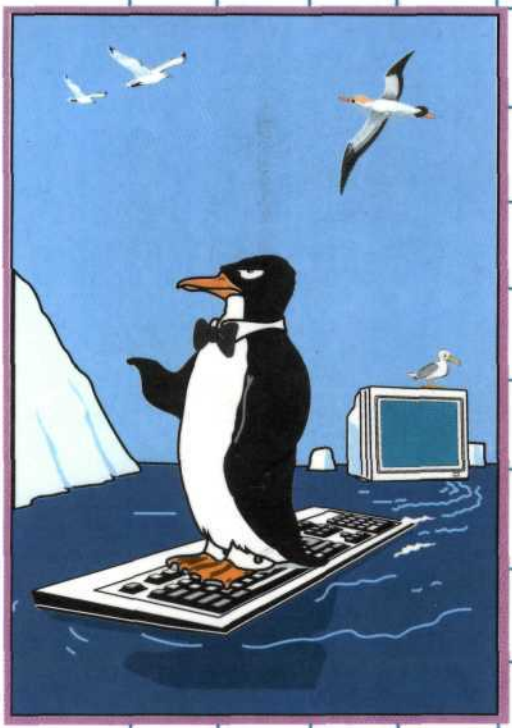
bhv®

Виктор Костромин

Linux для пользователя

САМОУЧИТЕЛЬ



Выбор дистрибутива

Инсталляция системы и программных пакетов

Файловая система Linux

Офисные Linux-приложения

Операционная система, конкурирующая с Windows

Виктор Костромин

САМОУЧИТЕЛЬ Linux для пользователя

Санкт-Петербург
«БХВ-Петербург»
2003

УДК 681.3.06
ББК 32.973.26-018.2
К72В

Костромин В. А.

К72В Самоучитель Linux для пользователя. — СПб.: БХВ-Петербург, 2003. - 672 с.: ил.

ISBN 5-94157-183-6

Книга посвящена использованию операционной системы Linux. Приводятся сведения об истории создания Linux и существующих дистрибутивах. Обсуждается инсталляция ОС Linux и ее настройка, описываются файловая система, графический интерфейс и интерфейс командной строки, подключение и настройка аппаратных средств, установка и обновление программных пакетов, работа в локальной сети и Интернете. Рассматриваются программы работы с текстом и операционные оболочки, предназначенные для манипулирования файлами. Приводятся ссылки на ресурсы Интернета.

Для опытных пользователей
и начинающих сетевых администраторов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Анна Кузьмина*
Редактор	*Андрей Майков*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.02.03.
Формат 70×100[1]/16. Печать офсетная. Усл. печ. л, 54,18.
Доп. тираж 5000 экз. Заказ № 751
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар №77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-183-6

Содержание

Предисловие

Для кого эта книга

Эта книга предназначена для активных пользователей персональных компьютеров. Поясню, что я имею в виду. На мой взгляд всех людей, которые каким-то образом связаны с применением компьютеров, можно разделить на программистов, администраторов и пользователей. Программисты создают программы, и о них мы здесь не говорим, как и о тех, кто изготавливает или собирает сами компьютеры. Администраторы (которых иногда называют системными программистами) вообще-то не пишут программ, они только устанавливают и настраивают программное обеспечение (разработанное другими), для того, чтобы пользователи могли работать за компьютерами. Администраторам приходится иногда создавать короткие программки, но в основном они пользуются теми средствами настройки и конфигурирования, которые заложены в программные средства их разработчиками. И, наконец, пользователи вообще не обязаны уметь настраивать программные средства, как и собирать компьютеры, они обязаны только уметь использовать свой компьютер для решения конкретных задач. В принципе пользователь может вообще не знать о том, что работой компьютера управляет операционная система. Ему достаточно уметь запускать одно конкретное приложение, которое он освоил, например, текстовый редактор (условно назовем такого пользователя "пассивным").

Но среди пользователей есть и такие, которые либо по собственной инициативе, либо по необходимости, вынуждены сами решать задачи администрирования своей вычислительной системы. Это может быть как домашний компьютер, для которого нет другого системного администратора, кроме самого пользователя, так и компьютер на работе. В последнем случае пользователь просто предпочитает не приглашать администратора при каждом затруднении, а самостоятельно пытается решить возникшие проблемы. Я не анализирую причин и побудительных мотивов, которые заставляют пользователя поступать таким образом. Важно только то, что человеку интересно или просто необходимо уметь устанавливать и настраивать программное обеспечение на своем компьютере, включая саму операционную систему. Именно таких пользователей я и называю "активными" (их часто называют "продвинутыми", но мне эта калька с английского режет слух). Можно было бы, наверное, найти и более точный термин, но в данном случае это не важно, поскольку далее настоящего предисловия это название я применять не собираюсь, а здесь вы меня, надеюсь, поймете правильно, учитывая приведенные разъяснения.

Итак, настоящая книга предназначена для "активных пользователей", к которым я отношу и себя. Предполагается, что читатель уже знает, что такое компьютер, и поработал с MS-DOS или MS Windows (уж если не версии XP или 2000, то хотя бы с Windows 95). Впрочем, я надеюсь, что книга будет полезна и начинающим системным администраторам.

О чем эта книга

Если вы относитесь к категории "активных пользователей", то вы уже что-то слышали о классе операционных систем, объединенных общим именем UNIX. ОС этого класса имеют ряд преимуществ перед обычно устанавливаемыми на персональные компьютеры ОС типа Windows. И если вы действительно "активный пользователь", то для вас вполне естественно желание хотя бы попробовать, что такое UNIX. Linux как раз и позволяет удовлетворить такое желание, поскольку с одной стороны она относится к классу UNIX, а с другой — работает на персональных компьютерах на основе процессоров Intel (хотя сейчас уже существуют ее варианты и для других процессоров).

В этой книге как раз и рассказывается об установке и настройке программного обеспечения на IBM-совместимом персональном компьютере, работающем под управлением операционной системы Linux, и даже конкретнее, под управлением одной из "веток" этой операционной системы. "Ветки" Linux называют дистрибутивами, так вот речь в книге пойдет о версиях Linux, основанных на дистрибутиве Red Hat. Материал, излагаемый в данной книге, основан на моем опыте установки и работы с дистрибутивами Black Cat Linux версий 5.2 (именно с нее началось мое знакомство с Linux) и 6.02, Red Hat версий 6.0, 6.2, 7.1 и 7.2 (русифицированной), ASPLinux 7.1 и ALTLinux Junior 1.0 и 1.1. Все эти дистрибутивы основаны на Red Hat Linux, и поэтому имеют много общего.

Однако не надо думать, что речь в книге пойдет только о самой операционной системе и ее настройке. В конце концов, сама по себе ОС выполняет только служебные функции, обеспечивая управление аппаратной частью компьютера и запуск нужных пользователю приложений. Поэтому основная задача данной книги в том, чтобы описать процесс создания на вашем компьютере "удобной" для пользователя (для вас) рабочей среды, состоящей из набора необходимых для вас приложений и обеспечивающей выполнение стоящих перед вами задач.

Книгу можно условно разделить на две части. Первая часть *(главы 1—15)* — это непосредственные рекомендации, касающиеся различных аспектов установки и настройки операционной системы и различных приложений. Предполагается, что читатель только что установил ОС Linux из дистрибутива и осваивает ее. Я надеюсь, что, прочитав даже только эту часть книги, чита-

тель сможет создать на персональном компьютере под ОС Linux удобную рабочую среду, сравнимую по возможностям с тем, что предоставляет ОС Windows 95 или Windows NT.

Вторая часть — это главы о внутреннем устройстве файловой системы, обновлении ядра и виртуальных компьютерах. Это те вопросы, которые могут показаться и неактуальными для начинающего пользователя. Но этот материал позволяет глубже понять устройство системы и эффективно решить проблему взаимодействия с миром Windows. Не надейтесь найти во второй части ответы на все вопросы, которые у вас могут возникнуть, никакого систематического отбора материала для этой части не производилось.

Как возникла эта книга

Мое знакомство с операционной системой Linux состоялось более двух лет назад. Не то, чтобы меня сильно "достала" (как говорят многие приверженцы Linux) ОС Windows, просто я всю свою жизнь с большим интересом учился, осваивал что-то новое. Поэтому новая ОС привлекла мое внимание, и я установил ее на домашнем компьютере (вместе с уже стоявшими на нем Windows 95 и Windows NT 4).

Когда я стал устанавливать и настраивать Linux, практически ничего не зная об этой ОС в частности, и имея очень слабые знания по UNIX вообще, я, естественно, начал с чтения различных руководств и HOWTO-файлов. Как оказалось, источники эти хотя и многочисленны, но пользоваться ими новичку крайне неудобно. Во-первых, значительная часть написана по-английски. Хотя я и читаю по-английски, но не так свободно, чтобы не считать недостатком отсутствие русскоязычной документации. Получается примерно так же, как читать неразборчиво написанный от руки текст: прежде чем уловить смысл написанного, приходится затратить существенные усилия на то, чтобы просто разобрать (узнать) отдельные слова. Во-вторых, новичку, естественно, хочется, чтобы его "за ручку" провели через несколько первоначальных этапов. А вместо этого приходится (по крайней мере, мне пришлось) по крохам отыскивать в разрозненных источниках нужную подсказку. Так что примерно через месяц после начала экспериментов с Linux, пройдя несколько этапов по 2—3 раза, повторяя при этом свои ошибки, я начал кое-что записывать, конспектировать разные руководства и документацию. Эти конспекты оказались очень полезны для меня самого. Я неоднократно пользовался ими, когда мне приходилось заново переустанавливать систему (я тогда еще не знал, как можно по-другому выбраться из некоторых затруднительных ситуаций, в которые попадал опять же из-за недостатка знаний).

Через некоторое время Linux перебрался и на пару компьютеров на моем рабочем месте. Мои конспекты здесь снова очень пригодились. Ведь запом-

нить с одного-двух раз все действия по установке и настройке различных программ практически невозможно, тем более, что в Linux многие настройки производятся путем прямого редактирования конфигурационных файлов.

Потом я выложил свои конспекты в Интернет, на сайте **http://linux-ve.chat.ru** и получил некоторое число довольно благожелательных отзывов от начинающих пользователей Linux. Оно и понятно: я описывал методы решения как раз тех проблем, с которыми сталкивается каждый новичок. Поэтому я решил, что если издать эти конспекты в виде книги, у нее тоже найдутся читатели. Надеюсь, что я не сильно ошибаюсь.

