ДЕНИС КОЛИСНИЧЕНКО

Руководство по командам и shell-программированию в Linux

КОМАНДНАЯ СТРОКА

ОБОЛОЧКИ BASH, TCSH, ZSH

НАПИСАНИЕ СЦЕНАРИЕВ
НА ЯЗЫКАХ ОБОЛОЧЕК BASH И TCSH

ОСОБЕННОСТИ ФАЙЛОВОЙ СИСТЕМЫ LINUX

КОМАНДЫ ОБРАБОТКИ ТЕКСТА. ЯЗЫК GAWK

КОМАНДЫ СИСТЕМНОГО АДМИНИСТРАТОРА

КОМАНДЫ ДЛЯ РАБОТЫ В СЕТИ И ИНТЕРНЕТЕ

ЗАГРУЗЧИКИ GRUB И GRUB2.
СИСТЕМЫ ИНИЦИАЛИЗАЦИИ LINUX

УПРАВЛЕНИЕ ПАКЕТАМИ

КРАТКОЕ РУКОВОДСТВО ПО СОЗДАНИЮ
СОБСТВЕННОГО ДИСТРИБУТИВА

bhv

Денис Колисниченко

Руководство по командам и shell-программированию в Linux

Санкт-Петербург

«БХВ-Петербург»

2011

УДК 681.3.068
ББК 32.973.26-018.1
К60

Колисниченко Д. Н.

К60 Руководство по командам и shell-программированию в Linux. — СПб.: БХВ-Петербург, 2011. — 288 с.: ил. — (БЛЦ)

ISBN 978-5-9775-0619-9

Рассмотрены команды Linux, основы работы в командной строке, а также настройка системы с помощью программ, обладающих только текстовым интерфейсом. Работа с системой выполняется только в режиме консоли, что требует определенной квалификации пользователя. Подробно описаны наиболее полезные команды Linux, особенности файловой системы Linux, системы инициализации, загрузчики GRUB и GRUB2. С позиции пользователя оценены интерактивные возможности оболочки zsh. Даны практические примеры разработки сценариев на языках оболочек bash и tcsh. Рассмотрено управление пакетами для наиболее актуальных на данный момент дистрибутивов. Для энтузиастов Linux написана отдельная глава о разработке собственного дистрибутива Linux и создании загрузочного LiveCD.

Для системных администраторов, программистов и квалифицированных пользователей Linux

УДК 681.3.068
ББК 32.973.26-018.1

Группа подготовки издания:

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

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 26.08.10.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 23,22.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

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

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

ISBN 978-5-9775-0619-9

Оглавление

Введение ... **1**

ЧАСТЬ I. Командная строка ... **5**

Глава 1. Введение в командную строку ... **7**

1.1. Вход в систему ... 7
1.2. Команды *poweroff*, *halt*, *reboot*, *shutdown* ... 10
1.3. Как работать в консоли ... 10
1.4. Графические терминалы ... 11

Глава 2. Командные интерпретаторы ... **13**

2.1. Файл /etc/shells ... 13
2.2. Оболочка *sh* ... 14
2.3. Оболочка *csh* ... 14
2.4. Оболочка *ksh* ... 15
2.5. Оболочка *bash* ... 15
2.6. Оболочка *zsh* ... 16
2.7. Оболочка *tcsh* ... 17
2.8. Оболочка *ash* ... 17
2.9. Выбор оболочки ... 17

Глава 3. Базовые команды Linux ... **18**

3.1. О командах Linux ... 18
3.2. Команда *arch*: вывод архитектуры компьютера ... 18
3.3. Команда *banner*: текстовый баннер ... 19
3.4. Команда *chsh*: изменение командного интерпретатора ... 19
3.5. Команда *cksum*: вычисление контрольной суммы файла ... 19
3.6. Команда *clear*: очистка экрана ... 19
3.7. Команда *date*: вывод даты и времени ... 20
3.8. Команда *echo*: вывод сообщения ... 21
3.9. Команда *exit*: выход из системы ... 21
3.10. Команда *env*: установка переменных окружения ... 21
3.11. Команды *man* и *info*: вывод справки ... 22

3.12. Команда *printenv*: вывод значения переменной окружения 22
3.13. Команда *reset*: сброс терминала 22
3.14. Команда *sleep*: пора спать 22
3.15. Команда *startx* — запуск графического интерфейса X.Org 23
3.16. Команда *tee*: перенаправление ввода 23
3.17. Команда *true*: успешное завершение 23
3.18. Команда *yes*: возвращает *y* 23

Глава 4. Файловая система. Команды для работы с файловой системой 24

4.1. Файловые системы, поддерживаемые Linux 24
4.1.1. Выбор файловой системы 25
4.1.2. Linux и файловые системы Windows 26
4.1.3. Сменные носители 27
4.2. Особенности файловой системы Linux 27
4.2.1. Имена файлов в Linux 27
4.2.2. Файлы и устройства 27
4.2.3. Корневая файловая система и монтирование 28
4.2.4. Стандартные каталоги Linux 29
4.3. Команды для работы с файлами и каталогами 30
4.3.1. Работа с файлами 30
4.3.2. Работа с каталогами 33
4.4. Команда *ln*: создание ссылок 35
4.5. Команды *chown*, *chmod* и *chattr* 35
4.5.1. Команда *chmod*: права доступа к файлам и каталогам 35
4.5.2. Команда *chown*: смена владельца файла 37
4.5.3. Специальные права доступа (SUID и SGID) 38
4.5.4. Команда *chattr*: атрибуты файла, запрет изменения файла 38
4.6. Монтирование файловых систем 39
4.6.1. Команды *mount* и *umount* 39
4.6.2. Файлы устройств и монтирование 39
Жесткие диски 40
Приводы оптических дисков 41
Дискеты и USB-накопители 42
4.6.3. Опции монтирования файловых систем 42
4.6.4. Монтирование разделов при загрузке 43
4.6.5. Подробно о UUID и файле /etc/fstab 45
4.6.6. Монтирование Flash-дисков 48
4.7. Настройка журнала файловой системы ext3 49
4.8. Файловая система ext4 49
4.8.1. Сравнение ext3 и ext4 50
4.8.2. Совместимость с ext3 51
4.8.3. Переход на ext4 51
4.9. Особые команды 52
4.9.1. Команда *mkfs*: создание файловой системы 52

4.9.2. Команда *fsck*: проверка и восстановление файловой системы 52
4.9.3. Команда *chroot*: смена корневой файловой системы 53
4.9.4. Установка скорости CD/DVD 53
4.9.5. Монтирование каталога к каталогу 54
4.9.6. Команды поиска файлов 54

Глава 5. Процессы 56

5.1. Команды *kill*, *killall*, *xkill* и *ps* 56
5.2. Программа *top*: кто больше всех расходует процессорное время 58
5.3. Команды *nice* и *renice*: изменение приоритета процесса 60
5.4. Перенаправление ввода/вывода 60

Глава 6. Запись CD/DVD из консоли 62

6.1. Команда *dd*: создание образа диска 62
6.2. Команды *cdrecord* и *dvdrecord*: запись образа на болванку 63
6.3. Команды очистки перезаписываемых дисков 64
6.4. Команда *mkisofs*: создание ISO-образа 64
6.5. Преобразование образов дисков 64
6.6. Создание и монтирование файлов с файловой системой 65

Глава 7. Команды для работы с текстом 66

7.1. Команда *cmp*: сравнение двух файлов 66
7.2. Команда *column*: разбивка текста на столбцы 66
7.3. Команда *comm*: еще одна команда для сравнения файлов 67
7.4. Команда *diff*: сравнение файлов 67
7.5. Команда *diff3*: сравнение трех файлов 68
7.6. Команда *egrep*: расширенный текстовый фильтр 69
7.7. Команда *expand*: замена символов табуляции пробелами 70
7.8. Команда *fmt* 70
7.9. Команда *fold* 70
7.10. Команда *grep*: текстовый фильтр 71
7.11. Команды *more* и *less*: постраничный вывод 71
7.12. Команды *head* и *tail*: вывод начала и хвоста файла 71
7.13. Команда *look* 71
7.14. Команда *sort*: сортировка файлов 72
7.15. Команда *split*: разбиение файлов на несколько частей 72
7.16. Команда *unexpand*: замена пробелов на символы табуляции 73
7.17. Команды *vi*, *nano*, *ee*, *mcedit*, *pico*: текстовые редакторы 73
7.18. Команда *wc*: подсчет слов в файле 77

Глава 8. Команды для работы с сетью и Интернетом 78

8.1. Команда *ifconfig*: управление сетевыми интерфейсами 78
8.2. Команда *route* 79

8.3. Команда *pppoeconf*: настройка DSL-соединения ... 80
8.4. Команда *pppconfig*: настройка модемного (PPP) соединения ... 84
8.5. Команда *wvdial*: настройка PPP-соединения ... 84
8.6. Текстовые браузеры ... 86
8.7. Команда *ftp*: FTP-клиент ... 87
8.8. Команда *wget*: загрузка файлов ... 88
8.9. Команды для диагностики сети ... 89

Глава 9. Команды системного администратора ... 94

9.1. Программы разметки диска ... 94
9.1.1. Программа *fdisk* ... 94
9.1.2. Программа *parted* ... 97
9.2. Информация о системе и пользователях ... 101
9.2.1. Команда *uptime*: информация о работе системы ... 101
9.2.2. Команда *users*: информация о пользователях ... 101
9.2.3. Команды *w, who, ftpwho* и *whoami*: информация о пользователях ... 101
9.3. Планировщик *at* ... 102
9.3.1. Команда *at*: добавление задания ... 102
9.3.2. Команды *atq* и *atrm*: очередь заданий и удаление задания ... 102
9.4. Планировщик *crond* ... 103
9.5. Планировщик *anacron* ... 104
9.6. Команда *date:* вывод и установка даты и времени ... 105
9.7. Команды *free* и *df*: информация о системных ресурсах ... 105
9.8. Команда *md5sum*: вычисление контрольного кода MD5 ... 106
9.9. Команда *ssh*: удаленный вход в систему ... 106
9.10. Устройства и драйверы ... 108

ЧАСТЬ II. ОПЕРАЦИОННАЯ СИСТЕМА ... 111

Глава 10. Загрузчики Linux ... 113

10.1. Основные загрузчики ... 113
10.2. Конфигурационные файлы GRUB и GRUB2 ... 114
10.2.1. Конфигурационный файл GRUB ... 114
10.2.2. Конфигурационный файл GRUB2 ... 116
10.3. Команды установки загрузчиков ... 120
10.4. Установка тайм-аута выбора операционной системы. Редактирование параметров ядра Linux ... 120
10.5. Установка собственного фона загрузчика GRUB и GRUB2 ... 124
10.6. Постоянные имена и GRUB ... 124
10.7. Восстановление загрузчика GRUB/GRUB2 ... 125
10.8. Две и более ОС Linux на одном компьютере ... 126
10.9. Загрузка с ISO-образов ... 128
10.10. Установка пароля загрузчика GRUB2 ... 128

Глава 11. Системы инициализации Linux 130

11.1. Начальная загрузка Linux 130
11.2. Система инициализации init 131
11.2.1. Файл /etc/inittab 131
11.2.2. Команда *init* 132
11.2.3. Команда *service* 133
11.2.4. Редакторы уровней запуска 133
11.3. Система инициализации upstart 136
11.3.1. Как работает upstart 136
11.3.2. Конфигурационные файлы upstart 136
11.4. Система инициализации Slackware 137

Глава 12. Команды управления пользователями 140

12.1. Многопользовательская система 140
12.2. Пользователь root 141
12.2.1. Максимальные полномочия 141
12.2.2. Как работать без root 141
Команда *sudo* 142
Команда *su* 142
Проблемы с *sudo* в Ubuntu и Kubuntu 143
Ввод серии команд *sudo* 144
12.2.3. Переход к традиционной учетной записи root 144
Преимущества и недостатки *sudo* 144
Традиционная учетная запись root в Ubuntu 145
Традиционная учетная запись root в Mandriva 146
Вход в качестве root в Fedora 146
12.3. Создание, удаление и модификация пользователей стандартными средствами 147
12.3.1. Команды *adduser* и *passwd* 147
12.3.2. Команда *usermod* 148
12.3.3. Команда *userdel* 149
12.3.4. Подробно о создании пользователей 149
12.4. Группы пользователей 151
12.5. Команды квотирования 151

Глава 13. Ядро 154

13.1. Команда *dmesg*: вывод сообщений ядра 154
13.2. Параметры ядра 163
13.3. Компиляция ядра 165
13.3.1. Установка исходных кодов ядра 166
13.3.2. Команда *make menuconfig*: настройка ядра 167
13.3.3. Команды компиляции ядра 170

ЧАСТЬ III. ПРОГРАММИРОВАНИЕ В LINUX 175

Глава 14. Программирование на языке C. Утилиты для программиста 177

14.1. Команда *gcc*: компилятор 177
14.2. Команда *make*: сборка проекта 179
14.3. Команды из пакета binutils 180
14.4. Другие полезные команды 181
14.5. Команда *gdb*: отладка программ 181

Глава 15. Командный интерпретатор *bash* 184

15.1. Настройка *bash* 184
15.2. Автоматизация задач с помощью *bash* 186
15.3. Привет, мир! 187
15.4. Использование переменных в собственных сценариях 187
15.5. Передача параметров сценарию 188
15.6. Массивы и *bash* 189
15.7. Циклы 189
15.8. Условные операторы 190
15.9. Функции 192
15.10. Примеры сценариев 192
15.10.1. Сценарий мониторинга журнала 192
15.10.2. Переименование файлов 193
15.10.3. Преобразование систем счисления 194

Глава 16. Сценарии на *tcsh* 195

16.1. Использование *tcsh* 195
16.2. Конфигурационные файлы *tcsh* 196
16.3. Создание сценариев на *tcsh* 197
16.3.1. Переменные, массивы и выражения 197
16.3.2. Чтение ввода пользователя 200
16.3.3. Переменные оболочки *tcsh* 200
16.3.4. Управляющие структуры 203
Условный оператор *if* 203
Условный оператор *if..then..else* 204
Оператор *foreach* 205
Оператор *while* 206
Оператор *switch* 207
16.3.5. Встроенные команды *tcsh* 207

Глава 17. Язык gawk 210

17.1. Введение в gawk 210
17.2. Основы языка 210
17.2.1. Образцы и действия 210

17.2.2. Операторы ... 211
17.2.3. Переменные ... 212
17.2.4. Ассоциативные массивы ... 212
17.2.5. Функции ... 212
17.2.6. Вывод с помощью *printf* ... 213
17.2.7. Управляющие структуры ... 214
Условный оператор *if..else* ... 214
Цикл *while* ... 214
Цикл *for* ... 215
17.3. Примеры ... 215

Глава 18. Собственный сервер для PHP-программиста ... 218

18.1. Зачем нужен сервер PHP-программисту? ... 218
18.2. Web-сервер ... 218
18.2.1. Установка Apache и PHP ... 218
18.2.2. Тестирование настроек Web-сервера ... 219
18.2.3. Конфигурационные файлы сервера. Команды запуска и останова сервера ... 221
18.3. Сервер баз данных MySQL ... 221
18.3.1. Установка сервера ... 221
18.3.2. Команды управления пользователями MySQL-сервера ... 222
18.3.3. Команды запуска и останова сервера ... 223
18.3.4. Программа MySQL Administrator ... 223
18.4. Быстрая настройка FTP-сервера ... 225

ЧАСТЬ IV. УПРАВЛЕНИЕ ПАКЕТАМИ ... 229

Глава 19. Введение в пакеты. Программы *rpm* и *dpkg* ... 231

19.1. Что такое пакет ... 231
19.2. Репозитарии пакетов ... 233
19.3. Программы для управления пакетами ... 234
19.4. Программа *rpm* (все Red Hat-совместимые дистрибутивы) ... 235
19.5. Программа *rpmbuild*: простая сборка пакетов исходного кода ... 236
19.6. Программа *dpkg*: управление DEB-пакетами ... 236
19.7. Команда *alien*: установка RPM-пакетов ... 238

Глава 20. Управление пакетами в Debian/Ubuntu ... 239

20.1. Программы для управления пакетами ... 239
20.2. Программа *apt-get* ... 239
20.2.1. Установка пакетов. Источники пакетов ... 239
20.2.2. Основные команды программы *apt-get* ... 240
Обновление источников ... 241
Удаление и переустановка пакетов ... 241

Обновление пакета и системы ... 242
Очистка кэша пакетов ... 242
Опции программы *apt-get* ... 242
Покдлючение репозитария Medibuntu в Ubuntu ... 243
Корова в *apt-get* ... 244
20.3. Программа *aptitude* ... 244

Глава 21. Управление пакетами в Fedora ... 245

21.1. Использование программы *yum* ... 245
21.2. Управление источниками пакетов ... 247
21.3. Установка пакетов через прокси-сервер ... 249
21.4. Плагины для программы *yum* ... 249

Глава 22. Управление пакетами в openSUSE. Менеджер пакетов *zypper* ... 250

Глава 23. Управление пакетами в Slackware ... 254

23.1. Особенности Slackware ... 254
23.2. Управление пакетами ... 255
23.2.1. Команда *installpkg*: установка пакетов ... 256
23.2.2. Команда *removepkg*: удаление пакетов ... 257
23.2.3. Команда *upgradepkg*: обновление пакетов ... 258
23.3. Нет нужного пакета — вам поможет программа *rpm2tgz* ... 258
23.4. Программа *slackpkg*: установка пакетов из Интернета ... 258

Глава 24. Управление пакетами в Mandriva ... 260

24.1. Команда *urpmi*: установка пакетов ... 260
24.2. Команда *urpme*: удаление пакетов ... 265
24.3. Поиск пакета. Получение информации о пакете ... 265

Заключение ... 266

Приложение. Создание дистрибутива ... 267

П1.1. Зачем нужно создавать еще один дистрибутив ... 267
П1.2. Инструменты для создания дистрибутива ... 268
П1.3. Этапы создания дистрибутива ... 269
П1.4. Процесс создания дистрибутива ... 269
П1.5. Развитие дистрибутива ... 272
П1.6. Быстрое создание LiveUSB ... 273

Предметный указатель ... 274

Введение

Linux — особенная операционная система, и сейчас мы поговорим о ее особенностях. Начнем с залога популярности Linux — лицензии GPL, по которой и распространяется эта операционная система. Согласно GPL, Linux распространяется абсолютно свободно. Заметьте, я не сказал "бесплатно". Многие думают, что Linux — это бесплатная операционная система. Отчасти это так. Но главное то, что она свободная, т. е. всем желающим доступен исходный код ядра (как и любых других Linux-программ) и вы можете распространять без всяких ограничений как любой дистрибутив Linux, так и исходные коды программ. Вы можете установить Linux на любое количество компьютеров и даже создать свой дистрибутив Linux и распространять его под другим названием — главное, чтобы он распространялся под лицензией GPL.

Сравните это с лицензией на Windows, где вы имеете право установить приобретенный дистрибутив только на несколько компьютеров (количество указывается в лицензии) и где вы не можете распространять дистрибутив. А в мире Linux никаких ограничений нет. Скачали дистрибутив (необязательно даже покупать его в магазине), установили на любое количество компьютеров (пока "болванка" не испортится), затем передали диск друзьям (желательно, пока он не испортился) — пусть они его распространяют дальше. А фанаты Linux могут даже создавать собственные дистрибутивы — как с нуля, так и на базе одного из существующих дистрибутивов (что намного проще). Кстати, в *приложении* этой книги мы как раз и поговорим о создании собственного дистрибутива.

Теперь, думаю, вам стало ясно, почему Linux настолько популярна. Но свободное распространение — это далеко не все. Судите сами: если операционная система бесплатная, но не соответствует требованиям пользователей, то ее установят, попробуют поработать и на следующий день деинсталлируют.

С технической точки зрения можно выделить следующие особенности Linux.

- *Реальная многозадачность* — Linux, как и ее родственник UNIX, использует режим разделения времени центрального процессора (time-sharing system). Работает это так: планировщик выделяет каждому процессу фиксированный интервал для выполнения. По окончанию выделенного времени планировщик приостанавливает выполнение процесса и передает управление другому процессу.

А другие операционные системы используют режим *вытесняющей многозадачности*, когда процесс сам должен уступить место "под солнцем", т. е. сам приостановить свое выполнение и передать право на выполнение другому процессу. Все бы хорошо, но некоторые процессы могут "узурпировать" все процессорное время, и якобы многозадачная операционная система превращается в… однозадачную со всеми вытекающими последствиями.

- *Многопользовательский доступ* — Linux не только многозадачная, но и многопользовательская операционная система. Это означает, что в системе могут *одновременно* работать несколько пользователей. Как именно — это уже другой вопрос. Один пользователь может находиться непосредственно за компьютером, а остальные — подключены по сети. Но не забывайте, что Linux можно установить и на мейнфрейм — суперкомпьютер с несколькими терминалами. Правда, такие компьютеры постепенно отходят, а их место занимают так называемые тонкие клиенты, когда терминал подключен к компьютеру не физически, а по сети. Но в любом случае, несмотря на способ соединения, Linux является многопользовательской системой.
- *Страничная организация памяти* — память в Linux организована в виде страниц по 4 Кбайт каждая. Когда физическая оперативная память заканчивается, включается механизм подкачки и неиспользуемые данные сбрасываются в область подкачки на жесткий диск. Правда, такой организацией памяти и механизмом подкачки сейчас никого не удивишь — все современные операционные системы работают примерно так же.
- *Загрузка выполняемых модулей "по требованию"* — чтобы ядро системы поддерживало определенное устройство или функцию (например, протокол), нужно добавить программный код в состав ядра. Но ядро, поддерживающее все возможности и все устройства, будет просто огромным. В Linux эта проблема решается загрузкой модулей, которые добавляют поддержку определенных устройств. Например, вам нужна поддержка звуковой платы Creative — загрузите модуль, реализующий поддержку этой платы. При этом в память загружаются только те модули, которые необходимы для полноценной работы конкретной системы, что позволяет оптимизировать использование ресурсов компьютера.
- *Совместное использование исполняемых программ* — если несколько пользователей запустили одну и ту же программу, то в память загружается всего одна копия этой программы, а не несколько, что позволяет экономить оперативную память.
- *Общие библиотеки* — библиотеки содержат наборы процедур, которые используются различными приложениями. Вместо того чтобы скомпилировать в один исполнимый файл все процедуры, приложение использует библиотеку (которая также может использоваться другими приложениями), что позволяет существенно экономить место на диске.
- *Поддержка различных файловых систем* — Linux поддерживает много различных файловых систем, в том числе и файловые системы Windows. О поддержке файловых систем мы поговорим в *главе 4*.

- *Поддержка разных аппаратных платформ* — Linux может работать не только на платформах x86/x64. Поддерживаются аппаратные платформы ARM, DEC Alpha, SUN Sparc, M68000 (Atari и Amiga), MIPS, PowerPC и др.

Мы перечислили далеко не все технические особенности Linux. Полный список может занять еще несколько страниц, введение затянется и книга покажется вам скучной. А этого нельзя допустить, поскольку данная книга — не просто справочник по командам ОС, а нечто большее — учебник по командной строке.

Технические особенности — это прекрасно, но чем же Linux хорош для обычного пользователя? Начнем с безопасности. Linux не страшны обычные вирусы, которые постоянно поражают Windows-компьютеры и которых насчитывается несколько сотен тысяч. Для Linux создано не более 1000 вирусов, да и то встретить такой вирус — это что-то из области ненаучной фантастики. К тому же вирус может причинить ущерб системе, только если пользователь откровенно не соблюдает правила безопасности, например всегда работает под учетной записью root.

Если раньше у Linux были проблемы с русским языком и наличием пользовательских приложений, то сейчас все иначе. С русским, как и с другими языками, проблем нет, поскольку все современные дистрибутивы уже давно перешли на UTF-8. А пользовательских приложений (работающих в графическом режиме) тоже достаточно — медиапроигрыватели, офисные пакеты, браузеры, почтовые клиенты — в общем все, что нужно обычному пользователю. К тому же для Linux разработаны эмуляторы (`wine`, `cedega`), позволяющие запускать Windows-игры.

Со временем Linux становится все проще и проще в использовании. Помню, как устанавливал в 1999 году свой первый дистрибутив (Red Hat). Перечитал множество документации (документации по Red Hat не нашел, зато воспользовался руководством по установке Slackware, где была подробно описана программа `fdisk` — в Red Hat того времени при установке как раз и вызывался `fdisk`) и только потом приступил к установке. На все про все потратил весь рабочий день (это было первое знакомство с Linux), а настройку системы отложил на следующий день. Но настроить за один день не получилось — то нужно было настроить монитор, то русифицировать X Window, то устранить некоторые проблемы в GNOME, установить русские шрифты для принтера и т. д. Работы хватало. Чтобы получить полностью рабочую систему (учитывая, что Linux я видел в первый раз в жизни), мне понадобилось около недели.

Сейчас все иначе. На установку системы в зависимости от дистрибутива и производительности компьютера уходит 15–40 минут. После этого в большинстве случаев вы получаете полностью рабочую систему. Вам останется только настроить соединение с Интернетом и доустановить необходимые программы. Если вы знаете, что делаете, то все это займет еще час, пусть два. А если не знаете, максимум — 1 день. Установка программ тоже стала намного проще — вам больше не нужно вручную бороться с зависимостями, нужно только указать менеджеру пакетов, какой пакет вам нужно установить, — все дополнительные пакеты будут установлены автоматически.

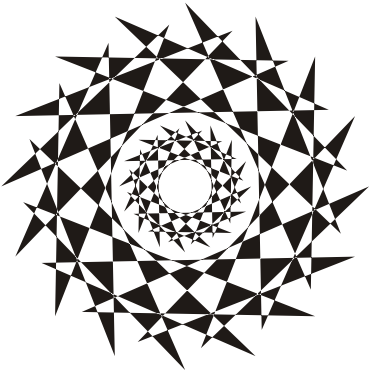
Конечно, есть частные случаи, когда не удается сразу заставить работать то или иное устройство, например Wi-Fi-адаптер. Но это частные случаи. Помню,

в Windows тоже пытался 4 часа настроить звуковую плату, причем драйвер у меня был...

В современных книгах по Linux все больше внимания уделяется графическим программам и графическому интерфейсу. Это правильно — нужно соответствовать времени, ведь Linux уже больше не воспринимается без графического интерфейса.

Но не нужно забывать, что изначально в Linux не было графического интерфейса. А была командная строка — вы вводили команду и получали результат выполнения (список файлов, список процессов, содержимое файла и т. д.). Даже когда появился графический интерфейс, многие пользователи предпочитали работать в командной строке. Сейчас все поменялось. Большинство пользователей работает исключительно в графическом интерфейсе, а некоторые даже не знают о существовании командной строки! Современные пользователи Linux уподобились Windows-пользователям, которые могут работать только с графическим интерфейсом и не знают ни одной команды операционной системы.

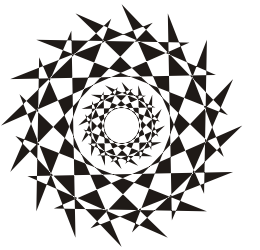
Эта книга посвящена командной строке Linux. В ней вообще не будут рассматриваться графические программы и графический интерфейс Linux. Только консоль, команды и файлы конфигурации — ничего больше. Зачем это нужно? А затем, что многие так называемые проблемы возникают от незнания и решаются вводом той или иной команды. Не нужно воспринимать данную книгу как сухой и скучный справочник по командам. Если вам нужен такой справочник, то вообще не нужно тратиться на покупку книги — он у вас всегда под рукой — это команда `man`. Данная книга — это учебник по использованию команд Linux. В ней вы найдете описание синтаксиса команд, описание параметров и, конечно же, практические примеры. Чувствую, введение затянулось, поэтому самое время приступить к чтению книги.



ЧАСТЬ I

Командная строка

Часть I полностью посвящена командной строке. Мы научимся правильно использовать командную строку, познакомимся с различными командными интерпретаторами, а также рассмотрим множество полезных команд Linux.



Глава 1

Введение в командную строку

1.1. Вход в систему

Настоящий линуксоид должен уметь работать в консоли. Ведь когда система Linux только появилась, существовала одна консоль, о графическом интерфейсе не было и речи. Знаете, почему UNIX и Linux отталкивали обычных пользователей? Потому что не было хорошего графического интерфейса. Раньше в Linux работали одни профессионалы. Сейчас все изменилось — в Linux очень удобный графический интерфейс, который с удовольствием используют и профессионалы (дождались наконец-то!), забывая о командной строке. Наш дистрибутив вообще ориентирован на работу в графическом режиме, а в официальных руководствах, которые можно найти в Интернете, о консоли вообще не упоминается. А ведь она есть! В этой главе мы поговорим о том, как правильно работать в консоли. Совсем необязательно работать полностью в текстовом режиме, вы можете использовать материал данной главы для эффективной работы с Терминалом — эмулятором консоли.

Обычные пользователи в консоль ни ногой — даже принципиально, мол, зачем возвращаться в DOS? Под "DOS" имелась в виду командная строка Linux. Да, ее вид не очень дружелюбный, но это только кажется. Стоит вам поработать в консоли, и вы поймете все ее преимущества. Начнем с того, что командная строка Linux намного удобнее командной строки DOS — об этом мы еще поговорим. В консоли можно выполнять те же операции, что и в графическом режиме, причем все намного быстрее. Хотите бороздить просторы Интернета? Пожалуйста, но без картинок. Не так красиво, но зато сэкономите трафик. А на обмен электронными сообщениями это никак не влияет. В консоли также можно работать и с документами, правда, тоже о графике можно забыть. Консоль позволяет эффективно использовать ресурсы старых компьютеров. Да, в графическом режиме на стареньком "Пентиуме" не поработаешь, зато в текстовом режиме его можно быстро превратить в очень полезный для всей сети компьютер — в шлюз, через который его более мощные собратья будут получать доступ к Интернету.

По умолчанию в современных дистрибутивах при входе в систему запускается графический менеджер регистрации (рис. 1.1). Однако из всех правил могут быть исключения. Пример тому дистрибутив Slackware — в нем сначала нужно выполнить вход в консоли (см. далее), а потом для запуска графического интерфейса ввести команду `startx`.

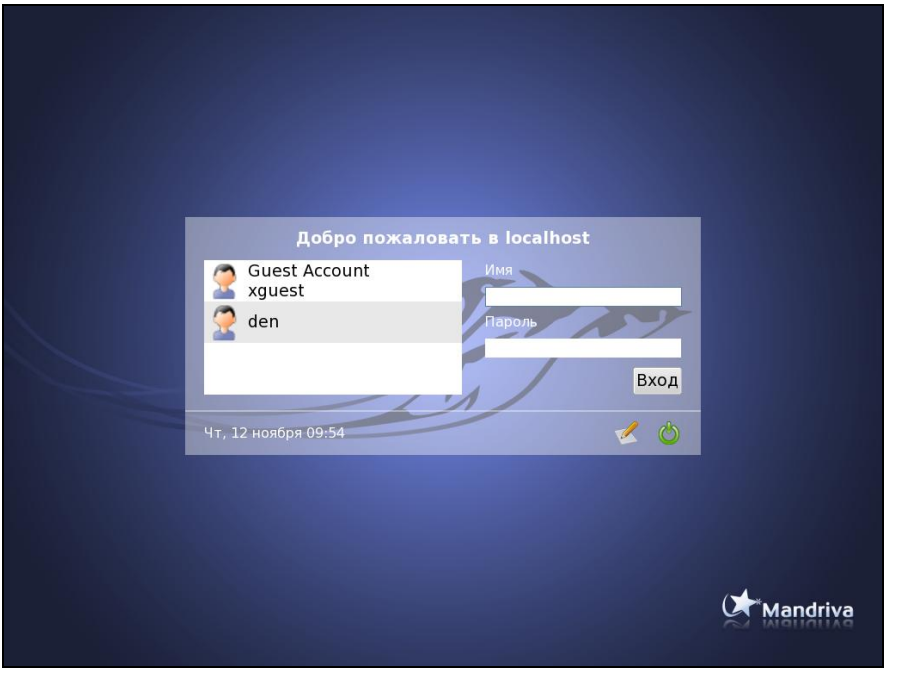


Рис. 1.1. Графический вход в систему (Mandriva 2010)

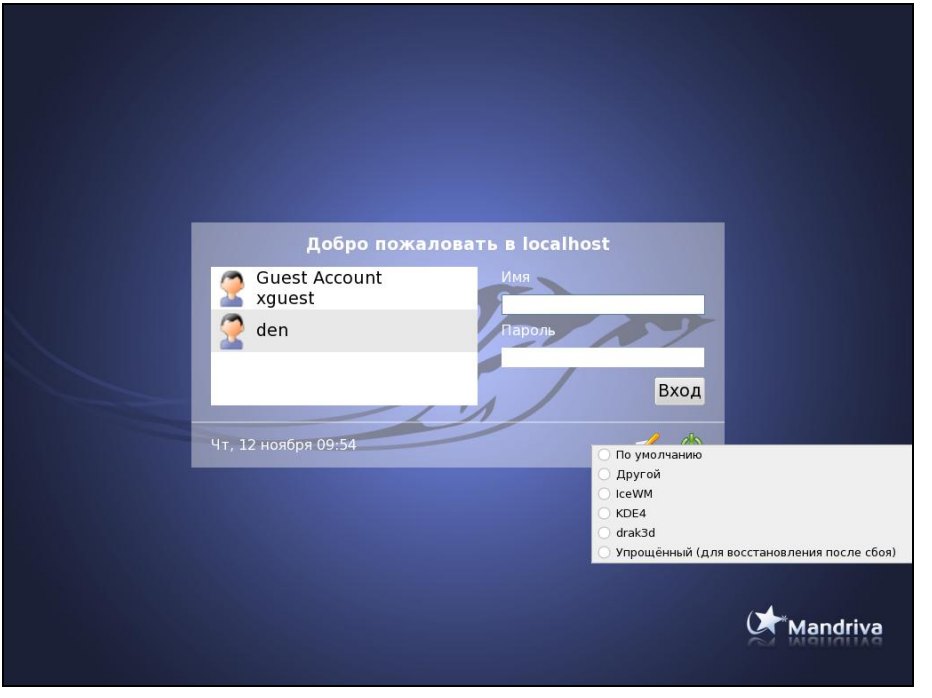


Рис. 1.2. Выбор типа сеанса (Mandriva 2010)

Для входа в систему вам нужно указать имя пользователя и пароль. После этого загрузится KDE или GNOME (в зависимости от того, какая графическая среда установлена в вашем дистрибутиве по умолчанию). Конечно, может быть загружена какая-то другая графическая среда, но обычно по умолчанию устанавливается KDE или GNOME. Для выбора графической среды нужно нажать кнопку **Тип сеанса** (или **Сеанс** — в Fedora и некоторых других дистрибутивах, а в некоторых дистрибутивах эта кнопка может быть представлена графической пиктограммой), как показано на рис. 1.2.

Сейчас вы находитесь в графическом режиме. Для того чтобы перейти из графического режима в консоль (рис. 1.3), нажмите клавиатурную комбинацию <Ctrl>+<Alt>+<F*n*>, где *n* — номер консоли (от 1 до 6). Чтобы перейти на первую консоль, нужно нажать комбинацию клавиш <Ctrl>+<Alt>+<F1>, на вторую — <Ctrl>+<Alt>+<F2> и т. д. Обратите внимание, что так можно перейти в консоль только из графического режима. Если вы уже находитесь в консоли, то для переключения между консолями служат комбинации клавиш <Alt>+<F1> ... <Alt>+<F6>, а также <Alt>+<F7> — для перехода в графический режим. Для лучшего запоминания эти комбинации клавиш приведены в табл. 1.1.

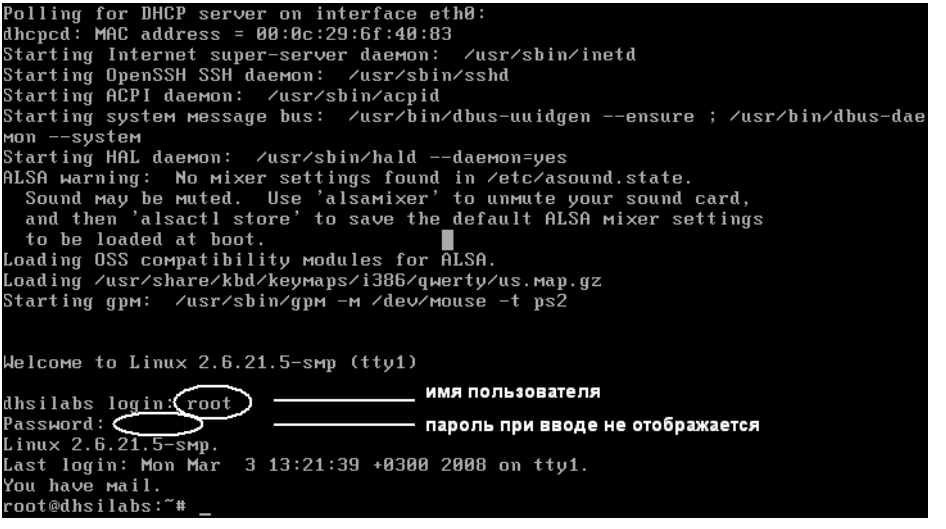


Рис. 1.3. Регистрация в консоли (Slackware)

Таблица 1.1. *Клавиши переключения между консолями и графическим режимом*

Комбинация клавиш	**Назначение**
<Ctrl>+<Alt>+<F*n*> (*n* от 1 до 6)	Переключение из графического режима в консоль с номером *n*
<Alt>+<F*n*> (*n* от 1 до 6)	Переключение между консолями
<Alt>+<F7>	Переключение из консоли в графический режим

1.2. Команды *poweroff*, *halt*, *reboot*, *shutdown*

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

Для перезагрузки компьютера существует команда `reboot`. Кроме нее вы можете использовать еще две команды — `halt` и `poweroff`:

- команда `halt` завершает работу системы, но не выключает питание. Вы увидите сообщение **`System is halted`**, свидетельствующее о возможности выключения питания. Эта команда предназначена для старых компьютеров, не поддерживающих расширенное управление питанием;
- команда `poweroff` завершает работу системы и выключает ее питание.

Самая "продвинутая" команда — `shutdown` — позволяет завершить работу и перезагрузить систему в назначенное время. Предположим, что вы хотите уйти пораньше, но компьютер нужно выключить ровно в 19:30 (вдруг некоторые пользователи задержались на работе, а вы выключите сервер, — некрасиво получится). Вот тут-то вам и поможет команда `shutdown`:

```
# shutdown -h 19:30 [сообщение]
```

> ***Примечание***
>
> Здесь и далее решетка (`#`) означает, что команда должна быть выполнена от имени пользователя root. Если перед командой ничего не указано или же указан символ доллара (`$`), команду можно выполнить от имени обычного пользователя.

Сообщение `[сообщение]` можно и не указывать, все равно Windows-пользователи его не увидят.

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

```
# shutdown -h now
```

Для перезагрузки системы есть опция `-r`:

```
# shutdown -r now
```

1.3. Как работать в консоли

Работа в консоли заключается во вводе нужной команды. Вы вводите команду (например, создания каталога, просмотра файла, вызова редактора и т. д.) и нажимаете клавишу <Enter>. Команда содержит как минимум имя запускаемой программы. Кроме имени программы команда может содержать параметры, которые будут переданы программе, а также символы перенаправления ввода/вывода (об этом чуть позже). Естественно, вам нужно знать имя программы, а также параметры, которые нужно ей передать. Если вы помните название программы, а назначение параметров забыли, вспомнить поможет команда `man`. Man (от англ. *manual*) — это справочная система Linux. В ней есть информация о каждой программе, которая установлена в вашей системе. Как система знает все обо всех программах? Все очень просто. Разработчики программ под Linux договорились, что вместе с программой будет поставляться специальный man-файл — файл справочной системы.

Понятно, если разработчик не добросовестный, он может и не создать файл справочной системы, но это происходит очень редко. Чтобы получить справку по какой-нибудь программе, нужно ввести команду:

```
man имя_программы
```

Вы никак не можете запомнить, как пишется та или иная команда? Если вы помните хотя бы, на какую букву она начинается, то воспользуйтесь функцией автодополнения командной строки: введите первые буквы команды и нажмите клавишу <Tab>. При первом нажатии система попытается дополнить команду, если это возможно. Иногда дополнить команду невозможно. Например, вы ввели букву `a` и нажали клавишу <Tab>. Ясное дело, в системе есть несколько команд, которые начинаются на букву "a". Тогда система не дополнит командную строку. Если вы хотите просмотреть все командаы на букву "a", тогда нажмите еще раз клавишу <Tab>.

> ***Примечание***
>
> Описанная здесь функция автодополнения работает в командной оболочке bash (которая используется по умолчанию). В следующей главе будут рассмотрены особенности и других оболочек.

Вам лень писать (даже с автодополнением) длинные команды? Тогда можно создать псевдонимы команд. Для этого в файл .bash_profile добавьте строки вида:

```
alias псевдоним='команда'
```

Например:

```
alias cfg-net='system-config-network'
```

Для того чтобы изменения вступили в силу, выйдите из консоли (команда `logout`) и заново зарегистрируйтесь.

Пожалуй, для полноценной работы с консолью вам нужно знать еще одну команду — `clear`. Данная команда очищает консоль (терминал). Очень полезная команда, особенно когда вы хотите все начать с "чистого листа".

1.4. Графические терминалы

Понимаю, что большинство дистрибутивов оснащены графическим интерфейсом, который к тому же запускается по умолчанию. Поэтому большинство пользователей не будут жертвовать удобным и привычным интерфейсом ради консоли.

Вместо того чтобы переключиться в консоль, можно использовать терминалы — эмуляторы консоли. Терминал — это графическая программа (рис. 1.4), в окне которой вы можете вводить команды и видеть результат их выполнения. Запустить терминал можно через меню GNOME/KDE (**Система | Стандартные | Терминал** или **Система | Системные | Терминал** — в зависимости от дистрибутива).

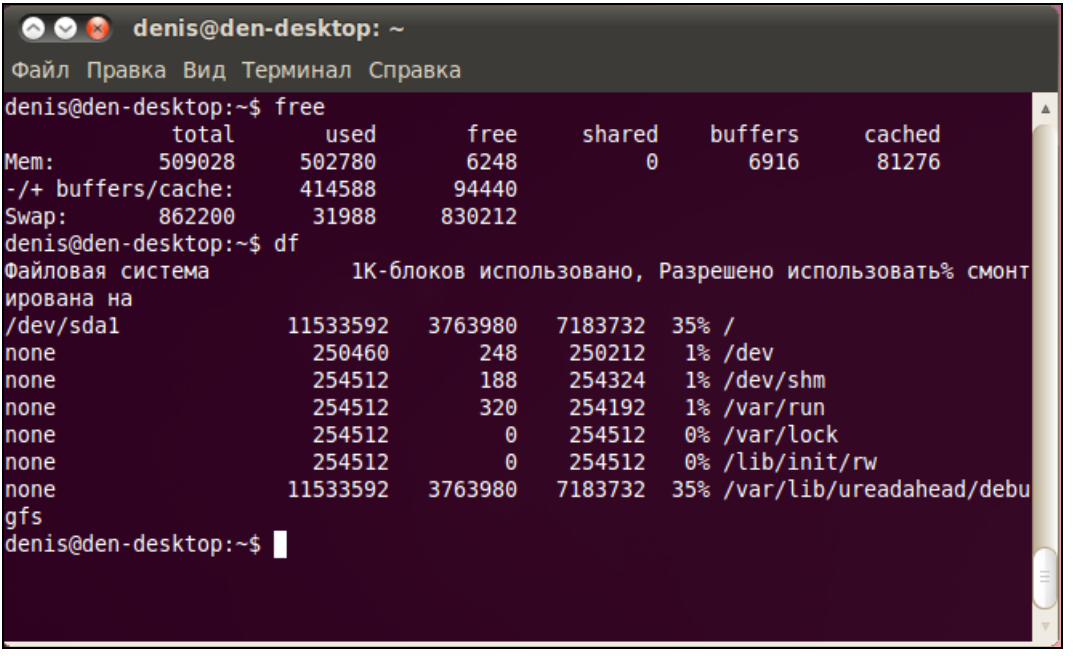
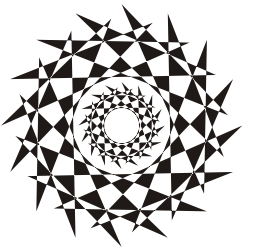


Рис. 1.4. Терминал



Глава 2

Командные интерпретаторы

2.1. Файл /etc/shells

По умолчанию во всех современных дистрибутивах используется командный интерпретатор `bash`. Основное предназначение `bash`, как и любой другой оболочки, — выполнение команд, введенных пользователем. Пользователь вводит команду, `bash` ищет программу, соответствующую команде, в каталогах, указанных в переменной окружения `PATH`. Если такая программа найдена, то `bash` запускает ее и передает введенные пользователем параметры. В противном случае выводится сообщение о невозможности выполнения команды.

Кроме `bash` существуют и другие оболочки — `sh`, `csh`, `ksh`, `zsh` и пр. Все командные оболочки, установленные в системе, прописаны в файле /etc/shells. Список оболочек может быть довольно длинным. В листинге 2.1 представлен файл /etc/shells дистрибутива Fedora (установка по умолчанию).

Листинг 2.1. Файл /etc/shells дистрибутива Fedora

```
/bin/ash
/bin/bash
/bin/csh
/bin/false
/bin/ksh
/bin/sh
/bin/tcsh
/bin/true
/bin/zsh
/usr/bin/csh
/usr/bin/ksh
/usr/bin/bash
/usr/bin/tcsh
/usr/bin/zsh
```

С точки зрения пользователя указанные оболочки мало чем отличаются. Все они позволяют выполнять введенные пользователем команды. Но оболочки используются

не только для выполнения команд, а еще и для автоматизации задач с помощью *сценариев*. Так вот, все эти оболочки отличаются синтаксисом языка описания сценариев.

Примечание

В листинге 2.1 программы /bin/false и /bin/true не являются оболочками. Это "заглушки", которые можно использовать, если вы хотите отключить ту или иную учетную запись пользователя. При входе пользователя в систему запускается установленная для него оболочка. Для каждого пользователя имеется возможность задать свою оболочку (изменить оболочку пользователь может самостоятельно командой `chsh`). Так вот, если для пользователя задать оболочку /bin/false (или /bin/true), он не сможет войти в систему. Точнее, он войдет в систему, но и сразу выйдет из нее, поскольку обе "заглушки" ничего не делают, а просто возвращают значение 0 (для false) или 1 (для true). Сессия же пользователя длится до завершения работы его оболочки.

2.2. Оболочка *sh*

Самым первым командным интерпретатором (оболочкой) в операционной системе UNIX (да, именно UNIX, поскольку корни Linux уходят в далекие 70-е годы) была `sh` (сокращение от *shell*). Данная оболочка до сих пор используется в современных версиях Linux (и FreeBSD).

Оболочка `sh` была разработана Стивеном Борном (Steve Bourne), поэтому ее второе название — Bourne Shell. Изначально `sh` была разработана для операционной системы AT&T (разработка Bell Labs). Чуть позже `sh` была усовершенствована и вошла в состав POSIX (Portable Operating System Interface for UNIX — Переносимый интерфейс операционных систем UNIX). Усовершенствованная версия `sh` до сих пор устанавливается (но не используется по умолчанию) в современных версиях FreeBSD.

С точки зрения пользователя оболочка `sh` не очень удобна, поэтому пользователи предпочитают другие оболочки, например `tcsh` или `bash`.

2.3. Оболочка *csh*

Оболочка `csh` (C Shell) по умолчанию используется в FreeBSD. Разработка `csh` началась еще в первых версиях BSD (Linux будет создан лет через 15). Тогда в институте Беркли начали создавать новую оболочку (`csh`), потому что не захотели мириться с ограничениями `sh`.

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

По сравнению с `sh`, у `csh` есть множество преимуществ: она умеет управлять заданиями, хранит историю ранее введенных команд, а также у `csh` есть сценарии, которые выполняются при входе пользователя (запуске оболочки) и при выходе

пользователя (когда пользователь вводит команду `exit`). В то время у `sh` не было таких сценариев, которые оказались очень удобными.

С точки зрения обычного использования оболочки (а не программирования) `csh` тоже была на высоте.

В последних версиях FreeBSD и Linux вместо `csh` используется ее усовершенствованная версия `tcsh`, а файл /bin/csh — это просто ссылка на /bin/tcsh.

2.4. Оболочка *ksh*

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

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

Несмотря на то что оболочка была разработана в 1986 году, она до сих пор используется в некоторых версиях UNIX по умолчанию, а также устанавливается по умолчанию во всех дистрибутивах Linux (но не используется по умолчанию). Правда, изначально `ksh` — это коммерческий продукт, поэтому в FreeBSD и Linux используется не `ksh`, а ее бесплатная версия — `pdksh`, но для краткости исполнимый файл называется `ksh`.

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

2.5. Оболочка *bash*

Командный интерпретатор `bash` (Bourne Again Shell) был разработан фондом свободного программного обеспечения (Free Software Foundation, FSF). За основу была взята оболочка `sh`. Оболочка стала очень популярной и сейчас используется по умолчанию во всех дистрибутивах Linux.

Оболочка `bash` может использоваться также и для запуска сценариев `sh`, поэтому `sh` во многих системах уже не устанавливается, а файл /bin/sh — это ссылка на /bin/bash.

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

Синтаксис `bash` довольно прост, поэтому бóльшая часть сценариев, разрабатываемых в Linux, пишется именно на `bash`.

2.6. Оболочка *zsh*

Оболочки `bash` и `tcsh` (современная версия `csh`) будут рассмотрены в *части III*, оболочка `ksh` используется редко. Поэтому сейчас мы поближе познакомимся с оболочкой `zsh`, которая становится все более популярной.

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

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

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

Можно использовать автодополнение `bash` — вводить начальные символы каталога и нажимать клавишу <Tab>. Это будет выглядеть примерно так:

```
cd /dir/sub [Tab]/subdi [Tab]
```

В `zsh` можно ввести:

```
/d/s/s
```

Затем нажать клавишу <Tab> — вы перейдете в нужный каталог. Например, для перехода в /etc/sysconfig/network, нужно ввести `/e/s/n` и нажать клавишу <Tab>. Кстати, команда `cd` уже не нужна.

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

/files/f1/sources/last

/files/f2/sources/last

Пусть мы находимся в каталоге /files/f1/sources/last, для перехода в каталог /files/f2/sources/last введите команду:

```
cd 1 2
```

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

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

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

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

Все возможности `zsh` в этой главе мы рассматривать не будем — их намного больше, чем вам кажется. Если вы заинтересовались, то прочитайте следующие страницы:

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

2.7. Оболочка *tcsh*

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

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

Оболочка `tcsh` очень удобна в использовании, но ее синтаксис сценариев сложнее, чем у `bash`. Однако в *части III* мы все же рассмотрим разработку сценариев на `tcsh`, чтобы вы смогли оценить сложность создания разработки сценариев на `bash` и на `tcsh`.

2.8. Оболочка *ash*

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

У `ash` всего 24 встроенных команд и 10 опций командной строки. Обычно `ash` используется при загрузке Linux в однопользовательском режиме (или в режиме восстановления).

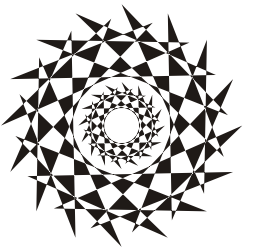
Оболочка `ash` совместима с `sh`, с ее помощью можно проверить сценарии на совместимость с традиционным синтаксисом `sh`. А в операционной системе NetBSD оболочка `ash` используется вместо /bin/sh.

2.9. Выбор оболочки

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

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

Довольно удобны в использовании оболочки `bash`, `tcsh` и `zsh`. Скорее всего, вы выберете одну из них. А вот для программирования вы будете использовать или `bash`, или `tcsh` (синтаксис `zsh` не очень понятен).



Глава 3

Базовые команды Linux

3.1. О командах Linux

Все команды Linux можно условно разделить на несколько групп:

- *команды общего назначения* — эти команды могут понадобиться в любой момент как при работе в консоли, так и при написании собственных сценариев;
- *команды для работы с файлами и каталогами* — эти команды будут рассмотрены в *главе 4* вместе с основами файловой системы Linux, без которых данные команды не будут понятны читателю;
- *команды обработки текста* — будут рассмотрены в *главе 7*;
- *команды для работы с сетью и Интернетом* — выйти в Интернет в Linux можно даже без запуска графического интерфейса, что и будет показано в *главе 8*;
- *команды системного администратора* — любой системный администратор просто обязан знать команды, представленные в *главе 9*.

Некоторые команды могут относиться к одной из групп, но в этой книге выделены в специальную главу, поскольку заслуживают отдельного разговора. Например, команды управления пользователями и группами выделены в *главу 12*. Можно было бы просто упомянуть команды `adduser` и `passwd` в *главе 9*, но в *главе 12* помимо рассмотрения формата важных конфигурационных файлов приводится описание множества дополнительных команд, так или иначе связанных с пользователями и группами пользователей.

В этой главе будут рассмотрены базовые команды Linux, т. е. команды общего назначения.

3.2. Команда *arch*: вывод архитектуры компьютера

Данная команда поможет узнать тип аппаратной платформы, например: i386, i586, i686 и др.

Пример использования:

```
$ arch
i686
```

3.3. Команда *banner*: текстовый баннер

Команда `banner` выводит строку (максимальная длина — 10 символов), рисуя буквы символом звездочки (`*`). Данную команду можно использовать в своих сценариях для вывода названия сценария. Пример использования:

```
$ banner Denix
```

3.4. Команда *chsh*: изменение командного интерпретатора

Команда `chsh` позволяет изменить командный интерпретатор, вывести список установленных интерпретаторов, а также установить командный интерпретатор по умолчанию. Синтаксис вызова программы:

```
$ chsh [параметры] интерпретатор
```

В качестве параметров вы можете передать:

- `-L` — выводит список установленных командных интерпретаторов (список хранится в файле /etc/shells);
- `-s` — устанавливает командный интерпретатор по умолчанию.

Примеры использования команды:

```
$ chsh zsh
$ chsh -s zsh
```

Первая команда изменяет текущий интерпретатор команд на оболочку `zsh`. Как только пользователь выйдет из системы и снова зайдет, будет запущен его интерпретатор по умолчанию (который установлен в файле /etc/passwd). Вторая команда устанавливает командный интерпретатор по умолчанию для текущего пользователя.

3.5. Команда *cksum*: вычисление контрольной суммы файла

Команда `cksum` вычисляет контрольную сумму (CRC) указанных файлов. Формат вызова:

```
$ cksum файлы
```

Пример:

```
$ cksum file1.txt file2.txt
```

3.6. Команда *clear*: очистка экрана

Команда `clear` очищает экран при работе в консоли (терминале).
Пример использования:

```
$ clear
```

3.7. Команда *date*: вывод даты и времени

Команда `date` относится как к командам общего назначения, так и к командам системного администратора. Обычные пользователи могут только просматривать дату и время в заданном формате, а пользователь с правами root может еще и устанавливать дату и время. Об установке даты и времени мы поговорим в *главе 9*, а сейчас разберемся с выводом даты и времени.

Команда `date` (без параметров) просто выводит текущую дату и время. Но вы можете уточнить формат вывода даты и времени так:

```
$ date +формат
```

Строка `формат` может состоять из модификаторов, указанных в табл. 3.1.

Таблица 3.1. *Модификаторы даты и времени*

Модификатор	Описание
`%%`	Знак %
`%a`	Сокращенное название дня недели (например, Вск)
`%A`	Полное название дня недели (например, Вторник)
`%b`	Сокращенное название месяца (например, Мар)
`%B`	Местное полное название месяца (напр., Март)
`%c`	Дата и время (в формате, заданном в настройках системы)
`%C`	Век
`%d`	День месяца (с предшествующим нулем, например, 07)
`%D`	Дата (в формате %m/%d/%y)
`%F`	Полная дата (в формате %Y-%m-%d)
`%g`	Последние две цифры года, соответствующего номеру недели в году согласно ISO 8601
`%G`	Год, соответствующий номеру недели в году согласно ISO 8601
`%H`	Час (00..23)
`%I`	Час (01..12)
`%j`	Номер дня в году
`%k`	Час (0..23)
`%l`	Час (1..12)
`%m`	Месяц (01..12)
`%M`	Минута (00..59)
`%n`	Вставляет разрыв строки
`%r`	12-часовое время (например, 12:12:07 PM)
`%R`	24-часовой формат часов и минут

Таблица 3.1 *(окончание)*

Модификатор	Описание
`%s`	Число секунд, прошедших с 1970-01-01 00:00:00 UTC (это так называемый timestamp)
`%S`	Секунда
`%t`	Вставляет символ табуляции
`%T`	Время (в формате %H:%M:%S)
`%U`	Номер недели в году (неделя начинается с воскресенья)
`%V`	Номер недели в году (неделя начинается с понедельника)
`%w`	День недели (0..6, где 0 — воскресенье)
`%y`	Последние две цифры года (00..99)
`%Y`	Год (все четыре цифры, например 2010)
`%z`	Часовой пояс в формате '+ччмм' (например, +0200)
`%:z`	Часовой пояс в формате '+чч:мм' (например, +02:00)
`%Z`	Алфавитное сокращение часового пояса (например, EST)

3.8. Команда *echo*: вывод сообщения

Команда `echo` выводит текстовую строку, указанную в качестве аргумента, например:

```
$ echo "Hello world!"
Hello world!
```

Обычно данная команда используется в сценариях командного интерпретатора для вывода сообщений на экран.

3.9. Команда *exit*: выход из системы

Для завершения сеанса работы в системе (при условии, что вы работаете в консоли) нужно использовать команду `exit`. Если не завершить сеанс работы, кто угодно сможет работать в системе под вашим именем (понятно, что во время вашего отсутствия за компьютером).

3.10. Команда *env*: установка переменных окружения

Команда `env` используется для установки переменных окружения во время выполнения команды, например ее можно использовать для установки переменной

EDITOR, которая содержит команду текстового редактора по умолчанию. Формат вызова команды:

```
$ env переменная=значение команда
```

Пример:

```
# env EDITOR=nano edquota -u user
```

3.11. Команды *man* и *info*: вывод справки

Команда man используется для получения справки о любой команде системы. Например, команда man ls выведет справку об использовании команды ls, которая выводит содержимое каталога. О том, как правильно использовать саму справочную систему, вам расскажет команда man man.

Команда info — это альтернативная справочная система. Если вы не нашли нужной информации с помощью команды man, попробуйте использовать команду info.

3.12. Команда *printenv*: вывод значения переменной окружения

Команда env используется для установки значений переменной окружения, а команда printenv позволяет просмотреть текущее значение переменной окружения, например:

```
$ printenv EDITOR
```

3.13. Команда *reset*: сброс терминала

Полезна при некоторых проблемах с терминалом (консолью). Команда reset производит сброс терминала и его начальную инициализацию, что позволяет восстановить параметры по умолчанию без выхода и входа в систему. Нужно отметить, что данную команду вы будете выполнять очень редко.

3.14. Команда *sleep*: пора спать

Команда sleep приостанавливает дальнейшее выполнение команд на указанное время. Данную команду полезно использовать в собственных сценариях, когда нужно приостановить выполнение сценария, чтобы пользователь успел прочитать сообщение. Когда мы будем разрабатывать собственные сценарии, мы будем часто пользоваться этой командой.

Формат вызова:

```
$ sleep число
```

По умолчанию число — это количество секунд, если после числа указан суффикс m, h, d, то считается, что число — это количество минут, часов и дней соответственно.

3.15. Команда *startx* — запуск графического интерфейса X.Org

Linux может запускаться на разных уровнях запуска. На пятом уровне запуска графический интерфейс X.Org (бывшее название — X Window) запускается автоматически (если он вообще был установлен). На третьем же уровне запуск графического интерфейса не производится. Если он вам тем не менее нужен, то его можно запустить с помощью команды `startx`. Никаких параметров не требуется.

3.16. Команда *tee*: перенаправление ввода

Перенаправляет стандартный ввод в два файла. Если задан только один файл, то в этот файл и на стандартный вывод. По умолчанию файлы, если существуют, перезаписываются. Чтобы файлы не были перезаписаны, указывается параметр `-a`.

Пример:

```
$ tee output.1 output.2
```

3.17. Команда *true*: успешное завершение

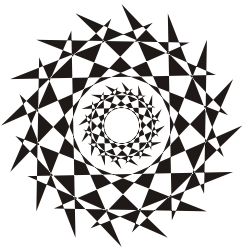
Данная команда ничего не делает, но всегда возвращает код возврата 0, что соответствует успешному завершению. Часто используется при написании сценариев.

3.18. Команда *yes*: возвращает *y*

Команда пригодится при написании сценариев. Она возвращает символ y (или строку, переданную ей в качестве параметра) и символ новой строки — как будто пользователь ввел `y` и нажал клавишу <Enter>.

Пример использования этой команды:

```
# yes | apt-get install mc
```



Глава 4

Файловая система. Команды для работы с файловой системой

4.1. Файловые системы, поддерживаемые Linux

Linux поддерживает много различных файловых систем. Начинающий пользователь просто теряется, когда видит такое многообразие выбора, — ведь в качестве корневой файловой системы доступны: ext2, ext3, ext4, XFS, ReiserFS, JFS.

Ранее "родной" файловой системой Linux была ext2 (файловая система ext использовалась разве что в самых первых версиях Linux), затем ей на смену пришла *журналируемая* версия файловой системы — ext3. Сегодня все современные дистрибутивы по умолчанию используют следующее поколение файловой системы — ext4. В этой главе помимо всего прочего вы найдете сравнение последних версий файловых систем Linux — ext3 и ext4.

> ***Примечание***
>
> Linux до сих пор поддерживает файловую систему ext, но она считается устаревшей, и рекомендуется воздержаться от ее использования.

Таким образом, в качестве корневой файловой системы и файловой системы других Linux-разделов используются файловые системы ext3, ext4, XFS, ReiserFS, JFS. Все перечисленные файловые системы (кроме ext2) ведут журналы своей работы, что позволяет восстановить данные в случае сбоя. Осуществляется это следующим образом — перед тем как выполнить операцию, *журналируемая* файловая система записывает эту операцию в журнал, а после выполнения операции удаляет запись из журнала. Представим, что после занесения операции в журнал произошел сбой (например, выключили свет). Позже, когда сбой будет устранен, файловая система по журналу выполнит все действия, которые в него занесены. Конечно, и это не всегда позволяет уберечься от последствий сбоя — стопроцентной гарантии никто не дает, но все же такая схема работы лучше, чем вообще ничего.

Файловые системы ext2 и ext3 совместимы. По сути, ext3 — та же ext2, только с журналом. Раздел ext3 могут читать программы (например, Total Commander и Ext2Fsd в Windows), рассчитанные на ext2. В свою очередь, ext4 — это усовершенствованная версия ext3.

В современных дистрибутивах по умолчанию задана файловая система ext4 (хотя в некоторых дистрибутивах все еще может использоваться ext3). При необходимости можно выбрать другие файловые системы. Далее мы рассмотрим их особенности,

чтобы понять, нужно ли их использовать или же остановить свой выбор на стандартной ext3/ext4.

- *Файловая система XFS* была разработана компанией Silicon Graphics в 2001 году. Основная особенность данной системы — высокая производительность (до 7 Гбайт/с). XFS может работать с блоками размером от 512 байтов до 64 Кбайт. Ясно, что если у вас много маленьких файлов, то в целях экономии места можно установить самый маленький размер блока. А если вы работаете с файлами большого размера (например, мультимедиа), то нужно выбрать самый большой размер блока — так файловая система обеспечит максимальную производительность (конечно, если "железо" позволяет). Учитывая высокую производительность этой файловой системы, ее нет смысла устанавливать на домашнем компьютере, поскольку все ее преимущества будут сведены на нет. А вот если вы будете работать с файлами очень большого размера, XFS проявит себя с лучшей стороны.
- *Файловая система ReiserFS* считается самой экономной, поскольку позволяет хранить несколько файлов в одном блоке (другие файловые системы могут хранить в одном блоке только один файл или одну его часть). Например, если размер блока равен 4 Кбайт, а файл занимает всего 512 байт (а таких файлов очень много в разных каталогах), то 3,5 Кбайт просто не будут использоваться. А вот ReiserFS позволяет задействовать буквально каждый байт вашего жесткого диска!

  Но у этой файловой системы есть два больших недостатка: она неустойчива к сбоям, и ее производительность сильно снижается при фрагментации. Поэтому, если вы выбираете данную файловую систему, покупайте UPS (источник бесперебойного питания) и почаще дефрагментируйте жесткий диск.
- *Файловая система JFS* (разработка IBM) сначала появилась в операционной системе AIX, а потом была модифицирована под Linux. Основные достоинства этой файловой системы — надежность и высокая производительность (выше, чем у XFS). Но у нее маленький размер блока (от 512 байтов до 4 Кбайт). Следовательно, она хороша на сервере баз данных, но не при работе с данными мультимедиа, поскольку блок в 4 Кбайт для работы, например, с видео в реальном времени будет маловат.

> ***Примечание***
>
> Особенности новой файловой системы ext4 будут рассмотрены в *разд. 4.8*.

4.1.1. Выбор файловой системы

Учитывая производительность рассматриваемых файловых систем, можно дать следующие рекомендации:

- для рабочей станции и сервера общего назначения оптимальной файловой системой являются ext3/ext4 или ReiserFS (в крайнем случае);
- на сервере баз данных можно использовать JFS — в этом случае (особенно, если база данных огромная) будет наблюдаться определенный прирост производительности;

- файловая система XFS — это удел станции мультимедиа, на обычной рабочей станции или обычном сервере ее использовать не следует.

Но производительность — это не единственный критерий выбора файловой системы, особенно для сервера. Да, производительность учитывать нужно, но кроме того нельзя пренебрегать и следующими факторами:

- *надежностью* — все-таки мы выбираем файловую систему для сервера, а не для домашнего компьютера;
- *наличием программ для восстановления файловой системы в случае сбоя* — сбой может произойти даже в случае использования самой надежной файловой системы, поэтому наличие программного комплекса для восстановления файловой системы не будет лишним;
- *максимальным размером файла* — сервер обрабатывает огромные объемы информации, поэтому данный критерий для нас также важен.

Файловые системы ext3/ext4, ReiserFS и XFS одинаково надежны, а вот надежность JFS иногда оставляет желать лучшего. Учитывая это, а также то, что программы для восстановления файловой системы имеются только для ext*, на сервере лучше использовать все-таки ext3/ext4.

Если вы уже интересовались характеристиками файловых систем, то могли в некоторых источниках встретить неправильную информацию о максимальном размере файла для файловой системы ext3. Так, иногда сообщается, что максимальный размер файла для ext3 равен 2 Гбайт, что делает ее непригодной для использования на сервере. Это не так. Раньше, во времена ext2 и ядер 2.2 и 2.4, действительно, существовало такое ограничение, но для ext2. Файловая система ext3 поддерживает файлы размером до 1 Тбайт, а максимальный размер тома (раздела) равен 4 Тбайт, что вполне достаточно даже для сервера. Если же вам нужна поддержка больших объемов данных, то рекомендую обратить внимание на другие файловые системы, например на ReiserFS (максимальный размер файла — 16 Тбайт) или на XFS/JFS (размер файла вообще исчисляется в *петабайтах*).

4.1.2. Linux и файловые системы Windows

Linux почти безо всяких ограничений поддерживает файловые системы FAT12 (DOS), FAT16 (или просто FAT, как в Windows 95) и FAT32 (Windows 98 и все последующие версии). Вы можете из Linux читать в файловых системах Windows файлы и каталоги, изменять, создавать новые файлы и каталоги, удалять их — в общем все, что можно делать в файловой системе непосредственно в Windows.

Однако файловые системы Windows не поддерживают установку прав доступа, поэтому можно даже не пытаться установить в Linux права доступа к файлу, который находится на Windows-разделе — у вас ничего не получится.

О файловой системе NTFS — отдельный разговор. По умолчанию (без перекомпиляции ядра) Linux умеет только читать данные, расположенные в NTFS-разделе. Однако даже после перекомпиляции ядра останется ряд ограничений на запись в NTFS-раздел: например, вы не можете создавать новые файлы, а можете только

редактировать уже имеющиеся. Кстати, поддержка NTFS современным ядром до сих пор экспериментальна, т. е. в один не совсем прекрасный момент при попытке записи вы можете потерять данные в вашем NTFS-разделе.

Я вас напугал? Существуют решения (мы их рассмотрим в этой книге), позволяющие снять бóльшую часть ограничений на запись в NTFS-разделы. Конечно, все эти решения не идеальные: что-то работает, но ужасно медленно, что-то снимает далеко не все ограничения на запись, но тем не менее все же есть возможность записывать данные в NTFS-раздел без потери данных.

4.1.3. Сменные носители

Linux превосходно работает со сменными CD/DVD- и USB-дисками, в большинстве случаев даже выполняется автоматическое монтирование и размонтирование сменных носителей (хотя эта функция доступна не во всех дистрибутивах). С другой стороны, автоматическое монтирование сменных носителей на сервере — это от лукавого, на домашнем компьютере — да, но не на сервере. О монтировании, в том числе автоматическом, мы поговорим чуть позже в этой главе.

4.2. Особенности файловой системы Linux

4.2.1. Имена файлов в Linux

По сравнению с Windows в Linux несколько другие правила построения имен файлов, вам придется с этим смириться. Начнем с того, что в Linux нет такого понятия, как расширение имени файла. В Windows, например, для файла Document1.doc именем файла является фрагмент Document1, а doc — это расширение. В Linux Document1.doc — это имя файла, никакого расширения нет.

Максимальная длина имени файла — 254 символа. Имя может содержать любые символы (в том числе и кириллицу), кроме / \ ? < > * " |. Но кириллицу в именах файлов я бы не рекомендовал вообще. Впрочем, если вы уверены, что не будете эти файлы передавать Windows-пользователям (на дискете, по электронной почте), — используйте на здоровье. А при обмене файлами по электронной почте (кодировка-то у всех разная, поэтому вместо русскоязычного имени пользователь увидит абракадабру) имя файла лучше писать латиницей.

Также вам придется привыкнуть к тому, что система Linux чувствительна к регистру в имени файла: FILE.txt и FiLe.Txt — это два разных файла.

Разделение элементов пути осуществляется символом / (прямой слэш), а не \ (обратный слэш), как в Windows.

4.2.2. Файлы и устройства

Пользователи Windows привыкли к тому, что файл — это именованная область данных на диске. Отчасти так оно и есть. Отчасти — потому, что приведенное определение файла было верно для DOS (Disk Operating System) и Windows.

В Linux же понятие файла значительно шире. Сейчас Windows-пользователи будут очень удивлены: в Linux есть файлы устройств, позволяющие обращаться с устройством, как с обычным файлом. Файлы устройств находятся в каталоге /dev (от *devices*). Да, через файл устройства мы можем обратиться к устройству! Если вы работали в DOS, то, наверное, помните, что что-то подобное было и там — существовали зарезервированные имена файлов: PRN (принтер), CON (клавиатура при вводе, дисплей при выводе), LPT*n* (параллельный порт, *n* — номер порта), COM*n* (последовательный порт).

Примечание

Кому-то может показаться, что разработчики Linux "украли" идею специальных файлов у Microsoft — ведь Linux появилась в начале 90-х годов, а DOS — в начале 80-х годов прошлого века. На самом деле это не так. Наоборот, Microsoft позаимствовала идею файлов устройств из операционной системы UNIX, которая была создана еще до создания DOS. Однако сейчас не время говорить об истории развития операционных систем, поэтому лучше вернемся к файлам устройств.

Вот самые распространенные примеры файлов устройств:

- /dev/sd*x* — файл жесткого диска или USB-накопителя;
- /dev/sd*xN* — файл устройства раздела на жестком диске, *N* — это номер раздела;
- /dev/mouse — файл устройства мыши;
- /dev/modem — файл устройства модема (на самом деле является ссылкой на файл устройства ttyS*n*);
- /dev/ttyS*n* — файл последовательного порта, *n* — номер порта (ttyS0 соответствует COM1, ttyS1 — COM2 и т. д.).

Примечание

В современных дистрибутивах имена вида /dev/hd*x* уже не используются (см. далее).

В свою очередь, файлы устройств бывают двух типов: блочные и символьные. Обмен информации с блочными устройствами, например с жестким диском, осуществляется блоками информации, а с символьными — отдельными символами. Пример символьного устройства — последовательный порт.

4.2.3. Корневая файловая система и монтирование

Наверняка на вашем компьютере установлена система Windows. Выполните команду **Пуск | Компьютер** (рис. 4.1).

