayout**. Если они включены и при этом конфликтуют с опциями, перечисленными в других разделах, приоритет имеют опции, находящиеся здесь.

Другие настройки, управляющие различными аспектами работы системы в файле **XF86Config**, используются реже. Чтобы узнать обо всех возможных опциях, обратитесь к **man**-странице **XF86Config**.

Проверка настройки X-Window

После того как работа с программой **xf86config** и/или редактирование файла **XF86Config** завершены, все готово к тестированию X-сервера.

Для запуска X-Window введите команду **startx**. Если все настроено правильно, экран на секунду погаснет, а затем появится заштрихованное поле с символом **x** посредине. **x** представляет собой курсор мыши. Через несколько секунд запустится менеджер окон, и курсором мыши можно будет управлять.

Если курсор мыши неподвижен или X-Window запускается, но затем завершает работу с ошибкой, проверьте файл **XF86Config** и убедитесь, что все настроено правильно.

ПРЕДУПРЕЖДЕНИЕ

Если при запуске X-Window изображение на экране искажается, дрожит и/или при этом слышен высокочастотный свист, исходящий от монитора, НЕМЕДЛЕННО выключите монитор или нажмите комбинацию клавиш Ctrl+Alt+Backspace для аварийного завершения работы X-сервера. Любой из подобных симптомов указывает, что установленная частота развертки выше, чем монитор может выдержать (а это значит, что трансформатор межстрочной развертки вот-вот выйдет из строя). После завершения работы X-сервера, перенастройте частоты развертки и/или режимы разрешения в файле **XF86Config** (или с помощью сценария **xf86config**). Затем вновь попытайтесь запустить сервер.

Если после выполнения команды **startx** монитор гаснет, а затем выключается или переходит в режим отключения (индикатор питания изменяет цвет, экран начинает

мигать или слышен звук статического разряда), это значит, что указаны параметры, недопустимые для монитора. Для аварийного завершения работы X-сервера нажмите Ctrl+Alt+Backspace. После этого управление вновь вернется к командной строке. Настройте заново частоты развертки и/или режимы разрешения и снова попытайтесь запустить сервер.

Следующий раздел посвящен файлу **.xinitrc** с пользовательскими настройками, он находится в начальном каталоге.

Файл .xinitrc с личными настройками

Изменение файла **.xinitrc** — основной способ управления такими настройками X-Window, как менеджер окон, автоматический запуск приложений, цвет фона или фоновое изображение. Каждой из этих операций посвящен один из следующих разделов.

Смена менеджера окон

В главах 4, 5 и 6 речь шла, в основном, о графической среде GNOME (Gnome Desktop Environment). Однако если Gnome вам не подходит, можно выбрать другой менеджер окон.

Существует две популярных графических среды: Blackbox и WindowMaker. Оба эти менеджера окон включены в набор портов FreeBSD (в каталоге **x11-wm**). В главе 15 рассказано о том, как работать с набором портов FreeBSD.

На рис. 34.1 и 34.2 показан внешний вид менеджеров Blackbox и WindowMaker.

Blackbox это идеальный менеджер окон для систем с небольшим объемом оперативной памяти. Кроме того, он часто используется на серверах, где обширные возможности других графических сред просто не нужны.

Рисунок 34.1 *Blackbox — небольшой и быстрый менеджер окон. Он имеет несколько интересных свойств и потребляет очень мало системных ресурсов.*

Рисунок 34.2 *WindowMaker основан на оконной системе, разработанной для операционной системы NextStep.*

Одно из наиболее интересных и мощных свойств WindowMaker — "док" (специальная панель) в правой части экрана. Существует много приложений, которые выполняются в ней: мониторинг сети, часы, проверка электронной почты или проигрывание компакт-дисков.

После установки нужного менеджера окон необходимо внести изменения в файл с личными настройками **.xinitrc**, чтобы он загружался вместо Gnome.

Для этого необходимо открыть файл **.xinitrc** в текстовом редакторе. Если файла не существует, он будет создан.

В данном случае устанавливается единственная опция — менеджер окон, запускаемый в X-Window, поэтому в файл добавляется лишь одна строка. Для WindowMaker это:

```
wmaker
```

а для Blackbox:

```
blackbox
```

После добавления этой строки необходимо сохранить файл и выйти из редактора. При следующем запуске X-Window, вместо Gnome будет запущен установленный менеджер окон.

Конфигурирование этих систем выходит за пределы данной книги. Если вы хотите узнать больше о Blackbox или WindowMaker, посетите Web-страницы **http://blackbox.alug.org** и **www.windowmaker.org**.

Информацию о многих популярных менеджерах окон, можно получить на Web-сайте **www.xwinman.org**.

Автоматический запуск приложений

Если нужно, чтобы при каждом входе в систему X-Window автоматически запускались какие-либо приложения, их следует добавить в файл **.xinitrc** до менеджера окон и указать в конце строки символ **&**, отвечающий запуску в фоновом режиме. В ином случае они будут запускаться в приоритетном режиме и управление не будет передано менеджеру окон.

Вот пример файла **.xinitrc**, в котором запускается окно X-терминала и X-часы в графической среде Blackbox.

```
xterm &
xclock &
blackbox
```

Установка цвета фона или фонового изображения

Многие менеджеры окон и графические среды наподобие WindowMaker и Gnome позволяют устанавливать цвет фона или фоновое изображение. Более простым средам (например, FVWM и wm2) это недоступно. Существует несколько способов установки цвета фона или фонового изображения посредством вызова внешних программ из файла **.xinitrc**.

Установка цвета фона

По умолчанию в X-Window используется заштрихованный фон, неудобный для глаз.

Программа **xsetroot** позволяет установить в качестве цвета фона или сплошной цвет или bitmap. Все доступные цвета перечислены в текстовом файле **/usr/X11R6/lib/X11/rgb.txt**. Вот пример установки сплошного цвета фона:

```
xsetroot -solid ForestGreen
```

Эту команду можно выполнить в окне X-терминала и цвет фона станет темно-зеленым.

Если цвет необходимо установить стационарно, эту команду следует добавить в файл **.xinitrc** до запуска менеджера окон. Например, ниже устанавливается цвет фона и запускается **twm**:

```
xsetroot -solid ForestGreen &
twm
```

Можно использовать только имя цвета, приведенное в файле **rgb.txt**. Если в имени есть пробел, его необходимо заключить в кавычки.

Кроме установки цвета фона или bitmap-изображения, программа **xsetroot** позволяет изменять курсор мыши. Об этом можно узнать на **man**-странице **xsetroot**.

Создавать собственные изображения для экранного фона или курсоров мыши позволяет специальная программа, показанная на рис. 34.3.

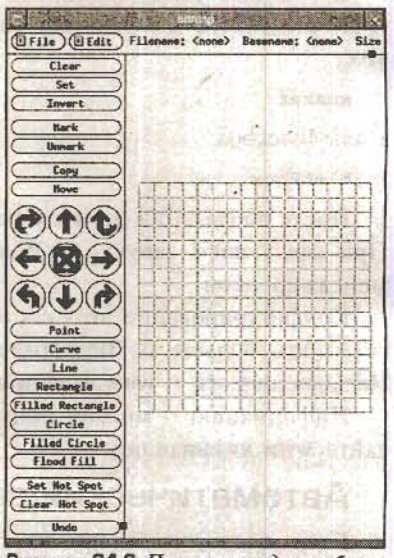


Рисунок 34.3 *Программа для работы с изображениями используется для создания фоновой текстуры и курсоров мыши для последующей установки посредством утилиты* **xsetroot**.

Настройка фонового изображения

В состав X-Window не включена программа, позволяющая устанавливать фоновое изображение. Однако она содержится в наборе портов FreeBSD и называется **xv**. Она находится в каталоге графических приложений. Выполнение программы **xv** в интерактивном режиме показано на рис. 34.4.

xv поддерживает распространенные графические форматы, включая GIF, JPEG и BMP, и позволяет установить любое из подобных изображений как фоновое.

Вот как можно воспользоваться командой **xv** для установки фонового изображения в X-Window:

```
xv -root -quit /home/foobar/images/
myimage.jpg
```

Рисунок **myimage.jpg** загружается в корневое окно (т.е. как фон менеджера окон). Вторая опция указывает **xv**, что после загрузки изображения необходимо завершить работу.

Чтобы фоновое изображение загружалось при каждом запуске X-Window, в файл **.xinitrc** необходимо добавить следующие строки:

```
xv -root -quit /home/foobar/images/myimage.jpg
twm
```

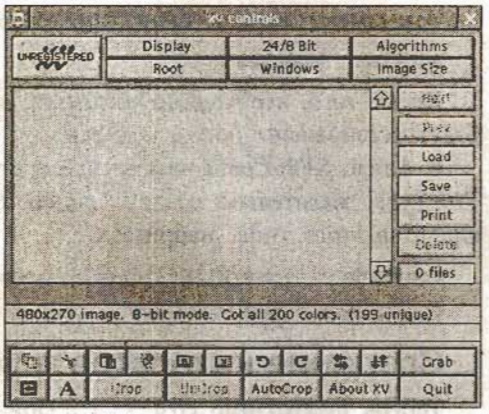


Рисунок 34.4 *xv позволяет не только устанавливать фоновое изображение X-Window. Это полнофункциональная программа просмотра изображений, имеющая интерактивный режим.*

Несколько полезных опций **xv**: **-max** — изменение размеров изображения для заполнения всей области экрана; **-maxspect** — заполнение всей области с сохранением пропорций изображения.

Работа со шрифтами

Рано или поздно вам придется устанавливать дополнительные шрифты, для работы в приложениях X-Window (например, в GIMP).

Система X-Window поддерживает несколько типов шрифтов. Наиболее часто устанавливаются шрифты Adobe Type 1 или TrueType. Они чрезвычайно популярны, поскольку поддерживаются Windows, Macintosh и новыми версиями Xfree86. В Internet можно найти тысячи как свободно распространяемых, так и коммерческих шрифтов Type 1 и TrueType.

В этом разделе обсуждаются вопросы установки шрифтов этих популярных типов.

Проверка файла XF86Config

Первое, что необходимо сделать, — убедиться, что в файл XF86Config включены модули, необходимые для поддержки требуемых шрифтов. О содержимом файла XF86Config подробно рассказано ранее в этой главе в разделе "Опции в файле XF86Config".

В файле **/etc/X11/XF86Config** нужно найти соответствующую строку **Load** в разделе **Modules**. Она имеет следующий вид: для шрифтов Adobe Type 1:

```
Load "type1"
```

для шрифтов TrueType:

```
Load "freetype"
```

Эти строки должны быть расположены в разделе под заголовком:

```
Section "Modules"
```

Кроме того, необходимо убедиться, что в раздел **files** включен каталог, в который будут установлены новые шрифты.

В файле **XF86Config** нужно найти строки **FontPath**. Если устанавливается шрифт TrueType, желательно создать запись **FontPath** с каталогом для размещения всех шрифтов этого типа, например:

```
FontPath   "/usr/X11R6/lib/X11/fonts/TrueType/"
```

После этого файл следует сохранить и выйти из редактора.

Создание каталогов и установка шрифтов

Теперь необходимо создать каталог для шрифтов командой (с правами доступа **root**):

```
mkdir /usr/X11R6/lib/X11/fonts/TrueType
```

После этого в системе будет создан каталог **TrueType**.

В него необходимо скопировать файлы шрифтов, добавляемых в систему. Соответственно, шрифты Type 1 следует копировать в **/usr/X11R6/lib/X11/fonts/Type1**, а TrueType — в **/usr/X11R6/lib/X11/fonts/TrueType**. Права доступа к шрифтам могут ограничиваться лишь чтением, поэтому их можно изменить командой:

```
chmod 444 имя_шрифта
```

(Убедитесь, что в данный момент вы находитесь в нужном каталоге. Об основных командах интерпретатора и навигации по каталогам рассказано в главе 8).

По завершении этих установок следует запустить несколько программ, настраивающих корректную работу шрифтов.

Для шрифтов TrueType потребуется программа **ttmkfdir**. Она включена в набор портов FreeBSD (в каталоге **x11-fonts**). Для шрифтов Adobe Type 1 подобные функции выполняет программа **type1inst**. Ее можно найти в том же каталоге набора портов. О том, как устанавливать программное обеспечение из него, рассказано в главе 15.

Когда все требуемые шрифты скопированы в соответствующий каталог, перейдите в него и запустите одну из следующих команд.

Для шрифтов TrueType:

```
ttmkfdir > fonts.scale
```

Для шрифтов Adobe Type 1:

```
type1inst > fonts.scale
```

После этого осталось запустить лишь одну команду — **mkfontdir**, чтобы новые шрифты распознавались системой и были добавлены в конфигурационный файл шрифтов. Для масштабируемых шрифтов (Type 1 и TrueType) следует указать опцию **-e**. В соответствующем каталоге запустите следующую команду:

```
mkfontdir -e /usr/X11R6/lib/font/encodings
```

После этого нужно перезапустить систему X-Window и новые шрифты будут готовы к использованию.

При каждом добавлении новых шрифтов необходимо повторить команду **ttmkfdir** (или **type1inst**) и **mkfontdir**.

Использование удаленных клиентов X-Window

Структура системы X-Window позволяет запускать приложения на одном хосте, а ввод и вывод принимать (отображать) на другом. Это полезно в том случае, когда на сервере имеются графические приложения, управлять которыми нужно с рабочей станции. Такое применение графической среды похоже на Telnet или SSH, реализующих подобную функциональность для текстовых приложений. Само приложение запускается на сервере, а ввод и вывод пересылаются локальной системе. X-Window расширяет этот подход и на графические приложения. По умолчанию эти возможности отключены, поэтому вначале нужно разрешить их использование.

Чтобы не было путаницы в терминологии, следует отметить, что понятия *клиент* и *сервер* в X-Window имеют обратный смысл. В сетевой терминологии клиентом является хост, запускающий приложения, находящиеся на другом хосте, сервере. В X-Window дело обстоит наоборот: сервер запускается в локальной системе, а клиент находится на удаленном хосте. Фактически, клиентом X-Window является любая программа, выполняющаяся в этой среде. Более того, локальный X-сервер может запустить клиент, расположенный на удаленной системе, которая сама является сервером. Например, можно запустить графический клиент управления базой данных, расположенный в удаленной системе, в локальном X-сервере. В данном случае клиент находится в удаленной системе, которая представляет собой сервер базы данных.

На системе, где отображается вывод и производится ввод данных для удаленного X-Window-приложения, необходимо запустить X-сервер. Однако платформы не обязательно должны быть одного типа. X-сервер можно запустить в системе FreeBSD, Linux, Solaris или Mac OS X. Более того, локальная система не обязательно должна быть UNIX-подобной. Существуют X-серверы для Windows, способные отображать X-приложения, запущенные во FreeBSD. На данный момент большинство из них является коммерческим. Система Xfree86 находится в стадии портирования на Windows, поэтому вскоре бесплатный X-сервер появится и в Windows-системах.

По умолчанию X-сервер не позволяет отображать удаленные приложения на экране локальной системы. Для разрешения этой функциональности существует несколько способов. Наиболее распространенным (и наименее защищенным!) является утилита **xhost**.

Использование программы xhost для отображения вывода удаленных графических приложений

xhost позволяет приложениям, выполняющимся в различных системах, отображаться на локальной системе и воспринимать ввод. Для просмотра текущих настроек необходимо запустить команду **xhost** без аргументов. Вот ее вывод:

```
bash$ xhost
access control enabled, only authorized clients can connect
bash$
```

В данном случае только те удаленные приложения могут подключаться к данной системе, которые внесены в список авторизации (authorization list). Список пуст, т.е. никакие системы не могут подключиться к данной.

Если требуется запускать приложения на хосте **lion**, его нужно добавить в список авторизации. Это достигается командой:

```
bash$ xhost +lion
lion being added to access control list
bash$
```

Теперь вывод команды **xhost** без аргументов выглядит так:

```
access control enabled, only authorized clients can connect
INET:lion.samplenet.org
```

Приложения с хоста **lion** могут отображать свой вывод в X-Window на локальной машине.

Такая настройка позволяет (к сожалению!) каждому, кто имеет учетную запись на машине **lion**, пересылать вывод на локальную систему. Это представляет серьезную проблему с защитой в ситуации, когда не всем пользователям можно доверять. Поэтому в таких средах использовать утилиту **xhost** не следует. Другие (более защищенные методы) кратко обсуждаются далее.

При использовании **xhost** желательно авторизовать хост непосредственно перед тем, как нужен доступ к нему, а затем по окончании сеанса работы с удаленным приложением отключить авторизацию. Для выполнения этого действия применяется следующая команда:

```
bash$ xhost -lion
lion being removed from access control list
bash$
```

После этого вывод приложений с хоста **lion** не сможет отображаться на локальной системе.

Утилита **xhost** позволяет запретить проверку авторизации, в результате чего любые клиенты смогут подключаться к системе. Это серьезная дыра в защите, но, если по каким-либо причинам вам требуется такая функциональность, можно воспользоваться следующей командой:

```
bash$ xhost +
access control disabled, clients can connect from any host
bash$
```

Для включения контроля доступа применяется команда **xhost -**. В ответ система выведет:

```
access control enabled, only authorized clients can connect
```

ПРЕДУПРЕЖДЕНИЕ

Поскольку программа **xhost** представляет серьезную угрозу для защиты системы, необходимо настроить брандмауэр, блокирующий доступ неавторизованных пользователей из внешней сети к портам, которые использует **xhost**: с 6000 по 6063 (см. главу 29).

После того как система позволит отображать вывод удаленных приложений, необходимо соединиться с удаленным хостом и запустить нужную программу. Для это-

го используются утилиты Telnet, SSH или **rlogin** (если удаленная система поддерживает **rlogin** с вашей).

СОВЕТ

Если вы соединяетесь с удаленной системой из X-сервера, работающего под Windows, используйте приложение Telnet или SSH, включенное в состав программного обеспечения сервера. Поскольку встроенная в Windows программа Telnet ничего не знает об X-сервере, она не позволит корректно выполнить процедуры, описанные далее.

Запуск удаленного приложения

Для запуска приложения на удаленном хосте и отображения его вывода на локальной машине необходимо в качестве значения переменной среды **DISPLAY** на удаленном хосте указать адрес локальной системы. Это можно сделать в сеансе Telnet, SSH или **rlogin**. Для запуска приложения из терминального окна используются следующие команды (в этом примере предполагается, что локальный хост называется **simba**):

```
bash$ DISPLAY=simba:0; export DISPLAY
bash$ xcalc &
```

Этот набор команд предназначен для интерпретатора в стиле **bourne** (т.е. **bourne**, **korn** или **bash**). Если используется **tcsh**, необходимо заменить первую строку следующей:

```
% setenv DISPLAY simba:0
```

В результате на удаленном хосте будет запущена программа **xcalc**, а ее вывод вы увидите на экране.

Переменная среды **DISPLAY** поддерживает следующий синтаксис:

```
DISPLAY=имя_хоста:дисплей#:экран#
```

дисплей# почти всегда равен 0, а **экран#**, как правило, можно опустить, если к системе присоединено не более одного экрана.

СОВЕТ

Информацию необходимо вводить в окне Telnet или SSH, подключенном к удаленному хосту. Ввод информации из локального окна **xterm** не даст желаемого эффекта.

Другие способы управления доступом клиентов

Существуют и другие способы управления тем, какие хосты могут направлять вывод приложений на локальный хост. Некоторые из них защищены лучше, чем метод **xhost**, описанный ранее. Однако все эти методы и их настройка выходят за пределы данной книги. Узнать о них подробнее можно на странице справочного руководства **Xsecurity**. (При доступе к ней обратите внимание на учет регистра в названии.)

xdm

xdm — это графический менеджер регистрации в системе X-Window. Когда он запущен вместе с системой, он выводит графическое, а не консольное приглашение для регистрации. Это приглашение похоже на экран регистрации в Windows NT/2000.

Если вы хотите воспользоваться менеджером **xdm**, в начальном каталоге необходимо создать файл **.xsession**. Этот файл является эквивалентом файла **.xinitrc** для **xdm**. Он содержит команды для запуска нужного менеджера окон, а также команды автоматического запуска приложений. Все команды в файле **.xsession** должны запускаться в фоновом режиме (**&**). Вот пример этого файла:

```
wmaker &
xterm &
```

После регистрации система автоматически запустит среду WindowMaker и терминальное приложение **xterm**.

Обратите внимание, что, в отличие от **.xinitrc**, файл **.xsession** должен быть выполняемым, иначе его настройки не будут работать.

Есть несколько методов автоматического запуска **xdm** при загрузке системы. Приведенный здесь является самым надежным, поскольку он позволяет зарегистрироваться на другой виртуальной консоли в текстовом режиме, если вдруг конфигурация X-Window станет некорректной. Безопаснее всего создать следующий файл в каталоге **/usr/local/etc/rc.d**:

```
#!/bin/sh
case "$1" in
      start)
            echo "************************************************"
            echo "* Starting the XDM login manager. Please wait... *"
            echo "************************************************"
            xdm
            ;;
      *)
            : #ничего не выполнять
            ;;
esac
```

Файл необходимо сделать выполняемым и перезагрузить систему. **xdm** будет запущен автоматически.

xdm имеет много возможностей по настройке, однако они выходят за рамки данной книги. За подробной информацией обратитесь к **man**-странице **xdm**.

6 ЧАСТЬ

Приложения

Основные конфигурационные файлы и сценарии FreeBSD

Список поддерживаемого оборудования

Решение проблем с инсталляцией и загрузкой

Источники информации

Справочник по командам

ПРИЛОЖЕНИЕ А

Основные конфигурационные файлы и сценарии FreeBSD

Новичков в мире FreeBSD часто смущает обилие конфигурационных файлов, используемых для ее тонкой настройки. Предлагаем вам краткий путеводитель, включающий основные конфигурационные файлы FreeBSD.

Расположение	*Назначение*
.htaccess	Файл контроля доступа сервера Apache.
/etc/csh.cshrc	Конфигурационный файл **csh** и **tcsh** для настройки среды.
/etc/csh.login	Здесь указывают программы, которые должны запускаться при регистрации в системе.
/etc/csh.login	Глобальные опции конфигурации по умолчанию для **csh** и **tcsh**, действующие для всех пользователей.
/etc/defaults/rc.conf	Этот файл содержит список обязательных заданий для процесса **init**.
/etc/defaults/rc.conf	Копия файла **/etc/rc.conf** с настройками по умолчанию.
/etc/dhclient.conf	Определяет поведение программы **dhclient**.
/etc/diskcheckd.config	Конфигурационный файл демона **diskcheckd**.
/etc/exports	Содержит перечень каталогов для общего доступа через NFS.
/etc/fstab	Позволяет указать системе все устройства, монтируемые на этапе загрузки.
/etc/ftpusers	Позволяет запретить доступ к системе по FTP для конкретного пользователя. Если в файле указана группа, то в доступе будет отказано всем ее членам.
/etc/group	Содержит имя всех групп, их идентификаторы и списки пользователей.
/etc/host.conf	Определяет порядок, в котором производится поиск именованных ресурсов, позволяет задать приоритеты этого поиска и т.д.
/etc/hosts	Позволяет установить соответствие между IP-адресами и именами хостов, в том числе именами-псевдонимами.
/etc/hosts.allow	Позволяет блокировать доступ с определенных хостов к указанным в файле службам системы.
/etc/inetd.conf	Конфигурационный файл суперсервера **inetd**. Демон **inetd** отвечает за прослушивание сетевых портов и запуск соответствующего демона при обнаружении попытки установить соединение с данным портом.
/etc/login.conf	Здесь можно указывать допустимое количество одновременных процессов, максимальное число открытых файлов, особенности работы командных интерпретаторов и др.
/etc/mail//aliases	Позволяет задать псевдонимы, перенаправляющие входящую электронную почту на другие адреса, в файлы или программы.

Расположение	*Назначение*
/etc/mail/access	Позволяет указать правила доступа, применяемые к отдельным хостам, подсетям или группам адресов.
/etc/mail/freebsd.mc	Редактируемый конфигурационный файл программы **sendmail**, после компиляции из него получается файл **sendmail.cf**.
/etc/mail/local-host-names	В этом файле указаны имена хостов, принадлежащих данному серверу (виртуальный хостинг).
/etc/mail/sendmail.cf	Основной конфигурационный файл **sendmail**.
/etc/mail/virtusertable	Позволяет поставить в соответствие почтовым адресам локальные учетные записи, другие адреса или сообщения об ошибках, т. е. наладить переадресацию.
/etc/master.passwd	Текстовая база данных о пользователях (пароль зашифрован с помощью MD5).
/etc/namedb/localhost.rev	Специальный файл зоны; необходим для работы службы DNS.
/etc/netgroup	Содержит информацию о сетевых группах, объединяющих определенные хосты, пользователей или домены.
/etc/netstart	Пользовательский сценарий для автоматического запуска сетевых сервисов.
/etc/nsswitch.conf	Во FreeBSD 5.0. заменяет файл **/etc/host.conf**.
/etc/pam.conf	Файл конфигурации подключаемых модулей аутентификации.
/etc/passwd	Текстовая база данных о пользователях.
/etc/ppp/ip-down	Сценарий, выполняемый **pppd** при разрыве соединения.
/etc/ppp/ip-up	Сценарий, выполняемый демоном **pppd** при установлении соединения.
/etc/ppp/options	Задает опции конфигурации демона **pppd**.
/etc/ppp/ppp.conf	Это файл для настройки пользовательского PPP.
/etc/printcap	Файл, определяющий настройки принтера, фильтры преобразования, местонахождение спула и т.д.
/etc/profile	Глобальные опции конфигурации по умолчанию для командных интерпретаторов типа Bourne (**sh**, **ksh**, **bash**), действующие для всех пользователей.
/etc/protocols	Содержит сведения о различных сетевых протоколах.
/etc/rc	Главный сценарий конфигурации.
/etc/rc.atm	Запускается сценарием **rc.network**; устанавливает ATM-службы для машин WAN.
/etc/rc.conf	Главный системный конфигурационный файл, содержит опции начального запуска FreeBSD, отменяет настройки по умолчанию из файла **/etc/defaults/rc.conf**.
/etc/rc.devfs	Конфигурирует файловую систему устройств.