Честно сказать, при подготовке книги меня очень воодушевлял пример книги В. Э. Фигурнова "IBM PC для пользователя". В свое время (которое как раз совпало с периодом, когда я осваивал компьютер), наверное, вся наша страна училась работать на IBM-совместимых компьютерах (помните PC/XT!?) именно по этой книге. Она появилась в ответ на насущнейшее требование времени и выдержала с тех пор множество переизданий. Думаю, что аналогичная потребность в книге по Linux имеется сейчас (2002 год), поскольку эта ОС динамично развивается, приобретает все больше почитателей и имеет серьезные преимущества по сравнению с MS Windows 95/98. Поэтому я, следуя примеру В. Э. Фигурнова, постарался отобрать весь самый необходимый для освоения Linux материал, систематически его изложить, чтобы начинающему пользователю было удобно с ним работать. Не думаю, что я достиг идеального варианта в выборе материала, поэтому с благодарностью приму все замечания и пожелания читателей на эту тему (как и любые другие замечания), с тем, чтобы учесть их в дальнейшей работе.

Типографские соглашения

Практически в каждом разделе книги приводятся какие-то примеры команд или сообщений, выдаваемых системой. Если речь идет о графическом режиме, то обычно приводится снимок экрана. Если же пример касается работы в текстовом режиме, то он выделяется шрифтом Courier — вот так:

```
[root]# sfdisk -l -x /dev/hda
Disk /dev/hda: 784 cylinders, 255 heads, 63 sectors/track
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

Device Boot    Start     End   #cyls    ttblocks   Id  System
/dev/hda1   *      0+     189     190-     1526143+   6  FAT16
/dev/hda2        190     783     594      4771305    5  Extended
```

Тем же шрифтом Courier выделены все упоминания команд Linux, встречающиеся в тексте.

В качестве приглашения оболочки в примерах используется строка `[root]#`, если команда должна выполняться от имени пользователя root, и строка [user] $, если команда может выполняться от имени обычного пользователя.

Если говорится о том, что надо нажать какую-то клавишу, то название клавиши (точнее то обозначение, которое нанесено на клавише) заключается в угловые скобки: <Enter>, <Esc>, <Ctrl>, <Alt>, <A>, <S> и т. д. Если должны быть нажаты одновременно несколько клавиш, то обозначения отдельных клавиш соединяются знаком + : <Ctrl>+<Alt>+<Del>, <Ctrl>+<X>. Если же нужно последовательно нажать несколько клавиш или соответствующих комбинаций, то разделителем будет служить запятая: <Ctrl>+<X>, <C> или <Esc>, <5>.

Отдельно нужно сказать о клавише <META>, которая часто упоминается в разных HOWTO и руководствах, а также используется в некоторых приложениях, например, редакторе Emacs. Говорят, такая клавиша была на старых клавиатурах UNIX-компьютеров, поэтому она и используется в UNIX-системах. Однако на PC-клавиатурах такой клавиши нет, и ее приходится эмулировать. В консоли вместо <META> вы можете использовать клавишу <Alt>. В системе X Window (в графической оболочке) это может не сработать. Поскольку <META> — это клавиша-модификатор, то она упоминается обычно, когда требуется нажать комбинацию <META> с какой-то другой клавишей. В таких случаях надо нажать клавишу <Esc>, отпустить ее, после чего нажать вторую требуемую клавишу. Но все сказанное относительно <META> надо иметь в виду, когда вы будете читать HOWTO и прилагаемые к программам руководства. В этой же книге я постараюсь корректно указывать, какие комбинации надо набирать на клавиатуре PC.

Ссылки на литературу и источники в Интернете сведены в *приложение* в конце книги (мне думается, что так их легче отыскать в процессе чтения). Однако нумерация ссылок производится в рамках каждой главы отдельно. Поэтому в тексте ссылка приводится с указанием на конкретный раздел приложения: [П12.7].

Благодарности

Эту свою книгу я хотел бы посвятить своим родителям: Костромину Алексею Гордеевичу и Костроминой Лидии Ермолаевне. Мама уже не увидит ее, а у отца еще есть шанс. Конечно, они не смогли бы покритиковать или похвалить ее содержание, но именно благодаря той привычке к труду, которую они во мне воспитали, эта книга может увидеть свет.

Я изучал Linux в основном по документации и различным книгам и статьям. Авторам этих руководств я очень благодарен, но хочу сразу принести свои извинения тем из них, материалы которых я использовал в данном руководстве без явной ссылки. Как уже было сказано, первоначально я просто

конспектировал документацию и различные интернет-источники. При этом я не очень заботился о ссылках, был бы материал интересен и полезен (ведь первый вариант этих заметок создавался для себя). Впоследствии я пытался восстановить ссылки, но, боюсь, мне это не везде удалось.

Черновой вариант книги я послал нескольким своим заочным (по контактам в Интернете) знакомым и очень благодарен им за те замечания, которые они высказали. Особенно мне хочется отметить Романа Сузи, который прислал множество замечаний и предложений по улучшению содержания. Я понимаю, какой огромный труд пришлось ему проделать, чтобы прочитать (и не бегло) более 500 страниц текста и прокомментировать этот текст, и поэтому выражаю ему свою искреннюю признательность. Я старался учесть и те замечания, которые были присланы читателями чернового варианта книги, размещенного в Интернете (в частности С. Воеводиным).

Большую помощь в работе над книгой мне оказали и сотрудники издательства "БХВ-Петербург" (ведь эта книга — первый мой опыт работы с издательством). Всем им большое спасибо.

Глава 1

ОС Linux: история и дистрибутивы

1.1. Что такое ОС вообще и Linux в частности

1.1.1. Семейство ОС типа UNIX

Операционная система — это комплекс программ, который обеспечивает управление аппаратными средствами компьютера, организует работу с файлами (в том числе запуск и управление выполнением программ), а также реализует взаимодействие с пользователем, т. е. интерпретацию вводимых пользователем команд и вывод результатов обработки этих команд.

Без операционной системы компьютер вообще не может функционировать в качестве такового. В таком случае он представляет собой не более чем совокупность неработающих электронных устройств, непонятно зачем собранных воедино.

На сегодняшний день наиболее известными операционными системами для компьютеров являются семейства операционных систем Microsoft Windows и UNIX. Первые ведут свою "родословную" от операционной системы MS-DOS, которой оснащались первые персональные компьютеры фирмы IBM. Операционная система UNIX была разработана группой сотрудников Bell Labs под руководством Денниса Ричи, Кена Томпсона и Брайана Кернигана (Dennis Ritchie, Ken Thompson, Brian Kernighan) в 1969 году. Но в наши дни, когда говорят об операционной системе UNIX, чаще всего имеют в виду не конкретную ОС, а скорее целое семейство UNIX-подобных операционных систем. Само же слово UNIX (заглавными буквами) стало зарегистрированной торговой маркой корпорации AT&T.

В конце 70-х годов (теперь уже прошлого столетия) сотрудники Калифорнийского университета в Беркли внесли ряд усовершенствований в исходные коды UNIX, включая работу с протоколами семейства TCP/IP. Их разработка стала известна под именем BSD ("Berkeley Systems Distribution"). Она распространялась под лицензией, которая позволяла дорабатывать и усовершенствовать продукт, и передавать результат третьим лицам (с исходными кодами или без них) при условии, что будет указано, какая часть кода разработана в Беркли.

Операционные системы типа UNIX, в том числе и BSD, изначально разрабатывались для работы на больших многопользовательских компьютерах — мейнфреймах. Но персональные компьютеры постепенно наращивали мощь своего аппаратного обеспечения, и в наши дни они уже превосходят по возможностям те мейнфреймы, для которых в 70-х годах разрабатывалась ОС UNIX. И вот, в начале 90-х годов студент хельсинкского университета Линус Торвальдс (Linus Torvalds) приступил к разработке UNIX-подобной ОС для IBM-совместимых персональных компьютеров.

1.1.2. Немного истории

Вот текст сообщения, которое Торвальдс отправил в группу новостей **comp.os.minix** 25 августа 1991 года:

> From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
> Newsgroups: comp.os.minix
> Subject: What would you like to see most in minix?
> Summary: small poll for my new operating system
> Message-ID:<1991Aug25.205708.9541@klaava.Helsinki.FI>
> Date: 25 Aug 91 20:57:08 GMT
> Organization: University of Helsinki
>
> Hello everybody out there using minix -
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).
> I've currently ported bash(1.08) and gcc(1.40), and things seem to work.
> This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)
> Linus (torvalds@kruuna.helsinki.fi)
>
> PS. Yes — it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

В этом сообщении Линус пишет, что он работает над (свободной) операционной системой для 386-х (486-х) компьютеров, и просит всех заинтересованных лиц сообщить, какие компоненты системы пользователи хотят видеть в первую очередь. Но, как видно из текста послания, оболочка bash и компилятор gcc у него уже работали. Работали они под управлением операционной системы Minix, которая была разработана профессором Э. Таненбаумом (Andy Tanenbaum) как учебное пособие для студентов-программистов. Minix работала на компьютерах с 286-ым процессором и послужила для Торвальдса прообразом новой ОС.

Файлы первого варианта Linux (версия 0.01) были опубликованы в Интернете 17 сентября 1991 года. Как пишет сам Торвальдс "As I already mentioned, 0.01 didn't actually come with any binaries: it was just source code for people interested in what linux looked like. Note the lack of announcement for 0.01: I wasn't too proud of it, so I think I only sent a note to everybody who had shown interest."[1]

Затем, 5 октября 1991 года была выпушена версия 0.02, которая уже работала. Впрочем, подробное изложение истории Linux не входит в задачи данной книги, поэтому продолжать данную тему я не буду, отсылая заинтересованных читателей к *(см. [ПЗ.1] приложения)*.

Л. Торвальдс не стал патентовать или иным образом ограничивать распространение новой ОС. С самого начала Linux распространяется на условиях, определяемых лицензией General Public License (GPL), принятой для программного обеспечения, разрабатываемого в рамках движения Open Source и проекта GNU *(см. [ПЗ.2]приложения)*. На Linux-сленге эту лицензию иногда называют Copyleft. Об этой лицензии, движении Open Source и проекте GNU необходимо поговорить особо.