Скорее всего, вы увидите пиктограмму гибкого диска (имя устройства — A:), пиктограммы разделов жесткого диска (пусть будет три раздела — C:, D: и E:), пиктограмму привода CD/DVD (F:). На рис. 4.1 нет диска A:, т. к. мой ноутбук не оснащен дисководом для гибких дисков.

Таким способом, с помощью буквенных обозначений A:, C:, D: и т. д. в Windows обозначаются корневые каталоги разделов жесткого диска и сменных носителей.

В Linux существует понятие *корневой файловой системы*. Допустим, вы установили Linux в раздел с именем /dev/sda3. В этом разделе и будет развернута корне-

вая файловая система вашей Linux-системы. Корневой каталог обозначается прямым слэшем (/), т. е. для перехода в корневой каталог в терминале (или консоли) нужно ввести команду `cd /`.

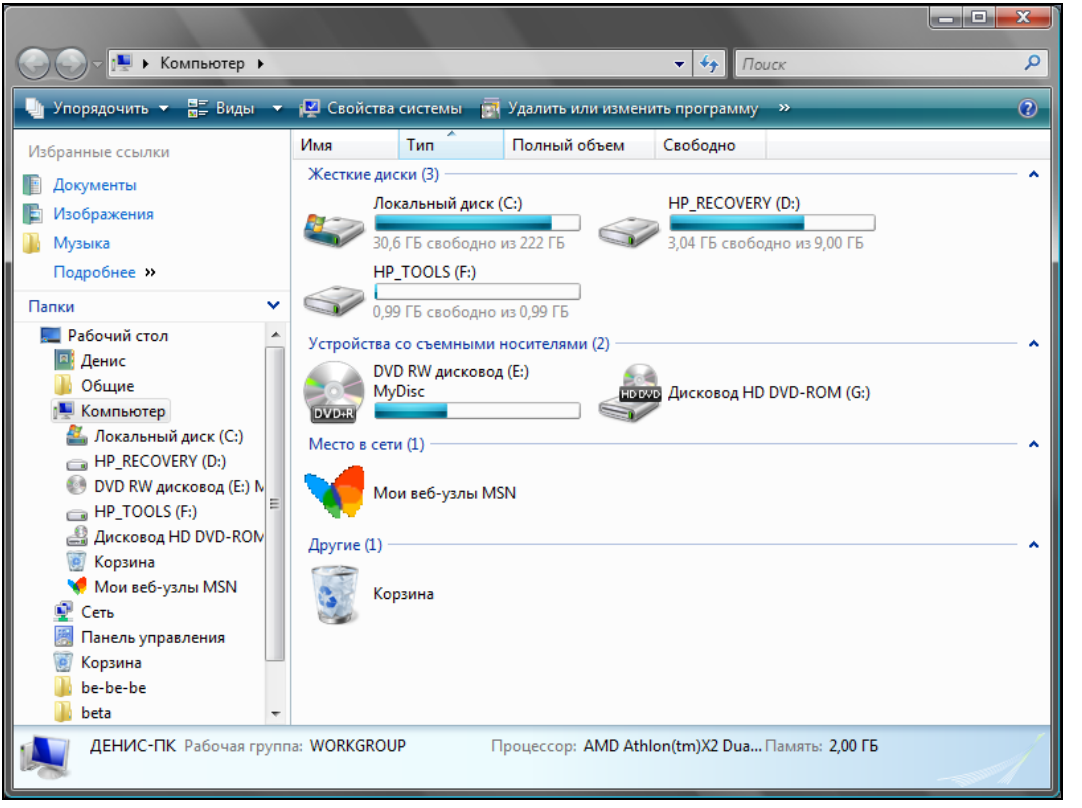


Рис. 4.1. Окно **Компьютер**

Понятно, что на вашем жестком диске есть еще разделы. Чтобы получить доступ к этим разделам, вам нужно *подмонтировать* их к корневой файловой системе. После монтирования вы можете обратиться к содержимому разделов через точку монтирования — назначенный вами при монтировании специальный каталог, например /mnt/cdrom. Монтированию файловых систем посвящен *разд. 4.6*, поэтому сейчас не будем говорить об этом процессе подробно.

4.2.4. Стандартные каталоги Linux

Файловая система любого дистрибутива Linux содержит следующие каталоги:

- / — корневой каталог;
- /bin — содержит стандартные программы Linux (`cat`, `cp`, `ls`, `login` и т. д.);
- /boot — каталог загрузчика, содержит образы ядра и Initrd (RAM-диска инициализации), может содержать конфигурационные и вспомогательные файлы загрузчика;

- /dev — содержит файлы устройств;
- /etc — содержит конфигурационные файлы системы;
- /home — содержит домашние каталоги пользователей;
- /lib — библиотеки и модули;
- /lost+found — восстановленные после некорректного размонтирования файловой системы файлы и каталоги;
- /media — в современных дистрибутивах содержит точки монтирования сменных носителей (CD-, DVD-, USB-накопителей);
- /misc — может содержать все, что угодно, равно как и каталог /opt;
- /mnt — обычно содержит точки монтирования;
- /proc — каталог псевдофайловой системы procfs, предоставляющей информацию о процессах;
- /root — каталог суперпользователя root;
- /sbin — каталог системных утилит, выполнять которые имеет право пользователь root;
- /tmp — каталог для временных файлов;
- /usr — содержит пользовательские программы, документацию, исходные коды программ и ядра;
- /var — постоянно изменяющиеся данные системы, например очереди системы печати, почтовые ящики, протоколы, замки и т. д.

В зависимости от дистрибутива, в корневом каталоге могут находиться дополнительные каталоги (либо, наоборот, отсутствовать некоторые каталоги, приведенные в списке). Например, в Debian и Ubuntu обязательно есть каталог /opt (ранее он использовался для установки альтернативного программного обеспечения, сейчас его можно использовать для чего угодно), а в Ubuntu есть каталог /cdrom (который по непонятным мне причинам не используется).

4.3. Команды для работы с файлами и каталогами

4.3.1. Работа с файлами

Здесь мы рассмотрим основные команды для работы с файлами в Linux (табл. 4.1), а в последующих разделах этой главы — команды для работы с каталогами, ссылками и поговорим о правах доступа к файлам и каталогам.

Таблица 4.1. *Основные команды Linux, предназначенные для работы с файлами*

Команда	Назначение
`touch <файл>`	Создает пустой файл
`cat <файл>`	Просмотр текстового файла

Таблица 4.1 *(окончание)*

Команда	Назначение
`tac <файл>`	Вывод содержимого текстового файла в обратном порядке, т. е. сначала выводится последняя строка, потом предпоследняя и т. д.
`cp <файл1> <файл2>`	Копирует файл `<файл1>` в файл `<файл2>`. Если `<файл2>` существует, программа попросит разрешение на его перезапись
`mv <файл1> <файл2>`	Перемещает файл `<файл1>` в файл `<файл2>`. Эту же команду можно использовать и для переименования файла
`rm <файл>`	Удаляет файл
`locate <файл>`	Производит быстрый поиск файла
`which <программа>`	Выводит каталог, в котором находится программа, если она вообще установлена. Поиск производится в каталогах, указанных в переменной окружения `PATH` (это путь поиска программ)
`less <файл>`	Используется для удобного просмотра файла с возможностью скроллинга (постраничной прокрутки)

Примечание

Все представленные команды предназначены для работы в консоли, т. е. в текстовом режиме. Понятно, что большинство современных дистрибутивов запускаются в графическом режиме, поэтому некоторые пользователи Linux даже не подозревают о том, что существует консоль. Да, таково новое поколение Linux-пользователей, которым проще использовать графический файловый менеджер, чем вводить команды. Но если вы хотите стать квалифицированным пользователем Linux, то просто обязаны знать, как работать в консоли, иначе уподобитесь Windows-пользователям, которые при каждом сбое переустанавливают операционную систему… Если вы пропустили *главу 3*, в которой рассматривается работа с консолью, настоятельно рекомендую прочитать ее!

Рассмотрим небольшую серию команд (протокол выполнения этих команд приведен на рис. 4.2):

```
touch file.txt
echo "some text" > file.txt
cat file.txt
cp file.txt file-copy.txt
cat file-copy.txt
rm file.txt
cat file.txt
mv file-copy.txt file.txt
cat file.txt
```

Первая команда (`touch`) создает в текущем каталоге файл file.txt. Вторая команда (`echo`) записывает строку `some text` в этот же файл. Обратите внимание на символ `>` — это символ перенаправления ввода/вывода, о котором мы поговорим чуть позже.

```
[root@localhost ~]# touch file.txt
[root@localhost ~]# echo "some text" > file.txt
[root@localhost ~]# cat file.txt
some text
[root@localhost ~]# cp file.txt file-copy.txt
[root@localhost ~]# cat file-copy.txt
some text
[root@localhost ~]# rm file.txt
rm: удалить обычный файл `file.txt'? y
[root@localhost ~]# cat file.txt
cat: file.txt: No such file or directory
[root@localhost ~]# mv file-copy.txt file.txt
[root@localhost ~]# cat file.txt
some text
[root@localhost ~]#
```

Рис. 4.2. Операции с файлом

Третья команда (`cat`) выводит содержимое файла — в файле записанная нами строка `some text`. Четвертая команда (`cp`) копирует файл file.txt в файл с именем file-copy.txt. После этого мы опять используем команду `cat`, чтобы вывести содержимое файла file-copy.txt, — надо же убедиться, что файл действительно скопировался.

Шестая команда (`rm`) удаляет файл file.txt. При удалении система спрашивает, хотите ли вы удалить файл. Если хотите удалить, то нужно нажать клавишу <Y>, а если нет, то <N>. Точно ли файл удален? Убедимся в этом: введите команду `cat file.txt`. Система нам сообщает, что нет такого файла.

Восьмая команда (`mv`) переименовывает файл file-copy.txt в файл file.txt. Последняя команда выводит исходный файл file.txt. Думаю, особых проблем с этими командами у вас не возникло, тем более что принцип действия этих команд вам должен быть знаком по командам DOS, которые, как квалифицированный пользователь Windows, вы должны знать наизусть.

Вместо имени файла иногда очень удобно указать *маску имени файла*. Например, у нас есть много временных файлов, имена которых заканчиваются фрагментом tmp. Для их удаления нужно воспользоваться командой: `rm *tmp`.

Если же требуется удалить все файлы в текущем каталоге, можно просто указать звездочку: `rm *`.

Аналогично, можно использовать символ `?`, который, в отличие от звездочки, заменяющей последовательность символов произвольной длины, заменяет всего один символ. Например, нам нужно удалить все файлы, имена которых состоят из трех букв и начинаются на s:

```
rm s??
```

Будут удалены файлы s14, sqm, sr6 и т. д., но не будут тронуты файлы, имена которых состоят более чем из трех букв и которые не начинаются на s.

Маски имен можно также использовать и при работе с каталогами.

4.3.2. Работа с каталогами

Основные команды для работы с каталогами приведены в табл. 4.2.

Таблица 4.2. *Основные команды для работы с каталогами*

Команда	Описание
`mkdir <каталог>`	Создание каталога
`cd <каталог>`	Изменение каталога
`ls <каталог>`	Вывод содержимого каталога
`rmdir <каталог>`	Удаление пустого каталога
`rm -r <каталог>`	Рекурсивное удаление каталога

При указании имени каталога можно использовать следующие символы:

- `.` — означает текущий каталог. Если вы введете команду `cat ./file`, то она выведет файл file, который находится в текущем каталоге;
- `..` — родительский каталог. Например, команда `cd ..` переведет вас на один уровень вверх по дереву файловой системы;
- `~` — домашний каталог пользователя (об этом мы поговорим позже).

Теперь рассмотрим пример работы с каталогами на практике. Выполните следующие команды:

```
mkdir directory
cd directory
touch file1.txt
touch file2.txt
ls
cd ..
ls directory
rm directory
rmdir directory
rm -r directory
```

Первая команда (`mkdir`) создает каталог directory в текущем каталоге. Вторая команда (`cd`) переводит (изменяет каталог) в только что созданный каталог. Следующие две команды `touch` создают в новом каталоге два файла — file1.txt и file2.txt.

Команда `ls` без указания каталога выводит содержимое текущего каталога. Команда `cd ..` переводит в родительский каталог. Как уже было отмечено, в Linux родительский каталог обозначается так: `..` (две точки), а текущий так: `.` (одна точка). То есть находясь в каталоге directory, мы можем обращаться к файлам file1.txt и file2.txt без указания каталога или же так: `./file1.txt` и `./file2.txt`.

Внимание!

Еще раз обратите внимание: в Linux в отличие от Windows для разделения элементов пути используется прямой слэш (/), а не обратный (\)!

Кроме обозначений .. и . в Linux часто используется обозначение ~ — это *домашний каталог*. Предположим, что наш домашний каталог /home/den. В нем мы создали подкаталог dir и поместили в него файл file1.txt. Полный путь к файлу можно записать так:

```
/home/den/dir/file1.txt
```

или же так:

```
~/dir/file1.txt
```

Как видите, тильда (~) заменяет часть пути. Удобно? Конечно!

Поскольку мы находимся в родительском для каталога directory каталоге, чтобы вывести содержимое только что созданного каталога в команде `ls`, нам нужно четко указать имя каталога:

```
ls directory
```

Команда `rm` используется для удаления каталога. Но что мы видим — система отказывается удалять каталог! Пробуем удалить его командой `rmdir`, но и тут отказ. Система сообщает нам, что каталог не пустой, т. е. содержит файлы. Для удаления каталога нужно удалить все файлы. Конечно, делать это не сильно хочется, поэтому проще указать опцию `-r` команды `rm` для рекурсивного удаления каталога. В этом случае сначала будут удалены все подкаталоги (и все файлы в этих подкаталогах), а затем будет удален сам каталог (рис. 4.3).

Команды `cp` и `mv` работают аналогично: для копирования (перемещения/переименования) сначала указывается каталог-источник, а потом каталог-назначение. Для каталогов желательно указывать параметр `-r`, чтобы копирование (перемещение) производилось рекурсивно.

```
[root@localhost ~]# mkdir directory
[root@localhost ~]# cd directory
[root@localhost directory]# touch file.txt
[root@localhost directory]# touch file2.txt
[root@localhost directory]# ls
file2.txt  file.txt
[root@localhost directory]# cd ..
[root@localhost ~]# ls directory
file2.txt  file.txt
[root@localhost ~]# rm directory
rm: невозможно удалить каталог `directory': Is a directory
[root@localhost ~]# rmdir directory
rmdir: `directory': Directory not empty
[root@localhost ~]# rm -r directory
rm: спуститься в каталог `directory'? y
rm: удалить пустой обычный файл `directory/file.txt'? y
rm: удалить пустой обычный файл `directory/file2.txt'? y
rm: удалить Каталог `directory'? y
[root@localhost ~]#
```

Рис. 4.3. Операции с каталогами

4.4. Команда *ln*: создание ссылок

В Linux допускается, чтобы один и тот же файл существовал в системе под разными именами. Для этого используются ссылки. Ссылки бывают двух типов: жесткие и символические. Жесткие ссылки жестко привязываются к файлу — вы не можете удалить файл, пока на него указывает хотя бы одна жесткая ссылка. А вот если на файл указывают символические ссылки, его удалению ничто не помешает.

Жесткие ссылки не могут указывать на файл, который находится за пределами файловой системы. Предположим, у вас два Linux-раздела: один корневой, а второй используется для домашних файлов пользователей и монтируется к каталогу /home корневой файловой системы. Так вот, вы не можете создать в корневой файловой системе ссылку, которая ссылается на файл в файловой системе, подмонтированной к каталогу /home. Это очень важная особенность жестких ссылок. Если вам нужно создать ссылку на файл, который находится за пределами файловой системы, вам следует использовать символические ссылки.

Для создания ссылок используется команда `ln`:

```
ln file.txt link1
ln -s file.txt link2
```

Первая команда создает жесткую ссылку link1, ссылающуюся на текстовый файл file.txt. Вторая команда создает символическую ссылку link2, которая ссылается на этот же текстовый файл file.txt.

Модифицируя ссылку (все равно какую — link1 или link2), вы автоматически модифицируете исходный файл — file.txt.

Особого внимания заслуживает операция удаления. По идее, если вы удаляете ссылку link2, файл file.txt также должен быть удален, но не тут-то было — вы не можете его удалить до тех пор, пока на него указывает хоть одна жесткая ссылка. При удалении ссылки link2 просто будет удалена символьная ссылка, но жесткая ссылка и сам файл останутся. Если же вы удалите ссылку link1, будет удален и файл file.txt, поскольку на него больше не ссылается ни одна жесткая ссылка.

4.5. Команды *chown*, *chmod* и *chattr*

4.5.1. Команда *chmod*: права доступа к файлам и каталогам

Для каждого каталога и файла вы можете задать права доступа. Точнее, права доступа автоматически задаются при создании каталога/файла, а вам при необходимости можно их изменить. Какая может быть необходимость? Например, вам нужно, чтобы к вашему файлу-отчету смогли получить доступ пользователи — члены вашей группы. Или вы создали обычный текстовый файл, содержащий инструкции командного интерпретатора. Чтобы этот файл стал сценарием, вам нужно установить право на выполнение для этого файла.

Существуют три права доступа: чтение (`r`), запись (`w`), выполнение (`x`). Для каталога право на выполнение означает право на просмотр содержимого каталога.

Вы можете установить разные права доступа для владельца (т. е. для себя), для группы владельца (т. е. для всех пользователей, входящих в одну с владельцем группу) и для прочих пользователей. Пользователь root может получить доступ к любому файлу или каталогу вне зависимости от прав, которые вы установили.

Чтобы просмотреть текущие права доступа, введите команду:

```
ls -l <имя файла/каталога>
```

Например,

```
ls -l video.txt
```

В ответ программа выведет следующую строку:

```
-r--r----- 1 den group 300 Apr 11 11:11 video.txt
```

В этой строке фрагмент `-r--r-----` описывает права доступа:

- первый символ — это признак каталога. Сейчас перед нами файл. Если бы перед нами был каталог, то первый символ был бы символом `d` (от *directory*);
- последующие три символа (`r--`) определяют *права доступа владельца файла или каталога*. Первый символ — это чтение, второй — запись, третий — выполнение. Как можно видеть, владельцу разрешено только чтение этого файла, запись и выполнение запрещены, поскольку в правах доступа режимы `w` и `x` не определены;
- следующие три символа (`r--`) задают *права доступа для членов группы владельца*. Права такие же, как и у владельца: можно читать файл, но нельзя изменять или запускать;
- последние три символа (`---`) задают *права доступа для прочих пользователей*. Прочие пользователи не имеют права ни читать, ни изменять, ни выполнять файл. При попытке получить доступ к файлу они увидят сообщение **`Access denied`**.

> ***Примечание***
>
> После прав доступа команда `ls` выводит имя владельца файла, имя группы владельца, размер файла, дату и время создания, а также имя файла.

Права доступа задаются командой `chmod`. Существуют два способа указания прав доступа: *символьный* (когда указываются символы, задающие право доступа, — `r`, `w`, `x`) и *абсолютный*.

Так уж заведено, что в мире UNIX чаще пользуются абсолютным методом. Разберемся, в чем он заключается. Рассмотрим следующий набор прав доступа:

```
rw-r-----
```

Данный набор прав доступа предоставляет владельцу право чтения и модификации файла (`rw-`), запускать файл владелец не может. Члены группы владельца могут только просматривать файл (`r--`), а все остальные пользователи не имеют вообще никакого доступа к файлу.

Возьмем отдельный набор прав, например для владельца: `rw-`

Чтение разрешено — мысленно записываем 1, запись разрешена — запоминаем еще 1, а вот выполнение запрещено, поэтому запоминаем 0. Получается число 110.

Если из двоичной системы перевести число 110 в восьмеричную, получится число 6. Для перевода можно воспользоваться табл. 4.3.

Таблица 4.3. *Преобразование чисел из двоичной системы в восьмеричную*

Двоичная система	Восьмеричная система	Двоичная система	Восьмеричная система
000	0	100	4
001	1	101	5
010	2	110	6
011	3	111	7

Аналогично произведем разбор прав для членов группы владельца. Получится двоичное 100, т. е. восьмеричное 4. С третьим набором (---) все вообще просто — это 000, т. е. 0.

Записываем полученные числа в восьмеричной системе в порядке владелец-группа-остальные. Получится число 640 — это и есть права доступа. Для того чтобы установить эти права доступа, выполните команду:

```
chmod 640 <имя_файла>
```

Наиболее популярные права доступа:

- 644 — владельцу можно читать и изменять файл, остальным пользователям — только читать;
- 666 — читать и изменять файл можно всем пользователям;
- 777 — всем можно читать, изменять и выполнять файл.

> ***Примечание***
>
> Напомню, что для каталога право выполнения — это право просмотра оглавления каталога.

Иногда символьный метод оказывается проще. Например, у нас есть файл script, который нужно сделать исполнимым, для этого можно применить команду:

```
chmod +x script
```

Для того чтобы снять право выполнения, указывается параметр `-x`:

```
chmod -x script
```

Подробнее о символьном методе вы сможете прочитать в руководстве по команде `chmod` (выполнив команду `man chmod`).

4.5.2. Команда *chown*: смена владельца файла

Если вы хотите "подарить" кому-то файл, т. е. сделать какого-то пользователя владельцем файла, то вам нужно применить команду `chown`:

```
chown пользователь файл
```

Примечание

Возможно, что после изменения владельца файла вы сами не сможете получить к нему доступ, ведь владельцем будете уже не вы.

4.5.3. Специальные права доступа (SUID и SGID)

Мы рассмотрели обычные права доступа к файлам, но в Linux есть еще так называемые *специальные права доступа*: SUID (Set User ID root) и SGID (Set Group ID root).

Данные права доступа позволяют обычным пользователям запускать программы, требующие для своего запуска привилегий пользователя root. Например, демон pppd требует привилегий root, но чтобы каждый раз при установке PPP-соединения (модемное, ADSL-соединение) не входить в систему под именем root, достаточно установить специальные права доступа для демона pppd. Делается это так:

```
chmod u+s /usr/sbin/pppd
```

Однако не нужно увлекаться такими решениями, поскольку каждая программа, для которой установлен бит SUID, является потенциальной "дырой" в безопасности вашей системы. Для выполнения программ, требующих прав root, намного рациональнее использовать программы `sudo` и `su` (описание которых можно получить по командам `man sudo` и `man su`).

4.5.4. Команда *chattr*: атрибуты файла, запрет изменения файла

С помощью команды `chattr` можно изменить атрибуты файла. Параметр `+` устанавливает атрибут, а параметр `-` атрибут снимает. Например:

```
# chattr +i /boot/grub/menu.lst
```

Данная команда устанавливает атрибут `i`, запрещающий любое изменение, переименование и удаление файла. Установить этот атрибут, равно как и снять его, имеет право только суперпользователь или процесс с возможностью `CAP_LINUX_IMMUTABLE`. Чтобы изменить файл, нужно очистить атрибут с помощью команды:

```
# chattr -i /boot/grub/menu.lst
```

Если установить атрибут `j`, то все данные, прежде чем они будут записаны непосредственно в файл, будут сохранены в журнал ext3. Данный атрибут имеет смысл, только если файловая система смонтирована с опциями `data=ordered` или `data=writeback` *(см. разд. 4.7)*. Когда файловая система смонтирована с опцией `data=journal`, данный атрибут не имеет значения, поскольку все данные файла и так уже журналируются. Об остальных атрибутах вы сможете прочитать в справочной системе:

```
man chattr
```

4.6. Монтирование файловых систем

4.6.1. Команды *mount* и *umount*

Чтобы работать с какой-либо файловой системой, необходимо *примонтировать* ее к корневой файловой системе. Например, вставив в дисковод дискету, нужно подмонтировать файловую систему дискеты к корневой файловой системе — только так мы сможем получить доступ к файлам и каталогам, которые на этой дискете записаны. Аналогичная ситуация с жесткими, оптическими дисками и другими носителями данных.

Если вы хотите заменить сменный носитель данных (дискету, компакт-диск), вам нужно сначала размонтировать файловую систему, затем извлечь носитель данных, установить новый и заново смонтировать файловую систему. В случае с дискетой о размонтировании должны помнить вы сами, поскольку при этом выполняется синхронизация буферов ввода/вывода и файловой системы, т. е. данные физически записываются на диск, если это еще не было сделано. А компакт-диск система не разрешит вам извлечь, если он не размонтирован. В свою очередь, размонтировать файловую систему можно, только когда ни один процесс ее не использует.

При завершении работы системы (перезагрузке, выключении компьютера) размонтирование всех файловых систем выполняется автоматически.

Команда монтирования (ее нужно выполнять с привилегиями root) выглядит так:

```
# mount [опции] <устройство> <точка монтирования>
```

Точка монтирования — это каталог, через который будет осуществляться доступ к монтируемой файловой системе. Например, если вы подмонтировали компакт-диск к каталогу /mnt/cdrom, то получить доступ к файлам и каталогам, записанным на компакт-диске, можно будет через точку монтирования (именно этот каталог /mnt/cdrom). Точкой монтирования может быть любой каталог корневой файловой системы, хоть /ааа-111. Главное, чтобы этот каталог существовал на момент монтирования файловой системы.

В некоторых современных дистрибутивах запрещен вход в систему под именем суперпользователя — root. Поэтому для выполнения команд с привилегиями root вам нужно использовать команду `sudo`. Например, чтобы выполнить команду монтирования привода компакт-диска, вам нужно ввести команду:

```
sudo mount /dev/hdc /mnt/cdrom
```

Перед выполнением команды `mount` команда `sudo` попросит вас ввести пароль root. Если введенный пароль правильный, то будет выполнена команда `mount`.

Для размонтирования файловой системы используется команда `umount`:

```
# umount <устройство или точка монтирования>
```

4.6.2. Файлы устройств и монтирование

В этой главе мы уже говорили о файлах устройств. Здесь мы вернемся к ним снова, но в контексте монтирования файловой системы.

Как уже было отмечено, для Linux нет разницы между устройством и файлом. Все устройства системы представлены в корневой файловой системе как обычные файлы. Например, /dev/fd0 — это ваш дисковод для гибких дисков, /dev/hda (/dev/sda) — жесткий диск. Файлы устройств хранятся в каталоге /dev.

Жесткие диски

С жесткими дисками сложнее всего, поскольку одно и то же устройство может в разных версиях одного и того же дистрибутива называться по-разному. Например, мой IDE-диск, подключенный как первичный мастер, в Fedora 5 все еще назывался /dev/hda, а в Fedora 8 он стал называться /dev/sda. Раньше накопители, подключающиеся к интерфейсу IDE (PATA), назывались /dev/hd*x*, а SCSI/SATA-накопители — /dev/sd*x* (где в обоих случаях *x* — буква устройства).

После принятия udev и глобального уникального идентификатора устройств (UUID) все дисковые устройства, вне зависимости от интерфейса подключения (PATA, SATA, SCSI), называются /dev/sd*x*, где *x* — буква устройства. Все современные дистрибутивы поддерживают udev и UUID. Так что не удивляйтесь, если вдруг ваш старенький IDE-винчестер будет назван /dev/sda. С одной стороны, это вносит некоторую путаницу *(см. разд. 4.6.5)*. С другой стороны, все современные компьютеры оснащены именно SATA-дисками (т. к. PATA-диски уже устарели, а SCSI — дорогие), а на современных материнских платах только один контроллер IDE (PATA), потому многие пользователи даже ничего не заметят.

> ***Пояснение***
>
> udev — это менеджер устройств, используемый в ядрах Linux версии 2.6. Пришел на смену более громоздкой псевдофайловой системе devfs. Управляет всеми манипуляциями с файлами из каталога /dev.

Рассмотрим ситуацию с жесткими дисками чуть подробнее. Пусть у нас есть устройство /dev/sda. На жестком диске, понятное дело, может быть несколько разделов (логических дисков). В нашем случае на диске имеются три раздела, которые в Windows называются C:, D: и E:. Диск C: обычно является загрузочным (активным), поэтому он будет записан в самом начале диска. Нумерация разделов жесткого диска в Linux начинается с 1, поэтому в большинстве случаев диску C: будет соответствовать имя /dev/sda1 — первый раздел на первом жестком диске.

Резонно предположить, что двум оставшимся разделам (D: и E:) были присвоены имена /dev/sda2 и /dev/sda3. Это может быть так — и не так. Сейчас поясню. Раздел может быть первичным (primary partition), расширенным (extended partition) или логическим (logical partition). Всего на диске может быть или четыре первичных раздела, или три первичных и один расширенный.

Пусть на жестком диске есть четыре первичных раздела, для которых зарезервированы номера 1, 2, 3, 4. Если разделы D: и E: — первичные, то им будут присвоены имена /dev/sda2 и /dev/sda3. Но в большинстве случаев данные разделы являются логическими, а логические разделы содержатся в расширенном разделе (там может быть максимум 11 логических разделов). При этом в Windows расширенному разделу не присваивается буква, потому что этот раздел не содержит данных пользователя,

а только информацию о логических разделах. Логические разделы именуются, начиная с 5, т. е. если разделы D: и E: — логические, то им будут присвоены имена /dev/sda5 и /dev/sda6 соответственно.

Узнать номер раздела очень просто: достаточно запустить утилиту, работающую с таблицей разделов диска — `fdisk` (будет рассмотрена в *главе 9*).

Чтобы узнать номера разделов первого жесткого диска (/dev/hda), введите команду:

```
# /sbin/fdisk /dev/sda
```

После этого вы увидите приглашение `fdisk`. В ответ на приглашение нужно ввести `p` и нажать клавишу <Enter>. Вы увидите таблицу разделов (рис. 4.4). После этого для выхода из программы введите `q` и нажмите клавишу <Enter>.

```
1) программами, запускаемым при загрузке (напр., старые версии LILO)
2) загрузкой и программами разметки из других ОС
   (напр., DOS FDISK, OS/2 FDISK)

Команда (m для справки): p

Диск /dev/sda: 160.0 ГБ, 160041885696 байт
255 heads, 63 sectors/track, 19457 cylinders
Units = цилиндры of 16065 * 512 = 8225280 bytes
Disk identifier: 0xe905e905

Устр-во Загр     Начало       Конец       Блоки   Id  Система
/dev/sda1   *           1         543     4361616    b  W95 FAT32
/dev/sda2             544       19457   151926705    f  W95 расшир. (LBA)
/dev/sda5             544        1021     3839503+  83  Linux
/dev/sda6            1022        1759     5927953+  83  Linux
/dev/sda7            1760        1825      530113+  82  Linux своп / Solaris
/dev/sda8            1826        5963    33238453+   b  W95 FAT32
/dev/sda9            5964       10101    33238453+   b  W95 FAT32
/dev/sda10          10102       14268    33471396    b  W95 FAT32
/dev/sda11          14269       16949    21535101    b  W95 FAT32
/dev/sda12          16950       19457    20145478+   b  W95 FAT32

Команда (m для справки):
```

Рис. 4.4. Таблица разделов жесткого диска

На рис. 4.4 изображена таблица разделов моего первого жесткого диска. Первый раздел (это мой диск C:, где установлена система Windows) — первичный. Сразу после него расположен расширенный раздел (его номер — 2). Следующий за ним — логический раздел (номер 5). Разделы с номерами 3 и 4 пропущены, потому что их нет на моем жестком диске. Это те самые первичные разделы, которые я не создал — они мне не нужны.

Приводы оптических дисков

Любое из устройств sd*x* может быть приводом для чтения CD- или DVD-дисков. Если система видит, что устройство является приводом CD-ROM, то автоматически создается ссылка /dev/cdrom. А если ваш привод умеет также читать и DVD-диски, то в каталоге /dev появится еще одна ссылка — /dev/dvd. Например, мой DVD-RW подключен как первичный подчиненный (/dev/sdb), а в каталоге /dev есть три фай-

ла: /dev/sdb, /dev/cdrom, /dev/dvd. Обратиться к устройству можно, используя любой из этих файлов.

Для монтирования привода для чтения оптических дисков нужно ввести одну из трех команд:

```
# mount /dev/sdb /mnt/cdrom
# mount /dev/cdrom /mnt/cdrom
# mount /dev/dvd /mnt/cdrom
```

После этого обратиться к файлам, записанным на диске, можно будет через каталог /mnt/cdrom. Напомню, что каталог /mnt/cdrom должен существовать.

Дискеты и USB-накопители

Аналогичная ситуация и с дискетами. В системе может быть установлено два дисковода для дискет — первый (/dev/fd0) и второй (/dev/fd1).

Для их монтирования можно использовать команды:

```
# mount /dev/fd0 /mnt/floppy
# mount /dev/fd1 /mnt/floppy
```

В Windows-терминологии устройство /dev/fd0 — это диск A:, а устройство /dev/fd1 — диск B:.

USB-накопители определяются как жесткие диски. Если в вашей системе есть один жесткий диск, то его имя будет /dev/sda. Когда вы подключите флешку или другой USB-накопитель (мобильный жесткий диск, card reader и т. д.), то в вашей системе появится новое устройство — /dev/sdb. Осталось только подмонтировать его. Далее мы поговорим о монтировании USB-накопителей более подробно.

Об автоматическом монтировании

Нужно отметить, что CD/DVD-приводы, а также USB-накопители монтируются автоматически (к каталогу /media/<ID накопителя>), поэтому все, что было сказано о монтировании таких носителей, — исключительно для общего развития.

4.6.3. Опции монтирования файловых систем

Теперь, когда мы знаем номер раздела, можно подмонтировать его файловую систему. Делается это так:

```
# mount <раздел> <точка монтирования>
```

Например:

```
# mount /dev/sda5 /mnt/win_d
```

У команды `mount` довольно много опций, но на практике наиболее часто используются только некоторые из них: `-t`, `-r`, `-w`, `-a`.

- Параметр `-t` позволяет задать тип файловой системы. Обычно программа сама определяет файловую систему, но иногда это у нее не получается. Тогда мы должны ей помочь. Формат использования этого параметра следующий:

```
# mount -t <файловая система> <устройство> <точка монтирования>
```

Например:

```
# mount -t iso9660 /dev/hdc /mnt/cdrom
```

Вот опции для указания наиболее популярных монтируемых файловых систем:

- `ext2` или `ext3` — файловая система Linux;
- `iso9660` — указывается при монтировании CD-ROM;
- `vfat` — FAT, FAT32 (поддерживается Windows 9*x*, ME, XP);
- `ntfs` — NT File System (поддерживается Windows NT, XP), будет использована стандартная поддержка NTFS, при которой NTFS-раздел доступен только для чтения;
- `ntfs-3g` — будет использован модуль ntfs-3g, входящий в большинство современных дистрибутивов. Данный модуль позволяет производить запись информации на NTFS-разделы.

> ***Примечание***
>
> Если в вашем дистрибутиве нет модуля ntfs-3g, т. е. при попытке указания данной файловой системы вы увидели сообщение об ошибке, тогда вы можете скачать его с сайта **www.ntfs-3g.org**. На данном сайте доступны как исходные коды, так и уже откомпилированные для разных дистрибутивов пакеты.

- Параметр `-r` монтирует указанную файловую систему в режиме "только чтение".
- Параметр `-w` монтирует файловую систему в режиме "чтение/запись". Данный параметр используется по умолчанию для файловых систем, поддерживающих запись (например, NTFS по умолчанию запись не поддерживает, как и файловые системы CD/DVD-дисков).
- Параметр `-a` используется для монтирования всех файловых систем, указанных в файле /etc/fstab (кроме тех, для которых указано `noauto` — такие файловые системы нужно монтировать вручную). При загрузке системы вызывается команда `mount` с параметром `-a`.

Если вы не можете смонтировать NTFS-раздел с помощью опции `ntfs-3g`, то, вероятнее всего, он был неправильно размонтирован (например, работа Windows не была завершена корректно). В этом случае для монтирования раздела нужно использовать опцию `-o force`, например:

```
sudo mount -t ntfs-3g /dev/sdb1 /media/usb -o force
```

4.6.4. Монтирование разделов при загрузке

Если вы не хотите при каждой загрузке монтировать постоянные файловые системы (например, ваши Windows-разделы), то вам нужно прописать их в файле /etc/fstab. Обратите внимание: в этом файле не нужно прописывать файловые системы сменных носителей (дисковода, CD/DVD-привода, Flash-диска). Следует отметить, что программы установки некоторых дистрибутивов, например Mandriva, читают таблицу разделов и автоматически заполняют файл /etc/fstab. В результате

все ваши Windows-разделы доступны сразу после установки системы. К сожалению, не все дистрибутивы могут похвастаться такой интеллектуальностью, поэтому вам нужно знать формат файла fstab:

```
устройство точка_монтирования тип_ФС опции флаг_РК флаг_проверки
```

Здесь: `тип_ФС` — это тип файловой системы, а `флаг_РК` — флаг резервного копирования. Если он установлен (`1`), то программа `dump` заархивирует данную файловую систему при создании резервной копии. Если не установлен (`0`), то резервная копия этой файловой системы создаваться не будет. `Флаг проверки` устанавливает, будет ли данная файловая система проверяться на наличие ошибок программой `fsck`. Проверка производится в двух случаях:

- если файловая система размонтирована некорректно;
- если достигнуто максимальное число операций монтирования для этой файловой системы.

Поле опций содержит важные параметры файловой системы. Некоторые из них представлены в табл. 4.4.

Таблица 4.4. *Опции монтирования файловой системы в файле /etc/fstab*

Опция	Описание
`auto`	Файловая система должна монтироваться автоматически при загрузке. Опция используется по умолчанию, поэтому ее указывать не обязательно
`noauto`	Файловая система не монтируется при загрузке системы (при выполнении команды `mount -a`), но ее можно смонтировать вручную с помощью все той же команды `mount`
`defaults`	Используется стандартный набор опций, установленных по умолчанию
`exec`	Разрешает запуск выполняемых файлов для данной файловой системы. Эта опция используется по умолчанию
`noexec`	Запрещает запуск выполняемых файлов для данной файловой системы
`ro`	Монтирование в режиме "только чтение"
`rw`	Монтирование в режиме "чтение/запись". Используется по умолчанию для файловых систем, поддерживающих запись
`user`	Данную файловую систему разрешается монтировать/размонтировать обычному пользователю (не root)
`nouser`	Файловую систему может монтировать только пользователь root. Используется по умолчанию
`umask`	Определяет маску прав доступа при создании файлов. Для файловых систем не Linux'а маску нужно установить так: `umask=0`
`utf8`	Применяется только на дистрибутивах, которые используют кодировку UTF-8 в качестве кодировки локали. В старых дистрибутивах (где используется KOI8-R) для корректного отображения русских имен файлов на Windows-разделах нужно задать параметры `iocharset=koi8-u,codepage=866`

Примечание

Редактировать файл /etc/fstab, как и любой другой файл из каталога /etc, можно в любом текстовом редакторе (например, `gedit`, `kate`), но перед этим нужно получить права root (команды `su` или `sudo`).

Рассмотрим небольшой пример:

```
/dev/sdc /mnt/cdrom auto umask=0,user,noauto,ro,exec 0 0
/dev/sda1 /mnt/win_c vfat umask=0,utf8 0 0
```

Первая строка — это строка монтирования файловой системы компакт-диска, а вторая — строка монтирования диска C:.

- Начнем с первой строки. `/dev/hdc` — это имя устройства CD-ROM. Точка монтирования — `/mnt/cdrom`. Понятно, что этот каталог должен существовать. Обратите внимание: в качестве файловой системы не указывается жестко iso9660, поскольку компакт-диск может быть записан в другой файловой системе, поэтому в качестве типа файловой системы задано `auto`, т. е. автоматическое определение. Теперь идет довольно длинный набор опций. Ясно, что `umask` установлен в ноль, поскольку файловая система компакт-диска не поддерживает права доступа Linux. Параметр `user` говорит о том, что данную файловую систему можно монтировать обычному пользователю. Параметр `noauto` запрещает автоматическое монтирование этой файловой системы, что правильно — ведь на момент монтирования в приводе может и не быть компакт-диска. Опция `ro` разрешает монтирование в режиме "только чтение", а `exec` разрешает запускать исполнимые файлы. Понятно, что компакт-диск не нуждается ни в проверке, ни в создании резервной копии, поэтому два последних флага равны нулю.
- Вторая строка проще. Первые два поля — это устройство и точка монтирования. Третье — тип файловой системы. Файловая система постоянна, поэтому можно явно указать тип файловой системы (`vfat`), а не `auto`. Опция `umask`, как и в предыдущем случае, равна нулю. Указание опции `utf8` позволяет корректно отображать русскоязычные имена файлов и каталогов.

4.6.5. Подробно о UUID и файле /etc/fstab

Пока вы еще не успели забыть формат файла /etc/fstab, нужно поговорить о UUID (Universally Unique Identifier), или о *длинных именах* дисков. В некоторых дистрибутивах, например в Ubuntu, вместо имени носителя (первое поле файла fstab) указывается его ID, поэтому fstab выглядит устрашающе, например вот так:

```
# /dev/sda6
UUID=1f049af9-2bdd-43bf-a16c-ff5859a4116a / ext3 defaults 0  1
# /dev/sda1
UUID=45AE-84D9  /media/hda1  vfat  defaults,utf8,umask=007 0 0
```

В SUSE идентификаторы устройств указываются немного иначе:

```
/dev/disk/by-id/scsi-SATA_WDC_WD1600JB-00_WD-WCANM7959048-part5 /
ext3  acl,user_xattr 1 1
/dev/disk/by-id/scsi-SATA_WDC_WD1600JB-00_WD-WCANM7959048-part7 swap swap
defaults 0 0
```

Понятно, что использовать короткие имена вроде /dev/sda1 намного проще, чем идентификаторы в стиле `1f049af9-2bdd-43bf-a16c-ff5859a4116a`. Использование имен дисков еще никто не отменял, поэтому вместо идентификатора носителя можете смело указывать его файл устройства — так вам будет значительно проще!

Но все же вам нужно знать соответствие длинных имен коротким именам устройств. Ведь система использует именно эти имена, а в файле /etc/fstab не всегда указывается, какой идентификатор принадлежит какому короткому имени устройства (или указывается, но не для всех разделов).

Узнать "длинные имена" устройства можно с помощью простой команды:

```
ls -l /dev/disk/by-uuid/
```

Результат выполнения этой команды приведен на рис. 4.5.

```
[den@localhost ~]$ ls -l /dev/disk/by-uuid/
итого 0
lrwxrwxrwx 1 root root 10 Фев 12 15:25 1c3b8bd3-c26f-449e-9eba-8f254fefe814 -> .
./../sda1
lrwxrwxrwx 1 root root 10 Фев 12 15:25 3106fa17-65ef-42e9-a1d4-2313daee96b5 -> .
./../sda2
[den@localhost ~]$ ls -l /dev/disk/by-label
итого 0
lrwxrwxrwx 1 root root 10 Фев 12 15:25 SWAP-sda2 -> ../../sda2
lrwxrwxrwx 1 root root 10 Фев 12 15:25 \x2f -> ../../sda1
[den@localhost ~]$
```

Рис. 4.5. Соответствие длинных имен дисков коротким

Спрашивается, зачем были введены длинные имена, если короткие имена были удобнее, во всяком случае для пользователей? Оказывается, разработчики Linux в первую очередь и заботились как раз о пользователях. Возьмем обычный IDE-диск. Как известно, данный диск можно подключить либо к первичному (primary), либо к вторичному (secondary), если он есть, контроллеру. В зависимости от положения перемычки выбора режима винчестер может быть либо главным устройством (master), либо подчиненным (slave). Таким образом, в зависимости от контроллера, к которому подключается диск, изменяется его короткое имя — hda (primary master), hdb (primary slave), hdc (secondary master), hdd (secondary slave). То же самое происходит с SATA/SCSI-винчестерами — при изменении параметров подключения изменяется и короткое имя устройства.

При использовании же длинных имен идентификатор дискового устройства остается постоянным вне зависимости от типа подключения устройства к контроллеру. Именно поэтому длинные имена дисков часто также называются *постоянными* именами (persistent name). Получается, что раньше вы могли ошибочно подключить жесткий диск немного иначе, и разделы, которые назывались, скажем, /dev/hda*N*, стали называться /dev/hdb*N*. Понятно, что загрузить Linux с такого диска не получится, поскольку везде указаны другие имена устройств. Если же используются длинные имена дисков, система загрузится в любом случае, как бы вы ни подключили жесткий диск. Удобно? Конечно.

Но это еще не все. Постоянные имена — это только первая причина. Вторая причина заключается в обновлении библиотеки libata. В новой версии libata все PATA-устройства именуются не как hd*x*, а как sd*x*, что (как отмечалось в этой главе ранее) вносит некую путаницу. Длинные имена дисков от этого не изменяются, поэтому они избавляют пользователя от беспокойства по поводу того, что его старый IDE-диск вдруг превратился в SATA/SCSI-диск.

При использовании UUID однозначно идентифицировать раздел диска можно несколькими способами:

- `UUID=45AE-84D9 /media/sda1 vfat defaults,utf8,umask=007, gid=46 0 0` — здесь с помощью параметра UUID указывается идентификатор диска;
- `/dev/disk/by-id/scsi-SATA_WDC_WD1600JB-00_WD-WCANM7959048-part7 swap swap defaults 0 0` — здесь указывается длинное имя устройства диска;
- `LABEL=/ / ext3 defaults 1 1` — самый компактный третий способ, позволяющий идентифицировать устройства по их метке.

> ***Примечание***
>
> Первый способ получения длинного имени в англоязычной литературе называется by-uuid, т. е. длинное имя составляется по UUID, второй способ называется by-id, т. е. по аппаратному идентификатору устройства. Третий способ называется by-label — по метке. Просмотреть соответствие длинных имен коротким можно с помощью команд:
>
> ```
> ls -l /dev/disk/by-uuid
> ls -l /dev/disk/by-id
> ls -l /dev/disk/by-label
> ```
>
> Но есть еще и четвертый способ, который называется by-path. В этом случае имя генерируется по sysfs. Данный способ является наименее используемым, поэтому вы редко столкнетесь с ним.

Узнать метки разделов можно с помощью команды:

```
ls -lF /dev/disk/by-label
```

Установить метку можно с помощью команд, указанных в табл. 4.5.

Таблица 4.5. *Команды для установки меток разделов*

Файловая система	Команда
ext2/ext3/ext4	`# e2label /dev/XXX <метка>`
ReiserFS	`# reiserfstune -l <метка> /dev/XXX`
JFS	`# jfs_tune -L <метка> /dev/XXX`
XFS	`# xfs_admin -L <label> /dev/XXX`
FAT/FAT32	Только средствами Windows
NTFS	`# ntfslabel /dev/XXX <метка>`

В файле /etc/fstab вы можете использовать длинные имена в любом формате. Можно указывать имена устройств в виде: `/dev/disk/by-uuid/*`, `/dev/disk/by-id/*` или `/dev/disk/by-label/*`, можно использовать параметры `UUID=идентификатор` или `LABEL=метка`. Используйте тот способ, который вам больше нравится.

4.6.6. Монтирование Flash-дисков

В последнее время очень популярна Flash-память. Уже сегодня Flash-память, точнее Flash-диски (они же USB-диски), построенные с использованием Flash-памяти, практически вытеснили обычные дискеты — они очень компактны и позволяют хранить довольно большие объемы информации. Сегодня никого не удивишь небольшим брелоком, вмещающим до 8 Гбайт информации.

Принцип использования Flash-диска очень прост — достаточно подключить его к шине USB, и через несколько секунд система определит диск. После этого с ним можно будет работать как с обычным диском. Да, Flash-диски не очень шустры, но молниеносной реакции от них никто и не ожидает — во всяком случае, они выглядят настоящими спринтерами на фоне обычных дискет.

Технология Flash-памяти нашла свое применение в различных портативных устройствах — от мобильных телефонов до цифровых фотоаппаратов. Вы можете подключить мобильник к компьютеру и работать с ним как с обычным диском — записывать на него мелодии и картинки. Аналогичная ситуация и с цифровым фотоаппаратом: когда вы фотографируете, то фотографии и видеоролики записываются на его Flash-память. Потом вам нужно подключить его к компьютеру и просто скопировать фотографии. Вы также можете записать фотографии (или другие файлы — не имеет значения) на фотоаппарат, используя встроенную Flash-память как большую дискету — для переноса своих файлов.

Все современные дистрибутивы умеют автоматически монтировать Flash-диски. После монтирования открывается окно с предложением просмотреть содержимое диска или же импортировать фотографии (в зависимости от типа подключенного устройства — обычный USB-диск или фотоаппарат).

Понятно, что нам, как настоящим линуксоидам, интересно, как самостоятельно смонтировать Flash-диск. Оказывается, тут все просто. USB-диск — это обычный накопитель, и его можно увидеть в каталоге /dev/disk/by-id. Напомню, что способ by-id подразумевает получение длинного имени по аппаратному идентификатору устройства, а поэтому с помощью каталога /dev/disk/by-id проще всего найти длинное имя USB-диска среди имен других накопителей: в его начале будет префикс `usb_`. Введите команду:

```
ls -l /dev/disk/by-id | grep usb
```

Результат выполнения этой команды представлен на рис. 4.6.

```
[root@localhost 001]# ls -l /dev/disk/by-id | grep usb
lrwxrwxrwx 1 root root  9 Фев 12 17:41 usb-AIT_Card_Reader_0_DISK01-0:0 -> ../../sdb
lrwxrwxrwx 1 root root 10 Фев 12 17:41 usb-AIT_Card_Reader_0_DISK01-0:0-part1 -> ../../sdb1
[root@localhost 001]#
```

Рис. 4.6. USB-диск найден

Исходя из рис. 4.6, для монтирования Flash-диска нужно выполнить команду:

```
# mount /dev/sdb1 /mnt/flash
```

4.7. Настройка журнала файловой системы ext3

Журналируемая файловая система имеет три режима работы: `journal`, `ordered` и `writeback`. Первый режим является самым медленным, но он позволяет минимизировать потери ваших данных в случае сбоя системы (или отключения питания). В этом режиме в системный журнал записывается все, что только можно, —это позволяет максимально восстановить файловую систему в случае сбоя.

В последовательном режиме (`ordered`) в журнал заносится информация только об изменении метаданных (служебных данных файловой системы). Данный режим используется по умолчанию и является компромиссным вариантом между производительностью и отказоустойчивостью.

Самым быстрым является режим обратной записи (`writeback`). Но использовать его я вам не рекомендую, поскольку особого толку от него не будет. Проще тогда уже при установке Linux выбрать файловую систему ext2 вместо ext3/ext4.

Если отказоустойчивость для вас на первом месте — выбирайте режим `journal`, во всех остальных случаях лучше выбрать `ordered`. Выбор режима осуществляется редактированием файла /etc/fstab. Например:

```
# Режим ordered используется по умолчанию,
# поэтому ничего указывать не нужно
/dev/sda1 / ext3 defaults 1 0
# На этом разделе важные данные — используем режим journal
/dev/sda2 /var ext3 data=journal 1 0
# Здесь ничего важного нет — режим writeback
/dev/sda3 /opt ext3 data=writeback 0 0
```

После изменения этого файла выполните команду:

```
# mount -a
```

Данная команда заново смонтирует все файловые системы, чтобы изменения вступили в силу.

4.8. Файловая система ext4

Файловая система ext4 заслуживает отдельного разговора. Все, что было сказано о файловых системах ранее, справедливо и для ext4, но у новой файловой системы есть ряд особенностей, о которых мы сейчас и поговорим.

Поддержка ext4 как стабильной файловой системы появилась в ядре Linux версии 2.6.28. Если сравнивать эту файловую систему с ext3, то производительность и надежность новой файловой системы существенно увеличена, а максимальный размер раздела доведен до 1024 петабайт (1 эксбибайт). Максимальный размер файла — более 2 Тбайт. Ресурс Phoronix (**www.phoronix.com**) произвел тестирование новой файловой системы на SSD-накопителе (такие накопители устанавливаются на современные нетбуки) — результат, как говорится, налицо: ext4 почти в два раза превзошла файловые системы ext3, XFS, JFS и ReiserFS.

Впрочем, когда я установил Fedora 11 на рабочую станцию, прироста производительности при работе с файлами мне почувствовать не удалось. Однако производительность — это не основной конек ext4. Но обо всем по порядку.

4.8.1. Сравнение ext3 и ext4

Описание особенностей файловой системы ext4 и ее преимуществ по сравнению с ext3 сведены в табл. 4.6.

Таблица 4.6. *Особенности ext4*

Особенность	Комментарий
Увеличенный размер файла и файловой системы	Для ext3 максимальный размер файловой системы составляет 32 Тбайт, а файла — 2 Тбайт, но на практике ограничения были более жесткими. Так, в зависимости от архитектуры, максимальный размер тома составлял до 2 Тбайт, а максимальный размер файла — до 16 Гбайт. В случае с ext4 максимальный размер тома составляет 1 эксбибайт (EiB) — это 2^{60} байт. Максимальный размер файла составляет 16 Тбайт. Такие объемы информации пока не нужны обычным пользователям, однако весьма пригодятся на серверах, работающих с большими дисковыми массивами
Экстенты	Основной недостаток ext3 — ее метод выделения места на диске. Дисковые ресурсы выделялись с помощью битовых карт свободного места, а такой способ не отличается ни скоростью, ни масштабируемостью. Получилось, что ext3 более эффективна для небольших файлов, но совсем не подходит для хранения больших файлов. Для улучшения выделения ресурсов и более эффективной организации данных в ext4 были введены *экстенты*. Экстент — это способ представления непрерывной последовательности блоков памяти. Благодаря использованию экстентов сокращается количество метаданных (служебных данных файловой системы), поскольку вместо информации о том, где находится каждый блок памяти, экстент содержит информацию о том, где находится большой список непрерывных блоков памяти. Для эффективного представления маленьких файлов в экстентах применяется уровневый подход, а для больших файлов используются деревья экстентов. Например, один индексный дескриптор может ссылаться на четыре экстента, каждый из которых может ссылаться на другие индексные дескрипторы и т. д. Такая структура является мощным механизмом представления больших файлов, а также более защищена и устойчива к сбоям
Отложенное выделение пространства	Файловая система ext4 может отложить выделение дискового пространства до последнего момента, что увеличивает производительность системы

Таблица 4.6 *(окончание)*

Особенность	Комментарий
Контрольные суммы журналов	Контрольные суммы журналов повышают надежность файловой системы
Большее количество каталогов	В ext3 могло быть максимум 32 000 каталогов, в ext4 количество каталогов не ограничивается
Дефрагментация "на лету"	Файловая система ext3 не особо склонна к фрагментации, но все же такое неприятное явление имеется. В ext4 производится дефрагментация "на лету", что позволяет повысить производительность системы в целом
Наносекундные временные метки	В большинстве файловых систем временные метки (timestamp) устанавливаются с точностью до секунды, в ext4 точность повышена до наносекунды. Кроме того, ext4 поддерживает временные метки до 25 апреля 2514 года, в отличие от ext3 (18 января 2038 г.)

4.8.2. Совместимость с ext3

Файловая система ext4 является прямо и обратно совместимой с ext3, однако все же существуют некоторые ограничения. Предположим, что у нас на диске имеется файловая система ext4. Ее можно смонтировать и как ext3, и как ext4 (это и есть прямая совместимость) — и тут ограничений никаких нет. А вот с обратной совместимостью не все так безоблачно — если файловую систему ext4 смонтировать как ext3, то она будет работать без экстентов, что снизит ее производительность.

4.8.3. Переход на ext4

Если вы при установке системы выбрали файловую систему ext3, то перейти на ext4 можно без потери данных и в любой удобный для вас момент. Откройте терминал и введите команду:

```
sudo tune2fs -O extents,uninit_bg,dir_index /dev/имя_устройства
```

На момент ввода этой команды устройство должно быть размонтировано.

> ***Внимание!***
>
> Если нужно преобразовать в ext4 корневую файловую систему, то данную команду нужно вводить с LiveCD, поддерживающего ext4.

После этого проверим файловую систему:

```
sudo fsck -pf /dev/имя_устройства
```

Затем смонтируем файловую систему так:

```
mount -t ext4 /dev/имя_устройства /точка_монтирования
mount -t ext4 /dev/disk/by-uuid/UUID-устройства /точка_монтирования
```

Если раздел автоматически монтируется через /etc/fstab, не забудьте исправить файловую систему на ext4:

```
UUID=UUID-раздела  /точка ext4    defaults,errors=remount-ro,relatime
     0  1
```

Если вы изменили тип файловой системы корневого раздела, то необходимо отредактировать файл /boot/grub/menu.lst и добавить опцию `rootfstype=ext4` в список параметров ядра, например:

```
title              Linux
root        (hd0,1)
kernel      /boot/vmlinuz-2.6.28.1 root=UUID=879f797c-944d-4c28-a720-
249730705714 ro quiet splash rootfstype=ext4
initrd             /boot/initrd.img-2.6.28.1
quiet
```

Совет

Рекомендую прочитать статью Тима Джонса "Анатомия ext4": **http://www.ibm.com/developerworks/ru/library/l-anatomy-ext4/index.html**.

4.9. Особые команды

4.9.1. Команда *mkfs*: создание файловой системы

С помощью команды `mkfs` мы можем создать файловую систему на разделе жесткого диска, например: `mkfs.ext2 /dev/hda1`.

Вообще, создать файловую систему нужного типа (если эта файловая система поддерживается ядром вашей системы) можно с помощью команды `mkfs.<имя_файловой_системы>`, например:

```
mkfs.ext3
mkfs.vfat
mkfs.reiserfs
```

Подробнее прочитать об этом можно, введя команду `man mkfs.<имя_файловой_системы>`.

4.9.2. Команда *fsck*: проверка и восстановление файловой системы

Для проверки файловой системы используется команда `fsck`. Использовать ее нужно так:

```
fsck <раздел>
```

Например:

```
fsck /dev/sda5
```

Перед использованием этой команды нужно размонтировать проверяемую файловую систему. Если нужно проверить корневую файловую систему, то нужно загрузиться с LiveCD и запустить `fsck` для проверки нужного раздела.

Если же жесткий диск "посыпался", т. е. появились "плохие" блоки, нужно, не дожидаясь полной потери данных, выполнить следующие действия:

1. Выполнить команду `fsck -c <раздел>` (данная команда пометит "плохие" блоки).
2. Сделать резервную копию всех важных данных.
3. Отправиться в магазин за новым жестким диском и перенести данные со старого жесткого диска на новый. Проверить жесткий диск на наличие плохих секторов можно программой `badblocks`.

Примечание

Программа `fsck` может проверять не только файловые системы ext2/ext3. Для проверки, например, vfat, можно использовать команду `fsck.vfat <раздел>`.

Для восстановления "упавшей" таблицы разделов можно использовать программу `gpart`. Только используйте ее осторожно и внимательно читайте все сообщения, выводимые программой.

4.9.3. Команда *chroot*: смена корневой файловой системы

Предположим, мы установили Windows после установки Linux, и программа установки Windows перезаписала начальный загрузчик. Теперь Windows загружается, а Linux — нет. Что делать? Нужно загрузиться с LiveCD (подробнее о LiveCD мы поговорим в *приложении*) и выполнить команду:

```
# chroot <раздел, содержащий корневую файловую систему>
```

Например, если Linux был установлен в раздел /dev/sda5, то нужно ввести команду:

```
# chroot /dev/sda5
```

Данная команда сменит корневую файловую систему, т. е. вы загрузите ядро Linux с LiveCD, а затем сделаете подмену корневой файловой системы. Вам останется только ввести команду записи загрузчика (например, `lilo`) для восстановления начального загрузчика.

4.9.4. Установка скорости CD/DVD

Программа `hdparm` позволяет ограничить скорость оптического привода (CDROM/DVDROM). Иногда нужно ограничить скорость, чтобы информация была считана без ошибок (как правило, если поверхность носителя информации немного повреждена). Рассмотрим команду ограничения скорости:

```
# hdparm -q -E<множитель> <устройство>
```

Множитель — это и есть скорость, например 1× соответствует скорости 150 KB/s для CD, 1385 kB/s для DVD. Чтобы установить вторую (2×, 300 КБ/с) скорость чтения для CD, используется команда:

```
# hdparm -q -E2 /dev/cdrom
```

Для ограничения скорости DVD можно использовать следующую команду:

```
# hdparm -q -E1 /dev/dvd
```

4.9.5. Монтирование каталога к каталогу

В Linux можно подмонтировать каталог к каталогу, а не только каталог к устройству. Делается это с помощью все той же команды `mount`, запущенной с параметром `--bind`:

```
# mount --bind исходный_каталог каталог_назначения
```

4.9.6. Команды поиска файлов

Для поиска файлов в Linux используется команда `find`. Это довольно мощная утилита со сложным синтаксисом и далеко не всегда нужная обычному пользователю. Обычному пользователю намного проще будет установить файловый менеджер mc и использовать встроенную функцию поиска.

Но команду `find` мы все же рассмотрим, по крайней мере, ее основы. Синтаксис команды следующий:

```
find список_поиска выражение
```

Мощность программы `find` заключается в множестве самых разных параметров поиска, которые не так легко запомнить — их просто много. К тому же `find` может выполнять команды для найденных файлов. Например, вы можете найти временные файлы и сразу удалить их.

Подробно опции команды `find` мы рассматривать не будем — это вы можете сделать самостоятельно с помощью команды `man find`. Зато мы рассмотрим несколько примеров использования этой команды.

Найти файлы с именем a.out (точнее, в имени которых содержится строка "a.out"), поиск начать с корневого каталога (/):

```
find / -name a.out
```

Найти файлы по маске *.txt:

```
find / -name '*.txt'
```

Найти файлы нулевого размера, поиск начать с текущего каталога (.):

```
find . -size 0c
```

Хотя для поиска пустых файлов намного проще использовать параметр `-empty`:

```
find . -empty
```

Найти файлы, размер которых от 100 до 150 Мбайт, поиск производить в домашнем каталоге и всех его подкаталогах:

```
find ~ -size +100M -size -150M
```

Найти все временные файлы и удалить их (для каждого найденного файла будет запущена команда `rm`):

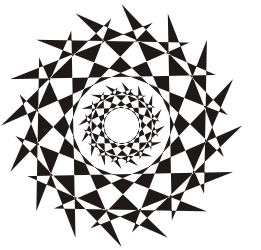
```
# find / -name *.tmp -ok rm {} \;
```

Вместо параметра `-ok` можно использовать параметр `-exec`, который также запускает указанную после него команду, но не запрашивает подтверждение выполнения этой команды для каждого файла.

Кроме команды `find` можно использовать команды `which` и `locate`. Первая выводит полный путь к программе или к сценарию, если программа или сценарий находится в списке каталогов, заданном в переменной окружения `PATH`:

```
which sendmail
```

Программа `locate` ищет в базе данных демона located файлы, соответствующие заданному образцу. Недостаток этой команды в том, что located имеется далеко не во всех дистрибутивах, поэтому команды `locate` у вас может и не быть. Зато если located имеется и запущен, поиск файлов будет осуществляться быстрее, чем с помощью `find`.



Глава 5

Процессы

5.1. Команды *kill*, *killall*, *xkill* и *ps*

Каждому процессу в Linux присваивается уникальный номер — идентификатор процесса (PID, Process ID). Зная ID процесса, вы можете управлять процессом, а именно — завершить процесс или изменить приоритет процесса. Принудительное завершение процесса необходимо, если процесс завис, и его нельзя завершить обычным образом. А изменение приоритета может понадобиться, если вы хотите, чтобы процесс доделал свою работу быстрее.

Предположим, у вас зависла какая-то программа, например файловый менеджер mc. Хоть это и маловероятно (не помню, чтобы он когда-нибудь зависал), но для примера пусть будет так. Принудительно завершить ("убить") процесс можно с помощью команды `kill`. Формат ее вызова следующий:

```
kill [параметры] PID
```

PID (Process ID) — это идентификатор процесса, который присваивается процессу системой; уникален для каждого процесса. Но мы знаем только имя процесса (имя команды), но не знаем идентификатор процесса. Узнать идентификатор процесса позволяет программа `ps`. Предположим, что mc находится на первой консоли. Поскольку он завис, вы не можете более использовать консоль, и вам нужно переключиться на вторую консоль (клавиатурной комбинацией <Alt>+<F2>). Зарегистрировавшись на второй консоли, введите команду `ps`. Она выведет список процессов, запущенных на второй консоли — это будет `bash` и сам `ps` (рис. 5.1).

Чтобы добраться до нужного нам процесса (mc), который запущен на первой консоли, введите команду `ps -a` или `ps -U root`. В первом случае вы получите список процессов, запущенных вами, а во втором — список процессов, запущенных от вашего имени (я предполагаю, что вы работаете под именем root). Обратите внимание: вы сами запустили процессы `mc` и `ps` (рис. 5.2), а от вашего имени (root) система запустила множество процессов. Следует заметить, что программа `ps` выводит также имя терминала (tty1), на котором запущен процесс. Это очень важно — если на разных консолях у вас запущены одинаковые процессы, можно легко ошибиться и завершить не тот процесс.

Теперь, когда мы знаем PID нашего процесса, мы можем его "убить":

```
# kill 2484
```

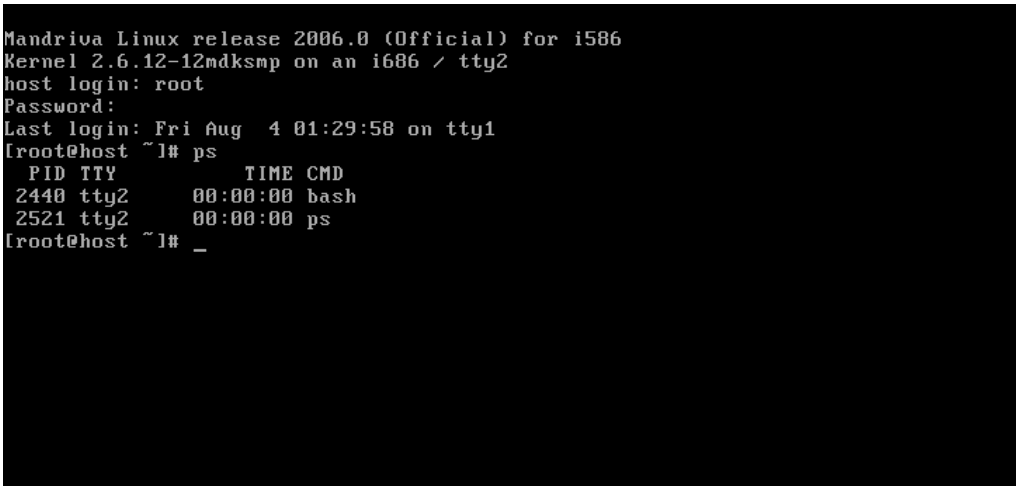


Рис. 5.1. Список процессов на текущей консоли

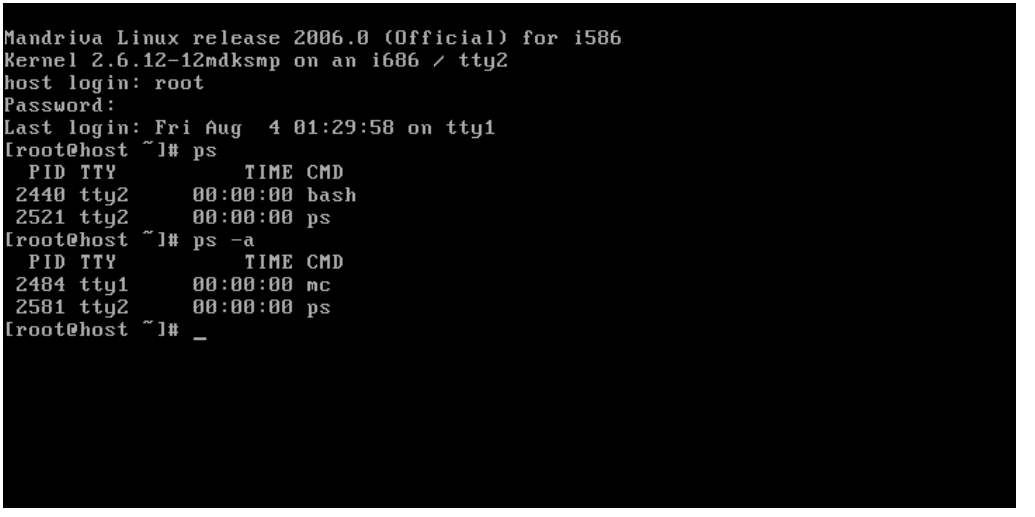


Рис. 5.2. Определение PID программы `mc`

Перейдите на первую консоль после выполнения этой команды — `mc` на ней уже не будет. Если выполнить команду `ps -a`, то в списке процессов `mc` тоже не будет.

Проще всего вычислить PID процесса с помощью следующей команды:

```
# ps -ax | grep <имя>
```

Например, `# ps -ax | grep firefox`.

Вообще-то все эти действия, связанные с вычислением PID процесса, мы рассмотрели только для того, чтобы познакомиться с командой `ps`. Так что, если вы знаете только имя процесса, гораздо удобнее использовать команду:

```
# killall <имя процесса>
```

Но имейте в виду, что данная команда завершит все экземпляры данного процесса. А вполне может быть, что у нас на одной консоли находится `mc`, который нужно "убить", а на другой — нормально работающий `mc`. Команда `killall` "убьет" оба процесса.

При выполнении команд `kill` и `killall` нужно помнить, что если вы работаете от имени обычного пользователя, они могут завершить только те процессы, которые принадлежат вам. А если вы работаете от имени пользователя root, то можете завершить любой процесс в системе.

Кроме команды `kill` пользователи, предпочитающие графический интерфейс, могут использовать программу `xkill`, позволяющую "убить" графическую программу. Введите команду `xkill`, указатель мыши изменится на череп, которым нужно будет щелкнуть по зависшему окну (иначе, если окно не зависло, зачем его "убивать"?). Процесс, относящийся к выбранному окну, будет немедленно завершен.

5.2. Программа *top*: кто больше всех расходует процессорное время

Иногда бывает, что система ужасно тормозит — весь день работала нормально, а вдруг начала притормаживать. Если вы даже не догадываетесь, из-за чего это случилось, вам нужно использовать программу `top` (рис. 5.3) — она выводит список процессов с сортировкой по процессорному времени. То есть на вершине списка будет процесс, который занимает больше процессорного времени, чем сама система. Вероятно, из-за него и происходит эффект "торможения".

```
top - 01:39:31 up 10 min,  3 users,  load average: 0.00, 0.00, 0.00
Tasks:  58 total,   1 running,  57 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0% us,  0.3% sy,  0.0% ni, 99.7% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:    189720k total,    68224k used,   121496k free,     5088k buffers
Swap:   128984k total,        0k used,   128984k free,    38072k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2599 root      16   0  1996 1012  804 R  0.3  0.5   0:00.06 top
    1 root      16   0  1564  540  472 S  0.0  0.3   0:00.55 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      10  -5     0    0    0 S  0.0  0.0   0:00.02 events/0
    5 root      16  -5     0    0    0 S  0.0  0.0   0:00.08 khelper
    6 root      11  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
    8 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   61 root      10  -5     0    0    0 S  0.0  0.0   0:00.03 kblockd/0
   93 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
   94 root      15   0     0    0    0 S  0.0  0.0   0:00.05 pdflush
   96 root      16  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
   95 root      25   0     0    0    0 S  0.0  0.0   0:00.00 kswapd0
  684 root      16   0     0    0    0 S  0.0  0.0   0:00.00 kseriod
  766 root      13  -5     0    0    0 S  0.0  0.0   0:00.00 ata/0
  775 root      18   0     0    0    0 S  0.0  0.0   0:00.00 scsi_eh_0
  784 root      16   0     0    0    0 S  0.0  0.0   0:00.02 kjournald
  924 root      15  -4  1564  496  420 S  0.0  0.3   0:00.08 udevd
```

Рис. 5.3. Программа `top`

На рис. 5.3 показано, что больше всего процессорного времени (0.3%) занимает программа `top`. Конечно, в реальных условиях все будет иначе. Выйти из про-

граммы `top` можно, нажав клавишу <Q>. Кроме <Q> действуют следующие клавиши:

- <U> — показывает только пользовательские процессы (т. е. те процессы, которые запустил пользователь, под именем которого вы работаете в системе);
- <D> — изменяет интервал обновления;
- <F> — изменяет столбец, по которому сортируются задачи. По умолчанию задачи сортируются по столбцу %CPU, т. е. по процессорному времени, занимаемому процессом;
- <H> — получить справку по остальным командам программы `top`.

Назначение столбцов программы `top` указано в табл. 5.1.