Расположение	*Назначение*
/etc/rc.diskless1 **/etc/rc.diskless2**	Эти файлы необходимы при бездисковой загрузке по протоколу BOOTP.
/etc/rc.firewall	Служит для конфигурирования брандмауэра **ipfw**.
/etc/rc.firewall6	Запускается сценарием **rc.network6**; конфигурирует брандмауэр **ip6fw**.
/etc/rc.i386	Установки, специфичные для платформы x86.
/etc/rc.local	Этот файл ныне устарел. Применяйте **rc.d**!
/etc/rc.network	Системный сценарий, выполняющийся во время загрузки. Содержит символические имена подсетей.
/etc/rc.network6	Совпадает со сценарием **rc.network**, добавлена поддержка IPv6.
/etc/rc.pccard	Запускает демон PC-карт для ноутбуков.
/etc/rc.serial	Настраивает терминалы и другие последовательные устройства.
/etc/rc.shutdown	Этот сценарий запускается процессом **init** на этапе останова системы.
/etc/rc.suspend	Сценарий, используемый демоном управления питанием APM.
/etc/rc.sysctl	Устанавливает переменные ядра.
/etc/resolv.conf	Файл, в котором хранится список серверов имен.
/etc/rpc	Содержит информацию о сервисах NFS и NIS, их портах и пр.
/etc/services	База данных, упорядоченная по типам существующих IP-сервисов (с привязкой каждого сервиса к конкретному порту).
/etc/shells	Файл, хранящий перечень разрешенных командных интерпретаторов.
/etc/skey.accesses	Файл, оговаривающий условия регистрации в системе с использованием системы S/Key (одноразовые пароли).
/etc/syslog.conf	Конфигурационный файл, управляющий поведением системы ведения журналов.
/etc/ttys	Этот файл содержит терминальные настройки.
/etc/X11/XF86Config	Главный конфигурационный файл системы X-Window.
/usr/local/etc/apache/access.conf **/usr/local/etc/apache/srm.conf**	В этих файлах ранее содержались определенные настройки, которые теперь включены в **httpd.conf**.
/usr/local/etc/apache/httpd.conf	Это главный конфигурационный файл сервера Apache.
/usr/local/etc/rc.d/	/usr/local/X11R6/etc/rc.d/ Дерево каталогов, содержащее новые сценарии, добавляемые пользователем или инсталлируемые программами.

Расположение	*Назначение*
/var/run/nologin	Размещение этого файла в каталоге **/var/run/** позволяет полностью отключить регистрацию пользователей по FTP.
~/.bash_logout	Индивидуальный файл пользователя, выполняемый при выходе из оболочки. В нем указывают процессы, которые должны запускаться при выходе из системы.
~/.bash_profile	Персональный инициализационный файл интерпретатора, выполняемый при регистрации.
~/.bashrc	Конфигурационный файл интерпретатора **bash**, который используется, если **bash** запускается не как начальный командный интерпретатор.
~/.cshrc	Конфигурационный файл, используемый команжными интерпретаторами **csh** и **tcsh**. Оказывает влияние не только на начальный интерпретатор, но и на все последующие, запущенные из него.
~/.forward	Управляет процессом пересылки (**forwarding**) почтовых сообщений конкретного пользователя.
~/.login	Конфигурационный файл **csh** и **tcsh**. Здесь пользователь указывает программы, которые должны запускаться при его регистрации в системе.
~/.profile	Конфигурационный файл пользователя для интерпретаторов в стиле Bourne (**sh**, **ksh** и **bash**).
~/.xinitrc	Этот файл позволяет управлять настройками X-Window.

ПРИЛОЖЕНИЕ Б

Список поддерживаемого оборудования

Требования к системе ▶

Поддерживаемое оборудование ▶

Видеокарты, поддерживаемые системой X-Window ▶

В этом приложении перечислено оборудование, с которым работает FreeBSD, а также видеокарты, поддерживаемые системой X-Window.

Требования к системе

Минимальные:

- процессор Intel 386sx или совместимый с ним
- 4 Мб оперативной памяти

Минимальная конфигурация, рекомендуемая авторами:

- процессор Intel 486DX2/66 или выше
- 32 Мб оперативной памяти
- Жесткий диск объемом не менее 1 Гб для полной инсталляции (включая X-Window) плюс сторонние приложения (для рабочей станции). Для сервера может потребоваться больший объем, все зависит от количества пользователей и поддерживаемых служб.

Если планируется использовать систему X-Window, рекомендуется следующая минимальная конфигурация:

- Монитор с разрешением до 1024x768
- Видеокарта с поддержкой SVGA (см. список) и не менее чем 1 Мб видеопамяти
- Трехкнопочная мышь

Поддерживаемое оборудование

Ниже приведен список оборудования, которое поддерживается FreeBSD.

Дисковые контроллеры (не SCSI)

Любые стандартные контроллеры типа MFM или RLL (WD1003)

Любые стандартные контроллеры с интерфейсом IDE (WD1007)

Контроллеры ATA

FreeBSD поддерживает режим UDMA на EIDE-контроллерах, которые способны его использовать.

Дисковые SCSI-контроллеры

Обратите внимание на то, что некоторые SCSI-контроллеры (например, SoundBlaster SCSI-контроллеры) используют чипсет другого производителя, например Adaptec. Если в списке отсутствует исходная карта, определите, какой чипсет используется в ней. Ваш чипсет может присутствовать в следующем списке.

Adaptec:

EISA SCSI-контроллеры серии 174X поддерживаются в стандартном и расширенном режиме

EISA/VLB/PCI SCSI-контроллеры серии 274X/284X/2920C/294X/2950/3940/3950 (Narrow/Wide/Twin)

Встроенные SCSI-контроллеры AIC-7850, AIC-7860, AIC-7880, AIC-789X

ISA SCSI-контроллеры серии 1510 (не для загрузочных устройств)

ISA SCSI-контроллеры серии 152X

Материнские платы на основе AIC-6260 и AIC-6360, включающие SCSI-карты AHA-152X и SoundBlaster

AdvanSys:

Поддерживаются все SCSI-контроллеры AdvanSys.

AMI:

Поддерживаются все дисковые контроллеры AMI FastDisk, которые относятся к линейке BusLogic MultiMaster

BusLogic:

Хост-адаптеры MultiMaster серии "W", включая BT-948, BT-958, BT-9580

Хост-адаптеры MultiMaster серии "C", включая BT-946C, BT-956C, BT-956CD, BT-445C, BT-747C, BT-757C, BT-757CD, BT-545C, BT-540CF

Хост-адаптеры MultiMaster серии "S", включая BT-445S, BT-747S, BT-747D, BT-757S, BT-757D, BT-545S, BT-542D, BT-742A, BT-542B

Хост-адаптеры MultiMaster серии "A", включая BT-742A, BT-542B

Compaq:

Контроллеры Intelligent Disk Array: IDA, IDA-2, IAES, SMART, SMART-2/E, Smart-2/P, SMART-2SL; Integrated Array; and Smart Arrays 3200, 3100ES, 221, 4200, 4200, 4250ES

DPT:

Поддерживаются контроллеры SmartCACHE Plus, SmartCACHE III, SmartRAID III, SmartCACHE IV и SmartRAID IV SCSI/RAID. DPT SmartRAID/CACHE V пока не поддерживается. Кроме того, имеется поддержка SCSI RAID-контроллера DPT PM3754U2-16M.

DTC:

EISA SCSI-контроллер в режиме определения 1542

SymBios (также NCR):

PCI SCSI-контроллеры 53C810, 53C810a, 53C815, 53C820, 53C825a, 53C860, 53C875, 53C875j, 53C885 и 53C896, включая ASUS SC-200; Data Technology DTC3130 (все варианты); Diamond FirePort (все); NCR cards (все); SymBios cards (все); Tekram DC390W, 390U и 390F; Tyan S1365

QLogic:

SCSI- и Fiber-контроллеры 1020, 1040, 1040B и 2100

ПРИМЕЧАНИЕ

FreeBSD поддерживает устройства с интерфейсом SCSI-I и SCSI-II. Однако устройства CD-RW и WORM-драйвер, включенный в FreeBSD, поддерживает в режиме только для чтения. Чтобы получить доступ к записи на эти устройства, воспользуйтесь утилитой **cdrecord**, входящей в набор портов FreeBSD.

Приводы CD-ROM

Любой привод, совместимый с интерфейсом ATAPI

SCSI-приводы

Приводы Matsushita/Panasonic (Creative Labs SoundBlaster) (модели 562/563)
Все приводы Sony

Сетевые карты

Adaptec:

Duralink PCI-адаптеры Fast Ethernet на основе чипсета Adaptec AIC-6195 Fast Ethernet, включая следующие:

Адаптер ANA-62011 64-бит, один порт, 10/100baseTX
Адаптер ANA-62022 64-бит, два порта, 10/100baseTX
Адаптер ANA-62044 64-бит, четыре порта, 10/100baseTX
Адаптер ANA-69011 32-бит, один порт, 10/100baseTX
Адаптер ANA-62020 64-бит, один порт, 100baseFX

Allied-Telesyn:

AT1700 и RE2000

Alteon Networks:

Сетевые карты Alteon Networks PCI Gigabit Ethernet на основе чипсетов Tigon 1 и Tigon 2, включая Alteon AceNIC (Tigon 1 и 2), 3Com 3c985-SX (Tigon 1 и 2), Netgear GA620 (Tigon 2), Silicon Graphics Gigabit Ethernet, DEC/Compaq EtherWORKS 1000, NEC Gigabit Ethernet

AMD:

PCnet/PCI (79c970 и 53c974 или 79c974)

RealTek:

Сетевые карты 8129/8139 Fast Ethernet, включая следующие:

Allied-Telesyn AT2550
Allied-Telesyn AT2500TX
Genius GF100TXR (RTL8139)
NDC Communications NE100TX-E
OvisLink LEF-8129TX
OvisLink LEF-8139TX
Netronix Inc. EA-1210 NetEther 10/100
KTX-9130TX 10/100 Fast Ethernet
Accton "Cheetah" EN1207D (MPX 5030/5038; клон RealTek 8139)
SMC EZ Card 10/100 PCI 1211-TX

Lite-On:

Сетевые карты 98713, 98713A, 98715 и 98725 Fast Ethernet, включая следующие:

LinkSys EtherFast LNE100TX
NetGear FA310-TX Rev. D1
Matrox FastNIC 10/100
Kingston KNE110TX

Macronix:

Сетевые карты 98713, 98713A, 98715, 98715A и 98725 Fast Ethernet, включая следующие:

NDC Communications SFA100A (98713A)

CNet Pro120A (98713 or 98713A)

CNet Pro120B (98715)

SVEC PN102TX (98713)

Macronix/Lite-On:

Сетевые карты PNIC II LC82C115 Fast Ethernet, включая LinkSys EtherFast LNE100TX версии 2

Winbond:

Сетевые карты W89C840F Fast Ethernet, включая Trendware TE100-PCIE

Via Technologies:

Сетевые карты VT3043 "Rhine I" и VT86C100A "Rhine II" Fast Ethernet, включая Hawking Technologies PN102TX и D-Link DFE-530TX

Silicon Integrated Systems:

Сетевые карты SiS 900 и SiS 7016 PCI Fast Ethernet

Sundance Technologies:

Сетевые карты ST201 PCI Fast Ethernet, включая D-Link DFE-550TX

SysKonnect:

Карты SK-984x PCI Gigabit Ethernet, включая следующие:

SK-9841 1000baseLX (оптоволоконный, однорежимный, одиночный порт)

SK-9842 1000baseSX (оптоволоконный, многорежимный, одиночный порт)

SK-9843 1000baseLX (оптоволоконный, однорежимный, двойной порт)

SK-9844 1000baseSX (оптоволоконный, многорежимный, двойной порт).

Texas Instruments:

Сетевые карты ThunderLAN PCI, включая следующие:

Compaq Netelligent 10, 10/100, 10/100 Proliant, 10/100 Dual-Port, 10/100 TX Embedded UTP, 10 T PCI UTP/Coax, and 10/100 TX UTP

Compaq NetFlex 3P, 3P Integrated и 3P w/BNC

Olicom OC-2135/2138, OC-2325, OC-2326 10/100 TX UTP

Карты Racore 8165 10/100baseTX и 8148 10baseT/100baseTX/100baseFX

ADMTek:

Сетевые карты PCI Fast Ethernet на основе AL981 и AN985

ASIX Electronics:

Сетевые карты PCI AX88140A, включая Alfa Inc. GFC2204 и CNet Pro110B

DEC:

Сетевые карты EtherWORKS III (DE203, DE204 и DE205)

Сетевые карты EtherWORKS II (DE200, DE201, DE202 и DE422)

Сетевые карты на основе DC21040, DC21041 и DC21140 (SMC Etherpower 8432T, DE245 и т.д.)

Сетевые карты FDDI (DEFPA/DEFEA)

Efficient:

ENI-155p ATM PCI

FORE:

PCA-200E ATM PCI

Fujitsu:

MB86960A/MB86965A

HP:

Карты PC Lan+ (номера моделей: 27247B и 27252A)

Intel:

EtherExpress ISA (не рекомендуется из-за неустойчивости драйвера)

EtherExpress Pro/10

EtherExpress Pro/100B PCI Fast Ethernet

Isolan:

AT 4141-0 (16-бит)

Isolink:

4110 (8-бит)

Novell:

NE1000, NE2000 и NE2100

Сетевые карты PCI, совместимые с NE2000, включая следующие:

- RealTek 8029
- NetVin 5000
- Winbond W89C940
- Surecom NE-34
- VIA VT86C926

3Com:

3C501

3C503 Etherlink II

3C505 Etherlink/+

3C507 Etherlink 16/TP

3C509

3C579

3C589 (PCMCIA)

3C590/592/595/900/905/905B/905C PCI and EISA (Fast) Etherlink III / (Fast) Etherlink XL

Серверный адаптер 3C980/3C980B Fast Etherlink XL
Адаптер 3CSOHO100-TX OfficeConnect

Toshiba:

Все Ethernet-карты компании Toshiba

Карты PCMCIA:

PCMCIA Ethernet-карты IBM и National Semiconductor

Устройства USB

USB-клавиатура
USB-мышь
USB-принтеры и кабели преобразования USB — параллельный принтер
USB-концентраторы

Звуковые устройства

16550 UART (Midi) (поддержка на экспериментальном уровне, требует применения определенных приемов при установке, есть документация)

Advance Asound 100, 110 и Logic ALS120

Звуковые карты на основе Aureal Vortex1/Vortex2 и Vortex Advantage поддерживаются сторонним драйвером

Creative Labs SB16, SB32, SB AWE64 (включая Gold), Vibra16, SB PCI (экспериментальная поддержка), SB Live! (экспериментальная поддержка) и большинство звуковых карт, совместимых с SoundBlaster

Creative Labs SB Midi Port (экспериментальная поддержка), SB OPL3 Synthesizer (экспериментальная поддержка)

Crystal Semiconductor CS461x/462x Audio Accelerator; поддержка Midi-порта в CS461x — экспериментальная

Аудиоконтроллер Crystal Semiconductor CS428x
CS4237, CS4236, CS4232, CS4231 (ISA)
ENSONIQ AudioPCI ES1370/1371
ESS ES1868, ES1869, ES1879, ES1888
Gravis UltraSound PnP, MAX
NeoMagic 256AV/ZX (PCI)
OPTi931 (ISA)
Midi-секвенсор, совместимый с OSS (экспериментальная поддержка)
Trident 4DWave DX/NX (PCI)
Yahama OPL-SAx (ISA)

Различные устройства

4-портовая последовательная карта AST, с использованием разделенного IRQ
8-портовая последовательная карта ARNET, с использованием разделенного IRQ
Высокоскоростная последовательная карта ARNET (теперь Digiboard) Sync 570/i
4-портовая последовательная карта Boca BB1004 (модемы не поддерживаются)
6-портовая последовательная карта Boca IOAT66 (модемы поддерживаются)
8-портовая последовательная карта Boca BB1008 (модемы не поддерживаются)

16-портовая последовательная карта Boca BB2016 (модемы поддерживаются)
Последовательная карта Cyclades Cyclom-y
4-портовая последовательная карта Moxa SmartIO CI-104J
4-портовая карта STB, с использованием разделенного IRQ
Последовательная карта SDL Communications RISCom/8
Высокоскоростные синхронные последовательные карты SDL Communications RISCom/N2 и N2pci
Многопортовые карты Specialix SI/XIO/SX с хостовыми картами старой (SIHOST2.x) и новой (на основе транспьютера, называемые JET) версий; поддерживаются шины ISA, EISA и PCI
Многопортовые последовательные карты: EasyIO, EasyConnection 8/32 & 8/64, ONboard 4/16 и Brumby
Connectix QuickCam
Устройство перехвата видеофреймов Matrox Meteor Video
Устройство перехвата видеофреймов Creative Labs Video Spigot
перехвата видеофреймов Cortex1
Различные устройства перехвата видеофреймов на основе чипов Brooktree Bt848 и Bt878
Приводы CD-R HP4020, HP6020, Philips CDD2000/CDD2660 и Plasmon
Bus-мышь
Мышь PS/2
Стандартный джойстик
Контроллеры управления питанием X-10
Приводы GPIB и Transputer
Ручные сканеры Genius и Mustek
Носители на магнитной ленте (только некоторые из старых моделей; достаточно устаревший драйвер)
PCMCIA- и ISA-адаптеры беспроводной сети Lucent Technologies WaveLAN/IEEE 802.11 стандартного (2 Мбит/с) и турбо-режима (6 Мбит/с), а также подобные им (NCR WaveLAN/IEEE 802.11, Cabletron RoamAbout 802.11 DS)

Видеокарты, поддерживаемые системой X-Window

Ниже приведен список видеокарт, поддерживаемых системой XFree86 на текущий момент. Список взят с Web-сайта проекта XFree86. Если ваша видеокарта отсутствует в этом списке, определите, какой чипсет она использует, и поищите его.

2 the Max MAXColor S3 Trio64V+
3DLabs Oxygen GMX
928Movie
AGX (generic)
ALG-5434
ASUS 3Dexplorer
ASUS PCI-AV264CT

ASUS PCI-V264CT
ASUS Video Magic PCI V864
ASUS Video Magic PCI VT64
AT25
AT3D
ATI 3D Pro Turbo
ATI 3D Pro Turbo PC2TV
ATI 3D Xpression
ATI 3D Xpression+
ATI 3D Xpression+ PC2TV
ATI 8514 Ultra (без VGA)
ATI All-in-Wonder
ATI All-in-Wonder Pro
ATI Graphics Pro Turbo
ATI Graphics Pro Turbo 1600
ATI Graphics Pro Turbo с чипсетом AT&T 20C408 RAMDAC
ATI Graphics Pro Turbo с чипсетом ATI68860 RAMDAC
ATI Graphics Pro Turbo с чипсетом ATI68860B RAMDAC
ATI Graphics Pro Turbo с чипсетом ATI68860C RAMDAC
ATI Graphics Pro Turbo с чипсетом ATI68875 RAMDAC
ATI Graphics Pro Turbo с чипсетом CH8398 RAMDAC
ATI Graphics Pro Turbo с чипсетом STG1702 RAMDAC
ATI Graphics Pro Turbo с чипсетом STG1703 RAMDAC
ATI Graphics Pro Turbo с чипсетом TLC34075 RAMDAC
ATI Graphics Ultra
ATI Graphics Ultra Pro
ATI Graphics Xpression
ATI Graphics Xpression с чипсетом ATI68860 RAMDAC
ATI Graphics Xpression с чипсетом ATI68860B RAMDAC
ATI Graphics Xpression с чипсетом ATI68860C RAMDAC
ATI Graphics Xpression с чипсетом ATI68875 RAMDAC
ATI Graphics Xpression с чипсетом AT&T 20C408 RAMDAC
ATI Graphics Xpression с чипсетом CH8398 RAMDAC
ATI Graphics Xpression с чипсетом Mach64 CT (264CT)
ATI Graphics Xpression с чипсетом STG1702 RAMDAC
ATI Graphics Xpression с чипсетом STG1703 RAMDAC
ATI Graphics Xpression с чипсетом TLC34075 RAMDAC
ATI Mach32
ATI Mach64
ATI Mach64 3D RAGE II
ATI Mach64 3D RAGE II+DVD
ATI Mach64 3D Rage IIC
ATI Mach64 3D Rage Pro
ATI Mach64 CT (264CT), Internal RAMDAC

ATI Mach64 GT (264GT), известный также как 3D RAGE, Int
ATI Mach64 VT (264VT), Internal RAMDAC
ATI Mach64 с чипсетом AT&T 20C408 RAMDAC
ATI Mach64 с чипсетом ATI68860 RAMDAC
ATI Mach64 с чипсетом ATI68860B RAMDAC
ATI Mach64 с чипсетом ATI68860C RAMDAC
ATI Mach64 с чипсетом ATI68875 RAMDAC
ATI Mach64 с чипсетом CH8398 RAMDAC
ATI Mach64 с чипсетом IBM RGB514 RAMDAC
ATI Mach64 с чипсетом Internal RAMDAC
ATI Mach64 с чипсетом STG1702 RAMDAC
ATI Mach64 с чипсетом STG1703 RAMDAC
ATI Mach64 с чипсетом TLC34075 RAMDAC
ATI Pro Turbo+PC2TV, 3D Rage II+DVD
ATI Ultra Plus
ATI Video Xpression
ATI Video Xpression+
ATI WinBoost
ATI Win Boost с чипсетом AT&T 20C408 RAMDAC
ATI WinBoost с чипсетом ATI68860 RAMDAC
ATI WinBoost с чипсетом ATI68860B RAMDAC
ATI WinBoost с чипсетом ATI68860C RAMDAC
ATI WinBoost с чипсетом ATI68875 RAMDAC
ATI WinBoost с чипсетом CH8398 RAMDAC
ATI WinBoost с чипсетом Mach64 CT (264CT)
ATI WinBoost с чипсетом STG1702 RAMDAC
ATI WinBoost с чипсетом STG1703 RAMDAC
ATI WinBoost с чипсетом TLC34075 RAMDAC
ATI WinCharger
ATI WinCharger с чипсетом AT&T 20C408 RAMDAC
ATI WinCharger с чипсетом ATI68860 RAMDAC
ATI WinCharger с чипсетом ATI68860B RAMDAC
ATI WinCharger с чипсетом ATI68860C RAMDAC
ATI WinCharger с чипсетом ATI68875 RAMDAC
ATI WinCharger с чипсетом CH8398 RAMDAC
ATI WinCharger с чипсетом Mach64 CT (264CT)
ATI WinCharger с чипсетом STG1702 RAMDAC
ATI WinCharger с чипсетом STG1703 RAMDAC
ATI WinCharger с чипсетом TLC34075 RAMDAC
ATI Win Turbo
ATI WinTurbo с чипсетом AT&T 20C408 RAMDAC
ATI WinTurbo с чипсетом ATI68860 RAMDAC
ATI WinTurbo с чипсетом ATI68860B RAMDAC
ATI WinTurbo с чипсетом ATI68860C RAMDAC

ATI WinTurbo с чипсетом ATI68875 RAMDAC
ATI WinTurbo с чипсетом CH8398 RAMDAC
ATI WinTurbo с чипсетом Mach64 CT (264CT)
ATI WinTurbo с чипсетом STG1702 RAMDAC
ATI WinTurbo с чипсетом STG1703 RAMDAC
ATI WinTurbo с чипсетом TLC34075 RAMDAC
ATI Wonder SVGA
ATI Xpert 98
ATI Xpert XL
ATI Xpert@Play PCI and AGP, 3D Rage Pro
ATI Xpert@Play 98
ATI Xpert@Work, 3D Rage Pro
Карты ATI, интегрированные на материнской плате Intel Maui MU440EX
ATrend ATC-2165A
AccelStar Permedia II AGP
Actix GE32+ 2MB
Actix GE32i
Actix GE64
Actix ProStar
Actix ProStar 64
Actix Ultra
Acumos AVGA3
Alliance ProMotion 6422
Ark Logic ARK1000PV (generic)
Ark Logic ARK1000VL (generic)
Ark Logic ARK2000MT (generic)
Ark Logic ARK2000PV (generic)
Avance Logic 2101
Avance Logic 2228
Avance Logic 2301
Avance Logic 2302
Avance Logic 2308
Avance Logic 2401
Binar Graphics AnyView
Boca Vortex (Sierra RAMDAC)
COMPAQ Armada 7380DMT
COMPAQ Armada 7730MT
California Graphics SunTracer 6000
Canopus Co
Canopus Total-3D
Cardex Challenger (Pro)
Cardex Cobra
Cardex Trio64
Cardex Trio64Pro

Chips & Technologies CT64200
Chips & Technologies CT64300
Chips & Technologies CT65520
Chips & Technologies CT65525
Chips & Technologies CT65530
Chips & Technologies CT65535
Chips & Technologies CT65540
Chips & Technologies CT65545
Chips & Technologies CT65546
Chips & Technologies CT65548
Chips & Technologies CT65550
Chips & Technologies CT65554
Chips & Technologies CT65555
Chips & Technologies CT68554
Chips & Technologies CT69000
Cirrus Logic GD542x
Cirrus Logic GD543x
Cirrus Logic GD5446 (noname card) 1MB upg
Cirrus Logic GD544x
Cirrus Logic GD5462
Cirrus Logic GD5464
Cirrus Logic GD5465
Cirrus Logic GD5480
Cirrus Logic GD62xx (laptop)
Cirrus Logic GD64xx (laptop)
Cirrus Logic GD754x (laptop)
Colorgraphic Dual Lightning
Creative Blaster Exxtreme
Creative Labs 3D Blaster PCI (Verite 1000)
Creative Labs Graphics Blaster 3D
Creative Labs Graphics Blaster Eclipse (OEM Model CT6510) XF86_SVGA
Creative Labs Graphics Blaster MA201
Creative Labs Graphics Blaster MA202
Creative Labs Graphics Blaster MA302
Creative Labs Graphics Blaster MA334
DFI-WG1000
DFI-WG5000
DFI-WG6000
DSV3325
DSV3326
DataExpert DSV3325
DataExpert DSV3365
Dell S3 805
Dell onboard ET4000

Diamond Edge 3D
Diamond Fire GL 1000
Diamond Fire GL 1000 PRO
Diamond Fire GL 3000
Diamond Multimedia Stealth 3D 2000
Diamond Multimedia Stealth 3D 2000 PRO
Diamond SpeedStar (Plus)
Diamond SpeedStar 24
Diamond SpeedStar 24X (поддерживается не полностью)
Diamond SpeedStar 64
Diamond SpeedStar A50
Diamond SpeedStar HiColor
Diamond SpeedStar Pro (не SE)
Diamond SpeedStar Pro 1100
Diamond SpeedStar Pro SE (CL-GD5430/5434)
Diamond SpeedStar64 Graphics 2000/2200
Diamond Stealth 24
Diamond Stealth 32
Diamond Stealth 3D 2000
Diamond Stealth 3D 2000 PRO
Diamond Stealth 3D 3000
Diamond Stealth 3D 4000
Diamond Stealth 64 DRAM SE
Diamond Stealth 64 DRAM с чипсетом S3 SDAC
Diamond Stealth 64 DRAM с чипсетом S3 Trio64
Diamond Stealth 64 VRAM
Diamond Stealth 64 Vidco VRAM (TI RAMDAC)
Diamond Stealth II S220
Diamond Stealth Pro
Diamond Stealth VRAM
Diamond Stealth Video 2500
Diamond Stealth Video DRAM
Diamond Stealth64 серии Graphics 2001
Diamond Stealth64 серии Graphics 2xx0 (864 + SDAC)
Diamond Stealth64 серии Graphics 2xx0 (Trio64)
Diamond Stealth64 серии Video 2001 (2121/2201)
Diamond Stealth64 Video 2120/2200
Diamond Stealth64 Video 3200
Diamond Stealth64 Video 3240/3400 (IBM RAMDAC)
Diamond Stealth64 Video 3240/3400 (TI RAMDAC)
Diamond Viper 330
Diamond Viper 550
Diamond Viper PCI 2MB
Diamond Viper Pro Video