В 1984 году американский ученый Ричард Столлман (Richard Stallman) основал Фонд Свободного Программного Обеспечения (Free Software Foundation). Целью этого фонда было устранение всех запретов и ограничений по распространению, копированию, модификации и изучению программного обеспечения. Ведь до тех пор коммерческие компании тщательно оберегали разработанное ими программное обеспечение, ограждали его патентами и знаками защиты авторских прав, держали в строжайшем секрете исходные коды программ, написанных на языках высокого уровня (типа C++). Столлман считал, что это наносит огромный вред развитию ПО, приводит к снижению качества программ и наличию в них огромного количества невыявленных ошибок. И, что хуже всего, это приводит к замедлению процесса обмена идеями в области программирования, тормозит создание нового ПО в силу того, что каждому программисту приходится полностью заново писать каждую программу, вместо того, чтобы заимствовать уже готовые куски исходного кода из готовых программ.

В рамках Фонда Свободного ПО была начата разработка проекта GNU — проекта создания свободного программного обеспечения. Аббревиатура GNU открывается рекурсивно — GNU's Not UNIX, т. е. то, что принадлежит проекту GNU, не является частью UNIX (потому что к тому времени даже само слово UNIX уже было зарегистрированной товарной маркой, т. е. перестало быть свободным). В "Манифесте GNU" *(см. [ПЗ.3] приложения),* который был

[1] "Как я уже упоминал, версия 0.01 распространялась без бинарников: это были просто исходные коды, предназначенные для тех, кому интересно, как выглядит linux. Обратите внимание на то, что не было объявления о выходе версии 0.01: я не очень ею гордился, так что просто послал сообщение всем, кто проявил какой-то интерес".

написан в 1985 г., Р. Столлман в качестве главной движущей силы, которая привела к возникновению FSF и проекта GNU, ставит свое неприятие прав собственности отдельных людей на программное обеспечение.

То, что разрабатываемое в рамках проекта GNU ПО свободно, не означает, что оно распространяется без лицензии и никак не защищено в юридическом смысле. Программы, разрабатываемые в рамках движения Open Source, распространяются на условиях лицензии General Public License (GPL) *(см. [ПЗ.2] приложения)*. Если сказать очень кратко, то суть этой лицензии состоит в следующем. Программное обеспечение, распространяемое под этой лицензией, можно как угодно дорабатывать, модифицировать, передавать или продавать другим лицам при условии, что результат такой переработки тоже будет распространяться под лицензией copyleft. Последнее условие — самое важное и определяющее в этой лицензии. Оно гарантирует, что результаты усилий разработчиков свободного ПО останутся открытыми и не станут частью какого-либо лицензированного обычным способом продукта. Оно также отличает свободное ПО от ПО, распространяемого бесплатно. Говоря словами создателей FSF, лицензия GPL "делает ПО свободным и гарантирует, что оно останется свободным".

Практически все ПО, распространяемое на условиях GPL, является почти бесплатным для пользователей (в большинстве случаев для того, чтобы получить его, вы должны заплатить только за CD-ROM с ПО или за трафик выхода в Интернет). Это не означает, что профаммисты перестают получать вознаграждение за свой труд. Основная мысль Р. Столлмана состоит в том, что нужно продавать не профаммное обеспечение, а труд профаммиста как такового. Например, источником дохода может быть сопровождение программных продуктов или их установка и конфигурация для внедрения на новых компьютерах и/или в новых условиях, преподавание и т. д. Хорошим вознаграждением может быть и получение автором свободных программ определенной известности, которая позволит ему в последующем получить высокооплачиваемую работу.

В рамках движения Open Source, и в частности проекта GNU, было разработано значительное количество профамм, наиболее известными из которых являются редактор Emacs и компилятор GCC (GNU C Compiler) — самый лучший и по сей день компилятор языка С. Открытость исходных кодов профамм оказывает очень благотворное влияние на качество профаммного обеспечения: все лучшее, все новые идеи и решения сразу же широко распространяются, а все ошибки замечаются и быстро устраняются. Начинает работать механизм естественного отбора, который подавлен в том варианте подхода к распространению программ, который практикуется в коммерческом ПО.

Но вернемся к истории собственно Linux. Надо сказать, что разработка Линуса Торвальдса представляла собой только ядро операционной системы. Это ядро "упало на подготовленную почву", в том смысле, что в рамках про-

екта GNU уже было разработано большое количество утилит разного рода. Но для превращения GNU в полноценную ОС не хватало ядра. Разработка ядра велась (оно называлось Hurd), но по каким-то причинам задерживалась. Поэтому появление разработки Л. Торвальдса было очень своевременным. Оно ознаменовало рождение операционной системы, распространяемой с открытыми исходными кодами.

Р. Столлман, конечно, прав, когда настаивает на том, что операционная система Linux должна называться GNU/Linux. Но так уж сложилось, что название ядра стало служить названием всей операционной системы, и мы в этой книге будем поступать так же.

1.1.3. Основные характеристики ОС Linux

В силу того, что исходные коды Linux распространяются свободно и общедоступны, к развитию системы с самого начала подключилось большое число независимых разработчиков. Благодаря этому на сегодняшний день Linux — самая современная, устойчивая и быстроразвивающаяся система, почти мгновенно вбирающая в себя самые последние технологические новшества. Она обладает всеми возможностями, которые присущи современным полнофункциональным операционным системам типа UNIX. Приведем краткий список этих возможностей.

Реальная многозадачность

Все процессы независимы; ни один из них не должен мешать выполнению других задач. Для этого ядро осуществляет режим разделения времени центрального процессора, поочередно выделяя каждому процессу интервалы времени для выполнения. Это существенно отличается от режима "вытесняющей многозадачности", реализованной в Windows 95, когда процесс должен сам "уступить" процессор другим процессам (и может сильно задержать их выполнение).

Многопользовательский доступ

Linux — не только многозадачная ОС, она поддерживает возможность одновременной работы многих пользователей. При этом Linux может предоставлять все системные ресурсы пользователям, работающим с хостом через различные удаленные терминалы.

Свопирование оперативной памяти на диск

Свопирование оперативной памяти на диск позволяет работать при ограниченном объеме физической оперативной памяти; для этого содержимое некоторых частей (страниц) оперативной памяти записываются в выделенную область на жестком диске, которая трактуется как дополнительная опера-

тивная память. Это несколько снижает скорость работы, но позволяет организовать работу программ, требующих большего объема ОЗУ, чем фактически имеется в компьютере.

Страничная организация памяти

Системная память Linux организована в виде страниц объемом 4 Кбайт. Если оперативная память полностью исчерпана, ОС будет искать давно не использующие страницы памяти для их перемещения из памяти на жесткий диск. Если какие-либо из этих страниц становятся нужны, Linux восстанавливает их с диска. Некоторые старые UNIX-системы и некоторые современные платформы (включая Microsoft Windows) переносят на диск все содержимое ОП, относящееся к неработающему в данный момент приложению, (то есть ВСЕ страницы памяти, относящиеся к приложению, сохраняются на диске при нехватке памяти), что менее эффективно.

Загрузка выполняемых модулей "по требованию"

Ядро Linux поддерживает выделение страниц памяти по требованию, при котором только необходимая часть кода исполняемой программы находится в оперативной памяти, а не используемые в данный момент части остаются на диске.

Совместное использование исполняемых программ

Если необходимо запустить одновременно несколько копий какого-то приложения (либо один пользователь запускает несколько идентичных задач, либо разные пользователи запускают одну и ту же задачу), то в память загружается только одна копия исполняемого кода этого приложения, которая используется всеми одновременно исполняющимися идентичными задачами.

Общие библиотеки

Библиотеки — наборы процедур, используемых программами для обработки данных. Существует некоторое количество стандартных библиотек, используемых одновременно более чем одним процессом. В старых системах такие библиотеки включались в каждый исполняемый файл, одновременное выполнение которых приводило к непродуктивному использованию памяти. В новых системах (в частности, в Linux) обеспечивается работа с динамически и статически разделяемыми библиотеками, что позволяет сократить размер отдельных приложений.

Динамическое кэширование диска

Кэширование диска — это использование части оперативной памяти для хранения часто используемых данных с диска, что существенно ускоряет

доступ к часто используемым программам и задачам. Пользователи MS-DOS работают со SmartDrive, который резервирует фиксированные области системной памяти для кэширования диска. Linux использует более динамичную систему кэширования: память, зарезервированная под кэш, увеличивается, когда память не используется, и уменьшается, если системе или процессу пользователя требуется больше памяти.

100%-ное соответствие стандарту POSIX 1003.1. Частичная поддержка возможностей System V и BSD

POSIX 1003.1 (Portable Operating System Interface — интерфейс мобильной операционной системы) задает стандартный интерфейс UNIX-систем, который описывается набором процедур языка С. Сейчас он поддерживается всеми новыми ОС. Microsoft Windows NT также поддерживает POSIX 1003.1. Linux 100%-но соответствует POSIX. Дополнительно поддерживаются некоторые возможности System V и BSD для увеличения совместимости.

System V IPC

Linux применяет технологию IPC (Internal System Call) для обмена сообщениями между процессами, использования семафоров и общей памяти.

Возможность запуска исполняемых файлов других ОС

Linux не является первой в истории операционной системой. Для ранее разработанных ОС, включая DOS, Windows 95, FreeBSD или OS/2, разработана масса различного, в том числе очень полезного и очень неплохого программного обеспечения. Для запуска таких программ под Linux разработаны эмуляторы DOS, Windows 3.1 и Windows 95. Более того, фирмой Vmware разработана система "виртуальных машин", представляющая собой эмулятор компьютера, в котором можно запустить любую операционную систему. Имеются аналогичные разработки и у других фирм. ОС Linux способна также выполнять бинарные файлы других Intel-ориентированных UNIX-платформ, соответствующих стандарту iBCS2 (intel Binary Compatibility).

Поддержка различных форматов файловых систем

Linux поддерживает большое число форматов файловых систем, включая файловые системы DOS и OS/2, а также современные журналируемые файловые системы. При этом и собственная файловая система Linux, которая называется Second Extended File System (ext2fs), позволяет эффективно использовать дисковое пространство.

Сетевые возможности

Linux можно интегрировать в любую локальную сеть. Поддерживаются все службы UNIX, включая Networked File System (NFS), удаленный доступ (telnet, rlogin), работа в TCP/IP-сетях, dial-up-доступ по протоколам SLIP и PPP и т. д. Также поддерживается включение Linux-машины как сервера или клиента для другой сети, в частности, работает общее использование (sharing) файлов и удаленная печать в Macintosh, NetWare и Windows.