Таблица 5.1. *Назначение столбцов программы* `top`

Столбец	Описание
PID	Идентификатор процесса
USER	Имя пользователя, запустившего процесс
PR	Приоритет процесса
NI	Значение `nice` *(см. разд. 5.3)*
VIRT	Виртуальная память, использованная процессом (в Кбайт)
RES	Размер процесса, не перемещенный в область подкачки (в Кбайт). Этот размер равен размерам сегментов кода и данных, т. е. RES = CODE + DATA
S	Состояние процесса: • R — выполняется; • S — "спит" (режим ожидания), в этом состоянии процесс выгружен из оперативной памяти в область подкачки; • D — "непрерываемый сон" (uninterruptible sleep), из такого состояния процесс может вывести только прямым сигналом от оборудования; • T — процесс в состоянии трассировки или остановлен; • Z ("зомби") — специальное состояние процесса, когда сам процесс уже завершен, но его структура еще осталась в памяти
%CPU	Занимаемое процессом процессорное время
%MEM	Использование памяти процессом
TIME+	Процессорное время, израсходованное с момента запуска процесса
COMMAND	Команда, которая использовалась для запуска процесса (обычно имя исполнимого файла процесса)

5.3. Команды *nice* и *renice*: изменение приоритета процесса

Предположим, что вы работаете с видео и вам нужно перекодировать файл из одного видеоформата в другой. Конвертирование видео занимает много процессорного времени, а хотелось бы все сделать как можно быстрее и уйти раньше домой. Тогда вам поможет программа `nice` — она позволяет запустить любую программу с указанным приоритетом. Ясно — чем выше приоритет, тем быстрее будет выполняться программа. Формат вызова команды следующий:

```
nice -n <приоритет> команда аргументы
```

Максимальный приоритет задается числом `-20`, а минимальный числом `19`. Приоритет по умолчанию равен `10`.

Если процесс уже запущен, тогда для изменения его приоритета можно использовать команду `renice`:

```
renice -n <приоритет> -p PID
```

5.4. Перенаправление ввода/вывода

С помощью перенаправления ввода/вывода мы можем перенаправить вывод одной программы в файл или на стандартный ввод другой программы. Например, у вас не получается настроить сеть, и вы хотите перенаправить вывод команды `ifconfig` в файл, а затем разместить этот файл на форуме, где вам помогут разобраться с этой проблемой. А можно перенаправить список всех процессов (командой `ps -ax`) команде `grep`, которая найдет в списке интересующий вас процесс.

Рассмотрим следующую команду:

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

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

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

```
ifconfig > ifconfig.txt
```

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

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

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

```
cat big_text
```

Но в файле big_text много строк, поэтому мы ничего не успеем прочитать. Следовательно, целесообразно отправить вывод команды `cat` какой-нибудь программе, которая будет выводить файл постранично, например:

```
cat big_text | more
```

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

```
less big_text
```

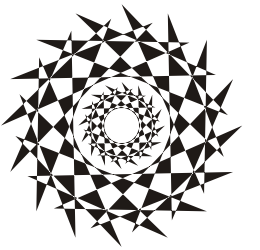
Вот еще один интересный пример. Допустим, мы хотим удалить файл file.txt без запроса — для этого можно указать команду:

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

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

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

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



Глава 6

Запись CD/DVD из консоли

6.1. Команда *dd*: создание образа диска

Довольно часто бывает нужно создать образ оптического диска (не знаю, как у вас, но у меня такая потребность возникает примерно один раз в неделю). Причина проста — или под рукой нет чистой болванки, или же нужно поработать с диском, который придется отдать, но при этом нет никакого желания записывать его на болванку.

В Windows для создания образа диска обычно используются посторонние программы, например Nero или WinImage. В Linux мы будем пользоваться только средствами операционной системы.

Образ CD/DVD-диска можно создать с помощью команды `dd`:

```
dd if=/dev/cdrom of=~/cd.iso
```

Вместо `/dev/cdrom` надо подставить имя файла устройства вашего привода CD/DVD (обычно этого делать не приходится, поскольку ссылка `/dev/cdrom` устанавливается самой системой на ваш привод CD/DVD).

Указанная команда создаст файл образа cd.iso, который будет записан в ваш домашний каталог. Аналогично с помощью этой команды можно создать и образ дискеты — только вместо `/dev/cdrom` нужно указать имя файла устройства `/dev/fd0`.

Что можно сделать с ISO-образом в Windows? Его можно записать на чистую болванку или же открыть в специальной программе (например, ISOpen или UltraISO) для изменения. В Linux открыть образ можно с помощью средств самой операционной системы. Для этого его надо просто подмонтировать к корневой файловой системе с помощью команды следующего формата:

```
# mount -o loop -t iso9660 образ точка_монтирования
```

- опция `-o loop` означает, что будет монтироваться не файл устройства, а образ диска, который записан на жесткий диск;
- параметр `-t 9660` задает тип файловой системы образа: iso9660 — стандартная файловая система для CD/DVD;
- после файловой системы указывается файл образа, например `~/cd.iso`;
- последний параметр — это точка монтирования, каталог, к которому будет подмонтирован образ (напомню, что каталог должен существовать).

Примечание

В большинстве случаев команду `mount` нужно выполнять от имени пользователя root или с помощью команд `sudo` или `su`.

В нашем случае для монтирования образа ~/cd.iso к каталогу /mnt/image нужно выполнить команду:

```
# mount -o loop -t iso9660 ~/cd.iso /mnt/image
```

После этого можно обращаться к образу как к обычному каталогу:

```
ls /mnt/image
```

6.2. Команды *cdrecord* и *dvdrecord*: запись образа на болванку

Предположим, у вас есть файл образа cd.iso, и нужно записать его на компакт-диск, но вы не хотите (или не имеете возможности) использовать графические программы вроде Nero или k3b. В этом случае вам нужно использовать программу cdrecord (пакет называется аналогично). Команда для записи образа на болванку CD-R очень проста и выглядит так:

```
# cdrecord dev=0,0,0 -dao speed=16 файл_образа
```

Для записи DVD-R используется аналогичная команда:

```
# dvdrecord dev=0,0,0 -dao speed=4 файл_образа
```

В этой команде вам нужно изменить параметр `dev` — идентификатор устройства CD/DVD. Если в вашей системе установлен только один привод CD/DVD, и он же является пишущим, тогда, скорее всего, у него будет идентификатор 0,0,0. Но если у вас несколько приводов CD/DVD (например, обычный и пишущий), вы должны ввести следующую команду:

```
# cdrecord -scanbus
```

Команда выведет список приводов CD/DVD, установленных в вашей системе (рис. 6.1). Вам нужно запомнить идентификатор нужного привода и использовать его при записи образа диска.

```
[den@localhost ~]$ cdrecord -scanbus
scsibus1000:
        1000,0,0 100000) *
        1000,1,0 100001) 'HL-DT-ST' 'DVDRAM GSA-4167B' 'DL11' Removable CD-ROM
        1000,2,0 100002) *
        1000,3,0 100003) *
        1000,4,0 100004) *
        1000,5,0 100005) *
        1000,6,0 100006) *
        1000,7,0 100007) *
[den@localhost ~]$
```

Рис. 6.1. Идентификаторы приводов CD/DVD

6.3. Команды очистки перезаписываемых дисков

Для очистки диска DVD-RW используется команда:

```
# dvd+rw-format -f имя_устройства_DVD-RW
```

Для быстрой очистки CD-RW введите команду:

```
# cdrecord -v blank=fast dev=0,0,0
```

Если нужно произвести полную, а не быструю очистку, замените `blank=fast` на `blank=all`.

6.4. Команда *mkisofs*: создание ISO-образа

Иногда нужно создать образ CD/DVD не с оригинального диска, а с каталогов файловой системы. Другими словами — у вас есть файлы и каталоги, которые вам нужно записать на CD/DVD. Технология CD/DVD не позволяет записывать файлы и каталоги непосредственно на носитель — вам нужно создать каталог, поместить в него все файлы и каталоги, которые вы хотите записать на оптический диск, затем создать по этому каталогу ISO-образ, а потом записать его на болванку.

Скопируйте все необходимые вам файлы в каталог ~/cd. Затем выполните команду:

```
mkisofs -r -jcharset koi8-r -o ~/cd.iso ~/cd
```

Эта команда создаст по каталогу ~/cd файл образа cd.iso и поместит его в ваш домашний каталог. Обратите внимание на кодировку локализованной версии — сейчас используется KOI8-R. Если у вас другая кодировка, например UTF-8, вы должны указать ее:

```
mkisofs -r -jcharset utf8 -o ~/cd.iso ~/cd
```

Указание кодировки необходимо для правильного отображения русскоязычных имен файлов и каталогов под управлением MS Windows.

После создания ISO-образа его нужно записать на носитель с помощью команды `cdrecord`, как было показано ранее. После записи не забудьте удалить образ, чтобы он не занимал места на диске.

Существует способ записи каталога на CD/DVD без создания промежуточного ISO-образа. Для этого служит команда:

```
mkisofs -jcharset кодировка /каталог | cdrecord -опции
```

6.5. Преобразование образов дисков

Иногда нужно записать созданный в другой программе образ диска, формат которого отличается от ISO9660. Чаще всего встречаются образы дисков в форматах IMG, BIN, CUE, NRG, CCD.

Если у файла образа "расширение" (в Linux нет понятия "расширение", поэтому данное слово взято в кавычки) img, то это еще не означает, что формат образа — ISO9660. Одни программы, например `K3b`, действительно создают образ в формате

ISO9660 и записывают его в файл с расширением img, а другие — могут записывать в файл с таким же расширением образы диска в собственных форматах.

Файлы .bin/.cue можно записать на диск с помощью программы `cdrdao` или преобразовать в ISO с помощью программы `bchunk`.

Nero записывает образы диска в формате NRG, который можно преобразовать в ISO с помощью программы `nrg2iso`. Если вам нужно открыть NRG-образ, чтобы просмотреть его содержимое, вы это можете сделать с помощью команды:

```
mount -t udf,iso9660 -o loop,ro,offset=307200 файл.nrg точка_монтирования
```

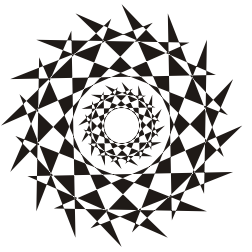
Образ в формате CloneCD (ccd) можно преобразовать в ISO с помощью программы `ccd2iso`.

6.6. Создание и монтирование файлов с файловой системой

Иногда (например, для создания мини-дистрибутива) нужно создать файл, содержащий собственную файловую систему. Первым делом нужно создать пустой файл, потом создать в нем файловую систему, а затем подмонтировать этот файл к корневой файловой системе. Все это можно сделать с помощью трех команд:

```
# dd if=/dev/zero of=/file.fs bs=1k count=100000
# mkfs.ext2 -F /file.fs
# mount -t ext2 -o loop file.fs /mnt/disk
```

Первая команда создает пустой файл размером почти 100 Мбайт (100 000 Кбайт), вторая команда создает в этом файле файловую систему типа ext2, третья — монтирует файл к каталогу /mnt/disk.



Глава 7

Команды для работы с текстом

7.1. Команда *cmp*: сравнение двух файлов

Команда `cmp` используется для сравнения двух файлов. Если файлы идентичны, то `cmp` вообще ничего не выводит. А вот если файлы отличаются, то `cmp` выводит номер строки и номер символа в строке, откуда начинается различие.

Команда `cmp` более универсальна, поскольку она может использоваться как для сравнения текстовых, так и двоичных файлов. А вот команда `diff` и ее аналоги умеют сравнивать только текстовые файлы.

Формат вызова команды следующий:

```
cmp [параметры] файл1 файл2
```

Параметры команды `cmp` указаны в табл. 7.1.

Таблица 7.1. *Параметры команды* `cmp`

Параметр	Описание
`-c`	Вывод отличающихся символов
`-i n`	Игнорировать первые *n* символов
`-l`	Вывод позиций всех отличий, а не только первого
`-s`	Не выводить информацию на экран, при этом код возврата будет следующим: 0 — файлы одинаковые; 1 — файлы отличаются; 2 — ошибка при открытии одного из файлов

7.2. Команда *column*: разбивка текста на столбцы

Команда `column` используется для разбивки текста на несколько столбцов. Текст может быть прочитан как из файла, так и со стандартного ввода, если файл не указан.

Формат вызова команды:

```
column [параметры] [файл]
```

Параметры команды `column` приведены в табл. 7.2.

Таблица 7.2. *Параметры команды* `column`

Параметр	Описание
`-c n`	Задает количество столбцов (число *n*)
`-s символ`	Указанный символ будет использоваться в качестве разделителя столбцов
`-t`	Текст будет форматироваться как таблицы. По умолчанию разделителем полей считается пробел, но с помощью параметра `-s` можно задать другой разделитель
`-x`	Сначала будут заполняться столбцы, а потом строки

7.3. Команда *comm*: еще одна команда для сравнения файлов

Команда `comm` сравнивает содержимое двух файлов, которые были перед этим отсортированы командой `sort`. Вывод программы располагается в три столбца. В первом выводятся строки из `файла1`, во втором — из `файла2`, а в третьем — строки, которые имеются в обоих файлах.

Формат вызова программы следующий:

```
comm [параметры] файл1 файл2
```

Параметры команды `comm` приведены в табл. 7.3.

Таблица 7.3. *Параметры команды* `comm`

Параметр	Описание
-1	Не выводить первый столбец
-2	Не выводить второй столбец
-3	Не выводить третий столбец
-12	Будет выведен только третий столбец
-13	Будет выведен только второй столбец
-23	Вывод только первого столбца

7.4. Команда *diff*: сравнение файлов

Команда используется для сравнения двух файлов. Формат вызова программы `diff`:

```
diff параметры файл1 файл2
```

В выводе программы отличающиеся строки помечаются символами > и <:

- строка из первого файла помечается символом <;
- строка из второго файла — символом >.

Самые полезные параметры программы `diff` приведены в табл. 7.4.

Таблица 7.4. *Некоторые параметры программы* `diff`

Параметр	Описание
`-a`	Сравнение всех файлов, в том числе бинарных
`-b`	Программа будет игнорировать пробельные символы в конце строки
`-B`	Игнорирует пустые строки
`-e`	Применяется для создания сценария для редактора ed, который будет использоваться для превращения первого файла во второй
`-w`	Игнорирует пробельные символы
`-y`	Вывод в два столбца
`-r`	Используется для сравнения файлов в подкаталогах. Вместо первого файла указывается первый каталог, вместо второго файла — соответственно второй каталог

7.5. Команда *diff3*: сравнение трех файлов

Похожа на команду `diff`, только используется для сравнения трех файлов. Формат вызова команды следующий:

```
diff3 [параметры] файл1 файл2 файл3
```

Программа выводит следующую информацию:

- **`====`** — все три файла разные;
- **`===1`** — первый файл отличается от второго и третьего;
- **`===2`** — второй файл отличается от первого и третьего;
- **`===3`** — третий файл отличается от первого и второго.

Параметры программы `diff3` представлены в табл. 7.5.

Таблица 7.5. *Параметры программы* `diff3`

Параметр	Описание
`-a`	Сравнивать файлы как текстовые, даже если они являются бинарными
`-A`	Создание сценария для редактора ed, который показывает в квадратных скобках все отличия между файлами
`-e`	Создает сценарий для `ed`, который помещает все отличия между файлами `файл2` и `файл3` в файл `файл1` (будьте осторожны!)
`-i`	Добавить команды `w` (сохранить файл) и `q` (выйти) в конец сценария ed

***Таблица 7.5** (окончание)*

Параметр	Описание
`-x`	Создание сценария редактора `ed`, который помещает отличия между файлами в файл `файл1`
`-X`	То же, что и `-x`, но отличия выделяются
`-3`	Создает сценарий `ed`, который помещает все различия между файлами `файл1` и `файл3` в `файл1`

7.6. Команда *egrep*: расширенный текстовый фильтр

Производит поиск строки в одном или нескольких файлах. Команда `egrep` похожа на команду `grep` (которая будет описана позже), но считается более быстрой и более функциональной. Если файлы не заданы, то программа читает текст из стандартного ввода.

Формат вызова программы:

```
egrep [параметры] строка файлы
```

Параметры команды `egrep` приведены в табл. 7.6.

***Таблица 7.6.** Параметры программы `egrep`*

Параметр	Описание
`-A n`	Вывод *n* строк после строки, в которой есть искомая строка
`-B n`	Вывод *n* строк перед строкой, содержащей искомую строку
`-b`	Выводит для каждой строки файла, где есть искомая строка, ее положение в файле
`-c`	Выводит количество совпадений, но не выводит сами совпадения
`-C`	Выводит две строки до и две строки после строки, которая содержит искомую строку
`-e строка`	Используйте данный параметр, если искомая строка начинается с символа "-"
`-f файл`	Производит поиск искомых строк, которые имеются в указанном файле
`-h`	Выводит строки, содержащие искомую строку, но не выводит имена содержащих их файлов
`-i`	Игнорировать регистр букв
`-n`	Выводит номера строк (и сами строки), содержащих искомую строку
`-s`	Не выводить сообщения об ошибке, если некоторые файлы не могут быть открыты
`-w`	Поиск совпадения целого слова с искомой строкой
`-x`	Поиск совпадения целой строки с искомой строкой

Пример использования:

```
egrep "ppp [11]" *
```

Данная команда ищет строку, заключенную в кавычки во всех файлах в текущем каталоге.

7.7. Команда *expand*: замена символов табуляции пробелами

Команда `expand` заменяет в указанных файлах символы табуляции на соответствующее количество пробелов. Команде можно передать только один параметр `-i`, означающий, что замена должна быть только в начале строки.

Формат вызова команды:

```
expand [-i] файлы
```

7.8. Команда *fmt*

Команда `fmt` форматирует текст, выравнивает его по правой границе и удаляет символы новой строки. Синтаксис вызова команды:

```
fmt [параметры] файлы
```

Параметры команды `fmt` приведены в табл. 7.7.

Таблица 7.7. *Параметры команды* `fmt`

Параметр	Описание
`-c`	Не форматировать первые две строки
`-p префикс`	Форматировать только строки, начинающиеся с указанного префикса
`-s`	Не объединять строки
`-t`	Начинать параграф с красной строки
`-w n`	Задает максимальную длину строку в *n* символов (по умолчанию 72)

7.9. Команда *fold*

Форматирует текст по правому краю. Производит разрыв строк, если это необходимо. Максимальная длина строки — 80 символов. Установить другую длину строки можно с помощью параметра `-w`, как и в случае с командой `fmt`. Кроме того, можно указать параметр `-s`, разрешающий разрыв строки только на пробеле.

Формат вызова команды:

```
fold [параметры] файлы
```

7.10. Команда *grep*: текстовый фильтр

Команда `grep` похожа на команду `egrep`, которая была описана ранее. Сейчас вместо описания параметров `grep` мы рассмотрим практическое использование этой команды.

Предположим, что у нас есть файл протокола /var/log/messages, и мы хотим вывести все сообщения, связанные с демоном `pppd`. Понятно, что вручную выделить все нужные сообщения будет довольно трудно. Но с помощью `grep` можно автоматизировать данную задачу:

```
cat /var/log/messages | grep ppp
```

Команда `cat /var/log/messages` передаст содержимое файла /var/log/messages на стандартный ввод команды `grep`, которая, в свою очередь, выделит строки, содержащие строку `ppp`.

> ***Совет***
>
> Вообще-то, просматривать журналы удобнее с помощью команды `tac`, которая выводит строки файла в обратном порядке — ведь сообщения дописываются в конец журнала, следовательно, если выводить строки в обратном порядке, то сначала получим самые новые сообщения, а потом уже все остальные:
>
> ```
> tac /var/log/messages | grep ppp
> ```

7.11. Команды *more* и *less*: постраничный вывод

Большой текстовый файл намного удобнее просматривать с помощью команд `less` или `more`. Программа `less` удобнее, чем `more`, если она есть в вашей системе:

```
tac /var/log/messages | grep ppp | less
```

7.12. Команды *head* и *tail*: вывод начала и хвоста файла

Команда `head` выводит первые десять строк файла, а `tail` — последние десять. Количество строк может регулироваться с помощью параметра `-n`.

Пример использования:

```
head -n 10 /var/log/messages
tail -n 15 /var/log/messages
```

7.13. Команда *look*

Производит поиск строк, начинающихся с указанной строки. Если файл не указан, то поиск производится в файле /usr/share/dict/words. Можно задать параметр `-a` — тогда поиск будет произведен в файле /usr/share/dict/web2.

7.14. Команда *sort*: сортировка файлов

Сортирует все указанные файлы, результат сортировки всех указанных файлов отправляется на стандартный вывод.

Синтаксис вызова команды:

```
sort [параметр]... [файлы]
```

Параметры команды `sort` приведены в табл. 7.8.

Таблица 7.8. *Параметры программы* `sort`

Параметр	Описание
`-b`	Пробелы в начале сортируемых полей или начале ключей будут игнорироваться
`-d`	При сортировке будут игнорироваться все символы, кроме букв, цифр и пробельных символов
`-f`	Игнорировать регистр букв
`-r`	Сортировка в обратном порядке
`-o файл`	Вывод результатов сортировки в файл
`-t символ`	Использование указанного символа в качестве разделителя полей

7.15. Команда *split*: разбиение файлов на несколько частей

Используется для разделения файлов на части. По умолчанию создаются части размером в 1000 строк. Изменить размер можно, указав количество строк, например:

```
split -500 файл1
```

В данном случае файл будет разбит на части по 500 строк в каждой (кроме, возможно, последней части, где может быть меньше строк).

Команду можно также использовать для разделения файлов на части по размеру информации, а не по количеству строк, например с помощью параметра `-b` можно указать количество символов в каждой части. Примеры вызова команды:

```
split -b100b файл
split -b100k файл
split -b100m файл
```

Первая команда разделит файл на части по 100 байтов каждая, вторая — на части по 100 Кбайт каждая, третья — по 100 Мбайт каждая.

7.16. Команда *unexpand*: замена пробелов на символы табуляции

Заменяет последовательные пробелы на символы табуляции. По умолчанию заменяются 8 пробелов на один символ табуляции. Количество пробелов можно задать с помощью параметра `-t n` (где `n` — количество пробелов).

Синтаксис вызова:

```
unexpand [параметры] файл
```

7.17. Команды *vi*, *nano*, *ee*, *mcedit*, *pico*: текстовые редакторы

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

Некоторые гурманы (я бы их назвал мазохистами) говорят, что к нему нужно привыкнуть. Может, и так, но сначала нужно изучить длинный man и выучить наизусть команды редактора. Как такового интерфейса пользователя практически нет, можно сказать, что вообще нет — то, что есть, сложно назвать интерфейсом. Однако в этой книге мы рассмотрим `vi`, хотя бы вкратце. Тому есть две причины. Первая — это критики. Мол, как это в книге, посвященной командной строке, не будет "классики". Вторая — некоторые системы, где по непонятным мне причинам до сих пор используется по умолчанию `vi`, а другие редакторы недоступны. Да, можно изменить переменную окружения `EDITOR`, но нет никакой гарантии, что в системе будет установлен какой-нибудь другой редактор.

Итак, приступим к рассмотрению редактора `vi`. Редактор `vi` может работать в трех режимах:

- основной (визуальный) режим — в нем и осуществляется редактирование текста;
- командный режим — в нем осуществляется ввод специальных команд для работы с текстом (если сравнивать `vi` с нормальным редактором, то этот режим ассоциируется с меню редактора, где есть команды вроде "сохранить", "выйти" и т. д.);
- режим просмотра — используется только для просмотра файла (если надумаете использовать этот режим, вспомните про команду `less`).

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

```
vi файл
vi -e файл
vi -R файл
```

Первая команда запускает `vi` и загружает файл. Вторая команда запускает `vi` в командном режиме и загружает файл. Третья команда — это режим просмотра

файла. Если указанный файл не существует, то он будет создан. По умолчанию активируется именно командный режим, поэтому в ключе `-e` нет смысла.

После запуска `vi` главное знать, как из него выйти. Ведь в нем не будет привычной строчки меню, также редактор не будет реагировать на привычные комбинации клавиш вроде <Alt>+<X>, <Ctrl>+<C>. На рис. 7.1 представлен редактор `vi`, в который загружен файл /etc/passwd.

В табл. 7.9 приведены основные команды редактора `vi`.

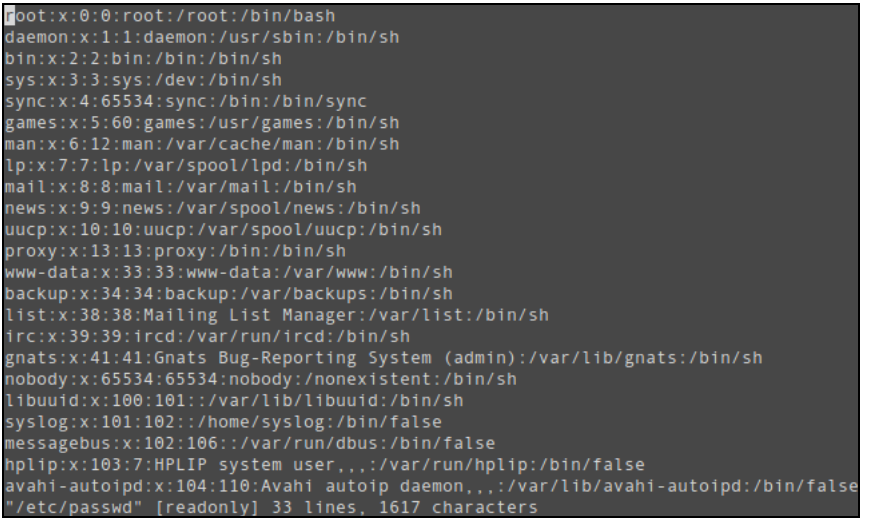


Рис. 7.1. Редактор `vi`

Таблица 7.9. *Основные команды редактора* `vi`

Команда	Описание
`:q!`	Выход без сохранения
`:w`	Сохранить изменения
`:w <файл>`	Сохранить изменения под именем `<файл>`
`:wq`	Сохранить и выйти
`:q`	Выйти, если нет изменений
`i`	Перейти в режим вставки символов в позицию курсора
`a`	Перейти в режим вставки символов в позицию после курсора
`o`	Вставить строку после текущей
`O`	Вставить строку над текущей
`x`	Удалить символ в позицию курсора
`dd`	Удалить текущую строку
`u`	Отменить последнее действие

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

Теперь немного практики, введите команду:

```
$ vi file.txt
```

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

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

```
cat file.txt
```

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

Теперь приступим к дальнейшему рассмотрению редактора. Если ввести не команду `i`, а команду `a`, то вы тоже перейдете в режим вставки, но с одним отличием. Введенный текст будет вставляться не перед символом, в котором находится курсор, а после него. Также в режим вставки можно перейти командами `o` и `O`. В первом случае будет добавлена пустая строка после текущей строки, а во втором — перед текущей строкой, а весь дальнейший ввод будет восприниматься именно как ввод текста, а не команд.

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

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

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

Для отмены последней операции используется команда `u`. Вот только истории изменений нет, да и по команде `u` отменяется вся предыдущая команда целиком. Например, вы создали файл, перешли в режим вставки (команда `i`) и ввели весь

текст Большой медицинской энциклопедии. Если вы введете команду `u`, то она отменит всю предыдущую команду, т. е. удалит весь введенный вами текст. Так что будьте осторожны.

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

```
man vi
```

А мы тем временем познакомимся с другими текстовыми редакторами. Самый удобный из известных мне текстовых редакторов — редактор `nano` (раньше он назывался `pico` и входил в состав почтового клиента `pine`). Редактор `nano` изображен на рис. 7.2.

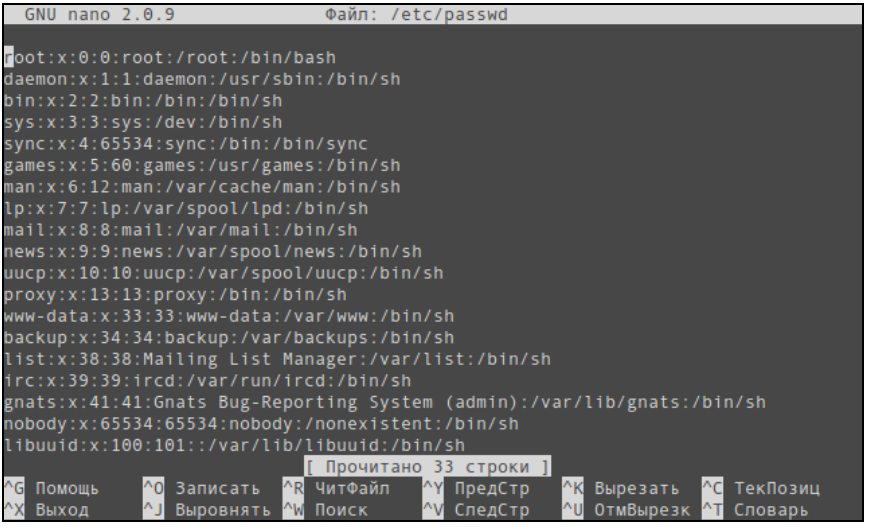


Рис. 7.2. Редактор `nano`

Внизу (под текстом) есть подсказка по комбинациям клавиш для управления редактором. Символ ^ означает <Ctrl>. То есть для выхода из редактора нужно нажать <Ctrl>+<X>, а для сохранения текста — <Ctrl>+<O>.

В некоторых системах (например, в FreeBSD) вместо `nano` используется редактор `ee`. Он похож на `nano`, но подсказки выводятся до текста (вверху экрана), а не после него, но идея та же. Также довольно удобный редактор `joe`.

В пакет `mc` (файловый менеджер) входит довольно удобный редактор `mcedit`, который запускается при нажатии клавиши <F4> в `mc` (рис. 7.3). Но вы можете запустить редактор отдельно:

```
mcedit <имя файла>
```

Кстати, редакторы `joe`, `nano` и `ee` запускаются аналогично:

```
joe <имя файла>
nano <имя файла>
ee <имя файла>
```

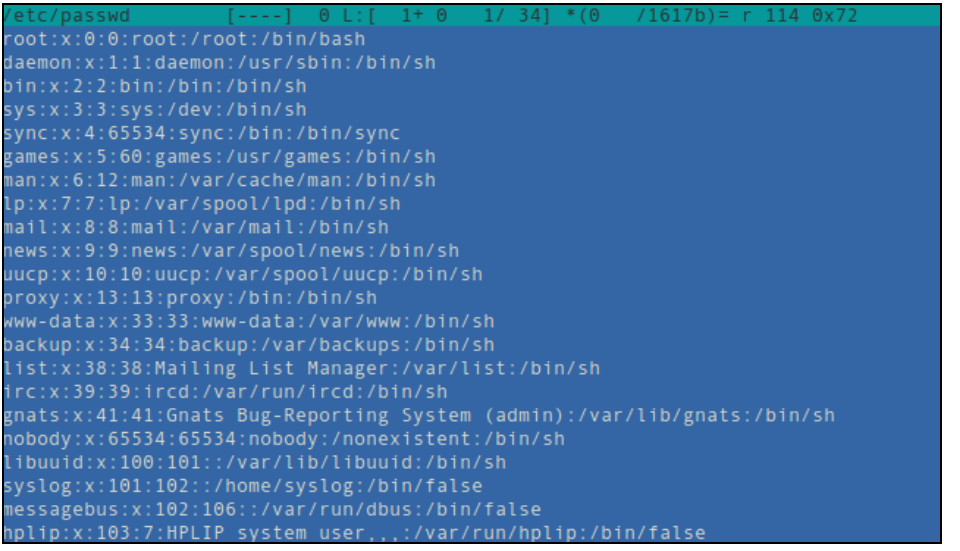


Рис. 7.3. Редактор `mcedit`

7.18. Команда *wc*: подсчет слов в файле

Команда `wc` используется:

- для подсчета слов в текстовом файле:

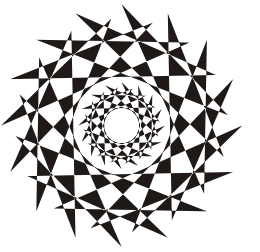
  ```
  wc /var/log/messages
  ```

- для подсчета количества строк (если задан параметр `-l`):

  ```
  wc -l /var/log/messages
  ```

- для подсчета количества символов (параметр `-c`):

  ```
  wc -c /var/log/messages.
  ```



Глава 8

Команды для работы с сетью и Интернетом

8.1. Команда *ifconfig*: управление сетевыми интерфейсами

Команда `ifconfig` используется для получения информации о сетевых интерфейсах и для установки параметров сетевых интерфейсов. Обычно эта команда вызывается при запуске системы сценариями инициализации системы, и вам не придется ее использовать для настройки интерфейсов вручную (разве что на самых древних дистрибутивах).

Формат вызова команды следующий:

```
ifconfig -a [параметры | семейство_протоколов]
ifconfig интерфейс [параметры | семейство_протоколов]
```

Чтобы просто просмотреть информацию о сетевых интерфейсах, введите команду:

```
ifconfig
```

Посмотрите на рис. 8.1. "Поднят" только интерфейс `lo` — это интерфейс локальный петли, используемой для тестирования сети. Если есть только интерфейс `lo`, значит, остальные сетевые интерфейсы не настроены. В современных дистрибутивах настройка интерфейсов производится автоматически при запуске системы — ведь практически всегда в сети есть DCHP-сервер, который и настраивает сетевые интерфейсы.

Настроить интерфейс можно и с помощью `ifconfig`. Вот пример настройки интерфейса `eth0` (первая сетевая плата), когда ему присваивается IP-адрес 192.168.1.7 и он поднимается ("up"):

```
# /sbin/ifconfig eth0 192.168.1.7 up
```

Для полной настройки сетевого интерфейса нужно использовать команду:

```
# /sbin/ifconfig eth0 адрес broadcast ш_адрес netmask маска
```

После имени интерфейса задается IP-адрес, затем широковещательный адрес (`ш_адрес`), потом сетевая маска. На рис. 8.2 показан уже настроенный интерфейс eth0.

```
[root@localhost /]# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:62 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3914 (3.8 Kb)  TX bytes:3914 (3.8 Kb)

[root@localhost /]#
```

Рис. 8.1. Настроен только интерфейс `lo`

```
[root@localhost /]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:6D:97:88:BC:96
          inet addr:192.168.1.7  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20d:87ff:fe88:bc96/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:672 (672.0 b)
          Interrupt:11 Base address:0xe800

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:83 errors:0 dropped:0 overruns:0 frame:0
          TX packets:83 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:5270 (5.1 Kb)  TX bytes:5270 (5.1 Kb)

[root@localhost /]#
```

Рис. 8.2. Интерфейсы `lo` и `eth0`

Вообще, перед тем как вводить команду `ifconfig`, неплохо было бы добавить модуль сетевой платы, без него `ifconfig` работать не будет. Вот пример добавления модуля для сетевой платы Realtek 8139:

```
# insmod rtl8139.o
```

8.2. Команда *route*

Команда `route` используется для вывода и изменения таблицы маршрутизации. Если ввести `route` без параметров, то мы увидим текущую таблицу маршрутизации. На маршрутизаторах она сложная, а на обычных компьютерах, как правило, нужно добавить только один маршрут по умолчанию, что можно сделать командой:

```
/sbin/route add default gw адрес_шлюза netmask 0.0.0.0 metric 1
```

8.3. Команда *pppoeconf*: настройка DSL-соединения

DSL (Digital Subscriber Line) — цифровая абонентская линия, позволяющая производить двунаправленный обмен данными по телефонной линии. Существуют несколько вариантов DSL-линий: ADSL, VDSL, SDSL, RADSL. Наиболее распространены ADSL-линии. ADSL (Asymmetric DSL) — асимметрическая цифровая линия. Для передачи данных используется витая пара телефонной сети. Скорость передачи данных зависит от расстояния, например 1,5 Мбит/с при расстоянии в 5–6 км. Но обычно скорость ограничивается провайдером и зависит от тарифного плана. Самый доступный тарифный план подразумевает скорость передачи данных 128 Кбит/с. Раньше были соединения и на скорости 64 Кбит/с, но сегодня уже таких нет. Сегодня обычным домашним пользователям доступны соединения со скоростью передачи данных от 1 до 20 Мбит/с. И это не просто слова. У меня синхронное соединение (когда скорость загрузки информации из Интернета равна скорости передачи данных в Интернет) 5 Мбит/с, а мой провайдер предлагает также асинхронные тарифные планы 10/1 и 20/2 Мбит/с (первое число — это скорость загрузки из Интернета, второе — скорость отправки в Интернет).

В Debian/Ubuntu/Denix для настройки DSL-соединения используется удобная программа pppoeconf, запустить ее можно так:

```
sudo pppoeconf
```

Согласно спецификации PPPoE существуют две стадии: стадия поиска и стадия сессии. На первой стадии производится отправка специальных пакетов PADI (PPPoE Active Discovery Initiation), которые позволяют найти активные концентраторы доступа PPPoE. Стадия сессии — это само соединение и передача информации.

Первым делом конфигуратор сообщит, что нашел сетевую плату (рис. 8.3), затем приступит к поиску PPPoE-концентратора (рис. 8.4).

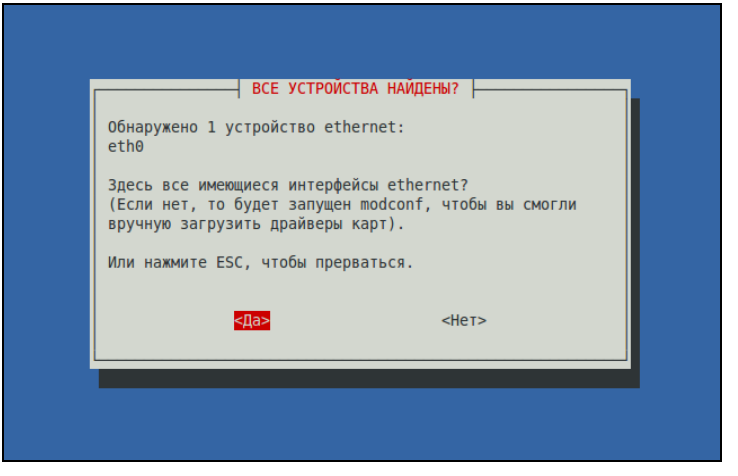


Рис. 8.3. Найдена сетевая плата, к которой подключен DSL-модем

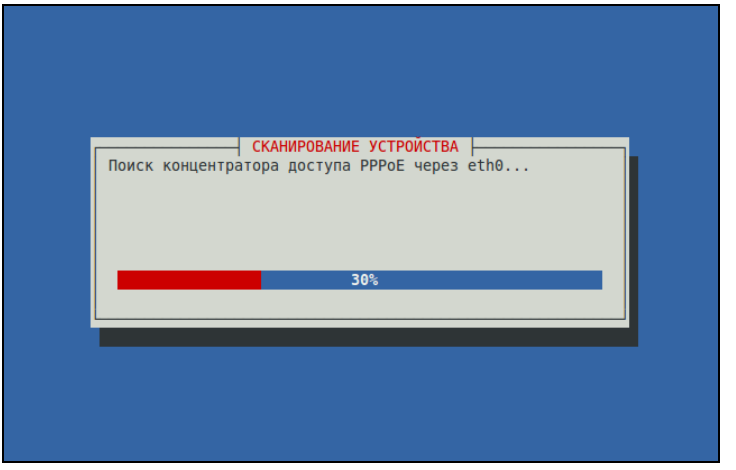


Рис. 8.4. `pppoeconf` нашел Ethernet-устройство

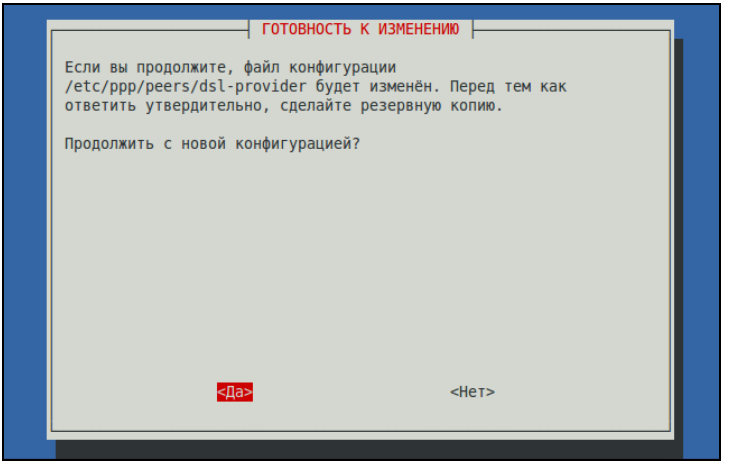


Рис. 8.5. Создайте копию файла /etc/ppp/peers/dsl-provider

После того как концентратор доступа будет найден, программа предложит вам создать резервную копию файла /etc/ppp/peers/dsl-provider (рис. 8.5), поскольку именно этот файл будет изменен в процессе настройки соединения. Если вы до этого не настраивали DSL-соединение, то в этом файле ничего не будет, поэтому можете даже не предпринимать никаких действий.

Далее программа предложит установить популярные опции соединения (`noauth` и `defaultroute`): не стоит от них отказываться, поскольку их использует большинство провайдеров (рис. 8.6).

Следующие два шага — ввод имени пользователя и пароля, которые используются для аутентификации на сервере провайдера. После этого программа предложит вам добавить полученные от провайдера IP-адреса DNS-серверов в файл /etc/resolv.conf. Не стоит от этого отказываться (рис. 8.7).

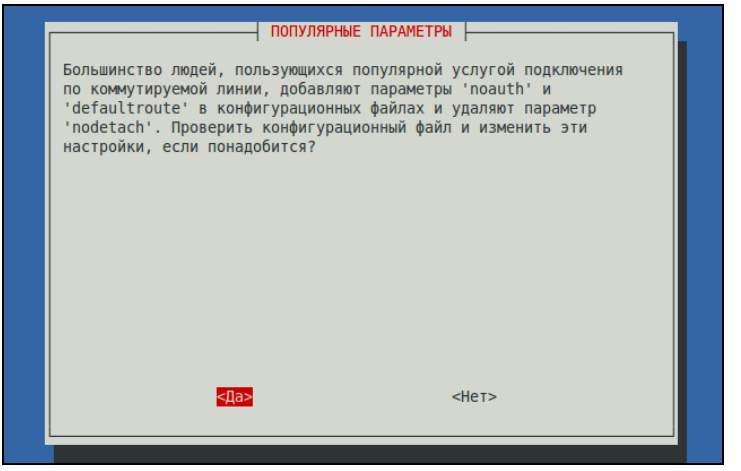


Рис. 8.6. Популярные опции соединения

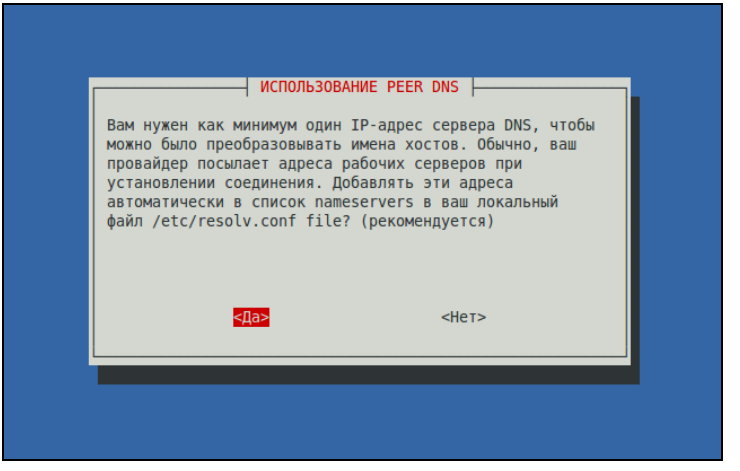


Рис. 8.7. Добавляем IP-адреса DNS-серверов в файл /etc/resolv.conf

На следующий вопрос можно просто ответить **Да**, не вникая в подробности (рис. 8.8). Если же вам интересно, прочитайте следующее примечание.

Примечание

Параметр MTU (Maximum Transmit Unit) задает максимальный размер пакета. По умолчанию данное значение может быть установлено автоматически, но не всегда оптимально. Если размер пакета будет большим, чем позволяет машрутизатор провайдера, тогда пакет будет разделен на несколько пакетов, что, естественно, скажется на скорости и пропускной способности соединения. Если размер пакета будет меньше, чем положено, то тоже не хорошо — канал будет использован нерационально, ведь будут проходить полупустые кадры. Поскольку у нас PPPoE, то нужно учитывать несколько факторов. Максимальный размер кадра Ethernet составляет 1518 байтов, из которых 18 уходит на заголовок и контроль, поэтому для полезных данных остается 1500 байтов. Обычно данное значение и указывается для Ethernet. Но ведь

по Ethernet мы собираемся передавать пакеты PPP, а PPPoE отбирает еще 6 байтов, PPP — 2 байта. Получается, что для PPPoE значение MTU должно быть равно 1492. При установке TCP-соединения каждая сторона устанавливает параметр MSS (Maximum Segment Size), максимальный размер TCP-сегмента. По умолчанию его размер равен MTU минус размер заголовков TCP/IP, которые занимают еще 40 байтов. То есть размер MSS для PPPoE равен 1452 байта (для обычного Ethernet — 1460). Вот откуда взялось значение 1452.

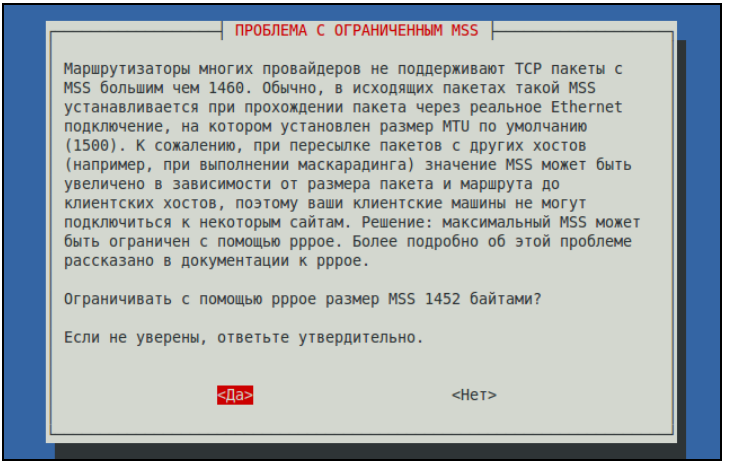


Рис. 8.8. Установка размера MSS

Следующий вопрос — хотите ли вы устанавливать соединение при загрузке системы (рис. 8.9)? Тут уж решайте сами. А после этого программа спросит вас, хотите ли вы установить соединение немедленно. Конечно, да! Можно сразу запускать браузер и заходить на любимую страничку.

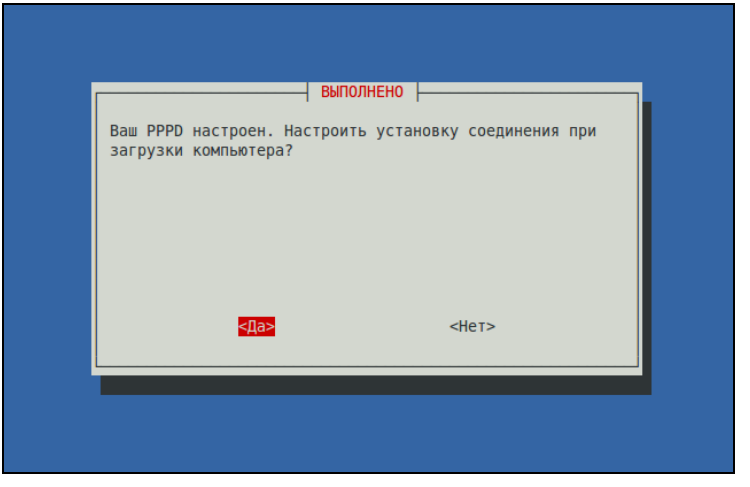


Рис. 8.9. Устанавливать соединение автоматически?

Для включения/отключения DSL-соединения используются следующие команды:

```
sudo pon dsl-provider
sudo poff dsl-provider
```

8.4. Команда *pppconfig*: настройка модемного (PPP) соединения

Кроме команды `pppoeconf` в дистрибутивах Debian/Ubuntu/Denix есть команда `pppconfig`, которая настраивает модемное PPP-соединение. Она напоминает программу `pppoeconf` — вам нужно запустить ее и следовать инструкциям мастера настройки соединения. Ничего сложного. Запускать программу нужно так:

```
sudo pppoeconf
```

8.5. Команда *wvdial*: настройка PPP-соединения

Программа `wvdial` также используется для создания PPP-соединения, но, чтобы не рассматривать две программы для создания таких соединений, мы рассмотрим ее для настройки GPRS/EDGE-соединения, что более актуально в наши дни, нежели модемное соединение.

Вам нужно раздобыть настройки GPRS вашего оператора. Проще всего найти компьютер, подключенный к Интернету, и зайти на сайт оператора. В большинстве случаев там все описано. Если нет, то придется звонить в службу поддержки оператора и записывать параметры под диктовку.

Для подключения к Интернету мы будем использовать программу `wvdial` — это значительно проще, чем создавать собственные скрипты для демона `pppd`. Данная программа обычно входит в состав дистрибутива, но не устанавливается по умолчанию. Поэтому ее нужно установить:

```
# rpm -ihv wvdial*
sudo apt-get install wvdial
# yum install wvdial
```

Откройте конфигурационный файл /etc/wvdial.conf и добавьте в него следующие строки (листинг 8.1).

Листинг 8.1. Добавляемый фрагмент конфигурационного файла /etc/wvdial.conf

```
[Dialer Defaults]
# Порт, к которому подключен телефон
Modem = /dev/ttyS0            # если телефон подключен к COM-порту
# Modem = /dev/ircomm0        # инфракрасный порт
# Modem = /dev/ttyUSB0        # телефон подключен по USB
# Modem = /dev/ttyACM0        # телефон подключен по USB (см. примечание)
```

```
# Скорость
Baud = 115200                # можно не изменять

# Стандартная строка инициализации модема
Init1 = ATZ

# Строка инициализации GPRS
Init2 = AT+CGDCONT=1,"IP","AP"

# Не изменяйте эти параметры
# Для контрактных абонентов некоторых операторов опцию ISDN нужно
# установить в 1
ISDN = 0
Modem Type = Analog Modem
Carrier Check = no

# Номер, по которому осуществляется соединение
Phone = *99#
# Имя пользователя и пароль
Username = логин
Password = пароль
```

Примечание

Некоторые телефоны, подключаемые к компьютеру по USB, определяются в Linux как устройства /dev/ttyACM*n* (где *n* — номер). Следовательно, если не выходит подключиться с использованием устройства /dev/ttyUSB0, тогда следует попробовать использовать устройство /dev/ttyACM*n*.

Давайте разберемся, что нужно изменить вам "под себя". Во-первых, имя модема. В данном случае используется первый последовательный порт (`ttyS0`). Если вы подключаетесь с помощью USB-кабеля, то нужно указать другое имя устройства, например `ttyUSB0`. Во-вторых, необходимо указать точку доступа своего оператора (`AP`) — эту информацию надо получить у него. Например, для МТС точка доступа будет такой: `internet.mts.ru`.

Затем следует изменить телефон доступа. Он зависит не только от оператора, но и от модели самого телефона, например:

- `*99#` — для телефонов Nokia, Ericsson, Motorola, SonyEricsson, Sendo;
- `*99***1#` — для телефонов Siemens, Alcatel, Handspring, LG, Panasonic, Mitsubishi, Sagem или если у вас Киевстар;
- `*99**1*1#` — для Samsung.

Комбинация логина и пароля зависит от вашего оператора — например, для МТС нужно указать имя пользователя `mts` и такой же пароль (тоже `mts`). Уточните эти параметры у своего оператора.

Почти все. Сохраните файл /etc/wvdial.conf и откройте /etc/resolv.conf. Добавьте в него IP-адреса DNS-серверов вашего оператора:

```
nameserver XXX.XXX.XXX.XXX
nameserver УУУ.УУУ.УУУ.УУУ
```

В большинстве случаев IP-адреса DNS-серверов передаются автоматически по протоколу DHCP, но в некоторых случаях их нужно указать явно в файле /etc/resolv.conf. По этому поводу вам тоже лучше проконсультироваться с вашим оператором.

Все готово. Для запуска GPRS-соединения введите команду: `# wvdial`.

В файле протокола /var/log/messages вы увидите заветные строки:

```
Serial connection established.
Using interface ppp0
```

Соединение с Интернетом установлено. Можете запускать браузер и работать.

Если же произошла ошибка, то ее поиск рекомендую проводить в следующем направлении:

- параметры доступа к GPRS — уточните их у оператора, возможно, вы что-то не так указали, например перепутали номер телефона или ввели не тот IP-адрес сервера DNS. Как уже отмечалось, параметры GPRS обычно выложены на сайте оператора. Если другого доступа к Интернету нет, можно оператору (в его службу поддержки) просто позвонить;
- сценарии — проверьте имя файла модема, правильность написания самих сценариев (опечатки тоже возможны);
- кабель — если вы все делаете правильно, а соединение установить не удается, попробуйте другой кабель. В продаже есть так называемые "неполноценные" кабели, которые немного дешевле, но именно с ними и возникают проблемы при установке GPRS-соединения.

Если все сделано правильно, а связи нет, поищите в Интернете рекомендации по установке GPRS-соединения с тем или иным оператором (иногда случаются недокументированные особенности).

Для отключения от Интернета введите команду: `# killall pppd`.

8.6. Текстовые браузеры

Если графический режим недоступен (например, на сервере), а по сети побродить хочется, можно использовать текстовый браузер `lynx`. В некоторых дистрибутивах вместо `lynx` используются браузеры `links` и `elinks`, но суть остается та же — просмотр страниц Интернета в текстовом режиме.

В современных дистрибутивах текстовые браузеры не устанавливаются по умолчанию, поэтому их нужно установить отдельно.

8.7. Команда *ftp*: FTP-клиент

Для открытия соединения с любым FTP-сервером введите команду:

```
ftp <имя или адрес FTP-сервера>
```

Можно просто ввести команду `ftp`, а в ответ на приглашение

```
ftp>
```

ввести команду:

```
open <имя или адрес FTP-сервера>
```

Лично мне больше нравится первый вариант, поскольку он позволяет сэкономить время. При подключении к серверу вы сможете ввести имя пользователя и пароль:

```
[den@dhsilabs ~]$ ftp
ftp> open ftp.narod.ru
Connected to ftp.narod.ru.
220 ftp.narod.ru (Libra FTP daemon 0.17 20050906)
500 Unrecognized command AUTH
Name (ftp.narod.ru:den): den
331 Password required
Password:
230 Logged in, proceed
Remote system type is UNIX.
ftp>
```

Подключившись к серверу, вы можете ввести команду `help`, чтобы просмотреть список доступных команд. Для получения справки по той или иной команде введите `help <имя_команды>`. Наиболее популярные команды приведены в табл. 8.1.

Таблица 8.1. *Некоторые команды FTP-клиента*

Команда	Описание
`ls`	Вывод содержимого каталога
`get`	Загрузить файл с сервера
`put`	Загрузить файл на сервер
`mget`	Получить несколько файлов с сервера. Допускается использование масок файлов, например *.rpm
`mput`	Загрузить несколько файлов на сервер
`cd`	Изменить каталог
`mkdir`	Создать каталог
`rmdir`	Удалить пустой каталог
`delete`	Удалить файл

Кроме ftp, в Linux есть и другие текстовые FTP-клиенты, например: NcFTP (**http://www.ncftp.com**), lukemftp (**ftp://ftp.netbsd.org/pub/NetBSD/misc/lukemftp/**), lftp (**http://ftp.yars.free.net/projects/lftp/**) и др. Все эти FTP-клиенты не входят в состав дистрибутива, их нужно устанавливать самостоятельно. Но стоит ли это делать — решать вам. Ведь все они подобны стандартному клиенту ftp и обладают двумя-тремя дополнительными функциями, которые, возможно, вам и не понадобятся. Например, NcFTP умеет докачивать файлы, а lftp — загружать одновременно несколько файлов. В любом случае, вы можете изучить документацию по тому или иному FTP-клиенту (ее легко найти в Интернете), а потом решить, стоит его использовать или нет.

8.8. Команда *wget*: загрузка файлов

Программа wget — это лучший текстовый менеджер закачки файлов. Программа поддерживает протоколы HTTP, HTTPS и FTP. Использовать ее нужно так:

```
wget [параметры] URL
```

Параметров у wget очень и очень много, и со всеми ними вы ознакомитесь на странице man wget. Самые полезные параметры собраны в табл. 8.2.

Таблица 8.2. *Некоторые параметры* `wget`

Параметр	Описание
`--background`	Перейди в фоновый режим после запуска
`--quiet`	Тихий режим, сообщения wget не выводятся
`--input-file=file`	Считать URL из файла file, файл не обязательно должен быть в формате HTML. Если вы указали URL в файле и в командной строке, то сначала будут загружены URL из командной строки, а потом из файла
`--force-html`	Обязательно считать файл, указанный в предыдущем параметре, HTML-файлом
`--tries=number`	Устанавливает количество попыток загрузки URL
`--no-clobber`	Если при загрузке файла оборвалось соединение, то этот параметр позволит продолжить загрузку с места обрыва
`--continue`	Возобновление загрузки файла, например, если прервалась связь. Этот параметр нужно использовать, если вы забыли указать параметр --no-clobber, а связь прервалась и вам нужно докачать файл, а не начинать его загрузку заново
`--wait=seconds`	Задает паузу в секундах между загрузками и повторами, что позволяет снизить нагрузку на сервер
`--quota=quota`	Задает максимальный размер загружаемых файлов (в байтах, килобайтах (после числа указывается k) и мегабайтах (после числа — m)). Квота не работает при загрузке одного файла, поскольку даже если квота превышена, то текущий файл загружается до конца (если есть физически место на диске)

***Таблица 8.2** (окончание)*

Параметр	Описание
`--http-user=user` `--http-passwd=pass`	Задают имя пользователя и пароль при HTTP-аутентификации, тип аутентификации устанавливается автоматически программой
`--proxy-user=user` `--proxy-passwd=pass`	Задает имя пользователя и пароль прокси-сервера
`--passive-ftp`	Пассивный режим FTP, обычно используется при наличии брандмауэра
`--recursive`	Включить рекурсивную загрузку, которая используется для рекурсивной загрузки сайтов
`--level=depth`	Максимальная длина рекурсивной загрузки (по умолчанию 5 уровней)

Примеры использования:

```
wget --recursive http://dkws.org.ua
wget http://dkws.org.ua/1.zip
```

Первая команда создаст пятиуровневую копию сайта **http://dkws.org.ua**, а вторая просто загрузит файл 1.zip с **http://dkws.org.ua**.

8.9. Команды для диагностики сети

Причины отказа сети могут быть физическими и программными. Физические связаны с неработающим сетевым оборудованием или повреждением среды передачи данных. Программные связаны с неправильной настройкой сетевого интерфейса. Как правило, избавиться от программных проблем помогает конфигуратор сети — вы еще раз его запускаете и настраиваете сетевые интерфейсы, только правильно. Если сомневаетесь в ваших действиях, обратитесь за помощью к более опытному коллеге.

Для диагностики работы сети мы будем использовать стандартные сетевые утилиты, которые входят в состав любого дистрибутива Linux. Предположим, что у нас не работает PPPoE/DSL-соединение. Проверить, "поднят" ли сетевой интерфейс, можно с помощью команды `ifconfig`. На рис. 8.10 видно, что сначала я предпринял попытку установить соединение (ввел команду `sudo pon dsl-provider`), а затем вызвал `ifconfig`, чтобы убедиться, установлено ли соединение. В случае если соединение не было бы установлено, интерфейса `ppp0` в списке не было. Интерфейс `eth0` (рис. 8.10) относится к первой сетевой плате (вторая называется eth1, третья — eth2 и т. д.), а интерфейс `lo` — это интерфейс обратной петли, который используется для тестирования программного обеспечения (у вас он всегда будет "поднят").

Если же интерфейс не поднят, нам нужно просмотреть файл /var/log/messages сразу после попытки установки сообщения:

```
tail -n 10 /var/log/messages
```

```
user@user-desktop:~$ sudo pon dsl-provider
Password:
Plugin rp-pppoe.so loaded.
user@user-desktop:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0D:87:88:BC:96
          inet6 addr: fe80::20d:87ff:fe88:bc96/64 Диапазон:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:629 errors:0 dropped:0 overruns:0 frame:0
          TX packets:121 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:104484 (102.0 KiB)  TX bytes:11682 (11.4 KiB)
          Interrupt:11 Base address:0xe800

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Диапазон:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:25 errors:0 dropped:0 overruns:0 frame:0
          TX packets:25 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1744 (1.7 KiB)  TX bytes:1744 (1.7 KiB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:193.254.218.243  P-t-P:193.254.218.129  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1488  Metric:1
          RX packets:107 errors:0 dropped:0 overruns:0 frame:0
          TX packets:95 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:32174 (31.4 KiB)  TX bytes:6001 (5.8 KiB)

user@user-desktop:~$
```

Рис. 8.10. Программа `ifconfig`

Данная команда просматривает "хвост" файла протокола (выводит последние 10 сообщений). В случае удачной установки соединения сообщения в файле протокола будут примерно следующими:

```
Feb  6 14:28:33 user-desktop pppd[5176]: Plugin rp-pppoe.so loaded.
Feb  6 14:28:33 user-desktop kernel: [17179852.932000] CSLIP: code copyright 198 9 Regents
                                     of the University of California
Feb  6 14:28:33 user-desktop kernel: [17179852.944000] PPP generic driver versio n 2.4.2
Feb  6 14:28:33 user-desktop pppd[5183]: pppd 2.4.4b1 started by root, uid 0
Feb  6 14:28:33 user-desktop pppd[5183]: PPP session is 2838
Feb  6 14:28:33 user-desktop kernel: [17179852.984000] NET: Registered protocol family 24
Feb  6 14:28:33 user-desktop pppd[5183]: Using interface ppp0
Feb  6 14:28:33 user-desktop pppd[5183]: Connect: ppp0 <--> eth0
Feb  6 14:28:33 user-desktop pppd[5183]: Remote message: Login ok
Feb  6 14:28:33 user-desktop pppd[5183]: PAP authentication succeeded
Feb  6 14:28:33 user-desktop pppd[5183]: peer from calling number 00:15:F2:60:28 :97
                                         authorized
Feb  6 14:28:33 user-desktop pppd[5183]: local  IP address 193.254.218.243
Feb  6 14:28:33 user-desktop pppd[5183]: remote IP address 193.254.218.129
Feb  6 14:28:33 user-desktop pppd[5183]: primary   DNS address 193.254.218.1
Feb  6 14:28:33 user-desktop pppd[5183]: secondary DNS address 193.254.218.27
```

Первая строчка — сообщение о том, что загружен модуль поддержки PPPoE. Следующие два сообщения информируют нас о поддержке нашим компьютером протоколов CSLIP и PPP. После сообщается, что демон `pppd` запущен, сообщается, от чьего имени он запущен (root) и версия самого `pppd`. Далее сообщается имя используемого интерфейса (`ppp0`) и имя вспомогательного интерфейса (помните, что протокол PPPoE подразумевает передачу кадров PPP по Ethernet) — `eth0`. Следующие два сообщения свидетельствуют об удачной регистрации:

```
Feb  6 14:28:33 user-desktop pppd[5183]: Remote message: Login ok
Feb  6 14:28:33 user-desktop pppd[5183]: PAP authentication succeeded
```

Затем система сообщает нам наш IP-адрес, адрес удаленного компьютера, который произвел аутентификацию, а также IP-адреса серверов DNS.

А вот пример неудачной попытки соединения:

```
Feb  6 09:23:48 user-desktop pppd[6667]: PPP session is 2336
Feb  6 09:23:48 user-desktop pppd[6667]: Using interface ppp1
Feb  6 09:23:48 user-desktop pppd[6667]: Connect: ppp1 <--> eth0
Feb  6 09:23:48 user-desktop pppd[6667]: Remote message: Login incorrect
Feb  6 09:23:48 user-desktop pppd[6667]: Connection terminated.
```

Причина неудачи понятна: имя пользователя или пароль неправильные, о чем красноречиво свидетельствует сообщение "Login incorrect". Для того чтобы изменить имя пользователя или пароль, запустите конфигуратор `pppoeconf`. Но не спешите этого делать: если в предыдущий раз соединение было установлено (а настройки соединения вы не изменяли), возможно, нужно обратиться к провайдеру — это явный признак неправильной работы оборудования на стороне провайдера.

Вот еще один пример, характерный для PPPoE:

```
Feb  6 09:23:48 user-desktop pppd[6667]: PPP session is 2336
Feb  6 09:23:48 user-desktop pppd[6667]: Using interface ppp1
Feb  6 09:23:48 user-desktop pppd[6667]: Connect: ppp1 <--> eth0
Feb  6 09:23:48 user-desktop pppd[6667]: Connection terminated.
```

Это явный пример неправильной работы оборудования провайдера. Возможно, нужно перезагрузить точку доступа (access point), т. е. просто выключите и включите ее. Если это не помогает, тогда обращайтесь к провайдеру.

Наиболее простая ситуация, когда сеть вообще не работает. В этом случае очень легко обнаружить причину неисправности. Если работает устройство, значит, повреждена среда передачи данных (сетевой кабель). В случае с модемной линией нужно проверить, нет ли ее обрыва. В случае с витой парой обрыв маловероятен (хотя возможен), поэтому нужно проверить, правильно ли обжат кабель (возможно, нужно обжать витую пару заново).

Намного сложнее ситуация, когда сеть то работает, то нет. Например, вы не можете получить доступ к какому-нибудь узлу, хотя пять минут назад все работало отлично. Если исключить неправильную работу удаленного узла, к которому вы подключаетесь, следует поискать решение в маршруте, по которому пакеты добираются от вашего компьютера до удаленного узла. Сначала *пропингуем* удаленный

узел. Для этого используется команда `ping` (прервать выполнение команды `ping` можно с помощью нажатия комбинации клавиш <Ctrl>+<C>):

```
ping dkws.org.ua

PING dkws.org.ua (213.186.114.75) 56(84) bytes of data.
64 bytes from wdt.org.ru (213.186.114.75): icmp_seq=1 ttl=58 time=30.7 ms
64 bytes from wdt.org.ru (213.186.114.75): icmp_seq=2 ttl=58 time=24.8 ms
64 bytes from wdt.org.ru (213.186.114.75): icmp_seq=5 ttl=58 time=12.2 ms
64 bytes from wdt.org.ru (213.186.114.75): icmp_seq=6 ttl=58 time=159 ms
64 bytes from wdt.org.ru (213.186.114.75): icmp_seq=7 ttl=58 time=19.3 ms
64 bytes from wdt.org.ru (213.186.114.75): icmp_seq=9 ttl=58 time=29.0 ms
...
```

В этом случае все нормально. Но иногда ответы от удаленного сервера то приходят, то не приходят. Чтобы узнать, в чем причина (где именно теряются пакеты), нужно выполнить трассировку узла:

```
tracepath dkws.org.ua
```

```
user@user-desktop:~$ tracepath dkws.org.ua
 1:  ip-193-254-218-243.romb.net (193.254.218.243)          0.320ms pmtu 1488
 1:  ip-193-254-218-129.romb.net (193.254.218.129)         94.739ms
 2:  sat-router.romb.net (193.254.218.2)                   16.841ms
 3:  border.romb.net (80.91.172.97)                        48.562ms
 4:  L9-KTU.rtr.newline.net.ua (80.91.178.81)             109.070ms
 5:  utel-gw.ix.net.ua (195.35.65.89)                     asymm  6  54.850ms
 6:  dc-m7i-1-ge.interfaces.dc.utel.ua (213.186.112.129)  asymm  7  29.092ms
 7:  no reply
 8:  no reply
 9:  no reply
10:  no reply
11:  no reply
12:  no reply
13:  no reply
14:  no reply
15:  no reply
16:  no reply
17:  no reply
18:  no reply
19:  no reply
20:  no reply
21:  no reply
22:  no reply
23:  no reply
24:  no reply
25:  no reply
26:  no reply
27:  no reply
28:  no reply
29:  no reply
30:  no reply
31:  no reply
     Too many hops: pmtu 1488
```

Рис. 8.11. Проблема с прохождением пакетов

В некоторых дистрибутивах вместо команды `tracepath` используется команда `traceroute`, а в Windows — `tracert`. На рис. 8.11 изображено выполнение команды `tracepath`. Сразу видно, что есть определенные проблемы с прохождением пакетов до удаленного узла.

Понятно, что по пути пакеты теряются. Для того чтобы выяснить причину, вам нужно обратиться к администратору того маршрутизатора, который не пропускает дальше пакеты. Причина именно в нем. В данном случае, как видно из рисунка, пакеты доходят до маршрутизатора dc-m7i-1-ge.interfaces.dc.utel.ua, а после него движение пакетов прекращается.

Если соединение установлено (о чем свидетельствует наличие поднятого интерфейса в выводе `ifconfig`), а Web-страницы не открываются, попробуйте пропинговать любой удаленный узел по IP-адресу. Если не знаете, какой узел пинговать (т. е. не помните ни одного IP-адреса), пропингуйте узел 213.186.114.75. Если вы получите ответ, а странички по-прежнему не открываются, когда вы вводите символьное имя, значит, у вас проблемы с DNS: сервер провайдера почему-то не передал вашему компьютеру IP-адреса DNS-серверов. Позвоните провайдеру, выясните причину этого, а еще лучше уточните IP-адреса серверов DNS и укажите их в файле /etc/resolv.conf. Формат этого файла прост:

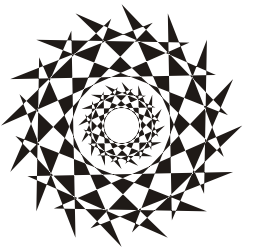
```
nameserver IP-адрес
```

Например:

```
nameserver 193.254.218.1
nameserver 193.254.218.27
```

Всего можно указать до четырех серверов DNS.

Если же не открывается какая-то конкретная страничка, а все остальные работают нормально, тогда понятно, что причина в самóм удаленном сервере, а не в ваших настройках.



Глава 9

Команды системного администратора

9.1. Программы разметки диска

В этой главе будут рассмотрены две программы для разметки диска — классическая программа `fdisk` и более продвинутая `parted`. Реально для разметки диска (если вам придется это делать) вы будете использовать `parted`, поскольку она умеет изменять размеры разделов, что пригодится при переразметке диска. А вот `fdisk` можно использовать разве что для разметки новых жестких дисков. Изменить размер раздела без потери данных `fdisk` не может. Вам нужно удалить один из разделов, а вместо него создать несколько разделов меньшего размера — только так и не иначе.

Зато `fdisk` установлена по умолчанию во всех дистрибутивах и ее не нужно доустанавливать самостоятельно.

9.1.1. Программа *fdisk*

Введите команду (можно использовать короткие имена):

```
# fdisk <имя_устройства>
```

Например, если вы подключили винчестер как вторичный мастер, то команда будет следующей:

```
# fdisk /dev/sda
```

Чтобы убедиться, что диск не размечен, введите команду `p`. Программа выведет пустую таблицу разделов (рис. 9.1).

Самое время создать раздел. Для этого используется команда `n` (рис. 9.2). Кстати, для справки можете ввести команду `m`, которая выведет список доступных команд `fdisk` (рис. 9.3).

```
Command (m for help): p

Disk /dev/sda: 1825 MB, 1825360896 bytes
64 heads, 63 sectors/track, 884 cylinders
Units = cylinders of 4032 * 512 = 2064384 bytes

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): _
```

Рис. 9.1. Таблица разделов пуста

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-884, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-884, default 884): +700M
```

Рис. 9.2. Создание нового раздела

```
64 heads, 63 sectors/track, 884 cylinders
Units = cylinders of 4032 * 512 = 2064384 bytes

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): m
'Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help): _
```

Рис. 9.3. Список команд программы `fdisk`

После ввода команды `n` программа попросит вас уточнить, какого типа должен быть раздел. Можно выбрать первичный или расширенный раздел. В нашем случае больше подойдет первичный, поэтому вводим букву `p`. Затем нужно ввести номер раздела. Поскольку это первый раздел, то вводим `1`. После чего `fdisk` попросит ввести номер первого цилиндра. Это первый раздел, поэтому вводим номер `1`. После ввода первого цилиндра нужно ввести номер последнего цилиндра. Чтобы не высчитывать на калькуляторе номер цилиндра, намного проще ввести размер раздела. Делается это так: `+`*<размер>*`M`. После числа должна идти именно буква `M`, иначе размер будет воспринят в байтах, а этого нам не нужно. Например, если вы хотите создать раздел размером 10 Гбайт, то введите `+10240M`.

Для создания второго раздела опять введите команду `n`. Программа вновь попросит тип раздела, номер первого цилиндра (это будет номер последнего цилиндра первого раздела плюс 1) и размер раздела. Если вы хотите создать раздел до "конца" диска, то просто введите номер последнего цилиндра.

Теперь посмотрим на таблицу разделов. Для этого опять введите команду `p` (рис. 9.4).