Diamond Viper VLB 2MB
Digital 24-plane TGA (ZLXp-E2)
Digital 24-plane+3D TGA (ZLXp-E3)
Digital 8-plane TGA (UDB/Multia)
Digital 8-plane TGA (ZLXp-E1)
EIZO (VRAM)
ELSA ERAZOR II
ELSA GLoria Synergy
ELSA GLoria-L
ELSA GLoria-L/MX
ELSA GLoria-S
ELSA GLoria-XL
ELSA Gloria-4
ELSA Gloria-8
ELSA VICTORY ERAZOR
ELSA Victory 3D
ELSA Victory 3DX
ELSA WINNER 1000/T2D
ELSA Winner 1000 R3D
ELSA Winner 1000AVI (версия AT&T 20C409)
ELSA Winner 1000AVI (версия SDAC)
ELSA Winner 1000ISA
ELSA Winner 1000PRO с чипсетом S3 SDAC
ELSA Winner 1000PRO с чипсетом STG1700 или AT&T RAMDAC
ELSA Winner 1000PRO/X
ELSA Winner 1000TRIO
ELSA Winner 1000TRIO/V
ELSA Winner 1000TwinBus
ELSA Winner 1000VL
ELSA Winner 2000
ELSA Winner 2000/Office
ELSA Winner 2000AVI
ELSA Winner 2000AVI/3D
ELSA Winner 2000PRO-2
ELSA Winner 2000PRO-4
ELSA Winner 2000PRO/X-2
ELSA Winner 2000PRO/X-4
ELSA Winner 2000PRO/X-8
ELSA Winner 3000
ELSA Winner 3000-L-42
ELSA Winner 3000-M-22
ELSA Winner 3000-S
EPSON CardPC (встроенная)
ET3000 (generic)

ET4000 (generic)
ET4000 W32i, W32p (generic)
ET4000/W32 (generic)
ET6000 (generic)
ET6100 (generic)
ExpertColor DSV3325
ExpertColor DSV3365
Карты, совместимые со стандартом VGA
Genoa 5400
Genoa 8500VL(-28)
Genoa 8900 Phantom 32i
Genoa Phantom 64i с чипсетом S3 SDAC
Genoa VideoBlitz III AV
Hercules Dynamite
Hercules Dynamite 128/Video
Hercules Dynamite Power
Hercules Dynamite Pro
Hercules Graphite HG210
Hercules Graphite Power
Hercules Graphite Pro
Hercules Graphite Terminator 64
Hercules Graphite Terminator 64/DRAM
Hercules Graphite Terminator Pro 64
Hercules Stingray
Hercules Stingray 128 3D
Hercules Stingray 64/V с чипсетом ICS5342
Hercules Stingray 64/V с чипсетом ZoomDAC
Hercules Stingray Pro
Hercules Stingray Pro/V
Hercules Terminator 3D/DX
Hercules Terminator 64/3D
Hercules Terminator 64/Video
Hercules Thriller3D
Integral FlashPoint
Intel 5430
Interay PMC Viper
JAX 8241
Jaton Video-58P
Jaton Video-70P
Jazz Multimedia G-Force 128
LeadTek WinFast 3D S600
LeadTek WinFast 3D S680
LeadTek WinFast S200
LeadTek WinFast S430

LeadTek WinFast S510
Leadtek WinFast 2300
MELCO WGP-VG4S
MELCO WGP-VX8
MSI MS-4417
Matrox Comet
Matrox Marvel II
Matrox Millennium 2/4/8MB
Matrox Millennium (MGA)
Matrox Millennium G200 4/8/16MB
Matrox Millennium G200 SD 4/8/16MB
Matrox Millennium II 4/8/16MB
Matrox Millennium II AGP
Matrox Mystique
Matrox Mystique G200 4/8/16MB
Matrox Productiva G100 4/8MB
MediaGX
MediaVision Proaxcel 128
Mirage Z-128
Miro Crystal 10SD с чипсетом GenDAC
Miro Crystal 12SD
Miro Crystal 16S
Miro Crystal 20SD PCI с чипсетом S3 SDAC
Miro Crystal 20SD VLB с чипсетом S3 SDAC (BIOS 3)
Miro Crystal 20SD с чипсетом ICD2061A (BIOS 2)
Miro Crystal 20SD с чипсетом ICS2494 (BIOS 1)
Miro Crystal 20SV
Miro Crystal 22SD
Miro Crystal 40SV
Miro Crystal 80SV
Miro Crystal 8S
Miro Crystal DVD
Miro miroCRYSTAL VRX
Miro miroMedia 3D
Miro MiroVideo 20TD
Miro Video 20SV
Neomagic
Number Nine FX Motion 331
Number Nine FX Motion 332
Number Nine FX Motion 531
Number Nine FX Motion 771
Number Nine FX Vision 330
Number Nine GXE Level 10/11/12
Number Nine GXE Level 14/16

Number Nine GXE64
Number Nine GXE64 Pro
Number Nine GXE64 с чипсетом S3 Trio64
Number Nine Imagine I-128 (2-8MB)
Number Nine Imagine I-128 Series 2 (2-4MB)
Number Nine Imagine-128-T2R
Number Nine Revolution 3D AGP (4-8MB SGRAM)
Number Nine Visual 9FX Reality 332
Oak 87 ISA (generic)
Oak 87 VLB (generic)
Oak ISA Card (generic)
Ocean (octek) VL-VGA-1000
Octek AVGA-20
Octek Combo-26
Octek Combo-28
Octek VL-VGA-26
Octek VL-VGA-28
Orchid Celsius (AT&T RAMDAC)
Orchid Celsius (Sierra RAMDAC)
Orchid Fahrenheit 1280
Orchid Fahrenheit VA
Orchid Fahrenheit-1280+
Orchid Kelvin 64
Orchid Kelvin 64 VLB Rev A
Orchid Kelvin 64 VLB Rev B
Orchid P9000 VLB
Orchid Technology Fahrenheit Video 3D
PC-Chips M567 Mainboard
Paradise Accelerator Value
Paradise/WD 90CXX
PixelView Combo TV 3D AGP (Prolink)
PixelView Combo TV Pro (Prolink)
RIVA TNT
RIVA128
Rendition Verite 1000
Rendition Verite 2x00
Revolution 3D (T2R)
S3 801/805 (generic)
S3 801/805 с чипсетом ATT20c490 RAMDAC
S3 801/805 с чипсетом ATT20c490 RAMDAC и ICD2061A
S3 801/805 с чипсетом Chrontel 8391
S3 801/805 с чипсетом S3 GenDAC
S3 801/805 с чипсетом SC1148{ 2,3,4} RAMDAC
S3 801/805 с чипсетом SC1148{ 5,7,9} RAMDAC

S3 864 (generic)
S3 864 с чипсетом ATT 20C498 или 21C498
S3 864 с чипсетом SDAC (86C716)
S3 864 с чипсетом STG1703
S3 868 (generic)
S3 868 с чипсетом ATT 20C409
S3 868 с чипсетом ATT 20C498 или 21C498
S3 868 с чипсетом SDAC (86C716)
S3 86C260 (generic)
S3 86C280 (generic)
S3 86C325 (generic)
S3 86C357 (generic)
S3 86C365 (Trio3D)
S3 86C375 (generic)
S3 86C385 (generic)
S3 86C391 (Savage3D)
S3 86C764 (generic)
S3 86C765 (generic)
S3 86C775 (generic)
S3 86C785 (generic)
S3 86C801 (generic)
S3 86C805 (generic)
S3 86C864 (generic)
S3 86C868 (generic)
S3 86C911 (generic)
S3 86C924 (generic)
S3 86C928 (generic)
S3 86C964 (generic)
S3 86C968 (generic)
S3 86C988 (generic)
S3 86CM65
S3 911/924 (generic)
S3 924 с чипсетом SC1148 DAC
S3 928 (generic)
S3 964 (generic)
S3 968 (generic)
S3 Aurora64V+ (generic)
S3 Savage3D
S3 Trio32 (generic)
S3 Trio3D
S3 Trio64 (generic)
S3 Trio64V+ (generic)
S3 Trio64V2 (generic)
S3 Trio64V2/DX (generic)

S3 Trio64V2/GX (generic)
S3 ViRGE (generic)
S3 ViRGE (старая версия сервера S3V)
S3 ViRGE/DX (generic)
S3 ViRGE/GX (generic)
S3 ViRGE/GX2 (generic)
S3 ViRGE/MX (generic)
S3 ViRGE/MX+ (generic)
S3 ViRGE/VX (generic)
S3 Vision864 (generic)
S3 Vision868 (generic)
S3 Vision964 (generic)
S3 Vision968 (generic)
SHARP 9080
SHARP 9090
SNI PC5H W32
SNI Scenic W32
SPEA Mercury 64
SPEA Mirage
SPEA/V7 Mercury
SPEA/V7 Mirage P64
SPEA/V7 Mirage P64 с чипсетом S3 Trio64
SPEA/V7 Mirage VEGA Plus
SPEA/V7 ShowTime Plus
STB Horizon
STB Horizon Video
STB LightSpeed
STB LightSpeed 128
STB MVP-2
STB MVP-2 PCI
STB MVP-2X
STB MVP-4 PCI
STB MVP-4X
STB Nitro (64)
STB Nitro 3D
STB Nitro 64 Video
STB Pegasus
STB Powergraph 64
STB Powergraph 64 Video
STB Powergraph X-24
STB Systems Powergraph 3D
STB Systems Velocity 3D
STB Velocity 128
STB Velocity 64 Video

STB nvidia 128
SiS 3D PRO AGP
SiS 5597
SiS 5598
SiS 6326
SiS SG86C201
SiS SG86C205
SiS SG86C215
SiS SG86C225
Sierra Screaming 3D
Sigma Concorde
Sigma Legend
Spider Black Widow
Spider Black Widow Plus
Spider Tarantula 64
Spider VLB Plus
TechWorks Thunderbolt
Techworks Ultimate 3D
Toshiba Tecra 540CDT
Toshiba Tecra 550CDT
Toshiba Tecra 750CDT
Toshiba Tecra 750DVD
Trident 3DImage975 (generic)
Trident 3DImage975 AGP (generic)
Trident 3DImage985 (generic)
Trident 8900/9000 (generic)
Trident 8900D (generic)
Trident Cyber 9382 (generic)
Trident Cyber 9385 (generic)
Trident Cyber 9388 (generic)
Trident Cyber 9397 (generic)
Trident TGUI9400CXi (generic)
Trident TGUI9420DGi (generic)
Trident TGUI9430DGi (generic)
Trident TGUI9440 (generic)
Trident TGUI9660 (generic)
Trident TGUI9680 (generic)
Trident TGUI9682 (generic)
Trident TGUI9685 (generic)
Trident TVGA 8800BR
Trident TVGA 8800CS
Trident TVGA9200CXr (generic)
Карты, не поддерживающие стандарт VGA
VI720

VL-41
VidTech FastMax P20
VideoLogic GrafixStar 300
VideoLogic GrafixStar 400
VideoLogic GrafixStar 500
VideoLogic GrafixStar 550
VideoLogic GrafixStar 560 (PCI/AGP)
VideoLogic GrafixStar 600
VideoLogic GrafixStar 700
ViewTop PCI
WD 90C24 (laptop)
WD 90C24A or 90C24A2 (laptop)
Weitek P9100 (generic)
WinFast 3D S600
WinFast 3D S600
WinFast S200
WinFast S430
WinFast S510
XGA-1 (шина ISA)
XGA-2 (шина ISA)

Приложение В

Решение проблем с инсталляцией и загрузкой

- Проблемы с инсталляцией системы
- Проблемы загрузки и вопросы, не связанные с инсталляцией

Это приложение поможет вам решить некоторые проблемы с установкой (о них рассказано в начале главы) и загрузкой системы (заключительная часть главы).

Проблемы с инсталляцией системы

В этом разделе приведены примеры проблем, возникающих при инсталляции системы, и возможные варианты их решения.

Загрузка с дискеты приводит к останову или перезагрузке

Есть несколько причин, вызывающих такую ситуацию. Первая — битая дискета. Помните, что загрузочный образ диска записывается без проверки формата и используется весь флоппи-диск. Одного сбойного сектора достаточно, чтобы вызвать проблемы. Запишите образ на другую дискету и попытайтесь загрузиться с нее.

Если это не помогает, убедитесь, что при загрузке файла по сети посредством FTP-клиента использовался двоичный, а не ASCII-режим.

Еще одной причиной может стать проверка на вирусы, осуществляемая материнской платой. Эту опцию следует запретить утилитой настройки BIOS.

И наконец, убедитесь, что дискета создавалась в режиме DOS, а не из командной строки DOS в Windows. Последнее обстоятельство также часто приводит к проблемам.

Загрузка с дискеты повисает в фазе Probing Devices (Проверка устройств)

При загрузке с дискеты иногда некорректно определяются приводы IDE Zip и Jaz. Если вы используете одно из таких устройств, отключите его и попытайтесь загрузиться заново. Если загрузка пройдет нормально, вы сможете установить FreeBSD, а по завершении инсталляции подключить это устройство.

Система загружается с CD-диска, но программа инсталляции показывает, что CD-ROM не найден

Причиной частенько является неправильно сконфигурированный CD-привод. В некоторых системах привод CD-ROM установлен как ведомое устройство второго контроллера при отсутствующем ведущем устройстве. Чтобы решить возникшую проблему необходимо установить привод CD-ROM как ведущее устройство второго контроллера.

Геометрия жесткого диска определяется неверно

Если FreeBSD не может корректно определить геометрию жесткого диска, существует два выхода из возникшей ситуации. Первый заключается в создании небольшого раздела DOS в начале диска. Как правило, в этом случае FreeBSD определяет геометрию правильно.

Второй способ предполагает запуск программы **pfdisk**, включенной в дистрибутив. **pfdisk** запускается в системе DOS и обычно правильно определяет геометрию жесткого диска. Затем FreeBSD можно указать параметры геометрии вручную.

Система Micron зависает при загрузке

В некоторых системах Micron поддержка PCI в BIOS содержат ошибки. Это приводит к неверному конфигурированию PCI-устройств на этапе их определения. Обойти эту проблему можно, запретив поддержку plug-and-play путем соответствующей настройки BIOS.

Сетевая карта 3Com PCI не работает с системой Micron

Это связано с предыдущей проблемой поддержки PCI в BIOS, содержащих ошибки в системах Micron. Отключите поддержку plug-and-play путем настройки BIOS.

SCSI-контроллер HP Netserver не определяется при загрузке

Это связано с конфликтом между адресами SCSI-контроллера шины EISA и шины PCI. SCSI-контроллер шины EISA использует слот 11, что вызывает конфликт с адресным пространством PCI. Для решения проблемы перейдите к интерфейсу командной строки **UserConfig** (когда это будет предложено при инсталляции). Отметьте, что визуальный режим не подходит, необходимо использовать командную строку. В приглашении **UserConfig**, введите следующие команды:

```
eisa 12
quit
```

Это позволит FreeBSD правильно определить контроллер и произвести установку системы. По ее завершении необходимо собрать специальное ядро, добавив в него строку:

```
options         EISA_SLOTS=12
```

В системе с видеокартой ATI Mach64 гаснет экран

Эта проблема связана с тем, что видеокарта ATI Mach64 конфликтует с четвертым последовательным портом. Для ее решения необходимо перейти к интерфейсу командной строки **UserConfig** и запретить устройства **sio0**, **sio1**, **sio2** и **sio3**. Затем выйти из него и продолжить загрузку.

Чтобы воспользоваться последовательными портами, необходимо внести изменения в исходный файл ядра **/usr/src/sys/i386/isa/sio.c**. Затем нужно найти подстроку **0x2e8** и удалить ее и предшествующую ей запятую. После этого заново собрать ядро. (О том, как это сделать, рассказано в главе 17).

Устройства, необходимые для установки FreeBSD, не обнаружены

Прежде всего, обратитесь к списку поддерживаемого оборудования в приложении Б и убедитесь, что ваше оборудования поддерживается системой. Если это так, но при этом используются ресурсы, не заданные по умолчанию, вам необходимо перейти к интерфейсу **UserConfig** и устранить возможные конфликты компонентов оборудования.

К программе **UserConfig** можно перейти в процессе инсталляции. Ее интерфейс выглядит следующим образом:

```
                    Kernel Configuration Menu

Skip kernel configuration and continue with installation.

     Start kernel configuration in full screen Visual mode.
     Start kernel configuration in CLI mode.

Here you have the chance to go into kernel configuration mode, making
any changes which may be necessary to properly adjust the kernel to
match your hardware configuration.

If you are installing FreeBSD for the first time, select Visual Mode
(press Down-Arrow then ENTER).

If you need to do more specialized kernel configuration and are an
experienced FreeBSD user, select CLI mode.

If you are certain that you do not need to configure your kernel then
simply press ENTER or Q now.
```

Воспользуйтесь полноэкранным визуальным режимом.

Эта программа предназначена для устранения аппаратных конфликтов.

Проблемы загрузки и вопросы, не связанные с инсталляцией

Здесь мы рассмотрим некоторые проблемы,возникающие при загрузке системы.

При попытке загрузиться FreeBSD выдает сообщение Missing Operating System (Отсутствует операционная система)

Обычно это означает, что FreeBSD неверно определила геометрию жесткого диска в процессе инсталляции. Существует два решения проблемы:

- Создать небольшой DOS-раздел в начале диска и установить на нем минимальный дистрибутив DOS. Это позволит FreeBSD правильно определить геометрию диска, необходимую для разбиения на разделы.
- Воспользоваться программой **pfdisk.exe**, включенной в дистрибутив на компакт-диске, для определения параметров геометрии, а затем вручную указать их редактору разделов FreeBSD.

В любом случае, систему необходимо переустановить.

Менеджер загрузки FreeBSD зависает на "F?"

Обычно это связано с неверным определением геометрии жесткого диска. См. выше.

Менеджер загрузки FreeBSD выдает сообщение Read Error (Ошибка чтения) и зависает

И опять-таки, это обычно связано с неверным определением геометрии жесткого диска. См. выше.

Менеджер загрузки FreeBSD отсутствует, загружается Windows

Одно из двух: или менеджер загрузки не был установлен при инсталляции FreeBSD, или же какое-либо выше действие при работе в Windows (например, **fdisk /mbr** или нечто подобное) удалило его. К счастью, все достаточно просто восстановить.

Для этого загрузитесь с дистрибутивного CD-диска или загрузочных дискет. Когда вы перейдете к интерфейсу программы **Sysinstall**, выберите **Configure** и **Fdisk** (см. главу 2). Если в системе имеется несколько жестких дисков, система запросит, для какого из них следует запустить **Fdisk**. Выберите ведущий диск, с которого загружается система. В программе **Fdisk** просто выберите **W**, чтобы сохранить изменения. Программа выдаст предупреждение, сообщая, что изменения можно вносить только в существующую инсталляцию. Выберите **Yes** и нажмите Enter. Когда система спросит о диспетчере загрузки, выберите **Master Boot Record**. По завершении этого процесса менеджер загрузки будет функционировать корректно.

FreeBSD обнаруживает меньше оперативной памяти, чем реально присутствует в системе

FreeBSD не всегда может корректно определить объем оперативной памяти в системе по данным BIOS. Обычно это приводит к тому, что FreeBSD обнаруживает не более 64 Мб RAM.

Для решения проблемы в конфигурационный файл ядра необходимо добавить строку:

```
options         "MAXMEM=n"
```

где **n** — объем памяти в килобайтах. Помните, что в двоичной математике килобайт соответствует 1024, а не 1000 байтам. Фактически, мегабайт это — 1048576 байта, а не 1000000. Соответственно, в одном мегабайте не 1000, а 1024 килобайта. Таким образом, в системе присутствует **k = m * 1024** Кб оперативной памяти, где **m** — размер в мегабайтах. Например, 128 Мб ОЗУ соответствует **128 * 1024 = 131072** Кб.

После внесения изменений необходимо собрать ядро (о том, как это сделать, рассказано в главе 17).

FreeBSD выдает сообщение Device Not Configured (Устройство не сконфигурировано) при попытке монтирования CD-диска

Этому может быть несколько причин. Простейшая заключается в отсутствии CD-диска в приводе. Убедитесь, что он там есть.

Вторая причина — привод ATAPI CD-ROM установлен как ведомое устройство второго контроллера при отсутствии на нем ведущего устройства. Необходимо сконфигурировать привод CD-ROM как ведущее устройство.

Третья причина — SCSI-привод CD-ROM не имеет достаточно времени, чтобы ответить на запрос инициализации шины, когда запускается ядро. В этом случае найдите в конфигурационном файле ядра строку **options SCSI_DELAY** и увеличьте время. (Оно задано в миллисекундах. По умолчанию используется значение 15000, т.е. 15 секунд.) Затем необходимо заново собрать ядро (см. главу 17).

Программы завершают работу с ошибками Signal 11

Эта ошибка подобна ошибке illegal operation (недопустимая операция) в Windows. Она означает, что программа попыталась обратиться к участку памяти, который ей не выделен. Это может быть результатом ошибки в программе, а если это происходит с утилитами, включенными во FreeBSD, — в самой системе.

Третьей возможной причиной является неисправное оборудование. Если проблема проявляется во время компиляции программного обеспечения, скорее всего, она связана с оборудованием. Такие проблемы вызывают плохие микросхемы RAM, перегрев процессора (проверьте, работает ли вентилятор и не завышена ли частота процессора), плохая кэш-память или источник питания.

Система выдает странные сообщения об ошибках при запуске top, ps и других системных утилит

Почти всегда эта проблема связана с рассинхронизацией версий системы (world) и ядра (kernel). Например, выполнена команда **make world**, но при этом новое ядро не собрано. Обратный вариант также может быть причиной: новое ядро создано из загруженного исходного кода, однако предварительно не выполнена команда **make world**.

Поскольку гораздо быстрее собрать ядро, чем всю систему, нужно попытаться собрать ядро и перезагрузить систему (о сборке ядра подробно рассказано в главе 17).

Если сборка ядра не решает проблемы, соберите заново всю систему (world). (Обратитесь к главе 18 за инструкциями о том, как выполнить **make world**.)

И в заключение, если никакой из этих подходов не решает проблемы, загрузите новый исходный код системы (см. главу 18), соберите систему, а затем и новое ядро.

Забыт пароль пользователя root

Если вы забыли пароль пользователя **root**, можно загрузиться в однопользовательском режиме и изменить его. Для этого следует нажать любую клавишу во время обратного отсчета при загрузке системы. Нажатие клавиши прервет загрузку и на экране появится приглашение

```
Ok
```

В нем необходимо ввести команду **boot -s**, которая продолжит загрузку системы в однопользовательском режиме. При запросе того, какой командный интерпретатор использовать, нажмите клавишу Enter. После этого система выдаст приглашение пользователя **root**:

```
#
```

Введите в нем **mount -u/** и нажмите клавишу Enter. Корневая файловая система будет смонтирована в режиме чтения-записи. Затем командой **mount -a** смонтируйте все остальные файловые системы. Для изменения пароля учетной записи **root** воспользуйтесь командой **passwd root**. При этом система не запросит старый пароль. Теперь достаточно ввести новый пароль и подтвердить его. После этого перегрузите систему командой **shutdown -r now**. По ее завершении система придет в обычное состояние.

ПРИЛОЖЕНИЕ

Источники информации

- Ресурсы, связанные непосредственно с FreeBSD
- Дополнительные ресурсы, связанные с BSD
- Другие ресурсы Internet

Сообщество пользователей и разработчиков FreeBSD достаточно велико и продолжает расти. Существует множество источников информации как о самой FreeBSD, так и о связанных с ней продуктах. Некоторые из них приведены в этом приложении.

Ресурсы, связанные непосредственно с FreeBSD

Web-сайты

www.freebsd.org

Это официальный Web-сайта проекта FreeBSD. Здесь можно найти новости, обновления и информацию о портах и пакетах, а также справочник по системе и различные обучающие руководства.

www.freebsddiary.org

На этом сайте собрано множество ответов на часто задаваемые вопросы (FAQS) и статей "how-to" по самым разным предметам. Здесь же находится ряд форумов, обсуждающих проблемы, связанные с FreeBSD и единственный в своем роде форум "FreeBSD Pets" (Домашние животные FreeBSD). Если вы пользователь FreeBSD и у вас есть домашний любимец, можете послать его фотографию в этот форум.

www.freebsdzine.org

Электронный журнал, освещающий различные темы, имеющие отношение к FreeBSD. Он создается пользователями сообщества FreeBSD для других пользователей этого сообщества.

www.freshports.org

Место, где можно найти самую свежую информацию о новых или обновленных портах FreeBSD.

www.freebsdmall.com

Неиссякаемый источник различных товаров, сопутствующих FreeBSD. Здесь можно отыскать компакт-диски, футболки, пиджаки, коврики для мыши, кофейные чашки, шапки, книги и другие вещички с символикой FreeBSD. Кроме того, здесь можно найти и службу профессиональной поддержки FreeBSD.

Списки рассылки

Все приведенные списки рассылки является официальными списками проекта. Чтобы подписаться на какой-либо из них, необходимо послать на адрес **majordomo@freebsd.org** сообщение следующего содержания:

```
subscribe имя-списка
```

Чтобы отписаться, достаточно послать на тот же адрес другое сообщение:

```
unsubscribe имя-списка
```

Многие списки генерируют большое количество почтовых сообщений. Поэтому существует возможность подписаться на многие из них в сокращенной форме (дайджест). В этом случае вам присылается сообщение тогда, когда объем всех сообщений превысит 100 Кб.