Работа на разных аппаратных платформах

Хотя ОС Linux первоначально была разработана для ПК на базе Intel 386/486, сейчас она может работать на всех версиях микропроцессоров от Intel, начиная с 386 и кончая многопроцессорными системами на Pentium III (с Pentium IV возникли определенные трудности, но, судя по сообщениям в Интернете, они были вызваны ошибками в реализации процессора). Так же успешно Linux работает на различных клонах Intel от других производителей; в Интернете встречаются сообщения о том, что на процессорах Athlon и Duron от AMD Linux работает даже лучше, чем на Intel. Кроме того, разработаны версии для других типов процессоров — ARM, DEC Alpha, SUN Spare, M68000 (Atari и Amiga), MIPS, PowerPC и др. (отметим, что в настоящей книге рассматривается только вариант для IBM-совместимых компьютеров).

1.2. Дистрибутивы Linux

В любой операционной системе можно выделить 4 основных части: ядро, файловую структуру, интерпретатор команд пользователя и утилиты. *Ядро* — это основная, определяющая часть ОС, которая управляет аппаратными средствами и выполнением программ. *Файловая структура* — это система хранения файлов на запоминающих устройствах. *Интерпретатор команд* или *оболочка* — это программа, организующая взаимодействие пользователя с компьютером. И, наконец, *утилиты* — это просто отдельные программы, которые, вообще говоря, ничем принципиально не отличаются от других программ, запускаемых пользователем, разве только своим основным назначением — они выполняют служебные функции.

Как уже говорилось выше, если быть точным, то слово "Linux" обозначает только ядро. Поэтому, когда речь идет об операционной системе, правильнее было бы говорить "операционная система, основанная на ядре Linux". Ядро ОС Linux разрабатывается под общим руководством Линуса Торвальдса и распространяется свободно (на основе лицензии GPL), как и огромное количество другого программного обеспечения, утилит и прикладных программ. Одним из следствий свободного распространения ПО для Linux явилось то, что большое число разных фирм и компаний, а также просто неза-

висимых групп разработчиков стали выпускать так называемые дистрибутивы Linux.

Дистрибутив — это набор программного обеспечения, включающий все 4 основные составные части ОС, т. е. ядро, файловую систему, оболочку и совокупность утилит, а также некоторую совокупность прикладных программ. Обычно все программы, включаемые в дистрибутив Linux, распространяются на условиях GPL, так что может сложиться впечатление, что дистрибутив может выпустить кто угодно, точнее любой, кто не поленится собрать коллекцию свободного ПО. И в какой-то степени это утверждение правдоподобно. Однако разработчик дистрибутива должен по крайней мере создать программу инсталляции, которая будет устанавливать ОС на компьютер, на котором никакой ОС еще нет. Кроме того, необходимо обеспечить разрешение взаимозависимостей и противоречий между разными пакетами (и версиями пакетов), что, как мы увидим позже, тоже является нетривиальной задачей.

Тем не менее, в мире существует уже более сотни различных дистрибутивов Linux и все время появляются новые. Более-менее полный список их можно найти на сервере **http://www.linuxhq.com,** где даны краткие характеристики каждому дистрибутиву (упоминаются и некоторые локализованные версии). Кроме того, там же есть ссылки на другие списки дистрибутивов, так что при желании можно найти все, что вообще существует в мире (правда, все это на английском языке, и русских локализаций там маловато упомянуто).

А. Федорчук в своей статье *(см. [П3.8] приложения)* предпринял попытку классификации дистрибутивов, положив в основу следующие критерии:

- структура файловой системы;
- программа инсталляции;
- используемое средство установки программных пакетов;
- состав утилит и прикладных программ, включенных в дистрибутив.

Хотя А. Федорчук и приходит к выводу, что различия между дистрибутивами несущественны и все более стираются, из его статьи все же следует, что на сегодняшний день выделяются по крайней мере 3 группы дистрибутивов, наиболее типичными представителями которых являются Red Hat, Slackware и Debian.

По какому же критерию выбрать дистрибутив? На мой взгляд, для пользователя нашей страны критериев два: дистрибутив должен быть русифицирован и должна существовать команда разработчиков, обеспечивающая поддержку дистрибутива. И лучше, если эта команда имеет от этой (или, может быть, какой-то другой) деятельности некоторый доход, т. е. функционирует как коммерческая фирма. Даже за тот сравнительно недолгий период, в течение которого я занимаюсь Linux, успели сойти со сцены несколько дистрибути-

BOB, команды поддержки которых работали "на общественных началах" и через некоторое время перестали поддерживать свои разработки.

В России в последнее время сложилось три команды разработчиков, создающих и поддерживающих русифицированные дистрибутивы.

Одна из команд сформировалась в Институте Логики (**http://www.iplabs.ru**). Эта команда некоторое время занималась русификацией дистрибутива Linux Mandrake Russian Edition, а в марте 2001 года организовала фирму ALTLinux (**http://www.altlinux.ru**) и выпустила собственный дистрибутив ALTLinux (который, впрочем, очень похож на Linux Mandrake Russian Edition).

Вторая команда представлена фирмой ASPLinux **(http://www.asplinux.ru, http://www.asp-linux.com, http://www.asp-linux.com.sg, http://www.asp-linux.co.kr),** которая тоже выпустила собственный дистрибутив ASPLinux. В состав этой команды вошли Л. Кантер и А. Каневский, которые раньше выпускали известный дистрибутив Black Cat Linux.

Третья команда, насколько я могу судить, представлена санкт-петербургской фирмой Linux Ink. **(http://www.linux-ink.ru),** которая выпускает Red Hat Linux Cyrillic Edition.

Конечно, имеются и другие русифицированные дистрибутивы. В 2000 году появились дистрибутив Best Linux (**http://bestlinux.net**), поддерживаемый фирмой SOT из Финляндии, и RosLinux. Описание нескольких русифицированных дистрибутивов Linux дано в книге А. Федорчука *(см. [П1.6] приложения)*. Но, на мой взгляд, если говорить о выборе дистрибутива, то на сегодняшний день заслуживают внимания только три дистрибутива: Red Hat Linux Cyrillic Edition, Linux Mandrake Russian Edition (и его потомок ALTLinux) и ASPLinux. Я могу привести следующие доводы в пользу такого выбора.

- ❒ Эти дистрибутивы принадлежат к семейству дистрибутивов, строящихся на основе Red Hat Linux, выпускаемого одноименной американской фирмой, а судя по материалам Интернета, Red Hat — это самый распространенный в мире дистрибутив.
- ❒ Эти дистрибутивы изначально русифицированы.
- ❒ В каждом из них имеется достаточно отлаженная процедура установки, автоматически распознающая большинство компонентов аппаратного обеспечения, что очень облегчает процедуру инсталляции системы.
- П Легко устанавливается (добавляется) дополнительное программное обеспечение, поскольку оно поставляется в RPM-пакетах (это такая технология распространения ПО, вроде программы setup под Windows).
- П Эти дистрибутивы поддерживаются сформировавшимися командами разработчиков и постоянно обновляются, поэтому можно рассчитывать на то, что вы будете иметь возможность работать с последними версиями Linux.

Несколько слов о нумерации версий. Надо различать номера версий дистрибутивов и номера версий ядра. Когда говорят о версиях Linux, то обычно имеют в виду версию ядра (ибо принадлежность операционной системы к Linux определяется тем, что ОС использует ядро Linux). Поскольку Линус Торвальдс продолжает координировать разработку ядра, то версии ядра развиваются последовательно, а не ветвятся и множатся, как дистрибутивы.

Версии ядра Linux принято обозначать тремя числами, разделенными точкой. Например, дистрибутив Black Cat версии 5.2 был построен на основе ядра версии 2.0.36, т. е. это был Linux версии 2.0.36. Версии ядра с нечетным значением второй цифры обычно не используются для создания дистрибутивов, потому что являются экспериментальными (отладочными). Они распространяются, в основном, для того, чтобы энтузиасты могли их протестировать с целью выявления всех недостатков. Естественно, что такая версия может работать неустойчиво. Версии с четной второй цифрой являются (считаются) устойчиво работающими. Вы, конечно, можете установить любую версию, но для новичков все же обычно рекомендуют выбирать версию ядра с четной второй цифрой в номере версии. Конечно, если вы устанавливаете полный дистрибутив, то выбор ядра за вас сделали его разработчики, но о нумерации версий вам надо знать, если вы когда-нибудь задумаетесь об обновлении ядра.

1.3. Требования к компьютеру

Я встречал упоминания о том, что существуют специальные версии Linux, которые работают даже на 8086-ом процессоре с 512 Кбайт памяти, а специально собранная версия может запускаться с одной-двух дискет без жесткого диска.

Так что, если у вас есть старенький компьютер, на котором никакая Windows не запускается, то вы с успехом можете использовать его для освоения Linux и, возможно, будете удивлены его возможностями. Но такие варианты в данной книге не рассматриваются.

Поскольку ОС Linux использует защищенный режим микропроцессора, то для установки этой ОС требуется как минимум 386-ой процессор. Судя по литературным источникам, годятся любые модификации: SX, DX и т. д. Дальнейшие требования к аппаратной части компьютера, на который устанавливается Linux, определяются уже тем, что вы хотите. Из табл. 1.1 видно, как возрастают требования к аппаратной части в зависимости от пожеланий пользователя (приводимые в таблице числа очень приблизительны, тут я не претендую на истину в последней инстанции).

Таблица 1.1. *Требования к аппаратуре*

Пожелания пользователя	Требования к памяти	Требования к объему жесткого диска
Минимальные требования: работа в текстовом режиме из командной строки shell	4 Мбайт	10 Мбайт
Работа в текстовом режиме через Midnight Commander	4 Мбайт	40 Мбайт
Для запуска графического интерфейса X Window	8 Мбайт, но будет работать очень медленно, 16 Мбайт — более-менее приемлемо	
Для работы с графическим интерфейсом X Window (запуск оконного менеджера)	16 Мбайт	300 Мбайт
Для запуска интегрированной графической среды KDE	32 Мбайт	500 Мбайт
Для запуска каждого отдельного большого приложения (типа GIMP, текстового процессора, базы данных или электронной таблицы)	+ 2 Мбайт	+50—100 Мбайт
Для работы с интегрированным офисным пакетом StarOffice	64 Мбайт	+ 250 Мбайт

Из этой таблицы можно заключить, что минимально приемлемой конфигурацией для освоения Linux является компьютер на 486-ом процессоре с 16 Мбайт ОЗУ и жестким диском объемом 300 Мбайт. Далее надо заботиться только о наращивании оперативной памяти и объема жесткого диска, тут лишнего никогда не будет.