```
Command (m for help): p

Disk /dev/sda: 1825 MB, 1825360896 bytes
64 heads, 63 sectors/track, 884 cylinders
Units = cylinders of 4032 * 512 = 2064384 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1         340      685408+  83  Linux
/dev/sda2             341         884     1096704   83  Linux

Command (m for help): _
```

Рис. 9.4. Создание второго раздела, вывод таблицы разделов

По умолчанию программа `fdisk` создает Linux-разделы. Если вы собираетесь работать только в Linux, можно оставить и так, но ведь не у всех есть Linux. Если вы снимете этот винчестер, чтобы, например, переписать у товарища большие файлы, то вряд ли сможете комфортно с ним работать. Прочитать данные (например, с помощью Total Commander) вам удастся, а что-либо записать — уже нет. Поэтому давайте изменим тип разделов. Для этого используется команда `t`. Введите эту команду. Программа запросит у вас номер раздела и тип файловой системы. С номером раздела все ясно, а вот с кодом файловой системы сложнее. Введите `L`, чтобы просмотреть доступные файловые системы (рис. 9.5).

```
 0  Empty           1e  Hidden W95 FAT1 80  Old Minix       be  Solaris boot
 1  FAT12           24  NEC DOS         81  Minix / old Lin bf  Solaris
 2  XENIX root      39  Plan 9          82  Linux swap / So c1  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  83  Linux           c4  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 5  Extended        41  PPC PReP Boot   85  Linux extended  c7  Syrinx
 6  FAT16           42  SFS             86  NTFS volume set da  Non-FS data
 7  HPFS/NTFS       4d  QNX4.x          87  NTFS volume set db  CP/M / CTOS / .
 8  AIX             4e  QNX4.x 2nd part 88  Linux plaintext de  Dell Utility
 9  AIX bootable    4f  QNX4.x 3rd part 8e  Linux LVM       df  BootIt
 a  OS/2 Boot Manag 50  OnTrack DM      93  Amoeba          e1  DOS access
 b  W95 FAT32       51  OnTrack DM6 Aux 94  Amoeba BBT      e3  DOS R/O
 c  W95 FAT32 (LBA) 52  CP/M            9f  BSD/OS          e4  SpeedStor
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi eb  BeOS fs
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a5  FreeBSD         ee  EFI GPT
10  OPUS            55  EZ-Drive        a6  OpenBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    56  Golden Bow      a7  NeXTSTEP        f0  Linux/PA-RISC b
12  Compaq diagnost 5c  Priam Edisk     a8  Darwin UFS      f1  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       a9  NetBSD          f4  SpeedStor
16  Hidden FAT16    63  GNU HURD or Sys ab  Darwin boot     f2  DOS secondary
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fd  Linux raid auto
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fe  LANstep
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid ff  BBT
1c  Hidden W95 FAT3 75  PC/IX
Hex code (type L to list codes): _
```

Рис. 9.5. Коды файловых систем

Код FAT32 — `b`. Введите его, и вы увидите сообщение программы, что тип файловой системы изменен (рис. 9.6).

Еще раз введите команду `p`, чтобы убедиться, что все нормально. Для сохранения таблицы разделов введите `w`, а для выхода без сохранения изменений — `q`.

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): b
Changed system type of partition 2 to b (W95 FAT32)

Command (m for help): _
```

Рис. 9.6. Тип файловой системы изменен

9.1.2. Программа *parted*

Утилита `parted` (название — сокращение от PARTition EDitor) является консольной программой, которая используется для создания, удаления, копирования, изменения размера и размещения разделов диска.

Программа поддерживает следующие таблицы разделов:

- ❒ BSD;
- ❒ MSDOS;
- ❒ SUN;
- ❒ MAC;
- ❒ PC98;
- ❒ GPT.

Кроме того, программа поддерживает прямой доступ (raw access) к диску, что полезно при работе с логическими томами (LVM) и RAID-массивами.

Программа `parted` поддерживает множество файловых систем, но не для всех файловых систем доступны все выполняемые программой действия. В табл. 9.1 представлена информация о действиях, которые можно выполнить над той или иной файловой системой.

Таблица 9.1. *Поддерживаемые действия*

Файловая система	Обнаружение	Создание	Изменение размера	Копирование	Проверка
ext3	+		+	+	+
ext2	+	+	+	+	+
fat32	+	+	+	+	+
fat16	+	+	+	+	+
ntfs	+	+	+	+	+
linux-swap	+	+	+	+	+
ReiserFS	+	+	+	+	+
JFS	+				
XFS	+				
UFS	+				

Программа не умеет создавать разделы ext3 и ext4, но она может создать раздел ext2, который можно без особых проблем преобразовать в ext3 или ext4 *(см. главу 4)*. Разделы типов JFS, XFS и UFS только обнаруживаются программой, но она не может выполнять над ними никаких действий.

Примечание

Для работы с NTFS-разделами обязательна установка пакета linux-ntfs.

Запустим `parted`:

```
# parted <имя устройства>
```

Например:

```
# parted /dev/sda
```

Примечание

Не забывайте, что программа должна быть выполнена от имени root! Получить права root можно с помощью команды `sudo`, например `sudo parted /dev/sda`.

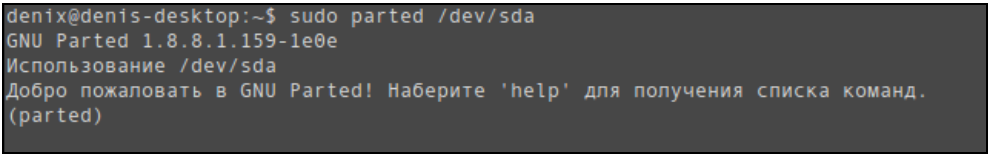


Рис. 9.7. Программа `parted`

Введите команду `print` для просмотра имеющихся разделов:

```
(parted) print
Disk geometry for /dev/sda: 0.000-9990.109 megabytes
Disk label type: msdos
Minor   Start     End   Type            Filesystem        Flags
1     0.031    512.000  primary         linux-swap
2   512.000    9990.109 primary         ext2              boot
```

Первая колонка — это номер раздела, вторая и третья — смещение (в мегабайтах) от "начала" диска. Следующая колонка — тип раздела, далее — тип файловой системы. Последняя колонка — флаги, например boot — загрузочный раздел.

Введите команду `help`, чтобы увидеть список команд `parted` (рис. 9.8). Команды `parted` приведены в табл. 9.2.

Таблица 9.2. *Основные команды* `parted`

Команда	Описание
`check n`	Проверить раздел с номером *n*
`cp [устройство] n m`	Копировать файловую систему из раздела *n* в раздел *m*, устройство — это номер устройства, где находится раздел *n*. Если устройство не задано, то считается, что используется текущее устройство
`mklabel тип`	Создает новую метку диска
`mktable тип`	Создает новую таблицу разделов
`mkfs n тип_фс`	Создает файловую систему заданного типа на разделе *n*

***Таблица 9.2** (окончание)*

Команда	Описание
`mkpart тип [фс] нач кон`	Создать раздел указанного типа, `[фс]` — необязательный параметр, задающий тип файловой системы. Параметры `нач` и `кон` задают начало и конец раздела
`move n нач кон`	Переместить раздел с номером *n*, `нач` и `кон` — конечные "координаты" раздела, его начало и конец, заданные как смещение от "начала" диска в мегабайтах
`print [devices|free|all|n]`	Отображает таблицу разделов (если параметры не заданы), список устройств (`devices`), свободное место (`free`), все найденные разделы (`all`) или информацию о разделе с номером *n*
`quit`	Выход из программы
`rescue нач кон`	Восстанавливает потерянный раздел в промежутке, заданном параметрами `нач` и `кон`
`resize n нач кон`	Изменяет размер раздела *n*
`rm n`	Удаляет раздел с номером *n*
`select устройство`	Выбирает устройство для редактирования, вам нет необходимости выходить из программы для изменения устройства
`set n флаг состояние`	Изменяет состояние флага для раздела *n*. Доступные флаги описаны в табл. 9.3
`unit устройство`	Устанавливает устройство по умолчанию
`version`	Выводит версию parted

```
GNU Parted 1.8.8.1.159-1e0e
Использование /dev/sda
Добро пожаловать в GNU Parted! Наберите 'help' для получения списка команд.
(parted) help
  check НОМЕР                      производит простую проверку файловой системы
  cp [ИЗ_УСТРОЙСТВА] ИЗ_НОМ В_НОМ скопировать файловую систему на другой раздел
  help [КОМАНДА]                   распечатать общую справку или справку по
        КОМАНДЕ
  mklabel,mktable ТИП_МЕТКИ         создать новую метку диска (таблицу раздела)
  mkfs НОМЕР ТИП_ФС                создать файловую систему ТИП_ФС на разделе
        НОМЕР
  mkpart ТИП_РАЗД [ТИП_ФС] НАЧ КОН создать раздел
  mkpartfs ТИП_РАЗД ТИП_ФС НАЧ КОН создать раздел с файловой системой
  move НОМЕР НАЧ КОН               переместить файловую систему НОМЕР
  name НОМЕР ИМЯ                   назначает имя разделу НОМЕР на ИМЯ
  print [devices|free|list,all|НОМЕР]     отображает таблицу разделов, доступные
        устройства, свободное место, все найденные разделы или определённый
        раздел
  quit                             выйти из программы
  rescue НАЧАЛО КОНЕЦ              восстановить потерянный раздел в промежутке
        от НАЧАЛА до КОНЦА
  resize НОМЕР НАЧ КОН             изменить размер файловой системы на разделе
        НОМЕР
  rm НОМЕР                         удалить раздел НОМЕР
  select УСТРОЙСТВО                выбор устройства для редактирования
  set НОМЕР ФЛАГ СОСТОЯНИЯ         изменить ФЛАГ на разделе НОМЕР
  toggle [НОМЕР [ФЛАГ]]            переключает состояния ФЛАГА на разделе НОМЕР
  unit УСТРОЙСТВО                  установить устройство по умолчанию на
        УСТРОЙСТВО
  version                                  отображает текущую версию GNU Parted
        и информацию о лицензии
(parted)
```

Рис. 9.8. Команда `help`

Примечание

Помнить формат каждой команды необязательно. Если вы введете команду, но не укажете ее параметры, то они будут запрошены.

Таблица 9.3. *Флаги разделов*

Флаг	Тип раздела	Описание
`boot`	Mac, msdos, pc98	Флаг загрузочного раздела. Нужен для некоторых операционных систем
`lba`	msdos	Нужен для MS DOS, MS Windows 9*x* и MS Windows ME, чтобы эти ОС использовали для раздела режим линейной адресации (LBA)
`swap`	Mac	Устанавливается, если раздел является разделом подкачки Linux
`root`	Mac	Устанавливается, если раздел является корневым разделом Linux
`raid`	msdos	Раздел используется в RAID-массиве
`lvm`	msdos	Раздел используется как физический том в LVM
`hidden`	msdos, pc98	Скрытый раздел, устанавливается, если нужно скрыть от операционных систем семейства Microsoft

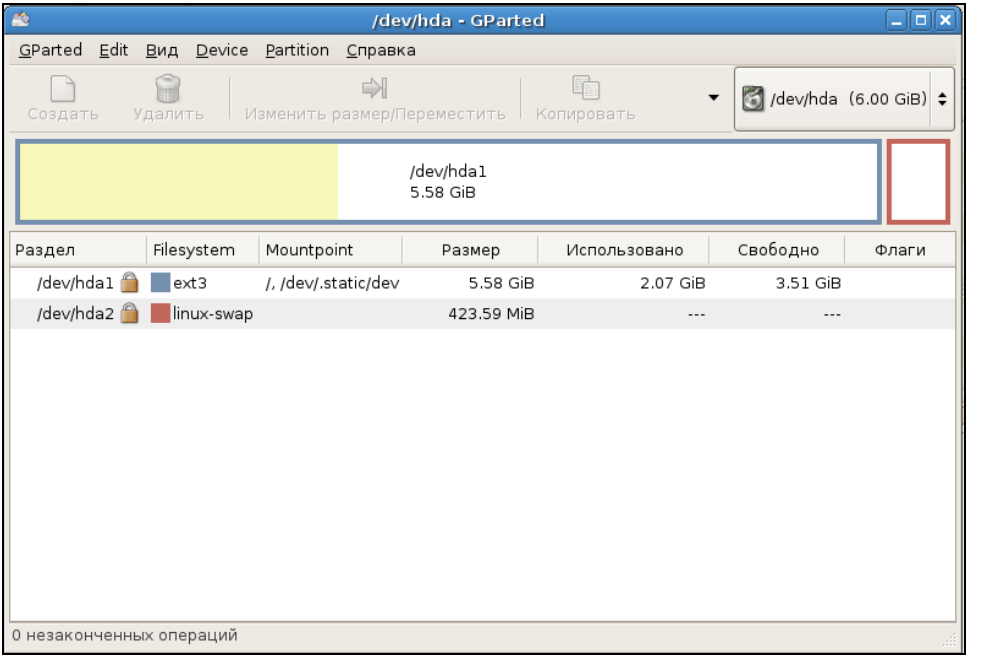


Рис. 9.9. Программа `GParted`

Как видите, программа `parted` намного более эффективна, чем `fdisk`. Любителям графического интерфейса можно порекомендовать графическую версию этой программы — `GParted` (рис. 9.9).

9.2. Информация о системе и пользователях

9.2.1. Команда *uptime*: информация о работе системы

Команда `uptime` (рис. 9.10) выводит статистическую информацию о работе системы: сколько времени прошло с момента последней перезагрузки (собственно, это и есть время "uptime"), сколько пользователей в данный момент подключено к системе и среднюю загрузку системы за последние 1, 5 и 15 минут.

```
[den@localhost ~]$ uptime
 13:02:26 up 6 min,  3 users,  load average: 0.57, 0.81, 0.44
[den@localhost ~]$
```

Рис. 9.10. Команда `uptime`

9.2.2. Команда *users*: информация о пользователях

Команда выводит информацию о пользователях, подключенных к системе в данный момент. На рис. 9.11 видно, что пользователь den подключился к системе двумя способами: вошел в консоли и в графическом режиме (или по FTP, ssh, telnet — способы подключения к системе могут быть разные).

```
[den@localhost ~]$ users
den den
[den@localhost ~]$
```

Рис. 9.11. Команда `users`

9.2.3. Команды *w*, *who*, *ftpwho* и *whoami*: информация о пользователях

Три родственные команды `w`, `who` и `whoami` выводят следующую информацию (рис. 9.12):

- команда `w` — список пользователей, подключенных к системе; виртуальный терминал, с которого работает пользователь; время входа в систему для каждого пользователя, статистику использования системы (IDLE — время простоя, JCPU — использование процессора), выполняемые каждым пользователем задачи;

- команда `who` — список пользователей, подключенных к системе; время и дату входа каждого пользователя;
- команда `whoami` — имя пользователя, который ввел команду.

```
[den@localhost ~]$ w
 13:04:08 up 7 min,  3 users,  load average: 0,18, 0,60, 0,41
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
den      pts/0     12:59    4:54   0.00s  1.14s kded [kdeinit] --new-startup
den      pts/1     13:02    0.00s  0.15s  0.01s w
[den@localhost ~]$ who
den      pts/0        2007-07-23 12:59
den      pts/1        2007-07-23 13:02
[den@localhost ~]$ whoami
den
[den@localhost ~]$
```

Рис. 9.12. Команды `w`, `who` и `whoami`

Команда `ftpwho` похожа на команду `who`, но выводит только пользователей FTP-сервера, подключенных в данный момент к серверу.

9.3. Планировщик *at*

9.3.1. Команда *at*: добавление задания

Иногда нужно просто выполнить определенные команды в определенное время (однократно), поэтому редактировать для этого таблицу crontab не совсем уместно. Такую задачу можно решить более рационально. Убедитесь, что у вас установлен и запущен демон `atd`. После этого введите команду:

```
at <время> [дата]
```

Затем просто вводите команды, которые вы хотите выполнить в указанное время. Для завершения ввода нажмите комбинацию клавиш <Ctrl>+<D>. Время указывается в AM/PM-формате — например, если вам нужно выполнить команды в 14:00, то вы должны ввести команду: `at 2pm`.

9.3.2. Команды *atq* и *atrm*: очередь заданий и удаление задания

Просмотреть очередь заданий можно командой `atq`, а удалить какое-либо задание — командой `atrm`.

На рис. 9.13 изображено добавление команды в очередь `atd`, просмотр очереди, удаление задачи и повторный просмотр очереди.

В целях повышения безопасности в файл /etc/at.deny можно добавить команды, которые запрещены для выполнения планировщиком `at`.

```
[den@localhost ~]$ at 2pm
warning: commands will be executed using (in order) a) $SHELL b) login shell c)
/bin/sh
at> mc
at> <EOT>
job 1 at 2007-07-17 14:00
[den@localhost ~]$ atq
1       2007-07-17 14:00 a den
[den@localhost ~]$ atrm 1
[den@localhost ~]$ atq
[den@localhost ~]$
```

Рис. 9.13. Использование `atd`

9.4. Планировщик *crond*

В Linux есть специальный демон `crond`, позволяющий выполнять программы по расписанию. Откройте конфигурационный файл демона `crond` — /etc/crontab (листинг 9.1).

Листинг 9.1. Пример файла /etc/crontab

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root nice -n 19 run-parts --report /etc/cron.hourly
02 4 * * * root nice -n 19 run-parts --report /etc/cron.daily
22 4 * * 0 root nice -n 19 run-parts --report /etc/cron.weekly
42 4 1 * * root nice -n 19 run-parts --report /etc/cron.monthly
```

Параметр `SHELL` задает имя программы-оболочки, параметр `PATH` — путь поиска программ, `MAILTO` — имя пользователя, которому будет отправлен отчет о выполнении расписания, а `HOME` — домашний каталог crond.

Но самое главное — не эти параметры, а сама таблица расписаний, занимающая в нашем случае последние четыре строки листинга. Согласно этой таблице, каждый час будут выполняться программы, из каталога /etc/cron.hourly, каждый день — из каталога /etc/cron.daily, каждую неделю — из каталога /etc/cron.weekly и раз в месяц — из каталога /etc/cron.monthly.

Предположим, вам нужно каждый день выполнять команду `update_av ftp://server.ru/bases/`. В каталоге /etc/cron.daily создайте файл update_av следующего содержания:

```
#!/bin/bash
update_av ftp://server.ru/bases/
```

Этот файл представляет собой небольшой bash-сценарий (сценарий командного интерпретатора). Теперь сделаем его исполнимым:

```
# chmod +x update_av
```

Правда, удобно?

Но иногда нам бывает нужно создать более гибкое расписание. Например, мы хотим, чтобы одна программа выполнялась в 7:00, а другая в 7:20. Тут простым добавлением сценария в каталог /etc/cron.daily уже не отделаешься. Чтобы создать такое расписание, вам придется изучить формат записей таблицы расписаний:

```
минуты (0–59) часы (0–23) день (1–31) месяц (1–12) день_недели (0–6,
0 — Вс) команда
```

Чтобы реализовать наше расписание, следует добавить в файл /etc/crontab следующие строки:

```
0   7      *      *      *      /usr/bin/command1 arguments
20  7      *      *      *      /usr/bin/command2 arguments
```

Первая команда будет запускаться каждый день в 7 часов утра, а вторая — тоже каждый день, но в 7:20.

Зная формат файла crontab, мы можем отредактировать стандартную таблицу расписаний *(см. листинг 9.1)*. Обратите внимание: команды, выполняемые ежедневно, будут запускаться в 4 часа утра. Это, конечно, удобно, но они не будут выполнены, если вы выключаете сервер на ночь. Поэтому давайте установим другое время, например 8 часов утра:

```
02 8 * * * root nice -n 19 run-parts --report /etc/cron.daily
```

Аналогичная ситуация и с еженедельным запуском. Программы будут запущены не только в 4:22 утра, но еще и в воскресенье. Однако на выходные вы точно выключаете свой сервер (впрочем, это зависит от политики организации — в некоторых организациях на выходные все компьютеры и не выключают). Поэтому целесообразно назначить запуск на понедельник в 8 часов 22 минуты:

```
22 8 * * 1 root nice -n 19 run-parts --report /etc/cron.weekly
```

С ежемесячным запуском вроде бы все нормально — программы будут выполняться в 4:42 первого числа каждого месяца. Хотя время лучше изменить на 8:42:

```
42 8 1 * * root nice -n 19 run-parts --report /etc/cron.monthly
```

9.5. Планировщик *anacron*

Планировщик `anacron` — непосредственный родственник `crond`, дальнейшее его развитие. Главное преимущество `anacron` заключается в том, что он, в отличие от `crond`, учитывает время, когда компьютер был выключен. Планировщик `crond` родом из UNIX, а эта операционная система устанавливалась только на серверах, которые всегда включены. Предположим, что вам нужно каждый понедельник в 7 часов утра рассылать некоторую информацию вашим сотрудникам. Вы настроили `crond` так, чтобы он запускал сценарий отправки сообщений каждый понедельник в 7 утра. Но вот беда — в 6 часов утра выключили электричество, а вклю-

чили его, скажем, в 7:20. Но 7:20 — это не 7:00, следовательно, `crond` не выполнит задание по отправке сообщений, а ваши сотрудники не получат важную информацию.

`Anacron` работает не так. Если он обнаружил, что некоторые задания не выполнены по тем или иным причинам (выключение электричества, перезагрузка компьютера), он обязательно выполнит их. Поэтому ваши сотрудники получат информацию, но с небольшой задержкой. Все же лучше, чем получить важную информацию лишь в следующий понедельник.

Но и у `anacron` есть свои недостатки. В частности, пользователи не могут создавать свои собственные расписания, а файл /etc/anacrontab может редактировать только root. К тому же более старый `crond` является более гибким в настройке — например, вы можете точно указать часы и минуты, а в случае с `anacron` можно указать только период, когда будет выполнена команда.

Формат файла /etc/anacrontab выглядит так:

```
Период          Задержка          ID          Команда
```

Например:

```
1   5           cron.daily        run-parts /etc/cron.daily
7   10          cron.weekly       run-parts /etc/cron.weekly
30  75          cron.monthly      run-parts /etc/cron.monthly
```

9.6. Команда *date:* вывод и установка даты и времени

Команда `date` используется для вывода текущей даты. Эта команда может применяться также для установки даты, если запущена от имени администратора.

Пример использования:

```
$ date
# date 1609171707
```

Первая команда выводит дату, а вторая — устанавливает дату (при условии, что команда запущена от имени root) 16 сентября (`1609`) 2007 года (`07`) и время 17:17. Как видите, установка даты осуществляется в формате MMddhhmmYY (MM — месяц, dd — число, hh — часы, mm — минуты, YY — год).

Команда `date` может вывести дату в указанном вами формате. В *главе 3* мы подробно рассмотрели пользовательское использование команды `date`, а именно форматированный вывод даты и времени.

9.7. Команды *free* и *df*: информация о системных ресурсах

Команда `free` выводит информацию об использовании оперативной и виртуальной памяти, а `df` — об использовании дискового пространства. На рис. 9.14 видно, что в системе установлено всего 384 Мбайт ОЗУ, из них 247 Мбайт занято

и 138 Мбайт — свободно. На жестком диске /dev/sda1 всего 2,8 Гбайт дискового пространства, из них свободно — 1,66 Гбайт.

```
[root@localhost den]# free
             total       used       free     shared    buffers     cached
Mem:        385628     247604     138024          0      30584     116056
-/+ buffers/cache:     100964     284664
Swap:       168640          0     168640
[root@localhost den]# df
Файловая система      Разм  Исп  Дост  Исп% смонтирована на
/dev/sda1             2,8G  1,1G  1,6G  42% /
[root@localhost den]#
```

Рис. 9.14. Команды `free` и `df`

9.8. Команда *md5sum*: вычисление контрольного кода MD5

Для проверки подлинности некоторых файлов, передаваемых через Интернет, используется алгоритм MD5 (точнее, контрольный код, вычисленный с использованием этого алгоритма). Разработчик программы выкладывает в Интернете пакет с этой программой и на своем сайте публикует контрольный код. Вы скачиваете пакет и вычисляете его контрольный код. Если коды отличаются, то файл при передаче был поврежден (или это другая версия пакета, которая, возможно, была подложена злоумышленником с целью ввода вражеского кода в вашу систему).

Использовать программу нужно так:

```
md5sum файл
```

9.9. Команда *ssh*: удаленный вход в систему

Раньше для организации удаленного доступа к консоли сервера использовался протокол Telnet. В каждой сетевой операционной системе, будь то FreeBSD или Windows 95 (которую, впрочем, сложно назвать сетевой), есть telnet-клиент. Данная программа так и называется — `telnet` (в Windows — telnet.exe).

После подключения с помощью `telnet` к удаленному компьютеру вы можете работать с ним как обычно. В окне telnet-клиента вы увидите как бы консоль удаленного компьютера: вы будете вводить команды и получать результат их выполнения — все так, как если бы вы работали непосредственно за удаленным компьютером.

Но технологии не стоят на месте, и протокол Telnet устарел. Сейчас ним практически никто не пользуется. На его смену пришел SSH (Secure Shell). SSH, как видно из названия, представляет собой безопасную оболочку. Главное отличие от Telnet состоит в том, что все данные (включая пароли доступа к удаленному компьютеру, передаваемые по SSH файлы) передаются в зашифрованном виде. Во времена Telnet участились случаи перехвата паролей и другой важной информации, что и стало причиной создания SSH.

SSH использует следующие алгоритмы для шифрования передаваемых данных: BlowFish, 3DES (Data Encryption Standard), IDEA (International Data Encryption Algorithm) и RSA (Rivest-Shamir-Adelman algorithm). Самыми надежными являются алгоритмы IDEA и RSA. Поэтому, если вы передаете действительно конфиденциальные данные, лучше использовать один из этих алгоритмов.

В состав любого дистрибутива Linux входит ssh-сервер (программа, которая и обеспечивает удаленный доступ к компьютеру, на котором она установлена) и ssh-клиент (программа, позволяющая подключаться к ssh-серверу). Для установки SSH-сервера нужно установить пакет openssh (это разновидность SSH-сервера), а для установки SSH-клиента — пакет openssh-clients.

Работать с ssh-клиентом очень просто. Для подключения к удаленному компьютеру введите команду:

```
ssh [опции] <адрес_удаленного_компьютера>
```

В качестве адреса можно указать как IP-адрес, так и доменное имя компьютера. Часто используемые опции программы `ssh` приведены в табл. 9.4.

Таблица 9.4. *Опции программы* `ssh`

Опция	Описание
`-c blowfish|3des|des`	Используется для выбора алгоритма шифрования при условии, что используется первая версия протокола SSH (об этом позже). Можно указать `blowfish`, `des` или `3des`
`-c шифр`	Задает список шифров, разделенных запятыми в порядке предпочтения. Опция используется для второй версии SSH. Можно указать `blowfish`, `twofish`, `arcfour`, `cast`, `des` и `3des`
`-f`	Переводит `ssh` в фоновый режим после аутентификации пользователя. Рекомендуется использовать для запуска программы X11. Например: `ssh -f server xterm`
`-l имя_пользователя`	Указывает имя пользователя, от имени которого нужно зарегистрироваться на удаленном компьютере. Опцию использовать не обязательно, поскольку удаленный компьютер и так запросит имя пользователя и пароль
`-p порт`	Определяет порт ssh-сервера (по умолчанию используется порт 22)
`-q`	"Тихий режим". Будут отображаться только сообщения о фатальных ошибках. Все прочие предупреждающие сообщения в стандартный выходной поток выводиться не будут
`-x`	Отключает перенаправление X11
`-X`	Задействовать перенаправление X11. Полезна при запуске X11-программ
`-1`	Использовать только первую версию протокола SSH
`-2`	Использовать только вторую версию протокола SSH. Вторая версия протокола более безопасна, поэтому при настройке SSH-сервера нужно использовать именно ее

9.10. Устройства и драйверы

При установке нового устройства у вас могут возникнуть некоторые проблемы. Вот список команд, позволяющих выяснить причину проблемы:

- `uname -a` — получить версию ядра, при установке настоящих Linux-драйверов (а не Windows-драйверов через ndiswrapper) нужно, чтобы *модуль* (так в Linux называются драйверы) был откомпилирован для соответствующей версии ядра;
- `lspci`, `lsusb`, `lshw` — помогают идентифицировать ваше устройство, выводят соответственно список PCI-устройств, список USB-устройств и список оборудования компьютера;
- `lsmod` — выводит список загруженных модулей (драйверов устройств).

Следующие команды пригодятся для настройки сетевых интерфейсов, в том числе и беспроводных:

- `iwconfig` — просмотреть информацию обо всех беспроводных интерфейсах;
- `iwlist scan` — найти беспроводные сети;
- `sudo dhclient wlan0` — обновить IP-адрес и другие сетевые параметры интерфейса `wlan0` (имя может быть другим), предварительно получив их от DHCP-сервера;
- `route` — просмотр и изменение таблицы маршрутизации;
- `sudo /etc/init.d/networking restart` — перезапуск сети;
- `dmesg | less` — просмотреть сообщения ядра;
- `sudo killall NetworkManager` — остановить NetworkManager;
- `iwevent` — просмотреть события беспроводной сети;
- `sudo /etc/init.d/dbus restart` — перезапустить все сетевые демоны.

Рассмотрим еще несколько программ. Программа `badblocks` позволяет проверить жесткий диск на наличие сбойных блоков. Проверка выполняется очень просто:

```
# badblocks -v <имя_устройства>
```

Например:

```
# badblocks -v /dev/hda
```

Параметр `-v` включает подробный режим работы, о каждом действии программа `badblocks` будет выводить отчет. Желательно выполнять программу в однопользовательском режиме, чтобы ее работе ничто не мешало.

Иногда мы произносим слово "глючит", даже не задумываясь о его смысле. Например, "глючит" оперативка. А что же это такое — "глюк"? Это неисправность устройства, проявляющаяся при определенных условиях или в определенных режимах его работы.

Конечно, "глюки" бывают не только аппаратными, но и программными (сбои программ, происходящие при определенных условиях). Довольно часто программные сбои происходят именно из-за мелких неисправностей аппаратуры. Например, причиной неожиданной ошибки при компиляции ядра может стать неисправный

модуль оперативной памяти. Ведь не секрет, что ОЗУ имеет модульную структуру. Возможно, что при обычной работе с системой один из модулей (неисправный) не задействован, поскольку его ресурсы не востребованы. Но когда система использует свои ресурсы на все 100%, происходит сбой.

Для тестирования оперативной памяти предназначена программа `memtest86`, которая работает по-особому. После ее установки создается специальный загрузочный образ, а для запуска программы вы должны перезапустить компьютер и выбрать этот образ из меню загрузчика. Но обо всем по порядку.

Установите пакет `memtest86`. Сразу после установки пакета выполните команду `memtest-setup`, которая настроит загрузочный модуль.

Вот теперь начинается самое интересное. Перезагрузите систему и выберите из меню загрузчика только что созданную программой `memtest-setup` запись. Запустится программа `memtest86`, которая сразу начнет тестировать вашу оперативную память, и если произойдет ошибка, программа непременно сообщит вам об этом.

С помощью программы `hdparm` можно выполнить тонкую настройку винчестера, что иногда позволяет существенно повысить его производительность. Раньше, скажем, года три назад, об этой программе много говорили, поскольку разработчики дистрибутивов устанавливали отказоустойчивые параметры винчестера, при которых он работал очень медленно. За пару минут можно было "разогнать" винчестер, но это не ускорение в прямом смысле слова, а просто использование всех возможностей. Обычно после установки дистрибутива для всех жестких дисков задавались минимальные параметры, обеспечивающие безотказную работу. Сейчас ситуация изменилась: после установки дистрибутива параметры жесткого диска включаются по максимуму и необходимость в hdparm практически отпала. Сейчас эта программа необходима в двух ситуациях:

1. Когда вы хотите просто узнать скорость работы винчестера.
2. Когда нужно понизить скорость привода CDROM. Это необходимо, когда вам нужно прочитать диск с дефектами поверхности (например, царапинами), — на скорости 4x вероятность чтения данных с такого диска существенно выше, чем на 24x (но этот способ не всегда приводит к желаемому результату, от работы с дефектными компакт-дисками вообще лучше воздержаться).

   Тестирование производительности жесткого диска выполняется командой:

   ```
   # hdparm -t /dev/sda
   ```

   Установка скорости 4x для привода CDROM:

   ```
   # hdparm -E 4 /dev/sr0
   ```

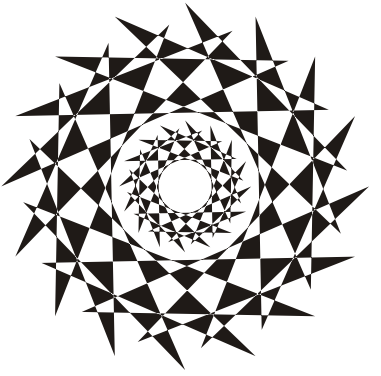
Кроме рассмотренных `hdparm`, `fsck`, `badblocks` и `memtest86` в вашем распоряжении имеются программы, приведенные в табл. 9.5.

Таблица. 9.5. *Утилиты для тестирования оборудования*

Программа	Описание
`hddtemp`	Выводит температуру винчестера (нужно заметить, что эта программа работает далеко не со всеми жесткими дисками)

Таблица. 9.5 *(окончание)*

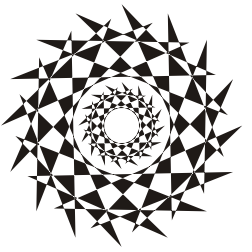
Программа	Описание
`smartmontools`	Мониторинг SMART-устройств. Как правило, это жесткие диски с возможностью самодиагностики. Такой диск сам "скажет", сколько ему осталось, чтобы его отказ не стал для вас неприятной неожиданностью
`bonnie`	Тестирование производительности винчестера
`cpuburn`	Тестирование процессора. выполняет так называемый стресс-тест процессора. Суть его заключается в том, что программа загружает ваш процессор на все 100% и продолжает работу в таком режиме. При наличии неполадок в системе, например нагрева процессора из-за плохой вентиляции, это обязательно проявится во время стресс-теста в виде сбоя
`screentest`	Тестирование/настройка монитора



ЧАСТЬ II

Операционная система

В *части II* мы рассмотрим загрузчики Linux, системы инициализации, научимся управлять пользователями и компилировать ядро Linux.



Глава 10

Загрузчики Linux

10.1. Основные загрузчики

Основное назначение загрузчика — запуск выбранной пользователем операционной системы. Наиболее популярным загрузчиком сегодня является GRUB, который мы здесь подробно рассмотрим. В более старых дистрибутивах по умолчанию применялся загрузчик LILO. Списывать со счета LILO пока нельзя, поскольку еще много систем используют именно его, да и в современных дистрибутивах есть возможность установить старый добрый LILO. Многие администраторы по привычке ставят LILO вместо более современного GRUB. Однако в данной книге загрузчик LILO рассмотрен не будет. Если он вам нужен, то рекомендую прочитать мою книгу "Linux. От новичка к профессионалу. 2-е изд.":

http://bhv.ru/books/book.php?id=186944

Кроме LILO и GRUB некоторые дистрибутивы могут включать собственные загрузчики — например, в ASPLinux таковым является ASPLoader. Подобные загрузчики мы рассматривать не будем, поскольку в большинстве случаев в дистрибутивах, использующих собственные загрузчики, имеется возможность установки GRUB или LILO.

Загрузчик GRUB (GRand Unified Bootloader) считается более гибким и современным, чем LILO. Благодаря иной схеме загрузки операционных систем GRUB "понимает" больше файловых систем, нежели LILO, а именно: FAT/FAT32, ext2, ext3, ReiserFS, XFS, BSDFS и др.

Время не стоит на месте. В свое время загрузчик GRUB пришел на смену LILO, поскольку последний не поддерживал загрузки с разделов, начинающихся после 1024-го цилиндра. Об этой проблеме знает, наверное, каждый Linux-пользователь — ведь всего несколько лет назад она была актуальной (пока все дистрибутивы не перешли на GRUB). Точно такая же участь постигла и GRUB — на его место пришел GRUB2, умеющий загружаться с файловой системы ext4. А загрузка с ext4-разделов просто необходима современному дистрибутиву.

GRUB2 — это не просто набор патчей для GRUB, а полностью новая разработка, созданная с нуля. Именно поэтому у GRUB2 совершенно другой формат конфигурационного файла.

Разработка "обычного" GRUB полностью прекращена, к нему выпускаются лишь патчи. Да, можно скачать патч, добавляющий к GRUB загрузку с разделов ext4.

Так, в Ubuntu 9.10, где по умолчанию впервые был установлен GRUB2, я его удалил (с сохранением конфигурационных файлов), затем установил GRUB (имеющаяся в составе версии 9.10 версия GRUB поддерживает ext4), создал вручную его конфигурационный файл и перезагрузил систему — она загрузилась без ошибок. Но, учитывая, что будущее все-таки за GRUB2, я вернул его обратно на заслуженное место.

Примечание

В Ubuntu GRUB2 используется, начиная с версии 9.10 — не зря я упомянул ее ранее. И в этой версии Ubuntu, и в новой — 10.04 — имеется один небольшой "глюк", связанный с установкой тайм-аута выбора операционной системы. Чуть позже мы решим эту проблему, а пока приступим к рассмотрению конфигурационных файлов GRUB2.

Примечание

На самом деле то, что называется GRUB2 — это GRUB v1.98. То есть *почти* вторая версия, а когда выйдет вторая версия (в смысле 2.0), пока никто не знает. Хотя ведущие разработчики дистрибутивов уже включили GRUB2 в состав дистрибутивов, что говорит о его надежности.

10.2. Конфигурационные файлы GRUB и GRUB2

10.2.1. Конфигурационный файл GRUB

Конфигурационным файлом GRUB служит файл /boot/grub/grub.conf (в старых версиях — /boot/grub/menu.lst, впрочем, menu.lst в новых версиях — это ссылка на grub.conf). Рассмотрим пример этого файла (листинг 10.1).

Листинг 10.1. Файл /boot/grub/grub.conf

```
# Следующие параметры будут описаны далее:
boot=/dev/hda
default=0
timeout=10
fallback=1
splashimage=(hd0,1)/grub/mysplash.xpm.gz

# По умолчанию скрывает меню (для того чтобы увидеть меню,
# нужно нажать <ESC>)
#hiddenmenu

# Главное загрузочное устройство GRUB (можно не указывать)
#groot=(hd0,1)
```

```
# Опции загрузчика по умолчанию (более подробно см. в man menu.lst)
# defoptions=quiet splash

# Опции ядра по умолчанию
# kopt=root=/dev/hda2 ro

# Предпочитаемые цвета
#color cyan/blue white/blue

title MDK
       root (hd0,1)
       kernel /vmlinuz-2.6.14-1.1263 ro root=/dev/hda2
       initrd /initrd-2.6.14-1.1263.img

title WinXP
       rootnoverify (hd0,0)
       makeactive
       chainloader+1
```

Параметр `boot` указывает загрузочное устройство, а параметр `default` — загрузочную метку по умолчанию. Метка начинается параметром `title` и продолжается до следующего `title`. Нумерация меток начинается с 0. Параметр `timeout` задает количество секунд, по истечении которых будет загружена операционная система по умолчанию.

Параметр `default` полезно использовать с параметром `fallback`. Первый задает операционную систему по умолчанию, а второй — операционную систему, которая будет загружена в случае, если с загрузкой операционной системы по умолчанию произошла ошибка.

Задать графическое изображение позволяет параметр `splashimage`. Чуть позже мы разберемся, как самостоятельно создать такое изображение.

При работе с GRUB вам поначалу будет трудно разобраться с именами разделов. GRUB вместо привычных /dev/hd* (или /dev/sd* для SCSI-дисков) использует свои имена. Перевести имя /dev/hd* в имя в формате GRUB просто. Во-первых, опускается /dev/. Во-вторых, устройства отсчитываются не с буквы "a", как в Linux, а с нуля. Разделы на дисках отсчитываются не с единицы, а тоже с нуля, причем номер раздела указывается через запятую. Потом все имя берется в скобки. Например, раздел /dev/hda1 в GRUB будет выглядеть как (hd0,0), а раздел /dev/hdb2 как (hd1,1). Впрочем, об именах разделов в GRUB мы еще поговорим, но чуть позже.

Параметр `rootnoverify` указывается для Windows (точнее, для всех операционных систем не типа Linux). Параметр `chainloader` указывается для операционных систем, поддерживающих цепочечную загрузку. Если Windows на вашем компьютере установлен в неактивном разделе, с которого Windows загружаться не может, перед параметром `chainloader` нужно указать параметр `makeactive`.

10.2.2. Конфигурационный файл GRUB2

В листинге 10.2 приведен основной конфигурационный файл GRUB2 — /boot/grub/grub.cfg. Этот конфигурационный файл не редактируется вручную. Для его создания используется утилита /usr/sbin/grub-mkconfig, которая генерирует этот конфигурационный файл на основе шаблонов, хранящихся в каталоге /etc/grub.d, и настроек из файла /etc/default/grub.

Листинг 10.2. Конфигурационный файл grub.cfg

```
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by /usr/sbin/grub-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#

### BEGIN /etc/grub.d/00_header ###
if [ -s /boot/grub/grubenv ]; then
  have_grubenv=true
  load_env
fi
set default="0"
if [ ${prev_saved_entry} ]; then
  saved_entry=${prev_saved_entry}
  save_env saved_entry
  prev_saved_entry=
  save_env prev_saved_entry
fi
insmod ext2
set root=(hd0,1)
search --no-floppy --fs-uuid --set 34eaa635-ef0e-4d5c-8b61-3c22c767834b
if loadfont /usr/share/grub/unicode.pf2 ; then
  set gfxmode=640x480
  insmod gfxterm
  insmod vbe
  if terminal_output gfxterm ; then true ; else
    # For backward compatibility with versions of terminal.mod that don't
    # understand terminal_output
    terminal gfxterm
  fi
fi
if [ ${recordfail} = 1 ]; then
```

```
  set timeout=-1
else
  set timeout=10
fi
### END /etc/grub.d/00_header ###

### BEGIN /etc/grub.d/05_debian_theme ###
set menu_color_normal=white/black
set menu_color_highlight=black/white
### END /etc/grub.d/05_debian_theme ###

### BEGIN /etc/grub.d/10_linux ###
menuentry "Denix, Linux 2.6.31-14-generic" {
        recordfail=1
        if [ -n ${have_grubenv} ]; then save_env recordfail; fi
       set quiet=1
       insmod ext2
       set root=(hd0,1)
       search --no-floppy --fs-uuid --set 34eaa635-ef0e-4d5c-8b61-3c22c767834b
       linux   /boot/vmlinuz-2.6.31-14-generic root=UUID=34eaa635-ef0e-4d5c-
8b61-3c22c767834b ro   quiet splash
       initrd /boot/initrd.img-2.6.31-14-generic
}
menuentry "Denix, Linux 2.6.31-14-generic (recovery mode)" {
        recordfail=1
        if [ -n ${have_grubenv} ]; then save_env recordfail; fi
       insmod ext2
       set root=(hd0,1)
       search --no-floppy --fs-uuid --set 34eaa635-ef0e-4d5c-8b61-3c22c767834b
       linux   /boot/vmlinuz-2.6.31-14-generic root=UUID=34eaa635-ef0e-4d5c-
8b61-3c22c767834b ro single
       initrd /boot/initrd.img-2.6.31-14-generic
}
### END /etc/grub.d/10_linux ###

### BEGIN /etc/grub.d/20_memtest86+ ###
menuentry "Memory test (memtest86+)" {
       linux16/boot/memtest86+.bin
}
menuentry "Memory test (memtest86+, serial console 115200)" {
       linux16/boot/memtest86+.bin console=ttyS0,115200n8
}
```

```
### END /etc/grub.d/20_memtest86+ ###

### BEGIN /etc/grub.d/30_os-prober ###
if [ ${timeout} != -1 ]; then
  if keystatus; then
    if keystatus --shift; then
      set timeout=-1
    else
      set timeout=0
    fi
  else
    if sleep --interruptible 3 ; then
      set timeout=0
    fi
  fi
fi
### END /etc/grub.d/30_os-prober ###

### BEGIN /etc/grub.d/40_custom ###
# This file provides an easy way to add custom menu entries. Simply type the
# menu entries you want to add after this comment. Be careful not to change
# the 'exec tail' line above.
### END /etc/grub.d/40_custom ###
```

Вы наверняка заметили, что синтаксис grub.cfg весьма напоминает синтаксис bash-сценариев. Параметры GRUB2 задаются в файле /etc/default/grub, а в файле grub.cfg описываются элементы меню загрузчика.

Рассмотрим описание элемента меню:

```
menuentry "Denix, Linux 2.6.31-14-generic" {
        recordfail=1
        if [ -n ${have_grubenv} ]; then save_env recordfail; fi
    set quiet=1
    insmod ext2
    set root=(hd0,1)
    search --no-floppy --fs-uuid --set 34eaa635-ef0e-4d5c-8b61-3c22c767834b
    linux  /boot/vmlinuz-2.6.31-14-generic root=UUID=34eaa635-ef0e-4d5c-
8b61-3c22c767834b ro   quiet splash
    initrd /boot/initrd.img-2.6.31-14-generic
}
```

В кавычках после `menuentry` находится описание элемента меню — можете заменить этот текст на все, что вам больше нравится. Далее следуют команды GRUB. Например, команда `insmod ext2` загружает модуль ext2. Это не модуль ядра Linux! Это модуль GRUB2 — файл ext2.mod, находящийся в каталоге /boot/grub.

Команда `set root` устанавливает загрузочное устройство. Формат имени устройства такой же, как в случае с GRUB2.

После служебного слова `linux` задается ядро (файл ядра) и параметры, которые будут переданы ядру. Служебное слово `initrd` указывает файл initrd.

Теперь рассмотрим файл /etc/default/grub, содержащий параметры GRUB2 (листинг 10.3). Поскольку этот файл вы будете редактировать чаще, чем grub.cfg, то комментарии для большего удобства я перевел на русский язык.

Листинг 10.3. Файл /etc/default/grub

```
# Если вы измените этот файл, введите команду 'update-grub'
# для обновления вашего файла /boot/grub/grub.cfg.

# Элемент по умолчанию, нумерация начинается с 0
GRUB_DEFAULT=0
# Чтобы увидеть меню GRUB, нужно или закомментировать следующую
# опцию или установить значение больше 0, но в этом случае
# нужно изменить значение GRUB_HIDDEN_TIMEOUT_QUIET на false
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
# Таймаут (в секундах)
GRUB_TIMEOUT="10"
# Название дистрибутива — вывод команды lsb_release или просто Debian
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
# Параметры ядра по умолчанию
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
GRUB_CMDLINE_LINUX=""

# Раскомментируйте для отключения графического терминала
# (только для grub-pc)
#GRUB_TERMINAL=console

# Разрешение графического терминала
#GRUB_GFXMODE=640x480

# Раскомментируйте следующую опцию, если вы не хотите передавать
# параметр "root=UUID=xxx" ядру Linux
#GRUB_DISABLE_LINUX_UUID=true

# Раскомментируйте, если нужно отключить генерацию элемента меню
# режима восстановления
#GRUB_DISABLE_LINUX_RECOVERY="true"
```

После изменения файла /etc/default/grub не забудьте запустить команду `update-grub` для обновления вашего /boot/grub/grub.cfg.

При редактировании конфигурации GRUB2 нужно придерживаться одной стратегии из двух возможных. Первая заключается в ручном редактировании файла grub.cfg — вы редактируете его вручную и больше не используете других программ вроде `grub-mkconfig` или `update-grub`. Вторая стратегия заключается в использовании вспомогательных программ, но тогда не нужно редактировать файл grub.cfg вручную, иначе при последующем изменении файла grub.cfg программами `grub-mkconfig` или `update-grub` все изменения, внесенные вручную, будут уничтожены.

По умолчанию команда `grub-mkconfig` генерирует конфигурационный файл на консоль, поэтому вызывать ее нужно так:

```
sudo grub-mkconfig > /boot/grub/grub.cfg
```

10.3. Команды установки загрузчиков

Установить GRUB/GRUB2, если вы это еще не сделали, можно следующей командой:

```
/sbin/grub-install <устройство>
```

Например:

```
/sbin/grub-install /dev/sda
```

После изменения конфигурационного файла переустанавливать загрузчик, как в случае с устаревшим LILO, не нужно.

10.4. Установка тайм-аута выбора операционной системы. Редактирование параметров ядра Linux

По умолчанию GRUB2 не отображает меню выбора операционной системы. Следовательно, вы не можете ни выбрать другую операционную систему (в том числе и Windows), ни изменить параметры ядра Linux, ни выбрать режим восстановления или режим тестирования памяти. Одним словом, такое поведение загрузчика создает определенные неудобства.

Чуть ранее было сказано, что для установки тайм-аута загрузчика нужно отредактировать следующие параметры:

```
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
# Таймаут (в секундах)
GRUB_TIMEOUT="10"
```

Все правильно, но если бы GRUB2 в Ubuntu был без "глюка". Вообще, "глюки" — это хорошо. Чем корявее будет Canonical делать свои дистрибутивы, тем больше будет работы у авторов книг и дистрибутивов на базе Ubuntu. Вы думаете,

почему я создал свой дистрибутив Denix (**denix.dkws.org.ua**)? Нет, не для того, чтобы гордо ткнуть себя в грудь (мол, я тоже могу сделать свой дистрибутив!). А для того, чтобы после каждой установки Ubuntu пользователи могли не тратить свое личное время, часами настраивая операционную систему.

Например, чтобы побороть такое неадекватное поведение (а каким его еще назвать, если программа не реагирует на установку параметров из конфигурационного файла?) загрузчика, мне пришлось потратить минут 15–20. К своему решению я пришел методом эксперимента, поэтому я не удивлюсь, если на каком-то форуме в Интернете вы найдете другое решение (не исключаю, может быть, даже лучшее).

Итак, откройте ваш файл /etc/grub.d/30_os-prober:

```
sudo nano /etc/grub.d/30_os-prober
```

Найдите в нем строку:

```
if [ "x${GRUB_HIDDEN_TIMEOUT}" = "x0" ]
```

Далее все значения -1 во фрагменте кода, представленном в листинге 10.4, замените на 1. Строки, которые нуждаются в редактировании, выделены полужирным. Изменять значение -1 в остальном коде, выходящем за рамки листинга 10.4, не нужно!

Листинг 10.4. Фрагмент файла /etc/grub.d/30_os-prober

```
if [ "x${GRUB_HIDDEN_TIMEOUT}" = "x0" ] ; then
    cat <
if [ \${timeout} != 1 ]; then
  if keystatus; then
    if keystatus --shift; then
      set timeout=1
    else
      set timeout=0
    fi
  else
    if sleep$verbose --interruptible 3 ; then
      set timeout=0
    fi
  fi
fi
EOF
      else
    cat << EOF
if [ \${timeout} != 1 ]; then
  if sleep$verbose --interruptible ${GRUB_HIDDEN_TIMEOUT} ; then
```

```
    set timeout=0
  fi
fi
EOF
```

После этого сохраните файл и введите команды:

```
sudo grub-mkconfig > /boot/grub/grub.cfg
sudo update-grub
sudo reboot
```

Да, после перезагрузки вы увидите меню GRUB2 (рис. 10.1). Для редактирования параметров ядра *(см. главу 13)*, которые передаются Linux, выделите загрузочную метку Linux и нажмите клавишу <e>. Если вы защитили GRUB от редактирования параметров ядра (как это сделать, будет показано в *разд. 10.10)*, вы увидите требование ввести имя пользователя и пароль (рис. 10.2). Если они правильные, вы сможете отредактировать загрузочную метку (рис. 10.3). В данном случае дополнительные параметры нужно вводить после параметра `splash` (строка параметров начинается после служебного слова `linux`). Кстати, если у вас проблемы с запуском Linux, то, чтобы увидеть больше диагностических сообщений, параметры `quiet` и `splash` лучше вообще удалить. Для возврата обратно в меню GRUB2, нажмите клавишу <Esc>, а для загрузки выбранной операционной системы нажмите клавиши <Ctrl>+<X>.

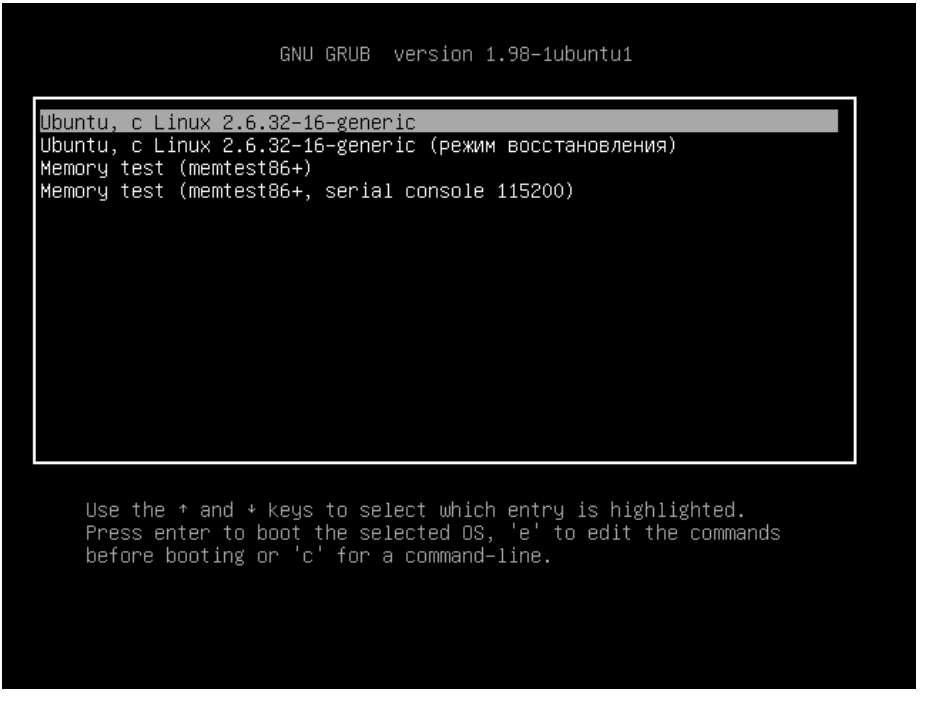


Рис. 10.1. Меню загрузчика

Рис. 10.2. Ввод имени пользователя и пароля

Рис. 10.3. Редактирование загрузочной метки

10.5. Установка собственного фона загрузчика GRUB и GRUB2

Вы хотите создать собственный фон для загрузчика GRUB? Это очень просто. Создайте или найдите в Интернете понравившуюся вам картинку. Уменьшите ее до размера 640×480 и конвертируйте в формат XPM. Все это можно сделать одной командой:

```
# convert image.jpg -colors 14 -resize 640x480 image.xpm
```

Затем сожмите картинку с помощью команды `gzip`:

```
# gzip image.xpm
```

Скопируйте сжатую картинку в каталог /boot/grub и пропишите в конфигурационном файле /boot/grub/grub.conf:

```
splashimage=(hd0,1)/grub/image.xpm.gz
```

Теперь разберемся, как установить графический фон в GRUB2. Убедитесь, что установлен пакет grub2-splashimages. Этот пакет содержит графические заставки для GRUB2, которые будут установлены в каталог /usr/share/images/grub. Если вам не нравятся стандартные картинки, то множество фонов для GRUB2 вы можете скачать с сайта **http://www.gnome-look.org/** или создать вручную, как было показано ранее. Вот только GRUB2 уже поддерживает форматы PNG и TGA, поэтому конвертировать в формат XPM не нужно. Будем считать, что картинка у нас уже выбрана. Осталось только установить ее как фон.

Откройте файл темы GRUB2. Он находится в каталоге /etc/grub.d. В Ubuntu и Debian он называется /etc/grub.d/05_debian_theme. В другие дистрибутивах (учитывая, что далеко не все современные дистрибутивы перешли на GRUB2, точное название не могу сказать) он может называться иначе.

Найдите в файле темы следующую строку:

```
for i in {/boot/grub,/usr/share/images/desktop-base}/
moreblue-orbit-grub.{png,tga} ; do
```

Замените ее на следующую строку:

```
for i in {/boot/grub,/usr/share/images/desktop-base,/usr/share/
images/grub}/имя_файла.{png,tga} ; do
```

Как видите, мы просто прописали выбранную вами картинку. Далее нужно обновить GRUB2:

```
sudo update-grub
```

10.6. Постоянные имена и GRUB

Как было отмечено в *главе 4*, все современные дистрибутивы перешли на так называемые постоянные ("длинные") имена. Раньше, когда еще никто не знал о длинных именах, запись в файле grub.conf могла выглядеть так:

```
kernel /boot/vmlinuz26 root=/dev/hda1 vga=0x318 ro
```

Эта запись указывает имя ядра (/boot/vmlinuz26). Все, что после него, — параметры, которые будут переданы ядру. Один из них (параметр root) — указывает имя корневой файловой системы. Здесь оно приведено еще в старом формате. Сейчас вы такие имена в grub.conf не увидите (если, конечно, сами не пропишете). Варианты указания длинных имен выглядят так:

```
root=/dev/disk/by-uuid/2d781b26-0285-421a-b9d0-d4a0d3b55680
root=/dev/disk/by-id/scsi-SATA_WDC_WD1600JB-00_WD-WCANM7959048-part5
root=LABEL=/
```

Какой вариант будет использоваться у вас, зависит от дистрибутива. Например, в Fedora применяют третий способ, а в openSUSE — второй.

10.7. Восстановление загрузчика GRUB/GRUB2

Что делать, если вы переустановили Windows, а она установила в MBR свой загрузчик, и теперь вы не можете загрузить Linux? Не переустанавливать же еще и Linux из-за такой мелочи!

Для восстановления загрузчика GRUB нужно загрузиться с LiveCD (подойдет любой LiveCD с любым дистрибутивом Linux) и ввести следующие команды:

```
mkdir /old
mkdir /old/dev
mount /dev/sdaN /old
```

> ***Примечание***
>
> Все команды нужно вводить от имени root. Для этого следует использовать команды su или sudo.
>
> В частности, в LiveCD Ubuntu нужно вводить все команды с использованием команды sudo, например, так:
>
> ```
> sudo mkdir /old
> sudo mkdir /old/dev
> …
> ```

Разберемся, что означают эти команды:

- первая из них создает каталог /old, который будет использоваться в качестве точки монтирования;
- вторая — создает в этом каталоге подкаталог dev, который пригодится для монтирования devfs — псевдофайловой системы;
- третья — используется для монтирования корневой файловой системы дистрибутива Linux, установленного на жестком диске в разделе /dev/sda*N* (где *N* — номер раздела), к каталогу /old. Предположим, что на вашем компьютере дистрибутив Linux был установлен в раздел /dev/sda5. Тогда вам нужно ввести следующую команду:

  ```
  mount /dev/sda5 /old
  ```

После этого нужно подмонтировать каталог /dev к каталогу /old/dev. Это делается с помощью все той же команды `mount`, но с параметром `--bind`:

```
mount --bind /dev /old/dev
chroot /old
```

Команда `chroot` заменяет корневую систему нашего LiveCD на корневую систему дистрибутива, установленного на винчестере. Вам остается лишь ввести команду:

```
/sbin/grub-install /dev/sda
```

Эта команда установит загрузчик GRUB так, как он был установлен до переустановки Windows. После установки GRUB нужно перезагрузить компьютер командой `reboot`.

Примечание

Дополнительную информацию о восстановлении загрузчика GRUB вы можете получить на моем форуме:

http://www.dkws.org.ua/phpbb2/viewtopic.php?t=3275

10.8. Две и более ОС Linux на одном компьютере

Рассмотрим другую ситуацию, часто возникающую на практике. Вы решили установить на свой компьютер (на котором уже была установлена Windows) операционную систему Linux. Все прошло гладко, и теперь вы с помощью GRUB можете запустить две системы — Windows и Linux. Но потом вы решили установить еще один дистрибутив Linux, однако старый удалять пока не хотите. Поэтому вы создали еще один Linux-раздел и установили в него новый дистрибутив, но после перезагрузки обнаружили небольшую проблему:

- в меню GRUB отображается только последний установленный дистрибутив и Windows, т. е. вы не можете загрузить первый дистрибутив. Так, Fedora, например, напрочь игнорирует все установленные до нее дистрибутивы, и поэтому после установки этого дистрибутива вы можете запустить только его и Windows;
- или в меню GRUB отображаются оба дистрибутива и Windows, но запустить вы можете только последний установленный дистрибутив (и, понятно, Windows). Такую картину я наблюдал после установки openSUSE — в моем загрузочном меню появилась метка для загрузки ранее установленного дистрибутива Fedora, но загрузить его не получалось.

Понятно, что восстановить загрузчик первого дистрибутива, воспользовавшись рекомендациями из предыдущего раздела, мы не можем, поскольку после этого мы сможем запустить только первый дистрибутив и Windows (на момент формирования файла grub.conf первого дистрибутива еще ничего не было известно о втором дистрибутиве, который вы недавно установили).

Наши действия будут зависеть от конкретной ситуации. Для большей определенности предположим, что первый дистрибутив был установлен в раздел /dev/sda5, а второй — в раздел /dev/sda6.

Если у вас проблема по первому случаю (когда ранее установленного дистрибутива вообще нет в загрузочном меню), тогда вам нужно примонтировать раздел первого дистрибутива (у нас это /dev/sda5) к каталогу /mnt (или к любому другому):

```
# mount /dev/sda5 /mnt
```

Затем надо открыть файл /mnt/boot/grub/grub.conf (/mnt/boot/grub/menu.lst).

> ***Внимание!***
>
> Исходя из приведенного здесь пути к файлу, мы понимаем, что открываем файл grub.conf первого дистрибутива.

Скопируйте из него метку загрузки первого дистрибутива. У меня сначала был установлен openSUSE 11.2, а потом я установил Fedora 12, поэтому загрузочная метка в моем случае выглядела так:

```
title openSUSE 11.2
    root (hd0,4)
        kernel /boot/vmlinuz-2.6.31-14-default root=/dev/disk/by-id/
    scsi-SATA_WDC_WD1600JB-00_WD-WCANM7959048-part5 vga=0x317
    resume=/dev/sda7 splash=silent showopts
    initrd /boot/initrd-2.6.31-14-default
```

> ***Совет***
>
> Обратите внимание: параметр `root` содержит постоянное (длинное) имя, поэтому его не придется изменять. Если же в вашем варианте параметр `root` содержит короткое имя вида /dev/sd*, его желательно заменить постоянным именем *(см. главу 4)*.

Скопированную загрузочную метку нужно вставить в файл /boot/grub/grub.conf — это файл конфигурации GRUB, используемый в настоящий момент. Файл сохраните, но пока не закрывайте и не перезагружайте компьютер. Обратите внимание: для загрузки нашего первого дистрибутива требуются файлы vmlinuz-2.6.22.5-31-default и initrd-2.6.22.5-31-default. Их нужно скопировать из каталога /mnt/boot в каталог /boot:

```
cp /mnt/boot/vmlinuz* /boot
cp /mnt/boot/initrd* /boot
```

Теперь можно перезагрузить компьютер. Первый дистрибутив, установленный в /dev/sda5, будет загружен.

Перейдем ко второму случаю. Он проще тем, что нам не нужно редактировать grub.conf, поскольку за нас это уже сделала программа установки второго дистрибутива. Вам нужно только подмонтировать каталог /dev/sda5 к каталогу /mnt и скопировать файлы vmlinuz* и initrd* из каталога /mnt/boot в каталог /boot. Вот и все.

Напоследок рекомендую прочитать тему форума, непосредственно относящуюся к рассматриваемому вопросу: **http://www.dkws.org.ua/phpbb2/viewtopic.php?t=3085.**

10.9. Загрузка с ISO-образов

Предположим, вы скачали ISO-образ новой версии Ubuntu, но у вас нет "болванки", чтобы записать на нее образ и загрузиться с полученного диска. Могу вас обрадовать: "болванка" вам для этого не понадобится — GRUB2 умеет использовать в качестве загрузочных устройств ISO-образы. Просто пропишите ISO-образ в конфигурационном файле GRUB2 и перезагрузите компьютер. Новая загрузочная метка появится в меню GRUB2, и, если ее выбрать, система загрузится с ISO-образа.

Итак, создайте в каталоге /boot подкаталог iso (название, сами понимаете, может быть любым), загрузите в него ISO-образ дистрибутива. Теперь вам осталось лишь отредактировать конфигурационный файл /boot/grub/grub.cfg, добавив в него вот такую загрузочную запись (выделенный полужирным шрифтом текст нужно записать в одну строку):

```
menuentry "Ubuntu LiveCD" {
    loopback loop /boot/iso/ubuntu.iso
    linux (loop)/casper/vmlinuz boot=casper iso-
scan/filename=/boot/iso/ubuntu.iso noeject noprompt --
    initrd (loop)/casper/initrd.lz
   }
```

Перезагружаемся и выбираем пункт меню **Ubuntu LiveCD**.

10.10. Установка пароля загрузчика GRUB2

Как уже отмечалось, начиная с версии 9.10 в Ubuntu используется загрузчик GRUB2 вместо обычного GRUB. По сравнению с GRUB, новый загрузчик одновременно и проще в обращении, и сложнее в настройке. Настраивать GRUB2 придется реже, но к его сложной настройке надо будет привыкать, — практически все современные дистрибутивы перешли на GRUB2.

В GRUB можно было задать общий пароль для всех загрузочных меток, а также установить пароль только на некоторые загрузочные метки. В GRUB2 можно сделать то же самое, но, кроме самого пароля, понадобится указать еще и имя пользователя, что усложняет злоумышленнику взлом системы, поскольку ему нужно будет знать не только пароль, но и имя пользователя. Защита отдельных загрузочных меток, как правило, используется редко, чаще устанавливается пароль на все метки сразу, что и будет продемонстрировано в этой главе.

Сначала установим простой (незашифрованный) пароль, а затем зашифруем его, чтобы никто не смог его прочитать, загрузившись с LiveCD. Прежде всего откройте файл /etc/grub.d/00_header:

```
sudo nano /etc/grub.d/00_header
```

В конец файла добавьте строки:

```
cat << EOF
set superusers="den"
password den 1234
EOF
```

Здесь имя пользователя — `den`, пароль — `1234`.

Теперь обновите GRUB2:

```
sudo update-grub
```

Можно также напрямую редактировать файл конфигурации GRUB2 — grub.cfg. В него следует добавить вот такие строки:

```
set superusers="user1"
password user1 password1
password user2 password2
```

Обратите внимание, что командами `password` заданы два пользователя: `user1` и `user2` с паролями `password1` и `password2` соответственно. Но пользователь `user1` является суперпользователем, т. е. может редактировать загрузочные метки GRUB2, а обычный пользователь (`user2`) может только загружать метки. Таким образом, у пользователя `user1` получится передать ядру новые параметры, а пользователь `user2` сможет только загрузить Linux с параметрами по умолчанию.

Можно даже задать условие, что метку Windows будет загружать только пользователь `user2`:

```
menuentry "Windows" --users user2 {
        set root=(hd0,2)
        chainloader +1
}
```

Теперь разберемся с шифрованием пароля. Команда `password` поддерживает только незашифрованные пароли. Если вы хотите использовать зашифрованные пароли, то нужно применить команду `password_pbkdf2`. Например:

```
password_pbkdf2 den зашифрованный_пароль
```

Получить зашифрованный пароль можно командой:

```
grub-mkpasswd-pbkdf2
```

После программа запросит у вас пароль и сообщит его хэш:

```
Your PBKDF2 is grub.pbkdf2.зашифрованный_пароль
```

Пример пароля:

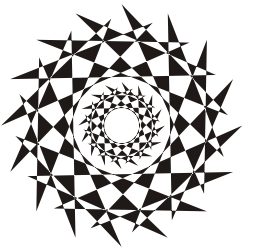
```
grub.pbkdf2.sha512.10000.9290F727ED06C38BA4549EF7DE25CF5642659211B7FC076F2D
28FEFD71784BB8D8F6FB244A8CC5C06240631B97008565A120764C0EE9C2CB0073994D79080
136.887CFF169EA8335235D8004242AA7D6187A41E3187DF0CE14E256D85ED97A97357AAA8F
F0A3871AB9EEFF458392F462F495487387F685B7472FC6C29E293F0A0
```

Весь этот хэш нужно скопировать в конфигурационный файл GRUB2:

```
password_pbkdf2 den
grub.pbkdf2.sha512.10000.9290F727ED06C38BA4549EF7DE25CF5642659211B7FC076F2D
28FEFD71784BB8D8F6FB244A8CC5C06240631B97008565A120764C0EE9C2CB0073994D79080
136.887CFF169EA8335235D8004242AA7D6187A41E3187DF0CE14E256D85ED97A97357AAA8F
F0A3871AB9EEFF458392F462F495487387F685B7472FC6C29E293F0A0
```

Если вы не использовали файл 00_header, а редактировали непосредственно файл grub.cfg, то команду `update-grub` вводить не нужно!

Дополнительную информацию вы сможете получить по адресам: **http://ubuntuguide.net/how-to-setup-boot-password-for-grub2-entries** и **http://grub.enbug.org/Authentication**.



Глава 11

Системы инициализации Linux

11.1. Начальная загрузка Linux

Давайте разберемся, как загружается Linux. В этой книге мы уже упоминали о начальной загрузке компьютера, поэтому сейчас начнем с того момента, когда загрузчик BIOS нашел загрузочное устройство, например жесткий диск. Далее загрузчик BIOS считывает первый (нулевой) сектор и передает ему управление. На этом работа загрузчика BIOS заканчивается.

В первом секторе находится главная загрузочная запись (Master Boot Record, MBR), состоящая из трех частей: первичного загрузчика, таблицы разделов диска (partition table) и флага загрузки.

Итак, из первой части MBR вызывается первичный загрузчик. Действия этого загрузчика зависят только от него самого. Предположим, что у нас установлен загрузчик LILO — намного проще рассматривать работу загрузчика на конкретном примере.

Загрузчик LILO состоит из двух частей: первая содержится в MBR, а вторая находится на диске в виде файла /boot/boot.b. Задача первой части — запуск вторичного загрузчика (второй части), который и производит дальнейшую загрузку системы. Первая часть ничего не знает о файловых системах, поэтому местонахождение второй части записано в "физических координатах", т. е. явно указаны цилиндр, головка, сектор жесткого диска.

Вторая часть загрузчика более интеллектуальная. Она уже "знает", что такое файловая система, а карта размещения файлов записана в файле /boot/map. Этот файл используется для поиска ядра и образа виртуального диска. Для чего нужен виртуальный диск? Представим, что мы еще не установили Linux, а только собираемся это сделать. Вставляем загрузочный диск, и загрузчик запускает не просто инсталлятор — на самом деле запускается операционная система Linux, ясно виден процесс загрузки ядра, и потом уже запускается программа установки. Но ядру нужно же откуда-то прочитать модули поддержки устройств и файловой системы — ведь корневая файловая система еще не создана. Вот все эти модули и находятся на виртуальном диске. Виртуальный диск загружается в память, ядро монтирует его, как обычную файловую систему, и загружает с него все необходимые модули. После этого виртуальный диск размонтируется и — в случае нормальной загрузки, а не установки Linux, — вместо него монтируется обычная корневая файловая система.

Для работы с виртуальным диском используется технология initrd (INITial RAM Disk). Файл образа виртуального диска находится в каталоге /boot и называется initrd-<версия ядра>.

11.2. Система инициализации init

В процессе запуска ядра монтируется корневая файловая система и запускается программа `init`, которая и выполняет дальнейшую инициализацию системы. Программа `init` — часть самой надежной и распространенной системы инициализации Linux, которая используется многими дистрибутивами: Fedora, ASPLinux, Mandriva, openSUSE и др.

Кроме системы инициализации init, существуют и другие системы, например initng и upstart, которые мы также рассмотрим в этой книге:

- система initng позволяет существенно ускорить запуск Linux, но она почему-то не прижилась в мире Linux и ни один дистрибутив не использует ее по умолчанию, очевидно, это из-за сложной настройки данной системы. Прочитать об этой системе можно в моей статье: **http://www.dkws.org.ua/index.php?page=show&file=a/system/initng/initng**;
- система upstart была специально разработана для дистрибутива Ubuntu Linux, но ее при желании можно установить в любом дистрибутиве.

11.2.1. Файл /etc/inittab

Итак, программа `init` читает конфигурационный файл /etc/inittab и запускает другие процессы, согласно инструкциям этого файла (листинг 11.1).

Листинг 11.1. Файл /etc/inittab

```
id:5:initdefault:

# Инициализация системы
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Что делать при нажатии CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# От UPS была получена команда, что пропало питание.
```

```
# Немного ждем и выключаем компьютер
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# От UPS получена команда, что питание возобновилось
# Отменяем shutdown
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown
Cancelled"

# Запуск gettys
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Однопользовательский режим
~~:S:wait:/bin/sh
```

Одна из главных инструкций файла /etc/inittab выглядит так:

```
id:<число>:initdefault:
```

Эта инструкция задает уровень запуска по умолчанию. Уровень запуска определяет, какие действия будут выполнены программой `init` (какие процессы будут запущены). Всего предусмотрено шесть уровней запуска:

- 0 — останов системы (ясно, что в качестве уровня по умолчанию этот уровень быть не может);
- 1 — однопользовательский режим (в него можно перейти сразу при загрузке, передав ядру параметр `single`);
- 2 — многопользовательский режим без поддержки сети;
- 3 — многопользовательский режим с поддержкой сети;
- 4 — не используется;
- 5 — многопользовательский графический режим с загрузкой X11 и поддержкой сети;
- 6 — перезагрузка системы.

В большинстве случаев в качестве уровня запуска по умолчанию устанавливается 3 или 5.

11.2.2. Команда *init*

Перейти на тот или иной уровень можно и после загрузки системы. Для этого используется команда:

```
# /sbin/init <уровень_запуска>
```

Примечание

Напомню, что решетка (#) перед командой означает, что команда должна быть выполнена от имени пользователя root.

"Вычислив" уровень запуска, `init` поочередно запускает сценарии из каталога /etc/rc.d/rcX.d, где X — это номер уровня запуска. Если зайти в один из этих каталогов, например в /etc/rc.d/rc3.d, то можно увидеть ссылки формата:

S<номер><имя>

Параметр <номер> определяет порядок запуска сценария (например, S10network запустится раньше, чем S11internet), а параметр <имя> — задает имя сценария. Сами сценарии находятся в каталоге /etc/rc.d/init.d.

Ссылки, начинающиеся на символ S, — это ссылки запуска (от *start*), при запуске соответствующих сценариев им будет передан аргумент `start`. Например, если `init` обнаружила в /etc/rc.d/rc3.d файл S10network, то она выполнит команду:

```
/etc/rc.d/init.d/network start
```

Если имя ссылки начинается на букву K (от *kill*), то это ссылка останова сервиса, например K01service. Данная ссылка указывает на команду:

```
/etc/rc.d/init.d/service stop
```

Вы можете запустить любой сценарий из каталога init.d непосредственно, передав ему параметры `start` (запуск), `stop` (останов) и др. (зависит от сервиса).

11.2.3. Команда *service*

А можете воспользоваться командой `service`:

```
# service <имя_сервиса> <start|stop|…>
```

Здесь <имя_сервиса> — это имя файла в каталоге /etc/rc.d/init.d.

11.2.4. Редакторы уровней запуска

Редактировать уровни запуска можно вручную, а можно и с помощью программ-конфигураторов. В Fedora (и ASPLinux) для редактирования уровней запуска используется конфигуратор `system-config-services` (рис. 11.1), а в Mandrake (Mandriva) можно воспользоваться командой `ntsysv` с параметром `--level` <номер сервиса> (рис. 11.2).

Конфигуратор в Fedora более удобен. Выбрать редактируемый уровень запуска можно с помощью меню **Изменить уровень.**

Конфигуратор `drakboot` (рис. 11.3), имеющийся в Linux Mandrake (Mandriva), позволяет указать, в каком режиме будет запускаться система — в графическом или в режиме консоли. По сути, конфигуратор позволяет выбрать уровень запуска (3 — консоль, 5 — графический режим).

В случае если система будет запускаться в графическом режиме, данный конфигуратор позволяет включить автовход. Для функции автовхода нужно указать два параметра — имя пользователя и графическую среду.

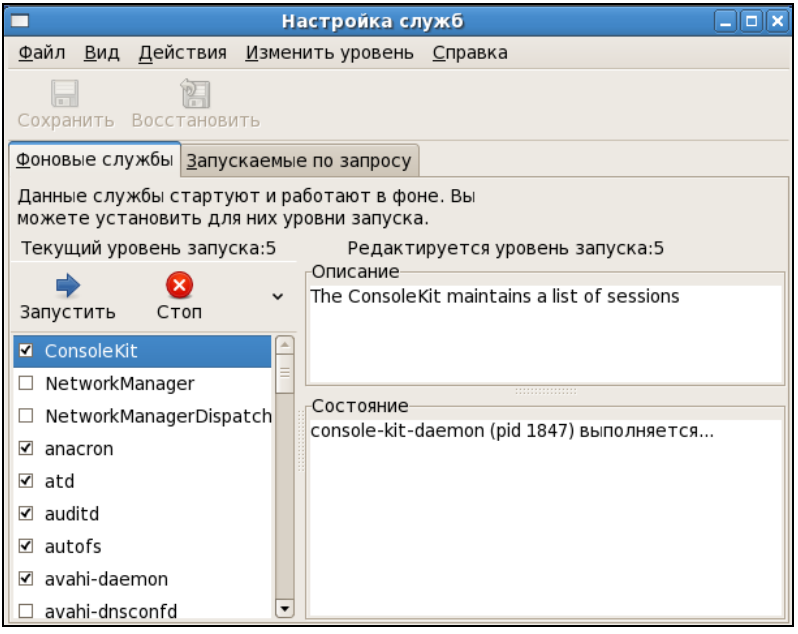


Рис. 11.1. Конфигуратор `system-config-services`

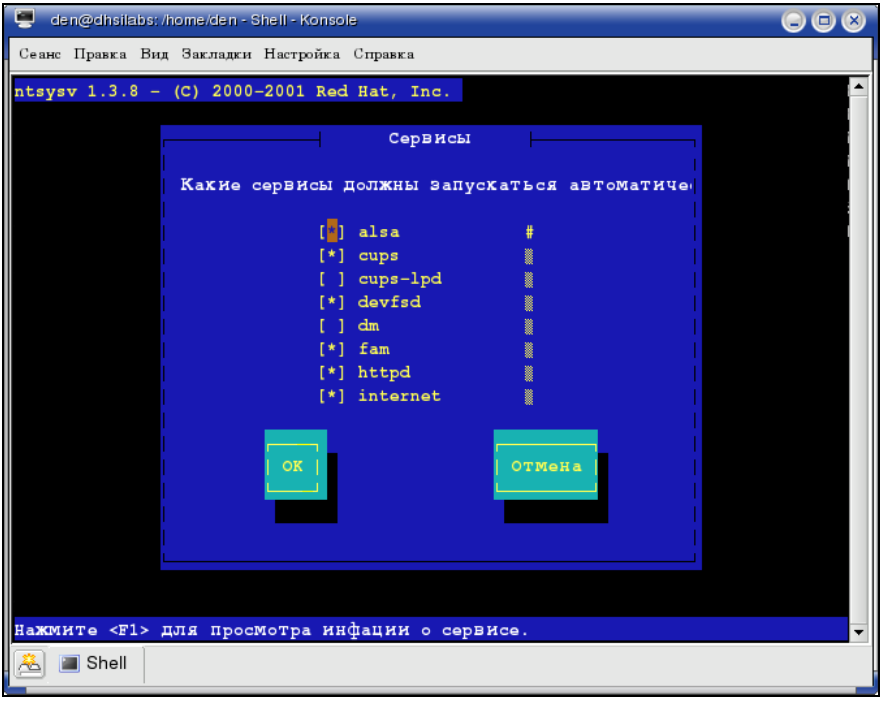


Рис. 11.2. Конфигуратор `ntsysv`

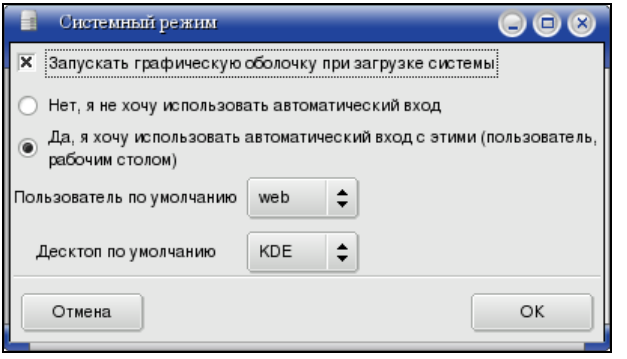


Рис. 11.3. Конфигуратор `drakboot`

Если автовход выключен, то при запуске X.Org система запросит у вас имя пользователя и пароль. Также у вас будет возможность выбрать графическую среду, с которой вы хотите работать. Если автовход включен, то будет выполнена автоматическая регистрация в системе выбранного пользователя с запуском выбранной графической среды. После этого вы можете работать в системе от имени этого пользователя. Из соображений безопасности конфигуратор не позволяет выбрать пользователя root.

Еще в Mandriva есть конфигуратор `drakxservices` (рис. 11.4), позволяющий редактировать сервисы, запускаемые на пятом уровне запуска (обычно этот уровень запуска и используется).

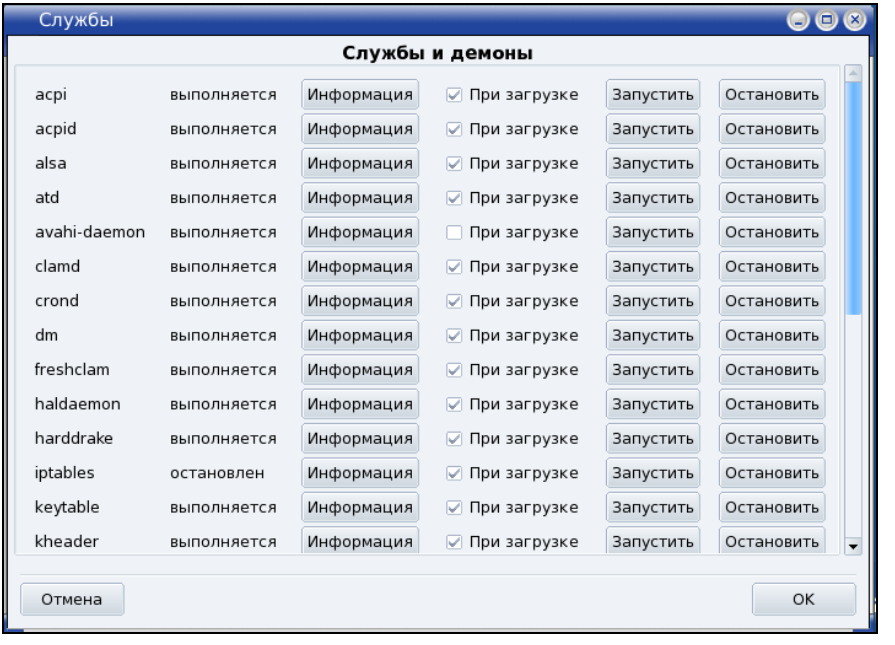


Рис. 11.4. Конфигуратор `drakxservices`

11.3. Система инициализации upstart

Система инициализации upstart была разработана Скотом Джеймсом Ремнантом (Scott James Remnant) для дистрибутива Ubuntu, однако upstart, если она вам понравилась, можно с успехом использовать в других дистрибутивах. Мы не будем рассматривать установку upstart на другой дистрибутив, а разберемся, как с ней работать в Ubuntu.

11.3.1. Как работает upstart

Upstart заменяет инициализирующие сценарии для поддержки событийно-ориентированного режима действий. Проще говоря, в upstart есть собственный процесс init, который запускается при запуске системы (аналогично программам `init` и `initng`). При запуске генерируется событие `startup`, при завершении работы — `shutdown`, при нажатии клавиатурной комбинации <Ctrl>+<Alt>+<Del> — событие `ctrl-alt-delete`.

Вы можете создавать собственные события. Вот небольшой пример создания события `my_event`:

```
on my_event
exec echo event received
console output
```

При получении этого события на консоль будет выведено сообщение:

```
event received
```

Файлы событий хранятся в каталоге /etc/event.d. Создайте в этом каталоге файл с именем my_event и поместите в него приведенный код. После этого вызвать событие вы можете командой:

```
initctl emit my_event
```

Подробнее об этой команде вы сможете прочитать на странице руководства (в Ubuntu оно на русском) — `man initctl`.

11.3.2. Конфигурационные файлы upstart

Исследуйте содержимое каталога /etc/event.d. В нем вы найдете файлы событий перехода на определенный запуск. В листинге 11.2 представлен файл события перехода на пятый уровень запуска — /etc/event.d/rc5.

Листинг 11.2. Файл события /etc/event.d/rc5

```
start on runlevel 5

stop on runlevel [!5]

console output
script
```

```
    set $(runlevel --set 5 || true)
    if [ "$1" != "unknown" ]; then
        PREVLEVEL=$1
        RUNLEVEL=$2
        export PREVLEVEL RUNLEVEL
    fi

    exec /etc/init.d/rc 5
end script
```

Не нужно быть гуру в программировании, чтобы понять, что делает этот сценарий — он выполняет сценарий /etc/init.d/rc, передав ему значение 5 — номер уровня запуска. Сценарий /etc/init.d/rc занимается запуском/остановкой служб на определенном уровне, который ему передается в качестве параметра.

Но самое интересное в upstart, что уровни запуска здесь — виртуальные. На самом деле, номера уровней запуска остались только ради совместимости с init, чтобы человеку, который впервые увидел upstart (точнее, дистрибутив с установленной системой инициализации upstart), было проще с ней разобраться. В upstart, благодаря событийно-ориентированному режиму, вообще отпадает необходимость в уровнях запуска, подобных тем, которые использовались в init. Загрузка того или иного сервиса происходит при наличии нужного аппаратного обеспечения: нет устройства — не будет загружен и сервис, требующий его.

Upstart можно использовать в режиме "горячей замены" — если вы в процессе работы системы подключите какое-нибудь устройство, например PCMCIA-карту или USB-устройство, будет сгенерировано соответствующее событие. После этого будут запущены все необходимые для обеспечения работы этого устройства процессы. Так, при подключении сетевой карты PCMCIA будет сгенерировано событие `network-interface-added`, которое запустит процесс настройки сетевой карты по DHCP, при этом будет сгенерировано новое событие — `network-interface-up` и т. д. Понятно, что если нет сетевых устройств, то и соответствующие им события не будут генерироваться.

11.4. Система инициализации Slackware

Система инициализации Slackware отличается от привычной системы init, используемой в SysV-системах. Она больше похожа на систему инициализации BSD-систем, хотя некоторые сходства с SysV все же есть.

Пояснение

Если вы совсем незнакомы с историей UNIX, то вам неизвестны и термины SysV (System V) и BSD. Считается, что UNIX "родилась" в 1969 году. В то время над проектом работали сотрудники компании Bell Labs (это подразделение AT&T) Руд Кенедей (Rudd Canaday), Дуг МакИлрой (Doug McIlroy), Дэннис Ричи (Dennis Ritchie) и Кен Томпсон (Ken Thompson). Позже UNIX заинтересовались другие организации, в частности, институт Беркли (Калифорния, США). В 1975 году появилась слегка модифицированная версия UNIX института Беркли, которая получила название BSD (Berkeley

Software Distribution), а версия AT&T (Bell Labs) стала называться System V (SysV). Обе системы были очень похожи друг на друга, но в то же время имели свои особенности. Например, BSD имела собственную систему инициализации, которая очень напоминает ту, что сейчас используется в Slackware Linux.

Если говорить о сходстве систем инициализации в стиле SysV и в стиле BSD, то у обеих систем присутствуют уровни запуска, имеется файл /etc/inittab — таблица инициализации *(см. разд. 11.2.1)*. Однако имена файлов системы инициализации BSD-стиля немного отличаются от имен файлов SysV-стиля.

Система инициализации Slackware построена таким образом, что вне зависимости от уровня запуска первым всегда запускается сценарий /etc/rc.d/rc.S. Он монтирует псевдофайловые системы /proc, sysfs и devfs, запускает систему hotplug (драйвер устройств, обеспечивающий их "горячее" подключение, т. е. подключение без выключения компьютера — например, USB-устройств), подключает разделы свопинга, монтирует и проверяет корневую файловую систему, монтирует другие файловые системы и т. д. Как видите, сценарий /etc/rc.d/rc.S выполняет бóльшую часть действий по инициализации системы. Обычно данный файл не требует изменения. Но иногда его приходится редактировать. Например, если вы создали файл подкачки и хотите, чтобы он подключался при загрузке системы, то команду `swapon <имя_файла>` нужно добавить в файл /etc/rc.d/rc.S после команды `/sbin/swapon -a`.

Сценарий /etc/rc.d/rc.S проверяет наличие файла /etc/rc.d/rc.modules.local, обеспечивающего загрузку модулей при старте системы. При условии что файл rc.modules.local существует, он запускается. В противном случае происходит поиск файла /etc/rc.d/rc.modules<-версия.ядра>, а если и его нет, тогда сценарий /etc/rc.d/rc.S пытается запустить файл /etc/rc.d/rc.modules. Один из этих файлов должен существовать, иначе система будет загружена без модулей, а это означает, что не будут работать некоторые устройства и поддерживаться некоторые файловые системы.

Кроме файла /etc/rc.d/rc.modules.local (или другого файла загрузки модулей, см. ранее), также используется файл /etc/rc.d/rc.netdevice. Он служит для загрузки модулей сетевых карт (точнее, сетевых интерфейсов).

Как уже было отмечено, файл /etc/rc.d/rc.S запускается вне зависимости от уровня запуска. Кроме этого файла в каталоге etc/rc.d вы найдете серию файлов rc.*N*, где *N* — номер уровня запуска. Данные файлы запускаются в зависимости от выбранного уровня запуска — например, на третьем уровне запуска будет запущен файл /etc/rc.d/rc.3. Каждый такой файл подготавливает систему к работе на выбранном уровне запуска. Уровень запуска по умолчанию, как и в случае с системой инициализации в стиле SysV, задается в файле /etc/inittab.

Сценарий /etc/rc.d/rc.inet1 отвечает за инициализацию сетевых интерфейсов и построение таблицы маршрутизации. Конфигурация сетевых интерфейсов хранится в файле /etc/rc.d/rc.inet1.conf. Вот фрагмент этого файла:

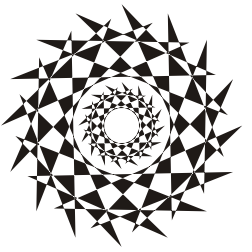
```
IPADDR[0]="192.168.1.1"
NETMASK[0]="255.255.255.0"
USE_DHCP[0]=""
DHCP_HOSTNAME[0]=""
```

Сценарий /etc/rc.d/r.inet2 управляет запуском сетевых служб и подключением сетевых файловых систем. Именно в этом файле происходит попытка монтирования файловых систем NFS и smbfs. Также из этого файла происходит запуск сетевых служб. Сценарии для запуска сетевых служб называются /etc/rc.d/rc.<название службы>, например /etc/rc.d/rc.sshd — сценарий запуска SSH-сервера. Однако некоторые сетевые сервисы, например sendmail и samba, в силу своих особенностей запускаются из файлов rc.*N*.

Иногда нужно обеспечить запуск сетевой службы, для которой нет собственного rc-файла. Тогда ее запуск можно или описать в файле /etc/rc.d/rc.local (что довольно просто) или создать собственный rc-файл и добавить его вызов в один из файлов rc.*N*. Шаблон собственного rc-файла приведен в листинге 11.3.

Листинг 11.3. Шаблон rc-файла для запуска сетевой службы

```
#! /bin/bash
start()
{
  echo "Service started"
  service_start
}
stop()
{
  echo "Service stoped"
  killall service
}

 case $1 in
  start)
    start ;;
  stop)
    stop ;;
  restart)
    stop
    sleep 2
    start ;;
  *)
    echo "Usage: service start|stop|restart"
esac
```



Глава 12

Команды управления пользователями

12.1. Многопользовательская система

Linux, как и UNIX, является многозадачной многопользовательской операционной системой. Это означает, что в один момент с системой могут работать несколько пользователей, и каждый пользователь может запустить несколько приложений. При этом вы можете зайти в систему локально, а кто-то — удаленно, используя один из протоколов удаленного доступа (Telnet, ssh) или по FTP. Согласитесь, очень удобно. Предположим, что вы забыли распечатать очень важный документ, а возвращаться домой уже нет времени. Если ваш компьютер должным образом настроен и подключен к Интернету, вы можете получить к нему доступ (даже если компьютер выключен, достаточно позвонить домой и попросить кого-нибудь включить его, а к Интернету компьютер подключится автоматически). После чего зайдите в систему по ssh (или подключитесь к графическому интерфейсу, если вы предпочитаете работать в графическом режиме) и скопируйте нужный вам файл. Даже если кто-то в момент вашего подключения уже работает с системой, вы не будете мешать друг другу.

Вы можете обвинить меня в рекламе Linux: мол, эта возможность была и в Windows 98, если установить соответствующее программное обеспечение вроде Remote Administrator. Должен отметить, что в Windows все иначе. Да, Remote Administrator предоставляет удаленный доступ к рабочему столу, но если за компьютером уже работает пользователь, то вы вместе работать не сможете — вы будете мешать ему, а он вам. Ведь все, что будете делать вы, будет видеть он, а все, что будет делать он, вы увидите у себя на экране, т. е. рабочий стол получится как бы общий. Если вы предварительно не предупредите пользователя о своем удаленном входе, он даже может подумать, что с системой что-то не то. Помню, со мной так и было — пользователь, работавший за компьютером, закрывал окна, которые я открывал, работая в удаленном режиме. Пришлось мне самому пойти к компьютеру того пользователя и попросить его не мешать.

В Linux же все так, как и должно быть. Несколько пользователей могут работать с системой и даже не подозревать о существовании друг друга, пока не введут соответствующую команду (`who`).

12.2. Пользователь root

12.2.1. Максимальные полномочия

Пользователь root обладает максимальными полномочиями в системе. Система полностью подвластна этому пользователю. Любая команда будет безоговорочно выполнена системой. Поэтому работать под именем пользователя root нужно с осторожностью. Всегда думайте над тем, что собираетесь сделать. Если вы дадите команду на удаление корневой файловой системы, система ее выполнит. Если же вы попытаетесь выполнить определенную команду, зарегистрировавшись под именем обычного пользователя, система сообщит вам, что у вас нет полномочий.

Представим, что кто-то решил пошутить и выложил в Интернете (записал на диск или прислал по электронной почте — не важно) вредоносную программу. Если вы ее запустите от имени пользователя root, система может быть уничтожена. Запуск этой же программы от имени обычного пользователя ничего страшного не произведет — система просто откажется ее выполнять.

Или же все может быть намного проще — вы ошибочно введете команду, которая разрушит вашу систему. Или просто отойдете ненадолго от своего компьютера, а тут сразу же появится доброжелатель, — имея полномочия пользователя root, уничтожить систему можно одной командой. Именно поэтому практически во всех современных дистрибутивах вход как root запрещен. В одних дистрибутивах вы не можете войти как root в графическом режиме (но можете войти в консоли, переключившись на первую консоль с помощью комбинации клавиш <Ctrl>+<Alt>+<F1>), а в других вообще не можете войти в систему как root — ни в графическом режиме, ни в консоли (пример такого дистрибутива — Ubuntu).

Отсюда можно сделать следующие выводы:

- старайтесь реже работать пользователем root;
- всегда думайте, какие программы вы запускаете под именем root;
- если программа, полученная из постороннего источника, требует root-полномочий, это должно насторожить;
- создайте обычного пользователя (даже если вы сами являетесь единственным пользователем компьютера) и рутинные операции (с документами, использование Интернета и т. д.) производите от имени этого пользователя;
- если полномочия root все же нужны, совсем необязательно заходить в систему под этим пользователем, достаточно запустить терминал и выполнить команду `su`. После этого в терминале можно выполнять команды с правами root. Если вы закроете терминал, то больше не сможете работать с правами root. Очень удобно — ведь обычно права root нужны для одной-двух операций (например, выполнить команду установки программы или создать/удалить пользователя).

12.2.2. Как работать без root

Некоторые операции, например установка программного обеспечения, изменение конфигурационных файлов, требуют полномочий root. Чтобы их временно

получить, нужно использовать команды sudo или su (эти команды, скорее всего, вы будете запускать в терминале).

Команда *sudo*

Команда sudo позволяет запустить любую команду с привилегиями root. Использовать ее нужно так:

```
sudo <команда_которую_нужно_выполнить_с_правами_root>
```

Например, вам необходимо изменить файл /etc/apt/sources.list. Для этого используется команда:

```
sudo nano /etc/apt/sources.list
```

Пояснение

Программа nano — это текстовый редактор, мы ему передаем один параметр — имя файла, который нужно открыть.

Если ввести эту же команду, но без sudo (просто nano /etc/apt/sources.list), текстовый редактор тоже запустится и откроет файл, но сохранить изменения вы не сможете, поскольку у вас не хватит полномочий.

Программа sudo перед выполнением указанной вами команды запросит у вас пароль:

```
sudo nano /etc/apt/sources.list
    Password:
```

Вы должны ввести свой *пользовательский пароль* — тот, который применяете для входа в систему, но не пароль пользователя root (кстати, мы его и не знаем).

Примечание

Использовать команду sudo имеют право не все пользователи, а только те, которые внесены в файл /etc/sudoers. Администратор системы (пользователь root) может редактировать этот файл с помощью команды visudo. Если у вас дистрибутив, который запрещает вход под учетной записью root (следовательно, у вас нет возможности отредактировать файл sudoers), то в файл sudoers вносятся пользователи, которых вы добавили при установке системы.

Команда *su*

Команда su позволяет получить доступ к консоли root любому пользователю (даже если пользователь не внесен в файл /etc/sudoers) при условии, что он знает пароль root. Понятно, что в большинстве случаев этим пользователем будет сам пользователь root — не будете же вы всем пользователям доверять свой пароль? Поэтому команда su предназначена в первую очередь для администратора системы, а sudo — для остальных пользователей, которым иногда нужны права root (чтобы они меньше отвлекали администратора от своей работы).

Использовать команду su просто:

```
su
```

После этого нужно ввести пароль пользователя root, и вы сможете работать в консоли как обычно. Использовать `su` удобнее, чем `sudo`, потому что вам не нужно вводить `su` перед каждой командой, которая должна быть выполнена с правами root.

Чтобы закрыть сессию `su`, нужно или ввести команду `exit` или просто закрыть окно терминала.

В случае если вы запускаете какую-нибудь графическую программу, требующую привилегий root, вы увидите окно с требованием ввести свой пароль, подобное изображенному на рис. 12.1.

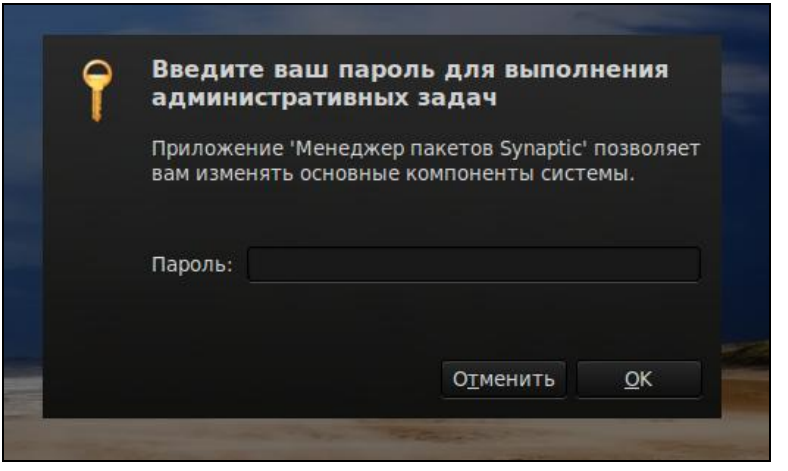


Рис. 12.1. Требование ввести пароль

Проблемы с *sudo* в Ubuntu и Kubuntu

Если вы в терминале хотите запустить графическую программу с правами root (например, `gedit`), желательно использовать не программу `sudo`, а программу `gksudo` или `gksu` (для Ubuntu) и `kdesu` (для Kubuntu). Программа `sudo` не всегда корректно работает с графическими приложениями, поэтому рано или поздно вы можете получить сообщение **`Unable to read ICE authority file`**, и после этого вообще станет невозможным запуск графических программ с правами root. Если это все же произошло, поправить ситуацию можно, удалив файл .{ICE,X}authority из вашего домашнего каталога:

```
rm ~/.{ICE,X}authority
```

Напомню, что тильда здесь означает домашний каталог текущего пользователя.

Графические приложения с правами root проще запускать, используя главное меню. Но не все приложения есть в главном меню, или не все приложения вызываются с правами root — например, в главном меню есть команда вызова текстового редактора, но нет команды для вызова текстового редактора с правами root. Поэтому намного проще нажать клавиатурную комбинацию <Alt>+<F2> и в от-

крывшемся диалоговом окне **Выполнить программу** ввести команду в соответствующее поле (рис. 12.2).

```
gksu <команда>
```

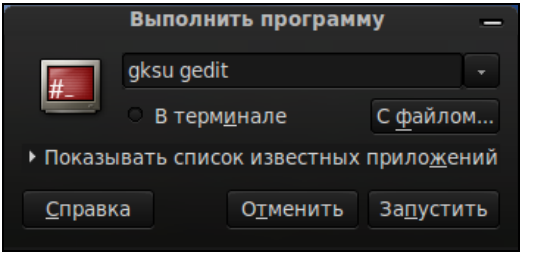


Рис. 12.2. Быстрое выполнение программы

Ввод серии команд *sudo*

Вам надоело каждый раз вводить `sudo` в начале команд? Тогда выполните команду:

```
sudo -i
```

Данная команда запустит оболочку root, т. е. вы сможете вводить любые команды, и они будут выполнены с правами root. Обратите внимание, что изменится приглашение командной строки (рис. 12.3). До этого приглашение имело вид `$`, что означало, что вы работаете от имени обычного пользователя, а после выполнения программы приглашение изменилось на # — это верный признак того, что каждая введенная команда будет выполнена с правами root.

Рис. 12.3. Оболочка root

Опция `-i` позволяет так же удобно вводить команды, как если бы вы использовали команду `sudo`.

12.2.3. Переход к традиционной учетной записи root

Преимущества и недостатки *sudo*

Как уже было отмечено, во многих дистрибутивах учетная запись root немного ограничена. В одних дистрибутивах она отключена, и для получения необходимых полномочий нужно использовать команду `sudo`, в других ограничивается использование учетной записи, например невозможно войти в графическом режиме как root.

Тем не менее возможность перейти к традиционной учетной записи root, т. е. заходить в систему под именем root, как вы заходите под именем обычного пользователя, имеется всегда. Чуть позже мы поговорим о том, как это сделать, но сначала рассмотрим преимущества (и недостатки) использования команды `sudo`.

К преимуществам `sudo` можно отнести следующее:

- вам не нужно помнить несколько паролей (т. е. ваш пароль и пароль пользователя root) — вы помните только свой пароль и вводите его, когда нужно;
- с помощью `sudo` вы можете выполнять практически те же действия, что и под именем root, но перед каждым действием у вас будет запрошен пароль, что позволит еще раз подумать о правильности своих действий;
- каждая команда, введенная с помощью `sudo`, записывается в журнал /var/log/auth.log, поэтому в случае чего вы хотя бы будете знать, что случилось, прочитав этот журнал. У вас также будет храниться история введенных команд с полномочиями root, в то время как при работе под именем root никакой журнал не ведется;
- предположим, некто захотел взломать вашу систему. Этот некто не знает, какие учетные записи есть в вашем компьютере, зато уверен, что учетная запись root есть всегда. Знает он также, что, завладев паролем к этой учетной записи, можно получить неограниченный доступ к системе. Но не к вашей системе — у вас учетная запись root отключена!
- вы можете разрешать и запрещать другим пользователям использовать полномочия root (позже мы разберемся, как это сделать), не предоставляя пароль root, т. е. практически нет риска скомпрометировать учетную запись root (впрочем, риск есть всегда, ведь при неправильно настроенной системе можно легко изменить пароль root с помощью команды `sudo`).

Но у `sudo` есть и недостатки:

- неудобно использовать перенаправление ввода/вывода, например команда:

```
sudo ls /etc > /root/somefile
```

работать не будет, вместо нее нужно использовать команду:

```
sudo bash -c "ls /etc > /root/somefile"
```

Длинновато, правда?

- имеются и неудобства, связанные с технологией NSS. К счастью, она используется не очень часто, поэтому основной недостаток `sudo` будет связан только с перенаправлением ввода/вывода.

Традиционная учетная запись root в Ubuntu

Вы все-таки хотите использовать обычную учетную запись root? Для этого достаточно задать пароль для пользователя root. Делается это командой:

```
sudo passwd root
```

Сначала программа запросит ваш пользовательский пароль, затем новый пароль root и его подтверждение:

```
Enter your existing password:
Enter password for root:
Confirm password for root:
```

После этого вы сможете входить в систему под учетной записью root.

Для отключения учетной записи root используется команда:

```
sudo passwd -l root
```

Помните, что после закрытия учетной записи root у вас могут быть проблемы с входом в систему в режиме восстановления, поскольку пароль root уже установлен (т. е. он не пустой, как по умолчанию), но в то же время учетная запись закрыта. Поэтому если вы уже включили учетную запись root, то будьте внимательны и осторожны. А вообще лучше ее не включать, а пользоваться командой `sudo -i`.

Традиционная учетная запись root в Mandriva

В Ubuntu учетная запись root отключена честно. В Linux Mandriva 2010 отключена лишь возможность графического входа в систему под именем root. Другими словами, вы можете переключиться в консоль, нажав клавиши <Ctrl>+<Alt>+<F1>, и войти в систему под именем root.

Сейчас мы разберемся, как же войти под именем root в графическом режиме. За регистрацию пользователей в системе в графическом режиме отвечает KDM (KDE Display Manager). Он-то и не пускает пользователя root в систему.

Для изменения поведения KDM следует открыть его конфигурационный файл. Это нужно сделать с привилегиями root:

```
su
kwrite /etc/kde/kdm/kdmrc (для Mandriva 2008)
kwrite /etc/alternatives/kdm4-config (для Mandriva 2009/2010)
```

В этом файле найдите строку:

```
AllowRootLogin=false
```

Значение директивы `AllowRootLogin` измените на `true`:

```
AllowRootLogin=true
```

После этого можно будет войти в систему под именем root. Кстати, при входе в систему вы получите предупреждение, а фон графического стола станет красным, извещая вас об опасности такого решения.

Вход в качестве root в Fedora

Как и в Mandriva, в Fedora 9 и 10 вход пользователя root ограничен менеджером рабочего стола. Введите команду:

```
su -c 'gedit /etc/pam.d/gdm'
```

Вы запустите с правами root текстовый редактор `gedit` для редактирования файла /etc/pam.d/gdm. Найдите в этом файле следующую строку

```
auth required pam_succeed_if.so user != root quiet
```

Закомментируйте ее (поставьте знак `#` перед ней) или вообще удалите эту строку.

В Fedora 11 и 12 дополнительно нужно открыть файл /etc/pam.d/gdm-password и найти следующую строку:

```
pam_succeed_if.so user != root quiet
```

Эту строку тоже нужно или закомментировать или удалить.

Если вы используете вход в систему по отпечатку пальца, тогда откройте файл gdm-fingerprint и закомментируйте в нем следующую строку:

```
pam_succeed_if.so user != root quiet
```

После этого сохраните файлы и завершите сеанс пользователя. После перезагрузки GDM вы сможете войти в систему как root.

Просмотреть видеоурок, демонстрирующий разрешение входа root в графическом режиме, можно по адресу:

http://dkws.org.ua/video-lessons/login_as_root_in_fedora_12.avi

12.3. Создание, удаление и модификация пользователей стандартными средствами

12.3.1. Команды *adduser* и *passwd*

Для добавления нового пользователя выполните следующие команды (от имени root):

```
# adduser <имя пользователя>
# passwd <имя пользователя>
```

Первая команда (`adduser`) добавляет пользователя, а вторая (`passwd`) изменяет его пароль. Ясно, что и в первом, и во втором случае вы должны указать одно и то же имя пользователя.

> ***Примечание***
>
> В некоторых дистрибутивах вместо команды `adduser` используется команда `useradd`, а команды `adduser` вообще не существует.

В некоторых дистрибутивах, например в Ubuntu и Debian, сценарий `adduser` не только добавляет пользователя, но позволяет указать дополнительную информацию о пользователе и сразу же задать пароль пользователя (рис. 12.4).

> ***Примечание***
>
> В некоторых дистрибутивах (например, в openSUSE) вместо команды `adduser` используется команда `useradd`. Программы `adduser` и `useradd` обычно находятся в каталоге /usr/sbin.

Обратите внимание: если пароль слишком прост для подбора, программа `passwd` выдаст соответствующее предупреждение — **`BAD PASSWORD`** и сообщит, чем же наш пароль плох (в нашем случае в основе пароля лежит словарное слово, что делает пароль легким для подбора).

> ***Примечание***
>
> Команду `passwd` может использовать не только администратор, но и сам пользователь для изменения собственного пароля.

```
root@denis-desktop:/home/denix# adduser denis
Добавляется пользователь `denis' ...
Добавляется новая группа `denis' (1001) ...
Добавляется новый пользователь `denis' (1001) в группу `denis' ...
Создаётся домашний каталог `/home/denis' ...
Копирование файлов из `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for denis
Enter the new value, or press ENTER for the default
        Full Name []: Denis
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Данная информация корректна? [Д/н] у
root@denis-desktop:/home/denix#
```

Рис. 12.4. Добавление нового пользователя в Ubuntu

12.3.2. Команда *usermod*

Для модифицирования учетной записи пользователя можно применять команду `usermod`. Формат вызова команды следующий:

```
usermod [параметры] LOGIN
```

Параметры команды `usermod` приведены в табл. 12.1.

Таблица 12.1. *Параметры команды* `usermod`

Параметр	Описание
`-a, --append`	Добавляет пользователя в дополнительную группу. Используется только с параметром `-G`
`-c, --comment`КОММЕНТАРИЙ	Изменяет комментарий пользователя. Комментарий удобнее изменять с помощью команды `chfn`
`-d, --home`КАТАЛОГ	Изменяет домашний каталог пользователя
`-e, --expiredate`ДАТА	Устанавливает дату отключения учетной записи. Дата устанавливается в формате ГГГГ-ММ-ДД
`-f, --inactive`ДНИ	Устанавливает число дней после даты отключения учетной записи, на протяжении которых можно изменить пароль. Если число дней равно 0, то учетная запись блокируется сразу после устаревания пароля
`-g, --gid`ГРУППА	Задает идентификатор (число, GID) начальной группы пользователя

Таблица 12.1 *(окончание)*

Параметр	Описание
`-G, --groupsГ1[,Г2,...[,ГN]]`	Задает список дополнительных групп, в которые будет входить пользователь. Список групп задается через запятую без дополнительных пробелов. Если пользователь уже является членом группы, которой нет в списке, он будет удален из этой группы. Поведение программы можно изменить, добавив параметр `-a`, в этом случае пользователь просто будет добавлен в дополнительные группы, а если он уже являлся членом других групп, которых нет в списке, то он не будет удален из них
`-l, --loginНОВОЕ_ИМЯ`	Изменяет имя пользователя
`-L, --lock`	Заблокировать пароль пользователя. Не используйте этот параметр вместе с параметрами `-p` или `-U`. Параметр `-L` блокирует только вход по паролю, но есть и другие способы аутентификации, поэтому если вам нужно полностью заблокировать учетную запись, используйте параметр `-e`
`-o, --non-unique`	При использовании с параметром `-u` позволяет задавать не уникальный идентификатор пользователя
`-p, --passwordПАРОЛЬ`	Задает новый пароль пользователя. Нужно указать не сам пароль, а его хэш. Поэтому для смены пароля проще использовать команду `passwd`
`-s, --shellОБОЛОЧКА`	Задает новую оболочку пользователя
`-u, --uidUID`	Задает идентификатор пользователя. Идентификатор должен быть уникальным, если не задан параметр `-o`. Идентификаторы 0–999 зарезервированы для системных учетных записей
`-U, --unlock`	Разблокировать пароль пользователя

12.3.3. Команда *userdel*

Для удаления пользователя используется команда `userdel`:

```
# userdel <имя пользователя>
```

12.3.4. Подробно о создании пользователей

Давайте разберемся, что же происходит при создании новой учетной записи пользователя.

Во-первых, создается запись в файле /etc/passwd. Формат записи следующий:

```
имя_пользователя:пароль:UID:GID:полное_имя:домашний_каталог:оболочка
```

Рассмотрим фрагмент этого файла (две строки):

```
root:x:0:0:root:/root:/bin/bash
den:x:500:500:Denis:/home/den:/bin/bash
```

- первое поле — это логин пользователя, который он вводит для регистрации в системе. Пароль в современных системах в этом файле не указывается, а второе поле осталось просто для совместимости со старыми системами. Пароли хранятся в файле /etc/shadow, о котором мы поговорим чуть позже;
- третье и четвертое поле — это UID (User ID) и GID (Group ID) — идентификаторы пользователя и группы соответственно. Идентификатор пользователя root всегда равен 0, как и идентификатор группы root. Список групп вы найдете в файле /etc/groups;
- пятое поле — это настоящее имя пользователя. Может быть не заполнено, а может содержать фамилию, имя и отчество пользователя — все зависит от педантичности администратора системы, т. е. от вас. Если вы работаете за компьютером в гордом одиночестве, то, думаю, свою фамилию вы не забудете. А вот если ваш компьютер — сервер сети, тогда просто необходимо указать Ф. И. О. каждого пользователя, а то когда придет время обратиться к пользователю по имени, вы его знать не будете (попробуйте запомнить 500 фамилий и имен!);
- шестое поле содержит имя домашнего каталога. Обычно это каталог /home/<имя_пользователя>;
- последнее поле — это имя командного интерпретатора, который будет обрабатывать введенные вами команды, когда вы зарегистрируетесь в консоли.

В целях безопасности пароли были перенесены в файл /etc/shadow (доступен для чтения/записи только пользователю root), где они и хранятся в закодированном виде (используется алгоритм MD5 или — в некоторых системах — Blowfish). Узнать, с помощью какого алгоритма зашифрован пароль, очень просто: посмотрите на шифр — если он короткий и не начинается с символа `$`, то применен алгоритм DES (самый слабый и ненадежный — как правило, используется в старых дистрибутивах). Если же шифр начинается с символов `$1$`, то это MD5, а если в начале шифра имеются символы `$2a$`, то это Blowfish.

Во-вторых, при создании пользователя создается каталог /home/<имя пользователя>, в который копируется содержимое каталога /etc/skel. Каталог /etc/skel содержит "джентльменский набор" — файлы конфигурации по умолчанию, которые должны быть в любом пользовательском каталоге. Название каталога skel (от *skeleton*) полностью оправдывает себя — он действительно содержит "скелет" домашнего каталога пользователя.

Примечание

Файл /etc/passwd можно редактировать с помощью обычного текстового редактора. То есть вы можете очень легко, не прибегая к помощи ни графического конфигуратора, ни команды `usermod`, изменить параметры учетной записи любого пользователя, например задать для него другую оболочку или прописать его настоящую фамилию. Однако нужно быть осторожным при изменении домашнего каталога пользователя! Если вы это сделали, то, чтобы у пользователя не возникло проблем с правами доступа для нового каталога, нужно выполнить команду:

```
chown -R <пользователь> <каталог>
```

12.4. Группы пользователей

Иногда пользователей объединяют в *группы*. Группы позволяют более эффективно управлять правами пользователей. Например, у нас есть три пользователя: igor, pavel, alex, которые должны совместно работать над проектом. Их достаточно объединить в одну группу — тогда пользователи будут иметь доступ к домашним каталогам друг друга (по умолчанию один пользователь не имеет доступ к домашнему каталогу другого пользователя, поскольку пользователи находятся в разных группах).

Создать группу, а также поместить пользователя в группу, позволяют графические конфигураторы. Вы можете использовать их — они очень удобные, но если вы хотите стать настоящим линуксоидом, то должны знать, что доступные в системе группы указываются в файле /etc/group. Добавить новую группу в систему можно с помощью команды `groupadd`, но, как правило, проще добавить в текстовом редакторе еще одну запись в файл /etc/group, а изменить группу пользователя еще проще — для этого достаточно отредактировать файл /etc/passwd.

12.5. Команды квотирования

Квотирование — это механизм ограничения дискового пространства пользователей. Linux — многопользовательская система, поэтому без ограничения дискового пространства вам не обойтись. Когда вы используете компьютер в гордом одиночестве, то все дисковое пространство доступно вам и только вам. А вот когда пользователей несколько, нужно ограничить доступное пространство, чтобы один из пользователей не "узурпировал" все место на диске. Как именно вы будете ограничивать дисковое пространство, решать только вам. Можно поделить дисковое пространство поровну между пользователями, можно одним пользователям отдать больше места, а другим — меньше.

На домашнем компьютере квотирование вряд ли понадобится, а на сервере, как правило, для каталога /home отводится отдельный раздел жесткого диска. Поэтому будем считать, что у нас есть отдельный раздел, который монтируется к каталогу /home.

Перед настройкой квот нужно установить пакет `quota`. Больше ничего устанавливать не нужно.

Чтобы пользователи не потеряли свои данные, перезагрузитесь в однопользовательский режим (параметр ядра `single`). Вот теперь можно приступать к редактированию квот. Первым делом разрешим устанавливать квоты на разделе, который содержит файлы пользователей. Откройте /etc/fstab:

```
# nano /etc/fstab
```

Далее добавьте параметр `usrquota` к списку параметров раздела:

```
/dev/sda5     /home         ext4   defaults,usrquota     0   2
```

Далее перемонтируем /home, т. к. мы только что изменили его параметры:

```
# mount -o remount /home
```

Механизм квотирования требует создания файлов aquota.user и aquota.group, но поскольку мы не будем устанавливать квоты для групп, а только для пользователей, то создадим только файл aquota.user:

```
# touch /home/aquota.user
# chmod 600 /home/aquota.user
```

После этого введите команду:

```
# quotacheck -vagum
```

Поскольку мы создали файл aquota.user вручную, то вы увидите сообщение об ошибке, но это только в первый раз — далее все будет нормально:

```
quotacheck: WARNING — Quotafile /home/aquota.user was probably truncated. Can't save quota settings...
quotacheck: Scanning /dev/sda5 [/home] quotacheck: Old group file not found. Usage will not be substracted.
        done
        quotacheck: Checked 3275 directories and 54301 files
```

Теперь отредактируем квоты для пользователя user:

```
# edquota -u user
```

Будет запущен текстовый редактор по умолчанию, и вы увидите следующий текст:

```
Disk quotas for user user (uid 1001):
   Filesystem   blocks    soft      hard     inodes    soft   hard
   /dev/sda5    16         0         0          5        0      0
```

Примечание

По умолчанию используется редактор `vi`, который, мягко говоря, не очень удобный. Для изменения редактора по умолчанию установите переменную окружения `EDITOR`. Например, `EDITOR=nano`.

Разберемся, что есть что:

- `blocks` — место в блоках, используемое пользователем (1 блок = 1 Кбайт);
- `soft` — максимальное дисковое пространство (в блоках по 1 Кбайт), которое может занимать пользователь. Если вы включите период отсрочки (grace period), то пользователь получит только лишь сообщение о превышении квоты;
- `hard` — жесткое ограничение, эту квоту пользователь превысить не может, даже если включен период отсрочки. Предположим, что вы хотите "отдать" пользователю 500 Мбайт. В качестве жесткой квоты можно установить значение 500 Мбайт (или 500 000 блоков), а в качестве "мягкой" квоты установить значение 495 Мбайт (495 000 блоков). Когда пользователь превысит 495 Мбайт, он получит сообщение о превышении квоты, а вот когда будет превышена жесткая квота, то пользователь больше не сможет сохранять файлы в своем домашнем каталоге;
- `inodes` — число используемых пользователем файлов.

Отредактируйте квоты так:

```
Disk quotas for user user (uid 1001):
  Filesystem   blocks     soft     hard   inodes   soft   hard
  /dev/sda5      16      495000   500000     5       0      0
```

Теперь сохраните файл, выйдите из редактора и введите команду:

```
# edquota -t
```

Сейчас мы установим период отсрочки:

```
Grace period before enforcing soft limits for users:
  Time units may be: days, hours, minutes, or seconds
  Filesystem     Block grace period     Inode grace period
  /dev/sda8           7days                   7days
```

Вы должны вместо `7days` вписать свой период отсрочки, при этом используйте названия единиц измерения времени на английском:

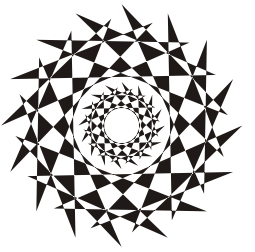
- `seconds` — секунды;
- `minutes` — минуты;
- `hours` — часы;
- `days` — дни;
- `weeks` — недели;
- `months` — месяцы.

Например:

- `24hours` — 24 часа;
- `2days` — 2 дня;
- `1weeks` — 1 неделя.

Для просмотра квот используется команда `repquota`, например:

```
# repquota /home
```



Глава 13

Ядро

13.1. Команда *dmesg*: вывод сообщений ядра

При загрузке Linux выводятся сообщения ядра: информационные (об имеющемся оборудовании, о поддерживаемых протоколах и технологиях и т. д.) и диагностические (например, об ошибках). Современные компьютеры настолько быстры, что вы просто не успеете прочитать эти сообщения. Не беда: все сообщения всегда можно прочитать после загрузки системы с помощью команды:

```
# dmesg | less
```

Сначала сообщается версия ядра и версия компилятора `gcc`, с помощью которого было откомпилировано ядро. Как видите, у нас имеется дистрибутив Fedora 12 (предпоследняя версия Fedora, но для анализа вывода ядра вам последняя и не нужна) и версия ядра 2.6.31.1-56.fc12:

```
Linux version 2.6.31.1-56.fc12.i686.PAE (mockbuild@x86-4.fedora.phx.
redhat.com) (gcc version 4.4.1 20090925 (Red Hat 4.4.1-17) (GCC) ) #1 SMP
Tue Sep 29 16:16:16 EDT 2009
```

После этого выводится список поддерживаемых процессоров:

```
KERNEL supported cpus:
  Intel GenuineIntel
  AMD AuthenticAMD
  NSC Geode by NSC
  Cyrix CyrixInstead
  Centaur CentaurHauls
  Transmeta GenuineTMx86
  Transmeta TransmetaCPU
```

Затем выводится карта физической (не виртуальной) памяти:

```
BIOS-provided physical RAM map:
BIOS-e820: 0000000000000000 — 000000000009fc00 (usable)
BIOS-e820: 000000000009fc00 — 00000000000a0000 (reserved)
BIOS-e820: 00000000000e4000 — 0000000000100000 (reserved)
BIOS-e820: 0000000000100000 — 0000000077fd0000 (usable)
BIOS-e820: 0000000077fd0000 — 0000000077fde000 (ACPI data)
BIOS-e820: 0000000077fde000 — 0000000078000000 (ACPI NVS)
BIOS-e820: 00000000fff00000 — 0000000100000000 (reserved)
```

Всего памяти 1023 + 896 = 1919 Мбайт (остальное "скушала" встроенная видеокарта):

```
1023MB HIGHMEM available.
896MB LOWMEM available.
```

Далее будет сказано, что найдена SMP MP-таблица (SMP, Symmetrical MultiProcessing; MP, MultiProcessing):

```
found SMP MP-table at 000ff780
```

Это означает, что у нас или двухпроцессорный компьютер, или уже у нашего компьютера двухъядерный процессор.

Современные процессоры поддерживают функцию NX-защиты (NX — сокращение от *No eXecute*). Это защита областей памяти, которая используется для предотвращения распространения вирусов, троянских коней и других вредоносных программ. Довольно часто вредоносные программы нарочно вызывают переполнение буфера, после чего записывают свой код в область данных и передают ему управление. Функция NX-защиты как раз предотвращает развитие данного сценария на аппаратном уровне. Подробно о NX-защите вы можете прочитать по адресу **http://ru.wikipedia.org/wiki/NX-бит**. А пока можете наблюдать за тем, как ядро включит NX-защиту (если она поддерживается вашим процессором):

```
Using x86 segment limits to approximate NX protection
```

Затем опять выводится информация о распределении памяти, что нам особо не интересно:

```
Zone PFN ranges:
  DMA             0 ->     4096
  Normal       4096 ->   229376
  HighMem    229376 ->   491472
Movable zone start PFN for each node
early_node_map[1] active PFN ranges
    0:        0 ->   491472
On node 0 totalpages: 491472
  DMA zone: 32 pages used for memmap
  DMA zone: 0 pages reserved
  DMA zone: 4064 pages, LIFO batch:0
  Normal zone: 1760 pages used for memmap
  Normal zone: 223520 pages, LIFO batch:31
  HighMem zone: 2047 pages used for memmap
  HighMem zone: 260049 pages, LIFO batch:31
  Movable zone: 0 pages used for memmap
DMI present.
Using APIC driver default
```

Следом — параметры ACPI (Advanced Configuration and Power Interface):

```
ACPI: RSDP 000F98B0, 0014 (r0 ACPIAM)
```

```
ACPI: RSDT 77FD0000, 0038 (r1 061107 RSDT1114 20070611 MSFT 97)
ACPI: FACP 77FD0200, 0084 (r2 061107 FACP1114 20070611 MSFT 97)
ACPI: DSDT 77FD0430, 4271 (r1  1ADNC 1ADNCB33 B33 INTL 20051117)
ACPI: FACS 77FDE000, 0040
ACPI: APIC 77FD0390, 005C (r1 061107 APIC1114 20070611 MSFT 97)
ACPI: MCFG 77FD03F0, 003C (r1 061107 OEMMCFG  20070611 MSFT 97)
ACPI: OEMB 77FDE040, 0071 (r1 061107 OEMB1114 20070611 MSFT 97)
ACPI: HPET 77FD46B0, 0038 (r1 061107 OEMHPET  20070611 MSFT 97)
ATI board detected. Disabling timer routing over 8254.
ACPI: PM-Timer IO Port: 0x808
ACPI: Local APIC address 0xfee00000
ACPI: LAPIC (acpi_id[0x01] lapic_id[0x00] enabled)
```

Обратите внимание: параметры контроллера прерываний (APIC, Advanced Programmable Interrupt Controller) выводятся для каждого процессора (или ядра процессора — Linux расценивает каждое ядро как отдельный процессор):

```
Processor #0 15:11 APIC version 16
ACPI: LAPIC (acpi_id[0x02] lapic_id[0x01] enabled)
Processor #1 15:11 APIC version 16
ACPI: IOAPIC (id[0x02] address[0xfec00000] gsi_base[0])
```

Здесь мы вывод немного сократим, потому что в нем не содержится ничего важного (зачем попусту тратить бумагу и ваше время), и перейдем сразу к параметрам ядра, которые были переданы загрузчиком:

```
Kernel command line: ro root=LABEL=/ rhgb quiet
```

Параметр `ro` говорит ядру о необходимости смонтировать корневую файловую систему в режиме "только чтение" (в процессе загрузки она будет перемонтирована в режиме "чтение/запись", `rw`). Корневая файловая система задается меткой `LABEL`. Помните, мы говорили о способах адресации разделов? Чтобы идентифицировать раздел, можно указать его короткое имя, вроде /dev/sda, длинное имя или метку. Fedora использует как раз последний вариант.

Современные дистрибутивы производят загрузку в графическом режиме. Если ранее сначала выводились диагностические сообщения ядра, затем сообщения системы инициализации, то сейчас вы едва успеете заметить сообщения ядра, как появится красивый графический индикатор, информирующий вас, сколько времени осталось до полной загрузки системы. Если вы предпочитаете видеть диагностические сообщения, а не графический индикатор загрузки, то уберите параметр `rhgb`. Это можно сделать в настройках загрузчика Linux *(см. главу 10)*.

Параметр `quiet` вообще выключает сообщения, выводимые ядром — т. е., если ядру передан данный параметр, сообщений ядра при загрузке системы вы вообще не увидите. Зато их можно потом получить командой `dmesg` или просто просмотреть файл /var/log/dmesg.

Далее ядро выполнит маппинг APIC, настроит математический сопроцессор (Floating Point Unit) и перейдет к инициализации первого процессора:

```
mapped APIC to ffffb000 (fee00000)
mapped IOAPIC to ffffa000 (fec00000)
Enabling fast FPU save and restore... done.
Enabling unmasked SIMD FPU exception support... done.

Initializing CPU#0
CPU 0 irqstacks, hard=c07a8000 soft=c0788000
PID hash table entries: 4096 (order: 12, 16384 bytes)
```

При инициализации процессора выводится его частота:

```
Detected 2194.946 MHz processor.
```

Понятно, что частота второго процессора будет такой же. Затем ядро инициализирует консоль:

```
Console: colour VGA+ 80x25
console [tty0] enabled
```

Далее вы увидите подробную информацию об использовании памяти:

```
Memory: 1940968k/1965888k available (2183k kernel code, 23600k reserved, 1117k data, 280k init, 1048384k highmem)
```

Итак, у нас "всего" 1 965 888 Кбайт, из которых доступно 1 940 968 Кбайт, а в скобках приведен отчет о том, куда ядро израсходовало память — сколько килобайт зарезервировано для кода ядра, данных и т. д.

После этого вы увидите информацию о выделении виртуальной памяти ядра:

```
virtual kernel memory layout:
    fixmap  : 0xffc53000 - 0xfffff000   (3760 kB)
    pkmap   : 0xff800000 - 0xffc00000   (4096 kB)
    vmalloc : 0xf8800000 - 0xff7fe000   ( 111 MB)
    lowmem  : 0xc0000000 - 0xf8000000   ( 896 MB)
    .init : 0xc073f000 - 0xc0785000   ( 280 kB)
    .data : 0xc0621f7d - 0xc0739544   (1117 kB)
    .text : 0xc0400000 - 0xc0621f7d   (2183 kB)
```

Пропустим пару бесполезных строк и перейдем к BogoMIPS:

```
Calibrating delay using timer specific routine.. 4500.53 BogoMIPS (lpj=2250269)
```

Ядро вычисляет производительность процессора в так называемых BogoMIPS. MIPS — это аббревиатура от *Millions of Instructions Per Second*. Bogo — происходит от *bogus* (фальшивый, поддельный). Префикс Bogo ставит под сомнение актуальность вычисленной ядром величины, поэтому Линус Торвальдс (BogoMIPS — это его изобретение) и назвал ее фальшивой. В Интернете можно найти довольно точное определение BogoMIPS: "сколько миллионов раз в секунду процессор

может ничего не делать". О производительности процессора по BogoMIPS можно судить лишь косвенно. Понятно, что чем производительнее процессор, тем больше будет сделано "пустых" операций, но одно дело, когда процессор просто ничего не делает, и совсем другое, когда он работает под "нагрузкой", т. е. выполняет арифметические и мультимедийные инструкции. Например, Duron 1,6 ГГц показывал результат в 3193,85 BogoMIPS, а Athlon X2 4200 — "всего" 4500, несмотря на бóльшую частоту и два ядра. На практике же Athlon X2 намного быстрее, чем Duron 1,6 ГГц.

Сразу после вычислений бесполезных BogoMIPS инициализируется система контроля доступа SELinux. SELinux может быть или выключена, или работать в одном из двух режимов — принудительном (permissive) или режиме предупреждений. В данном случае SELinux инициализируется и переходит в принудительный режим:

```
Security Framework v1.0.0 initialized
SELinux:  Initializing.
SELinux:  Starting in permissive mode
selinux_register_security:  Registering secondary module capability
```

После SELinux инициализируются модули безопасности Linux (LSM, Linux Security Modules):

```
Capability LSM initialized as secondary
```

Опять пропустим несколько строк и перейдем к информации о кэше процессора:

```
CPU: L1 I Cache: 64K (64 bytes/line), D cache 64K (64 bytes/line)
CPU: L2 Cache: 512K (64 bytes/line)
CPU 0(2) -> Core 0
```

У нас 64 Кбайт кэша первого уровня (точнее 2 по 64 Кбайт) и 512 Кбайт кэш-памяти второго уровня.

Еще через несколько строк вы увидите полное наименование процессора:

```
CPU0: AMD Athlon(tm) 64 X2 Dual Core Processor 4200+ stepping 02
```

Далее ядро Linux перейдет к инициализации второго процессора или второго ядра процессора (в зависимости от того, какая у вас машина — многопроцессорная или многоядерная). Фактически заново будет выведена уже известная нам информация — будет вычислен BogoMIPS, выведена информация о кэш-памяти, наименование процессора. Не перестаю удивляться значениям BogoMIPS. Несколькими строками ранее было вычислено значение 4500 BogoMIPS, а затем ядро сообщает, что второй процессор, точнее второе ядро процессора, выполняет "всего" 4389,59 BogoMIPS, а общее значение BogoMIPS для двух ядер — 8890. Конечно, эти цифры ничего не означают, но все же интересно. Получается, что если верить BogoMIPS, второе ядро менее производительно, чем первое:

```
Initializing CPU#1
Calibrating delay using timer specific routine.. 4389.59 BogoMIPS
(lpj=2194797)
```

```
CPU: After generic identify, caps: 178bfbff ebd3fbff 00000000 00000000
00002001 00000000 0000001f 00000000
CPU: L1 I Cache: 64K (64 bytes/line), D cache 64K (64 bytes/line)
CPU: L2 Cache: 512K (64 bytes/line)
CPU 1(2) -> Core 1
CPU: After all inits, caps: 178bf3ff ebd3fbff 00000000 00000410
00002001 00000000 0000001f 00000000
Intel machine check architecture supported.
Intel machine check reporting enabled on CPU#1.
CPU1: AMD Athlon(tm) 64 X2 Dual Core Processor 4200+ stepping 02
Total of 2 processors activated (8890.13 BogoMIPS).
```

Потом будет выведено общее количество процессоров (или ядер одного процессора), а также размер внутренних структур ядра:

```
Brought up 2 CPUs
sizeof(vma)=84 bytes
sizeof(page)=32 bytes
sizeof(inode)=336 bytes
sizeof(dentry)=132 bytes
sizeof(ext3inode)=488 bytes
sizeof(buffer_head)=56 bytes
sizeof(skbuff)=180 bytes
sizeof(task_struct)=1552 bytes
```

А также дата и время загрузки:

```
Time:  9:14:10  Date: 02/12/08
```

Затем вы увидите строки, относящиеся к инициализации шины PCI, распределению PCI-ресурсов. Они нам не интересны, поэтому их подробно рассматривать не станем.

Далее перед вашим взором предстанет строка, появление которой означает поддержку технологии PnP (Plug and Play) операционной системой Linux. Ничего удивительного в этом нет — было бы удивительно, если бы Linux не поддерживала PnP:

```
Linux Plug and Play Support v0.97 (c) Adam Belay
```

Следующий этап — подготовка к загрузке поддержки USB (загружаются основные драйверы USB):

```
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
```

Затем ядро выводит параметры таймеров, диапазоны ввода/вывода PnP, параметры моста PCI. Если вы не профессионал в аппаратных средствах компьютера, вся эта информация не представляет для вас никакого интереса.

После этого вы увидите строки, свидетельствующие о регистрации протокола TCP/IP:

```
NET: Registered protocol family 2
IP route cache hash table entries: 32768 (order: 5, 131072 bytes)
TCP established hash table entries: 131072 (order: 8, 1572864 bytes)
TCP bind hash table entries: 65536 (order: 7, 524288 bytes)
TCP: Hash tables configured (established 131072 bind 65536)
TCP reno registered
```

Затем ядро проверяет RAM-диск и освобождает память, используемую для initrd:

```
checking if image is initramfs... it is
Freeing initrd memory: 2911k freed
```

APM (Advanced Power Management) не очень надежно работает на SMP-машинах (в том числе и на многоядерных машинах, которые тоже являются SMP-машинами), поэтому ядро отключило APM:

```
apm: BIOS version 1.2 Flags 0x03 (Driver version 1.16ac)
apm: disabled — APM is not SMP safe.
```

Опять пропустим несколько строк и перейдем сразу к режимам планировщика ввода/вывода. Ядро регистрирует режимы noop, anticipatory, deadline и cfq. Последний режим используется по умолчанию. О выборе планировщика вы можете прочитать в моей статье:

http://www.dkws.org.ua/index.php?page=show&file=a/system/tuning

```
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered (default)
```

Далее мы лишь с краткими комментариями остановимся на самых значимых сообщениях ядра (поскольку полный вывод ядра можно рассматривать очень долго).

Инициализация PnP карт расширений ISA — такие устройства не найдены (интересно, у кого-то до сих пор используется ISA?):

```
isapnp: Scanning for PnP cards...
isapnp: No Plug & Play device found
```

Загрузка драйвера RTC (часов реального времени):

```
Real Time Clock Driver v1.12ac
```

Инициализация последовательных портов:

```
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing enabled
serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
00:06: ttyS0 at I/O 0x3f8 (irq = 4) is a 16550A
```

Инициализация драйвера `RAMDISK` (диска в памяти, который используется в качестве корневой файловой системы, пока ядро не подмонтирует раздел жесткого диска):

```
RAMDISK driver initialized: 16 RAM disks of 16384K size 4096 blocksize
```

Контроллер PS/2 (к нему подключаются клавиатура и мышь) занял прерывания 1 и 12:

```
PNP: PS/2 Controller [PNP0303:PS2K,PNP0f03:PS2M] at 0x60,0x64 irq 1,12
serio: i8042 KBD port at 0x60,0x64 irq 1
serio: i8042 AUX port at 0x60,0x64 irq 12
```

Загрузка драйвера USB HID (Human Interface Device):

```
cpuidle: using governor menu
usbcore: registered new interface driver hiddev
usbcore: registered new interface driver usbhid
drivers/hid/usbhid/hid-core.c: v2.6:USB HID core driver
```

Регистрирование сетевых протоколов:

```
Initializing XFRM netlink socket
NET: Registered protocol family 1
NET: Registered protocol family 17
```

Активация функции PowerNow! — энергосберегающей технологии AMD:

```
powernow-k8: Found 1 AMD Athlon(tm) 64 X2 Dual Core Processor 4200+
processors (2 cpu cores) (version 2.00.00)
powernow-k8: MP systems not supported by PSB BIOS structure
powernow-k8: MP systems not supported by PSB BIOS structure
```

Освобождение неиспользуемой памяти ядра:

```
Freeing unused kernel memory: 280k freed
Write protecting the kernel read-only data: 849k
```

Регистрация USB-контроллера, версия USB 2.0:

```
ehci_hcd 0000:00:13.5: EHCI Host Controller
ehci_hcd 0000:00:13.5: new USB bus registered, assigned bus number 1
ehci_hcd 0000:00:13.5: debug port 1
ehci_hcd 0000:00:13.5: irq 16, io mem 0xfe7ff000
ehci_hcd 0000:00:13.5: USB 2.0 started, EHCI 1.00, driver 10 Dec 2004
usb usb1: configuration #1 chosen from 1 choice
```

Найден USB-хаб на 10 портов:

```
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 10 ports detected
...
```

Найден USB-хаб, 2 порта:

```
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 2 ports detected
```

Далее ядро будет выводить информацию о каждом найденном USB-хабе, что не очень интересно.

Инициализация подсистемы SATA и загрузка библиотеки libata версии 2.21 (именно эта библиотека "превратила" все имена /dev/hd* в /dev/sd*):

```
SCSI subsystem initialized
libata version 2.21 loaded.
```

Прерывания 14 и 15 используются первым и вторым жестким диском (IDE).

На данной материнской плате всего один IDE-контроллер, к которому можно подключить два IDE-устройства:

```
ata1: PATA max UDMA/100 cmd 0x000101f0 ctl 0x000103f6 bmdma 0x0001ff00 irq 14
ata2: PATA max UDMA/100 cmd 0x00010170 ctl 0x00010376 bmdma 0x0001ff08 irq 15
```

Как видите, IDE-контроллер слабенький, потому что поддерживает максимум UDMA/100 (более мощные поддерживают UDMA/133).

Первое IDE-устройство — это жесткий диск Western Digital емкостью 160 Гбайт:

```
ata1.00: ATA-7: WDC WD1600JB-00REA0, 20.00K20, max UDMA/100
ata1.00: 312581808 sectors, multi 16: LBA48
```

Второе IDE-устройство — привод DVD-RW производства LG:

```
ata1.01: ATAPI: HL-DT-ST DVDRAM GSA-4167B, DL11, max UDMA/33
```

Первичный мастер (primary master) настроен на использование UDMA/100 (максимум, что можно выжать из контроллера), а первичный подчиненный (primary slave) настроен на UDMA/33:

```
ata1.00: configured for UDMA/100
ata1.01: configured for UDMA/33
```

Подмонтирована корневая файловая система типа ext3 с последовательным (ordered) режимом работы журнала:

```
EXT3-fs: mounted filesystem with ordered data mode.
SELinux:  Disabled at runtime.
SELinux:  Unregistering netfilter hooks
```

Загружен драйвер сетевой платы Gigabit Ethernet производства Realtek (RTL):

```
r8169 Gigabit Ethernet driver 2.2LK-NAPI loaded
ACPI: PCI Interrupt 0000:02:00.0[A] -> GSI 19 (level, low) -> IRQ 16
PCI: Setting latency timer of device 0000:02:00.0 to 64
eth0: RTL8168b/8111b at 0xf8862000, 00:19:db:c7:e1:b5, XID 38000000 IRQ 16
```

Добавлен своп-раздел на /dev/sda7, размер раздела 530 104 Кбайт:

```
Adding 530104k swap on /dev/sda7.  Priority:-1 extents:1 across:530104k
```

После завершающего сообщения ядра запускается программа инициализации системы — в случае с Fedora программа /sbin/init, в других дистрибутивах может использоваться другая программа.

13.2. Параметры ядра

Параметры ядра позволяют управлять поведением ядра. Как уже говорилось, мы можем передать параметры ядра непосредственно при загрузке, используя меню загрузчика, или же прописать параметры ядра в файлах конфигурации загрузчика. Первый случай подходит для "одноразового" использования того или иного параметра, а второй — если параметр нужен для корректной работы системы. Поэтому, чтобы не указывать его каждый раз при загрузке Linux, намного проще прописать его в файле конфигурации загрузчика.

Если вы используете GRUB/GRUB2, то передать параметры ядра можно так: сначала надо выбрать образ (метку), а затем нажать клавишу <e> (рис. 13.1). Появится строка, в которой вы можете отредактировать параметры ядра, указанные в файле конфигурации загрузчика (рис. 13.2).

В случае с LILO (правда, сомневаюсь, что вы уже где-то встретите этот загрузчик) нужно нажать клавишу <Esc>, после чего вы перейдете в текстовый режим, где увидите список меток и приглашение boot, например:

```
Linux
Linux-failsafe
boot:
```

Чтобы передать ядру Linux параметры, вы должны ввести их в следующем формате:

```
метка параметры
```

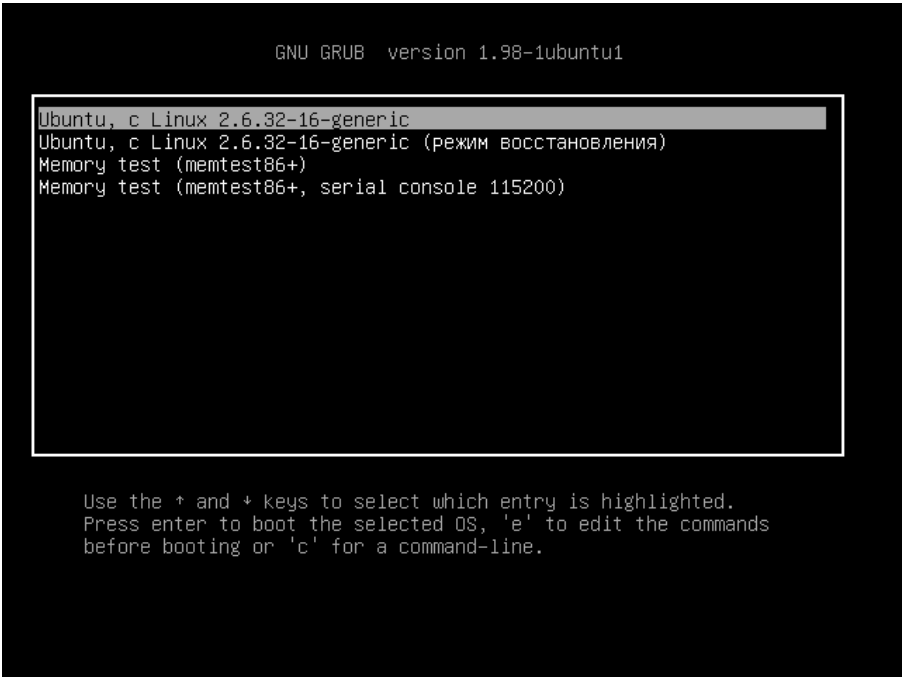


Рис. 13.1. Выбор метки (GRUB2)

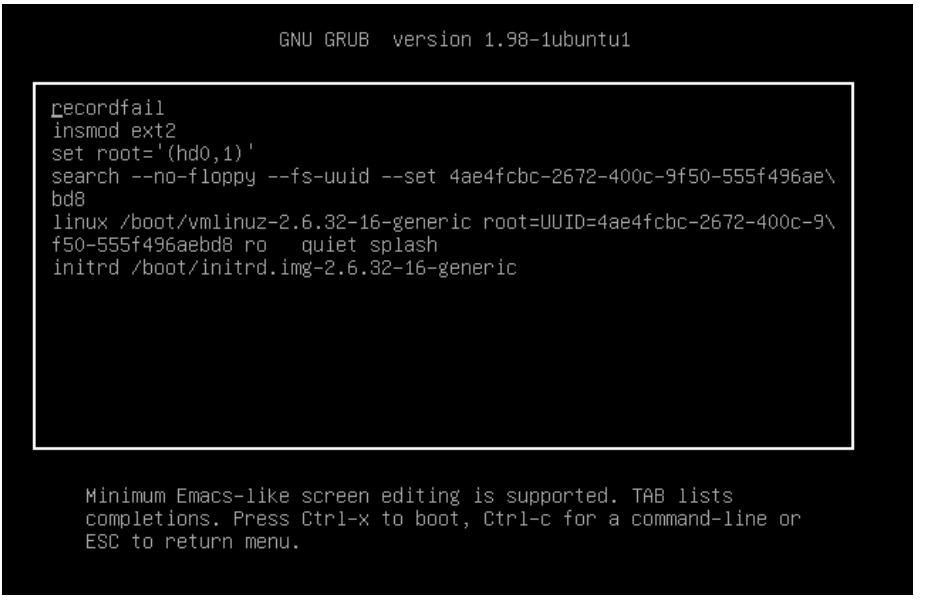


Рис. 13.2. Редактирование параметров ядра (GRUB2)

Например: `Linux noapic`.

О том, как сохранить параметры ядра в файле конфигурации загрузчика, было сказано в *главе 10*.

Параметров ядра очень много, поэтому в табл. 13.1 собраны самые полезные.

Таблица 13.1. *Некоторые параметры ядра Linux*

Параметр	Описание
`root=устройство`	Позволяет указать корневую файловую систему. Например, `root=/dev/hda5`
`ro`	Монтирует корневую файловую систему в режиме "только чтение". Используется по умолчанию. После проверки файловой системы программой `fsck` корневая файловая система перемонтируется в режим `rw`
`rw`	Монтирует корневую файловую систему в режиме "чтение/запись". При использовании этого параметра нельзя запускать программы типа `fsck`. Перед запуском `fsck` нужно перемонтировать корневую файловую систему в режиме `ro`
`mem=`	Определяет объем памяти, установленной в компьютере. Иногда ядро неправильно определяет объем оперативной памяти. Вы можете помочь ему в этом, указав параметр `mem`. Только указывать его нужно правильно, например: `mem=768M` После числа обязательно должна следовать буква `M`, иначе ядро "подумает", что объем оперативной памяти 768 байт

Таблица 13.1 *(окончание)*

Параметр	Описание
`init=`	Позволяет задать программу инициализации. По умолчанию используется программа /sbin/init, но вы можете задать другую программу
`reboot=`	Позволяет задать тип перезагрузки компьютера. Возможные значения: `cold` и `warm`, т. е. "холодная" или "горячая" перезагрузка
`single`	Однопользовательский режим для администрирования системы, например, в случае отказа
`nodmraid`	Отключает программные RAID-массивы, организованные на уровне BIOS
`noapic`	Полезен, если вы при загрузке увидите сообщение: **`kernel panic — not syncing: IO-APIC + timer doesn't work!`** Подробнее об этом параметре вы можете прочитать по адресу: **`http://www.dkws.org.ua/phpbb2//viewtopic.php?topic=2973&forum=5`**
`nopcmcia`	Отключает PCMCIA-карты (для ноутбуков). Полезен, если вы подозреваете, что у вас проблемы с PCMCIA-картой
`nodma`	Отключается DMA (Direct Memory Access, прямой доступ к памяти) для всех IDE-устройств
`noapm`	Отключает APM (Advanced Power Management) — расширенное управление питанием
`nousb`	Отключает поддержку USB
`noscsi`	Отключает поддержку SCSI
`pci=noacpi`	Не использовать ACPI для управления PCI-прерываниями
`apci=off`	Полностью отключает ACPI (Advanced Configuration and Power Interface). Полезен на некоторых ноутбуках, когда не удается установить (а потом загрузить) Linux
`edd=off`	Отключает EDD (Enhanced Disk Drive). Если при загрузке Linux вы видите сообщение **`Probing EDD`** и загрузка на этом останавливается, тогда вам поможет параметр ядра `edd=off`

Примечание

С дополнительными параметрами ядра вы можете ознакомиться по адресу: **http://dkws.org.ua/phpbb2/viewtopic.php?t=3031**.

13.3. Компиляция ядра

Linux, в отличие от многих других операционных систем, позволяет обычному пользователю проникнуть в святая святых — в собственное ядро. Любой желающий может загрузить исходные коды ядра и откомпилировать ядро операционной системы.

Вообще, перекомпиляция ядра — весьма специфическая операция. Раньше ее нужно было делать довольно часто — практически каждый Linux-пользователь со стажем хотя бы раз в жизни перекомпилировал ядро. Зачем? Например, чтобы включить дополнительные функции. Или, наоборот, выключить поддержку некоторых устройств и некоторые ненужные функции — так ядро окажется компактнее и система будет работать быстрее.

Сейчас я даже не знаю, зачем может понадобиться перекомпиляция ядра. Это настолько редкая операция в наше время, что даже исходные коды ядра перестали поставляться на дистрибутивных дисках. Исключение составляет дистрибутив Mandriva 2010 — один из немногих, на дистрибутивном диске которого до сих пор есть исходники ядра. Вот на его примере мы и будем рассматривать компиляцию ядра — ну не хочется мне "тянуть" исходные коды (более 53 Мбайт) с **www.kernel.org**.

13.3.1. Установка исходных кодов ядра

Запустите программу `drakrpm` и в группе **Разработка | Ядро** найдите и выберите пакет **kernel-source** (рис. 13.3). Программа `drakrpm` предложит вам установить дополнительные пакеты — соглашайтесь (рис. 13.4).

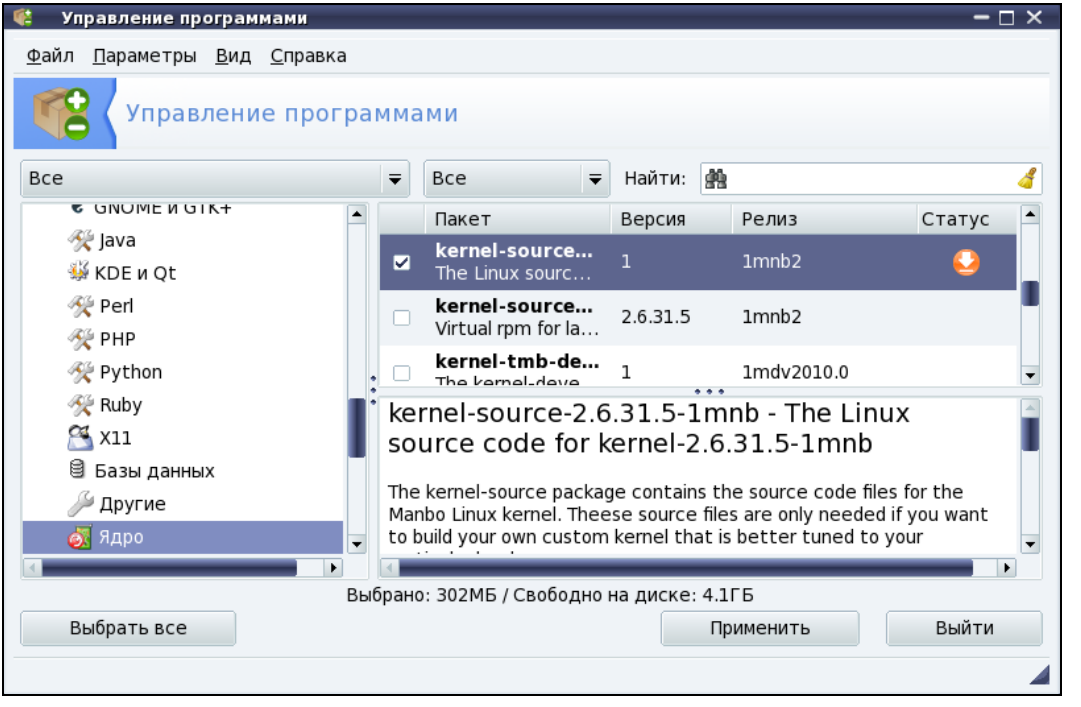


Рис. 13.3. Установка пакета kernel-source

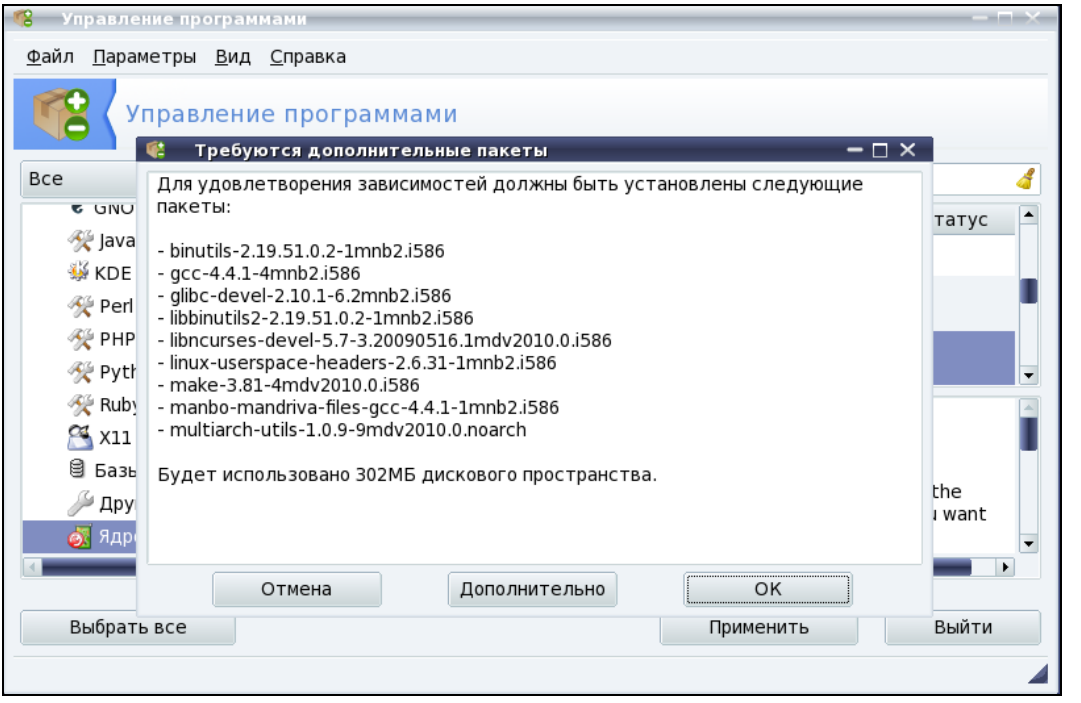


Рис. 13.4. Установка дополнительных пакетов

13.3.2. Команда *make menuconfig*: настройка ядра

После установки исходников перейдите в каталог /usr/src/linux:

```
# cd /usr/src/linux
```

Теперь введите одну из двух команд:

```
# make menuconfig
# make xconfig
```

Эти команды предназначены для вызова конфигуратора ядра — с его помощью вы можете включить/выключить функции ядра. Мне больше нравится конфигуратор `menuconfig` (рис. 13.5) — его можно запускать как в консоли, так и в терминале (если вы предпочитаете графический режим). А `xconfig` можно запускать только в графическом режиме.

> ***Совет***
>
> Графический конфигуратор `xconfig` основан на библиотеке Qt, поэтому он будет работать, если у вас установлена графическая среда KDE. Если вы используете GNOME, то вам нужно запустить конфигуратор `gconfig`, который основан на библиотеке GTK. А еще проще: используйте `menuconfig` — он будет работать в любом случае.

Ядро не компилируют без определенной цели. Скорее всего, вы знаете, какую именно опцию ядра вам нужно включить или, наоборот, выключить. Но если вы

хотите откомпилировать ядро эксперимента ради, в качестве путеводителя послужит табл. 13.2, в которой описаны основные разделы опций ядра.

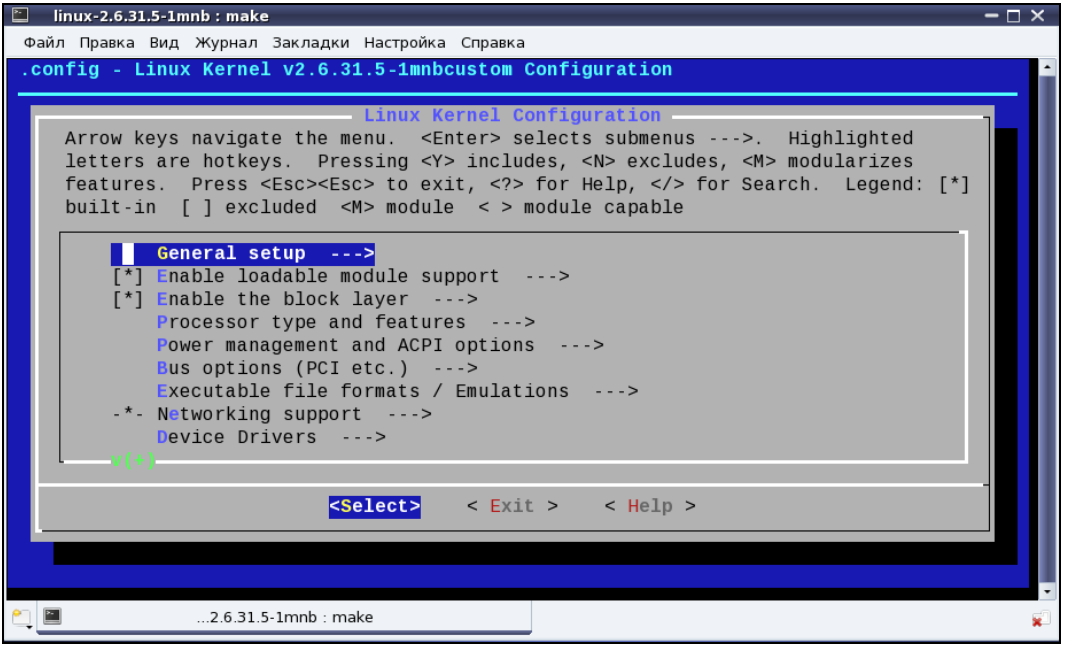


Рис. 13.5. Конфигуратор `make menuconfig`

Таблица 13.2. *Разделы опций ядра*

Раздел	Описание
General setup	Общие параметры, например поддержка своп-памяти, межпроцессного взаимодействия System V, Sysctl. Если не знаете, для чего нужна та или иная опция, выделите ее и нажмите клавишу <F1>. А уж если не знаете английский, то до его изучения лучше опции не выключать!
Enable loadable module support	Поддержка загружаемых модулей. Драйверы устройств в Linux разработаны в виде модулей ядра. Здесь вы можете указать, нужна ли вам поддержка модулей. Обычно отключать поддержку модулей на обычных машинах не рекомендуется. Если же вы хотите построить мало обслуживаемый сервер, работающий по принципу "построил и забыл", отключение поддержки загружаемых модулей позволит даже повысить безопасность сервера, поскольку злоумышленник не сможет добавить свой код в ядро путем загрузки модуля. Однако в этом случае ядро будет очень громоздким, потому что вам придется все нужные вам функции, которые были реализованы в виде модулей, компилировать в ядро
Enable the block layer	В этом разделе вы можете включить поддержку больших блочных устройств размером более 2 Тбайт

Таблица 13.2 *(окончание)*

Раздел	Описание
Processor type and features	Здесь вы можете выбрать тип вашего процессора и включить/выключить различные функции процессора
Power management and ACPI options	Опции управления питанием (ACPI, APM)
Bus options	Здесь вы можете включить/выключить поддержку различных системных шин, а также определить их функции
Executable file formats / Emulations	Параметры поддержки форматов исполнимых файлов
Networking support	Сетевые опции ядра
Device drivers	Драйверы устройств. Здесь вы можете определить, какие устройства должна поддерживать ваша система, а какие — нет
Firmware drivers	Драйверы микропрограммного обеспечения (поддержка различных BIOS)
File systems	Здесь вы можете определить, какие файловые системы должна поддерживать ваша система, а какие — нет
Kernel Hacking	Различные параметры, относящиеся непосредственно к ядру
Security options	Параметры безопасности
Cryptographic API	Параметры криптографии (поддержка различных алгоритмов шифрования данных)
Virtualization	Параметры виртуализации
Library routines	Поддержка различных библиотечных функций (но если заглянуть в этот раздел, то вы увидите, что все эти функции связаны с вычислением контрольной суммы CRC)
Unofficial 3rd party kernel additions	Неофициальные дополнения ядра (от посторонних разработчиков). Сюда разработчики дистрибутива (не ядра) могут включать дополнительные модули. Вообще будьте осторожны, потому что среди них могут быть экспериментальные функции, включение которых отрицательно отразится на стабильности системы

Опции ядра могут быть либо включены, либо выключены. Если опция выключена, то ее код исключается из ядра (при компиляции ядра он не будет учитываться). Если же опция включена, то ее код будет включен в состав ядра. Но есть еще третье состояние опции — M. Это означает, что опция будет включена в ядро как модуль. После сборки ядра и модулей все опции, скомпилированные в режиме M, будут "лежать" на диске, пока не понадобятся ядру. А как только это произойдет, будет загружен нужный модуль. Вообще о модулях мы уже говорили, поэтому не будем повторяться.

При выходе из конфигуратора ядра он спросит вас, хотите ли вы сохранить изменения конфигурации ядра (рис. 13.6)? Конечно, хотим!

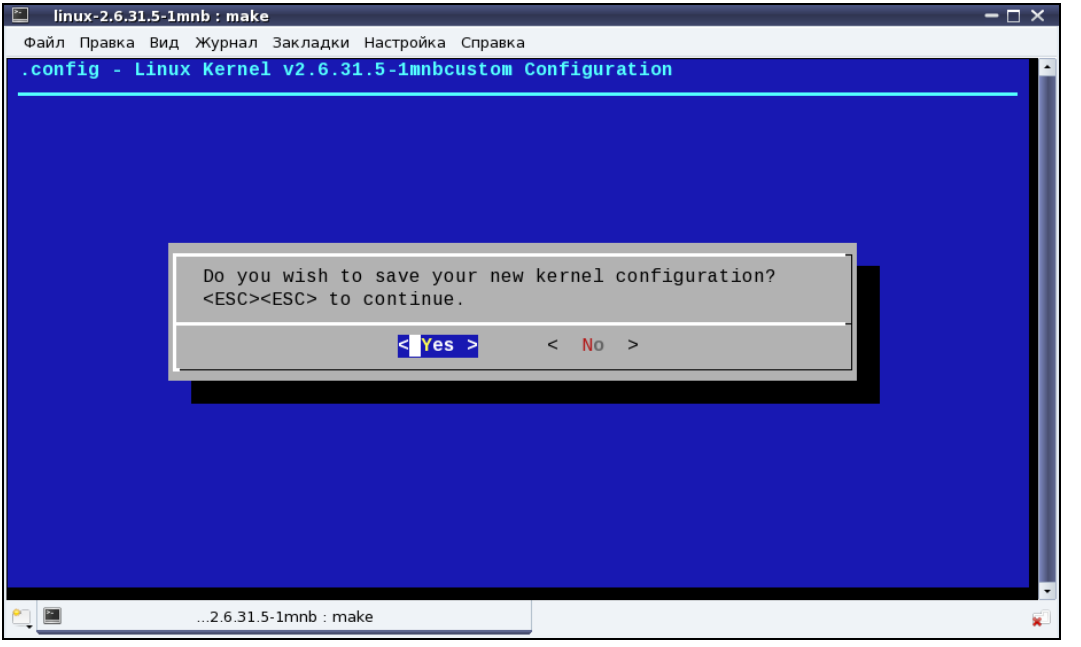


Рис. 13.6. Сохранить изменения?

13.3.3. Команды компиляции ядра

После настройки ядра конфигуратор сообщит, что для построения ядра нужно ввести команду `make` (рис. 13.7), а для вывода справки — `make help`.

Спешить с вводом `make` не будем — ее ввести мы успеем всегда. Лучше введите `make help` для ознакомления с параметрами команды `make`. Если вы внимательно прочитаете вывод команды `make help`, то узнаете, что команда `make` (без параметров) аналогична команде `make all`, которая соберет следующие цели (отмеченные `*`):

- `vmlinuz` — собирает обычное, "большое" ядро;
- `modules` — собирает модули ядра;
- `bzImage` — собирает сжатое ядро, которое помещается в каталог arch/i386/boot.

Теперь понятно, что мы можем просто ввести команду `make` — она проделает как раз то, что нам нужно: `# make` (рис. 13.8).

Примечание

Компиляция ядра (рис. 13.8) — довольно утомительный процесс. Нет, вам ничего делать не нужно, но именно это и утомляет, поэтому можете смело отойти от компьютера и выпить чашечку чая или кофе — не волнуйтесь, вы успеете перекусить к моменту завершения сборки ядра. На среднем компьютере время выполнения этой команды составляет 1–2 часа (этого хватит, чтобы и в магазин сходить, а не только кофе выпить). Один раз я рискнул запустить эту команду в эмуляторе (в виртуальной машине VMware) — на выполнение операции понадобилось 5 часов.

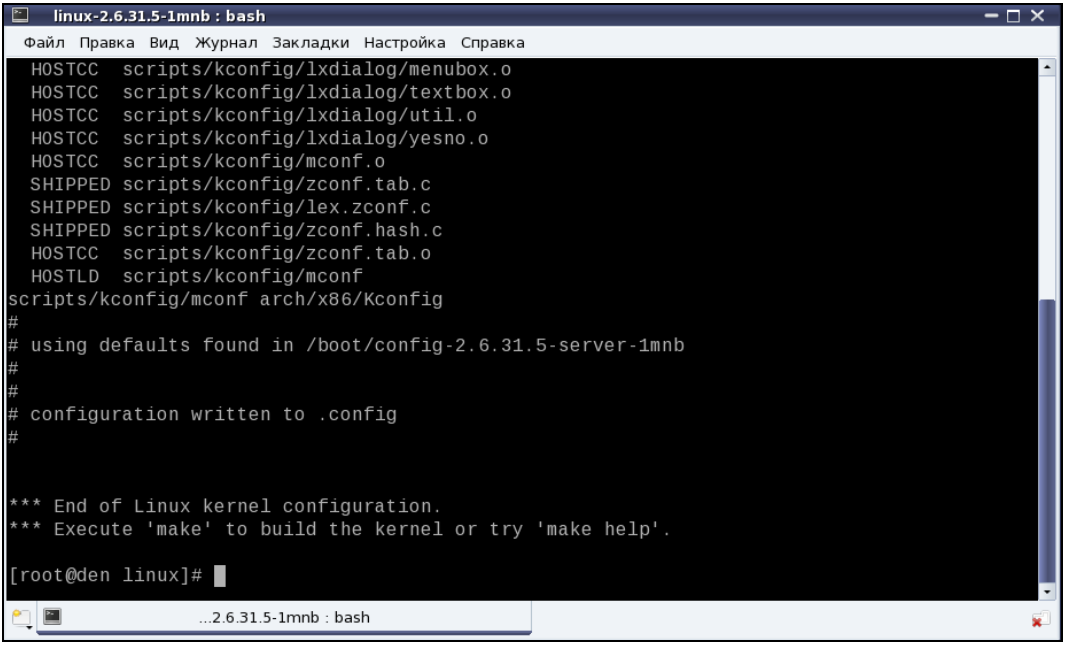


Рис. 13.7. Введите команду `make`

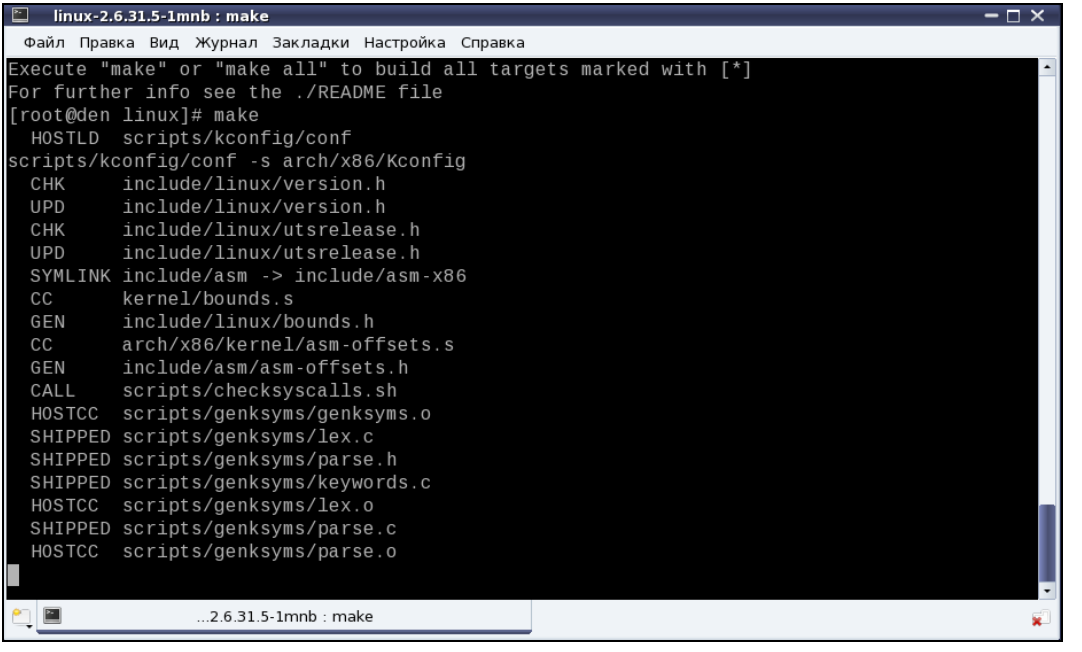


Рис. 13.8. Компиляция ядра

Но это еще не все. Мы только откомпилировали модули и ядро, но пока не устанавливали их. Для установки модулей введите команду:

```
# make modules_install
```

К счастью, данная операция занимает намного меньше времени.

После установки модулей следует установить ядро. Это можно сделать с помощью команды:

```
# make install
```

В процессе установки (рис. 13.9) будет добавлена соответствующая метка в файл конфигурации загрузчика. После этого следует ввести команду `reboot` для перезагрузки.

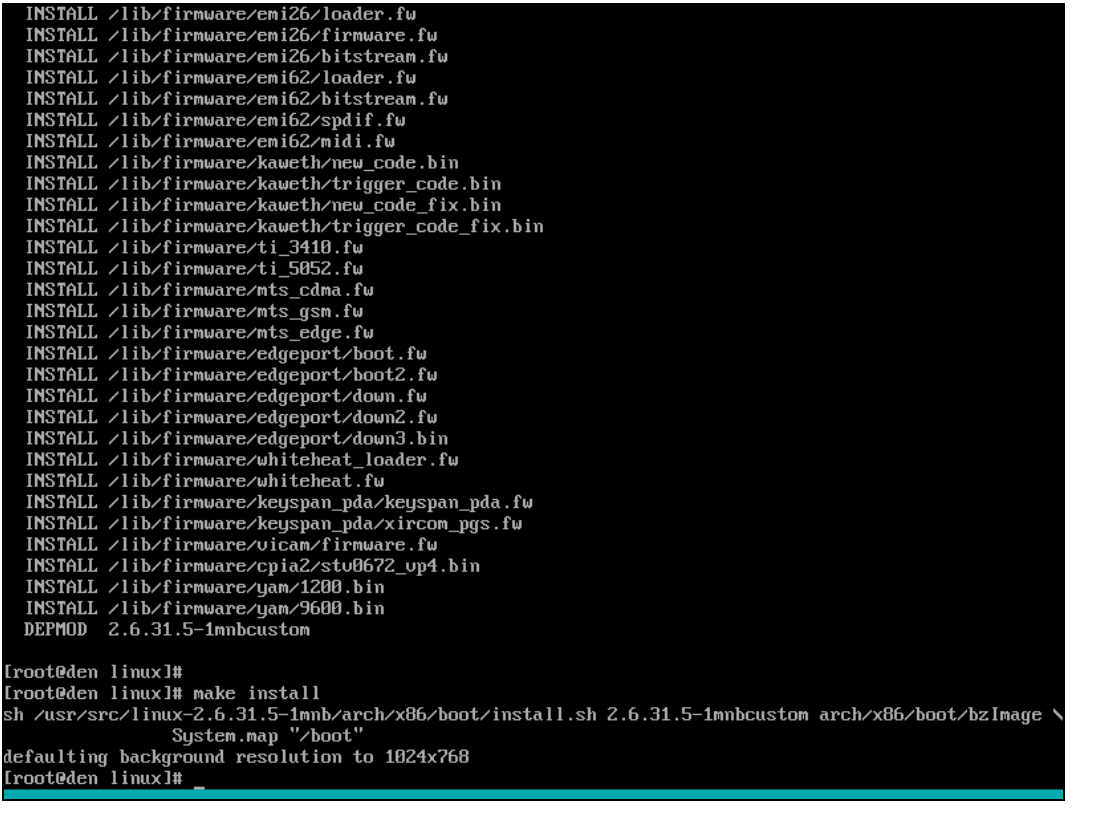


Рис. 13.9. Установка модулей и ядра (`make install`)

При перезагрузке нужно выбрать метку custom_версия_ядра, например **Linux с ядром custom_2.6.31.5-1mnbcustom** (рис. 13.10). Как только система загрузится, убедитесь, что все нормально работает.

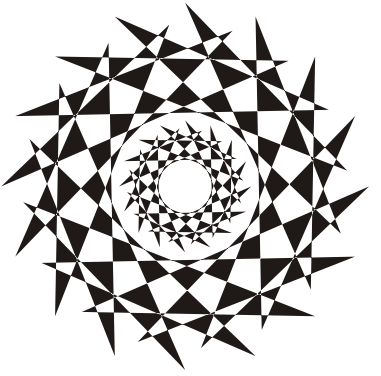
В процессе компиляции ядра создается много ненужных после его завершения файлов (у меня такого "мусора" насобиралось на 3 Гбайт), а команда `make clean`

позволяет весь этот "мусор" удалить. Так что в заключение произведем "генеральную уборку" — откройте терминал и от имени root введите команды:

```
# cd /usr/src/linux
# make clean
```

Поздравляю! Вы успешно справились с перекомпиляцией ядра.

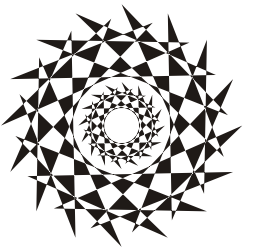
Рис. 13.10. Выбор нового ядра при загрузке



ЧАСТЬ III

Программирование в Linux

Часть III посвящена командам для программиста. В *главе 14* мы рассмотрим все команды, относящиеся к программированию в Linux на языке C. Конечно, сам язык C мы рассматривать не будем — этому посвящены другие книги. Но зато мы рассмотрим программирование на языках командных оболочек `bash` *(глава 15)* и `tcsh` *(глава 16)*, а также язык обработки текста gawk *(глава 17)*.



Глава 14

Программирование на языке C. Утилиты для программиста

14.1. Команда *gcc*: компилятор

Для компиляции программ на языке C используется компилятор `gcc`, который установлен по умолчанию в большинстве дистрибутивов. Давайте проверим, установлен ли `gcc` в вашей системе. Введите команду:

```
gcc --version
```

Если в ответ вы получите сообщение, что команда не найдена, значит, нужно установить `gcc`. Установке программ в Linux посвящена вся *часть IV*. В ней вы обязательно найдете инструкции по установке программ в вашем дистрибутиве.

Попробуем откомпилировать простейшую программу. Создайте файл hello.c (листинг 14.1).

Листинг 14.1. Файл hello.c

```
#include <stdio.h>

int main(int argc, char** argv) {

    printf("Hello, world!\n");

    return 0;

}
```

Теперь введите команду:

```
gcc -o hello hello.c
```

Параметр `-o` задает имя исполнимого файла, который должен создать компилятор (по умолчанию, если имя не задано, будет создан файл a.out). После компиляции программы в нашем случае будет создан файл hello, который можно запустить так:

```
./hello
```

Вы увидите знакомое всем программистам приветствие:

```
Hello, world!
```

У компилятора `gcc` очень много параметров, их можно разбить на следующие группы:

- общие параметры;
- параметры языка;
- параметры препроцессора;
- параметры компоновщика;
- параметры каталогов;
- параметры отладки;
- параметры оптимизации.

К общим параметрам относятся параметры `-x`, `-c`, `-v` и `-o`. С параметром `-o` мы уже знакомы — он задает имя результирующего файла. Параметр `-v` полезен начинающим программистам, которые хотят больше знать о процессе компиляции или же компиляции сложных проектов, чтобы получить больше информации о самом процессе.

Параметр `-x` задает тип языка программы, например:

```
gcc -x c -o hello hello.c
```

В качестве типа языка программы можно указать:

- `c` — самый обычный язык C;
- `objective_c` — язык Objective C, не нужно путать с C++;
- `c++` — это значение нужно указать, если программа написана на C++;
- `assembler` — для программ, написанных на языке Ассемблера.

Вообще, указывать тип языка необязательно, поскольку в большинстве случаев компилятор сам определит его.

Параметр `-c` полезен, если вы хотите только откомпилировать вашу программу, но не вызывать компоновщик. В результате будет создан объектный файл с "расширением" .o. Процесс компиляции состоит из двух основных этапов: трансляции, когда создается объектный файл, и компоновки, когда создается исполняемый файл. Зачем вам нужен объектный файл, вы должны знать сами. Новичкам, которые только-только начинают программировать на C, рекомендую ознакомиться с процессом компиляции более подробно:

http://ru.wikipedia.org/wiki/Компилятор

Из всех параметров языка мне пригодился только параметр `ANSI`, который отключает все функции GNU C, несовместимые со стандартом ANSI C. Данный параметр пригодится при компиляции некоторых программ, написанных со строгим соблюдением стандарта ANSI:

http://ru.wikipedia.org/wiki/ANSI_C

А вот среди параметра препроцессора нашлось сразу три полезных параметра:

- `-include <файл>` — позволяет откомпилировать указанный файл раньше, чем все остальные, иногда это полезно;

- `-nostdinc` — запрещает использовать системный каталог с заголовками. При этом компилятор будет искать заголовки в каталогах, указанных с помощью параметра `-I`;
- `-nostdinc++` — то же самое, но для заголовков языка C++.

Самым важным параметром компоновщика является параметр `-l`, позволяющий указать явно библиотеку, которая будет использоваться при сборке вашей программы. Например:

```
gcc -lmy_lib hello.c
```

Кроме этого вы также можете использовать параметры `-nostdlib` и `-static`. Первый запрещает использовать все стандартные библиотеки, при этом будут использоваться библиотеки из каталогов, которые заданы параметром `-L`. Второй параметр задает статическую линковку, учтите, что размер результирующего файла существенно увеличится.

К параметрам каталогов относятся уже знакомые нам параметры `-I` и `-L`. Первый задает каталог с файлами заголовков, а второй — каталог с библиотеками.

К параметрам отладки относится параметр `-g`, добавляющий отладочную информацию для отладчика `gdb`. Вы можете отлаживать свои программы, только если добавили отладочную информацию с помощью параметра `-g`.

Компилятор `gcc` может оптимизировать ваши программы, в результате они будут либо быстрее запускаться, либо размер исполнимого файла станет меньше. Поэкспериментируйте с параметрами `-O1`, `-O2`, `-O3` — это различные уровни оптимизации.

14.2. Команда *make*: сборка проекта

Серьезные проекты состоят не из одного С-файла, а, как минимум, из нескольких. Чтобы не вводить команды компиляции несколько раз, программисты используют специальные файлы с именем Makefile.

В файле Makefile содержится список файлов, которые нужно откомпилировать и опции компилятора для каждого файла. Для нашей простой программы создайте Makefile с содержимым из листинга 14.2.

Листинг 14.2. Файл Makefile

```
hello: hello.c
<TAB>  gcc -o hello hello.c
```

Первая строка — это описание файла morning.c. Вторая строка — команда для компиляции этого файла. Обратите внимание, что перед командой `gcc` должен быть символ табуляции. Именно символ табуляции, а не 8 пробелов!

Формат Makefile следующий:

```
цель1: список_файлов
    последовательность команд
```

```
цельN: список_файлов
    последовательность команд
```

В Makefile вы можете использовать макроопределения, например:

```
CC=gcc
PATH=/usr/include /usr/my/include
FLAGS=-O3 -Wall -I${PATH}
…
$(CC) $(FLAGS) -c hello.c
```

К макроопределению можно обратиться с помощью конструкции `$(имя)`.

После создания Makefile (он должен находиться в одном каталоге с файлом hello.c) для компиляции нашего проекта уже не нужно вводить команду с параметрами компилятора, достаточно просто ввести команду:

```
make
```

Параметры команды `make` представлены в табл. 14.1.

Таблица 14.1. *Параметры команды* `make`

Параметр	Описание
`-C каталог`	Перейти в указанный каталог перед началом сборки программы
`-d`	Вывести отладочную информацию
`-f файл`	Использовать указанный файл вместо Makefile
`-i`	Игнорировать ошибки компилятора
`-l каталог`	В указанном каталоге будет произведен поиск файлов, указанных в Makefile
`-k`	Продолжить работу после ошибки, если это возможно
`-o файл`	Пропустить указанный файл
`-s`	Тихий режим. Не выводить команды перед их выполнением
`-w`	Показывать текущий каталог до и после выполнения каждой команды

14.3. Команды из пакета binutils

В пакете binutils вы найдете несколько полезных утилит (табл. 14.2). Подробно мы их описывать не будем, вы сами сможете прочитать справку по этим командам, если они вас заинтересуют.

Таблица 14.2. *Команды из пакета binutils*

Параметр	Описание
`ld`	Компоновщик (создает из объектного файла исполняемый)
`nm`	Выводит названия идентификаторов из двоичных файлов

Таблица 14.2 *(окончание)*

Параметр	Описание
`objcopy`	Копирует и транслирует двоичные файлы
`objdump`	Выводит информацию из двоичных файлов
`size`	Выводит размер секций двоичного файла
`strings`	Выводит печатаемые строки из двоичных файлов

Лично мне пригодилась утилита `strings`, она подходит для поиска строк как в двоичных файлах, так и на разделах жесткого диска в случае повреждения таблицы разделов. Например:

```
strings /dev/sdaX | grep 'искомая строка'
```

14.4. Другие полезные команды

Как программисту, вам могут понадобиться следующие команды:

- `strip` — удаляет таблицу символов из объектного файла, что позволяет сократить размер этого файла, правда, отладить этот файл уже не получится;
- `gprof` — профайлер для ваших программ. Профайлер — это программа, позволяющая оптимизировать выполнение вашей программы, для каждого вызова функции профайлер выводит время ее выполнения, что позволяет проанализировать, где происходит задержка выполнения, и оптимизировать код программы при необходимости.

14.5. Команда *gdb*: отладка программ

Ошибки в программах можно разделить на две большие группы: синтаксические и логические. Первые исправляются еще при компиляции, а вторые — это ошибки алгоритма, вычислить такие ошибки бывает нелегко. Ведь с точки зрения синтаксиса все правильно, но программа делает не то, что вам хочется. Как раз для таких случаев и предназначены отладчики. В Linux используется отладчик `gdb`.

Формат вызова отладчика такой:

```
gdb    [-help] [-nx] [-q] [-batch] [-cd=dir] [-f] [-b bps] [-tty=dev]
       [-s symfile] [-e prog] [-se prog] [-c core] [-x cmds] [-d dir]
       [prog[core|procID]]
```

Описание команд `gdb` приведено в табл. 14.3.

Чтобы отладчик `gdb` мог отлаживать программу, нужно добавить служебную информацию к исполнимому файлу. Как правило, это делается при компиляции программы (параметр `-g`):

```
gcc -g -o hello hello.c
```

После этого вызываем отладчик так:

```
gdb hello
```

Таблица 14.3. *Параметры* `gdb`

Параметр	Описание
`-help`	Показать краткое описание всех параметров `gdb`
`-nx`	Запрет обработки команд файла инициализации .dgbinit
`-q`	Не показывать приветствие и информацию об авторских правах
`-batch`	Командный режим. Если отладчик возвращает 0, то были выполнены все команды, которые указаны в файле, заданном параметром `-x`. Если же были ошибки, отладчик возвращает ненулевое значение
`-f`	Используется при отладке программ с помощью интерфейса редактора `Emacs`
`-b bps`	Устанавливает скорость обмена (в битах в секунду) информации по последовательному интерфейсу, если вы отлаживаете программу удаленно. Вряд ли вы будете сейчас использовать эту опцию
`-tty=терминал`	Использовать указанный терминал в качестве стандартного вывода отладчика и отлаживаемой программы
`-s файл`	Читает таблицу символов из указанного файла
`-write`	Разрешить запись в исполняемые файлы
`-e программа`	Использовать указанную программу как фильтр дампа
`-se`	Считать таблицу символов из указанного файла, этот файл будет использоваться в качестве исполняемого
`-core=файл`	Указать файл дампа
`-x файл`	Выполнить указанные в файле команды, используется вместе с параметром `-batch`
`-d каталог`	Добавить указанный каталог к списку каталогов исходного текста
`[progcore|procID]`	Отлаживаемый объект: программа (prog), файл дампа (core), идентификатор процесса (procID) при отладке уже запущенной программы
`-p PID`	Подключиться к уже запущенному процессу

Если же программа запустилась и завершилась с ошибкой, будет создан файл дампа (core), его тоже можно "скормить" отладчику:

```
gdb hello core
```

Можно даже подключиться к уже запущенному процессу, для этого нужно знать PID этого процесса (его можно узнать с помощью команд `ps` и `top`):

```
gdb 1234
```

Процесс отладки программы осуществляется путем ввода команд отладчика. Самые важные команды отладчика представлены в табл. 14.4.

Таблица 14.4. *Команды* `gdb`

Команда	Описание
`break [файл:]функция`	Установить точку останова (breakpoint)
`run [параметры]`	Запускает программу и передает ей указанные параметры
`bt`	Обратная трассировка программы
`print выражение`	Вывести значение выражения. В выражении можно использовать переменные из вашей программы
`C`	Продолжить выполнение программы после остановки
`next`	Выполнить следующую строку программы. Если следующая строка — вызов функции, то она будет выполнена за один шаг
`step`	Выполнить следующую строку программы. Если следующая строка — вызов функции, то будут выполнены пошагово все операции функции. Команду `next` можно использовать, если вы точно уверены, что все ваши функции корректно работают. А вот `step` используется для более глубокой отладки программы
`help [имя]`	Выводит общую справку или справку по указанной команде
`quit`	Выход

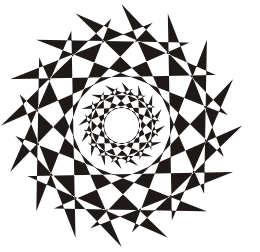
Несколько слов о самом процессе отладки. Сначала нужно запустить `gdb` и загрузить в него программу. Затем нужно установить точку останова, например:

```
break main
```

Далее запускаем программу:

```
run
```

Выполнение программы дойдет до функции `main()` и остановится. Затем с помощью команд `next` и `step` вы запускаете пошаговое выполнение программы, а с помощью команды `print` просматриваете значение ее переменных.



Глава 15

Командный интерпретатор *bash*

15.1. Настройка *bash*

`bash` — это самая популярная командная оболочка (командный интерпретатор) Linux. Основное предназначение `bash` — выполнение команд, введенных пользователем. Пользователь вводит команду, `bash` ищет программу, соответствующую команде, в каталогах, указанных в переменной окружения `PATH`. Если такая программа найдена, то `bash` запускает ее и передает ей введенные пользователем параметры. В противном случае выводится сообщение о невозможности выполнения команды.

Файл /etc/profile содержит глобальные настройки `bash`, он влияет на всю систему — на каждую запущенную оболочку. Обычно /etc/profile не нуждается в изменении, а при необходимости изменить параметры `bash` редактируют один из файлов:

- ~/.bash_profile — обрабатывается при каждом входе в систему;
- ~/.bashrc — обрабатывается при каждом запуске дочерней оболочки;
- ~/.bash_logout — обрабатывается при выходе из системы.

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

```
source ~/.bashrc
```

Данная строка означает, что нужно прочитать файл .bashrc. Поэтому будем считать основным конфигурационным файлом файл .bashrc. Но помните, что он влияет на оболочку текущего пользователя (такой файл находится в домашнем каталоге каждого пользователя, не забываем: "~" означает домашний каталог). Если же вдруг понадобится задать параметры, которые повлияют на всех пользователей, то нужно редактировать файл /etc/profile.

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

Какие настройки могут быть в .bashrc? Как правило, в этом файле задаются псевдонимы команд, определяется внешний вид приглашения командной строки, задаются значения переменных окружения.

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

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

Псевдонимы работают просто: при вводе команды `l` на самом деле будет выполнена команда `ls -l`.

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

```
пользователь@компьютер:рабочий_каталог
```

Значение `PS1` при этом будет:

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

В табл. 15.1 приведены допустимые модификаторы командной строки.

Таблица 15.1. *Модификаторы командной строки*

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

Таблица 15.1 *(окончание)*

Модификатор	Описание
`\$`	Если UID пользователя равен 0, будет выведен символ #, иначе — символ $
`\\`	Обратный слэш
`$ ( )`	Подстановка внешней команды

Вот пример альтернативного приглашения командной строки:

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

Вид приглашения будет такой:

```
[denis@host] 12/06/10 $
```

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

Установить переменную окружения можно с помощью команды `export`, что будет показано позже.

15.2. Автоматизация задач с помощью *bash*

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

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

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

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

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

15.3. Привет, мир!

По традиции напишем первый сценарий, выводящий всем известную фразу: "Привет, мир!" (Hello world!). Для редактирования сценариев вы можете использовать любимый текстовый редактор, например `nano` или `ee` (листинг 15.1).

Листинг 15.1. Первый сценарий

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

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

```
# Комментарий
```

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

```
$ chmod +x hello
```

Для запуска сценария введите команду:

```
./hello
```

На экране вы увидите строку:

```
Привет, мир!
```

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

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

15.4. Использование переменных в собственных сценариях

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

Для объявления переменной используется следующая конструкция:

```
переменная=значение
```

Пример объявления переменной:

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

Обратите внимание на следующие моменты:

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

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

```
read ADDRESS
```

или со стандартного вывода программы:

```
ADDRESS='hostname'
```

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

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

- `BASH` — полный путь до исполняемого файла командной оболочки `bssh`;
- `BASH_VERSION` — версия bash;
- `HOME` — домашний каталог пользователя, который запустил сценарий;
- `HOSTNAME` — имя компьютера;
- `RANDOM` — случайное число в диапазоне от 0 до 32 767;
- `OSTYPE` — тип операционной системы;
- `PWD` — текущий каталог;
- `PS1` — строка приглашения;
- `UID` — ID пользователя, который запустил сценарий;
- `USER` — имя пользователя.

Для установки собственной переменной окружения используется команда `export`:

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

15.5. Передача параметров сценарию

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

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

Рассмотрим небольшой пример обработки параметров сценария. Я понимаю, что конструкцию `case-esac` мы еще не рассматривали, но общий принцип должен быть понятен (листинг 15.2).

Листинг 15.2. Пример обработки параметров сценария

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

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

Думаю, приведенных комментариев достаточно, поэтому подробно рассматривать работу сценария из листинга 15.2 не будем.

15.6. Массивы и *bash*

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

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

echo $ARRAY[0]
```

15.7. Циклы

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

Синтаксис цикла `for` выглядит так:

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

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

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

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

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

```
1
2
3
```

Синтаксис цикла `while` выглядит немного иначе:

```
while условие
do
команды
done
```

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

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

15.8. Условные операторы

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

```
if условие_1 then
    команды_1
elif условие_2 then
    команды_2
...
```

```
elif условие_N then
    команды_N
else
    команды_N+1
fi
```

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

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

```
# Переменная N = 10
[ N==10 ]

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

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

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

Применять данные выражения нужно следующим образом:

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

Например:

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

В квадратных скобках вы также можете задать выражения для проверки существования файла и каталога:

- `-e файл` — условие истинно, если файл существует;
- `-d каталог` — условие истинно, если каталог существует;
- `-x файл` — условие истинно, если файл является исполнимым.

С оператором `case` мы уже немного знакомы, но сейчас рассмотрим его синтаксис подробнее:

```
case переменная in
значение_1) команды_1 ;;
...
```

```
значение_N) команды_N ;;
*) команды_по_умолчанию;;
esac
```

Значение указанной переменной по очереди сравнивается с приведенными значениями (`значение_1 … значение_N`). Если есть совпадение, то будут выполнены команды, соответствующие значению. Если совпадений нет, то будут выполнены команды по умолчанию. Пример использования `case` был приведен в листинге 15.2.

15.9. Функции

В `bash` можно использовать функции. Синтаксис объявления функции следующий:

```
имя() { список; }
```

Вот пример объявления и использования функции:

```
list_txt()
{
echo "Выводим текстовые файлы"
ls *.txt
}
```

15.10. Примеры сценариев

15.10.1. Сценарий мониторинга журнала

Начнем с простого сценария мониторинга журнала (листинг 15.3). Системные журналы постоянно обновляются, а наш сценарий будет каждые 3 секунды выводить последние 15 строк выбранного вами журнала. Сценарий будет полезен при настройке системы, когда нужно постоянно просматривать журналы, чтобы понять, как система реагирует на новые настройки. Для прекращения работы сценария нужно нажать клавиши <Ctrl>+<C>.

Листинг 15.3. Мониторинг системного журнала

```
#!/bin/bash
# Интервал обновления, в секундах
INT=3

while [ true ]
do
# Выводим последние 15 строк журнала
tail -n 15 $1
# Ждем
```

```
sleep $INT
# Две пустые строки
echo; echo
done
```

Формат вызова следующий:

```
./сценарий файл_журнала
```

Например:

```
./script /var/log/messages
```

15.10.2. Переименование файлов

Следующий сценарий сложнее: он ищет файлы в текущем каталоге, в именах которых есть пробелы, и заменяет пробелы на символы подчеркивания (листинг 15.4).

Листинг 15.4. Сценарий rename_blanks

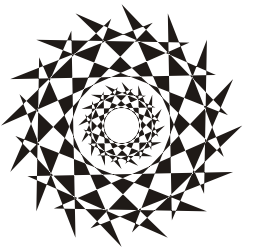
```
#! /bin/bash
#

num=0                          # Сколько файлов мы переименовали

for filename in *              # Перебираем все файлы в текущем каталоге
do
# Передаем имя файла фильтру grep
# Если имя файла содержит пробел, то код завершения
# последней операции равен 0
     echo "$filename" | grep -q " "
     if [ $? -eq 0 ]
     then
# Если код завершения равен 0, переименовываем
# файл
       fname=$filename
       n='echo $fname | sed -e "s/ /_/g"'
       mv "$fname" "$n"
       let "num += 1"
     fi
done

echo "Переименовано файлов: $num"

exit 0
```

15.10.3. Преобразование систем счисления

Сейчас мы напишем простенький сценарий, преобразующий десятичное число в шестнадцатеричное с помощью программы `dc` (листинг 15.5). Самим преобразованием будет заниматься программа `dc`, а наш сценарий только подготовит данные для этой программы.

Листинг 15.5. Сценарий dec_hex

```
#!/ bin/bash
B=16   # Основание системы счисления
# Проверяем, является ли параметр $1 числом
if [ -z "$1" ]
then
echo "Использование: dec_hex число"
exit 1
fi
# Передаем число ($1), систему счисления
# программе dc
echo ""$1" "$B" o p" | dc
```



Глава 16

Сценарии на *tcsh*

16.1. Использование *tcsh*

Как уже было отмечено в *главе 2*, изначально оболочка `tcsh` была разработана для операционной системы TENEX, которая использовалась в далеком прошлом на компьютерах DEC PDP-10. В Linux по умолчанию используется оболочка `bash`, а вот в FreeBSD по умолчанию используется именно `tcsh`. Поэтому данная глава будет особенно полезна для пользователей Linux, которые планируют перейти на FreeBSD, или же для пользователей FreeBSD, которые хотят больше узнать про свою любимую оболочку.

Вообще, выбор оболочки больше зависит от ваших предпочтений. В какой бы операционной системе вы ни работали, вы всегда сможете изменить оболочку по умолчанию: в FreeBSD вам никто не запретит использовать по умолчанию `bash`, равно как и в Linux использовать `tcsh`. В некоторых дистрибутивах, например в Ubuntu, `tcsh` по умолчанию не установлена. Для ее установки нужно ввести команду:

```
sudo apt-get install tcsh
```

Сейчас не хочется делать ни экскурс в историю, ни проводить сравнение с `csh`. Скажу только, что во всех свободных системах (Linux, FreeBSD) файл /bin/csh — это ссылка на /bin/tcsh.

Как и `bash`, `tcsh` позволяет создавать сценарии, а язык оболочки `tcsh` насчитывает 65 встроенных команд. Просмотреть список этих команд можно командой `builtins`. На рис. 16.1 приведен вывод этой команды.

Основные интерактивные возможности `tcsh` следующие:

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

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

Для автоматического дополнения слов, как и в `bash`, используется клавиша <Tab>, но правильнее использовать комбинации клавиш <Ctrl>+<I> и <Ctrl>+<D>.

Первая обеспечивает автодополнение слова, а вторая выводит список подходящих вариантов для автодополнения.

```
denis-desktop:~> builtins
:          @          alias      alloc      bg         bindkey    break
breaksw    builtins   case       cd         chdir      complete   continue
default    dirs       echo       echotc     else       end        endif
endsw      eval       exec       exit       fg         filetest   foreach
glob       goto       hashstat   history    hup        if         jobs
kill       limit      log        login      logout     ls-F       nice
nohup      notify     onintr     popd       printenv   pushd      rehash
repeat     sched      set        setenv     settc      setty      shift
source     stop       suspend    switch     telltc     termname   time
umask      unalias    uncomplete unhash     unlimit    unset      unsetenv
wait       where      which      while
denis-desktop:~>
```

Рис. 16.1. Список встроенных команд оболочки `tcsh`

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

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

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

```
!1
```

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

```
$ command &
```

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

```
$ fg 1
```

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

```
$ bg 1
```

16.2. Конфигурационные файлы *tcsh*

Основные глобальные файлы конфигурации `tcsh` — /etc/csh.cshrc, /etc/csh.login, /etc/csh.logout. Первый файл считывается при запуске каждого экземпляра `tcsh` в интерактивном режиме, второй — только если `tcsh` является оболочкой по умол-

чанию для запустившего ее пользователя. Третий файл считывается при выходе из tcsh при условии, что tcsh является оболочкой по умолчанию для этого пользователя (является так называемым login shell).

При регистрации пользователя в домашнем каталоге создаются файлы ~/.cshrc, ~/.login и ~/.logout, которые являются пользовательскими аналогами /etc/csh.cshrc и /etc/csh.login. Содержимое этих файлов копируется из каталога /etc/skel (или /usr/share/skel/ — в некоторых системах).

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

Также в пользовательском каталоге есть файл ~/.history, содержащий историю введенных команд.

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

Рассмотрим пример определения псевдонимов команд:

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

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

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

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

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

```
setenv  EDITOR  nano
```

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

16.3. Создание сценариев на *tcsh*

16.3.1. Переменные, массивы и выражения

Переменные, как уже было показано, устанавливаются командой set:

```
set name = denis
```

Здесь мы установили переменную name (значение — denis). Вывести значение переменной можно так:

```
echo $name
```

Если вы введете просто команду set (без параметров), то вы увидите список уже установленных переменных, в том числе и служебных (рис. 16.2).

```
denis-desktop:~> set
_       jobs

addsuffix
argv    ()
autoexpand
autolist
csubstnonl
cwd     /home/denix
dirstack        /home/denix
echo_style      both
edit
gid     1000
group   denix
history 100
home    /home/denix
killring        30
owd
path    (/usr/local/sbin /usr/local/bin /usr/sbin /usr/bin /sbin /bin /usr/games
)
prompt  %U%m%u:%B%~%b%#
prompt2 %R?
prompt3 CORRECT>%R (y|n|e|a)?
shell   /usr/bin/tcsh
```

Рис. 16.2. Команда `set`

Удалить переменную можно командой `unset`:

```
unset name
```

Тогда при обращении к переменной вы увидите сообщение:

```
name: Undefined variable.
```

Переменные окружения принято устанавливать командой `setenv` так:

```
setenv переменная значение
```

Обратите внимание, что между именем переменной и значением нет знака равенства. Пример:

```
setenv  EDITOR  nano
```

В `tcsh` также можно создавать массивы. Вот пример создания и использования массива:

```
$ set nums = (one two three four five)

$ echo $nums

one two three four five

$ echo $nums[3]

three

$ echo $nums[1-3]

   one two three
```

Как видите, мы можем вывести сразу весь массив, конкретный элемент и диапазон элементов, что очень удобно. Нумерация элементов массива начинается с 1.

Отдельного разговора заслуживают числовые переменные. Чтобы присвоить переменной число или результат арифметического выражения, нужно использовать символ `@`, при этом символ `$` перед именем переменной указывать не нужно.

Рассмотрим несколько примеров:

```
$ @ num = 0

$ echo $num

0

$ @ num = ( 2 + 2 ) * 2

$ echo $num

8

$ @ num += 5

$ echo $num

13

$ @ num++

$ echo $num

14

$ @ num2 = 5

$ @ num = $num2 + 5

$ echo $num

10

$ @ num = 1

echo $nums[$num]

one
```

Когда вы используете переменные в выражениях, тогда нужно указывать символ `$`, а после символа `@` указывать `$` не нужно. После `@` обязательно должен быть пробел.

Общий синтаксис при работе с числовыми переменными выглядит так:

```
@ переменная оператор выражение
```

Оператор — это один из операторов присваивания C: =, +=, – =, *=, /= или %=. Выражение — это арифметическое выражение, которое тоже строится по тем же правилам, что и в языке C. Ничего удивительного: `tcsh` предназначена для тех, кто знаком с языком C, и должна была облегчить создание сценариев C-программистам.

Теперь рассмотрим пример работы с массивами чисел:

```
$ set a = (0 0 0 0 0)

$ @ a[1] = 10

$ @ a[3] = ($ages[1] + 5)

$ echo $a[3]

15
```

16.3.2. Чтение ввода пользователя

Для чтения ввода пользователя используется конструкция `"$<"`, например:

```
echo -n "Введите строку: "

set line = "$<"
```

16.3.3. Переменные оболочки *tcsh*

При создании сценариев на `tcsh` вы можете использовать переменные оболочки, представленные в табл. 16.1.

Таблица 16.1. *Переменные оболочки* `tcsh`

Переменная	Описание
`argv`	Массив содержит аргументы командной строки (параметры, переданные сценарию). `argv[1]` — первый параметр, а `argv[0]` — это имя самого сценария. Обратиться к параметрам можно через конструкцию `argv[n]` или `$n`, например `argv[1]` или `$1`. Элемент массива `argv[*]` содержит все параметры вместе
`autolist`	Контролирует завершение команд и переменных. См. `man tcsh`
`autologout`	Включает возможность автоматического выхода. По умолчанию — 60 минут, если вы суперпользователь. Данная переменная не устанавливается (по умолчанию) для обычных пользователей
`cdpath`	Влияет на команду `cd`, задает путь поиска команд, обычно устанавливается в файле ~/.login, например: `set cdpath = (/home/denix /home/denix/bin)`
`cwd`	Содержит имя текущего каталога
`fignore`	Содержит массив суффиксов, которые `tcsh` будет игнорировать при завершении имен файлов
`gid`	Содержит идентификатор группы
`histfile`	Переменная содержит полное имя файла, в котором находится история команд. По умолчанию ~/.history
`history`	Размер списка истории
`home` или `HOME`	Имя домашнего каталога пользователя

***Таблица 16.1** (окончание)*

Переменная	Описание
mail	Содержит имя файла для проверки почты. TC Shell каждые 10 минут проверяет почту
owd	Предыдущий рабочий каталог
path или PATH	Путь поиска программ, обычно устанавливается в конфигурационных файлах tcsh: set path = (/usr/bin /bin /usr/local/bin /usr/bin/X11 ~/bin .)
prompt	Содержит формат приглашения командной строки, аналогична переменной PS1 в bash. Модификаторы формата описаны в табл. 16.2. Значение по умолчанию: set prompt = '! $ '
prompt2	Содержит формат приглашения командной строки для управляющих структур while и foreach
prompt3	Содержит формат приглашения командной строки при проверке правописания
savehist	Количество команд, которые будут сохранены в файл истории
shell	Полный путь к исполнимому файлу оболочки
status	Код завершения последней выполненной команды
tcsh	Версия tcsh
time	Может содержать только число или же буквы и цифры. Если переменная содержит только число, то это количество секунд процессорного времени. Если выполнение какой-то команды заняло больше времени, чем указано в time, то после выполнения команды будет выведено, сколько времени она занимала. Если time = 0, то после каждой команды будет выведено время выполнения. Если time содержит буквы и цифры (табл. 16.3), то это просто формат времени
user	Имя пользователя

***Таблица 16.2.** Формат командной строки*

Модификатор	Описание
%/	Текущий каталог
%~	То же, что и %/, но заменяет путь к домашнему каталогу пользователя тильдой
%! или %h или !	Текущий номер события
%m	Имя узла без домена

Таблица 16.2 *(окончание)*

Модификатор	Описание
%M	Полное имя узла, с доменом
%n	Имя пользователя
%t	Время дня
%p	Время дня (с секундами)
%d	День недели
%D	День месяца
%W	Месяц, формат мм
%y	Год, формат гг
%Y	Год, формат гггг
%#	Решетка (#) для суперпользователя или знак больше (>) для обычного пользователя
%?	Код завершения последней команды

Таблица 16.3. *Формат времени*

Модификатор	Описание
%U	Время, проведенное командой в пользовательском режиме (в процессорных секундах)
%S	Время, проведенное командой в режиме ядра (в процессорных секундах)
%W	Сколько раз процесс команды был выгружен на диск
%X	Средний размер сегмента кода программы, в килобайтах
%D	Средний объем памяти, используемый командой, в килобайтах
%K	Общий размер памяти, занятый командой (считается как %X+%D), в килобайтах
%M	Максимальный объем памяти, занятый командой, в килобайтах
%F	Количество ошибок страниц памяти
%I	Количество операций ввода
%O	Количество операций вывода

Перед тем как приступить к рассмотрению управляющих структур, рассмотрим применение скобок в tcsh. Предположим есть переменная:

```
$ set aa=abra
```

Потом нам нужно вывести ее в составе строки, для этого мы будем использовать фигурные скобки:

```
$ echo ${aa}cadabra
abracadabra
```

16.3.4. Управляющие структуры

Условный оператор *if*

Синтаксис оператора `if` очень прост:

```
if (выражение) команда
```

Команда будет выполнена, если выражение истинно. Выражения формируются так же, как в языке C. В листинге 16.1 представлен небольшой сценарий, проверяющий количество аргументов, переданных ему.

Листинг 16.1. Первый сценарий на `tcsh`

```
#!/bin/tcsh

if ( $#argv == 0 ) echo "Аргументы не заданы"
```

Также можно использовать следующее выражение:

```
-n имя_файла
```

В данном случае возможные варианты n представлены в табл. 16.4.

Таблица 16.4. *Значения n*

n	Описание
b	Файл является блочным устройством (обмен данными с устройством осуществляется блоками данных)
c	Файл является символьным устройством (обмен данными с устройством осуществляется посимвольно)
d	Файл является каталогом
e	Файл существует
g	Для файла установлен бит SGID *(см. главу 4)*
k	Для файла установлен "липкий" бит
l	Файл является символической ссылкой
o	Файл принадлежит текущему пользователю
p	Файл является именованным потоком (FIFO)
r	У пользователя есть право чтения файла
s	Файл не пустой (ненулевой размер)
S	Файл является сокетом
t	Дескриптор файла открыт и подключен к экрану
u	Для файла установлен SUID *(см. главу 4)*
w	У пользователя есть право записи файла

Таблица 16.4 *(окончание)*

n	Описание
x	У пользователя есть право выполнения файла
X	Файл является встроенной командой или его исполнимый файл найден при поиске в каталогах, указанных в $path
z	Файл пуст (нулевой размер)

Рассмотрим пример использования данного условия:

```
if -e $1 echo "Файл существует"
```

Условный оператор *if..then..else*

Условный оператор `if..then..else` похож на `if`, только добавляется блок `else` (иначе), который выполняет команды в случае ложности условия. Сокращенная версия оператора выглядит так:

```
if (выражение) then

    команды, которые будут выполнены в случае истинности выражения

endif
```

Полная версия оператора выглядит так:

```
if (выражение) then

    команды, которые будут выполнены в случае истинности выражения

else

    команды, которые будут выполнены в противном случае (когда выраже-
ние = false)

endif
```

Существует еще одна форма этого оператора, точнее, это отдельный оператор `if..then..elif`:

```
if (выражение1) then

     команды (если выражение1 = true)

else if (выражение2) then

    команды (если выражение2 = true)

. . .

else

     команды (если ни одно из выражений не равно true)

endif
```

Рассмотрим небольшой пример использования условного оператора (листинг 16.2).

Листинг 16.2. Пример использования условного оператора

```
#!/bin/tcsh

# Получаем число из командной строки
set num = $argv[1]

set flag
#

if ($num < 0) then

    @ flag = 1

else if (0 <= $num && $num < 50) then

    @ flag = 2

else if (50 <= $num && $num < 1000) then

    @ flag = 3

else

    @ flag = 4

endif

#

echo "Flag: ${flag}."
```

Оператор *foreach*

Оператор `foreach` удобно использовать для перебора массивов, его синтаксис:

```
foreach индекс-цикла (список-аргументов)
     команды
end
```

Также цикл `foreach` удобно использовать при работе с файлами, например:

```
foreach f ( *.txt )
   echo $f
end
```

Здесь мы в цикле выводим имена всех текстовых файлов в текущем каталоге. Конечно, для этой цели проще вызвать команду `ls`, но здесь мы продемонстрировали использование `foreach`.

Оператор *while*

Оператор `while` — это еще один вариант цикла. Команды, находящиеся в теле цикла, выполняются до тех пор, пока выражение истинно:

```
while (выражение)

    команды

end
```

Рассмотрим пример сценария, вычисляющего факториал числа n, при этом n задается в командной строке (листинг 16.3).

Листинг 16.3. Пример использования `while`

```
#!/bin/tcsh

set n = $argv[1]

set i = 1

set fact = 1

#

while ($i <= $n)

    @ fact *= $i

    @ i++

end

#

echo "Факториал числа $n равен $fact"
```

В циклах вы можете использовать операторы `break` и `continue`. Оператор `break` прерывает цикл и передает управление оператору, следующему за `end`. Оператор `continue` прерывает только текущую итерацию цикла и передает управление оператору `end`, который после получения управления начнет следующую итерацию цикла.

Оператор *switch*

В ряде случаев использовать оператор `switch` намного удобнее, чем серию условных операторов. Синтаксис `switch` следующий:

```
switch (строка)

    case образец1:

        команды1

    breaksw


    case образец2:

        команды2

    breaksw

    ...

    default:

        команды по умолчанию

    breaksw

endsw
```

Вот пример использования данного оператора:

```
set string = test

switch (string)
    case test:
          echo "Строка: test"
    breaksw
    case text:
          echo "Строка: text"
    breaksw
    default:
          echo "Строка не опознана"
    breaksw
endsw
```

16.3.5. Встроенные команды *tcsh*

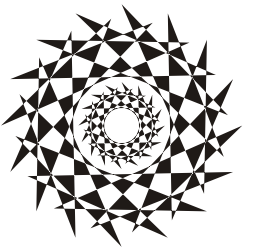
Оболочка `tcsh` обладает встроенными командами (как, впрочем, и любая другая оболочка). В табл. 16.5 приведены самые полезные встроенные команды `tcsh`.

Таблица 16.5. *Самые полезные встроенные команды* `tcsh`

Команда	Описание
`@`	Вычисляет арифметическое выражение. Данная команда была подробно описана ранее при написании сценариев
`alias`	Создает псевдоним для команд оболочки, эту команду мы тоже рассмотрели ранее
`alloc`	Отображает отчет о количестве свободной и использованной памяти
`bg`	Перемещает приостановленную задачу в фоновый режим
`builtins`	Отображает список встроенных команд
`cd или chdir`	Изменяет текущий каталог
`dirs`	Отображает стек каталогов
`echo`	Отображает свои аргументы. Обычно используется при написании сценариев
`exec`	Запускает другую программу в этой же оболочке
`exit`	Осуществляет выход из `tcsh`
`fg`	Перемещает задачу на "передний план", т. е. делает ее активной
`filetest`	В основном используется при написании сценариев. Позволяет проверить тип файла (устройство, каталог, файл), существование файла и осуществить ряд других полезных проверок (см. табл. 16.4)
`glob`	Похожа на `echo`, но не отображает пробелы между аргументами и не выводит символ новой строки после своего вывода
`hashstat`	Выводит статистику механизма хэширования `tcsh`
`history`	Отображает введенные ранее команды
`jobs`	Отображает список задач (приостановленные команды и те, которые выполняются в фоновом режиме)
`kill`	Завершает задачу или процесс
`limit`	Позволяет ограничить ресурсы текущего процесса и всех процессов, которые он будет создавать
`login`	Используется для входа пользователя. Сопровождается именем пользователя
`logout`	Завершает сессию, если вы используете `tcsh` в качестве оболочки по умолчанию
`ls-F`	Похожа на `ls -F`, но выполняется быстрее
`nice`	Позволяет изменить приоритет процесса *(см. разд. 5.3)*
`nohup`	Позволяет вам выйти из системы без завершения процессов, выполняемых в фоновом режиме
`notify`	Уведомляет вас при изменении статуса одной из ваших задач

Таблица 16.5 *(окончание)*

Команда	Описание
popd	Удаляет каталог из стека каталогов
printenv	Отображает список всех переменных окружения
pushd	Изменяет рабочий каталог и помещает новый каталог на вершину стека каталогов
rehash	Пересоздает внутренние таблицы, используемые механизмом хэширования. Допустим, вы создали свой сценарий и поместили его в /usr/bin. Оболочка tcsh не увидит этот файл до тех пор, пока вы не обновите внутренние таблицы
repeat	Команде нужно передать два аргумента — количество повторений и простую команду (без потоков и списка команд), и повторяет эту команду указанное количество раз
sched	Выполняет команду в указанное время, например: `$ sched 10:00 echo "Уже 10 часов!"`
set	Объявляет и инициализирует локальную переменную
setenv	Объявляет и инициализирует переменную окружения
source	Запускает сценарий оболочки, указанный в качестве аргумента. Данная команда не порождает новый процесс, а просто обрабатывает сценарий. Данную команду можно использовать как include в обычных языках программирования
stop	Останавливает задачу или процесс (которая или который выполняется в фоновом режиме)
suspend	Приостанавливает текущую оболочку и помещает ее в фоновый режим
time	Запускает команду, указанную в качестве параметра. После выполнения команды отображает информацию о ее выполнении (время выполнения)
umask	Изменяет маску прав доступа по умолчанию (см. man umask)
unalias	Удаляет псевдоним команды
unhash	Выключает механизм хэширования. См. также hashstat и rehash
unlimit	Удаляет лимиты текущего процесса
unset	Удаляет объявление переменной
unsetenv	Удаляет объявление переменной окружения
where	В качестве аргумента нужно передать команду, после чего получите информацию о том, является ли команда встроенной или исполнимым файлом и где находится этот файл. Похожа на команду which
which	Подобна стандартной утилите which, но работает быстрее и владеет информацией обо всех командах и псевдонимах. Выводит местонахождение исполнимого файла, если он вообще существует. Поиск производится в каталогах, указанных в переменной окружения path



Глава 17

Язык gawk

17.1. Введение в gawk

gawk (GNU awk) — небольшой язык программирования, ориентированный на обработку текстовых файлов. Язык awk (далее будем называть этот язык просто awk) удобно использовать для обработки дампов баз данных, системных файлов. Он подобен языку Perl, но проще в использовании. По сути, awk является прародителем Perl. Как и Perl, awk построен на регулярных выражениях и средствах для обработки шаблонов.

Язык awk назван по первым буквам фамилий его создателей: Alfred V. Aho, Peter J. Weinberger и Brian W. Kernighan.

Основное назначение языка, как уже было сказано, — обработка текстовых файлов, поэтому awk пригодится администраторам для анализа системных журналов и создания различных систем статистики.

17.2. Основы языка

17.2.1. Образцы и действия

Сценарий awk пишется, как и сценарий для оболочек `bash` и `tcsh`: это обычный текстовый файл, в начале которого задается интерпретатор:

```
#!/usr/bin/awk -f
```

Сценарий на языке awk состоит из одной или нескольких строк вида:

```
образец { действие }
```

Образец — это строка (или регулярное выражение), которая должна быть в обрабатываемом сценарием текстовом файле. Действие будет применено к строке, соответствующей образцу. Если образец — это регулярное выражение, то он заключается в слэши:

```
/выражение/
```

Если образец не указан, то действие будет применено ко всем строкам обрабатываемого файла.

Особого внимания заслуживают образцы BEGIN и END. Первый образец запускает команды до обработки файла, а второй — после обработки файла. Например:

```
BEGIN {
    print "Мой сценарий\n"
    counter=0
}
{
    print "Основная программа\n"
}
END {
    print "Сценарий завершен"
    printf ("\nСчетчик:\t%d\n", counter)
}
```

17.2.2. Операторы

Основные операторы языка awk представлены в табл. 17.1.

Таблица 17.1. *Основные операторы языка awk*

Оператор	Описание
<	Меньше
>	Больше
<=	Меньше или равно
>=	Больше или равно
==	Равно
!=	Не равно
!~	Отрицание. Используется при задании образцов. Означает, что поле или переменная не соответствует регулярному выражению
~	Поле или переменная соответствует регулярному выражению
||	Логическое ИЛИ (OR)
&&	Логическое И (AND)
*	Умножение
/	Деление
%	Получение остатка от деления
+	Сложение
-	Вычитание
=	Присваивает переменной значение или результат выражения
++	Увеличивает значение переменной
--	Уменьшает значение переменной

17.2.3. Переменные

Строковые переменные инициализируются пустой строкой, а числовые переменные — нулем:

```
counter=0
name=""
```

Служебные переменные языка awk приведены в табл. 17.2.

Таблица 17.2. *Служебные переменные языка awk*

Переменная	Описание
`$0`	Текущая запись (в одной переменной)
`$n, n` — число	Поле с номером *n* в текущей записи
`FILENAME`	Имя текущего файла, null для стандартного ввода
`FS`	Разделитель полей
`NF`	Количество полей в записи
`NR`	Номер записи
`OFS`	Разделитель полей при выводе (по умолчанию пробел)
`ORS`	Разделитель записей при выводе (по умолчанию символ новой строки)
`RS`	Разделитель записей при вводе (по умолчанию символ новой строки)

17.2.4. Ассоциативные массивы

В отличие от других языков программирования, awk поддерживает ассоциативные массивы (правда, такие массивы поддерживает еще PHP и некоторые другие языки, но во многих языках таких массивов нет). В ассоциативном массиве в качестве индекса выступает не число, а строка.

Пример объявления такого массива:

```
массив[строка] = значение
```

Затем для перебора массива используется вот такая конструкция:

```
for (элемент in массив) действие
```

Например:

```
for (item in count) printf "%-20s%-20s\n"
```

17.2.5. Функции

В табл. 17.3 приведены функции, которые язык awk предоставляет для обработки чисел и строк.

Таблица 17.3. *Функции языка awk*

Функция	Описание
`length(str)`	Возвращает длину строки (количество символов в `str`). Если строка не указана, возвращает количество символов в текущей записи
`int(num)`	Возвращает целую часть числа `num`
`index(str1,str2)`	Возвращает индекс строки `str2` в строке `str1` или 0, если строки `str2` нет в строке `str1`
`split(str,arr,del)`	Разбивает строку `str` на элементы массива `arr`, параметр `del` — это разделитель, который используется для разделения строки
`sprintf(fmt,args)`	Форматирует аргументы (параметр `args`) в соответствии со строкой формата `fmt`
`substr(str,pos,len)`	Возвращает подстроку строки `str` длины `len`, которая начинается на позиции `pos`
`tolower(str)`	Возвращает копию строки `str`, где все символы будут в нижнем регистре
`toupper(str)`	Возвращает копию строки `str`, где все символы будут в верхнем регистре

17.2.6. Вывод с помощью *printf*

Команда `printf` используется для вывода аргументов в соответствии с заданным форматом:

```
printf "строка формата", аргумент1, …, аргументN
```

Строка формата определяет, как будут отображены аргументы. В строке формата вы можете использовать `\t` для вставки символа табуляции и `\n` для новой строки на экране.

Для каждого аргумента в строке формата должен быть свой модификатор вывода. Модификатор имеет формат:

```
%[-][x[.y]]conv
```

Символ `%` обязателен, и он означает, что перед нами — модификатор. Символ "-" означает, что вывод должен быть отформатирован по левому краю. `x` — минимальная ширина поля, а `y` — количество цифр после запятой (при выводе числа с дробной частью), а `conv` задает формат аргумента (табл. 17.4).

Таблица 17.4. *Формат аргумента*

conv	Описание
`d`	Десятичное число
`e`	Экспоненциальная запись числа

Таблица 17.4 *(окончание)*

conv	Описание
f	Число с плавающей запятой
o	Беззначное восьмеричное число
s	Строка
x	Беззначное шестнадцатеричное число

17.2.7. Управляющие структуры

Условный оператор *if..else*

Условный оператор работает так же, как и в любом другом языке программирования:

```
if (условие)
          {команды1}
     [else
          {команды2}]
```

Если условие истинно (значение выражения — true), то будут выполнены `команды1`, в противном случае будут выполнены `команды2`. Вот пример условного оператора:

```
if (num == "one")
               print "Число: 1"
            else
               print "Другое число:", num
       }
```

Цикл *while*

Цикл `while` выполняется до тех пор, пока условие истинно. Синтаксис цикла следующий:

```
while (условие)
    {команды}
```

Рассмотрим небольшой пример:

```
count = 1

    while (count <= 10)

        {

        print n

        n++

        }
```

Цикл *for*

Синтаксис цикла `for` следующий:

```
for (инициализация; условие; инкремент)

     {команды}
```

Цикл `for` — это цикл со счетчиком. Он выполняется, пока истинно условие. Перед запуском цикла выполняются команды инициализации, как правило, сбрасывается счетчик (или инициализируется). Далее проверяется условие: если оно истинно, выполняются команды. После выполнения команд происходит инкремент счетчика.

Пример цикла `for`:

```
for (n=1; n <= 10; n++)

        print n
```

Для ассоциативных массивов синтаксис цикла `for` немного другой:

```
for (элемент in массив)

     {команды}
```

В теле цикла вы можете использовать команды `break` (прерывает цикл) и `continue` (прерывает текущую итерацию).

17.3. Примеры

Теперь рассмотрим несколько примеров. Предположим, у нас есть файл friends с данными о наших друзьях в таком формате:

```
ник год_рождения e-mail        icq
```

Количество полей, сами понимаете, может быть произвольным. Для большей определенности файл friends представлен в листинге 17.1.

Листинг 17.1. Файл friends

```
den    1983   den@localhost 111111111
evg    1982   evg@localhost 111111112
max    1987   max@localhost 111111114
fox    1980   fox@localhost 111111113
dm     1979   dm@locahost   111111119
vvv    1980   vvv@localhost 111111117
ppt    1982   ppt@localhost 111111115
```

Самая простая программа на языке awk выглядит так:

```
{print}
```

Данная программа просто выведет весь этот файл. Для ее запуска введите команду:

```
gawk '{print}' friends
```

Попробуем усложнить программу и вывести всех, кто родился в 1980 году. Для этого нужно использовать регулярные выражения. Регулярные выражения заключаются в слэши:

```
gawk '/1980/' friends
```

Вывод будет таким:

```
fox 1980   fox@localhost 111111113
vvv 1980   vvv@localhost 111111117
```

Спрашивается, почему мы не оформляем наши программы в сценарии, а запускаем их из командной строки? Да потому что так проще — ради программы, состоящей из одной строки, не хочется создавать файл, устанавливать его права, передавать ему параметры. А так все видно: каким интерпретатором обрабатывается программа (gawk), текст программы и обрабатываемый файл (friends).

Теперь попробуем вывести только определенные поля. Порядок вывода полей значения не имеет:

```
gawk '{print $2, $1}' friends
```

Мы вывели второе и первое поле:

```
1983 den
1982 evg
1987 max
...
```

Можно усложнить программу и вывести только несколько полей, но которые соответствуют регулярному выражению, например:

```
gawk '/1980/ {print $2, $1}' friends
```

Вывод программы:

```
1980 fox
1980 vvv
```

Выведем все записи, где в первом поле есть буква d:

```
gawk '$1 ~ /d/' friends
```

Вывод программы:

```
den 1983   den@localhost 111111111
dm  1979   dm@locahost   111111119
```

До этого мы выводили записи, которые соответствуют определенному регулярному выражению. Давайте выведем записи, где поля соответствуют тому или иному регулярному выражению. Например, вот программа, выводящая записи, где второе поле равно 1982:

```
gawk '$2 == 1982' friends
```

Вывод программы:

```
evg 1982   evg@localhost 111111112
ppt 1982   ppt@localhost 111111115
```

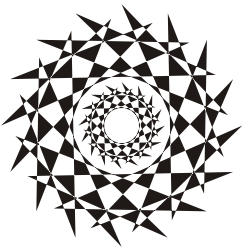
Теперь напишем awk-сценарий — более сложную программу, которая не помещается в одну строку. Программа подсчитывает, сколько человек родилось в 1980 году. Данную программу можно было бы записать в одну строку, но мы используем образцы `BEGIN` и `END` для улучшения вывода программы.

Листинг 17.2. Сценарий на awk

```
#!/usr/bin/awk -f
BEGIN {
                print "Список друзей"
                year=0
}
{
                if ($2 == "1980") year=year+1
}
 END {
                printf ("\tВсего записей:\t%d\n", NR)
                printf ("\tВ 1980 году родилось друзей:\t%d\n", year)
}
```

Чтобы запустить сценарий, нужно сделать его исполнимым и передать ему файл друзей:

```
$ chmod +x friends.awk
$ ./friends.awk friends
```



Глава 18

Собственный сервер для PHP-программиста

18.1. Зачем нужен сервер PHP-программисту?

В этой главе мы рассмотрим настройку сервера для PHP-программиста. Зачем он нужен? А нужен он для "себя любимого". Попытаться настроить, попробовать, как работает. Можно использовать его и как экспериментальную площадку. При этом можно не платить за хостинг — ведь для экспериментов вполне хватит и локального сервера.

Настройку сервера будем рассматривать на примере Ubuntu. Некоторым Linux-пользователем Ubuntu не нравится: уж больно она проста. Но простота не означает примитивность, что мы и докажем в этой главе, превратив обычную рабочую станцию в сервер сети — настроим Web-сервер с поддержкой PHP, FTP-сервер и сервер баз данных MySQL. В общем, все, что нужно PHP-программисту для разработки Web-приложений.

Если Ubuntu — это не ваш дистрибутив, то ничего страшного. Порядок действий в вашем дистрибутиве будет такой же, а именно: установка пакетов, редактирование конфигурационных файлов и запуск служб. Подробно об установке пакетов мы поговорим в следующей части: если вы еще не умеете устанавливать пакеты в вашем дистрибутиве, тогда обязательно прочитайте главу об установке пакетов в вашем дистрибутиве.

Примечание

Если же экспериментировать вам не хочется, равно как и изучать PHP, то лучше вообще не читайте эту главу, особенно если вы постоянно подключены к Интернету — зачем вам целый набор лишних служб, за которыми вы не будете присматривать?

18.2. Web-сервер

18.2.1. Установка Apache и PHP

Понимаю, что процесс установки пакетов мы еще не рассматривали, поэтому просто введите следующую команду:

```
sudo apt-get install apache2 php5
```

Эта команда установит Web-сервер Apache2 и интерпретатор PHP версии 5. После этого нужно установить следующие пакеты:

- ph5-cli — интерпретатор PHP, работающий в режиме командной строки (command-line interpreter);
- php5-imap — поддержка протоколов POP/IMAP для PHP;
- php5-gd — поддержка графических функций PHP;
- php5-mysql — поддержка функций для работы с базой данных MySQL.

Для этого введите команду:

```
sudo apt-get install php5-cli php5-imap php5-gd php5-mysql
```

18.2.2. Тестирование настроек Web-сервера

Протестируем Web-сервер. Откройте браузер и введите адрес:

```
http://localhost
```

Должна открыться страница, изображенная на рис. 18.1.

Теперь протестируем поддержку PHP. Поместите файл test.php в каталог /var/www/. Это можно сделать с помощью команды:

```
sudo nano /var/www/test.php
```

```
<?

phpinfo();

?>
```

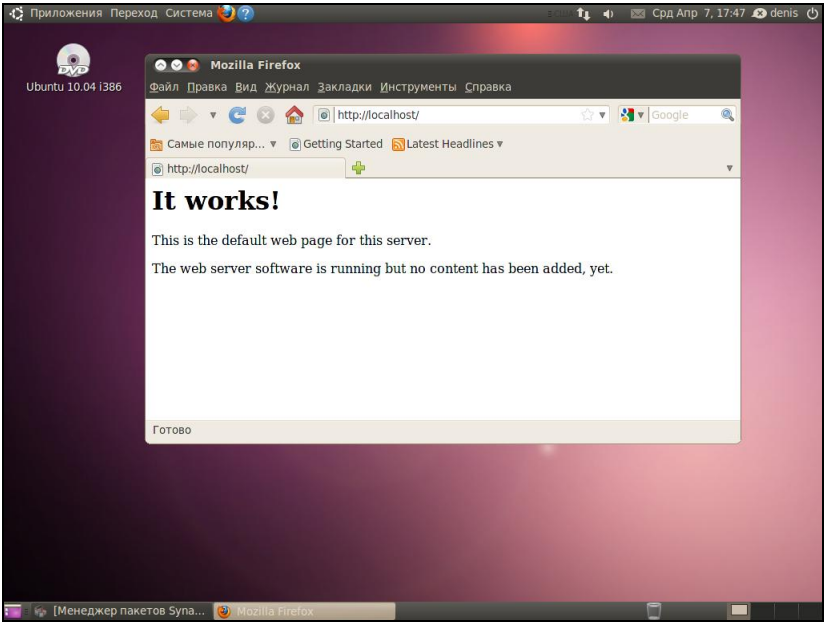


Рис. 18.1. Тестовая страница Apache

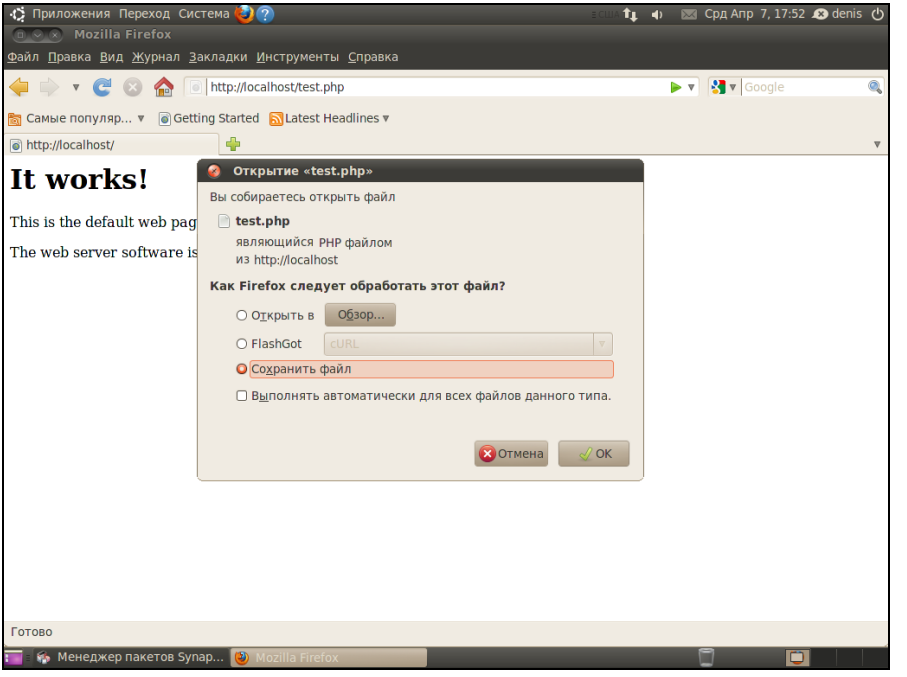


Рис. 18.2. Поддержки PHP нет!

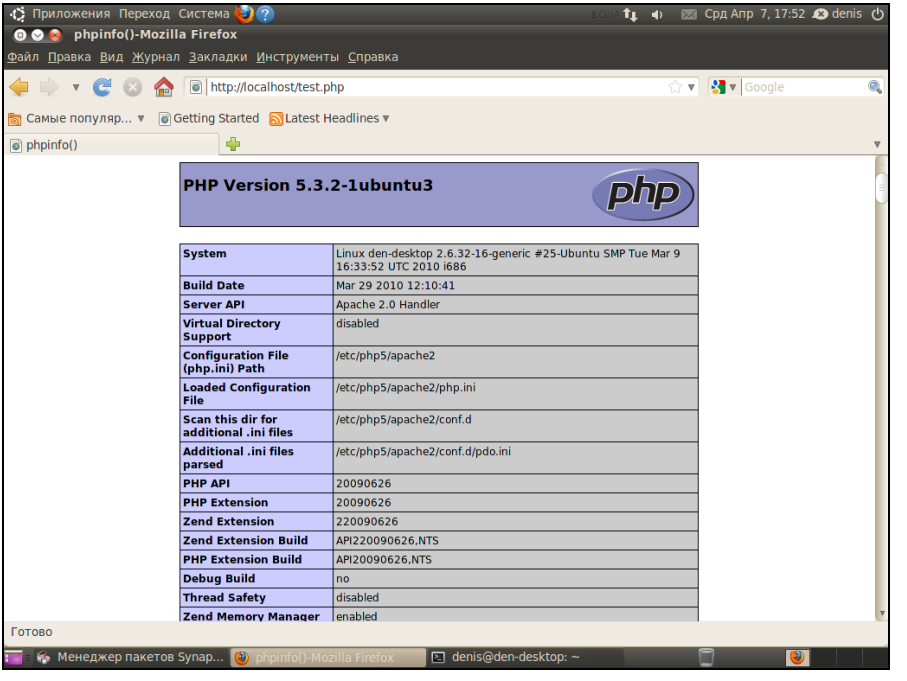


Рис. 18.3. Тестовый сценарий

Чтобы создать файл в этом каталоге, нужны права root. После создания файла введите в строке браузера адрес **http://localhost/test.php**.

Скорее всего (если вы после установки Apache не перезагружали компьютер), увидите картинку из серии "не ждали" (рис. 18.2).

Похоже, что поддержка PHP не установлена. Но на самом деле поддержка PHP уже установлена, просто нужно перезагрузить Apache — введите в терминале команду:

```
sudo service apache2 restart
```

Да, теперь и в Ubuntu есть команда `service`! После этого в окне браузера вы должны увидеть информацию о своем сервере и PHP (рис. 18.3).

Как вы уже догадались, каталог /var/www является корневым для вашего сервера. Если создать в нем файл test.html, то он будет доступен по адресу **http://localhost/test.html**.

18.2.3. Конфигурационные файлы сервера. Команды запуска и останова сервера

Файлы конфигурации сервера находятся в каталоге /etc/apache2. Основной файл конфигурации называется apache2.conf. По умолчанию его настройки устроят большинство пользователей. Если вы планируете использовать Web-сервер не только локально (для экспериментов с PHP), но и как Web-сервер своей домашней сети, откройте файл apache2.conf и найдите директиву:

```
#ServerName new.host.name
```

Нужно ее раскомментировать и указать имя сервера, которое будут вводить пользователи в строке браузера. Данное имя должно быть зарегистрировано в DNS-сервере вашей сети (или указано в файле /etc/hosts каждого компьютера сети). Обычно здесь указывается имя компьютера, например:

```
ServerName user-desktop
```

После этого можно будет обращаться к серверу по адресу **http://user-desktop/**.

Для остановки и перезапуска Web-сервера используются соответственно следующие команды (перезапуск необходим после изменения конфигурационных файлов сервера):

```
sudo service apache2 stop
sudo service apache2 restart
```

18.3. Сервер баз данных MySQL

18.3.1. Установка сервера

Для установки введите следующую команду:

```
sudo apt-get install mysql-server-5.1 mysql-client-5.1 mysql-admin
```

Первый пакет содержит последнюю версию MySQL-сервера (на данный момент эта пятая версия), во втором пакете находится MySQL-клиент, т. е. программа, ко-

торая будет подключаться к MySQL-серверу, передавать ему SQL-запросы и отображать результат их выполнения. Третий пакет содержит программу для администрирования MySQL-сервера. Все необходимые дополнительные пакеты будут установлены автоматически.

При установке MySQL-сервера будет запрошен пароль пользователя root. Это пароль пользователя MySQL — не нужно путать с системным пользователем root!

18.3.2. Команды управления пользователями MySQL-сервера

Для изменения пароля root (который вы ввели при установке сервера MySQL) используется следующая команда:

```
sudo mysqladmin -u root password ваш_пароль
```

Этот пароль вы будете использовать для администрирования сервера (пароль может и должен отличаться от того, который вы используете для входа в систему). Для обычной работы с сервером рекомендуется создать обычного пользователя. Для этого введите команду:

```
mysql -u root -p mysql
```

Программа `mysql` является клиентом MySQL-сервера. Она должна подключиться к базе данных `mysql` (служебная база данных), используя имя пользователя root (`-u root`). Поскольку вы только что указали пароль для пользователя root (до этого пароль для root не был задан), вам нужно указать параметр `-p`. После того как программа `mysql` подключится к серверу, вы увидите приглашение программы. В ответ на него нужно ввести следующий SQL-оператор:

```
insert into user(Host, User, Password, Select_priv, Insert_priv,
Update_priv, Delete_priv)
values ('%','username',password('123456'),'Y','Y','Y','Y');
```

Только что мы создали пользователя с именем `username` и паролем `123456`. Этот пользователь имеет право использовать SQL-операторы `select` (выборка из таблицы), `insert` (добавление новой записи в таблицу), `update` (обновление записи), `delete` (удаление записи). Если вам нужно, чтобы ваш пользователь имел право создавать и удалять таблицы, добавьте привилегии `Create_priv` и `Drop_priv`:

```
insert into user(Host, User, Password, Select_priv, Insert_priv,
Update_priv, Delete_priv, Create_priv, Drop_prov)
values ('%','username',password('123456'),'Y','Y','Y','Y','Y','Y');
```

> ***Внимание!***
>
> Этот SQL-оператор можно записать в одну строку, можно разбить на несколько строк — как вам будет удобно. Но в конце каждого SQL-оператора должна быть точка с запятой! Помните об этом.

Для выхода из программы `mysql` нужно ввести команду `quit`.

Кроме программы `mysql` в состав MySQL-клиента входит одна очень полезная программа — `mysqlshow`, которая может вывести список таблиц, которые находятся

в той или иной базе данных. Кроме этого, она еще много чего может, но в данный момент нам нужен пока список таблиц — чтобы вы знали, какие таблицы есть в той или иной базе данных:

```
mysqlshow -p <база данных>
```

18.3.3. Команды запуска и останова сервера

Для управления сервером используется программа /etc/init.d/mysql. Раньше нужно было вызывать /etc/init.d/mysql непосредственно, сейчас можно вызвать управляющую программу через команду `service`, что намного удобнее. Для запуска сервера следует передать этой программе параметр `start`, для останова — `stop`, а для перезапуска — `restart`:

```
sudo service mysql start
sudo service mysql stop
sudo service mysql restart
```

Также для управления сервером можно использовать программу `mysqladmin`, узнать больше о ней можно с помощью команды:

```
man mysqladmin
```

18.3.4. Программа MySQL Administrator

Устанавливая сервер, мы установили графическую программу MySQL Administrator (пакет mysql-admin). Запустите программу командой меню **Приложения | Программирование | MySQL Administrator**. Укажите адрес сервера `localhost`, имя пользователя — `root`, пароль, который вы указали при установке сервера (рис. 18.4), и нажмите кнопку **Connect**. Далее управлять сервером будет существенно проще (рис. 18.5).

Пройдемся по основным разделам программы MySQL Administrator:

- **Server Information** — общая информация о сервере (см. рис. 18.5);
- **Service Control** — управление запуском сервиса MySQL (здесь вы можете перезапустить сервер);
- **Startup Parameters** — параметры, указываемые при запуске сервера;
- **User Administration** — здесь можно добавить новых пользователей MySQL и установить права пользователей;
- **Server Connections** — позволяет просмотреть текущие соединения с сервером;
- **Server Logs** — журналы сервера;
- **Backup** — создать резервную копию сервера;
- **Restore Backup** — восстановление из резервной копии;
- **Replication Status** — состояние репликации сервера;
- **Catalogs** — позволяет просмотреть имеющиеся базы данных и таблицы внутри них.

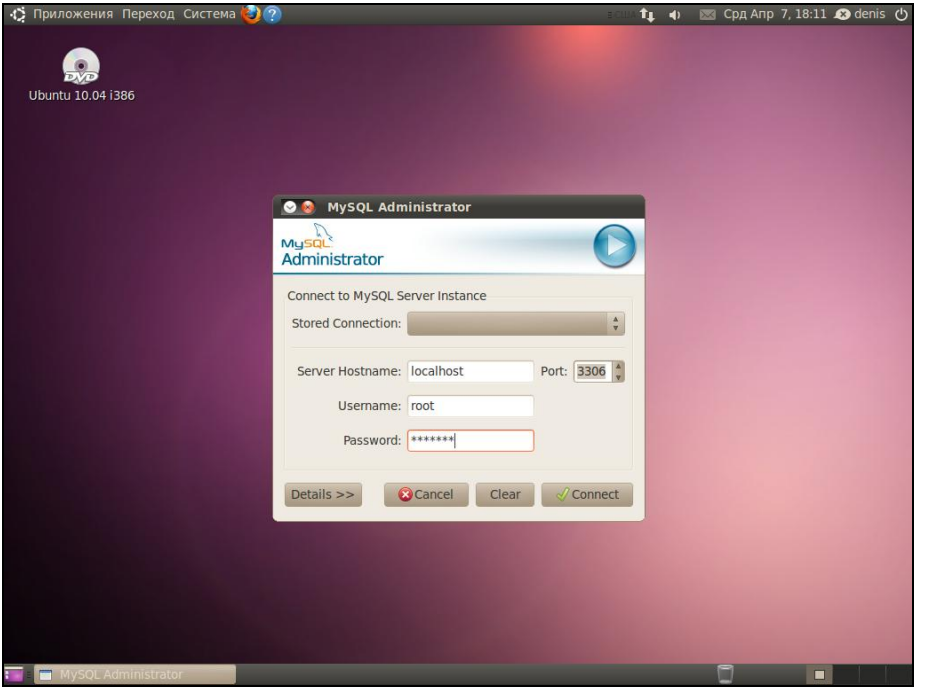


Рис. 18.4. Вход на сервер

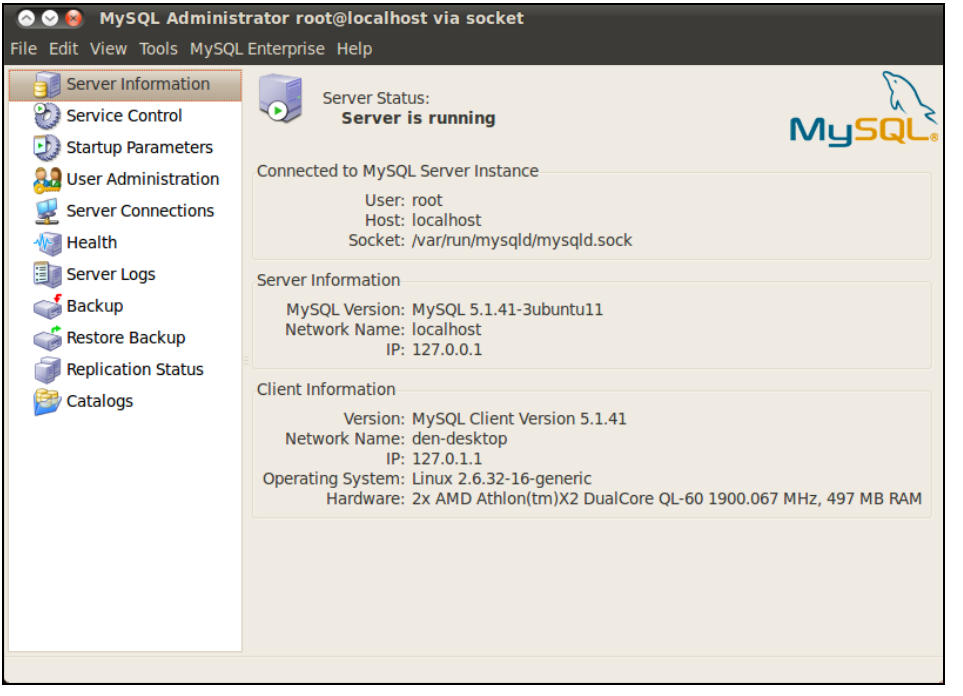


Рис. 18.5. Основное окно программы MySQL Administrator

18.4. Быстрая настройка FTP-сервера

Рассмотрим настройку FTP-сервера ProFTPD. На мой взгляд, этот сервер наиболее гибкий в настройке. Установите два пакета: proftpd-basic и gadmin-proftpd. Первый содержит FTP-сервер, а второй — графическую программу для его настройки.

После установки сервер практически готов к работе. Его нужно только запустить (по умолчанию он добавляется в автозапуск, поэтому после перезагрузки эта команда не потребуется):

```
sudo /etc/init.d/proftpd start
```

Попробуем подключиться к вашему серверу:

```
ftp <имя_вашего_компьютера>
```

FTP-сервер запросит имя пользователя и пароль. Нужно вводить имя пользователя, зарегистрированное на вашем компьютере (на FTP-сервере). После этого вы получите доступ к домашнему каталогу пользователя, который будет для вас корневым, т. е. за пределы этого каталога выйти не получится. Это сделано из соображений безопасности, чтобы никто не мог получить доступ к файловой системе сервера.

Такая конфигурация сервера не очень удобна. Когда нужно предоставить доступ пользователям к их домашним каталогам — все просто прекрасно. Но предположим, вы хотите сделать свою коллекцию фильмов доступной всем пользователям сети. Можно, конечно, записать их в ваш домашний каталог, а потом всем предоставить имя пользователя и пароль, но это нежелательно с точки зрения безопасности — все смогут прочитать ваши файлы, причем не только прочитать, а изменить и даже удалить. Поэтому намного рациональнее предоставить всем пользователям анонимный доступ к вашей коллекции фильмов. Каждый желающий сможет скачать фильм, но никто не сможет ничего удалить.

Основной файл конфигурации сервера ProFTPD называется /etc/proftpd/proftpd.conf. По умолчанию в конфигурационном файле указываются далеко не все опции, которые доступны, поэтому для создания полноценного конфигурационного файла нужно запустить программу `gadmin-proftpd`:

```
gksudo gadmin-proftpd
```

При первом запуске программа сообщит вам, что в конфигурационном файле не хватает многих опций, и предложит создать полноценный конфигурационный файл — согласитесь, нажав кнопку **Yes**.

В верхней части окна программы настройки FTP-сервера расположены кнопки управления сервером (рис. 18.6):

- **Activate/Deactivate** — информируют о статусе сервера (зеленая кнопка говорит о том, что сервер запущен);
- **Shutdown** — завершает работу сервера.

На вкладке **Servers** можно управлять разными серверами, если у вас их несколько — точнее, редактировать их файлы конфигурации. У нас будет только один сервер, поэтому я не думаю, что кнопки **Add** (Добавить сервер) и **Delete** (Удалить сервер) вам пригодятся.

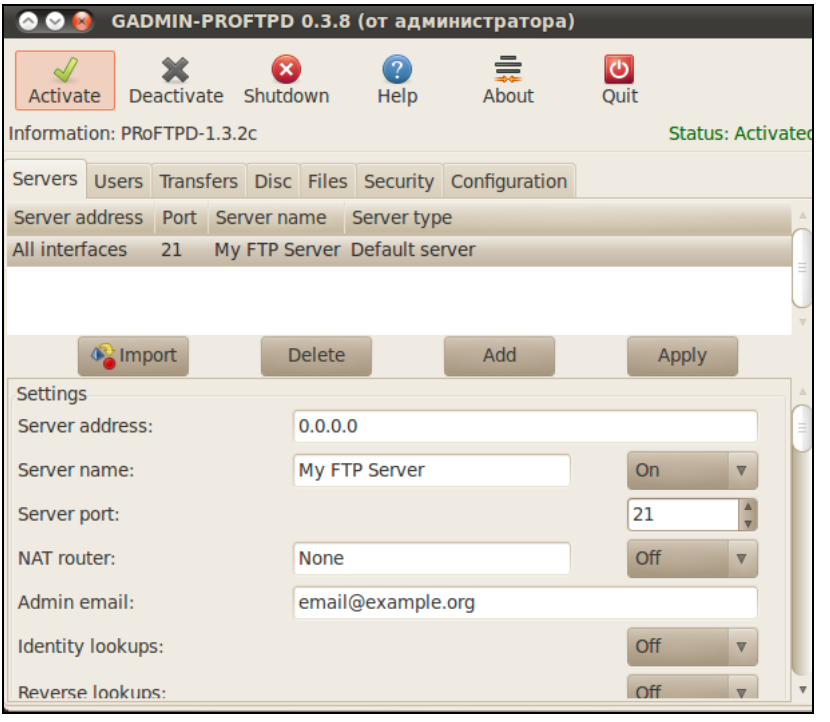


Рис. 18.6. Основное окно программы

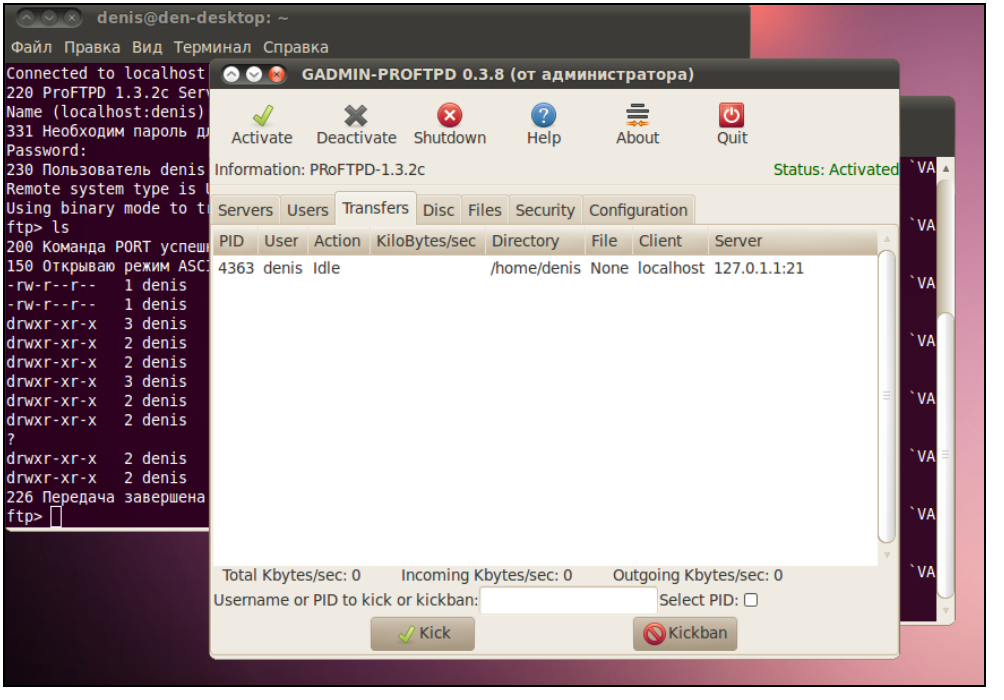


Рис. 18.7. Информация об использовании сервера

Ознакомимся с другими вкладками программы настройки:

- **Users** — позволяет управлять пользователями выбранного сервера;
- **Transfers** — информация об использовании выбранного сервера (рис. 18.7);
- **Disk** — информация об использовании дискового пространства локального компьютера;
- **Files** — позволяет сгенерировать файлы статистики сервера;
- **Security** — информация, касающаяся безопасности сервера;
- **Configuration** — позволяет редактировать конфигурационный файл сервера.

Очень удобна вкладка **Configuration**, позволяющая редактировать файл конфигурации — текстовые редакторы вам больше не понадобятся (рис. 18.8).

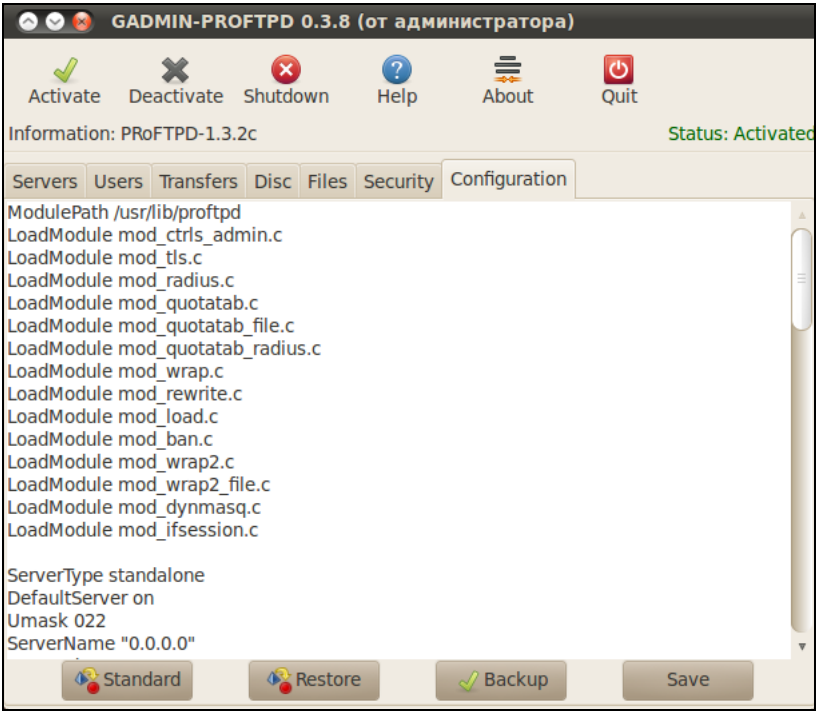


Рис. 18.8. Редактирование файла конфигурации

Теперь самое время заняться предоставлением анонимного доступа к серверу. Предположим, что наша коллекция с фильмами и другими огромными файлами будет размещаться в каталоге /var/ftp/pub. Поэтому секция `Anonymous` конфигурационного файла proftpd.conf будет выглядеть так:

```
<Anonymous /var/ftp/pub>

# Имя пользователя и группы
#(пользователь ftp создается при установке FTP-сервера)
```

```
User                        ftp
Group                       nogroup

# Определяем стандартный псевдоним anonymous
UserAlias                   anonymous ftp

# Отключает проверку командного интерпретатора пользователя — это лишнее
RequireValidShell           off

# Максимальное количество анонимных клиентов
MaxClients                  10

# Запрещаем записывать файлы в каталог /var/ftp/pub
<Directory *>
     <Limit WRITE>
       DenyAll
     </Limit>
 </Directory>

</Anonymous>
```

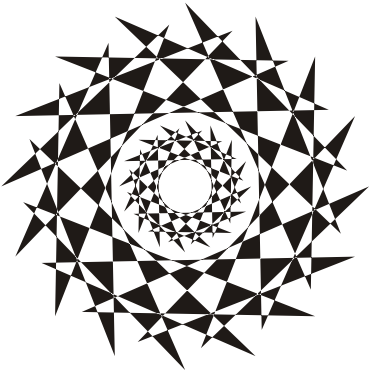
Теперь любой пользователь сможет подключиться к вашему серверу, используя имя пользователя `anonymous` и любой e-mail в качестве пароля. Они получат доступ к файлам в каталоге /var/ftp/pub. Файлы можно будет только скачивать, все остальные действия (изменение, удаление) запрещены, запрещена и загрузка файлов на сервер. Если нужно разрешить загрузку файлов на сервер, то в конце секции `Anonymous` (перед `</Anonymous>`) добавьте следующую секцию:

```
<Directory incoming>
    Umask 022
         <Limit WRITE>
            DenyAll
          </Limit>
          <Limit STOR>
             AllowAll
          </Limit>
</Directory>
```

Пользователи смогут просматривать каталог /var/ftp/pub/incoming и загружать в него новые файлы.

После внесения изменений в конфигурационный файл нужно перезагрузить сервер, чтобы сервер перечитал конфигурационный файл.

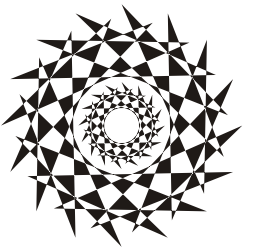
Теперь, после проделанной работы, у вас есть Web-сервер с поддержкой PHP и MySQL, сервер баз данных MySQL и FTP-сервер — все, что нужно для разработки и тестирования ваших Web-приложений.



ЧАСТЬ IV

Управление пакетами

Часть IV посвящена управлению пакетами в Linux. Мы рассмотрим способы установки, обновления и удаления программного обеспечения в разных дистрибутивах Linux.



Глава 19

Введение в пакеты. Программы *rpm* и *dpkg*

19.1. Что такое пакет

В Windows программное обеспечение устанавливается с помощью мастера установки — программы setup.exe или install.exe. Мастер установки свой для каждой программы, т. е. программа setup.exe, предназначенная для установки MS Office, не установит Photoshop.

В Linux все иначе. Здесь используются два основных способа установки программного обеспечения:

- с помощью пакетов;
- из исходных кодов.

Пакет содержит все необходимое для установки программы. Существуют два основных типа пакетов:

- RPM-пакеты — применяются во всех Red Hat-совместимых дистрибутивах (Red Hat, Fedora, Mandriva, ALT Linux, ASPLinux и др.);
- DEB-пакеты — применяются в дистрибутиве Debian и в дистрибутивах, основанных на Debian (Ubuntu, Kubuntu, Edubuntu, Denix, Mint и др.).

> ***Примечание***
>
> В Slackware Linux используется собственный формат пакетов, не совместимый ни с RPM, ни с DEB. Об установке пакетов в Slackware мы поговорим отдельно.

Если в вашем дистрибутиве нет нужной вам программы, попробуйте найти ее пакет на следующих сайтах:

- **http://rpmfind.net** и **http://rpm.pbone.net** (для RPM-пакетов);
- **http://www.debian.org/distrib/packages** и **http://packages.ubuntu.com/** (для DEB-пакетов).

Если же вы не можете найти пакет программы в Интернете, тогда придется компилировать программу самому (при условии, что вы нашли архив с исходным кодом программы). Да, в Linux некоторые программы распространяются только в исходных кодах. Для установки такой программы нужно распаковать архив с исходными кодами (желательно, в каталог /usr/src), затем перейти в только что

созданный каталог (содержащий исходные коды устанавливаемой программы) и выполнить следующие команды:

```
./configure
make
make install
```

Сценарий `configure` проверит, содержит ли ваша система необходимые библиотеки или программы, после чего, если все нормально, будет создан файл `Makefile`. Если вы увидели сообщение об ошибке, внимательно прочитайте его и попытайтесь устранить причину ошибки, например установите недостающую библиотеку. Ясно, что в случае ошибки вводить последние две команды не нужно.

Вторая команда (`make`) на основании созданного файла Makefile компилирует программу. А последняя команда (`make install`) — устанавливает программу и дополнительные файлы в дерево файловой системы (программы обычно в каталог /usr/bin, документацию — в /usr/share/doc, конфигурационные файлы — в /etc и т. д.).

> ***Совет***
>
> Для получения подробных инструкций по установке и удалению таких программ лучше всего просмотреть файл README, который обычно присутствует в архиве.

Устанавливаемая программа, как правило, состоит из набора файлов, например исполнимого и конфигурационного файлов, файла справки. В зависимости от организации программы установки все эти файлы могут быть:

- заархивированы каждый отдельно — в этом случае мы получаем набор из $N + 1$ файлов (N — это файлы программы плюс программа установки);
- заархивированы в один общий архив — у нас будет два файла: архив и программа установки;
- заархивированы в саму программу установки — самый удобный случай, когда у нас всего один файл — программа установки.

Как уже было отмечено, в современных дистрибутивах Linux все файлы, относящиеся к той или иной программе, помещаются в один файл — пакет. Пакет — это не просто архив, содержащий файлы программы. В пакете, кроме файлов программы, хранится служебная информация, описывающая процесс установки программы:

- *пути* — ведь один файл нужно скопировать, например, в каталог /usr/bin, а другой — в /usr/share/doc;
- *дополнительные действия* — например, создание каталога, установка тех или иных прав доступа к файлам и каталогам программы;
- *зависимости* — одна программа для своей работы может требовать какую-то библиотеку (без которой она не будет запускаться, поскольку использует функции этой библиотеки). Тогда в пакете указывается, что он *зависит* от другого пакета, содержащего библиотеку. При установке менеджер пакетов проверяет зависимости: если установлены не все пакеты, от которых зависит устанавли-

ваемый пакет, установка будет прервана — пока вы не установите все необходимое. Правда, имеется возможность установки программы без удовлетворения зависимостей (тогда информация о зависимостях будет просто проигнорирована), но в большинстве случаев установленная таким образом программа работать не будет;

- *конфликты* — аналогично, одна программа может в системе конфликтовать с другой программой. Например, программы `sendmail` и `postfix` являются MTA-агентами (MTA, Mail Transfer Agent). Поскольку в системе может быть только один MTA-агент, установить можно или `sendmail`, или `postfix`, т. е. пакет sendmail конфликтует с пакетом postfix и наоборот.

Пакеты называются также RPM-файлами (или DEB-файлами — для дистрибутивов на основе Debian). С Debian все просто: пакеты были так названы, потому что последние три символа имени у файлов пакетов — deb (сокращение от Debian). Название RPM-файлов берет начало с разработок компании Red Hat, которая впервые предложила технологию RPM. Тогда в дистрибутиве Red Hat появился менеджер пакетов rpm (Red Hat Package Manager), откуда и название пакетов.

В имени пакета зашифрована некоторая информация о программе. Сделано это исключительно для удобства — можно узнать версию и другую информацию о программе, только лишь взглянув на название пакета, например:

```
program-1.5-14.i586.rpm
```

Здесь `program` — название программы, `1.5` — ее версия, `14` — выпуск пакета, `i586` — архитектура, на которую рассчитана программа. Не нужно пытаться устанавливать программы для архитектур i586/686 на компьютер с процессором Intel 386 или 486. Если программа независима от архитектуры, то указывается параметр `noarch` (обычно так делают для документации, примеров конфигурационных файлов, т. е. для пакетов, содержащих информацию, которая не зависит от архитектуры).

19.2. Репозитарии пакетов

Репозитарий — это хранилище пакетов. Репозитарий может быть локальным, например каталог на жестком диске или DVD-диск, или же сетевым — сервер в Интернете или в локальной сети, содержащий RPM-пакеты. Для чего создаются репозитарии? Для централизованного управления обновлением пакетов. Представьте, что у нас нет репозитариев. Тогда, чтобы узнать, вышла ли новая версия нужной вам программы, вам пришлось бы посещать сайт ее разработчика или по крайней мере сайт разработчика дистрибутива Linux. А это не очень удобно. Один раз вы можете забыть, а потом вообще вам надоест это дело. Проще дождаться выхода новой версии дистрибутива и обновить все программы за один раз.

Так и было раньше. И если бы не забота разработчиков Linux о нас с вами, репозитарии вообще не были бы созданы. Сейчас поясню. Вот вышла программа, ее включили в состав дистрибутива, но полностью не протестировали (протестировать все невозможно). Оказалось, что программа работает неправильно,

но только при определенных условиях, например с определенным форматом файла. Или же Linux был установлен на сервер, были установлены сетевые службы, например Web-сервер. Через некоторое время оказалось, что в этой версии Web-сервера есть "дыра", поэтому вскорости выпустили новую версию. Пользователь, установивший программу, ничего не подозревая о том, что вышла новая ее версия, мог бы мучаться минимум полгода или даже год — до выхода следующей версии дистрибутива. А тот сервер могли бы взломать уже на следующий день после обнаружения "дыры". Но не тут-то было. С помощью репозиториев можно быстро и удобно отслеживать обновления тех или иных пакетов. Причем это делает сам менеджер пакетов, а вам лишь остается указать, какие обновления нужно загружать, а какие — нет.

Практически все системы управления пакетами современных дистрибутивов поддерживают хранилища пакетов. В следующем разделе мы рассмотрим программы управления пакетами, использующиеся в современных дистрибутивах.

19.3. Программы для управления пакетами

Для управления пакетами в разных дистрибутивах используются разные программы. В табл. 19.1 приведены программы управления пакетами, которые можно встретить в современных дистрибутивах.

Таблица 19.1. *Программы управления пакетами*

Программа	Дистрибутив	Описание
`rpm`	Red Hat-совместимые дистрибутивы (Fedora, Mandriva, ALT Linux, ASPLinux, openSUSE и др.)	Простой менеджер пакетов. Работает в текстовом режиме. Не умеет разрешать зависимости пакетов
`rpmdrake`	Дистрибутивы, основанные на Mandriva	Графический менеджер пакетов. Умеет разрешать зависимости и управлять источниками пакетов
`urpmi`	Дистрибутивы, основанные на Mandriva	Текстовый менеджер пакетов, поддерживающий источники пакетов и автоматически разрешающий зависимости
`dpkg`	Дистрибутивы, основанные на Debian (Ubuntu, Kubuntu и др.)	Простой менеджер пакетов. Работает в текстовом режиме. Не умеет разрешать зависимости пакетов
`apt`	Debian, Ubuntu (и клоны), ALT Linux и др.	Мощный менеджер пакетов, работающий в текстовом режиме. Умеет разрешать зависимости пакетов и поддерживает репозитарии (источники пакетов)
`yum`	Fedora и др.	Мощный менеджер пакетов, работающий в текстовом режиме. Умеет разрешать зависимости пакетов и поддерживает репозитарии (источники пакетов)

Таблица 19.1 *(окончание)*

Программа	Дистрибутив	Описание

`gpk-application`
`pirut`
или
`system-config-packages`
	Fedora и дистрибутивы, основанные на нем (ASPLinux)	Графический менеджер пакетов. Впервые появился в одной из последних версий дистрибутива Red Hat, затем "перекочевал" в Fedora. По функциям похож на `rpmdrake`, хотя последний все же удобнее. В любом случае в Fedora вам придется довольствоваться только этим менеджером (если не считать yum).
		В последних версиях Fedora используется программа `gpk-application`

`pkgtool`	Slackware	Менеджер пакетов Slackware, заслуживающий отдельного разговора
`zypper`	openSUSE	Менеджер пакетов SUSE. Работает в текстовом режиме. Умеет разрешать зависимости пакетов

Примечание

Наверное, в таблице вы обратили внимание на фразу "умеет разрешать зависимости пакетов". Это означает следующее: если при установке пакета будет обнаружено, что для корректной его установки ему нужны дополнительные пакеты, то менеджер пакетов установит их. Если же менеджер пакетов не умеет разрешать зависимости, то он только сообщит, что установить пакет невозможно, и выведет лишь список файлов (файлов, а не пакетов!), которые нужны для установки данного пакета. А уж какой файл в каком пакете находится, вам придется догадываться самостоятельно.

19.4. Программа *rpm* (все Red Hat-совместимые дистрибутивы)

Если вы хотите установить пакет, который не входит в состав дистрибутива (например, загруженный из Интернета), вам следует использовать программу `rpm` (для установки пакетов, которые входят в состав дистрибутива, намного удобнее использовать графический менеджер пакетов `rpmdrake`).

Программа `rpm` — полноценный текстовый менеджер пакетов, позволяющий устанавливать, удалять пакеты, просматривать информацию об уже установленных и новых пакетах, обновлять пакеты.

Чтобы установить пакет с помощью `rpm`, выполните команду:

```
# rpm -ihv <имя_пакета>
```

Удалить пакет так же просто:

```
# rpm -e <имя_пакета>
```

Для обновления пакета используется команда:

```
# rpm -U <имя_пакета>
```

Просмотреть, установлен ли тот или иной пакет, можно с помощью команды:

```
# rpm -qa | grep <имя_пакета>
```

Если вы хотите просмотреть информацию о пакете, то введите команду:

```
# rpm -qi <имя_пакета>
```

Просмотреть список файлов, входящих в состав пакета, можно командой:

```
# rpm -ql <имя_пакета>
```

Наконец, вывести все пакеты можно командой:

```
$ rpm -qa | grep more
```

Примечание

Программа `rpm` может также использоваться и для сборки собственных пакетов, но данная операция выходит за рамки этой книги. Вы можете прочитать мою статью о сборке собственных RPM-пакетов на сайте **http://www.dkws.org.ua/index.php?page=show&file=a/system/rpm_create**.

19.5. Программа *rpmbuild*: простая сборка пакетов исходного кода

Обычно пакеты содержат уже откомпилированные версии программ. Но встречаются также и пакеты, содержащие исходный код. У таких пакетов "двойное расширение", т. е. не просто .rpm, а .src.rpm. При установке такого пакета будет установлен исходный код программы, который можно будет потом откомпилировать. Но если вам нужна сама программа, а не ее исходный код, нет необходимости устанавливать пакет, а потом компилировать исходный код. Проще сразу вызвать программу `rpmbuild` так:

```
# rpmbuild --rebuild package.src.rpm
```

Если в процессе выполнения команды не возникнет ошибка, собранный RPM-пакет, содержащий уже откомпилированную версию программы, вы найдете в каталоге /usr/src/redhat/RPMS. Все, что вам останется, — это установить пакет с помощью команды `rpm`.

19.6. Программа *dpkg*: управление DEB-пакетами

Программа `dpkg` используется для установки, удаления и управления пакетами Debian/Ubuntu. Программа `dpkg` вызывается из командной строки. Формат вызова следующий:

```
dpkg [ключи] действие
```

Для запуска `dpkg` нужно обладать полномочиями root, получить которые можно с помощью команды `sudo`. Рассмотрим, как нужно использовать программу `dpkg`.

Предположим, у нас есть пакет package.deb. Для его установки введите команду:

```
sudo dpkg -i /путь/package.deb
```

Как видите, для установки пакета не нужно делать ничего сложного. Если вам интересно, то процесс установки пакета состоит из следующих шагов:

1. Извлечение управляющих файлов из пакета.
2. Если уже была установлена старая версия этого пакета, то из старого пакета запускается сценарий `prerm` (данный сценарий подготавливает систему к удалению старой версии пакета). Другими словами, если нужно, то обновление пакета выполняется автоматически.
3. Выполняется сценарий `preinst`, если он есть в данном пакете.
4. Распаковываются остальные файлы из пакета (если был установлен старый пакет, то файлы не удаляются, а сохраняются в другом месте, чтобы их можно было восстановить, если что-то пойдет не так).
5. Если была установлена старая версия пакета, то выполняется сценарий `postrm` (действия после удаления) из старого пакета. Данный сценарий запускается сразу после выполнения сценария `preinst` нового пакета, поскольку старые файлы удаляются во время записи новых файлов.
6. Выполняется настройка пакета:
   - распаковываются новые конфигурационные файлы, а старые сохраняются, если нужно будет их восстановить в случае ошибки во время установки нового пакета;
   - запускается сценарий `postinst`, если он есть в данном пакете.

Удалить пакет тоже просто:

```
sudo dpkg -r package
```

При удалении пакета не нужно указывать путь к пакету и расширение пакета.

Но установка и удаление пакетов — это далеко не единственные действия, которые можно выполнить с помощью программы `dpkg`. Другие действия программы `dpkg`, которые могут быть интересны каждому пользователю Ubuntu, представлены в табл. 19.2.

Таблица 19.2. *Вспомогательные действия программы* `dpkg`

Действие	Описание
`-l [`*образец*`]`	Вывести все установленные пакеты, имена которых соответствуют образцу. Образец задается с помощью масок `*` и `?`, например образец `a*` соответствует любому имени пакета, начинающемуся на букву "a" (рис. 19.1). Если образец не задан, выводятся все пакеты
`-L <`*имя_пакета*`>`	Вывести имена файлов из указанного пакета (пакет должен быть установлен)
`-p <`*имя_пакета*`>`	Вывести информацию об установленном пакете
`-s <`*имя_пакета*`>`	Вывести информацию о статусе пакета
`--unpack <`*имя_пакета*`.deb>`	Распаковать, но не устанавливать пакет (полезно, если устанавливать пакет не нужно, а нужно достать из него один или несколько файлов)

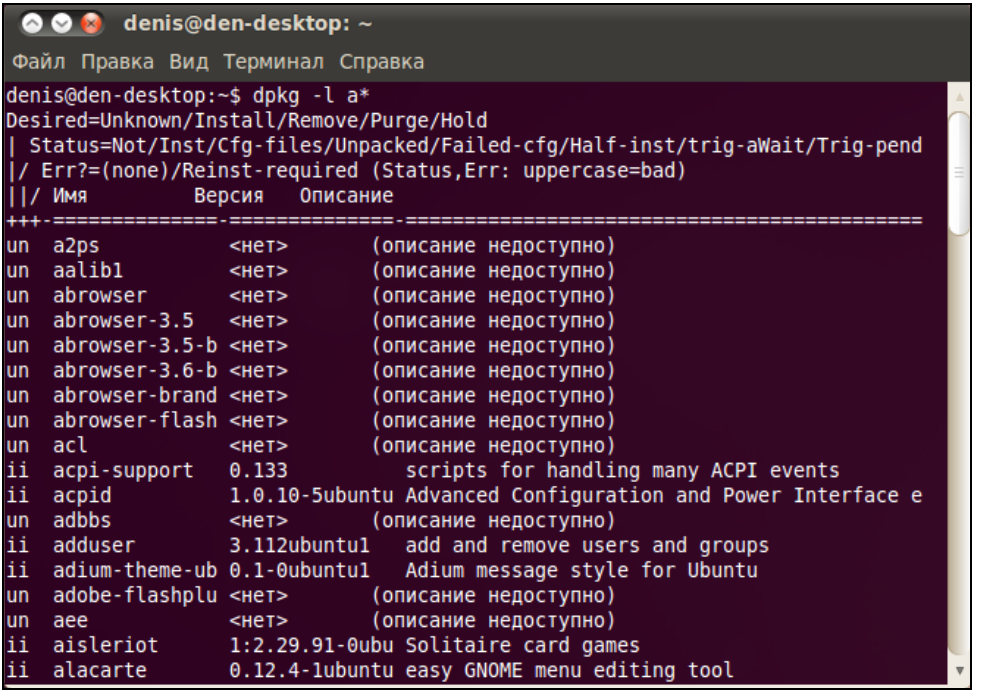


Рис. 19.1. Вывод всех пакетов, имена которых начинаются на букву "a"

Если вы хотите получить более подробную информацию о программе `dpkg`, введите команду (страница руководства будет на русском языке):

```
man dpkg
```

19.7. Команда *alien*: установка RPM-пакетов

Если у вас есть RPM-файл, его можно преобразовать в формат DEB с помощью команды `alien`. Сразу хочу заметить, что установка таких пакетов крайне нежелательна, поскольку нет никакой гарантии, что установленная программа будет работать, но если другого выхода нет, можно попробовать:

```
sudo alien package_file.rpm
```

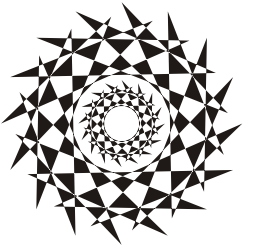
Если система сообщит вам, что команда `alien` не найдена, то ее нужно установить с помощью команды:

```
sudo apt-get install alien
```

Перед выполнением этой команды нужно подключиться к Интернету.

Удалить программу можно с помощью команды:

```
sudo apt-get remove alien
```



Глава 20

Управление пакетами в Debian/Ubuntu

20.1. Программы для управления пакетами

Для управления пакетами в Debian/Ubuntu используются следующие программы:

- `dpkg` — базовая программа, не умеет разрешать зависимости пакетов и не поддерживает источники пакетов;
- `apt-get` — поддерживает источники пакетов и умеет разрешать зависимости. Чаще всего вы будете использовать эту программу;
- `aptitude` — программа для установки пакетов с псевдографическим интерфейсом, умеет разрешать зависимости и поддерживает источники пакетов;
- `alien` — используется для установки RPM-пакетов в Debian, была рассмотрена в предыдущей главе.

20.2. Программа *apt-get*

20.2.1. Установка пакетов. Источники пакетов

В предыдущей главе была рассмотрена базовая программа управления пакетами в Debian — `dpkg`. Функции у этой программы тоже базовые, поэтому использовать вы ее будете очень редко. Вместо нее вы будете использовать более функциональную программу `apt-get`.

Программа `apt-get` применяется не только в Debian/Ubuntu/Denix, но и в других дистрибутивах, причем даже в Red Hat-совместимых (например, в ALT Linux), но там она используется для установки RPM-пакетов, а не DEB. Вообще, выбор менеджера пакетов зависит от разработчиков дистрибутива. В одной версии дистрибутива может использоваться `apt-get`, в другой — `yum`, а в третьей — какой-то новый и перспективный менеджер пакетов.

Предположим, что у нас есть пакет package.deb. При его установке обнаружилось, что он требует пакет lib.deb, который не установлен. Вы находите в Интернете нужный пакет, устанавливаете его, а затем устанавливаете пакет package.deb. Не очень удобно, правда?

Намного проще выполнить команду:

```
sudo apt-get install package
```

Программа `apt-get` просматривает файл /etc/apt/sources.list — в этом файле перечислены источники (репозитарии) DEB-пакетов. В качестве источника может выступать как компакт-диск, содержащий пакеты, так и сервер в Интернете. Программа находит указанный пакет, читает служебную информацию о нем, затем разрешает зависимости (т. е. устанавливает все другие пакеты, нужные для работы программ устанавливаемого пакета), а после устанавливает нужный нам пакет. Все загруженные программой `apt-get` и менеджером Synaptic (о нем — далее) пакеты записываются в каталог /var/cache/apt/archives.

Взглянем на файл /etc/apt/sources.list:

```
sudo nano /etc/apt/sources.list
```

Найдите и раскомментируйте следующую строку:

```
deb http://ru.archive.ubuntu.com/ubuntu/ lucid-backports main restricted
universe multiverse
```

Данная строка подключает репозитарий backports, содержащий много полезных программ. Данный репозитарий содержит не всегда бесплатные программы и не поддерживается командой Ubuntu.

Также можно раскомментировать строку, подключающую репозитарий партнеров Canonical (компании-разработчика Ubuntu):

```
deb http://archive.canonical.com/ubuntu lucid partner
```

Наверное, вам интересно, какие программы находятся в том или ином репозитарии Ubuntu? В репозитарии main находятся основные программы, они распространяются свободно и регулярно поддерживаются (обновляются). В репозитарии restricted содержатся программы, которые распространяются по несвободным лицензиям, а также имеют ограниченную поддержку. Репозитарий universe содержит программы с открытыми лицензиями, поддержка программ из этого репозитария не гарантируется, но вполне возможна, все зависит от разработчика программы. В репозитарии multiverse содержатся программы, которые распространяются несвободно и без всякой поддержки и гарантий. Репозитарий security содержит исправления пакетов из репозитариев main и restricted. Наконец, в репозитарии backports есть неофициальные пакеты свежих версий программ, собранные из исходных текстов энтузиастами Ubuntu (а не разработчиками программ).

Чтобы настроить менеджер пакетов на русские репозитарии (соответственно скорость загрузки пакетов будет выше), замените во всех строках файла /etc/apt/sources.list адрес `archive.ubuntu.com` на `ru.archive.ubuntu.com`.

20.2.2. Основные команды программы *apt-get*

Понятно, что программа `apt-get` может использоваться не только для установки пакетов. Общий формат вызова этой программы следующий:

```
apt-get [опции] команды [пакет]
```

Основные команды `apt-get` представлены в табл. 20.1.

Таблица 20.1. *Основные команды программы* `apt-get`

Команда	Описание
`update`	Синхронизирует файлы описаний пакетов (внутреннюю базу данных о пакетах) с источниками пакетов, которые указаны в файле /etc/apt/sources.list
`upgrade`	Обновляет указанный пакет. Может использоваться для обновления всех установленных пакетов. При этом установка новых пакетов не производится, а загружаются и устанавливаются только новые версии уже установленных пакетов
`dist-upgrade`	Обновление дистрибутива. Для обновления всех пакетов рекомендуется использовать именно эту команду
`install`	Установка одного или нескольких пакетов
`remove`	Удаление одного или нескольких пакетов
`check`	Используется для поиска нарушенных зависимостей
`clean`	Используется для очистки локального хранилища полученных пакетов (перед установкой пакет загружается в локальное хранилище, а затем устанавливается оттуда; данная команда может очистить хранилище для экономии дискового пространства)
`autoclean`	Похожа на `clean`, но удаляет только те файлы пакетов, которые больше не могут быть получены и использованы

Рассмотрим подробнее базовые операции с пакетами.

Обновление источников

После каждого изменения файла sources.list нужно вводить команду:

```
sudo apt-get update
```

Если не ввести эту команду, то изменения из файла sources.list не будут прочитаны и программа `apt-get` будет использовать старый список источников пакетов.

Удаление и переустановка пакетов

Для удаления пакета используется команда `remove`:

```
sudo apt-get remove <пакет>
```

Некоторые пользователи в случае сбоя программы пытаются ее переустановить. Для этого они ее удаляют, а затем устанавливают заново. Это в корне неправильно. Чтобы переустановить пакет, гораздо правильнее использовать следующую команду:

```
sudo apt-get —-reinstall install <пакет>
```

Если вы все-таки хотите сначала удалить пакет, а потом установить его снова, то вместо удаления нужно использовать зачистку, когда удаляются не только исполнимые файлы программ, но и конфигурационные файлы, что очень важно. Есть вероятность, что причина некорректной работы программы кроется в неправильных настройках. А если вы удалите программу, но оставите настройки, после по-

вторной установки программы проблема останется. Для удаления и программы, и конфигурационных файлов нужно использовать опцию `--purge`:

```
sudo apt-get --purge remove <пакет>
```

Обновление пакета и системы

Для обновления одного пакета используется команда `upgrade`, а для обновления всей системы — команда `dist-upgrade`. Некоторые пользователи пытаются обновить всю систему командой `upgrade`, но этого не нужно делать, иначе велика вероятность "падения" системы после такого обновления. Команды обновления пакета и системы выглядят так:

```
sudo apt-get upgrade <пакет>
sudo apt-get dist-upgrade
```

Очистка кэша пакетов

Как мы уже знаем, установка программы состоит из трех основных этапов: загрузка пакетов, установка пакетов и настройка пакетов. Скачанные с Интернета пакеты помещаются в каталог /var/cache/apt/archives. Регулярно очищайте этот каталог — этим вы сэкономите немало места на диске. Если же у вас соединение с учетом трафика, тогда записывайте скачанные пакеты на CD/DVD. Если нужно будет переустановить Ubuntu, то вы можете установить все необходимые вам программы, не загружая их повторно с Интернета. Удалить все DEB-пакеты из каталога /var/cache/apt/archives можно командой:

```
sudo rm /var/cache/apt/archives/*.deb
```

Вообще вместо ручного удаления можно использовать команду `clean`:

```
sudo apt-get clean
```

Но эта команда оставляет некоторые файлы, поэтому лучше удалить пакеты вручную. На некоторых системах команда `sudo apt-get clean` включена в автозапуск, поэтому при каждой перезагрузке скачанные пакеты будут удалены.

Если вам интересно, вот назначение каталогов локального кэша:

- /var/cache/apt/archives/ — область хранения полученных файлов;
- /var/cache/apt/archives/partial/ — область хранения получаемых файлов (пакетов);
- /var/state/apt/lists/ — область хранения информации о состоянии каждого ресурса пакетов, который задан в списке источников;
- /var/state/apt/lists/partial/ — временная область хранения информации в процессе скачивания.

Опции программы *apt-get*

Иногда нужно не просто выполнить команду, а указать программе, как ее нужно выполнить. Например, можно попросить программу удалить пакет, а можно уточнить, что вы хотите также удалить и конфигурационные файлы, как было показано ранее. Для такого рода уточнений используются опции команды `apt-get`. Список самых полезных опций приведен в табл. 20.2.

Таблица 20.2. *Самые полезные опции программы* `apt-get`

Опция	Описание
`-d, --download-only`	Только скачать пакеты. Пакеты будут скачаны с источника пакетов и помещены в локальный кэш пакетов, но не будут установлены
`-f, --fix-broken`	Программа попытается исправить систему с нарушенными зависимостями. Если опция используется с командами установки и удаления пакетов, то, возможно, некоторые пакеты будут пропущены, чтобы разрешить все зависимости. Вообще `apt-get` не допускает нарушений зависимостей, но если вы удалили некоторые пакеты с помощью `dpkg`, которая не поддерживает зависимостей, то возможны нарушения зависимостей, которые можно попытаться исправить этой опцией
`-m, --ignore-missing, --fix-missing`	Игнорировать отсутствующие пакеты. Позволяет пропустить некоторые пакеты при установке (например, те, которые не могут быть получены или загрузились с ошибками), но такое не всегда возможно, т. к. очень часто устанавливаемые группой пакеты зависят друг от друга
`-q, --quiet`	Тихий режим. Выводится минимум сообщений. Полезно для использования в сценариях `bash`/`tcsh`
`-s, --simulate`	Симулирует процесс установки, пакеты реально не устанавливаются. Состояние системы не изменяются, проверяется возможность установки пакетов
`-y, --yes`	Автоматически отвечать "да" на все вопросы программы, полезно при написании сценариев
`-b, --compile, --build`	Откомпилировать пакеты исходного кода после их получения
`--no-upgrade`	Не производить обновление пакетов. Обычно используется вместе с install, когда при установке пакета не будет произведено обновление уже установленных пакетов
`--force-yes`	Принудительное выполнение заданной операции. Очень опасная операция, которая может потенциально уничтожить систему. Лучше не использовать эту опцию!
`--purge`	Полное удаление пакета. Данная опция была описана ранее
`--reinstall`	Переустанавливает пакет. Данная опция была описана ранее

Покдлючение репозитария Medibuntu в Ubuntu

Репозиторий Medibuntu содержит кодеки и различные проигрыватели мультимедиа. Для его установки введите команды:

```
sudo wget http://www.medibuntu.org/sources.list.d/$(lsb_release -cs).list \
 --output-document=/etc/apt/sources.list.d/medibuntu.list &&
sudo apt-get -q update &&
sudo apt-get --yes -q --allow-unauthenticated install medibuntu-keyring &&
sudo apt-get -q update
```

Корова в *apt-get*

Оказывается, пасхальные яйца есть не только в Windows, но и в Linux. Введите команду `sudo apt-get moo`, и вы увидите изображение… коровы (рис. 20.1).

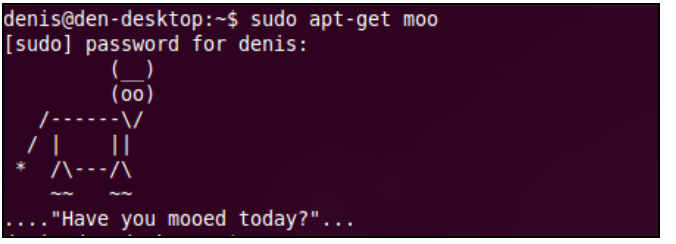


Рис. 20.1. Корова в *apt-get*

20.3. Программа *aptitude*

Программа `aptitude` обладает псевдографическим интерфейсом, что облегчает установку программ новичкам, если они-таки окажутся в консоли. Ситуации бывают всякие. Например, новичок привык использовать Synaptic (графический менеджер в Ubuntu), но после очередного эксперимента графический интерфейс уже не запускается, а установить пакеты нужно, например переустановить пакет графического драйвера. В этой ситуации как раз и поможет `aptitude`. Хотя, если вы разберетесь с `apt-get`, использовать `aptitude` вам не захочется — до сих пор сам не могу к ней привыкнуть (рис. 20.2).

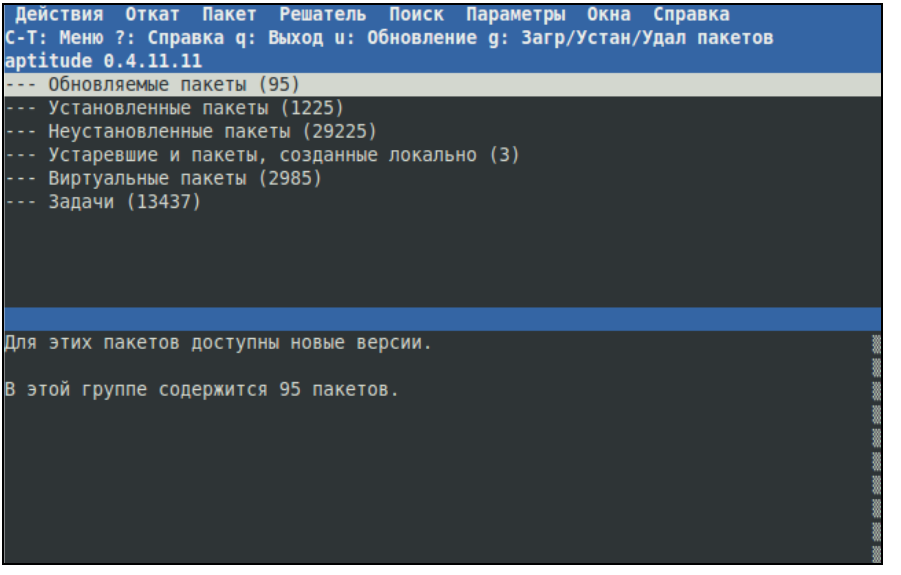
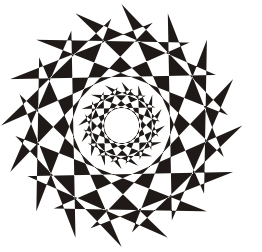


Рис. 20.2. Программа `aptitude`



Глава 21

Управление пакетами в Fedora

21.1. Использование программы *yum*

Программа `yum` (Yellow dog Updater Modified) используется во многих дистрибутивах, в том числе и в Fedora.

`Yum` работает аналогично другим подобным программам (`urpmi`, `apt`) — когда вы устанавливаете пакет, `yum` производит поиск пакета в репозиториях, перечисленных в конфигурационном файле, загружает пакет и устанавливает его. В качестве репозитария могут выступать как дистрибутивные диски, так и серверы Интернета.

Общий формат вызова `yum` выглядит так:

```
yum команда [пакет(ы)]
```

Команды программы `yum` приведены в табл. 21.1.

Таблица 21.1. *Использование программы* `yum`

Команда	Описание
`yum install пакет`	Установить пакета из репозитария (также устанавливаются пакеты, необходимые для работы устанавливаемого пакета, т. е. разрешаются зависимости)
`yum remove пакет`	Удалить пакет, а также все пакеты, которые зависят от данного
`yum update`	Проверить наличие обновлений всех пакетов. Если обновления есть, то они будут установлены
`yum update пакет`	Проверить обновления конкретного пакета. Если есть свежая версия, то она будет установлена
`yum check-update`	Только проверка наличия обновлений (обновления не устанавливаются)
`yum check-update пакет`	Проверка наличия обновлений конкретного пакета (обновления не устанавливаются)
`yum info пакет`	Вывести информацию о пакете
`yum list`	Выводит список всех пакетов. Выводятся как установленные, так и доступные для установки (в репозиториях) пакеты
`yum list a*`	Вывести список всех пакетов, которые начинаются на букву "a"

Таблица 21.1 *(окончание)*

Команда	Описание
`yum search строка`	Найти все пакеты, в описаниях которых есть указанная строка
`yum groupinstall "группа"`	Установить все пакеты из указанной группы
`yum grouplist`	Вывести список групп пакетов

При установке пакетов с помощью `yum` не нужно далеко отходить от компьютера. Довольно часто нужные пакеты находятся не на локальных источниках, а на серверах в Интернете, поэтому `yum` выведет общий объем пакетов, которые вы собираетесь установить, и спросит вас, хотите ли вы их установить:

```
Total download size: 10.5 M
It this ok [Y/N]:
```

Если вы согласны для установки выбранных пакетов загрузить 10,5 Мбайт файлов, нажмите клавишу <Y>, если передумали — нажмите <N>. Довольно удобно, иначе (с учетом того, что при разрешении зависимостей будут установлены дополнительные пакеты) можно при установке одного небольшого, на первый взгляд, пакета превысить месячную норму по трафику.

Получить информацию о пакете, как было показано в табл. 21.1, можно с помощью команды:

```
yum info пакет
```

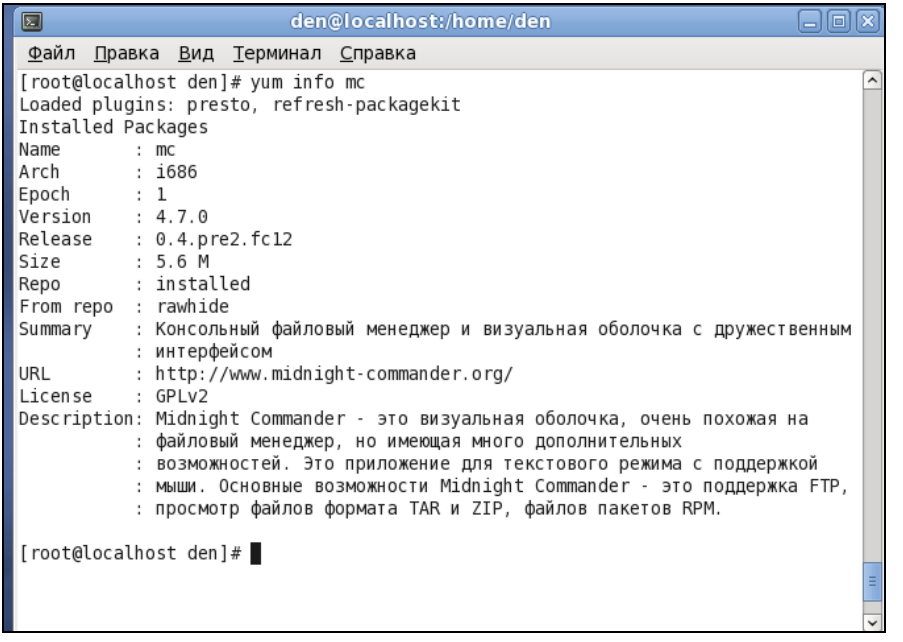


Рис. 21.1. Вывод информации о пакете

При этом на экран выводится следующая информация (рис. 21.1):

- **Name** — имя пакета;
- **Arch** — архитектура компьютера;
- **Epoch** — как бы подверсия пакета. Используется, когда требуется уменьшить версию или релиз пакета по сравнению с имеющимся в репозитарии;
- **Version** — версия пакета;
- **Release** — релиз пакета (можете считать это подверсией пакета);
- **Size** — размер занимаемого места на диске;
- **Repo** — хранилище пакета или значение **installed**, если пакет уже установлен;
- **From repo** — из какого хранилища пакет был установлен (для установленных пакетов);
- **Summary** — общая информация о пакете;
- **URL** — Web-страница разработчика программы;
- **License** — лицензия, по которой распространяется программа;
- **Description** — описание пакета.

Для вывода всех пакетов можно использовать команду `yum list`, но пакетов слишком много, поэтому использовать ее неудобно. Удобнее задать маску имени пакета, например `yum list a*`, — в этом случае будут выведены все пакеты, начинающиеся на букву "a".

21.2. Управление источниками пакетов

Источники пакетов `yum` описываются в файле конфигурации /etc/yum.conf. Откройте этот файл (листинг 21.1).

> ***Совет***
>
> Обычно файл /etc/yum.conf приходится редактировать редко. Но помните, что делать это можно только от имени пользователя root. Если вы привыкли к графическому режиму, то в терминале для редактирования этого файла нужно ввести команду:
> `su -c <редактор> /etc/yum.conf`

Листинг 21.1. Конфигурационный файл yum.conf

```
[main]
cachedir=/var/cache/yum
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
```

```
metadata_expire=1800

# PUT YOUR    REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

Ранее репозитарии описывались непосредственно в файле yum.conf (как в случае с urpmi.cfg). Но потом было принято решение хранить описания репозитариев в отдельных файлах (REPO-файлах) в каталоге /etc/yum.repos.d. Каждый файл в этом каталоге называется так: `<имя репозитария>.repo`.