Общие списки

Любой пользователь может подписаться на общие списки рассылки и участвовать в дискуссиях. Однако до того, как посылать свои сообщения, следует ознакомиться с правилами (которые присылаются после подписки). Ниже приведены списки рассылки и их короткие описания.

freebsd-advocacy

Список для обсуждения достоинств FreeBSD и различных способов "продвижения" этой системы.

freebsd-announce

Важные объявления, связанные с FreeBSD. Этот список предназначен только для чтения и создает небольшой трафик.

freebsd-arch

Обсуждение архитектуры и дизайна системы.

freebsd-bugs

Сообщения об ошибках FreeBSD. Обратите внимание, что сообщения об ошибках направляются не в этот список, а в базу данных GNATS, причем в специальной форме (в виде отчета об ошибке). Форму можно найти по адресу **http://www.freebsd.org/send-pr.html**. После этого пересланный отчет об ошибке добавляется в список рассылки.

freebsd-chat

Общее (не техническое) обсуждение системы FreeBSD ее пользователями.

freebsd-commit

Изменения, вносимые в дерево исходного кода FreeBSD, добавляются в этот список. Этот список предназначен только для чтения.

freebsd-config

Здесь обсуждаются средства установки и конфигурирования FreeBSD (например, новые средства с графическим интерфейсом, которые, возможно, заменят **Sysinstall**). Эта рассылка предназначена для разработчиков и программистов.

freebsd-current

Если вы следите за ветвью CURRENT исходного кода FreeBSD, вам просто НЕОБХОДИМО подписаться на этот список рассылки. Кроме того, чтение рассылки предотвратит возможный запуск вами команды **make world** в момент, когда дерево исходного кода повреждено, что может сделать систему нестабильной. Вам не стоит работать с версией CURRENT, если вы не профессионал и не готовы иметь дело с

подобными проблемами. Это также значит, что здесь не обсуждаются вопросы типа "how-to" общего характера. Сюда можно направить лишь узко технические вопросы, связанные с поведением версии CURRENT. Вопросы другого типа не получают ответов или, чаще, вам предложат разместить их в списке **freebsd-questions**.

freebsd-isp

В этом списке обсуждаются поставщики услуг Internet, использующие FreeBSD. Отметьте, что этот список предназначен не для пользователей.

freebsd-jobs

Если вы профессионал во FreeBSD и ищете работу, объявления об этом можно найти в данном списке рассылки. Если вы работодатель и вам нужен администратор, разместите здесь сообщение "help wanted" (требуется помощь).

freebsd-newbies

Список, предназначенный для новичков во FreeBSD. Обратите внимание, что это не тот список, куда можно обратиться за помощью. Здесь новички обмениваются историями о своем опыте работы с FreeBSD и пр. Вопросы следует размещать в **freebsd-questions**, а не в **freebsd-newbies**.

freebsd-policy

Решения о политике системы, которые принимает основная команда разработчиков FreeBSD. Этот список генерирует небольшой объем почты и предназначен только для чтения.

freebsd-questions

Место для размещения технических вопросов о FreeBSD. Если вы размещаете вопрос, он должен быть достаточно детальным. Другими словами, не размещайте вопросов типа: "Демон **pppd** не работает. Что я делаю не так?" Это совершенно бесполезно для решения проблемы. Чтобы получить ответ, необходимо добавить такую информацию, как сообщения об ошибках в log-файлах, какая конфигурация системы используется и т.д. Кроме того, будьте вежливы и не посылайте разгневанных сообщений, если вам не ответят сразу. Помните, что большинство людей, отвечающих на вопросы этого списка, заняты другой работой и посвящают решению ваших проблем свое свободное время. Никто не получает денег за то, что помогает вам. Чтобы послать сообщение в список, необязательно подписываться на него. Просто укажите корректный почтовый адрес, по которому вам смогут ответить.

freebsd-stable

Этот список предназначен для обсуждения вопросов, связанных с ветвью STABLE исходного кода FreeBSD. Как и в случае с CURRENT, если вы работаете с версией *STABLE, вам просто НЕОБХОДИМО подписаться на этот список.* Хотя серьезные проблемы, которые могут испортить дистрибутив и сделать систему непригодной к загрузке, встречаются в STABLE гораздо реже, чем CURRENT, иногда может случиться и такое. Из этого списка вы узнаете, когда с деревом исходного кода возможны проблемы и когда вполне безопасно им пользоваться.

freebsd-security-notifications

Именно здесь размещаются объявления о дырах в защите и способах их устранения. Если вас беспокоит защищенность вашей системы, обязательно подпишитесь на этот список.

Технические списки

Существует несколько технических списков, предназначенных для обсуждения таких вопросов, как портирование на различные платформы, портирование различного программного обеспечения под FreeBSD, использование FreeBSD в портативных системах и т.д. Поскольку эти списки неинтересны большинству пользователей, я не привожу их здесь. Полный список технических списков рассылки можно найти по адресу **http://www.freebsd.org/doc/en_US.ISO_8859-1/books/handbook/eresources.html#ERESOURCES-MAIL**.

Дискуссионные группы USENET, посвященные FreeBSD

Ниже приведен список дискуссионных групп, посвященных FreeBSD. Они доступны в сети USENET:

- **comp.unix.bsd.freebsd.announce**
- **comp.unix.bsd.freebsd.misc**

Каналы IRC

Существует несколько каналов IRC, посвященных вопросам FreeBSD. Здесь не приведен полный список, поскольку некоторые предназначены для обсуждения более общих вопросов и редко связаны непосредственно с FreeBSD. Однако существует два канала, на которых вы наверняка сможете получить помощь по FreeBSD:

- В сети EFNet: **#freebsdhelp**
- В сети Undernet: **#freebsd**

Новичку может показаться, что пользователи игнорируют его, но, скорее всего, это не так. Многие люди работают и одновременно зарегистрированы на канале. Кроме того, в периоды высокой перегруженности сети в IRC может наблюдаться длительный промежуток времени между размещением сообщения и его получением другими пользователями.

Не забывайте, что люди будут обращаться с вами так же, как и вы с ними. Если вы невежливы из-за того, что не получаете ответов (или тех ответов, которые вам нужны), то, скорее всего, получите вежливый отпор. Вас будут игнорировать, а оператор канала, возможно, просто запретит вам регистрироваться на этом канале.

Дополнительные ресурсы, связанные с BSD

В этом разделе перечислены ресурсы, посвященные общим вопросам BSD, а не специфическим темам FreeBSD. Так как FreeBSD является наиболее популярной из операционных систем клона BSD, большинство информации связано именно с ней.

Web-сайты

www.daemonnews.org

Основной сайт новостей, имеющих отношение к BSD. На этом сайте имеются форумы и статьи "how-to". Здесь же размещены комиксы "Source Wars".

www.bsdtoday.com

Еще один сайт, посвященный ежедневным новостям по BSD.

www.maximumbsd.com

Здесь размещены новости о BSD, а также архивы и ссылки на обучающие руководства.

Дискуссионные группы USENET

В сети USENET доступны следующие дискуссионные группы, посвященные BSD:

- **comp.bugs.4bsd**
- **comp.bugs.4bsd.ucb-fixes**
- **comp.unix.bsd**

Другие ресурсы Internet

В этом разделе рассказано о ресурсах Internet, которые в определенной степени связаны с FreeBSD. Здесь приведены домашние страницы, посвященные некоторым программным продуктам, описанным в этой книге.

Web-сайты

www.slashdot.org

News for nerds. Stuff that matters. ("Новости для тупиц. Вещи, которые нужны") Это основной сайт новостей. Здесь можно найти статьи обо всем — начиная от программного обеспечения с открытым кодом и заканчивая обзорами кинофильмов и последними разработками в области термоядерного синтеза. На сайте slashdot рассказано о самых разных темах, часто в техническом или научном стиле.

www.xfree86.org

Это домашняя страница проекта XFree86, который занимается свободно распространяемой реализацией системы X-Window, включенной в состав FreeBSD.

www.gnu.org

Домашняя страница проекта GNU (этот рекурсивный акроним означает GNU's Not UNIX, "GNU — это не UNIX"). Многие из утилит, включенные в состав FreeBSD (например, компилятор GCC, язык обработки шаблонов GAWK, редактор EMACS, компилятор Фортран 77), разработаны проектом GNU. На этом сайте можно найти информацию обо всех продуктах, разработанных в рамках этого проекта.

www.gnome.org

Домашняя страница проекта Gnome Desktop Environment (Графическая среда GNOME). Новости, обучающие руководства и многое другое, посвященное Gnome.

www.kde.org

Основной конкурент Gnome. Если вы предпочитаете KDE, посетите этот сайт.

www.apache.org

Домашняя страница проекта Web-сервера Apache, самого популярного Web-сервера в мире. Он является свободно распространяемым Web-сервером и включен в FreeBSD.

www.mysql.com

База данных SQL с открытым кодом, включенная во FreeBSD. Этот сайт содержит документацию, обучающие руководства и т.д. по программному обеспечению базы данных MySQL.

www.postgresql.org

Еще одна база данных SQL, распространяемая в открытом коде и включенная в состав FreeBSD. Она имеет больше свойств, чем MySQL, но работает несколько медленнее.

www.php.net

Домашняя страница языка PHP, используемого для разработки Web-приложений. PHP — это конкурент технологии Active Server Pages (ASP), распространяемый в открытом коде.

www.perl.com

Информация и обучающие руководства по языку программирования Perl, включенному в стандартную часть FreeBSD.

www.python.org

Домашняя страница языка программирования Python, популярной альтернативы языку Perl.

www.sendmail.org

Домашняя страница Sendmail, почтового агента по умолчанию во FreeBSD.

www.postfix.org

Популярный почтовый агент, альтернативный Sendmail. Postfix — это недавно появившаяся замена Sendmail. Новым пользователям гораздо проще конфигурировать не Sendmail, а этот продукт.

Дискуссионные группы USENET

В сети USENET доступны приведенные ниже дискуссионные группы, посвященные не совсем обычным для FreeBSD вопросам:

Дискуссионные группы по UNIX общего характера

- **comp.unix**
- **comp.unix.questions**
- **comp.unix.admin**
- **comp.unix.programmer**
- **comp.unix.shell**
- **comp.unix.user-friendly**
- **comp.security.unix**
- **comp.sources.unix**
- **comp.unix.advocacy**
- **comp.unix.misc**

Дискуссионные группы, посвященные X-Window

- **comp.windows.x.i386unix**
- **comp.windows.x**
- **comp.windows.x.apps**
- **comp.windows.x.announce**
- **comp.windows.x.intrinsics**
- **comp.windows.x.motif**
- **comp.windows.x.pex**
- **comp.emulators.ms-windows.wine**

ПРИЛОЖЕНИЕ

Справочник по командам

Перечень команд

Данное приложение не заменяет справочные руководства и не содержит подробного описания всех опций каждой команды. Здесь дано лишь назначение команд и их краткое описание.

Для тех, кто никогда раньше не имел дело с UNIX, предлагаем несколько наиболее распространенных команд, так сказать обязательный джентльменский набор:

- **passwd** — применяется для смены пароля;
- **ls** — выводит листинг файлов текущего каталога;
- **find** — ищет файлы или каталоги, удовлетворяющие заданным условиям;
- **mkdir** — создает новый каталог;
- **rmdir** — удаляет указанный каталог;
- **cd** — выполняет переход в другой каталог;
- **cat** — принимает из стандартного ввода любые символы и затем отражает их в стандартный вывод; допускает перенаправление ввода и вывода;
- **cp** — применяется для копирования файлов;
- **rm** — используется для удаления файлов;
- **more** — выводит на экран содержимое файла, делая паузу после каждой страницы; для продолжения нажмите пробел;
- **vi** — текстовый редактор;
- **grep** — ищет указанную строку в одном или нескольких файлах;
- **who** — сообщает о пользователях, зарегистрированных в системе;
- **chmod** — изменяет права доступа к файлам или каталогам;
- **man** — форматирует и выводит страницы руководства по заданной команде.

Названия команд будет проще запомнить, если принять во внимание, что они представляют собой сокращения английских слов. Если требуется изменить что-то уже существующее, команда, выполняющая подобное действие, начинается с **ch** (change). Когда что-нибудь нужно сделать впервые, используется команда, которая начинается с **mk** (make), а когда необходимо что-то полностью отменить или удалить, следует команда, которая начинается с **rm** (remove). Так, например, для создания нового каталога можно использовать команду **mkdir**, а для удаления каталога — команду **rmdir**.

В первую очередь нужно научиться получать помощь в тех случаях, когда не совсем понятно, как использовать команду. Для этого есть несколько возможностей. Можно, например, ввести после имени команды опцию **-?**, **-h** или **--help**. Большинство команд во FreeBSD, как и в других системах UNIX, в ответ на это выводят краткую подсказку о синтаксисе команды и используемых ею опциях.

Мощным инструментом для извлечения информации о потенциальных возможностях системы служит команда **man**. Она принимает в качестве аргумента название команды, о которой вы хотите получить информацию. Так, если нужна информация о команде **ls**, наберите в командной строке **man ls**.

В выводе **man** будет дан синтаксис команды и подробное описание ее работы. Если в описании синтаксиса аргумент заключен в квадратные скобки, то он не обязателен. Если два синтаксических элемента разделены вертикальной чертой (|), то допустим

только один из них. Когда описание команды занимает более одного экрана, перемещаться по тексту можно, нажимая клавишу пробела. Чтобы вернуться в командную строку, нажмите клавишу Q.

Перечень команд

&

Оператор оболочки **&**, расположенный после любой команды, дает указание системе выполнять эту команду в фоновом режиме.

|

Оператор | (канал) можно указывать между отдельными программами командной строки для передачи по каналу выходных данных одной команды в другую. Например, можно перенаправить по каналу вывод команды **cat** в команду **sort** для сортировки содержимого файла:

```
$ cat long_list.txt | sort
```

>

Оператор > (стандартный вывод) позволяет перенаправить вывод программы в файл или на другое устройство. Необходима осторожность, поскольку этот оператор перезаписывает существующий файл! Чтобы сохранить распечатку текущего каталога в текстовом файле, введите в командной строке:

```
$ ls > dir.txt
```

Если файл **dir.txt** существует, он будет перезаписан!

<

Оператор < (стандартный вход) используется для передачи в программу содержимого файла или ввода из другого устройства. Например, этот оператор можно использовать аналогично команде **cat** для сортировки файла и сохранения результатов:

```
$ sort < unsorted.txt > sorted.txt
```

Обратите внимание на тонкую разницу между этими двумя командами:

```
cat > filename
cat < filename
```

Как же не ошибиться? Зададим себе вопрос: куда направлена стрелка? Если в команду, то это перенаправление ввода, а если из команды, то это перенаправление вывода.

>>

Оператор >> (добавление) не перезаписывает указанный файл, а добавляет вывод программы в конец этого файла. Его можно использовать, например, для построения журналов:

```
$ cat newfile.txt >> oldfile.txt
```

Содержимое файла **newfile.txt** будет дописано в конец файла **oldfile.txt**.

<<

Оператор << (вложенный документ) используется, чтобы сообщить программе, как будет обозначен конец ввода. Например, чтобы использовать командный интерпретатор в качестве текстового редактора, укажите ему, чтобы он перестал принимать ввод, когда появится слово **end**:

```
$ > document.txt << end
This is a line of text.
end
```

После того как появится слово **end**, оболочка запишет текст в файл **document.txt**, поскольку слов **end** означает, что ввод завершен.

adduser

Команда **adduser** запускает сценарий, облегчающий создание учетной записи нового пользователя. Ее может применять только пользователь **root**. После команды **adduser** следует указать имя новой учетной записи (***username***):

```
# adduser frodo
```

alias

Команда **alias** применяется для создания псевдонимов или альтернативных имен команд. Как правило, псевдонимы являются сокращением действительных команд. В следующем примере пользователь добавляет псевдоним **dir** для команды **ls**:

```
alias dir=ls
```

Команда **alias**, введенная без параметров, позволяет получить список всех текущих псевдонимов.

at

Команда **at** позволяет запланировать выполнение какой-либо программы в заданное время.

atq

Команда **atq** позволяет вывести список заданий, ожидающих выполнения в очереди, в частности задания, запланированные по команде **at**.

atrm

Команда **atrm** позволяет удалить задание из очереди работ, запланированных на выполнение командой **at**.

bash

Команда **bash** запускает интерпретатор **bash**, предоставляющий много полезных возможностей.

batch

В отличие от **at**, команда **batch** начинает выполнять заранее запланированные задания тогда, когда средняя загрузка не превышает указанного уровня. Как правило, используется для выполнения задач системного администрирования.

bc

Позволяет выполнять простые математические операции, предоставляя язык описания калькуляторов, по своей структуре похожий на язык C. Команда **bc** особенно полезна для перевода из одной системы счисления в другую.

bg

Команда интерпретатора **bg** позволяет перевести приостановленный процесс в фоновый режим. Предположим, что вы запустили команду на переднем плане (без использования **&** после команды). Потом выяснилось, что она будет работать долго, а вам нужен доступ к командному интерпретатору. Текущий процесс можно приостановить с помощью клавиш **Ctrl+Z**. Затем наберите **bg**, чтобы перевести этот процесс в фоновый режим.

cal

Отображает календарь на текущий месяц.

cat

Утилита **cat** применяется для соединения нескольких файлов в один, а также для прокрутки содержимого файла на экране. Команде **cat** нужно указывать файл ***filename***, который требуется вывести на дисплей:

```
cat filename
```

Команда **cat** обычно используется совместно с перенаправлением или конвейером.

cd

Данная команда применяется очень часто. Существует три способа ее применения:

```
cd ..
```

Перемещение на один каталог вверх по дереву каталогов.

```
cd ~
```

Переход в home-каталог из любого места, где бы ни находился пользователь в настоящее время. Соответствует команде **cd** без параметров.

```
cd directory_name
```

Переход в указанный каталог. Это может быть каталог, указанный относительно текущего расположения или относительно корневого каталога. В последнем случае перед именем каталога нужно поместить прямой слэш (/).

chfn

Команда **chfn** позволяет корректировать информацию о пользователе (которую можно получить, используя команду **finger**). С помощью этой команды можно вводить или обновлять информацию, извлекаемую сетевым инструментальным средством **finger** из записей файла **/etc/passwd**. За командой **chfn** должно следовать имя пользователя.

```
# chfn bilbo
```

chgrp

Команда **chgrp** применяется для изменения группы, ассоциированной с файлом или каталогом. Владелец файла (и **root**) имеет право изменить группу, связанную с файлом. Эта команда имеет следующий простой формат:

```
chgrp new_group file
```

chmod

Команда **chmod** позволяет изменить права доступа, связанные с файлом или каталогом. Права доступа к объекту можно указывать двумя способами: с использованием цифровой или символьной системы кодирования.

При использовании любого способа кодирования команда имеет одинаковый формат. Вначале должна быть указана команда **chmod**, затем права доступа (абсолютные или относительные), затем объекты, для которых нужно изменить режим доступа:

```
chmod permissions file
```

С каждым объектом связаны три набора пользователей: владелец объекта, группа объекта и все прочие. В символьной системе кодирования буква **u** (user) обозначает пользователя, **g** (group) — группу, **o** (other) — всех прочих пользователей и **a** (all) — всех. Имеется три основных типа прав доступа: **r** (read) — право на чтение, **w** (write) — право на запись и **x** (execute) — право на выполнение. Права доступа всех трех типов можно менять с помощью знаков плюс (+) и минус (-). Например, следующая команда предоставит право на чтение и выполнение файла **test1** его владельцу и группе:

```
chmod ug+rx test1
```

Чтобы лишить права на чтение и выполнение файла **test1** его владельца и группу, нужно вместо знака плюс (+) указать знак минус (-):

```
chmod ug-rx test1
```

При использовании цифровой системы кодирования указывается абсолютное значение прав доступа, т.е. их предыдущее состояние не учитывается. Права доступа, ассоциированные с файлом, кодируются следующим образом: 4 означает право на *чтение*, 2 — право на *запись* и 1 — право на *выполнение*. Сумма этих чисел составляет код прав доступа для каждой из категорий (владелец, группа, все остальные). При цифровом кодировании всегда указывают все три категории пользователей. Так, чтобы предоставить права доступа на чтение, запись и выполнение владельцу файла **test1** и запретить доступ к данному файлу всем остальным, используется код **700**:

```
chmod 700 test1
```

Чтобы сделать тот же файл доступным владельцу для чтения и записи, а также предоставить права на его чтение группе и запретить доступ всем прочим, нужно выполнить следующие арифметические операции. Для пользователя: право на чтение (4), права на запись (2). Сумма равна **6**. Для группы: только права на чтение (**4**). Для всех прочих — никаких прав (**0**). Поэтому команда примет вид:

```
chmod 640 test1
```

chown

Эта команда позволяет изменить идентификатор владельца файла, что дает возможность управлять доступом к файлу или каталогу. Владелец файла имеет право передать свои права, связанные с файлом, другому пользователю. Эта команда имеет следующий простой формат:

```
chown new_user_id file
```

chroot

Команда **chroot** позволяет указать в качестве корневого каталога произвольное место в файловой системе. Например, работая с сервером Internet, можно указать в качестве корневого каталога **/usr/ftp**. В таком случае любой, кто зарегистрируется по FTP (по умолчанию он попадает в корневой каталог), на самом деле попадет в каталог **/usr/ftp**. Это позволяет защитить остальную часть структуры каталогов от просмотра и вмешательства.

Эта команда имеет следующий формат:

```
chroot <original filesystem location> <new filesystem location>
```

chsh

Эта программа позволяет сменить тип оболочки, используемой при регистрации. Новая оболочка должна быть доступна в системе и разрешена к применению (указана в файле **/etc/shells**). В качестве параметра должно быть указано имя оболочки:

```
# chsh zsh
```

cp

Команда **cp** — это сокращение от **copy**. Чтобы скопировать содержимое ***old_file*** в ***new_file***, наберите:

```
cp old_file new_file
```

Если путь не указан, предполагается, что оба файла находятся в текущем каталоге. Часто используются опции **-r** (рекурсивное копирование каталога) и **-i** (интерактивное копирование с запросом на подтверждение перезаписи существующих файлов).

cpio

Команда **cpio** позволяет вводить и выводить файлы из файловых архивов. Она работает во многом аналогично команде **tar** (сокращение от tape archive — архив на ленте), но имеет несколько иной формат.

crontab

Команда **crontab** добавляет задания или команды для периодического запуска в определенное время. Формат команды:

```
crontab filename
```

где ***filename*** указывает файл **crontab** данного пользователя. Чтобы получить список всех текущих заданий, можно ввести команду **crontab** с опцией **-l**. Опция **-e** позволяет создавать или редактировать задание, а опция **-r** — удалять задание.

cu

cu — это программа связи с текстовым интерфейсом, позволяющая дозваниваться на другие компьютеры.

cut

Эта программа позволяет вырезать указанные столбцы или поля из входного текста. Команда **cut** может применяться в качестве текстового фильтра для обработки вывода других текстовых утилит или содержимого файлов, выборочно отображая поля текста. Формат команды:

```
cut [options] filename
```

date

Отображает текущую дату и время. Пользователь **root** может с помощью этой команды изменить дату и время.

dc

dc — настольный калькулятор с управлением из командной строки, позволяющий выполнять введенные из командной строки или файла формулы расчета в обратной польской нотации.

dd

Команда **dd** предназначена для преобразования файловых форматов. По умолчанию копирует файл со стандартного ввода на стандартный вывод, меняя размер блока и выполняя указанные преобразования.

df

Команда **df** позволяет узнать объем свободного дискового пространства в любой смонтированной в данный момент файловой системе.

diff

Команда **diff** ***file1 file2*** сравнивает содержимое файлов ***file1*** и ***file2*** и выводит различия между ними.

dir

Команда **dir** предназначена для вывода содержимого каталогов. Она имеет опции командной строки, во многом аналогичные команде **ls**.

dmesg

Команда **dmesg** позволяет вывести содержание буфера системных сообщений.

du

Команда **du** показывает, какой объем дискового пространства используется различными файлами или каталогами и какие объекты системы занимают наибольший или наименьший объем дискового пространства.

dump

Команда **dump** (прерогатива оператора **root**), позволяет создавать резервную копию всей файловой системы или отдельных ее каталогов. Эта команда используется в сочетании с командой **restore**, позволяющей извлекать файлы и каталоги из копий, сделанных командой **dump**.

echo

Команда **echo** обеспечивает эхо-повтор строки на терминале.

ed

Это очень простой строковый редактор.

ee

Редактор FreeBSD Easy Editor. Если в качестве параметра указано имя файла, он автоматически открывается.

elm

elm — программа обработки почты, позволяющая создавать письма, редактировать их и отправлять почту. С ее помощью можно раскладывать почту по различным папкам, а также указывать место размещения входящей почты.

emacs

Программа **emacs** (edit macros) представляет собой текстовый редактор GNU.

env

Команда **env** применяется для просмотра экспортируемых системных переменных. В результате выполнения этой команды выводится список с двумя столбцами. В нем слева имя переменной, а справа — ее значение. Команда применяется без параметров.

ex

ex представляет собой символическую ссылку на редактор **vi**.

fdformat

С помощью этой команды выполняется форматирование дискет.

fetchmail

Эта программа позволяет получать почту.

fg

Процессы могут выполняться в фоновом режиме или на переднем плане. Команда **fg** позволяет выбрать приостановленный процесс и перевести его в режим выполнения на переднем плане.

file

Команда **file** выполняет для каждого переданного ей параметра серию проверок, позволяющих определить тип файла.

find

Команда **find** позволяет выполнять поиск файлов в любом указанном каталоге, а также во всех его подкаталогах. Просматривая указанные каталоги, **find** создает список файлов, которые отвечают указанному критерию. Затем с файлами из списка могут быть выполнены определенные действия. Файлы могут сравниваться по имени, размеру, времени создания, времени модификации и т.д. Синтаксис команды включает саму команду, имя каталога, в котором нужно начать поиск, имя файла (можно применять символы-заместители) и перечень действий, которые должны быть выполнены со списком.

finger

Команда **finger** позволяет просматривать информацию о пользователях системы (которая обычно находится в файле **/etc/passwd**).

fmt

Команда **fmt** позволяет форматировать входной текст, разбивая его на страницы и устанавливая размеры строк, указанные в командной строке. Формат команды:

```
fmt filename
```

ftp

Это программа поддержки протокола FTP. Она предоставляет возможность перемещать файлы с удаленного компьютера на локальный или с локального на удаленный.

grep

Команда **grep** (global regular expression parse) обеспечивает поиск указанного шаблона ***pattern*** в указанном объекте. Эта команда имеет формат:

```
grep [pattern] filename
```

Команду **grep** можно использовать для обработки результатов других команд. В частности, в следующем примере в команду **grep** передается вывод команды **ps** для фильтрации всех вхождений слова **root** (опция **-v** означает "все, кроме указанного текста"):

```
ps -ef |grep -v root
```

Та же команда без параметра **-v** (**ps -ef |grep root**) позволяет найти в списке процессов все вхождения слова **root**.

gunzip

Эта программа позволяет распаковывать файлы, сжатые с помощью команды **gzip**, преобразуя их в первоначальную форму.

gzip

gzip — это GNU-версия программы архивации **zip**. Команда вызова этой программы выглядит просто:

```
gzip filename
```

halt

Команда **halt** позволяет выполнить корректный останов системы. Эту команду может применять только **root**.

head

Команда **head** (антипод команды **tail**) представляет собой текстовый фильтр; она отображает в стандартный вывод заданное число строк от начала указанного файла. Ее формат:

```
head [-nnn] filename
```

hostname

Команда **hostname** позволяет либо отобразить текущее имя хоста или домена, либо установить имя хоста системы.