Снова сошлюсь на книгу А. Федорчука *(см. [П1.6] приложения)*, в которой большая глава посвящена выбору аппаратной платформы для Linux. В ней автор подробно рассматривает, как Linux относится к каждому компоненту компьютерной аппаратуры, начиная с чипсета и системной платы и кончая периферийными устройствами и источниками бесперебойного питания. Однако, на мой взгляд, на практике выбор компьютера определяется не операционной системой, а, в первую очередь, материальными возможностями владельца. И надо отнести к достоинствам ОС ее способность управлять не только самыми последними и "навороченными" моделями, но и уже "вышедшими из моды" или "морально устаревшими" экземплярами. Ведь так называемое "моральное устаревание" как раз и вызвано тем, что новые версии ПО от самых известных производителей заставляют списать в утиль

вполне работоспособное оборудование. В этом смысле Linux имеет огромное преимущество, заключающееся в том, что она способна работать даже на тех компьютерах, где альтернативой ей может быть только MS-DOS. Конечно, в таких случаях мы получим только режим командной строки, но, судя по различным источникам в Интернете, это не мешает использовать старые компьютеры для выполнения различных вспомогательных задач, например, в качестве маршрутизаторов.

Но вопросы использования Linux для этих целей не попадают в сферу нашего интереса. Если же говорить о типичном пользователе, то, судя по моему опыту, если вы можете на компьютере работать с ОС Windows 95, а тем более с Windows NT или Windows 2000, то такой компьютер вполне годится для запуска Linux.

1.4. Где взять Linux?

И в заключение *главы 1* краткий ответ на вопрос, сформулированный в заголовке раздела.

Как было сказано, Linux вместе с огромным количеством прикладных программ распространяется практически бесплатно. Это значит, что пользователь, который не собирается модифицировать ПО или заниматься его продажами, имеет полное право скопировать весь дистрибутив Linux или любые его части у знакомого, скачать из Интернета или купить CD-ROM с Linux у торговцев в подземном переходе, не опасаясь, что подвергнется преследованию за нарушение лицензионных требований (которые почему-то называются "соглашениями"), выставленных фирмой-разработчиком.

Из трех перечисленных вариантов приобретения дистрибутива я бы предложил выбрать приобретение его на CD-ROM. Покупать желательно не в подземном переходе (хотя первый свой дистрибутив я приобрел на местном рынке и не пожалел), а в одной из компьютерных фирм или через интернет-магазин. Это дает возможность выбора и некоторые гарантии, по крайней мере, по обмену бракованного диска. Только имейте в виду, что разброс цен может быть очень велик. Существуют красивые упаковки с ценой более 1000 рублей (и это право продавца — назначить цену). И тот же дистрибутив (может быть только без печатного руководства по инсталляции) можно купить за сотню-другую.

Сам я в последнее время пользуюсь услугами интернет-магазинов. Конкретный адрес я не указываю (реклама у нас теперь платная), но таковых теперь множество, так что отсутствие здесь конкретного адреса не препятствие для тех, у кого есть желание приобрести дистрибутив.

Глава 2

Инсталляция ОС Linux на компьютер с Windows

К фирменным русифицированным дистрибутивам Linux прилагается краткое руководство по инсталляции системы. А на дистрибутивном диске (дисках) обычно имеется и достаточно полное руководство по установке Linux в электронной версии (такое руководство для Red Hat Linux Russian Edition вы можете найти на сайте **http://www.linux-ink.ru).** Кроме того, на русском языке опубликована отличная книга М. Уэлша и др. *(см. [П4.1] приложения)*, которую легко найти и в Интернете под названием "Инсталляция Linux и первые шаги" (правда, в одной из более ранних редакций). Поэтому я не буду в этой книге подробно излагать этапы инсталляции системы, отсылая читателя к упомянутым руководствам. Вместо этого расскажу только о том, что нужно знать и приготовить до начала инсталляции, а также подробно рассмотрю вопрос об особенностях установки Linux на компьютер, уже работающий под одной из ОС семейства Windows, и об организации многовариантной загрузки.

Дело в том, что большинство из тех, кто начинает осваивать Linux, являются уже опытными пользователями ОС Windows, чаще всего Windows 95 или Windows 98. Действительно, пока еще вряд ли Linux является той операционной системой, с которой пользователь начинает свое знакомство с компьютером. И, естественно, если пользователь раньше работал с ОС Windows и решил поэкспериментировать с Linux, то ему не хочется терять свою привычную среду и все, что было наработано, настроено и отлажено под Windows. К счастью, терять и необязательно, потому что на одном компьютере вполне могут ужиться две и даже больше ОС (хватило бы места на диске!). Именно поэтому ниже будет рассказано о том, как установить ОС Linux на компьютер, который до тех пор работал под одной из операционных систем фирмы Microsoft.

2.1. Подготовка к инсталляции

Устанавливать Linux можно одним из следующих способов:

- с локального CD-ROM;
- с жесткого диска, на который скопирован дистрибутив Linux;
- с файл-сервера локальной сети по NFS;

- с другого компьютера в локальной сети через SMB;
- с удаленного компьютера (в том числе из Интернета) по протоколу FTP;
- с одного из WWW-серверов Интернета по протоколу HTMP.

На мой взгляд, наиболее удобен и практичен вариант установки Linux с CD-ROM, тем более, что купить нужный компакт-диск теперь не проблема.

Прежде чем приступить к инсталляции, соберите (запишите на листе бумаги) всю необходимую информацию о конфигурации вашего компьютера. Если ваш компьютер пока что работает под ОС Windows 95/98, то многое можно найти, щелкнув правой кнопкой мыши по значку My Computer (Мой компьютер), а затем выбрав команду Properties (Свойства). Там можно найти почти всю необходимую информацию. Если чего-то не найдете, придется искать другими способами, вплоть до того, что открыть компьютер и прочитать надписи на самих устройствах.

Чтобы ничего не забыть, предлагаю вам пользоваться следующим списком. Не ленитесь и постарайтесь записать о каждом устройстве как можно больше данных (какие только сможете найти), это все может пригодиться при установке и настройке, когда искать эти данные будет сложнее.

- BIOS:
  - фирма-производитель;
  - версия.
- Контроллеры жестких дисков: тип (IDE или SCSI) и объем ваших жестких дисков (если у вас IDE-диски, вы должны проверить, что BIOS вашего компьютера обеспечивает доступ к ним в режиме LBA):
  - hda (Master на 1-ом контроллере или Primary Master);
  - hdb (Slave на 1-ом контроллере или Primary Slave);
  - hdc (Master на 2-ом контроллере или Secondary Master);
  - hdd (Slave на 2-ом контроллере или Secondary Slave);
  - фирма-производитель и номер модели SCSI адаптера (если имеется).
- Объем оперативной памяти (в килобайтах).
- CD-ROM:
  - тип интерфейса (IDE, SCSI или иной);
  - для не-IDE, не-SCSI CD-ROM — фирма-производитель и номер модели.
- Мышь:
  - тип (serial, PS/2, или bus mouse);
  - протокол (Microsoft, Logitech, MouseMan и т. д.);

- число кнопок;
- для мыши на последовательном порту также номер порта, к которому она подсоединена.

О Адаптер дисплея:

- фирма-производитель;
- номер модели (или чипсет, который использован);
- количество видеопамяти;

❐ Монитор:

- фирма-производитель;
- номер модели;
- граничные значения (min, max) частоты вертикальной и горизонтальной развертки (эти данные вы можете найти только в документации на монитор, их Windows не выдает, а между тем они очень важны при настройке графического интерфейса).

❐ Если вы собираетесь работать в сети (а UNIX вообще в первую очередь — сетевая ОС), то запишите следующие данные:

- фирма-производитель и номер модели сетевой карты;
- ваш IP-адрес;
- сетевое имя вашего компьютера;
- маска подсети;
- IP-адрес шлюза (gateway IP address);
- IP-адреса серверов (основного и резервных) доменных имен (DNS server);
- IP-адреса серверов WINS (Windows Internet Name Service);
- имя домена вашей организации.

О Тип и производитель звуковой карты и игровых контроллеров (если таковые имеются).

2.2. Предостережения и рекомендации

Перед тем, как приступить к экспериментам по установке Linux как второй ОС, очень рекомендую принять некоторые меры предосторожности. Дело в том, что вам, возможно, придется произвести переразбиение диска, изменить загрузочные записи и поэкспериментировать с загрузочными и конфигурационными файлами. Все эти действия далеко не безобидны, и результатом может стать то, что компьютер вообще перестанет загружаться. Сумеете ли вы восстановить после этого всю нужную вам информацию с

диска — это еще вопрос. Мой опыт говорит о том, что оказаться в такой ситуации очень легко.

Поэтому, во-первых, необходимо изготовить загрузочную или спасательную дискету для вашей старой системы (если вы это еще не сделали). Во-вторых, стоит сохранить всю ценную для вас информацию, наработанную в старой системе (сделать back-up всех ценных файлов). И, в-третьих, подготовить (разыскать, запасти) комплект установочных файлов для вашей старой системы.

И еще одна важная рекомендация: если что-то пошло не так, не паникуйте. Могу поделиться своим печальным опытом: когда я первый раз устанавливал Linux на компьютере, на котором стояла Windows NT, я быстро потерял возможность ее загружать, и, не разобравшись в ситуации, посчитал, что ничего другого не остается, как отформатировать диск и установить все заново. Теперь я понимаю, что мог бы все восстановить, если бы не принял поспешного решения. Поэтому могу сказать, что Вернер Альмесбергер прав, когда в своем руководстве по загрузчику LILO дает следующие советы оказавшимся в затруднительной ситуации.

- **Не паникуйте.** Если что-то не работает, попробуйте выяснить, что не так, перепроверьте свои предположения и только затем пытайтесь внести необходимые исправления.
- **Читайте документацию.** Особенно в тех случаях, когда система делает не то, что вы от нее ожидаете.

Можно добавить еще один общеизвестный совет.

Смотрите log-файлы, т. е. протоколы работы системы (забегая вперед, скажу, что искать их надо в каталоге /var/log).