В листинге 21.2 приведен пример описания источника пакетов Fedora, взятый из файла fedora.repo.

Листинг 21.2. Пример описания источника пакетов

```
[fedora]
name=Fedora $releasever — $basearch
ba-
seurl=http://download.fedora.redhat.com/pub/fedora/linus/releases/$releasever/
Everything/$basearch/os/
mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=fedora-
$releasever&arch=$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora file:///etc/pki/rpm-gpg/RPM-
GPG-KEY
```

Теперь разберемся, что здесь что. В квадратных скобках указывается сокращенное имя репозитария. Параметр `name` задает полное имя источника пакетов. Интернет-адрес (URL) источника пакетов указан параметром `baseurl`, а параметр `mirrorlist` задает список зеркал — копий репозитария, которые будут использоваться, если URL источника, указанный в `baseurl`, недоступен.

Параметр `enabled`, установленный в `1`, указывает на то, что данный источник активный, и `yum` использует его при установке пакетов. Следующий параметр, `gpgcheck`, указывает на то, что программа `yum` должна проверить подпись источника (если `gpgcheck=1`), а ключ, используемый для проверки подписи, задан параметром `gpgkey`.

Добавление источника производится путем добавления соответствующего ему REPO-файла в каталог /etc/yum.repos.d. Где этот файл взять? Обычно такие файлы представлены в виде RPM-пакетов на Web-серверах репозитариев. Поэтому нужно просто скачать RPM-пакет и установить его. Например, для установки REPO-файла популярного репозитария RPM Fusion нужно выполнить команду:

```
su -c 'rpm -Uvh http://download1.rpmfusion.org/free/fedora/rpmfusion-free-
release-stable.noarch.rpm
http://download1.rpmfusion.org/nonfree/fedora/rpmfusion-nonfree-release-
stable.noarch.rpm'
```

Что делает данная команда, ясно и без комментариев. Если вы не можете найти соответствующий источнику REPO-файл, его можно написать вручную по формату листинга 7.3. При этом нужно еще знать базовый URL источника пакетов.

Удалять файлы источников пакетов, если сам источник уже не нужен, совсем не обязательно. Достаточно установить параметр `enabled` для источника в `0`. Тогда этот источник не будет использоваться.

21.3. Установка пакетов через прокси-сервер

По умолчанию `yum` полагает, что наш компьютер напрямую подключен к Интернету (не через прокси-сервер). Если вы подключаетесь к Интернету по локальной сети, т. е. через прокси-сервер, данный факт нужно отразить в файле yum.conf, иначе вы не сможете устанавливать пакеты.

Узнайте у администратора сети параметры подключения к прокси-серверу (адрес, порт, имя пользователя и пароль) и пропишите их в файле yum.conf таким вот образом:

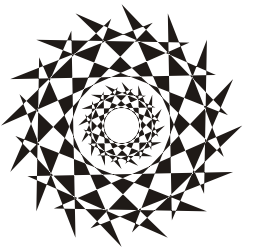
```
# Адрес прокси и его порт
proxy=http://proxy.company.ru:8080
# Имя пользователя и его пароль
proxy_username=dhsilabs
proxy_password=secret
```

21.4. Плагины для программы *yum*

Для `yum` доступно множество плагинов. Мы установим два плагина: fastestmirror и presto. Первый плагин позволяет найти самый быстрый источник пакетов, что существенно сокращает время установки пакетов. А второй пытается загружать только обновленные части пакетов вместо полной загрузки пакетов при обновлении, что сокращает трафик и уменьшает время обновления.

Для установки этих плагинов введите команды:

```
# yum install yum-plugin-fastestmirror
# yum install yum-presto
```



Глава 22

Управление пакетами в openSUSE. Менеджер пакетов *zypper*

Менеджер пакетов `zypper` работает по уже знакомому нам сценарию. Имеется список источников пакетов (каталог /etc/zypp/repos.d), который просматривается перед установкой пакета с целью определения хранилища, в котором находится устанавливаемый пакет. Затем менеджер пакетов загружает необходимый пакет (или пакеты) и устанавливает его.

Зайдите в каталог /etc/zypp/repos.d. В нем вы обнаружите несколько REPO-файлов, в каждом из которых прописан один репозитарий. В листинге 22.1 представлен репозитарий установочного DVD.

Листинг 22.1. Репозитарий установочного DVD (локальный репозитарий)

```
[openSUSE 11.2-0]
name=openSUSE 11.2-0
enabled=1
autorefresh=0
baseurl=cd:///
path=/
type=yast2
gpgcheck=1
keeppackages=0
```

Параметр `baseurl` задает путь к источнику пакетов, а параметр `enabled`, установленный в `1`, говорит о том, что этот репозитарий активный.

Параметр `gpgcheck` означает проверку подписей GPG. Если ключей для репозитария нет, можно выключить этот параметр (однако из соображений безопасности это не рекомендуется). Выключать данный параметр нужно, только если вы полностью доверяете источнику пакетов. Если включен параметр `keeppackages`, менеджер пакетов не будет удалять пакеты после их установки.

Пример сетевого источника пакетов Main Repository (OSS) приведен в листинге 22.2.

Листинг 22.2. Пример сетевого репозитария

```
[repo-oss]
name=openSUSE-11.2-Oss
enabled=1
autorefresh=1
baseurl=http://download.opensuse.org/distribution/11.2/repo/oss/
path=/
type=NONE
keeppackages=0
```

Как видите, параметр `baseurl` указывает не на локальное устройство, а на сервер в Интернете. Также обратите внимание на опцию `autorefresh` (автоматическое обновление) — для сетевого репозитария она установлена в `1`, поскольку пакеты в хранилище могут меняться (например, там появляются новые версии пакетов). А для локального репозитария автоматическое обновление отключено, потому что пакеты в нем будут одни и те же.

Если установить опцию `keeppackages` в 1, то для этого репозитария менеджер пакетов будет сохранять все загруженные пакеты. Если `keeppackages`=0, то после установки загруженный пакет удаляется.

Основной файл конфигурации менеджера пакетов называется /etc/zypp/zypp.conf, но в нем нет ничего интересного — обычно все опции там закомментированы, поскольку параметры по умолчанию устраивают всех, и их редко приходится менять.

> ***Примечание***
>
> Если у вас нет соединения с Интернетом или же оно медленное, вам придется использовать только один источник пакетов — локальный установочный DVD. Поэтому откройте терминал, введите команду `su`, а затем — `gedit`. Вы запустите обычный текстовый редактор от имени администратора. Перейдите в каталог /etc/zypp/repos.d и откройте все файлы кроме openSUSE 11.2-0.repo (этот файл описывает установочный DVD). Установите для всех сетевых источников пакетов параметр `enabled` в 0.

Файлы репозитариев обычно не нужно подключать вручную — вы скачиваете из Интернета YMP-файл, в котором описаны все необходимые репозитарии и пакеты, которые нужно установить (хотя могут быть прописаны только репозитарии — без пакетов). Данный файл представлен в формате XML (eXtended Markup Language). В секции `<repository>` описывается один репозитарий. Если репозитариев несколько, то и секций `<repository>` будет несколько. В листинге 22.3 представлена секция `<repository>` YMP-файла для главного сетевого репозитария — Main Repository (OSS).

Листинг 22.3. Секция <repository> YMP-файла для главного сетевого репозитария

```
<repository recommended="true">
        <name>Main Repository (OSS)</name>
        <summary>Main OSS Repository</summary>
```

```
      <description>The largest and main repository from openSUSE for open
source software</description>
      <url>http://download.opensuse.org/repositories/openSUSE:11.3/standard/
</url>
    </repository>
```

Каждый пакет, который нужно установить, прописывается в отдельной секции YMP-файла: `<item>` (листинг 22.4).

Листинг 22.4. Секция <item> YMP-файла для установки пакета w32codec-all

```
<item>
       <name>w32codec-all</name>
       <summary>Win 32 Codecs</summary>
       <description>This packages contains the media player windows codec
dlls for several multimedia formats.</description>
    </item>
```

Понятно, что если нужно установить несколько пакетов, то и секций `<item>` будет несколько.

Пояснение

В листингах 22.3 и 22.4 приведены фрагменты файла codecs-gnome.ymp, благодаря которому в openSUSE устанавливается поддержка форматов мультимедиа.

Примечание

Приведенная здесь информация нужна лишь для общего развития — вам никогда не придется изменять YMP-файлы (хотя кто знает, что нас ждет в этой жизни?), а установка таких файлов производится автоматически, практически без вмешательства пользователя.

Теперь перейдем непосредственно к использованию менеджера пакета `zypper`. Формат вызова `zypper` следующий:

```
zypper <команда> [пакеты]
```

Основные команды `zypper` приведены в табл. 22.1.

Таблица 22.1. *Основные команды* `zypper`

Команда	Описание
`sl`	Выводит список используемых репозитариев
`sa URL имя`	Добавляет репозитарий (`URL` — адрес репозитария, а `имя` — имя, под которым он будет отображаться). Пример: `zypper sa http://ftp.uni-kl.de/pub/linux/suse/update/10.3 SUSE-Linux-10.3-Updates`

Таблица 22.1 *(окончание)*

Команда	Описание
`sd URL|имя`	Удаляет репозитарий. При удалении вы можете указать URL или имя репозитария
`install пакеты`	Устанавливает пакеты. Пример: `zypper install mc` Если нужно установить несколько пакетов, то имена пакетов разделяются пробелами
`search маска`	Ищет пакеты по маске. Маска — это часть имени (или полное имя) пакета. Пример: `zypper search mc*`
`list-updates`	Отображает доступные обновления
`update пакет`	Обновляет пакет. Если пакет не задан, обновляет всю систему
`info пакет`	Выводит информацию о пакете
`remove пакет`	Удаляет пакет

Глава 23

Управление пакетами в Slackware

23.1. Особенности Slackware

Slackware в плане установки пакетов — довольно специфический дистрибутив. Мне частенько приходилось слышать мифы о сложности установки и управления пакетами в Slackware. Но все эти мифы, как оказалось, от незнания. Просто пользователям, привыкшим к Red Hat-совместимым дистрибутивам, трудно привыкнуть к особенностям Slackware. Возможно, "коренным" пользователям Slackware трудно привыкнуть к обращению с RPM-пакетами… Тут утверждать не буду, потому что сам начинал свой путь линуксоида с дистрибутива Red Hat.

Но однажды я не выдержал и установил на свой компьютер Slackware. Цель была одна — разобраться с установкой пакетов. Неужели все так сложно? Как оказалось, ничего сложного нет, если разобраться в особенностях Slackware, не известных пользователям Red Hat.

Прежде чем приступить к рассмотрению системы управления пакетами, приведу ряд мифов, которые мне удалось разрушить:

- *в Slackware нет системы управления пакетами* — очевидно, данный миф сотворили пользователи, которые никогда не устанавливали Slackware, потому что такая система есть. Другое дело, что она не поддерживает RPM/DEB-пакеты. Пакеты Slackware выполнены в виде обычных TGZ-архивов. Но и формат пакетов RPM — это тоже слегка модифицированный архивный формат, просто его назвали иначе, а в Slackware используются обычные архивы. Хорошо это или плохо, решать вам. Но учитывая, что Slackware появился намного раньше, чем Red Hat с его системой RPM, использование архивов TGZ вполне закономерно;
- *в Slackware нет зависимостей пакетов* — это тоже миф, но в нем есть доля правды. Зависимости есть, но программы для установки пакетов их не обрабатывают — обработка зависимостей возложена на пользователя. Хорошо это или плохо? С одной стороны, есть вероятность недоустановить какой-то пакет или же удалить пакет, необходимый другим пакетам, что нарушит зависимости пакетов. Можно также установить пакет, который будет конфликтовать с уже установленными пакетами. Одним словом, при установке программного обеспечения нужно четко себе представлять, что вы делаете, а то очень легко превратить свою систему в мусорку, для наведения полного порядка в которой

поможет только переустановка системы. Если в дистрибутивах, основанных на RPM/DEB, можно положиться на менеджера пакетов, то в Slackware нужно рассчитывать только на себя, поэтому перед установкой пакета поможет прочтение соответствующей пакету документации. С другой стороны, пакеты в Slackware довольно объемные и содержат практически все необходимое для работы конкретного программного продукта. Например, чтобы установить PHP в Mandriva, вам понадобится 21 пакет, причем каждый из этих пакетов каким-то образом зависит от других пакетов группы. А вот для установки PHP в Slackware нужно установить всего один пакет, который включает все необходимое. Поэтому можно сказать, что разрешение зависимостей в Slackware совсем необязательно;

- *в Slackware отсутствует механизм обновления системы* — комментарии здесь примерно такие же, как и в предыдущем случае. Такой механизм есть, и его использовать не сложно, нужно только знать как;
- *в Slackware неудобно устанавливать программы, не входящие в состав дистрибутива*, — вот тут огромная доля правды. Можно даже сказать, что это не миф… С самой установкой ничего сложного нет, есть сложности с поиском необходимых пакетов. Но об этом мы поговорим чуть позже.

Вот теперь можно приступить к рассмотрению системы управления пакетами Slackware.

23.2. Управление пакетами

Для управления пакетами в Slackware используются четыре основные программы:

- `pkgtool` — псевдографический (использует текстовые меню) менеджер пакетов, позволяющий устанавливать, удалять и обновлять пакеты (рис. 23.1).

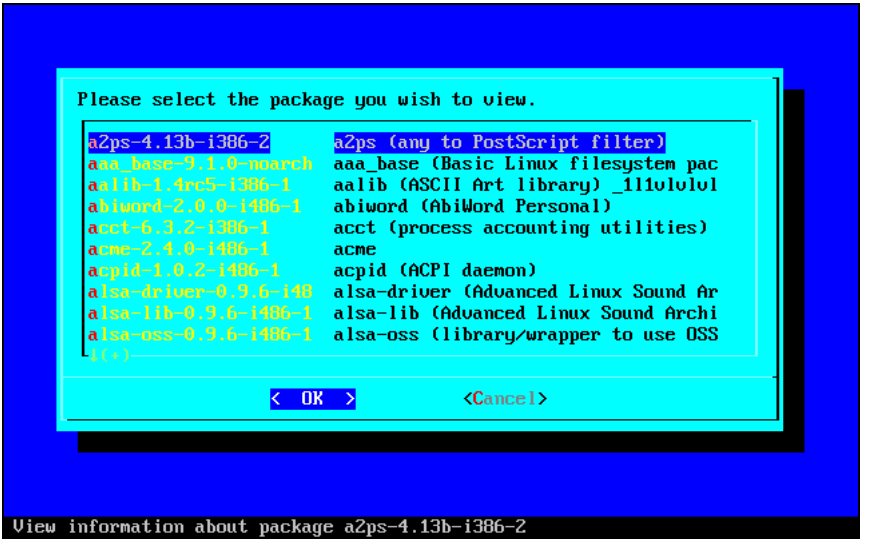


Рис. 23.1. Программа `pkgtool`

В его работе несложно разобраться, поэтому мы подробно его рассматривать не будем. А любителям графических конфигураторов наверняка понравится графическая версия этой программы — `XPKGTOOL` (рис. 23.2);

- `installpkg` — программа для установки пакетов;
- `removepkg` — программа удаления пакетов;
- `upgradepkg` — программа обновления пакетов.

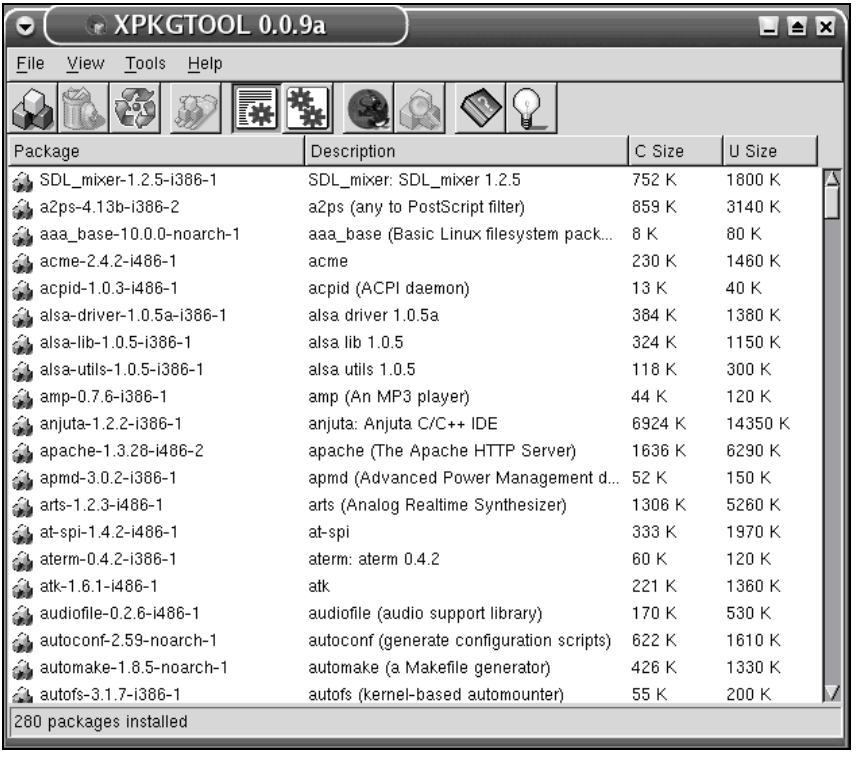


Рис. 23.2. Программа `XPKGTOOL`

23.2.1. Команда *installpkg*: установка пакетов

Перед рассмотрением программы `installpkg` определимся со структурой пакета. Как уже было отмечено, пакет с программным обеспечением в Slackware — это обычный TGZ-архив, предназначенный для распаковки в корневой каталог файловой системы. Вот пример структуры каталогов вымышленного пакета, содержащего всего одну программу — `program`:

```
./
usr/
usr/bin/
usr/bin/program
```

```
usr/man/
usr/man/man1
usr/man/man1/program.1.gz
install/
install/doinst.sh
```

Обратите внимание на каталог install — в нем находится сценарий `doinst.sh`, запускающийся после установки пакета.

Синтаксис команды для установки пакета:

```
# installpkg <опция> <имя пакета>
```

Вы можете задать одну из трех опций программы:

- `-m` — используется для сборки пакета (действие `makepkg`) в текущем каталоге;
- `-warn` — режим предупреждений: установка пакета не производится, однако выводится список планируемых действий. Если вы устанавливаете пакет на критически важной системе или просто не уверены в своих действиях, перед установкой пакета рекомендуется использовать режим предупреждений;
- `-r` — рекурсивно устанавливает все пакеты из текущего каталога и всех его подкаталогов.

Информация об установленных пакетах хранится в файле /var/log/packages. При установке пакетов вы можете указывать сразу несколько пакетов, а также использовать маски имен (типа `gnome*`).

Как уже было отмечено, при установке пакетов не проверяются зависимости пакетов, поэтому желательно первую установку производить в режиме `-warn`. Также `installpkg` не сообщит, если вы попытаетесь установить уже установленный пакет. Программа просто перезапишет старые файлы новыми версиями (из устанавливаемого пакета). Вы думаете, что это недостаток? Может, и так, зато легко производить обновление пакета — можно просто использовать программу `installpkg`, хотя для более безопасного обновления рекомендуется использовать программу `upgradepkg`.

Пример вызова программы:

```
# installpkg bash-2.04b-i386-2.tgz
```

23.2.2. Команда *removepkg*: удаление пакетов

Формат вызова программы `removepkg` такой же, как и в предыдущем случае:

```
# removepkg <опция> <имя пакета>
```

Опций у `removepkg` немного больше — четыре:

- `-copy` — копирует пакет в резервный каталог, но не удаляет его (см. опцию `preserve`);
- `-keep` — сохраняет временные файлы, которые программа создает при удалении пакета. Полезно при тестировании созданных вами пакетов (если вы разработчик/сборщик пакета);
- `-preserve` — удаляет пакет, но перед удалением копирует его в резервный каталог. Место на диске с этой опцией не сэкономишь, зато пакеты можно не удалять полностью из системы;

- `-warn` — режим предупреждения: не удаляет пакет, а просто показывает список действий, которые будут выполнены при удалении пакета.

  Пример вызова программы:

  ```
  # removepkg bash
  ```

23.2.3. Команда *upgradepkg*: обновление пакетов

Использовать программу обновления пакетов очень просто:

```
# upgradepkg <имя пакета>
```

Программа сначала устанавливает новую версию пакета, а затем — удаляет старую, дабы в системе не остались старые версии файлов.

23.3. Нет нужного пакета — вам поможет программа *rpm2tgz*

Иногда просто невозможно найти программу, распространяющуюся в пакете Slackware, — большинство пакетов распространяется в формате RPM. В этом случае можно попробовать использовать программу `rpm2tgz`, преобразующую пакет формата RPM в формат Slackware. При этом следует понимать, что данная программа преобразует лишь формат пакетов, она не занимается разрешением зависимостей и т. п., т. е. нет никакой гарантии, что после такого преобразования установленная программа будет работать.

> ***Совет***
>
> Вы думаете, что для вашей программы нет Slackware-пакета? А может, вы не там искали? Попробуйте посетить сайт **http://linuxpackages.net/** — там есть очень много Slackware-пакетов.

23.4. Программа *slackpkg*: установка пакетов из Интернета

Наверное, вы заметили, что программа `installpkg` занимается установкой пакетов из локального каталога. А что делать, если пакет находится в Интернете? Понятно, что его нужно скачать и установить программой `installpkg`, но если вы привыкли к программам вроде `yum`, Slackware вам может показаться несколько ущербным и малофункциональным дистрибутивом.

На помощь приходит программа `slackpkg`, позволяющая несколько автоматизировать установку пакетов из сетевых источников — в Slackware сетевые источники называются *зеркалами* (от англ. *mirrors*). Программа `slackpkg` может скачать и установить пакет, находящийся на одном из серверов-зеркал. Но эта программа не занимается разрешением зависимостей, а только несколько упрощает установку и обновление пакетов. Не нужно думать, что `slackpkg` — это замена `installpkg`, она всего лишь ее полезное дополнение, позволяющее немного облегчить установку пакетов.

Программа `slackpkg` находится в каталоге extra. После установки программы slackpkg нужно подготовить ее к работе. Первым делом откройте ее главный конфигурационный файл /etc/slackpkg/mirrors и раскомментируйте географически ближайшее к вам зеркало. Зеркало — это просто адрес FTP-сервера, содержащего Slackware-пакеты:

ftp://ftp.nluug.nl/pub/os/Linux/distr/slackware/slackware-12.0/

> ***ВНИМАНИЕ!***
>
> Помните, что `slackpkg` позволяет использовать только одно зеркало. Если вы раскомментируете несколько зеркал, будет использоваться первое раскомментированное зеркало.

После редактирования файла зеркал нужно подготовить программу для работы с GPG-ключами.

Для этого введите команды:

```
# mkdir ~/.gnupg
# gpg --keyserver pgp.mit.edu --search security@slackware.com
```

При выполнении второй команды на экран будет выведено следующее сообщение:

```
gpg: searching for "security@slackware.com" from HKP server pgp.mit.edu
Keys 1-2 of 2 for "security@slackware.com"
(1)     Slackware Linux Project <security@slackware.com>
          1024 bit DSA key 40102233, created 2003-02-25
(2)     Slackware Linux Project <security@slackware.com>
          1024 bit DSA key 40102233, created 2003-02-25
Enter number(s), N)ext, or Q)uit >
```

Как видите, вас просят выбрать номер GPG-ключа. Введите номер одного из доступных GPG-ключей (список ключей перед вами, обычно можно ввести `1`).

Теперь вам осталось ввести еще одну команду:

```
gpg --fingerprint security@slackware.com
```

Все, программа `slackpkg` готова к использованию.

Перед установкой пакетов не помешает обновить список пакетов активного зеркала. Для этого используется команда:

```
# slackpkg upgrade
```

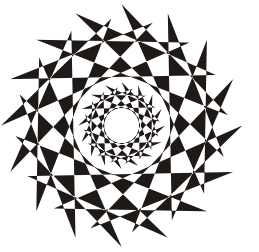
Чтобы иметь постоянно свежие сведения о пакетах, рекомендуется регулярно выполнять эту команду.

Для установки пакета введите команду:

```
# slackpkg install <пакет>
```

Для обновления пакета используется команда:

```
# slackpkg upgrade <пакет>
```



Глава 24

Управление пакетами в Mandriva

24.1. Команда *urpmi*: установка пакетов

Программа `urpmi` представляет собой систему управления пакетами, использующуюся в Mandriva. Как уже было отмечено в табл. 19.1, `urpmi` поддерживает зависимости пакетов. Конечно, обычным пользователям намного проще использовать программу `rpmdrake` для установки/удаления пакетов и управления источниками пакетов. Но `rpmdrake` — это всего лишь оболочка для системы `urpmi`, поэтому настоящий линуксоид должен знать, как работает `urpmi`.

Не нужно расценивать `urpmi` как замену `rpm` — система `urpmi` просто делает управление пакетами проще (хотя желающие могут использовать утилиту `rpm`, если сочтут ее более удобной).

> ***Примечание***
>
> Я, например, предпочитаю использовать `rpm` для локальной установки пакетов (когда пакет из какого-либо источника уже закачан на мой компьютер).

Для установки пакета служит команда:

```
# urpmi <имя пакета>
```

Так, для установки пакета `mc` (файловый менеджер Midnight Commander) следует ввести команду:

```
# urpmi mc
```

Программа просматривает список источников пакетов, хранящийся в файле /etc/urpmi/urpmi.conf. Если она находит пакет в одном из источников, то устанавливает его вместе со всеми необходимыми для его работы пакетами (при этом urpmi автоматически разрешает зависимости пакетов).

Существуют три вида репозитариев, поддерживаемых `urpmi`:

- хранилища на съемных носителях (removable) — репозитарии на компакт-дисках, DVD, ZIP-носителях, Flash-дисках и т. д.;
- локальные (local) — находятся в каталоге на жестком диске;
- удаленные (distant server) — пакеты находятся на удаленном FTP- или HTTP-сервере.

Просмотреть список источников пакетов можно с помощью команды:

```
# urpmq --list-media
```

Добавить источники пакетов можно с помощью команды:

```
# urpmi.addmedia <источник>
```

Получить список источников можно на сайте Easy Urpmi (**http://easyurpmi.zarb.org**). Зайдите на этот сайт, выберите версию вашего дистрибутива (2010), архитектуру и нажмите кнопку **Добавить официальные источники** (рис. 24.1). В появившемся окне нажмите кнопку **OK** (рис. 24.2). Браузер скачает файл источника пакетов, запустит средство добавления источника, которое запросит у вас пароль root, после этого нужно нажать **Да** (рис. 24.3) для установки источника пакетов. После установки официальных источников установите PLF-источники.

После добавления источников пакетов мой файл конфигурации /etc/urpmi/urpmi.cfg (Madnriva 2010, платформа i586) стал выглядеть так, как показано в листинге 24.1. Листинг я несколько сократил, потому что в противном случае он бы растянулся на четыре страницы.

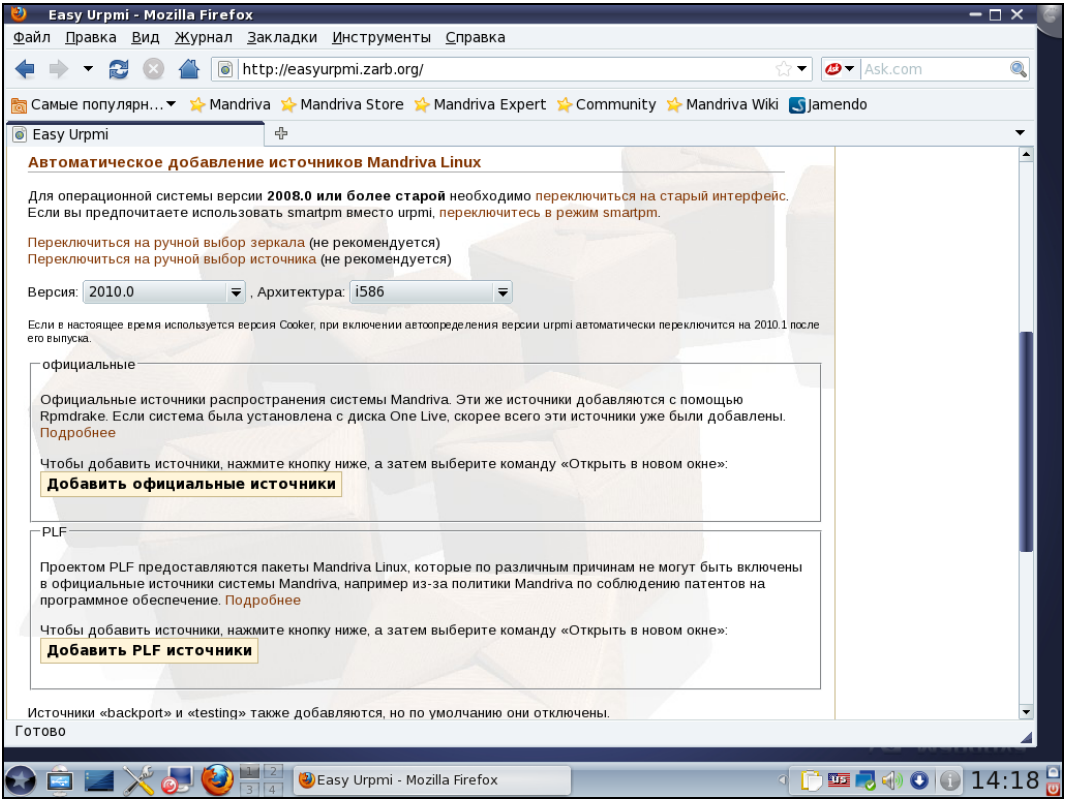


Рис. 24.1. Сайт Easy Urpmi

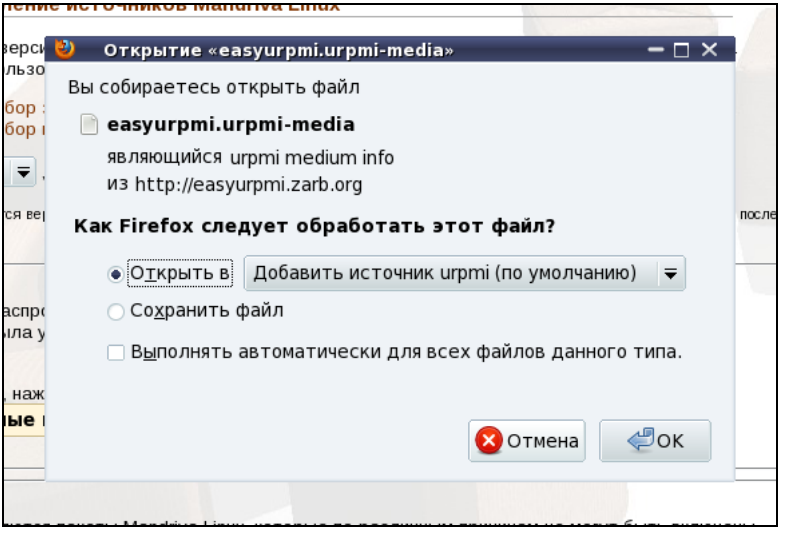


Рис. 24.2. Установка источника пакетов

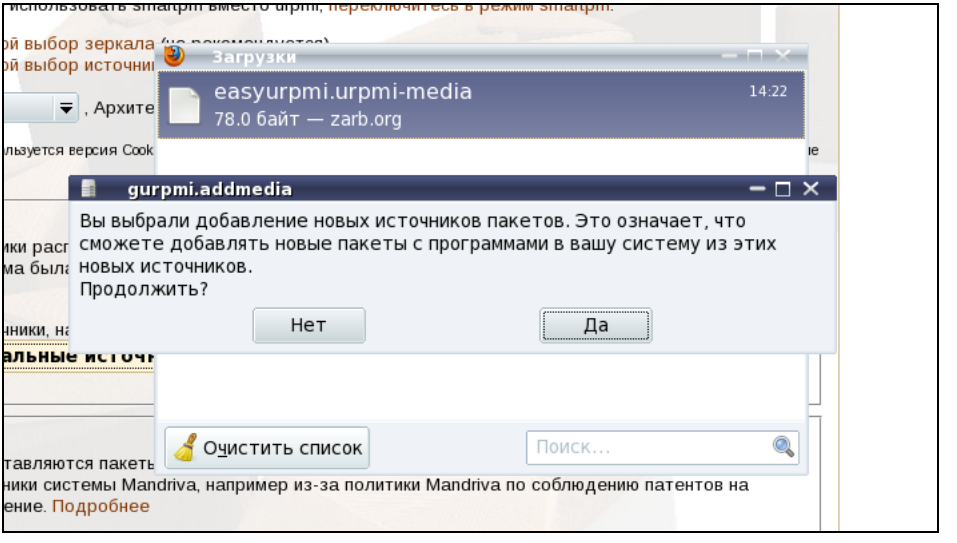


Рис. 24.3. Нажмите **Да** для установки источника пакетов

Листинг 24.1. Фрагмент файла /etc/urpmi/urpmi.cfg

```
{
}

Main\ media cdrom://i586/media/main {
```

```
key-ids: 70771ff3
}

Contrib\ media cdrom://i586/media/contrib {
key-ids: 78d019f5
}

Main {
key-ids: 70771ff3
mirrorlist: http://api.mandriva.com/mirrors/basic.2010.0.i586.list
with-dir: media/main/release
}

Main\ Updates {
key-ids: 22458a98
mirrorlist: http://api.mandriva.com/mirrors/basic.2010.0.i586.list
update
with-dir: media/main/updates
}

...
Contrib {
key-ids: 78d019f5
mirrorlist: http://api.mandriva.com/mirrors/basic.2010.0.i586.list
with-dir: media/contrib/release
}

...

Non-free {
key-ids: 70771ff3
mirrorlist: http://api.mandriva.com/mirrors/basic.2010.0.i586.list
with-dir: media/non-free/release
}

...

debug_non-free_release {
ignore
key-ids: 70771ff3
mirrorlist: http://api.mandriva.com/mirrors/basic.2010.0.i586.list
with-dir: media/debug_non-free/release
}

...
```

```
PLF\ Free {
key-ids: caba22ae
mirrorlist: http://plf.zarb.org/mirrors/2010.0.i586.list
update
with-dir: media/../../../../2010.0/free/release/binary/i586
}

PLF\ Free\ debug {
ignore
key-ids: caba22ae
mirrorlist: http://plf.zarb.org/mirrors/2010.0.i586.list
with-dir: media/../../../../2010.0/free/release/debug/i586
}

...
```

Для обновления репозитария (списка пакетов) используется команда:

```
# urpmi.update <имя источника>
```

Удалить источник пакетов можно или путем удаления информации о нем из файла urpmi.cfg, или с помощью команды:

```
# urpmi.removemedia <имя источника>
```

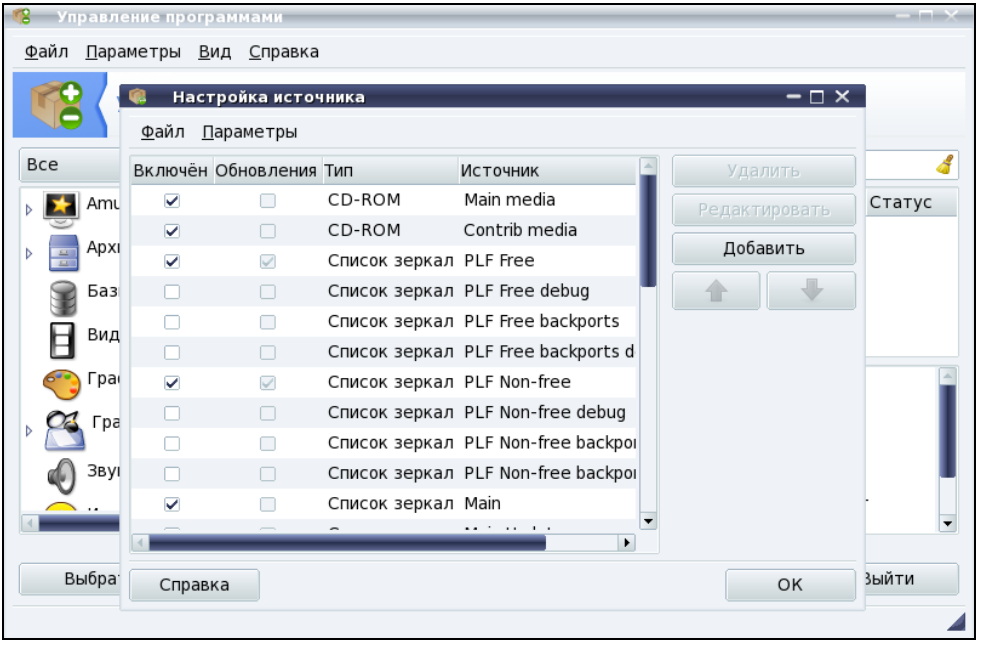


Рис. 24.4. Менеджер источников программ (после установки дополнительных источников)

Для обновления всего списка пакетов используется команда:

```
# urpmi.update -a
```

Если вам от редактирования конфигурационных файлов вручную становится не по себе, вы можете использовать один из графических менеджеров управления источниками пакетов — тот, который вам больше понравится. Для этого выполните команду меню **Параметры | Менеджер источников**. На рис. 24.4 изображен **Менеджер источников** после установки дополнительных источников пакетов.

Для обновления всей системы, т. е. получения списка новых версий пакетов, используется команда:

```
# urpmi --auto-select
```

24.2. Команда *urpme*: удаление пакетов

Для удаления пакета нужно ввести команду:

```
# urpme <пакет>
```

Если пакет нужен для работы других пакетов, то программа спросит у вас, хотите ли вы удалить и эти пакеты, иначе придется отказаться от удаления выбранного пакета.

24.3. Поиск пакета. Получение информации о пакете

Найти пакеты, содержащие в названии определенную строку, можно с помощью команды:

```
# urpmq <строка>
```

Команда `urpmf` позволяет получить различную информацию о пакете, например:

- `urpmf <файл>` — выводит пакеты, содержащие указанный файл;
- `urpmf --group <группа>` — выводит пакеты, входящие в указанную группу;
- `urpmf --size <пакет>` — выводит размер указанного пакета;
- `urpmf --summary <пакет>` — выводит общую информацию о пакете.

Заключение

Искреннее надеюсь, что данная книга вам понравилась и помогла настроить операционную систему Linux и познакомиться с ее внутренним миром. Упор в книге был сделан именно на практическое применение операционной системы. Хотя без теории далеко не уйдешь, но когда ее много, тоже не хорошо. Вспомнилось одно выражение: теория — это когда вы все знаете, но ничего не работает, а практика — это когда вы ничего не понимаете, но зато все работает. В книге теория и практика сбалансированы так, чтобы вы свои знания могли применить на практике.

Связаться с автором книги, т. е. со мной, всегда можно на форуме сайта **www.dkws.org.ua**, а оставить отзыв о книге можно на сайте издательства — **www.bhv.ru**.

Приложение

Создание дистрибутива

П1.1. Зачем нужно создавать еще один дистрибутив

В настоящее время существует огромное количество дистрибутивов Linux. Я даже не могу сказать, сколько именно. Каждый день появляются новые дистрибутивы. Все это благодаря популярности Linux и простоте создания собственного дистрибутива. Чтобы создать свой дистрибутив, даже не требуются навыки программирования (хотя и приветствуются), что и будет показано в этом *приложении*.

Так почему же создаются новые дистрибутивы? Ведь не всеми разработчиками движет желание похвастаться перед друзьями и коллегами! Все гораздо проще (или сложнее). Ни один из существующих дистрибутивов не идеален. В каждом есть недоработки, причем иногда такие существенные, что не хочется даже использовать тот или иной дистрибутив.

Основная причина, которая движет разработчиками "дочерних" дистрибутивов, — это желание усовершенствовать уже существующий дистрибутив. Да, именно уже существующий. Создание дистрибутива с нуля — занятие неблагодарное. Небольшой команде из одного-двух человек это не под силу. Вам много чего придется разработать, а на все это нужно время. Даже если вы создадите собственный дистрибутив, то на это понадобится много времени, а чтобы создать дистрибутив в приемлемые сроки, нужна команда побольше. Одним энтузиазмом сыт не будешь, поэтому вам придется финансировать свою команду. А это деньги. Если вы готовы инвестировать средства в разработку еще одного дистрибутива, это хорошо — скоро появится еще один достойный дистрибутив.

Если же желание вкладывать средства отсутствует, тогда можно создать дистрибутив на базе одного из имеющихся дистрибутивов. Желательно выбирать дистрибутив, для которого разработаны средства автоматизации этого процесса. Такие средства разработаны для Debian, Ubuntu, OpenSUSE и некоторых других дистрибутивов.

Учитывая, что вам нужно будет только доработать уже готовый дистрибутив, могу заверить, что на создание дистрибутива будет потрачено гораздо меньше времени. По сути, все зависит только от ваших доработок. Можно только доустановить программы и сменить обои рабочего стола. Потом запаковать все в ISO-образ. В конечном итоге на все про все уйдет несколько часов.

В этом разделе мы будем рассматривать процесс создания дистрибутива на базе Ubuntu. Этот дистрибутив в свое время я выбрал в качестве базового для создания собственного дистрибутива Denix (**http://denix.dkws.org.ua**). Все просто: я знаю, о чем пишу — опыт создания дистрибутива на базе Ubuntu у меня есть. Может, со временем попытаюсь "слепить" дистрибутив на базе OpenSUSE, тогда в этой книге появится еще одно приложение.

П1.2. Инструменты для создания дистрибутива

Существуют два способа создания дистрибутива. Первый заключается в использовании специального средства для создания дистрибутива. Для Ubuntu — это UCK (Ubuntu Customization Kit) и Reconstructor (**https://www.reconstructor.org/wiki/reconstructor/**). Преимущество этого способа в простоте. Вы устанавливаете и запускаете UCK (или Reconstructor — принцип тот же), дальше в графическом окне изменяете параметры дистрибутива, например выбираете другую графическую тему, добавляете или удаляете программы. А вот недостаток более существенный, чем преимущество. Вы ограничены лишь возможностями UCK. Если UCK может изменить тот или иной параметр, вы его измените. А если нет, то вы уже ничего не сделаете.

Да, в UCK есть командная строка. Вы можете подмонтировать файловую систему вашего будущего дистрибутива и работать с ней в терминале UCK. Данную возможность я использовал для установки/удаления пакетов (мне удобнее использовать командную строку, чем интерфейс UCK) и для редактирования некоторых системных файлов.

Но есть такие ситуации, когда одной командной строки мало. Например, вы создаете дистрибутив с улучшенной поддержкой видеокарт ATI. Как вы протестируете, корректно ли установлены драйверы? А никак. Вам придется собрать дистрибутив, записать его на болванку, запустить на другом компьютере и посмотреть. Если что-то не так, то опять запустить UCK, отредактировать системные файлы и повторить процесс сборки заново (а на все это нужно время!). Согласитесь, не очень удобно.

Но можно пойти иным путем, в чем и заключается второй способ. Вы устанавливаете базовый дистрибутив на свой компьютер и настраиваете его так, как вам хочется. При этом вы не используете UCK, а просто настраиваете дистрибутив, устанавливаете программы и драйверы средствами самого дистрибутива. Когда все будет готово, вы установите программу для резервного копирования (она называется `remastersys`), которая потом создаст готовый LiveCD. Этот способ чуть сложнее, потому что всю настройку нужно будет сделать не средствами UCK, а вручную (а тут нужны некоторые знания и навыки), но зато вы ничем не ограничены.

Если же вам по душе простой способ, рекомендую прочитать мою статью "Конструктор для тукса: пошаговое руководство по созданию своего дистрибутива на базе Ubuntu", в которой рассматривается UCK:

http://www.xakep.ru/post/48566/default.asp

Далее мы будем рассматривать второй способ создания дистрибутива. Кстати, у второго способа есть еще одно немаловажное преимущество. Для использования UCK нужно примерно 6 Гбайт дополнительного свободного места на диске. А для использования `remastersys` — только место, необходимое для создаваемого образа (обычно это 1 Гбайт, может, чуть больше).

П1.3. Этапы создания дистрибутива

Первым делом нужно определиться, что такое дистрибутив. Дистрибутив — это набор программ (пакетов), оснащенный программой установки. Выходит, для создания собственного дистрибутива нужно определить набор программ, которые будут входить в состав дистрибутива, и разработать установщик. Поскольку мы будем использовать инсталлятор Ubuntu, то разрабатывать его нам не придется.

Вот основные этапы создания дистрибутива (понятно, что на базе имеющегося дистрибутива, при создании дистрибутива с нуля придется много чего еще сделать).

1. Определить состав программного обеспечения — возможно, вам захочется установить дополнительные программы или удалить уже установленные, чтобы сделать дистрибутив компактнее.
2. Добавить в дистрибутив новые функции — возможно, все новые функции как раз и будут заключаться в дополнительных программах, а может, вы будете редактировать файлы конфигурации дистрибутива и даже установите новое ядро.
3. Изменить внешний вид дистрибутива — это уже по желанию, сами понимаете, менять графическую тему совсем не обязательно, но если вы делаете дистрибутив для себя любимого, наверняка захочется установить предпочитаемую тему.
4. Создать LiveCD — чтобы можно было распространять ваш дистрибутив, нужно создать LiveCD, который помимо всего прочего будет включать инсталлятор дистрибутива. В этом нам поможет программа `remastersys`.

П1.4. Процесс создания дистрибутива

Первым делом нужно установить Ubuntu на свой компьютер. Процесс установки Ubuntu описывать не буду — будем считать, что Ubuntu уже установлена (уверен, что с процессом установки вы и так справитесь).

После установки дистрибутива нужно изменить состав программного обеспечения. В *главе 20* была подробно описана программа `apt-get`, с помощью которой можно установить и удалить программное обеспечение.

Какое программное обеспечение доустановить? Лично я привык к программе GIMP, но из состава Ubuntu 10 ее удалили. Установить ее можно командой:

```
sudo apt-get install gimp
```

Программное обеспечение — это дело ваших личных предпочтений. Поэтому ничего советовать не буду. Возможно, вам вообще чаще придется пользоваться командой `apt-get remove`, потому что вам захочется создать компактный дистрибутив.

Бич Ubuntu — отсутствие кодеков. Давайте добавим в наш дистрибутив поддержку форматов мультимедиа. Это и будет наша новая возможность. Все необходимое программное обеспечение для воспроизведения популярных форматов мультимедиа находится в репозитарии Medibuntu.

Все необходимые инструкции по подключению источника пакетов Medibuntu есть на его домашней странице: **http://www.medibuntu.org/**. В частности, приводится универсальная команда подключения репозитария:

```
sudo wget --output-document=/etc/apt/sources.list.d/medibuntu.list
http://www.medibuntu.org/sources.list.d/$(lsb_release -cs).list && sudo
apt-get --quiet update && sudo apt-get --yes --quiet --allow-
unauthenticated install medibuntu-keyring && sudo apt-get --quiet update
```

Команда слишком запутанная и на самом деле состоит из четырех команд. Вводить ее вручную не нужно. Просто зайдите на страницу **https://help.ubuntu.com/community/Medibuntu**, скопируйте оттуда эту команду и вставьте в терминал. Как только вы нажмете клавишу <Enter>, начнется процесс подключения репозитария. Через некоторое время (все зависит от скорости вашего интернет-соединения и производительности процессора) репозитарий будет подключен.

Не знаю, как вы, а я не люблю вводить команды, не понимая, что они делают. Поэтому разберем нашу универсальную команду. Первая команда выглядит так:

```
sudo wget --output-document=/etc/apt/sources.list.d/medibuntu.list
http://www.medibuntu.org/sources.list.d/$(lsb_release -cs).list
```

Данная команда получает файл **http://www.medibuntu.org/sources.list.d/$(lsb_release -cs).list**. При этом производится подстановка вывода команды `sb_release -cs`, которая выводит название вашей версии Ubuntu. Например, если у вас Ubuntu 10.04, то будет получен файл **http://www.medibuntu.org/sources.list.d/lucid.list**, а если до сих пор Ubuntu 9.10, то система получит файл **http://www.medibuntu.org/sources.list.d/karmic.list**. Полученный файл будет сохранен как /etc/apt/sources.list.d/medibuntu.list.

Далее выполняются три команды:

```
sudo apt-get --quiet update
sudo apt-get --yes --quiet --allow-unauthenticated install medibuntu-keyring
sudo apt-get --quiet update
```

Первая обновляет список пакетов apt-get, далее устанавливается пакет medibuntu-keyring, содержащий GPG-ключи репозитария Medibuntu. Опция `--allow-unauthenticated` разрешает установку неаутентифицированного пакета. Аутентифицировать пакет мы не можем, поскольку пока не установили ключи. После этого еще раз обновляет список пакетов.

Все, репозитарий Medibuntu готов к работе.

Первым делом нужно установить основные кодеки, поэтому введите одну из двух команд (в зависимости от архитектуры вашей системы):

```
sudo apt-get install w32codecs   (если у вас 32-битная система)
sudo apt-get install w64codecs   (для 64-битной системы)
```

Обратите внимание: архитектура кодеков должна совпадать именно с архитектурой системы, а не процессора. Ведь на 64-битный процессор вы можете смело установить 32-битную Ubuntu и она будет прекрасно работать. Но все программы, которые вы устанавливаете, должны быть также 32-битными!

Кроме пакета w??codecs вам нужно еще установить следующие пакеты:

```
sudo apt-get install non-free-codecs
sudo apt-get install libdvdcss2
```

Первый пакет — это дополнительные кодеки. Пакет довольно большой, и после его установки дисковое пространство уменьшится примерно на 75 Мбайт. Когда устанавливаешь Ubuntu на нетбук с ограниченным накопителем, то приходится учитывать каждый мегабайт, иначе для собственных файлов места не останется.

Второй пакет обеспечивает воспроизведение лицензионных DVD, которые обычно зашифровываются с помощью системы CSS (Content Scramble System).

Практически все. После установки описанных пакетов вы сможете воспроизводить звук и видео через стандартные проигрыватели Ubuntu, а окно файлового менеджера Ubuntu будет отображать миниатюры видеофайлов, как в Windows. Но я бы еще установил культовый проигрыватель MPlayer — это самый известный проигрыватель для Linux. Кроме самого MPlayer вам понадобится графическая оболочка для него. Мне больше всего нравится SMPlayer, отлично подходящая для воспроизведения как фильмов, так и музыки. Однако программа SMPlayer разработана с использованием библиотек KDE, поэтому при установке SMPlayer будут установлены необходимые библиотеки, а это дополнительное дисковое пространство. Поэтому если место на диске ограничено, то можно вместо SMPlayer использовать менее эффектную и менее функциональную оболочку `gnome-mplayer`.

Описывать процесс изменения графической темы не стану — думаю, вы сами в состоянии это сделать. Проблемы могут возникнуть разве что при изменении графической темы GDM (GNOME Display Manager). Если не сможете ее изменить, по следующему адресу вы найдете подробные инструкции:

http://www.dkws.org.ua/phpbb2/viewtopic.php?t=4549.

Сами же графические темы всегда можно скачать по адресу **www.gnome-look.org**.

Осталось самое малое — установить `remastersys` и создать LiveCD. Откройте файл sources.list:

```
sudo nano /etc/apt/sources.list
```

Добавьте в него следующую строку:

```
deb http://www.remastersys.klikit-linux.com/repository remastersys/
```

Сохраните файл и введите две команды:

```
sudo apt-get update
sudo apt-get install remastersys
```

Формат вызова `remastersys` следующий:

```
sudo remastersys backup|clean|dist [cdfs|iso] [filename.iso]
```

Пройдемся по опциям:

- `backup` — создание резервной копии дистрибутива, включая пользовательские данные (каталог /home);
- `clean` — удаляет временные файлы, которые образуются в процессе создания дистрибутива. Обязательно введите эту команду после создания дистрибутива (для экономии места), но только после того, как скопируете образ дистрибутива в другой каталог, иначе он тоже будет удален;
- `dist` — создание дистрибутивного образа, то же самое, что и `backup`, но без копирования пользовательских данных из каталога /home;
- `cdfs` — создание файла с файловой системой, без создания ISO-образа (подходит, если вы хотите создать ISO-образ другой программой);
- `iso` — используется по умолчанию, создает ISO-образ дистрибутива;
- `[filename.iso]` — последний параметр задает имя ISO-образа, файл помещается в каталог /home/remastersys.

Мне больше нравится опция `backup`, поскольку при создании образа сохраняются и настройки пользователя, в том числе меню, графическая тема, фон рабочего стола. Но только убедитесь, чтобы в вашем домашнем каталоге не было ничего лишнего (что может увеличить размер образа, например музыки и видео).

П1.5. Развитие дистрибутива

Технически, вы уже создали дистрибутив. Но что делать дальше? Можно на этом остановиться — мол, результат достигнут. А можно купить хостинг, выложить дистрибутив в Сети, создать репозитарии для собственных пакетов, привлечь разработчиков к созданию программ именно для вашего дистрибутива, нанять дизайнера для разработки уникальной графической темы и логотипа. Одним словом, можно развивать свой дистрибутив.

Но на все это нужны средства. А здесь решайте сами, хотите вы или нет вкладывать в дистрибутив. На первом этапе понадобится совсем немного — хостинг на несколько гигабайтов обойдется примерно в 10 долларов в месяц (в год, соответственно, 120). А вот привлечение программистов и дизайнеров стоит больших денег. Если вы планируете распространять дистрибутив на коммерческой основе (продавать диски, обеспечить техническую поддержку компаниям, купившим дистрибутив), у него должен быть товарный вид — без него никто не заинтересуется вашей разработкой. Да и не забывайте, что в дистрибутиве должна быть "изюминка" — то, что отличает его от других дистрибутивов (например, уникальное программное обеспечение). Если ваш дистрибутив будет копией Ubuntu с другой графической темой и кодеками, вряд ли вы сможете заработать на нем много денег. Такой дистрибутив не сможет перейти из разряда хобби в средство обогащения. В любом случае желаю вам удачи в этом нелегком деле и буду рад видеть вас на своем форуме **www.dkws.org.ua**, где мы сможем поделиться опытом *дистрибутивостроения*.

П1.6. Быстрое создание LiveUSB

Иногда нужно создать загрузочную флешку LiveUSB. Основная причина — необходимость загрузки Linux на нетбуке, где обычно нет приводов CD/DVD. Установить Linux на такой компьютер можно, только загрузившись с флешки.

Создать загрузочную флешку очень просто. Сейчас, ради простоты, мы немного отступим от концепции книги — обратимся к *графической* программе. В этом *приложении* мы рассматривали создание дистрибутива на базе Ubuntu. В Ubuntu есть программа для создания LiveUSB. Загрузитесь с LiveCD Ubuntu (его можно скачать с **www.ubuntu.com**). Понятно, что загрузку нужно производить на компьютере с CD-приводом. Выберите команду меню **System | Administration | Startup Disk Creator**. В появившемся окне нужно выбрать образ Ubuntu (обычно уже выбран, поскольку вы загрузились с LiveCD) и флешку, которая станет загрузочной. Если флешка не пустая, нажмите кнопку **Erase Disk**, а затем кнопку **Make Startup Disk** (рис. П1.1).

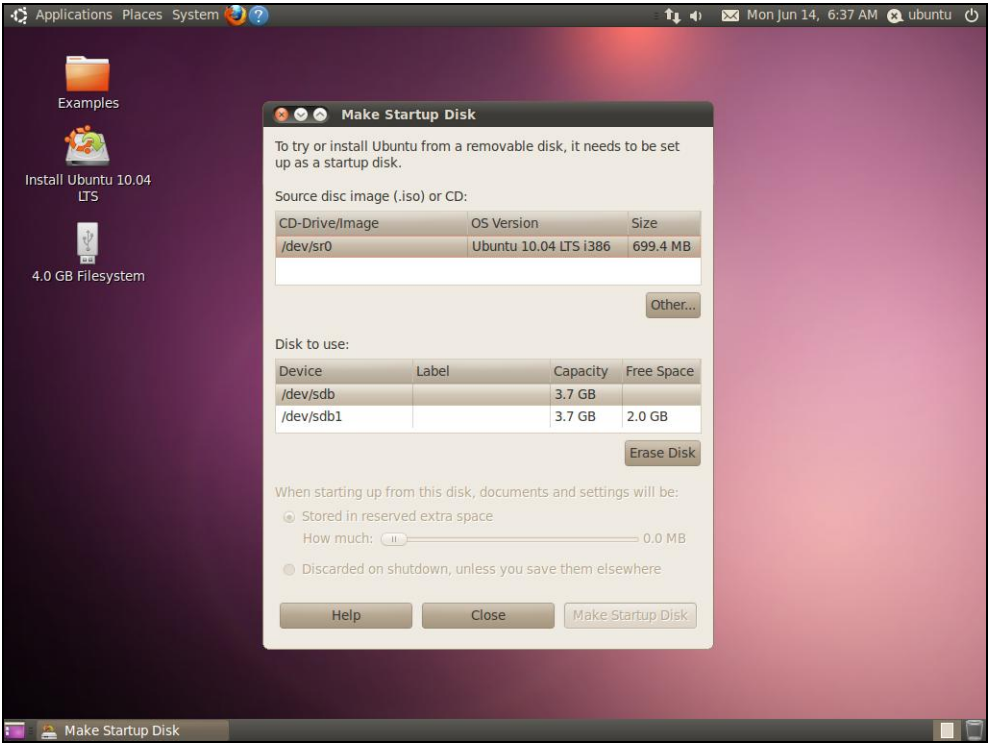


Рис. П1.1. Создание LiveUSB

Вот и весь процесс создания диска. Немного подождите, и загрузочная флешка будет готова. Вам останется только загрузиться с нее.

Предметный указатель

3

3DES 107

A

Almquist shell 17
anacron 104
Apache2 219
APIC 156
APM 169
ash 17
AT&T 14
awk 210

B

bash 13, 15, 184
binutils 180
BIOS 130
Blowfish 150
BlowFish 107
BogoMIPS 157

C

CRC 19
crontab 102
csh 14

D

Debian 147
DEB-пакеты 231
Denix 239, 270
DES 150
DOS 7, 27

E, F

ext4 49
Fedora 9, 146, 154
FreeBSD 14, 195

G

gawk 210
GDM 147
GID 150
GNOME Display Manager 273
GPRS 84
GRUB 163
GRUB2 128

I, J

IDEA 107
initrd 131
JFS 97

K

KDE Display Manager 146
ks 15
Kubuntu 143

L

LILO 163
LiveCD 53, 270
LiveUSB 275
login shell 197

M

Mandriva 43, 146, 260
MBR 130
MD5 150
Medibuntu 243
Midnight Commander 260
MSS 83
MTU 82
MySQL 218
MySQL Administrator 223

N

ndiswrapper 108
NX-защита 155

P

pdksh 15
PHP 218
PID 56
PnP 159
POSIX 14
ProFTPD 225

R

Reconstructor 270
remastersys 270
Remote Administrator 140
rpm 235
RPM-пакеты 231, 233
RSA 107

S

SELinux 158
SGID 38
sh 14
Slackware 7, 137, 254
SMP 155
SSH 106
SUID 38

T

tcsh 17, 195
Telnet 106
TENEX 17, 195

U

Ubuntu 45, 143, 195, 218
Ubuntu Customization Kit 270
udev 40
UFS 97
UID 150
UNIX 7, 73
UUID 40

X, Z

XFS 97
zsh 16

А, Б

Автодополнение 11
Браузер:
  elinks 86
  links 86
  lynx 86

В, Д

Выражение 199
Директива:
  AllowRootLogin 146
Диск:
  USB-диск 48
  виртуальный 130
  гибкий 40

Ж

журнал 49
журналы 24

З

Загрузчик:
  ASPLoader 113
  GRUB 113
  GRUB2 113
  LILO 113, 130

К

Каталог:
  /etc/cron.daily 103
  /etc/cron.hourly 103
  /etc/cron.weekly 103
  /etc/event.d 136
  /etc/rc.d 133
  /etc/rc.d/init.d 133
  /etc/skel 150, 197
  /etc/zypp/repos.d 250
  /var/cache/apt/archives 240
  домашний 33
  признак каталога 36
  родительский 33
  текущий 33
Команда:
  /sbin/grub-install 120
  /sbin/init 132
  adduser 147
  alias 197
  alien 238, 239
  apt 234
  apt-get 239, 240
(окончание рубрики см. на стр. 276)

Команда *(окончание)*:
aptitude 239
arch 18
at 102
atq 102
atrm 102
banner 19
builtins 195
cat 30
cd 33
cdrecord 63
chmod 36
chmod +x 187
chown 37
chroot 53
chsh 19
cksum 19
clear 11, 19
cmp 66
column 66
comm 67
configure 232
convert 124
cp 31
date 20, 105
dd 62
df 105
diff 67
diff3 68
dmesg 108, 154
dpkg 234, 236, 239
dvd+rw-format 64
echo 21
edquota 152
egrep 69
env 21
exit 21, 143
expand 70
fdisk 41, 94
find 54
fmt 70
fold 70
free 105
fsck 44, 52
ftp 87, 225
ftpwho 102
gcc 177
gdb 181
gksu 144
gpart 53
gparted 101
gprof 181
gproftpd 225
grep 71
groupadd 151
grub-mkconfig 120
grub-mkpasswd-pbkdf2 129
gzip 124
halt 10
hdparm 53
head 71
ifconfig 60, 78, 89
info 22
insmod 79
iwconfig 108
iwlist 108
kdesu 143
kill 56
killall 57
ld 180
less 31, 71
ln 35
locate 31, 55
logout 10, 11
look 71
ls 33
lsmod 108
lspci, lsusb, lshw 108
make 180, 232
man 22
md5sum 106
mkdir 33
mkfs 52
mkisofs 64
more 71
mv 31
nice 60
nm 180
ntsysv 133
objcopy 181
objdump 181
parted 97
passwd 147
ping 92
poweroff 10
pppconfig 84
pppoeconf 80
printenv 22
ps 56
quotacheck 152
reboot 10
renice 60
repquota 153
reset 22
rm 31, 33
rmdir 33

route 79, 108
rpm 234
service 133
setenv 197
shutdown 10
size 181
sleep 22
sort 72
split 72
ssh 107
startx 7, 23
strings 181
strip 181
su 142
sudo 142
swapon 138
system-config-packages 235
tac 31
tail 71, 89
tee 23
top 58
touch 30
tracepath 93
traceroute 93
true 23
umount 39
uname 108
unexpand 73
update-grub 120
uptime 101
userdel 149
usermod 148
users 101
wc 77
wget 88
which 31, 55
who 140
yes 23
yum 234, 245
Команда who 101
Консоль 7, 9
Конфигуратор:
drakboot 133
drakxservices 135
pppoeconf 91
rpmdrake 234
system-config-services 133

М

Массивы 189
менеджер пакетов 251

О

Оператор 199
case 191
if 190

П

Пакет 231
DEB 236
kernel-source 166
PRM 238
зависимости 232
конфликты 233
Переменные 187
окружения 188
специальные 188
Переменные окружения 198
Планировщик
anacron 104
atd 102
crond 103
Программа
/usr/sbin/grub-mkconfig 116
aptitude 244
badblocks 108
CloneCD 65
hdparm 109
installpkg 256
ISOpen 62
memtest 109
pkgtool 255
removepkg 256
rpm2tgz 258
slackpkg 258
UltraISO 62
upgradepkg 256
urpmi 260
vwdial 84
xpkgtool 256
zypper 252

Р

Редактор:
joe 76
mcedit 76
nano 76
pico 76
vi 73
Репозитарий 233

С

Сеть:
отказ работы 89

Система инициализации:
  init 131
  upstart 131, 136
Событие:
  network-interface-added 137
  network-interface-up 137
Соединение
  DSL 80
Сценарий 14

Т

Терминал 11
Технология RPM 233
Точка монтирования 45

Ф

Файл
  .{ICE,X}authority 143
  .bash_history 184
  .bash_profile 11
  /boot/boot.b 130
  /boot/grub/grub.cfg 116
  /boot/grub/grub.conf 114
  /boot/grub/menu.lst 114
  /boot/map 130
  /etc/apt/sources.list 240
  /etc/boot/grub.conf 163
  /etc/crontab 103
  /etc/csh.cshrc 196
  /etc/csh.login 196
  /etc/csh.logout 196
  /etc/default/grub 118
  /etc/fstab 43, 151
  /etc/group 151
  /etc/init.d/rc 137
  /etc/inittab 131
  /etc/kde/kdm/kdmrc 146
  /etc/pam.d/gdm-password 146
  /etc/passwd 149
  /etc/ppp/peers/dsl-provider 81
  /etc/profile 184
  /etc/rc.d/rc.S 138
  /etc/resolv.conf 86, 93
  /etc/shadow 150
  /etc/shells 13
  /etc/sudoers 142
  /etc/urpmi/urpmi.conf 260
  /etc/wvdial.conf 84, 86
  /etc/yum.conf 247
  /etc/zypp/zypp.conf 251
  /var/log/messages 86
  ~/.bash_logout 184
  ~/.bash_profile 184
  ~/.bashrc 184
  ~/.history 197
  apache2.conf 221
  aquota.user 152
  fstab 49
  права доступа 35
  устройства 39
Файловая система
  ext2 24
  ext3 24
  ext4 49
  журналируемая 24, 49
Файловая система JFS 25
Файловая система ReiserFS 25
Файловая система XFS 25
Файловые системы 24, 25

Ц

Цикл:
  for 190
  while 190

Я

Ядро 154
  конфигуратор 167
  модуль ядра 169