ifconfig

Это одна из нескольких программ, применяемых для настройки сетевых интерфейсов. С ее помощью удобно проверять используемые в данный момент сетевые интерфейсы, а также получать данные об интерфейсах и трафике в них.

irc

Программа Internet Relay Chat. Программа предоставляет дисплей с разделенными окнами, что позволяет читать поступающие сообщения и набирать ответы.

ispell

Эта гибкая интерактивная программа для проверки орфографии.

kill

По умолчанию действие команды состоит в передаче процессу сигнала о его прекращении. Команда **kill** *n* завершает работу процесса с идентификатором *n*.

less

Команда **less** нужна для просмотра содержимого файла. Команда имеет синтаксис:

```
less filename
```

Утилита **less** аналогична **more**, но имеет больше свойств.

ln

Команда **ln** *file1 file2* создает ссылку *file2*, указывающую на *file1*. Для поиска в каталоге файлов, представляющих собой символические ссылки, применяется команда **ls** с опцией **-l**.

locate

Команда **locate** позволяет получить сведения о местонахождении файлов. Может применяться для быстрого поиска файлов в системе, поскольку в ней предусмотрена

единая база данных **locatedb** о местонахождении файлов, расположенная в каталоге **/var/db/locate**.

login

Программа **login** применяется при регистрации в системе. Она позволяет в любой момент переключиться с одной учетной записи на другую.

logout

С помощью команды **logout** можно выйти из текущей учетной записи пользователя системы. Если пользователь зарегистрирован только в этой учетной записи, то после выполнения этой команды он выйдет из системы.

look

Команда **look** применяется для поиска в текстовых файлах строк, содержащих указанную строку. Ее можно также использовать для проверки правописания слов, поскольку по умолчанию она выполняет поиск в системном словаре **words**, который находится в каталоге **/usr/share/dict/words**.

lpc

Программа **lpc** позволяет управлять очередями принтеров и демонами печати. С ее помощью можно запрещать или разрешать использовать принтер или буферную очередь принтера, изменять порядок заданий буферной очереди, а также определять состояние принтеров, буферных очередей печати и демонов печати.

lpq

Отображает текущий список заданий в очереди принтера. Если она вызвана без параметров, то выдает перечень заданий в очереди печати.

lpr

Команда **lpr** позволяет вывести на печать указанные файлы. Если в качестве параметров не указаны имена файлов, подразумевается стандартный ввод. Ниже приведен пример команды **lpr**:

```
lpr /etc/hosts
```

lprm

Команда **lprm** *n* удаляет задание с номером *n* из очереди принтера. Другие опции команды позволяют удалять задания определенного пользователя и т.д. С помощью этой программы можно остановить задание на печать, указав его номер в командной строке.

ls

Выводит содержимое текущего каталога. Часто используемые опции: **-l** (вывод атрибутов каждого элемента), **-a** (отображение скрытых файлов) и **-F** (формат, помогающий различать типы файлов).

lynx

lynx — это быстрый, компактный и эффективный Web-браузер, предназначенный только для текста. Запускайте **lynx** из командной строки консоли или окна терминала X11, указав Web-адрес:

```
$ lynx http://www.mcp.com
```

mail

Эта программа предоставляет простой интерфейс для отправки, обработки и чтения почты, она также очень удобна для отправки однострочных почтовых сообщений из командной строки. Но лучше использовать почтовую программу Pine.

make

Запущенная в каталоге порта, эта команда получает файлы порта и собирает его.

make install

Запущенная в каталоге какого-либо порта, эта команда инсталлирует этот порт, если он уже собран. Если нет, она вначале получает необходимые файлы и собирает порт.

make deinstall

Запущенная в каталоге порта, эта команда деинсталлирует порт. Кроме того, она удаляет все зависимости, которые не требуются никаким другим портам или пакетам.

make clean

Запущенная в каталоге порта, эта команда удаляет все рабочие и объектные файлы, созданные при сборке порта, освобождая, таким образом, дисковое пространство.

make distclean

Запущенная в каталоге порта, эта команда удаляет все рабочие и объектные файлы, созданные при сборке порта, а также полученные исходные файлы дистрибутива.

makewhatis

Команда **makewhatis** позволяет создать базу данных команды **whatis**, находящуюся в каталоге **/usr/man**.

man

Команда **man** позволяет форматировать и отображать на консоли справочные руководства. Команда

```
man command
```

отображает страницу справочного руководства команды ***command***.

mkdir

Команда **mkdir** применяется для создания нового каталога.

more

more — это фильтр, позволяющий постранично просматривать текст на экране. Формат: **more** ***filename***. С помощью этой команды текст можно перелистывать только вниз, в отличие от **less**, которая позволяет перемещаться по тексту туда-сюда.

mount

Команда **mount** дает возможность подключить файловую систему, указанную в особом формате (часто это имя устройства), к каталогу, указанному в качестве параметра. Так, команда:

```
mount filesystem point_to_mount
```

монтирует файловую систему ***filesystem*** в каталог, заданный точкой_монтирования ***point_to_mount***, и делает ее доступной к использованию. Файлы имеет право монтировать только **root**. При вызове команды **mount** без параметров она выдает перечень всех смонтированных файловых систем.

mv

Команда **mv** (move) служит для перемещения объектов из одного места в другое. Если в качестве параметров даны два файла (**mv** ***old_file new_file***), команда перемещает первый файл во второй, что равносильно уничтожению второго файла и переименованию первого. Если последний параметр указывает на существующий каталог (**mv** ***filename1 filename2 directoryname***), команда перемещает указанные перед ним файлы в этот каталог. Если в качестве первого параметра дан список файлов, то последним параметром обязательно должен быть указан каталог. Часто используются опции **-r** (рекурсивное копирование каталога) и **-i** (интерактивное перемещение с запросом на подтверждение перезаписи существующих файлов).

netstat

Команда **netstat** показывает состояние сетевых соединений. Опция **-r** позволяет получить информацию о таблице маршрутизации.

newgrp

Команда **newgrp** используется для добавления новой группы. С помощью этой команды можно временно стать членом другой группы, чтобы можно было работать с принадлежащими разным владельцам файлами и каталогами.

passwd

Обычный пользователь (не **root**) вызывает команду **passwd** без параметров. Команда дает ему возможность сменить пароль. Пользователь **root** может указать в качестве аргумента имя пользователя ***username*** и изменить его пароль.

```
passwd [username] [newpassword]
```

pdksh

Командный интерпретатор **pdksh** — функциональный аналог коммерческого командного интерпретатора Korn.

ping

Команда **ping** позволяет проверить каналы связи и определить, функционирует ли конкретный хост. Команду **ping** сопровождает IP-адрес или доменное имя:

```
# ping staffnet.com
```

Работу команды **ping** можно остановить нажатием клавиш **Ctrl+C**.

pkg_add

Команда устанавливает указанный пакет в системе. Формат:

```
pkg_add package_name
```

pkg_info

Отображает список установленных в системе пакетов и дает их короткие описания.

pkg_delete

Данная команда удаляет указанный пакет из системы. Ее формат:

```
pkg_delete package_name
```

pppd

Эта программа работает в системе демоном, поддерживая соединение PPP с провайдером Internet и обеспечивая связь и преобразование форматов данных, поступающих на компьютер и исходящих из него.

pppstats

Позволяет распечатать сетевую статистику протокола PPP и определить, активно ли соединение PPP и какой объем информации был передан.

pr

Команда **pr** выполняет основные функции форматирования текстовых документов для печати и может также применяться для преобразования входного текста в разные форматы.

procmail

Команда **procmail** обрабатывает входящую почту, выполняя поиск указанных строк в сообщениях. В случае нахождения требуемой строки, она может обрабатывать почтовые сообщения в соответствии с указанными фильтрами или правилами: отвергать, записывать в файлы или отвечать на них.

ps

Команда **ps** позволяет получить список текущих процессов. Часто используются опции: **-l** (подробный список), **-a** (вывод всех процессов) и **-x** (вывод процессов, не присоединенных к терминалу, в частности демонов).

pwd

Команда **pwd** позволяет получить информацию о текущем рабочем каталоге. Она сообщает вам, в каком каталоге вы сейчас находитесь.

reboot

Перезагружает систему.

red

Редактор **red** (restricted ed) представляет собой вариант команды **ed**, который позволяет редактировать файлы только в текущем каталоге. В этом редакторе нет команды выхода в командный интерпретатор.

restore

Программа восстановления резервной копии, созданной программой **dump**. В команде **restore** предусмотрена встроенная справка и интерактивный режим восстановления файлов и каталогов из резервной копии, созданной ранее с помощью команды **dump**.

rm

Команда **rm** (remove) позволяет удалить указанные после нее файлы. При использовании с опцией **-r** команда **rm** удаляет файлы рекурсивно. Если, будучи **root**, ввести команду **rm -r /**, то все файлы в системе будут уничтожены. Эту команду удобно применять в сочетании с командой **find** для поиска и удаления файлов, принадлежащих определенному пользователю или группе. По умолчанию команда **rm** не удаляет каталоги.

rmdir

Команда **rmdir** позволяет удалить каталог, если он пуст. Формат команды:

```
rmdir dir_name
```

rmuser

Эта команда запускает сценарий, удаляющий учетную запись пользователя из системы.

route

Команда **route** позволяет просматривать или настраивать таблицу IP-маршрутизации. Это сетевая утилита, с помощью которой можно следить за обменом информацией через интерфейсы компьютера.

sed

sed (stream editor) — это неинтерактивный текстовый редактор, позволяющий манипулировать символьными потоками. Команда **sed** может применяться для быстрого выполнения глобальных операций поиска и замены в потоках символьной информации.

```
# cat employees.txt | sed 's/Bill/William/g' >newemployees.txt
```

В этом примере все вхождения строки **Bill** в первоначальном файле будут заменены строкой **William**, а затем создан новый файл.

shutdown

Останавливает или перезагружает систему. Выполнять ее имеет право только **root**.

sort

Применяется для создании упорядоченных по алфавиту списков строк, выбранных из файлов. Информация может быть отсортирована в прямом и обратном порядке. Так, например, команда:

```
sort filename
```

сортирует содержимое указанного файла в алфавитном порядке. По умолчанию вывод пересылается в поток **STDOUT**.

su

Команда **su** позволит на время стать другим пользователем. Если идентификатор пользователя не задан, считается, что вы хотите стать **root**. В любом случае происходит запуск командного интерпретатора, и текущий пользователь становится пользователем с другим UID и GID. Если эту команду применяет пользователь, отличный от **root**, команда **su** запрашивает пароль.

swapon

Команда **swapon** устанавливает область подкачки в файле или на блочном устройстве, заданном в качестве параметра. Команда **swapon** позволяет запустить страничный обмен и обычно выполняется во время начальной загрузки системы.

tail

Команда **tail** (хвост) позволяет направить на стандартный вывод указанное количество строк заданного файла. Формат:

```
tail [-nnn] filename
```

По умолчанию **tail** отображает последние 10 строк файла. Установленное по умолчанию значение в 10 строк можно изменить, указав параметр *-nnn*.

talk

Команда **talk** применяется для "живого" общения с другим пользователем посредством терминала.

tar

tar — это программа архивации, предназначенная для сохранения и извлечения файлов из архивного файла. Команда **tar** имеет следующий формат:

```
tar action optional_functions <file(s)/directory(ies)>
```

Если последним параметром является каталог, команда **tar** обрабатывает также все подкаталоги этого каталога.

tcsh

Расширенный командный интерпретатор, обладающий всеми средствами **csh** плюс многочисленные усовершенствования, включая редактирование командной строки, управление заданиями и ведение хронологии команд.

telnet

С помощью команды **telnet** можно регистрироваться на удаленной компьютерной системе, выполнять программы или просматривать данные.

top

Команда **top** отображает список процессов и статистику использования ими ресурсов системы.

touch

Команда **touch** ***filename*** обновляет время доступа к файлу ***filename***. Если указанный файл не существует, он создается (с нулевой длиной).

tput

Команда **tput** предназначена для изменения или перевода в исходное состояние установок терминала.

tr

Команда **tr** служит для транслитерации символов. Она представляет собой текстовый фильтр, позволяющий преобразовывать наборы символов, указанные в командной строке. Классический пример — преобразование всех символов текстового файла из прописных в строчные:

```
# cat uppercase.txt | tr A-Z a-z > lowercase.txt
```

umount

Команда **umount** позволяет размонтировать файловую систему, которая была перед этим смонтирована в определенном месте дерева каталогов. Формат этой команды приведен ниже:

```
umount filesystem
```

uptime

С помощью этой команды можно узнать продолжительность непрерывной работы системы. Команда **uptime** показывает, как долго проработала система, кто в настоящее время в ней зарегистрирован и какова была средняя загрузка системы за последние 5, 10 и 15 минут.

vi

Текстовый редактор **vi**. Если в качестве параметра указано имя файла, он автоматически открывается.

vipw

Средство для прямого редактирования файла **/etc/master.passwd**. По завершении работы обновляет базу данных.

vmstat

Позволяет вывести на экран статистику виртуальной памяти. Эта команда показывает, какой объем дискового пространства используется системой (обычно в разделе файла подкачки).

w

Команда **w** дает возможность выяснить, кто зарегистрирован в системе и чем занят.

wall

Отображает содержимое стандартного ввода на все терминалы всех пользователей, зарегистрированных в настоящее время в системе. По существу, она аналогична команде **write**, выполненной на всех терминалах (write all), отсюда и название.

wc

Команда **wc** (word count) позволяет подсчитать число символов, слов и строк в файле и получить на экране небольшой отчет. Команда задается в форме:

```
wc filename
```

и по умолчанию показывает все три результата, но можно сократить этот отчет с использованием опций **-c**, **-w** или **-l**.

whatis

Эта команда выполняет в базе данных **whatis** поиск имен команд и выводит на печать краткую сводку о назначении каждой команды. В качестве параметра принимает имя другой команды:

```
# whatis emacs
```

whereis

Программа **whereis** позволяет найти команду, а также источники информации и справочные руководства по этой команде.

who

Команда **who** позволяет узнать регистрационное имя, тип терминала, время регистрации и имя хоста каждого пользователя, зарегистрированного в системе.

whoami

Команда **whoami** позволяет узнать, под каким именем пользователь работает в настоящее время. Это нужно, если оператор **root** забыл, в какой ипостаси он в данный момент пребывает.

Вопросы локализации FreeBSD

- Языки, символы, кодировки
- Локализация
- Русификация консоли
- Русификация X-Window
- Часто задаваемые вопросы
- Несколько слов о проблемах украинизации
- Полезные ссылки

У пользователей, которые только начинают работать с FreeBSD, часто возникают вопросы, связанные с локализацией. Что же такое локализация и как ее настроить?

Так уж сложилось, что разработчики вычислительных машин говорили по-английски. По мере завоевания компьютерами планеты Земля постепенно стала нарастать потребность в обработке информации не только на английском, но и на других языках — немецком, японском, китайском, русском, украинском.

Термин *интернационализация* (в компьютерном мире принято сокращать это слово до **i18n**, где 18 — это число пропущенных букв в слове *internationalization*) подразумевает такой способ проектирования ПО, при котором возможность многоязыковой поддержки закладывается в программу еще на стадии ее разработки. При этом национально-зависимая часть приложения хранится отдельно от остальной части приложения и представляет собой набор данных — *объектов локализации*. Благодаря этому изменение языкового окружения под конкретные требования выполняется легко и просто, не требуя перекомпиляции приложения.

В свою очередь, *локализация* (localization, или сокращенно **l10n**) — это процесс адаптации какого-либо ПО под конкретные национальные требования. Технически локализация сводится к изготовлению объектов локализации, соответствующих требованиям конкретного языка.

В стандарте **POSIX** (Portable Operating System) определены средства локализации, названные **locale** (в переводе с английского **locale** — местные особенности). Структурно средства локализации **POSIX** состоят из следующих компонентов:

- Набор библиотечных вызовов (locale API): **setlocale()**, **isalpha()**, **toupper()** и др.
- Переменные окружения, для управления средствами **locale**: **LANG**, **LC_CTYPE** и др.
- Утилита для получения информации о средствах локализации: **locale**.
- Исходные тексты описания **locale**: locale definitions, Character Set Definition File.
- Утилита для компиляции объектов локализации: **localedef**.
- Собственно объекты локализации (наборы данных **locale**).

Локализация — понятие широкое. Необходимо организовать ввод и вывод символов национального алфавита, а также настроить систему в соответствии с принятыми в данной стране обычаями и традициями (местная денежная единица, формат представления даты и времени и пр.).

Таким образом, русификация операционной системы подразумевает три задачи: русификацию консоли, системную локализацию и русификацию приложений. Потребуется также уделить внимание выбору правильного значения локализации, установить средства локализации (если они еще не установлены), включить локализацию и проверить правильность функционирования средств локализации.

Языки, символы, кодировки

Люди мыслят словами, вычислительные машины — числами. Чтобы добиться взаимопонимания между ними, возникает необходимость в таблицах соответствия, называемых таблицами кодирования символов. Самой известной таблицей такого рода явля-

ется код ASCII (American Standard Code of Information Interchange), разработанный для телеграфа. Код содержит 128 7-битных комбинаций, причем первые 32 кодовых комбинаций используются для передачи управляющих сигналов.

Этот код компания IBM и применила для своих компьютеров. Для кодирования всех символов английского алфавита было достаточно однобайтовых комбинаций. IBM расширила таблицу до 256 символов и добавила коды символов псевдографики, что позволило выводить на печать таблицы и математические формулы. Затем компания разработала несколько кодовых таблиц, обеспечивающих использование компьютеров в других странах, где говорят и пишут на языках, отличных от английского.

Поскольку символы русского языка можно было разместить во второй половине таблицы по-разному, именно так и получилось. Появилось сразу несколько *таблиц кодировки* (*charset*) кириллицы: IBM-866, CP-1251, KOI8-r, ISO-8859-5 и, как следствие, необходимость конвертации из одной кодировки в другую. С китайским и японским языками дело обстояло еще сложнее, поскольку 256 символов недостаточно для отображения всех символов этих алфавитов.

В связи с этим была разработана универсальная кодовая таблица — UNICODE (Universal Character Set), содержащая символы, имеющиеся в языках всех народов мира, а также различные служебные, математические, технические, лингвистические и прочие знаки и символы. UNICODE использует 16-битные коды, что обеспечивает представление 65536 символов. В настоящее время в стандарт UNICODE (ISO 10646), входят практически все символы (~40 000), где-либо в мире используемые. Сравнительно недавно (в сентябре 1998 г.) в этот перечень был добавлен символ валюты EURO.

Чаще всего набор символов определяет языком и соответствующей ему системой письменности. Конкретный набор символов может являться подмножеством другого, более обширного набора символов или же комбинацией нескольких наборов (или их частей). Например, широко распространенный набор символов для представления русского языка: KOI8-R, содержит в себе символы из наборов LATIN, CYRILLIC, BOX DRAWING, BLOCK ELEMENT и т.д.. А конкурирующий с ним набор Windows-1251 содержит больше символов из набора CYRILLIC (русские, украинские, белорусские), но меньше из BOX DRAWING и BLOCK ELEMENT. Интересно, что в KOI8-R нет символа EURO.

При нажатии клавиш микропроцессор клавиатуры генерирует определенный скан-код. Далее скан-код преобразуется в код символа (ASCII или UNICODE) в соответствии с *таблицей раскладки клавиатуры* и передается приложению.

Для обеспечения поддержки ввода русскоязычных символов необходимо указать предпочитаемую раскладку клавиатуры для русского языка — специальные файлы с расширением ***.kbd** обычно лежат в каталоге **/usr/share/syscons/keymaps**.

Для вывода символов на монитор используется драйвер экрана. Этот драйвер может работать либо в режиме UTF, либо в байтовом режиме. В первом режиме последовательность байтов, получаемая от приложения, по алгоритму UTF преобразуется непосредственно в коды UNICODE. Затем драйвер экрана обращается к *таблице экранного шрифта* (SFM), загруженной в память, где хранятся изображения соответствующих символов. В байтовом режиме драйвер экрана использует дополнительную *таблицу перекодировки* — ACM (Application Charset Map).

Современные видеокарты позволяют, кроме использования аппаратных шрифтов, загружать и другие наборы символов. Это, в частности, позволяет видеокартам изображать символы кириллицы и дополнительные "экзотические" значки, отсутствующие в стандартном шрифте видеокарты. Драйвер **syscons** поддерживает это свойство видеокарт, то есть предоставляет пользователю возможность загружать шрифты из файла.

Готовые файлы шрифтов находятся в каталоге **/usr/share/syscons/fonts**. Там же есть и файл с кратким описанием шрифтов. Обратите внимание, что для каждого набора символов существует по крайней мере три разных файла, с разными размерами (в пикселах) символов — 8×8, 8×14, 8×16. В стандартном режиме, на экране изображается 25 строчек по 80 знаков каждая, но в других режимах на экране может быть 30, 43, 50 или даже 60 строк (все зависит от возможностей видеокарты). В одних режимах размер одиночного знака складывается из матрицы 8×16 точек, в других — 8×14 или 8×8.

В стандартном режиме (80×25) используются шрифты 8×16, но на всякий случай рекомендуется загружать все три типа шрифтов. Замечу, что обычно шрифты загружаются при старте системы, а какие именно — записано в соответствующих строчках файла **/etc/rc.conf**. Поэтому "рекомендуется загружать" означает, что нужно подправить определенные строчки в **rc.conf**.

Локализация

С точки зрения пользователя локализация — это настройка системы на особенности определенной страны. Все основные параметры локализации стандартизированы и управляются несколькими переменными, показанными в таблице Е.1:

Таблица Е.1. Категории локализации

Переменная	*На что она влияет*
LANG	Влияет на все параметры локализации. Ее значение может перекрываться другими переменными.
LC_COLLATE	На порядок сортировки
LC_CTYPE	На алфавит
LC_MESSAGES	На язык сообщений программ
LC_NUMERIC	На правила представления чисел с плавающей запятой
LC_PAPER	На формат бумаги
LC_TIME	На представление времени и даты
LC_MONETARY	На представление денежных единиц
LC_NAME	На формат записи имен
LC_ALL	Влияет на все параметры. Перекрывает значения всех остальных переменных.

Современные дистрибутивы FreeBSD по умолчанию содержат системные средства локализации.

Команда locale

Если ввести команду **locale**, система выдаст значения вышеперечисленных переменных окружения, управляющих локализацией:

```
$ locale
LANG=ru
...
```

Для украинизированной системы значения будут несколько другими, например:

```
$ locale
LANG=uk_UA.KOI8-U
...
```

Приведенные выше примеры дают представление об общем подходе к локализации.

Переменная LANG

Как правило, управление локализацией осуществляется с помощью переменной LANG. Присвоение этой переменной разных значений изменяет языковое окружение:

```
$ LANG=uk_UA.KOI8-U date
Пнд Гру  2 9:11:28 EET 2002
$ LANG=ru_RU date
Пон Дек 2 9:12:32 EET 2002
```

Необходимо сказать несколько слов о формате имени этой переменной. Он таков:

```
ll[_CC[.EEEE]][@ddddd]
```

где **ll** — это двухбуквенный код языка (стандарт ISO 639), указанный в нижнем регистре; **CC** — двухбуквенный код страны (стандарт ISO 3166), указанный в верхнем регистре; **EEEEE** — название таблицы кодировки в верхнем регистре; **ddddd** — название диалекта языка (при необходимости).

Хотя в принципе допустимо задавать короткое именование, вроде LANG=ru_RU или даже LANG=ru, лучше использовать _полное_ имя : LANG=ru_RU.KOI8-R. Переменная LINGUAS является GNU-расширением LANG и используется, если необходимо задать несколько вариантов локализации, причем задаваемые значения разделяются двоеточием.

Русификация консоли во FreeBSD

О настройке локализации написано много. Обратитесь к ресурсам, перечисленным в конце главы. Данный раздел основан на статьях гуру в области системного администрирования Ивана Паскаля (**http://www.citforum.ru/operating_systems/freebsd/rus.shtml**) и Rainman (**http://lasource.r2.ru/os/unix/russian_freebsd.html**).

Для русификации консоли требуется настроить корректное отображение символов на экране, ввод и переключение раскладок клавиатуры, выбрать профиль локализации и сконфигурировать некоторые программы.

В состав FreeBSD входит два варианта кириллических шрифтов: CP-866 и KOI8-R. Основной русской кодировкой UNIX-подобных систем де-факто является KOI8-R. В любом случае придется иметь дело с обеими кодировками. Существует и два варианта отображения символов: непосредственно с помощью набора шрифтов KOI8-R, а также с использованием альтернативного набора шрифтов CP-866 и автоматической перекодировкой символов KOI8-R в CP-866 при выводе на экран.

```
keymap=ru.koi8-r.win
```

Строчка **scrnmap=koi8-r2cp866** задает аргумент для **vidcontrol**, но уже не для установки шрифтов, а для загрузки в **syscons** таблицы перекодировки (**/usr/share/syscons/scrnmaps/***). Как вы помните, шрифты рекомендуется устанавливать для кодировки cp866, а чтобы тексты в кодировке koi8-r отображались правильно, необходима таблица для перевода из koi8-r в cp866. Если этого не сделать, то некоторые буквы (**п**, **р**, **я**, **с**) могут отображаться неправильно.

Все переменные, установленные в **/etc/rc.conf**, — по сути, установки для команд **vidcontrol** и **kbdcontrol**. Если интересно, как они обрабатываются, смотритефайл **/etc/rc.i386**. Последние три строки являются аргументами для утилиты **vidcontrol**, которая должна будет загрузить в видеоконтроллер новый набор шрифтов (**/usr/share/syscons/fonts/***).

Напомню, что для стандартного режима отображения (25 строк) необходим только шрифт с матрицей 8x16. Руководство по FreeBSD рекомендует использовать шрифт **font8x16b**; однако его особенность — резкое различие начертаний символов латиницы

И хотя второй метод кажется неоправданно сложным, он имеет некоторые преимущества. Дело в том, что шрифты cp866 лучше подходят для видеокарт PC-компьютеров. Каждый символ на экране рисуется в матрице шириной в 9 точек, при этом, загружая шрифты, можно "перерисовать" только 8 из них. Девятая колонка служит для того, чтобы символы, если даже они полностью занимают все восемь "программируемых" колонок, все-таки не сливались между собой. С другой стороны, есть такие символы псевдографики, которые предназначены для рисования рамочек и просто горизонтальных линий. Для них, наоборот, желательно, чтобы символы сливались, иначе горизонтальные линии будут выглядеть пунктирными.

Видеоконтроллер учитывает эти различия и отрисовывает девятую колонку по-разному. Для обычных символов эта колонка остается пустой (независимо от содержимого остальных восьми колонок), а для псевдографических — девятая колонка точек просто повторяет восьмую. Проблема в том, что в разных кодировках псевдографика попадает в разный диапазон символов, поэтому в шрифтах cp866 "псевдографика" отображается правильно (и горизонтальные линии рисуются слитно), а в koi8-r — нет (и те же линии рисуются пунктиром).