Как уже было сказано, собственно процедуру установки ОС Linux я здесь не описываю, отсылая читателя к подробным руководствам. Однако несколько советов, касающихся тех решений, которые вы принимаете в ходе инсталляции, мне все же хотелось бы дать.

Во-первых, не спешите и внимательно читайте те сообщения, которые появляются на экране, а также вдумывайтесь в то, какие варианты вы выбираете из числа предлагаемых вам на появляющихся экранных формах. В подтверждение этого совета могу рассказать, что когда я ставил Red Hat 7.1, то автоматически давил на кнопку Next, считая, что предлагаемый по умолчанию вариант вполне приемлем. В результате, после завершения инсталляции я не смог достучаться до компьютера ни по одному из сетевых протоколов (Telnet, FTP, NFS, Samba), хотя вроде бы задавал работу компьютера в сети. Оказалось, что в варианте, предлагаемом по умолчанию, устанавливается firewall, который закрывает доступ из сети. Чтобы такой доступ открыть, надо в ходе инсталляции явно задать, какие сервисы оставить открытыми. Но мы же так спешим!

Во-вторых, я не рекомендую соглашаться с тем, что система при загрузке автоматически выходит в графический режим. В конце концов, набрать в командной строке startx вовсе не сложно, а справиться с настройками графической оболочки новичку вряд ли удастся, если с ее загрузкой что-либо не в порядке.

После того как необходимые меры предосторожности приняты, следует решить, каким образом вы будете осуществлять многовариантную загрузку, и подготовить диск к установке нескольких ОС, для чего разбить его на соответствующее число разделов (partition). Но прежде, чем перейти к конкретным процедурам подготовки диска, мне представляется необходимым хотя бы кратко рассмотреть устройство диска и процедуры загрузки ОС, чтобы не вклинивать теоретические вопросы и объяснения в изложение конкретных процедур. У кого не хватает терпения на чтение этих теорий, тот может пропустить их и перейти сразу к вопросу о выборе программы-загрузчика.

2.3. Разделы на диске и процесс загрузки

2.3.1. Что такое "геометрия диска"?

Как вы знаете, жесткие диски представляют собой несколько пластин с магнитным покрытием, расположенных на одной оси и вращающихся с большой скоростью. Считывание/запись информации осуществляется с помощью головок диска, расположенных одна под другой между пластинами и перемещающихся от центра к краям пластин. Окружность на магнитной пластине, которую описывает головка при вращении пластин, называется дорожкой, а совокупность таких дорожек, расположенных одна под другой (определяемая каждым фиксированным положением головок), называется цилиндром. Каждая дорожка разбита на сектора, и в сектор можно записать 512 байт полезной информации. Поэтому диски часто характеризуются совокупностью трех цифр: числом цилиндров/числом дорожек в цилиндре/числом секторов на дорожке или C/H/S (от первых букв соответствующих английских терминов: Cylinder/Head/Sector, т. е. цилиндр/головка/сектор). Эти три цифры называют "геометрией диска". Диск с геометрией C/H/S имеет объем C×H×S×512 байт.

Диски являются блочными устройствами, т. е. считывание и запись информации производятся блоками, и минимальный размер блока равен одному сектору (512 байт). Для того чтобы записать информацию на диск, надо "позиционировать головку", т. е. указать контроллеру, в какой сектор эту информацию записать. Сектора как раз адресуются путем указания номера цилиндра, номера считывающей головки (или дорожки) и порядкового номера сектора на дорожке.

2.3.2. Разделы диска и таблица разбиения диска

Физические диски в Intel-системах принято разбивать на разделы. Повелось это, кажется, из-за того, что первые версии MS-DOS не могли обеспечить доступ к большим дискам (а объемы дисков росли быстрее, чем возможности DOS). Тогда придумали разбиение дисков на разделы. Для этого в нулевой сектор диска (нулевой сектор первой дорожки на нулевом цилиндре) стали записывать так называемую таблицу разбиения диска на разделы (partition table). Каждый раздел может трактоваться как отдельный физический диск. В частности, в разные разделы могут быть установлены разные операционные системы.

Таблица разделов содержит 4 записи по 16 байтов для 4 разделов, которые называют первичными. Каждая запись имеет следующую структуру:

```
struct partition {
   char active;     /* 0x80: раздел активный (загрузочный),
                           0: не активный */
   char begin[3];   /* CHS первого сектора, 24 бита */
   char type;       /* тип раздела (например, 83 — LINUX_NATIVE,
                    /* 82 - LINUX_SWAP, 85 - LINUX_EXTENDED) */
   char end[3];     /* CHS последнего сектора, 24 бита */
   int start;       /* номер начального сектора
                           (32-бита, счет начинается с 0) */
   int length;      /* число секторов в разделе (32 бита) */
};
```

Таблица разделов диска создается обычно с помощью программы fdisk. В ОС Linux имеется как стандартная программа fdisk (которая, впрочем, существенно отличается от программы fdisk в MS-DOS и Windows), так и еще две программы для работы с разделами диска: cfdisk и sfdisk. Программа cfdisk, как и fdisk, предназначена для работы с таблицей разделов диска: она не обращает никакого внимания на информацию, которая уже имеется на диске. Отличается она только несколько более удобным интерфейсом, предоставляющим пользователю не просто подсказку по командам, а систему меню. Программа sfdisk обладает несколько более широкими возможностями, в частности, она позволяет произвести некоторые операции над существующими разделами диска.

DOS использует поля begin и end таблицы разбиения диска и функции прерывания 13 BIOS (Int 13h) для доступа к диску, и поэтому не может использовать диски объемом более 8,4 Гбайт, даже с новым BIOS (об этом будет рассказано ниже), а разделы не могут быть более 2,1 Гбайт (но это уже из-за ограничений файловой системы FAT 16).

Linux использует только поля start и length таблицы разбиения диска и поддерживает разделы, содержащие до 2^{32} секторов, т. е. размер раздела может достигать 2 Тбайт.

Поскольку в таблице разбиения отведено только 4 строки для задания разделов, число первичных разделов на диске с самого начала ограничено: их может быть не более 4. Когда стало ясно, что и 4-х разделов мало, были изобретены логические разделы. Для этого один из первичных разделов объявляется "расширенным" (тип раздела — 5, или F, или 85 в шестнадцатеричной системе), и в нем создаются "логические разделы". Расширенные разделы сами по себе не используются, они могут лишь хранить логические разделы. Первый сектор расширенного раздела хранит таблицу разделов с четырьмя входами: один используется для логического раздела, другой для еще одного расширенного раздела, а два не используются. Каждый расширенный раздел имеет свою таблицу разбиения, в которой, как и в первичном расширенном разделе, используются только две строки, задающие один логический и один расширенный раздел. Таким образом, получается цепочка из таблиц разделов, где первая описывает три основных раздела, а каждая следующая — один логический раздел и положение следующей таблицы.

Программа sfdisk в Linux показывает всю цепочку:

```
[root]# sfdisk -l -x /dev/hda

Disk /dev/hda: 784 cylinders, 255 heads, 63 sectors/track
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

Device     Boot    Start     End     #cyls    #blocks    Id  System
/dev/hda1  *         0+      189      190-    1526143+    6  FAT16
/dev/hda2          190       783       594    4771305     5  Extended
/dev/hda3            0         -         0          0     0  Empty
/dev/hda4            0         -         0          0     0  Empty

/dev/hda5          190+      380      191-    1534176     6  FAT16
-                  381       783       403    3237097+    5  Extended
-                  190       189         0          0     0  Empty
-                  190       189         0          0     0  Empty

/dev/hda6          381+      783      403-    3237066     7  HPFS/NTFS
-                  381       380         0          0     0  Empty
-                  381       380         0          0     0  Empty
-                  381       380         0          0     0  Empty
```

Число логических разделов в принципе не ограничено, потому что каждый логический раздел может содержать таблицу разделов и вложенные логические разделы. Однако реально ограничения все же существуют, например, Linux может работать не более чем с 15 разделами на SCSI-дисках и не более чем с 63-мя разделами на IDE-дисках.

Расширенный раздел как на физическом диске, так и в расширенном разделе вложенного расширенного раздела (предыдущего уровня) может быть только один: ни одна из существующих программ разбиения дисков (fdisk и ее усовершенствованные аналоги) не умеет создавать более одного расширенного раздела.

В Linux диск в целом (то есть физический диск) доступен по имени устройства /dev/hda, /dev/hdb, /dev/sda и т. п. Первичные разделы обозначаются дополнительной цифрой в имени устройства: /dev/hda1, /dev/hda2, /dev/hda3, /dev/hda4, а логические разделы в Linux доступны по именам /dev/hda5, /dev/hda6... (начиная с номера 5). Из сказанного выше должно быть ясно, почему могут быть пропущены имена /dev/hda3 и /dev/hda4 (третий и четвертый первичные разделы просто не были созданы) и сразу после /dev/hda2 вы увидите /dev/hda5 (логический раздел в расширенном разделе /dev/hda2), а далее нумерация идет последовательно.

В Windows логические разделы получают однобуквенные имена, начиная с последнего задействованного имени первичного раздела. Если, например, имеется один жесткий диск с двумя простыми первичными разделами (C: и D:) и одним расширенным разделом, в котором созданы два логических раздела, то эти логические разделы именуются E: и F:. Впрочем, в Windows NT и 2000 с помощью администратора дисков разделам могут быть присвоены другие буквенные имена.

2.3.3. Процесс загрузки ОС фирмы Microsoft

Какую бы операционную систему мы ни рассматривали, для того, чтобы ОС могла начать управлять компьютером, ее необходимо загрузить в оперативную память. Поэтому давайте кратко рассмотрим, как происходит процесс загрузки разных ОС. Поскольку нас интересует только загрузка с жестких дисков, то мы не будем рассматривать особенности загрузки с дискеты, CD-ROM и по сети. Начнем с доброй старой MS-DOS и MS Windows (не забывайте, что разработка и совершенствование персональных компьютеров шли параллельно с развитием ОС от Microsoft, и решения, использованные в этих ОС, оказывали сильное влияние на те решения, которые принимались разработчиками аппаратных средств).

Как вы знаете, при включении компьютера вначале запускается программа POST (Power On Self Test). Она определяет количество доступной памяти, тестирует ее, определяет наличие других компонентов (клавиатура, винче-

стер и т. д.), инициализирует карты адаптеров. На экране обычно появляются сообщения о количестве памяти, о ее тестировании, перечень обнаруженных устройств (гибкие и жесткие диски, процессор, COM-порты и т. д.).