Итак, для использования первого способа русификации достаточно добавить в файл **/etc/rc.conf** три строки:

```
font8x16=koi8-r-8x16
font8x14=koi8-r-8x14
font8x8=koi8-r-8x8
```

При использовании шрифтов CP-866 и перекодировки (рекомендуется именно этот вариант!) необходимо в файле **/etc/rc.conf** в секции System console options написать следующее:

```
keymap=ru.koi8-r
keychange="61  ^[[K"
scrnmap=koi8-r2cp866
font8x16=cp866b-8x16
font8x14=cp866-8x14
font8x8=cp866-8x8
```

Первая строчка — **keymap=ru.koi8-r** — служит аргументом для утилиты **kbdcontrol**, которая при старте системы загрузит в **syscons** раскладку клавиатуры, соответствующую **ru.koi8-r**. Специальные файлы с расширением ***.kbd** обычно находятся в каталоге **/usr/share/syscons/keymaps**.

Естественно, что в этой раскладке предусмотрен модификатор для переключения между основной и "альтернативной" клавиатурами. Он соответствует клавише **[CapsLock]**, хотя эта клавиша не всем нравится (переключение заглавных/строчных букв при этом осуществляется комбинацией **Shift+CapsLock**). Обратите внимание, что последовательностью ^[обозначен символ с кодом 27 (он же 0033 или 0x1B). И в файле должен присутствовать именно он. Ввести его можно либо в каком-нибудь редакторе


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

Для установки украинской локали переменная LANG должна быть, соответственно, установлена в значение **uk_UA.koi8-u**. И все остальные действия следует должным образом откорректировать.

Все произведенные изменения в **/etc/rc.conf** возымеют действие только после перезагрузки; а в файле **/etc/ttys** — после перезагрузки или после посылки сигнала SIGHIP процессу **/sbin/init**.

Русификация приложений

Дальше просто необходимо настроить некоторые программы. Самое главное — указать тип терминала, поддерживающий русский язык. В файле **/etc/ttys** меняем тип с **cons25** на **cons25r** для всех виртуальных консолей. Получаем такие записи:

```
ttyv0  "/usr/libexec/getty Pc"  cons25r  on  secure
# Virtual terminals
ttyv1  "/usr/libexec/getty Pc"  cons25r  on  secure
ttyv2  "/usr/libexec/getty Pc"  cons25r  on  secure
ttyv3  "/usr/libexec/getty Pc"  cons25r  on  secure
ttyv4  "/usr/libexec/getty Pc"  cons25r  on  secure
ttyv5  "/usr/libexec/getty Pc"  cons25r  on  secure
```

Русификация X-Window

Для рядового пользователя самым важным моментом является русификация X-Window. Это позволит работать с документами на национальных языках. Как же русифицировать систему X-Window?

Для русификации **XFree86** понадобится русские шрифты. Их необходимо установить из портов или пакетов, либо выбрать их при инсталляции. Чтобы выяснить какие шрифты имеются в наличии, воспользуйтесь командой:

```
ls -la /usr/ports/russian/
```

Теперь установим шрифты из **/usr/ports/russian/X.language/** или же найдем соответствующий **package**:

```
# cd /usr/ports/russian/X.language
# make install
# make clean
```

Эта последовательность команд установит последнюю версию KOI8-R-шрифтов. Система XFree86 включает русские шрифты, но эти лучше.

Найдем в файле **/etc/XF86Config** раздел **Files** и добавим следующие строки так, чтобы они находились выше всех остальных строк, указывающих пути к каталогам, содержащим шрифты.

```
Section "Files"
FontPath     "/usr/X11R6/lib/X11/fonts/cyrillic/"
FontPath     "/usr/X11R6/lib/X11/fonts/cyrillic/misc"
FontPath     "/usr/X11R6/lib/X11/fonts/cyrillic/75dpi"
FontPath     "/usr/X11R6/lib/X11/fonts/cyrillic/100dpi"
```

Если вам нужно более высокое разрешение, поменяйте местами строки, заканчивающиеся указанием шрифтов с разрешением 75 dpi и 100 dpi.

В раздел **Keyboard** добавим следующие строки:

```
Section      "Keyboard"
XkbRules     "xfree86"
XkbModel     "pc101"
XkbLayout    "ru"
XkbOptions   "ctrl:nocaps,grp:ctrl_shift_toggle"
```

Переключателем в данной конфигурации будет комбинация клавиш Ctrl-Shift.

А, к примеру, запись **XkbOptions "grp:caps_toggle"** говорит о том, что в качестве переключателя используется клавиша CapsLock.

В некоторых версиях XFree86 (в частности в XFree86 4.3.0) по умолчанию используется только одна раскладка. Для того чтобы использовать и русскую, и английскую раскладки, необходимо исправить опцию **XkbLayout** следующим образом:

```
Option "XkbLayout"   "us,ru"
```

Если при запуске X-Window вместо осмысленного текста вы видите на экране смесь греческих букв с французскими диакритическими символами, следует проверить правильность путей к шрифтам и перенести кириллические шрифты в начало списка.

Не забудьте включить локализацию, используя переменные LANG и LC_ALL. Проверьте, чтобы в файле псевдонимов **/usr/X11R6/lib/X11/locale/locale.alias** в колонке полного описания псевдонимов стояло ru_RU.KOI8-R, а не ru_SU.KOI8-R.

Если вам хочется иметь удобный переключатель клавиатуры, воспользуйтесь программой **xrus (usr/ports/russian/xrus)**. Программа **xrus** — это русификатор для X-Window. Алгоритм его работы сводится к замене таблиц символов, поэтому он должен работать практически везде, в том числе и в старых реализациях XFree86, где отсутствовала штатная поддержка русского языка (**Xkb**), а также при использовании X-терминалов сторонних фирм.

Очевидно, что система X-Window, которая работает в графическом режиме и сама отрисовывает "в графике" все символы, не пользуется шрифтами, загруженными в видеокарту и "таблицей перекодировки" (screen map) драйвера **syscons**. Менее очевидно другое, что она также не использует и ту часть **syscons**, которая работает с клавиатурой. Для особенно любознательных сообщим, что X-сервер переключает **syscons** в режим выдачи "сырых" скан-кодов и сам переводит их в коды символов по своим правилам. Поэтому, никакие назначения клавиш, сделанные в таблице раскладки клавиатуры **syscons**, в X-Window не действуют.

Для изменения "раскладки клавиатуры" в X-Window пользуйтесь программой **xmodmap** или модулем **xkb**.

Часто задаваемые вопросы

Почему во FreeBSD русификация стандартно сделана через scrnmap из koi8-r в cp866, а не используется непосредственно koi8-r?

На самом деле существует два способа русификации консоли FreeBSD. Первый способ заключается в установке шрифтов koi8-r и изменении параметра ядра SC_MOUSE_CHAR. Второй сводится к использованию шрифтов cp866 и таблицы перекодировки scrnmap для преобразования из koi8-r в cp866.

Во FreeBSD стандартным является использование второго способа. Дело в том, что некоторые видеоадаптеры аппаратно расширяют (то есть копируют девятый бит изображения символа из его восьмого бита) символы, соответствующие псевдографике в оригинальной таблице символов. При использовании шрифтов koi8-r псевдографика получается "рваной", а русские буквы сливаются. Кроме того, при работе со шрифтами koi8-r и использовании мыши в консоли происходит искажение символов на экране. Поэтому нужно переопределить опцию SC_MOUSE_CHAR в конфигурационном файле ядра следующим образом:

```
options  SC_MOUSE_CHAR=0x03
```

Для некоторых адаптеров можно выбрать шрифты koi8-r без использования таблицы перекодировки **scrnmap**, однако предложенный способ более универсален. В частности, если понадобится просмотреть информацию в другой кодировке, достаточно сменить **scrnmap**, например, cp1251 на cp866.

Как русифицировать виджеты в Netscape Navigator?

Нужно файл **Netscape.ad** поместить в каталог **/usr/X11R6/lib/X11/app-defaults** под именем Netscape и поменять везде шрифты на шрифты koi8-r от Cronyx.

Не удается заставить webglimpse понимать русские буквы

Убедитесь, что кодировка, приходящая от клиентов, совпадает с той, на которую настроен **webglimpse**, и что она совпадает с кодировкой документов. Это может делать, например, русский Apache.

Поставьте **glimpse** версии не ниже 4.1, собранный с поддержкой locale (посмотрите в Makefile значение для ISO_CHAR_SET, оно должно быть равным 1). Если вы храните документы в кодировке CP1251, то должна иметься соответствующая настройка для locale.

Нужны перекодировщики русских текстов

Посмотрите в коллекции портов раздел **russian**, рекомендуется **ru-d1489-1.3** либо **trans-1.20**.

Несколько слов о проблемах украинизации

KOI8-u — преемник стандартов KOI8 и KOI8-r, принятых в качестве стандартов обмена информацией в Internet. KOI8-r был зарегистрирован как RFC-1489 Андреем Черновым и используется сейчас как стандарт Internet в основном в Unix-системах.

Стандарт на кодировку KOI8-r предполагает использование только символов кириллицы, имеющихся в русском языке. Зарегистрированного стандарта на кодировку кириллицы, в котором предусмотрены все символы, необходимые в русском, украинском и белорусском языках, на данный момент не существует.

Ныне применяется несколько кодировок кириллицы, исторически сложившихся, и каждая используется в пределах одной операционной системы: cp866 — в MS-DOS, cp1251 — в MS-Windows.

Долгое время в Украине, и повсюду в Internet для обмена информацией на украинском языке применялась кодировка KOI8-r с добавлениями украинских букв, что не было документировано ни одним стандартом и вызывало разногласия при использовании. Самым близким к потребностям русского, украинского и белорусского языков была кодировка ISO-IR-111, но и она не может быть использована по причине отсутствия в нем украинской буквы 'Г'.

В настоящее время ведутся работы по стандартизации новой кодировки KOI8-ru, которая будет включать в себя все символы русского, украинского и белорусского алфавитов. KOI8-ru будет полностью совместима с KOI8-r и ISO-IR-111. Кодировка KOI8-ru используется пока в Internet как стандарт де-факто.

Полезные ссылки

1. **http://www.freebsd.org.ua/** — Украинский сайт, посвященный популяризации FreeBSD.
2. **http://www.freebsd.org.ua/handbook/** — Руководство Пользователя FreeBSD
3. **http://www.freebsd.org.ua/qa/** — Вопросы и Ответы по FreeBSD.
4. **http://www.freebsd.org.ua/faq/** — Часто задаваемые вопросы по FreeBSD.
5. **http://koi8.pp.ru/framed-koi8.html** — Основные принципы работы с кодировками.

6. **http://www.kde.org.ua/docs/** — Статьи по проблемам локализации на украинском сайте, посвященном среде KDE.
7. **http://www.freebsd.org.ru/** — Российский сайт, посвященный FreeBSD.
8. **http://www.freebsd.org.ru/how-to/russian/** — Как русифицировать систему.
9. **http://www.pi2.ru/prof/** — Домашняя страничка Дмитрия Карпова. Здесь есть много интересного для системных администраторов.
10. **http://www.freebsd.org.ua/articles/fonts/** — Шрифты и FreeBSD. Сайт содержит описание различных файлов шрифтов, которые могут использоваться с FreeBSD и драйвером системной консоли, системой X11, программами Ghostscript и Groff. Дает массу полезной информации по русификации дисплея и клавиатуры, включению 8-bit в различных программах и использованию русских шрифтов.
11. **http://www.citforum.ru/operating_systems/freebsd.shtml** — Статьи по FreeBSD на CIT-форуме.
12. **http://www.citforum.ru/operating_systems/freebsd/index.shtml** — FreeBSD глазами администратора (статья Ивана Паскаля).
13. **http://charts.unicode.org/** — Здесь можно посмотреть набор символов **UNICODE**.
14. **http://www.unixfaq.ru/index.pl?id=82** — Вопросы локализации на сервере **unixfaq.ru**.
15. **http://www.opennet.ru/base/rus/freebsp48_gtk.txt.html** — Русификация FreeBSD 4.8 на форуме OpenNET.
16. **http://pascal.tsu.ru/unix/syscons/russian.html** — Статья Ивана Паскаля "Русификация syscons".
17. **http://pascal.tsu.ru/unix/syscons/vidcontrol.html** — Статья Ивана Паскаля о работе программы vidcontrol.
18. **http://www.sensi.org/~alec/locale/** — "Локализация как она есть". Раздел написан Александром Воропаевым.
19. **http://www.unixos.kiev.ua/stat/index.php** — Недавно открывшийся некоммерческий ресурс, который ставит своей задачей продвижение открытых операционных систем в Украине.
20. **http://unix1.jinr.ru/~lavr/RELENG_4.html** — Настраиваем FreeBSD.
21. **http://www.vpti.vladimir.ru/rus/archives/software/setup-xkb.html** — Настройка xkb.
22. **http://www.linux.org.ua/cgi-bin/twiki/view/Locale/LocalizationTechnology** — Замечательная статья на украинском языке, посвященная проблемам локализации. Если на вашей машине требуется поддержка украинского языка или вы пишите программы с его поддержкой, вы почерпнете здесь много интересного.

Предметный указатель

А

Агент пересылки почты 466
Адрес
  сети 427
  широковещательный 427
Анализатор протокола 552
Аргументы командной строки 245
Ассоциативный массив 397
Атаки
  на службы 574
  "с плацдарма" 576

Б

Базовые службы ввода/вывода 83
Бесклассовая междоменная маршрутизация 431
Бит устойчивости 195
Блоки 184
Брандмауэр 532, 544, 562
  включение 533
  включение поддержки 563
  конфигурирование 533
  правила 533

В

Вид доступа 191
Виртуальная приватная сеть 602
Виртуальные десктопы 98
Виртуальный хостинг 525
Выход из цикла 253

Г

Главная загрузочная запись 76, 83, 207
Группа wheel 189

Д

Дейтаграмма 528
Демилитаризованная зона 541, 563
Демоны
  cron 279
  DHCP 633
  dhcpd 636
  gated 540
  inetd 617
  lpd 305
  mountd 603
  NAT 532
  natd 533
  NFS 603
  PortSentry 568
  pppd 452
  named 581, 586
  nmbd 615
  smbd 615
  sshd 572
  zebra 540
  автоматического монтирования 608
  ввода/вывода 606
  маршрутизации 541
  монтирования 603
Динамическая сборка 287
Динамически загружаемые модули 507
Директивы 592
Домен 581, 613
  имен 434
  коллизий 421
Дуплекс 420

Ж

Жесткая ссылка 152
Жесткое ограничение 185
Журнальные файловые системы 181

З

Загрузочный сектор 84
Загрузчик 84, 208
Записи
  CNAME 595
  MX 596
  NS 594
  PTR 596
  SOA 593
Защита 543, 555
  FTP-сервера 556
  источники информации 577
  модели 543

руководство 578
сервера Apache 557
служб электронной почты 555
физическая 577
Зона 582
forward 589
master 588
подчиненная 588

И

Идентификатор
группы 196
setgid 196
пользователя 196
Имиджи загрузочных дискет 43
Индексные дескрипторы 184
Инкапсуляция 425
Интернационализация 732
Интерпретатор Perl 391
командный 66, 141, 222
Bourne (sh) 141
Bourne Again Shell (bash) 143
C (csh) 142
Korn (ksh или pdksh) 142, 266
sh 141
Tcsh (tcsh) 143
Интерфейс 531
Ethernet 531
История команд 163
Итератор 592

К

Кабель 417
перекрестный 418
прямой 418
Квота 170, 183
Код завершения 260
Команда
abort 319
arp 429
at 282
awk 161
cd 149
chfn 204
chmod 195, 534
chown 617
chroot 544
cp 149
cut 160
df 169, 606
disable 321
dmesg 180
down 321
du 170
egrep 158
expr 246
fgrep 158
fmt 161
getfacl 198
grep 158
hostname 444
ifconfig 441
kill 277
ln 152
lpc 319
lpd 314
lpq 317
lpr 314
lprm 317
ls 147
mount 171
mount_nfs 606
mv 150
printenv 236
printf 248
pwd 149
renice 278
restart 322
rm 151
rmdir 151
rmuser 203
route 443
Script-Fu 125
sed 161
setfacl 198
shift 252
sockstat 568
sort 159
status 319
stop 319
su 551
topq 322
touch 152
tr 159
umount 172

unalias 233
up 322
wc 157
Коммутатор 420
Конвейер 239
Контроллер домена 620
Конфигурирование ресурсов 208
Концентратор 419

Л

Литералы 395
Логические операторы AND/OR 250
Локализация 732

М

Маршрутизатор 422, 576
стандартный 530, 539
статический 540
функции 528
Маршрутизация 540
Маска максимальных прав доступа 198
Массив 268, 394
Менеджер
окон 95
пакетов 224
Метасимволы 154
Метод аутентификации 551
Микроархитектура ядра 328
Микшер 131
Модуль 407
mod_perl 558
mod_php 558
mod_ssl 558, 559
Perl 408
Мосты 422
Мультимедиа 130

Н

Набор операторов
push() 396
pop() 396
shift() 396
unshift() 396
Настройка
bash 235
приглашения 235
консоли 59
сети 438
Начальная загрузка 83
Начальный каталог 147, 226

О

Область подкачки 51
Оболочка 222
Обратные кавычки 399
Однопользовательский режим 208
Оператор
=- 402
break 253
case 258
continue 253
elif 256
false 253
if 255
options 590
true 253
zone 593
die 404
my 407
замены 402
конкатенации 395
математические 395
перевода 402
присваивания 402
сравнения 395
строкового повторения 395
транслитерации 402
условные 254
увеличения 395
чтения потока 404
Очередь
печати 305
сообщений 473

П

Пакет 613, 293
обновление 293
удаление 293
Панель задач 97
Параметр 533
natd_interface 533
Пароль 545
root 189
одноразовый 549
правила задания 545
устаревание 547

Переменные 243
  командного интерпретатора 237
  среды 236
  ввода-вывода 162
Перенаправление пакетов 531
Перехват прерываний 262
Период
  аренды 634
  отсрочки 185
Подключаемые модули аутентификации 450
Подстановка команд 246
Полудуплекс 420
Пользователь
  root 543
  daemon 307
Порты FreeBSD 294
Почтовый клиент 466
Преобразование сетевых адресов 528, 530
Префикс
  $ 396
  @ 396
Программа
  cap_mkdb 548
  chat 454
  Crack 545, 546
  dhclient 627, 630, 631
  dhcpconf 633, 634
  fdisk 363
  GIMP 123
  gzip 572
  keyinit 549
  Midnight Commander 738
  Mutt 138
  perl 394
  Reporter 546
  sftp 557
  showmount 605
  SSH 554
  sysctl 383
  Sysinstall 531, 601
  sysinstall 363, 629
  tcpdump 553
  tripwire 571
  uuencode и uudecode 138
  xsetroot 664
  Xwrapper 639
Просмотр 612
Протокол
  ARP 429
  DHCP 627
  FTP 135
  NetBIOS 600
  UDP 601
  POP3 476
Процедуры
  local() 406
Процесс начальной загрузки 207
Псевдоним 233
Псевдопользователь 189

P

Рабочая группа 613
Разделы BSD 361
Разделяемая библиотека 286
Расширения UFS 197
Регулярное выражение 400
Редактор 116
  ee 116
  vi 118
Режим доступа 191

C

Самотестирование 83
Свопинг 51
Сервер
  Apache 494, 558, 575
  NFS 600, 602
    конфигурирование 602
Серверные расширения 509
Сертификаты 559
Сетевая маска 430
Сетевая файловая система 600
Символ обратной косой черты 242
Символические ссылки 153
Символы-заместители 259
Синхронная запись 381
Система
  Apache-SSL 558
  BIND 582, 583
  CGIWrap 560
  Kerberos 551
  Samba 611
    журнальные файлы 622
    компоненты 623
    конфигурирование 614

переменные 622
SWAT 615
Tripwire 570
доменных имен 581
структура 581
пакетов 285
Скалярные переменные 395
Скрытые файлы 148
Слайсы 361
Специальные переменные 245
Список
рассылки 578
управления доступом 197
распаковки 296
Статическая сборка 287
Строка запроса 515
Суперблок 182
Суперсервер 216
Сценарий CGI 559, 631
/sbin/dhclient-script 631
конфигурация 86
конфигурирования ресурсов 209, 211
управление выполнением 86

Т

Таблица
перекодировки 733, 737
раскладки клавиатуры 733
электронного шрифта 733
Таблицы кодировки 733
Теневые пароли 376
Типы mime 109
Точка монтирования 53, 171, 169

У

Удаление пользователя 203
Управление доступом 619
на уровне общих ресурсов 621
на уровне пользователей 619
Установка путей 235
Устройство vp0 177
Утилита
dd 44
gunzip 571
make 224
kbdcontrol 736
perldoc 408
ping 448
pkg_add 287
pkg_create 287
pkg_delete 287
pkg_info 287
pkg_update 287
pkg_version 287
ps 275
restore 386
tcpdump 552
top 272
tripwire 571
twcheck 571
xhost 668
Учетная запись 621

Ф

Файлы
$HOME/.profile 737
fstab 178
.htaccess 501
.xinitrc 662
/bin/tcsh 223
/etc/csh.cshrc 230
/etc/csh.login 230, 737
/etc/diskcheckd.conf 181
/etc/dhclient.conf 632
/etc/fstab 178, 738
/etc/exports 604
/etc/fstab 607
/etc/ftpusers 523
/etc/group 205
/etc/hosts.allow 569
/etc/login.conf 547
/etc/master.passwd 203
/etc/netstart 445, 564
/etc/passwd 203
/etc/printcap 312
/etc/profile 232, 737
/etc/rc 212
/etc/rc.i386 737
/etc/rc.conf 183, 211, 445, 534, 536, 539, 564, 734
/etc/resolv.conf 449
/etc/shells 225, 523
/etc/skeykeys 551
/etc/syslog.conf 218
/etc/ttys 209, 738
/usr/bin/perl 391

/usr/lib/perl5 407
/usr/local/crack 546
/usr/local/etc 214
/usr/local/lib/perl5 407
/usr/ports/www/cgiwrap 560
/var/db/pkg 287, 298
/var/run/nologin 523
/var/spool 306
localhost 597
базовых директив 592
главный 591
зоны 583, 590
конфигурации dhcpd 634
конфигурации сети 450
обратного просмотра 596
подсказки об устройствах 333
создание 590
шрифтов 734
Файловые дескрипторы 264
Файловые системы smbfs 611, 624
Фильтры 124
преобразования 311
Функции 263

Х

Хаб 419
Хост 556

Ц

Циклы 249
for 251
until 250
while 249

Ш

Шлюз 430, 528, 537
NAT 530
конфигурирование 530
беспроводной 538

Я

Ядро 209, 328, 561
монолитное 328
уровень защиты 562

Специальные термины

a2ps 308
AbiWord 110
Access Control Lists 197
ACM 733
adduser 201
alias 233
AND/OR 259
ARP 429
ASCII 733
associative array 397
awk 161
Balsa 137
bash 143
BIOS (Basic Input/Output Services) 83
boot0 83, 207
boot1 84, 207
boot2 84, 208
bootstrapping process 207
Bourne 240
bridges 422
cd 149
CGI 511
CGI-программа 560
chgrp 193
chmod 534
chsh 227
CIDR (Classless Inter-Domain Routing) 431
clean 86
cnw 538
committers 375
conversion filters 311
CP-866 735
crontab 280
Crossover Cables 418
CVSup 344
DHCP 435
dial-on-demand 456
diamond operator 404
diskcheckd 181
disklabel 208
dmesg 209, 330
dot-файлы 148
dump 179, 385
dynamic linking 287
echo 262
edquota 184
errors.txt 73
escape-символ 156

F.D. 0 STDIN 264
F.D. 1 STDOUT 264
F.D. 2 STDERR 264
fdimage.exe 43
Fetchmail 483
fi 255
File Manager 99
FIPS 73
for 398
foreach 398
fsck 179, 209
FTP 135, 517
FTP-доступ 520
FTP-сервер 520
gedit 114
GENERIC 329
getopts 270
getty 87
Gnome 105
Gnome Control Center 108
Gnome File Manager 106
GQview 126
grace period 185
grep 158
halt 93
hard limit 185
hard link 152
home directory 189
home-каталог 147
ICMP 425
ifconfig 426
IMAP 481
inetd 216
IP-адрес 426
IP-адреса 627
  аренда 627
IP-алиасинг 447
IP-маскарадинг 427
kern.flp 43
kernel 209
KOI8-R 733, 735
less 157
LINT 333
loader 208
login shell 226
lptcontrol 306
lsmod 208
Lynx 135
MAC-адрес 428
Mail Transfer Agent 466
Mail User Agent 466
mailq 473
maintainer 297
make world 340
Makefile 296
man 144
Master Boot Record 207
master boot record 76
maximum permissions mask 198
mergemaster 345, 351
microkernel architecture 328
more 157
mount 171
Mount point 53
mpg123 133
msfroot.flp 43
MTU 442
NAT 427, 528, 530
netmask 430
NFS 600
package manager 224
PAP- и CHAP-аутентификация 455
partitions 174
patches 373
pdksh 142
persistent connection 456
Pine 137
ping 448
pkg_add 287
PortSentry 566
POST (Power On Self Test) 83
Post Office Protocol 476
PPP 532
  протокол 532
preen 178
print queue 305
printf 241
ProFTPD 526
qpopper 477
query string 515

Red Hat Linux 537
  Сетевая конфигурация 537
relaying 474
resource configuration 209
Resource Configuration script 209
restorrb.exe 73
root 88
routers 422
Sawfish 109
Sawfish Configurator 112
sed 161, 234
SFM 733
sh 141
shadow passwords 376
shared library 286
shell 222
shutdown 91
Simple Mail Transfer Protocol 464
single-user mode 208
Slackware Linux 537
  настройки сети 537
slices 174, 207, 361
smb.conf 615
SMB/CIFS 611
SMTP 464
soft limit 185
soft links, symbolic links 153
Soft Updates 181
split() 397
StarOffice 127
StarOffice Explorer 129
static linking 287
STDIN 244
STDOUT 241
Straight-through 418
strict mode 407
stty 234
switch 420
synchronous writes 381
sysinstall 438
syslogd 218
TCP 423
TCP/IP 425
UDP 424
UFS (Universal File System) 166
UFS Extensions 197
umass 177
UNICODE 733
UNIX File System 166
upgrade kit 302
uudecode 138
uuencode 138
vi 227
Web-ресурсы 578
while 398
wl 538
WU-FTPD 526
xdm 669
XF86Config 651
xf86config 640
XMMS 133
xv 664
xvidtune 64

Артур Гриффитс

GCC. Полное руководство

ISBN уточняется
70x100/16

Выход из печати — IV кв. 2003г.

Назначение книги

GCC (GNU Compiler Colection) - основной компилятор проекта GNU. Он поддерживает набор всех наиболее используемых языков программирования и обеспечивает перенос программ на десятки аппаратных платформ. Все свободно распространяемое программное обеспечение, включая компиляторы, на том или ином уровне основываются на GCC.

В книге даются подробные сведения о получении, конфигурировании, установке и тестированию компилятора. Описано построение компилятора для кросс-платформенной компиляции и создания встраиваемых систем.

Книга детально описывает компиляцию программ на языках C, C++, Objective-C, Fortran, Java и Ada. А также сочетание в одной программе нескольких языков программирования и включение в нее частей, написанных на ассемблере или языках системного уровня. Кроме того, описывается применение установок локализации и интернационализации приложений, поддержка приложениями национальных языков и стандартов.

«Полное руководство» содержит сведения об устройстве и работе компилятора GNU. В нем содержится описание всех доступных опций компилятора и подробно описана работа всех утилит. Большое внимание уделяется методам устранения ошибок и отладке не только компилируемых программ, но и также и самого компилятора. Рассмотрен пример добавления в GCC дополнительного языка верхнего уровня и описано использование с GCC ассемблеров и компоновщиков, предоставляемых поставщиками оборудования. Вы сможете найти практически любые сведения, достаточные не только для разрешения ваших проблем, но и для участия в разработке и поддержке самого компилятора GCC.

Книга будет полезна программистам-разработчикам и руководителям программных проектов. Администраторам и системным программистам, которым приходится заниматься переносом программного обеспечения и приложений. Пользователям, заинтересованным в использовании программ с открытым исходным кодом. И всем сторонникам развития движения по созданию свободно распространяемых программ.

С.Л.Скловская

Команды Linux. Справочник, 3-е изд., перераб. и дополненное

ISBN уточняется
70x100/16, тв. переплет
Выход из печати — I кв. 2004г.

Что представляют собой команды? Это — основа всех основ. Однако информация о них подчас рассредоточена в разных местах, не всегда отличается полнотой и удобством поиска, а иногда в иных дистрибутивах и вовсе отсутствует.

Книга «Команды Linux. Справочник. Издание второе, переработанное и дополненное» содержит свыше 500 команд: классических и новых, популярных и не очень, незаслуженно забытых или не нашедших у нас должного распространения. Все они расположены в алфавитном порядке. Кроме того, приведена полная классификация команд, позволяющая ознакомиться со всеми компонентами тематического раздела и в дальнейшем получить информацию о конкретных командах.

Второе издание справочника дополнено новыми командами, особое внимание уделено тем из них, что работают с изображениями. В описания многих команд внесены изменения. Справочник рассчитан на пользователей Linux без привязки к определенному дистрибутиву, на пользователей Free BSD и классических ОС Unix.

Болл Билл Хойт Даф и др.

Red Hat Linux. Настольная книга пользователя

ISBN 5-93772-105-5
70x100/16, тв. переплет
Выход из печати — IV кв. 2003г.

Книга Red Hat Linux. Настольная книга пользователя написана корифеями в области системного администрирования и программирования. Данная книга охватывает практически все аспекты работы с операционной системой Linux, причем подход к рассмотрению сугубо практический.

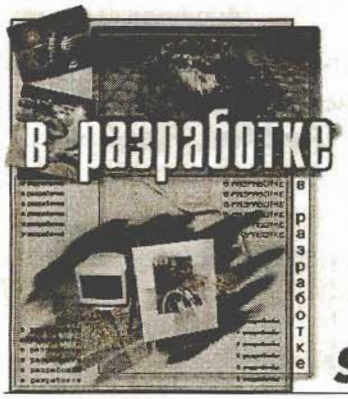
Первая часть книги позволит новичкам легко и быстро начать работать с Linux. Подробно освещается инсталляция и настройка системы, управление загрузкой, конфигурирование X Window System и основные пользовательские сервисы.

Большая часть книги посвящена системному администрированию и настройке системы. Даны практические рекомендации по организации печати, резервному копированию и восстановлению. Подробно изложена методика настройки и сборки ядра.

Большое внимание в книге уделено сетевым функциям Linux, в том числе использованию протоколов *TCP/IP и сетевой файловой системы NFS, а также настройке Apache, DNS и сервер баз данных.*

В отличие от большинства книг, рассчитанных в основном на системных администраторов, в этой не забыты и простые пользователи. Вы найдете доходчивое описание работы с офисными пакетами, средствами мультимедиа и игровыми пакетами. Наконец-то Linux поселится и на домашних машинах!

Для широкого круга пользователей и системных администраторов Linux.



Болл Билл

Red Hat Linux. Энциклопедия пользователя

ISBN 5-93772-103-9
70x100/16, тв. переплет, 896с.

Назначение книги

В компьютерном мире время течет очень быстро: меняются релизы операционных систем, появляются новые приложения, усовершенствуется пользовательский интерфейс. Не является исключением и Linux. Однако есть то, что остается неизменным: фундамент, основные принципы и дух Unix. Если вы хотите разобраться в тонкостях работы вашей системы – эта книга для вас! Она охватывает практически все аспекты работы с операционной системой Linux.

Отличительная особенность этой книги – универсальность. Вы можете пользоваться ею при работе с системой Linux версий 7, 8 и 9. Здесь описаны принципы, общие для всех версий.

Еще один важный аспект состоит в том, что в качестве приложения к книге вы можете заказать в издательстве CD-ROM с любой из этих операционных систем.

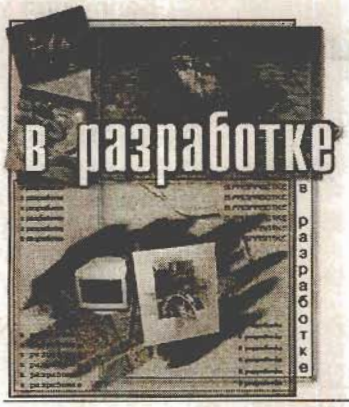
Первая часть книги поможет вам начать работать с Linux. Подробно освещается инсталляция и настройка системы, управление загрузкой, конфигурирование системы X Window System, все основные пользовательские сервисы.

Вторая часть посвящена сетевым возможностям Linux. Книга поможет разобраться в тонкостях протоколов TCP/IP и FTP, организовать для пользователей прозрачный доступ к удаленным файловым системам с помощью NFS и наладить сетевую информационную службу NIS. Вы узнаете, как подключиться к Internet, наладить sendmail, настроить серверы Apache, Samba и DNS.

Вторая половина книги раскрывает тайны системного администрирования. Здесь собрано все, что должен знать грамотный администратор: организация файловой системы, управление учетными записями пользователей, поддержка программного обеспечения, меры по защите данных. Кроме того, даны практические рекомендации по организации печати, резервному копированию и восстановлению системы. Подробно изложена методика настройки и сборки ядра.

Приятная изюминка этой книги – масса полезных ссылок.

Для широкого круга пользователей и системных администраторов Linux.



Анонимный автор

Максимальная безопасность Linux/Unix-систем: SuSE, debian, Red Hat, FreeBSD...

ISBN уточняется
70x100/16, мягк. переплет, 400 с.

Назначение книги

Это уникальная книга, существенно отличающаяся от наводнивших рынок многочисленных изданий, посвященных системному администрированию Linux, UNIX и сетей на базе протокола TCP/IP. Она написана не просто специалистом по компьютерной безопасности и системному администрированию, а опытным хакером, имеющим большой опыт взлома современных информационных систем. Поэтому даже такие, казалось бы подробно описанные в литературе темы, как установка операционной системы Linux/Unix, управление пользователями или конфигурирование популярных сетевых служб, открываются читателю в новом ракурсе — как шаги, обеспечивающие существенную защиту системы от взлома.

Кроме стандартных задач системного администрирования Linux, автор подробно рассматривает:

- методы и приемы, позволяющие устранить факторы, угрожающие защите компьютера или сети под управлением ОС Linux/Unix;
- основные дыры в защите *различных* дистрибутивов системы Linux и *популярных приложений*, методы взлома и используемые взломщиками программные средства;
- коммерческие и свободно распространяемые программы и утилиты для выявления и устранения слабых мест в защите;
- безопасное конфигурирование и использование популярных приложений и служб на базе ОС Linux/Unix, в том числе telnet, Sendmail, Apache, FTP и многих других;
- применение технологий SSL, различных алгоритмов шифрования и вычисления контрольных сумм для обеспечения защиты и целостности данных при передаче;
- безопасность самих дополнительных средств защиты, возможные ограничения их применения.

Уникальное руководство, написанное хакером, по защите сервера и рабочей станции Linux

Эта книга предназначена прежде всего *для системных администраторов*, но пригодится и *обычным пользователям* ОС Linux/Unix, желающим защитить свои компьютеры и обеспечиваемые ими сетевые службы от вторжения и возможных угроз извне. Даже опытные администраторы UNIX будут шокированы, узнав, сколько узких мест в защите имеют их Linux-системы при стандартной установке, и к чему эти дыры могут привести при подключении компьютера к Internet или к локальной сети. Реализация же предлагаемых в книге рекомендаций позволяет быстро и зачастую без дополнительных финансовых затрат добиться уровня защиты Linux/Unix, достаточного для работы критически важных приложений *в производственной среде*, и обеспечить безопасную работу основных служб Internet.

SAMS

Томас Шенк

Red Hat Linux для системных администраторов. Энциклопедия пользователя

ISBN 966-7393-78-X
70x100/16, тв. переплет, 672 с.

Назначение книги

Книга является единственным источником важнейшей информации, необходимой системному администратору для управления любой средой, включающей компьютеры с операционной системой Red Hat Linux. С помощью этого всеобъемлющего руководства можно получить практический опыт и глубокие знания, позволяющие в полной мере использовать мощнейшие средства системного администрирования Red Hat Linux. Автор долго работал ведущим системным администратором в компании Deja.com, где управлял одним из крупнейших Web-сайтов в Internet на базе Linux. Он делится своим уникальным опытом, рассматривая системное администрирование крупномасштабных сетей на базе Linux.

Книга адресуется:

- пользователям Red Hat Linux, желающим лучше узнать средства системы и изучить основы системного администрирования;
- системным администраторам, обслуживающим большие сети на базе ОС Red Hat Linux;
- специалистам по созданию гетерогенных корпоративных сетей.

В книге рассматривается

- работа в сетевой среде, включающей операционные системы NT, NetWare и UNIX;
- управление дисками и файловыми системами;
- резервное копирование и аварийное восстановление;
- конфигурирование оконной системы X Window System;
- командные интерпретаторы и создание сценариев;
- контроль и настройка системы;

С помощью этого всеобъемлющего руководства можно получить практический опыт и глубокие знания, позволяющие в полной мере использовать мощнейшие средства системного администрирования Red Hat Linux.

- настройка и безопасная эксплуатация сетевых служб.

Представлена подробная информация о том, как:

- оперативно решать проблемы, возникающие в Red Hat Linux;
- создавать шаблоны установки;
- выявлять и устранять сбои в системе и компонентах оборудования;
- эффективно управлять использованием дискового пространства с помощью квот;
- использовать технологию RAID для повышения эффективности, надежности и производительности системы;
- определить, какие резервные копии необходимо создавать и как это лучше всего делать;
- выбрать лучшие носители информации и внешние устройства для своей среды;
- предотвратить продолжительные простои путем разработки плана аварийного восстановления;
- расширить спектр возможностей за счет интеграции Linux с другими сетевыми операционными системами;
- выработать и реализовать эффективные процедуры защиты для потребностей бизнеса;
- определить необходимые действия в случае взлома системы.

Стивен Прата

Язык программирования С. Лекции и упражнения. Учебник

ISBN 5-93772-049-0
70x100/16, мягкий переплет, 896 стр.

SAMS

Книга Стивена Праты «Язык программирования С. Лекции и упражнения. Учебник» — подробный учебник по языку С, построенный на основе принципа «от простого к сложному». В начальных главах описываются элементарные конструкции языка (операторы, выражения и т.д.), в последних главах рассматриваются сложные структуры данных. Здесь вы найдете полное описание синтаксиса языка, основных приемов и методов программирования, а также примеры реальных коммерческих программ. Все примеры протестированы, а исходные коды программ можно найти на web-сайте www.diasoft.kiev.ua.

Рассмотрены все нововведения, привнесенные в язык С новейшим стандартом С99. Причем теоретические положения иллюстрируются конкретными практическими примерами.

Книга будет полезна начинающим и опытным программистам, использующим язык программирования С в своей повседневной деятельности.

Краткое оглавление

Глава 1. Готовимся изучать язык С
Глава 2. Введение в язык С
Глава 3. Представление данных в языке С
Глава 4. Строки символов и форматированный ввод/вывод
Глава 5. Операции, выражения и операторы
Глава 6. Управляющие операторы языка С: циклы
Глава 7. Управляющие операторы языка С: ветвление и безусловные переходы
Глава 8. Символьный ввод/вывод и перенаправление
Глава 9. Функции
Глава 10. Массивы и указатели
Глава 11. Символьные строки и строковые функции
Глава 12. Классы хранения, связывание и управление памятью
Глава 13. Файловый ввод/вывод
Глава 14. Структуры и другие формы данных
Глава 15. Дополнительные сведения о битах
Глава 16. Препроцессор и библиотека языка С
Глава 17. Расширенное представление данных
Приложение А. Ответы на контрольные вопросы
Приложение В. Справочник по языку С
Глоссарий. Словарь терминов по языку С
Предметный указатель

Стивен Прата

Язык программирования C++. Лекции и упражнения. Учебник

ISBN 5-93772-073-3

70x100/16, тв. переплет, 1104 стр.

SAMS

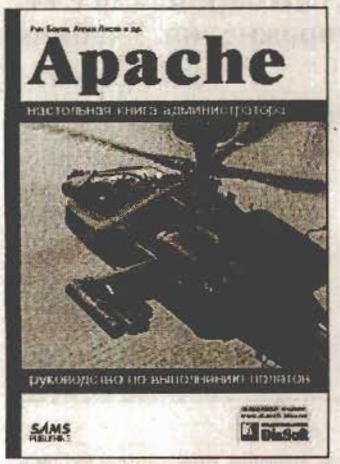
В новом издании популярной книги известного автора и специалиста по C++ рассматривается устоявшаяся версия языка, соответствующая стандарту ISO/ANSI. Изложение материала не привязано ни к какой конкретной реализации C++. В основе изложения материала лежит оригинальная методика, разработанная автором и неоднократно проверенная на различных группах пользователей. Книга охватывает обширный круг вопросов, необходимых для создания полноценных программ на C++, — от знакомства с основами синтаксиса языка до многочисленных новых функциональных возможностей C++, таких как классы, объекты, наследование, полиморфизм, виртуальные функции, стандартная библиотека шаблонов STL, RTTI и др.

Практические задания к главам книги составлены на основе реальных проектов. Издание рассчитано на пользователей с любым уровнем подготовки. Для начинающих эта книга послужит превосходным учебником, а опытные программисты смогут воспользоваться ею в качестве справочника, облегчающего их ежедневный труд.

Книгу можно рекомендовать как методическое пособие преподавателям высших и средних учебных заведений, а также в качестве учебника для студентов.

Краткое оглавление

Глава 1. Начальные сведения
Глава 2. Приступаем к изучению языка C++
Глава 3. Представление данных
Глава 4. Составные типы данных
Глава 5. Циклы и выражения сравнения
Глава 6. Операторы ветвления и логические операции
Глава 7. Функции языка C++
Глава 8. Работа с функциями
Глава 9. Модели памяти и пространства имен
Глава 10. Объекты и классы
Глава 11. Работа с классами
Глава 12. Классы и динамическое распределение памяти
Глава 13. Наследование классов
Глава 14. Повторное использование программного кода в C++
Глава 15. Дружественные конструкции, исключения и прочее
Глава 16. Класс string и стандартная библиотека шаблонов
Глава 17. Ввод/вывод данных и работа с файлами
Приложение A. Системы счисления
Приложение B. Ключевые слова языка C++
Приложение C. Таблица кодов ASCII
Приложение D. Приоритет операций
Приложение E. Другие операции
Приложение F. Шаблон класса STRING
Приложение G. Методы и функции библиотеки STL
Приложение H. Рекомендуемая литература
Приложение I. Преобразование программного кода в соответствии со стандартом ANSI/ISO C++
Приложение J . Ответы на вопросы для повторения



Рич Боуэн

Apache. Настольная книга администратора

ISBN 5-93772-055-5
70x100/16, тв. переплет, 384 с.

Настольная книга администратора Apache — это практическое руководство по инсталляции, конфигурированию и администрированию самого популярного Web-сервера. Она поможет установить сервер с нужными вам модулями и современными функциональными возможностями и станет основой для дальнейшего профессионального совершенствования.

Системные администраторы почерпнут здесь много полезного об особенностях настройки виртуальных хостов, работе с CGI-сценариями, серверными расширениями, модулем Perl и PHP. Web-дизайнеры найдут в книге массу полезной информации о создании динамического контента, индивидуального для каждого посетителя сайта.

Книга расскажет, что следует предпринять, чтобы Web-сервер работал устойчиво и был неуязвим для внешних атак. Много внимания в ней уделяется настройке производительности, анализу log-файлов и созданию всевозможных статистических отчетов.

Подробно описаны стандартные модули Apache и модули специального назначения. Это удобный в повседневной работе справочник, предназначенный для широкого круга пользователей и администраторов сервера Apache.

Краткое оглавление

Часть I. Инсталляция и конфигурирование сервера Apache
Глава 1. Начало работы
Глава 2. Инсталляция сервера Apache
Глава 3. Запуск, останов и повторный запуск сервера
Глава 4. Директивы конфигурирования
Глава 5. Утилиты конфигурирования
Глава 6. Файлы конфигурации .htaccess
Глава 7. Виртуальные хосты
Часть II. Современные технологии конфигурирования
Глава 8. MIME и типы файлов
Глава 9. Отображение URL-адресов
Глава 10. Согласование контента
Глава 11. Индексация каталога
Глава 12. Сервер Apache на платформе Microsoft Windows
Глава 13. Настройка производительности
Часть III. Динамический контент
Глава 14. Обработчики и фильтры
Глава 15. CGI-сценарии
Глава 16. Серверные расширения (SSI)
Глава 17. Модуль mod_per
Глава 18. PHP
Часть IV. Безопасность и мониторинг
Глава 19. Защита сервера Apache
Глава 20. Динамический контент и проблемы безопасности
Глава 21. Аутентификация, авторизация и управление доступом
Глава 22. SSL
Глава 23. О пользе и вреде Web-насекомых
Глава 24. Ведение журналов
Часть V. Модули
Глава 25. Общие сведения о модулях Apache
Глава 26. Модули, поставляемые с Apache
Глава 27. Модули Apache специального назначения
Часть VI. Приложения
Приложение A. Лицензия на Apache
Приложение B. Опции командной строки сценария конфигурации
Приложение C. Регулярные выражения
Приложение D. Пример кода для модуля mod_perl
Приложение E. История Apache
Приложение F. Где найти дополнительную информацию об Apache
Предметный указатель

Андре Лесса

Разработка Web-приложений на Python. Руководство разработчика Web-сайтов

ISBN 5-93772-010-5

70x100/16, тв. переплет, 688 с.

Python — мощный интерпретируемый объектно-ориентированный язык программирования высокого уровня, идеально приспособленный для Web. Он развивается в рамках проекта с открытым исходным кодом и предоставляется бесплатно для всеобщего пользования.

Среда программирования Python реализована на многих платформах, включая Microsoft Windows, UNIX/Linux и Macintosh. Исходный код интерпретатора и библиотек Python на языке C++ предоставлен для общего пользования.

Еще одна реализация этого языка, получившая название JPython и также общедоступная, написана на языке Java.

Благодаря этим особенностям язык программирования Python открывает путь ко многим программным технологиям, поэтому разработки на этом языке ведутся во многих направлениях. К наиболее интересным и перспективным из них относится:

- создание серверов и клиентов OLE, COM, DCOM и ActiveX для предоставления удобных сценарных средств в среде Windows (пакет win32com, проект Python for Delphi и др.);
- упрощенная поддержка распределенных объектов в рамках технологий CORBA, ILU, ADO, DCOM (проекты CORBA IDL Parser, Fnorb, OMF, Hector и др.);
- интерпретация исходного кода таких языков, как Java, которые в основной реализации поддерживают только компиляцию (проект JPython и др.);

Книга содержит все, что необходимо разработчику, использующему язык программирования Python

- разработка более дружественных, альтернативных интерфейсов к существующим библиотекам объектных модулей, таким как MFC, Tk, KDE, Motif и многим другим (проекты Pythonwin, wxPython, PyKDE, PyOpenGL и пр.);
- создание объектно-ориентированных интерфейсов к базам данных (проект Zope, MetaKit, модуль shelve);
- реализация сложных структур данных, включая многомерные массивы, поддержка графических и других двоичных файловых форматов;
- обработка данных, в том числе с использованием таких языков разметки, как XML, SGML, HTML, XDR.

Все эти направления разработки, а также многие другие подробно рассматриваются в книге "Python. Энциклопедия программиста". В ней даны многочисленные примеры, ссылки на все существующие источники в Web.

Книга содержит все, что необходимо разработчику, использующему язык программирования Python: дано описание синтаксиса, основных средств и модулей, среды разработки, способов оптимального использования мощных средств языка.

Дэвид М. Бизли

Язык программирования Python. Справочник

ISBN 966-7393-54-2

70x100/16, 336 с., мягк. переплет

Язык программирования Python получил широкую известность сравнительно недавно, но теперь его достоинства начинают признавать очень многие.

Этот язык привлекает программистов:

- Является интерпретируемым языком, поэтому он широко применяется для создания сценариев операционной системы (например, в Linux и UNIX). Сценарии, написанные на языке Python, практически не требуют отладки, поэтому он является повседневным инструментом системных администраторов. Он широко применяется для создания сценариев CGI (на основе модуля mod_python для сервера Apache). На этом языке пишутся сценарии для многих известных программных комплексов, например таких, как Delphi, COM+, MFC, Java, поскольку он значительно упрощает работу с этими программными комплексами.
- В программах, написанных на этом языке, можно легко разобраться, поскольку он имеет очень простой синтаксис. Строки документирования программ можно вызывать для справки непосредственно в интерпретаторе, строить на их основе документацию к программам.
- Это объектно-ориентированный язык, вплоть до уровня структур данных. В связи с этим работа программиста значительно упрощается. Модульные, не зависящие друг от друга программные компоненты можно легко отлаживать отдельно, а затем объединять в единый комплекс. Для любых, самых сложных структур данных предусмотрен оптимальный набор методов. Об их создании программисту заботиться не приходится.

Первая книга на русском языке, посвященная интерпретируемому, интерактивному, объектно-ориентированному языку высокого уровня с открытым исходным кодом

- В языке Python структура программы обозначается отступами. Его программы эстетически привлекательны: никаких фигурных скобок, которые иногда так трудно согласовать между собой (C, C++), никаких режущих глаз знаков доллара, обратного слэша, амперсанда, в нагромождении которых просто теряешься (Perl).

Этот язык привлекает математиков:

- Поддерживает целые числа неограниченной длины (единственным ограничением является объем памяти компьютера). В нем есть поддержка комплексных чисел, встроены мощные функции работы с последовательностями (например, map, filter, reduce, zip), предусмотрена анонимная функция (lambda).
- На этом языке удобно проводить эксперименты со структурами данных, реализовать сложные алгоритмы, проверять гипотезы. Например, в модуле AVL реализована разработанная Адельсоном-Вельским структура, которая одновременно является сбалансированным деревом и списком. В модуле PLEX по исходным данным пользователя осуществляется синтез недетерминированного, а затем детерминированного конечного автомата, который применяется в качестве лексического анализатора.

Робин Бурк и др.

Unix для системных администраторов. Энциклопедия пользователя

ISBN 966-7393-07-0

60x84/8, тв. переплет, 864 с.

Книга представляет собой полное руководство по установке, сопровождению и администрированию основных вариантов UNIX — SVR4, BSD, Linux, HP-UX, AIX, IRIX, Solaris и SunOS. Подробно рассматриваются вопросы, связанные с инсталляцией и настройкой, конфигурированием ядра, управлением производительностью, а также работой с файловой системой со стороны пользователя и администратора.

В первых четырех главах раскрываются базовые понятия Unix: концепция этой операционной системы, регистрация в системе, учетные записи пользователей, файловые системы, настройка переменных среды, файлы инициализации.

В глава 5 представлены сто команд, необходимых для того, чтобы приступить к работе в Unix.

Работа в сети — тема главы 6. Помимо знакомства с основными понятиями и некоторыми сетевыми службами Internet, читатель узнает, как соединяться с удаленными узлами, передавать файлы. Электронная почта и Usenet (служба новостей) рассматриваются в главе 7, а вопросы администрирования этих служб — в главах 24-25. В главе 7 также описывается передача аудио- и видеоданных по Internet.

Главы с 9 по 13 посвящены оболочкам Unix, написанию сценариев. В главе 13 проводится сравнительный анализ наиболее популярных из этих оболочек.

Рассмотрены отдельные важные аспекты и представлена целостная картина работы Unix. Материал ориентирован на решение практических задач!

И если первые тринадцать глав издания носят общий характер, то часть III полностью посвящена администрированию системы. Здесь рассмотрены вопросы инсталляции Unix, запуска и останова системы, управления пользователями.

Конфигурирование ядра, управление вводом/выводом, типы файловых систем — темы главы 19. О том, как работать в сети, готовить к установке и настройке TCP/IP, работать с DNS, читатель узнает из главы 20. Здесь также описывается поиск и устранение сетевых проблем: неполадки системы доменных имен, NFS.

Уделяется внимание учету ресурсов и производительности системы. Эти проблемы раскрываются в главах 21-22.

Выполнение резервного копирования и восстановления описывается в главе 28.

КНИЖНЫЙ КЛУБ «КОМП@С»
(ОФИЦИАЛЬНЫЙ ПАРТНЕР ТОРГОВО-ИЗДАТЕЛЬСКОГО ДОМА «ДС»)

Приглашаем всех желающих вступить в книжный клуб «Комп@с». Вступив в клуб, Вы получите скидки, содействие в поиске и покупке необходимых Вам книг. Мы своевременно реагируем на запросы читателей и своей издательской деятельностью способствуем их профессиональному росту. Вы раньше других будете иметь возможность познакомиться с новинками, получая наш прайс-лист, иллюстрированный каталог по почте либо обратившись на наш сайт www.diasoft.kiev.ua.

Вы сможете:
- пользоваться системой скидок при покупке книг;
- высказать свое мнение, пожелания и замечания относительно книг издательства;
- найти новых друзей и партнеров по интересам;
- заочно встретиться и пообщаться с авторами книг;
- участвовать в розыгрыше призов бесплатной лотереи.