После завершения тестирования POST вызывает Int 19h, которое пытается найти загрузочное устройство. Поиск производится в том порядке, который определен в Setup BIOS, и осуществляется путем опроса нулевых секторов соответствующих устройств. Если диск является загрузочным, то в его нулевом секторе находится главная загрузочная запись — Master Boot Record (MBR). Последние два байта MBR — это "магическое число", которое является признаком того, что данный сектор есть MBR, а, следовательно, диск является загрузочным. Кроме "магического числа" MBR содержит таблицу разделов диска, о которой уже было сказано выше, и маленькую программу — первичный загрузчик, объемом всего 446 (0x1BE) байт.

В табл. 2.1 представлена структура главного загрузочного сектора, создаваемого при инсталляции Windows.

Таблица 2.1. *Структура главного загрузочного сектора*

Смещение	Содержание
0x000	Код первичного загрузчика
0x1BE	Таблица разбиения диска
0x1FE	"Магическое число" (OxAA55)

MS-DOS, Windows 95 и NT записывают DOS MBR при инсталляции. Стандартное для MS содержимое MBR можно также записать командой fdisk /mbr.

Но вернемся к описанию процесса загрузки. Прерывание 19h BIOS загружает первичный загрузчик в память компьютера и передает управление этой программе. Но такой маленькой программе не под силу загрузить ОС; все, что она может сделать — это загрузить в память более мощную программу — вторичный загрузчик.

Для этого она ищет в таблице разделов активный раздел и считывает в память вторичный загрузчик, который располагается начиная с первого логического сектора активного раздела. Обратите внимание на слово "начиная". Дело в том, что вторичный загрузчик в разных системах имеет разную длину.

В разделе, отформатированном под файловую систему FAT, вторичный загрузчик занимает один сектор (512 байт). В разделе, отформатированном под файловую систему NTFS, вторичный загрузчик занимает уже несколько секторов.

Вторичный загрузчик загружает первый слой программ, необходимых для запуска операционной системы. В случае MS-DOS программа-загрузчик загружает IO.SYS по адресу 700h, затем MSDOS.SYS и передает управление разделу SYSINIT модуля IO.SYS.

Если по каким-либо причинам на диске не найден активный раздел, процесс загрузки продолжается обработкой прерывания 18h. Эта ветвь реально используется очень редко, но такая возможность может быть очень полезна в некоторых ситуациях. При удаленной загрузке, когда операционная система загружается с сервера, это прерывание перенаправляется программой POST на ROM сетевой карты.

Для других ОС от Microsoft процесс загрузки происходит аналогично.

- Windows 95 загружается так же, как и DOS, но заменяет IO.SYS и MSDOS.SYS своими файлами. DOS сохраняются в файлах IO.DOS и MSDOS.DOS соответственно. Когда вы выбираете загрузку сохраненного DOS, Windows 95 переименовывает свои файлы в файлы с расширением w40 и восстанавливает первоначальные имена системных файлов DOS. Процесс продолжается с загрузки IO.SYS. Таким образом, загрузочные сектора DOS и Windows 95 одинаковые.
- Windows NT4 использует MBR DOS, но заменяет загрузочную запись активного раздела таким образом, что вместо IO.SYS загружается NTLDR. Это уже мощная программа, которая многое может сделать. В частности, она находит файл boot.ini и, если параметр timeout больше 0, предлагает меню загрузки.

Каждая строка секции [operating systems] файла boot.ini определяет один из вариантов загрузки и строится по следующему шаблону

```
адрес_вторичного_загрузчика="название_варианта"
```

Адресом вторичного загрузчика может являться указание на конкретный раздел диска или на файл загрузчика. Вот пример файла boot.ini:

```
[operating systems]
multi(0)disk(0)rdisk(0)partition(3)\WINNT="Windows NT Workstation 4.00 RUS"
multi(0)disk(0)rdisk(0)partition(3)\WINNT="Windows NT Workstation 4.00 RUS [VGA mode]" /basevideo /sos
C:\="Microsoft Windows"
C:\BOOTSECT.LNX="Linux"
```

Если пользователь выбирает NT, то выполняется загрузка по адресу раздела, указанному в первой строке раздела. В строке, соответствующей Microsoft Windows, указан просто диск "C:\", т. к. имя загрузочного файла берется по умолчанию: bootsect.dos. Файл грузится в память и загрузка продолжается так, как если бы загрузочная запись раздела была загружена программным кодом из MBR.

Для загрузки других систем можно воспользоваться таким же приемом. Для этого в boot.ini нужно добавить строки, содержащие ссылки на другие загрузочные файлы. При выборе такой строки будет загружаться соответствующая ОС. В приведенном выше примере этим способом обеспечивается загрузка Linux. Для этого в файле C:\BOOTSECT.LNX должно быть предварительно записано содержимое загрузочной записи, создаваемой Linux (точнее — LILO, стандартным загрузчиком Linux).

2.3.4. Проблемы с большими дисками

В MS-DOS и первых версиях Windows доступ к дискам был организован через прерывание 13 (Int 13h) BIOS (в том числе на этапе начальной загрузки ОС). При этом использовалась адресация секторов на диске на основе указания номеров цилиндра, головки и сектора на дорожке (C/H/S). Точнее:

- AH — выбор операции;
- CH — младшие 8 битов номера цилиндра;
- CL — биты 7—6 соответствуют старшим битам номера цилиндра, биты 5—0 соответствуют номеру сектора;
- DH — номер считывающей головки;
- DL — номер диска (80h или 81h).

(Заметим в скобках, что нумерацию физических цилиндров и дорожек принято начинать с 0, а сектора на дорожке нумеруют, начиная с 1). Однако практически головок было не более 16-ти, а число секторов на дорожке — не более 63, и хотя для указания цилиндра использовалось 10 битов, все равно BIOS не мог работать с дисками объемом более 1024x63x16x512 = 528 Мбайт.

Для преодоления этого ограничения стали применять разные хитрые приемы (подробнее об этом вы можете узнать из *[П4.2] приложения)*. Например, Extended CHS (ECHS) или "Large disk support" (иногда обозначается просто как "Large") использует еще три незанятых бита номера головки для увеличения числа адресуемых цилиндров. Это позволило использовать "фальшивую геометрию диска" в 1024 цилиндра, 128 считывающих головок и 63 сектора на дорожку. Трансляцию Extended CHS в реальный CHS-адрес (который может иметь до 8192 цилиндров) осуществляет BIOS. Это позволяет работать с дисками объемом до 8192×16×63×512 = 4 227 858 432 байт или 4,2 Гбайт.

Но разработчики все увеличивали плотность записи на диск, число пластин и дорожек, изобретали другие способы увеличения объема дисков. В частности, число секторов на дорожках стало разным (на более длинных дорожках, расположенных ближе к краю пластин, число секторов стали увеличивать). В результате три числа C/H/S уже перестали правильно отражать "геометрию диска", а старые версии BIOS перестали обеспечивать доступ ко всему дисковому пространству.

Тогда придумали другой прием для работы с большими дисками через Int 13h — линейную адресацию блоков ("Linear Block Addressing" или LBA). Если не вдаваться в подробности, то можно сказать, что все сектора на диске нумеруются последовательно, начиная с первого сектора на нулевой дорожке нулевого цилиндра. Вместо CHS-адреса каждый сектор получает логический адрес — просто его порядковый номер в общем массиве секторов. Нумерация логических секторов начинается с нуля, причем нулевой сектор содержит главную загрузочную запись (MBR). В Setup BIOS поддержка преобразования линейного номера в CHS-адрес обозначается как "поддержка LBA". Таким образом, в новых версиях BIOS обычно имеется выбор из трех вариантов: "Large", "LBA" и "Normal" (последнее означает, что трансляция адресов не производится).

Но и в режиме LBA обращение к физическому диску все равно осуществляется через функции Int 13h, которые используют 3D-нотацию (C,H,S). В силу этого возникает ограничение на возможный объем диска: BIOS, и, следовательно, MS-DOS и ранние версии Windows, не могли адресовать диски объемом более 8,4 Гбайт.

Надо заметить, что указанное ограничение относится только к дискам с интерфейсом IDE. В контроллерах SCSI-дисков номер сектора переводится в команды SCSI, а далее сам диск находит нужную позицию, поэтому такого ограничения на объем диска не возникает.

Еще раз хочется отметить, что все перечисленные ограничения существенны только на этапе загрузки ОС, поскольку сама Linux и последние версии Windows при работе с дисками уже не используют прерывание 13 BIOS, а используют собственные драйверы для работы с дисками. Но, прежде чем система сможет использовать собственный драйвер, она должна как минимум его загрузить. Поэтому на этапе начальной загрузки любая система вынуждена пользоваться BIOS. Это и вызывает ограничения на размещение многих систем за пределами 8 Гбайт, они не могут оттуда загружаться, хотя после успешной загрузки могут работать с дисками гораздо большего объема. Для того чтобы понять, как можно обойти эти ограничения, нам потребуются некоторые знания о том, как происходит загрузка ОС Linux.

2.4. Выбор загрузчика

Прежде чем приступать к установке второй (третьей и т. д.) ОС, надо выбрать способ организации выбора ОС на этапе загрузки компьютера. Эту задачу решают программы-загрузчики. Существует несколько программ такого рода. Раз уж речь у нас идет о Linux, то первым делом надо упомянуть программу LILO, которая входит в состав любого дистрибутива Linux.

2.4.1. Загрузчик LILO из дистрибутива ОС Linux

Загрузчик LILO создан Вернером Альмесбергером (Werner Almesberger). LILO может загружать ядро Linux как с дискеты, так и с жесткого диска, а также может загружать другие операционные системы: PC/MS-DOS, DR DOS, OS/2, Windows 95/98, Windows NT, 386BSD, SCO UNIX, UNIXware и т. д. Может быть задан выбор до 16 разных операционных систем на этапе загрузки.