И главное, впервые членам клуба будет предоставляться бесплатное комплексное программное обеспечение, позволяющее осуществлять связь с книжной базой данных, получать оперативные изменения в базе, компоновать заказы, получать информацию о книгах издательств – объем, содержание (краткое или глобальное), аннотация, текст одной из глав, внешний вид обложки и т.д. С более подробной информацией Вы можете ознакомиться на сайте www.diasoft.kiev.ua

ПРИЕМ В КЛУБ

Для того чтобы стать членом книжного клуба, достаточно купить одну из книг нашего издательства в любой торговой точке, заполнить анкету, помещенную в книге, и отправить ее по адресу издательства либо купить три любых книги на одном из торговых мест ТИД «ДС».

Все члены клуба получают карточку с индивидуальным номером, дающую право на скидку.

Карточка выдается сразу же при покупке на торговом месте ТИД «ДС» или у партнеров ТИД «ДС» (адреса смотри на следующей странице). В остальных случаях (заказ по почте, через электронный магазин, а также покупка у независимых торговых организаций книги ТИД «ДС») Вам необходимо заполнить анкету, помещенную в книге, и отправить ее в издательство. Действие карточки распространяется на все торговые точки, перечисленные в этом пункте, кроме независимых торговых организаций.

СИСТЕМА СКИДОК

Став членом книжного клуба, Вы получаете первоначальную скидку в размере 5% на весь ассортимент книг *(книги ТИД «ДС» + более 1000 наименований книг издательств Украины и стран СНГ).*

Карточка действительна на протяжении года со дня выдачи; по истечении срока подлежит замене.

При покупке в течение квартала пяти и более книг карточка заменяется на новую с 10% скидкой.

Более подробную информацию о системе скидок можно получить, обратившись на наш сайт по адресу www.diasoft.kiev.ua.

АНКЕТА

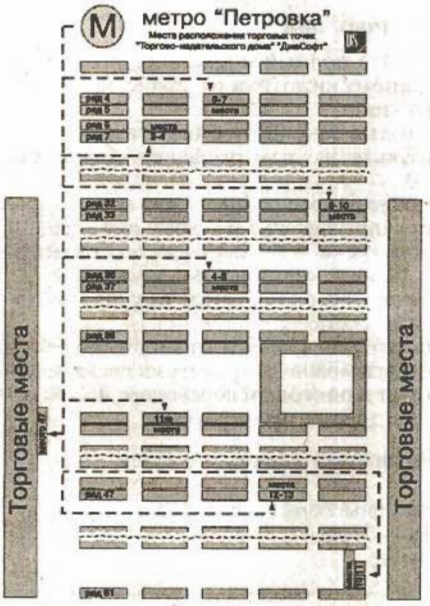
1. Название книги _____
2. Где, по какой цене, когда (с точностью до месяца) Вы приобрели эту книгу?_____
_____
3. Ф.И.О. _____
4. Кем и где работаете _____
5. *Параметры Вашего ПК* _____
6. Адрес, телефон, e-mail, web-узел _____

Дата заполнения "___" _____ _____ г.

Вышлите анкету по адресу: Украина, 03055, Киев-55, а/я 100, «Издательство «ДиаСофт».

ТОРГОВЫЕ МЕСТА ТОРГОВО-ИЗДАТЕЛЬСКОГО ДОМА «ДС»

КИЕВ

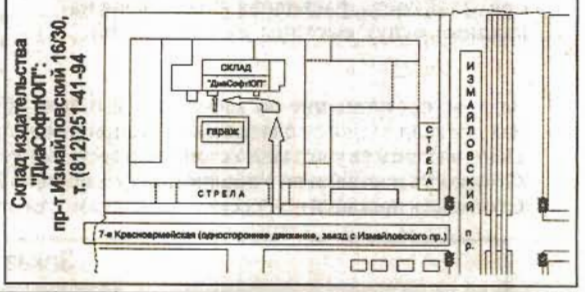


ХАРЬКОВ

ДНЕПРОПЕТРОВСК

САНКТ-ПЕТЕРБУРГ

пр-т Обуховской обороны, 105, ДК им. Крупской, м."Елизаровская"
190103. пр-т Лермонтовский, 44, а/я 66, ООО "ДиаСофтЮП"



ПАРТНЕРЫ ТИД «ДС»

Киев
т./ф. 212-12-54, 216-35-64
Книга-почтой: 03055, а\я 100
e-mail: books@diasoft.kiev.ua
e-mail: bss@diasoft.kiev.ua
e-mail: stepanb@akcecc.kiev.ua

Днепропетровск
т.(0562)33-27-74, 34-02-09
Книга-почтой: 49008, а\я 466
e-mail: diasoft@mail.dnepr.net

Харьков
т. (0572)47-20-67
e-mail: books@rail.kharkov.com

Львов
т./ф. (0322)39-87-08
e-mail: vlas@txnet.com

Москва
т. (095)726-80-67
e-mail: diasoft_msk@rosmail.ru

Санкт-Петербург
т. (812)251-41-94
e-mail: diasoft_spt@mail.convey.ru
Индекс 190103, а\я №66
ООО "ДиаСофт ЮП"

АДРЕСА МАГАЗИНОВ, ГДЕ МОЖНО КУПИТЬ КНИГИ ТИД «ДС»

КИЕВ
"Знання", ул. Крещатик, 46
т. (044)224-22-91
"Техническая книга",
ул.Красноармейская,51 т. (044)227-25-86
"Сучасник", пр-т Победы, 29
т. (044)274-52-35
"Библиотечный коллектор",
пр-т 40-летия Октября, 100/2
т. (044)263-20-54, 263-20-04
тел./факс (044)263-60-56

ДНЕПРОПЕТРОВСК
"Технічна книга",
пл.Островского, 1
т. (0562)33-09-55
"Техническая книга"
пр.К.Маркса 40
т. (0562)744-86-72
"Книги", пр-т Гагарина, 98,
т. (0562)43-63-81

ДОНЕЦК
ЧП "ИнфоКом"ул.Артема 83
т. (0622)382-64-69

КРИВОЙ РОГ
"Букинист-Солон"
пл.Освобождения 1
т. (0564)92-37-32

ЛЬВОВ
"Техническая книга", пл.Рынок,10
т. (0322)72-54-06
"Влас", Университет Львовский
политехник, корпуса 4,5,
т. (0322)39-8708

ХАРЬКОВ
"Вища школа",ул.Петровского,6
т. (0572)47-80-20
"Books", ул. Сумская, 51,
т. (0572)14-04-71

МОСКВА
"Библио-Глобус", ул.Мясницкая,6
т. (095)928-87-44
"Дом технической книги",
Ленинский пр-т, 40, т. (095)137-60-38
"Московский дом книги",
ул.Новый Арбат, 8
"Мир", Ленинградский пр-т, 78
т. (095)152-45-11
"Мир печати" ул.2-я Тверская Ямская,54
т. (095)978-50-47, 978-55-07
"Молодая гвардия", ул.Большая Полянка,28
т. (095)238-11-44, 238-00-32

САНКТ-ПЕТЕРБУРГ
"Дом книги", Невский пр-т, 28
т. (812)318-64-16
"Техническая книга",
ул. Пушкинская, 2,
т. (812)164-65-65
"Энергия", Московский пр-т, 189
т. (812)443-01-47

МИНСК
"Книга XXI"
пр-т Ф.Скорины, 92,
ст.м. Московская,
т. (0172)64-31-05, 64-27-97

Как сделать заказ и получить книги

Для организаций

1) Получите полный прайс-лист по адресу, указанному ниже.
2) Аккуратно заполните бланк заказа.
3) Отправьте его нам по факсу, почте или e-mail.
4) В течение одного рабочего дня в Ваш адрес будет направлен счет по факсу или e-mail. В течение срока действия счета мы гарантируем наличие книг и неизменность цены.
5) После оплаты счета Ваш заказ, а также оригинал счета, расходная и налоговая накладные будут высланы Вам посылкой.

Для частных лиц

1) Получите полный прайс-лист по адресу, указанному ниже (или см. прайс, приведенный в книге)
2) Аккуратно заполните бланк заказа
3) Отправьте его нам по факсу, почте или e-mail.
4) В течение одного рабочего дня в Ваш адрес будет направлен счет по факсу или e-mail. В течение срока действия счета мы гарантируем наличие книг и неизменность цены. ***Внимание!*** Не оплачивайте покупку до получения счета.
5) Оплатите счет в любом коммерческом банке. Деньги можно отправить на наш расчетный счет и почтовым переводом, но это несколько дороже, чем через банк.

Заказы, поступившие по электронной почте, обрабатываются в первую очередь.
Заказы «наложенным платежом» — не принимаются.
Цены на книги не учитывают стоимость доставки, коммисионные банка или почтового перевода.
Стоимость доставки по Украине или курьером по Киеву — **7 грн.**
Стоимость доставки по России — в зависимости от региона.

Заказ

от «_____» _____ 200__ г.

Название организации: _____
Для организаций

Фамилия Имя Отчество: _____
Для частных лиц

Адрес доставки: _____ _____
Почтовый индекс

_____

ИНН: _____ № св-ва плательщика НДС: _____
Только для организаций

Фамилия Имя Отчество: _____
Контактного лица

Телефон: / _____ / _____ Факс: _____

E-mail: _____

Код	Название книги	К-во

Получайте полный прайс-лист на все книги и направляйте заказы по адресу:
г.Киев : 03055 а/я 100, тел./факс (044) 212-1254, 216-3564 e-mail: books@diasoft.kiev.ua
stepanb@akcecc.kiev.ua

Индекс 190103, А/Я №66 ООО "ДиаСофт ЮП"

www.diasoft.kiev.ua — всегда полный ассортимент наших книг

Прайс-лист ТиД "ДиаСофт"

Код	Автор	Название	Стр.	Формат
		Использование ПК в целом.		
4043	Нортон П., Гуд	Работа на персональном компьютере. Самоучитель	584	84x108/16
7277	Клименко	Эффективный самоучитель работы на ПК. Основной курс 2-издание	496	70x100/16
6173	Клименко	Эффективный самоучитель работы на ПК.	736	70x100/16
7261	Михлин Е.М.	Эффективный самоучитель работы на ПК. 2-издание	624	70x100/16
7276	Вебер	Сборка, конфигурирование, настройка, модернизация и разгон ПК 5-и	704	70x100/16
6169	Домарев	Безопасность информационных технологий	688	84x108/16
6583	Брукс Чарльз Д	Аттестация А+. Техник по обслуживанию ПК. Организация, обслужива	816	84x108/16
4020	Нортон П., Гуд	Внутренний мир персональных компьютеров, 8е издание. Избранное (	548	84x108/16
4683	Канер Сэм и др	Тестирование программного обеспечения	544	60x84/16
6538	Макгрегор	Тестирование объектно-орентированого программного обеспечения. [	432	70x100/16
		Операционные системы.		
5511	Кессел Пол	Microsoft Windows 2000 Professional. Энциклопедия пользователя	832	70x100/16
5954	Браун Т.	Microsoft Windows 2000 Server. Энциклопедия пользователя	672	70x100/16
6521	Чуприн А.И.	Эффективный самоучитель работы в Windows XP Professional	336	70x100/16
6884	Оуглтри Терри	Microsoft Windows XP	848	70x100/16
6455	Анонимный ав	Максимальная защита в Linux. Второе издание	752	70x100/16
6523	Скловская	Команды Linux. Справочник.2-е изд. переработанное и дополненное.	720	70x100/16
5717	Шенк Т.	Red Hat Linux для системных администраторов. Энциклопедия пользо	672	84x108/16
6174	Болл, Питтс	Red Hat Linux 7 в офисе и дома	448	70x100/16
5758	Сэтчэлл Стеф	Linux IP Stacks в комментариях (+ CD-ROM)	288	70x100/16
3926	Паркер Тим	Linux 5.2. Энциклопедия пользователя + 2 CD-ROM	688	84x108/16
6718	Болл Билл, Пи	Red Hat Linux 7.x Энциклопедия пользователя	880	70x100/16
1557	Бурк Р., Хорва	Unix для системных администраторов. Энциклопедия пользователя (	864	60x84/8
4318	Бурк Робин,Хо	UNIX для Internet . Энциклопедия пользователя (+ CD-ROM)	496	84x108/16
44	Кепли М.	Ответы на актуальные вопросы по OS/2 Warp	352	60x84/16
6496	Эбен Майкл	FreeBSD. Энциклопедия пользователя	736	70x100/16
6982	Билл Кэлкинс	Solaris 8. Сертификация системного администратора	928	70x100/16
8345	Пэкер Алан Н.	Конфигурирование и настройка баз данных на платформе SolarisTM и	512	70x100/16
		Программирование.		
5003	Вильямс М.	Программирование в Windows 2000. Энциклопедия пользователя. (+	640	84x108/16
6522	Саймон	Windows 2000 API. Энциклопедия программиста 2-е издание (+ CD-R	1088	70x100/16
5953	Гриффитс А.	Прогаммирование GNOME/GTK+. Энциклопедия программиста (+ C	720	70x100/16
6704	Майо Джозев	C#. Искусство программированияю Энциклопедия программиста	656	70x100/16
6894	Микелсен Клау	Язык программирования C#. Лекции и упражнения	912	84x108/16
8344	Смайли Джон	Учимся программировать на C# вместе с Джоном Смайли	528	70x100/16
6905	Гилберт Стиве	Самоучитель Visual C++6 в примерах	496	70x100/16
7167	Олафсен	Visual C++6 и MFC Энциклопедия пользователя	992	70x100/16
6913	Прата С.	Язык программирования C. Лекции и упражнения	896	84x108/16
7464	Седжвик Робер	Фундаментальные алгоритмы на C.Части 1-4.Анализ,структуры данны	672	70x100/16
7463	Седжвик Робер	Фундаментальные алгоритмы на C.Части 1-5.Анализ,структуры данны	1136	70x100/16
7465	Седжвик Робер	Фундаментальные алгоритмы на C.Часть 5.Алгоритмы на графах.	480	70x100/16
5804	Хэзфилд Р.	Искусство программирования на C. Фундаментальные алгоритмы, стр	736	84x108/16
8326	Смайли Джон	Учимся программировать на C++ вместе с Джоном Смайли	560	70x100/16
7168	Прата С.	Язык программирования C++. Лекции и упражнения	656	84x108/16
6885	Седжвик Робер	Фундаментальные алгоритмы на C++.Анализ/ Структуры данных/Сорт	688	70x100/16
6914	Седжвик Робер	Фундаментальные алгоритмы на C++.Алгоритмы на графах.	496	70x100/16
4754	Либерти ,Джес	C++ Энциклопедия пользователя с приложением	584	84x108/16
3621	Калверт Ч.	Delphi 4. Самоучитель (без приложения)	192	84x108/16
2757	Калверт Ч.	Delphi 4. Энциклопедия пользователя (+ CD-ROM)	400	84x108/16
93	Конопка Р.	Создание оригинальных компонент в среде Delphi	512	60x84/16
5845	Кандзюба С.П.	Delphi 6/7. Базы данных и приложения. Лекции и упражнения	576	70x100/16
7469	Бакнелл Джули	Фундаментальные алгоритмы и структуры данных в Delphi	560	70x100/16
7078	Глинский Я.Н.	Бейсик. Qbasic и Visual BaSIC	192	60x84/16
8038	Поль Киммел	Visual Basic .NET. Искусство программирования.	720	70x100/16
6964	Зеленяк О.П.	Практикум программирования на Turbo Pascal. Задачи, алгоритмы и р	320	60x84/16
6103	Глинский Я.Н.	Turbo Pascal 7.0 и Delphi. Учебное пособие	208	60x84/16
7063	Голубь Н.Г.	Искусство программирования на Ассемблере. Лекции и упражнения	656	70x100/16

6148	Лейнекер	COM+ Энциклопедия программиста + CD-ROM	656	70x100/16
5799	Клименко	Kylix 1.0. Базы данных и приложения. Лекции и упражнения	288	70x100/16
8263	Сергиенко А.М	VHDL для проектирования вычислительных устройств	208	60x84/16
		Офисные пакеты.		
7251	Линд Дебби	Lotus Notes и Domino 5\6 Энциклопедия программиста.	1024	70x100/16
4303	Нортон Питер (	Microsoft Office 2000.Избранное от Питера Нортона	552	84x108/16
6622	Кишик А.	Office XP. Эффективный самоучитель. Быстро...просто...наглядно	432	70x100/16
		Текстовые редакторы.		
6454	Кишик А.Н.	Word 2002. Эффективный самоучитель. Быстро...просто...наглядно	256	70x100/16
		Электронные таблицы.		
6104	Кишик А.Н.	Excel 2002. Эффективный самоучитель. Быстро...просто...наглядно	240	70x100/16
		Настольные базы данных.		
6623	Послед Б.С.	Access 2000. Приложения баз данных. Лекции и упражнения	656	70x100/16
5096	Форт С.,Хоуи Т	Программирование в среде Access 2000. Энциклопедия пользователя	544	84x108/16
		Базы данных клиент-сервер.		
4925	Бэлсон Дон и д	Внутренний мир Oracle8. Проектирование и настройка.(+ CD-ROM)	800	84x108/16
5057	Грин Джо	Oracle 8/8i Server. Энциклопедия пользователя.	576	84x108/16
6168	AIS	Oracle 8. Энциклопедия пользователя. Изд. 2-е, перераб., дополн. + C	864	60x84/8
6728	Дэн Хотка	Oracle 9i	560	70x100/16
7217	Кайт Том	Oracle для проффессионалов. Книга 1 Архитектура и основные особе	672	70x100/16
	Кайт Том	Oracle для проффессионалов. Книга 1.Расширение возможностей и за	848	70x100/16
6965	Сичкаренко В.А	SQL-99. Руководство разработчика баз данных.	816	70x100/16
5640	Бьелетич Шарс	MS SQL Server 2000. Энциклопедия пользователя	688	70x100/16
6729	Гешвинде Эва	PostgreSQL. Руководство разработчика и администратора	608	70x100/16
2010	Мак-Налли Д.	Informix. Энциклопедия пользователя	800	60x84/8
		Графика и анимация.		
2472	Боутон Г. и др.	Внутренний мир Adobe Photoshop 5 (+ CD-ROM)	496	84x108/16
6694	Кишик	Adobe Photoshop 7.0. Эффективный самоучитель	368	70x100/16
7262	Блатнер Дэвид	Adobe Photoshop 7.0. Искусство допечатной подготовки.	762	70x100/16
7218	Воробей Серге	Adob Illustrator 10 Эффективный самоучитель	400	70x100/16
7219	Мак-Клеланд	Adob Illustrator 10 Полное руководство	848	70x100/16
7148	Ковтанюк Ю.С.	Самоучитель CorelDRAW 11	528	70x100/16
7147	Ковтанюк Ю.С.	CorelDRAW 11 для дизайнера	1024	70x100/16
6171	Хаббелл, Борд	3D Studio VIZ 3 + CD-ROM	624	70x100/16
570	Эллиотт С.	Внутренний мир 3D Studio MAX. Том 1 (+ CD-ROM)	752	84x108/8
649	Эспиноза Д.	Внутренний мир 3D Studio MAX Том 2-й (+ CD-ROM)	432	84x108/16
2860	Бордмэн Т., Ха	Внутренний мир 3D Studio MAX 2. Том 2: Моделирование и материал	368	84x108/16
3925	Маэстри Джорд	Внутренний мир 3D Studio MAX 2. Том 3: Анимация (+ CD-ROM)	408	84x108/16
5080	Миллер Ф.	Внутренний мир 3D Studio MAX 3: моделирование,материалы визуали	720	84x108/16
4634	Бордмэн	Внутренний мир 3D Studio MAX 3 : Моделирование, материалы и визу	456	84x108/16
6925	Ли Ким	3D Studio MAX 4/5 для дизайнера. Искуство трехмерной анимации.	832	70x100/16
7278	Тёмин	3D Studio MAX 5. Эффективный самоучитель 2-издание	464	70x100/16
5029	Мортиер Р.	Внутренний мир Bryce 4 для дизайнера (+ CD-ROM)	336	70x100/16
6170	Китченс, Гавен	BRYCE для дизайнера + CD-ROM	656	84x108/16
5069	Мильберн Кер	Внутренний мир Flash 4 для дизайнера. (+ CD-ROM)	448	70x100/16
6094	Кишик А.Н.	Flash 5.0. Анимация. Эффективный самоучитель. Быстро...просто...на	240	70x100/16
6195	Дэн Аблан	LightWave 6/7 для дизайнера: Искусство трехмерного дизайна + CD-R	864	70x100/16
6915	Шрайнер Дейв	OpenGL.Официальный справочник	512	70x100/16
6803	Ву Мэйсон	OpenGL.Официальное руководство программиста	592	70x100/16
		Издательские системы.		
5163	Хансен Хан	Разработка сценариев для PageMaker (+ CD-ROM)	704	60x84/16
6566	Компания Adob	Page Maker 7.0 . Учебник от Adobe	384	70x100/16
7061	Блатнер Дэвид	QuarkXPress 5. Искусство допечатной подготовки.	912	70x100/16
6844	Анита Деннис	PDF и Adobe Acrobat	384	70x100/16
		CAD-пакеты.		
4753	Барчард Билл	Внутренний мир AutoCAD 2000. (+ CD-ROM)	688	84x108/16
6926	Чуприн А.И.	AutoCAD 2000/2002. Лекции и упражнения.	784	70x100/16
6780	Чуприн А.И.	AutoCAD 2002. Трехмерное проектирование. Лекции и упражнения.	528	70x100/16
6559	Харрингтон	AvtoCAD 2002 для конструкторов. Искусство проектирования	944	70x100/16
6235	Титаренко	Mechanical Desktop 4,5,6. Искусство трехмерного проектирования	304	70x100/16

		Сети, коммуникации и сетевые продукты.		
3947	Спортак М. и д	Компьютерные сети. Книга 1: High-Performance Networking. Энциклопе	432	60x84/8
3927	Спортак М. и д	Компьютерные сети. Книга 2: Networking Essentials. Энциклопедия пол	432	84x108/16
6384	Спортак Марк	Компьютерные сети и сетевые технологии	736	70x100/16
6843	Гринфилд Дев	Оптические сети	256	70x100/16
6785	Остерлох	TCP/IP.Семейство протоколов передачи данных в сетях компьютеров	576	70x100/16
6893	Микелсен Хизе	Маршрутизация в IP-сетях. Принципы, протоколы, настройка	512	70x100/16
1295	Грин Д. и др.	Microsoft BackOffice 2. Энциклопедия пользователя (+ CD-ROM).	800	60x84/8
		Использование Интернет.		
7462	Хан Харли	Эффективный самоучитель работы в Internet,2-е изд.	432	70x100/16
802	Акоста Н. и др.	Внутренний мир World Wide Web.	544	84x108/8
1032	Оливер Дик, Ф.	Популярные Web-броузеры. Энциклопедия пользователя (+ CD-ROM	460	60x84/8
6145	Киселев Ю.Н.	Электронная коммерция. Практическое руководство	224	60x84/16
6716	Эвод Илайес	Электронная коммерция. Практическое руководство	608	70x100/16
5258	Холден Грег,Уз	Apache Server в комментариях.(+ CD-ROM)	480	84x108/16
6951	Боузн Рич и др	Apache. Настольная книга администратора	384	70x100/16
		Программирование для Интернет.		
5165	Вайк Аллен	JavaScript в примерах	304	70x100/16
5355	Вайк А., Вагнер	JavaScript. Энциклопедия пользователя (+ CD-ROM)	464	84x108/16
6705	Вайк Аллен	JavaScript.Справочник	896	70x100/16
6886	Аллен Поль Р.	J2EE. Разработка бизнес приложений	736	70x100/16
8011	Седжвик Робер	Фундаментальные алгоритмы на JAVA. Анализ/Структуры данных/Сор	688	70x100/16
7988	Томсон Л.,Вел	Разработка Web-приложений на PHP и MySQL (+ CD-ROM)	672	70x100/16
6729	Гешвинде Эва	Разработка Web-приложений на PHP и PostgreSQL. Руководство разр	608	70x100/16
5963	Вайнман Л.,Ви	Динамический HTML. Руководство разработчика Web-сайтов (+ CD-	464	70x100/16
8520	Вайнман Линда	Креативный Web-дизайн на HTML 4	528	70x100/16
6495	Иеркс Анна-Ма	Dreamweaver 4. Искусство создания web-сайтов	688	70x100/16
5149	Бизли Девид М	Язык программирования Python. Справочник	336	70x100/16
6147	Лесса Андре	Python. Руководство разработчика.	688	70x100/16
5720	Вайк А.	PHP. Справочник	448	70x100/16
7461	Кишик А.Н.,Гал	Flash MX.Эффективный самоучитель.	416	70x100/16
7389	Китинг Д. И др.	Flash MX. Искусство создания web-сайтов 2-е изд.	848	70x100/16
6094	Кишик А.Н.	Flash 5.0. Анимация. Эффективный самоучитель. Быстро...просто...на	240	70x100/16
6729	Гешвинде	PostgreSQL. Руководство разработчика и админ.	608	70x100/16
6742	Уолтер С.	ASP.NET. Искусство создания web-сайтов	672	70x100/16
		Мультимедиа.		
314	Скибб Л., Хейф	Оптимизация мультимедиа ПК (+ CD-ROM)	352	60x84/16
		Экономика, бухгалтерский учет		
6962	Чистов Д.В. и д	Хозяйственные операции в компьютерной бухгалтерии 7.7 в Украине	528	60x84/16
6717	Бююль, Цефел	SPSS: искусство обработки информации 2-издание	608	70x100/16
6963	Нареш Малхот	Маркетинговые исследования и эффективный анализ статистических данных	768	70x100/16
7166	Соломон, Майк	Поведение потребителя. Искусство и наука побеждать на рынке	784	70x100/16
4849	Устинова Г.М.	Информационные системы менеджмента	368	60x90/8
		Словари		
5228	Митчелл Ш.	Толковый словарь компьютерных технологий	720	70x100/16

Научное издание

Эбен Майкл, Таймен Брайан

FreeBSD. Энциклопедия пользователя

Третье издание переработанное и дополненное

Заведующий редакцией *С.Н.Козлов*
Научный редактор: *Т.И.Клочкова*
Верстка *Г.А.Булавко*
Главный дизайнер *О.А.Шадрин*

ООО «ДиаСофтЮП», 196105, Санкт-Петербург, пр. Ю.Гагарина, д. 1, ком. 108.
Лицензия №000328 от 9 декабря 1999 г.

Сдано в набор 29.09.2003. Подписано в печать 20.10.2003. Формат 70×100/16.
Бумага типографская. Гарнитура Таймс. Печать офсетная. Печ.л. 48,00.
Тираж 2000 экз. Заказ № 390

Отпечатано с готовых диапозитивов
в ФГУП ордена Трудового Красного Знамени «Техническая книга»
Министерства Российской Федерации по делам печати,
телерадиовещания и средств массовых коммуникаций
198005, Санкт-Петербург, Измайловский пр., 29.