LILO представляет собой комплект из нескольких программ: собственно загрузчика, программ, используемых для установки и настройки загрузчика, и служебных файлов:

- программа /sbin/lilo, которая запускается из-под Linux, служит для того, чтобы записать всю информацию, необходимую на этапе загрузки, в соответствующие места. Ее необходимо перезапускать каждый раз после внесения изменений в ядро или в конфигурационный файл LILO;
- различные служебные файлы, которые нужны LILO во время загрузки. Эти файлы обычно располагаются в каталоге /boot. Самые важные из них — это собственно загрузчик (смотри ниже) и map-файл (/boot/map), в котором указывается местоположение ядра. Еще один важный файл — это файл конфигурации LILO, который обычно имеет имя /etc/lilo.conf;
- *П* собственно загрузчик — это та часть LILO, которая первой загружается в память через прерывание BIOS, и которая загружает ядро Linux или загрузочный сектор другой операционной системы. Загрузчик тоже состоит из двух частей. Первая часть записывается в загрузочный сектор и служит для загрузки второй части, которая значительно больше по размеру. Обе части обычно хранятся на диске в файле /boot/boot.b.

Надо иметь в виду, что формат загрузочного сектора, создаваемого LILO, отличается от формата DOS MBR, так что если записать загрузочный сектор LILO в MBR, то ранее установленные системы от Microsoft перестанут загружаться (если не принять дополнительных мер).

Загрузочный сектор LILO спроектирован так, чтобы его можно было использовать как загрузочный сектор раздела, в частности, в нем есть место для таблицы разделов.

Загрузочный сектор LILO при инсталляции системы можно разместить в следующих местах:

- П загрузочный сектор дискеты в формате Linux (/dev/fd0,...);
- П MBR первого жесткого диска (/dev/hda, /dev/sda,...);
- О загрузочный сектор первичного раздела файловой системы Linux на первом жестком диске (/dev/hda1, /dev/hda2,...);
- П загрузочный сектор логического раздела в расширенном разделе первого жесткого диска (/dev/hda5,...). Правда большинство программ типа `fdisk`

не предполагают, что можно загружаться из расширенного раздела и отказываются объявлять его активным. Поэтому в состав LILO включена специальная программа (activate), которая позволяет обойти это ограничение. Но программа fdisk из дистрибутива Linux поддерживает возможность активизации расширенного раздела. Для этого надо использовать либо опцию `-b`, либо переменную BOOT.

Загрузочный сектор LILO не может быть размещен в следующих местах:

- загрузочный сектор дискеты или первичного раздела, отформатированных в других файловых системах;
- в swap-разделе Linux;
- на втором жестком диске.

Кроме того, имейте в виду, что LILO во время загрузки нужны еще следующие файлы:

- /boot/boot.b;
- /boot/map (создается при запуске /sbin/lilo);
- все загружаемые версии ядра (если вы выбираете версию ядра на этапе загрузки);
- загрузочные сектора других операционных систем, которые будут загружаться через LILO;
- выдаваемые при загрузке сообщения (если таковые определены).

Следовательно, как загрузочный сектор LILO, так и перечисленные файлы (в том числе те, которые вы будете устанавливать впоследствии) должны находиться в пределах первых 1024 цилиндров на жестком диске, т. к. они должны быть доступны через BIOS.

Начиная с версии 21, LILO выводит на экран меню выбора загружаемой системы (раньше надо было для вызова меню нажимать клавишу табуляции).

2.4.2. Другие загрузчики ОС

Кроме LILO для загрузки Linux можно использовать и другие загрузчики.

- Если у вас до установки Linux уже стояла ОС Windows NT, то вторым доступным для вас загрузчиком является OS Loader от NT. По сравнению с LILO загрузчик OS Loader имеет, по крайней мере, два преимущества. Во-первых, сохраняется вся старая конфигурация (у меня уже была возможность загружать по выбору Windows NT или Windows 95), и, во-вторых, можно установить Linux на диск, который не может быть загрузочным в Linux, например, второй диск на втором контроллере (Secondary Slave).
- Если у вас была установлена только ОС Windows 95 или Windows 98 и не было Windows NT или Windows 2000, то OS Loader у вас не установлен, и

если вы почему-либо не хотите ставить LILO, можно воспользоваться программой-загрузчиком loadlin.exe (обычно поставляется вместе с дистрибутивом Linux).

- В состав некоторых дистрибутивов Linux в последнее время включается программа-загрузчик GRUB.
- В составе дистрибутива OS/2 фирмы IBM имеется программа-загрузчик, которая называется Boot Manager. Во многих руководствах ее рекомендуют использовать для организации многовариантной загрузки.
- В разных источниках упоминается также программа System Commander, которая тоже является многовариантным загрузчиком.
- Еще один многовариантный загрузчик входит в состав пакета PartitionMagic фирмы Power Quest, о котором мы будем говорить в следующем подразделе.

Кроме перечисленных я встречал упоминания еще о ряде загрузчиков (часть из которых можно найти в каталоге /public/ftp/pub/Linux/system/boot/loaders на сайте **ftp://metalab.unc.edu).** Но, поскольку я этими программами не пользовался, то рассказать о них подробнее не могу, и все мои последующие рекомендации будут основаны на использовании загрузчиков LILO, NT Loader и loadlin.exe. Если вы хотите установить другую программу-загрузчик, то вы должны почитать руководство по ее установке и использованию.

2.4.3. Варианты загрузки

Итак, на мой взгляд, выбор варианта загрузки производится следующим образом.

- Если у вас установлена Windows NT или Windows 2000, то используйте NT Loader.
- Если у вас стоит Windows *95* или Windows 98 на FAT16, и вы не хотите ставить программу-загрузчик из другой ОС или от независимого разработчика, то можете либо использовать LILO, либо сначала запускать DOS, а затем загружать Linux с помощью loadlin.exe (или другой аналогичной программы, их существует несколько, но другие мы рассматривать не будем).
- Если у вас установлена Windows 95 OSR2 или Windows 98 на FAT32, и вы не хотите ставить программу-загрузчик из другой ОС или от независимого разработчика, то вы должны будете использовать loadlin.exe. Многие руководства HOWTO утверждают, что не нужно использовать LILO, если активный раздел у вас форматирован в системе FAT32, хотя причины этого мне неизвестны. Но моя собственная попытка загружать Linux через NT Loader, установленный в FAT32-разделе, окончилась неудачей. Так что мне тоже пришлось в этом случае воспользоваться программой

loadlin.exe, которая с успехом справилась с задачей, и при этом вообще произвела на меня самое благоприятное впечатление, так что я рекомендую вам применять ее для загрузки Linux.

В следующих разделах я расскажу о том, как установить Linux, используя все три варианта загрузки: через загрузчик NT Loader *(см. разд. 2.6),* загрузчик LILO *(см. разд. 2.7) и* загрузчик loadlin.exe *(см. разд. 2.8).* Но до установки загрузчика надо подготовить разделы на диске (или, по крайней мере, продумать, как их организовать).

2.5. Подготовка разделов на диске

2.5.1. Рекомендации по созданию разделов

Рекомендации тут давать довольно сложно, т. к. во многом это зависит от воли и потребностей хозяина диска. Но все же попробую сформулировать некоторые предложения. При этом диски и разделы буду именовать так, как это принято в Linux, т. е. /dev/hda, /dev/hdb и т. д. для дисков, и /dev/hda1, /dev/hda2 и т. д. — для разделов на диске.

Разбивать диск на разделы необходимо потому, что Windows и Linux используют разные способы организации хранения информации на диске и разные способы организации доступа к этой информации. Поэтому лучше всего каждой операционной системе выделить на диске отдельный раздел (или даже несколько, как мы увидим ниже).

Давайте вначале рассмотрим простой случай — когда объем вашего диска не превышает 8,4 Гбайт (точнее — когда число цилиндров не превышает 1024). В этом случае все просто: вы просто делите диск пропорционально тому, сколько места требуется для установки каждой из операционных систем, которые вы хотите установить. Можете воспользоваться следующими данными о размерах дискового пространства, минимально необходимого для установки операционных систем в стандартной конфигурации (табл. 2.2).

Таблица 2.2. *Требования ОС к дисковому пространству*

Операционная система	Требует
Windows 95	100 Мбайт
Windows 98	200 Мбайт
Windows NT	200 Мбайт
Windows 2000	700 Мбайт
Linux Red Hat 6.2 (в режиме рабочей станции с KDE)	700 Мбайт

Однако помните, что надо учесть не только объем файлов самой операционной системы, но и того программного обеспечения, которое вы планируете в ней запускать, а также оставить существенный резерв для того программного обеспечения, которое вы захотите установить в последующем (это неизбежно!). Учтите, что те 700 Мбайт, которые указаны для Linux в приведенной выше таблице, включают место для всего ПО, которое устанавливается вместе с Linux по умолчанию, в том числе, например, мощный текстовый процессор Lyx. Оценки же, которые даны для Windows, касаются только самой ОС. Если, например, вместе с Windows 2000 установить MS Office 2000 в стандартной конфигурации, то места на диске потребуется много более гигабайта.

Судя по моему опыту, для нормальной работы с Windows 95/98, Windows NT и Linux вполне достаточно выделить разделы объемом 800—1000 Мбайт (конечно, если вы не ставите громоздких программных пакетов, вроде CorelDRAW), а вот для Windows 2000 требуется уже побольше.

Теперь рассмотрим вопрос о выделении разделов для Linux. Тут одним разделом не обойтись. Во-первых, надо выделить отдельный раздел подкачки (swap-раздел) для Linux. При планировании объема swap-раздела Linux учтите следующее.

- ❒ В Linux RAM и пространство swap складываются, образуя общую виртуальную память. Например, если у вас 8 Мбайт ОЗУ (RAM) и 12 Мбайт swap-пространства, вы имеете 20 Мбайт виртуальной памяти.
- О Для работы Linux надо иметь, по крайней мере, 16 Мбайт виртуальной памяти, так что при 4 Мбайт ОЗУ вы должны выделить под swap не менее 12 Мбайт.
- ❒ В Linux размер одного swap-раздела не может превышать 128 Мбайт. То есть раздел под swap вы можете выделить произвольного размера, но Linux не сможет использовать более 128 Мбайт. Если вы хотите иметь виртуальной памяти больше, надо создавать два swap-раздела или использовать файл подкачки.
- П Рассчитывая размер пространства свопинга, имейте в виду, что слишком большое количество этого пространства может оказаться вовсе бесполезным. На компьютере с 16 Мбайт ОЗУ при стандартной конфигурации Linux и стандартном наборе ПО вполне достаточно иметь 48 Мбайт пространства свопинга, а для минимальной конфигурации Linux можно обойтись вообще без swap-пространства. Конечно, точное значение этого параметра существенно зависит от того набора приложений, которое будет у вас установлено.

В общем, долгие размышления по поводу объема swap-раздела нужны только в том случае, когда у вас маленький диск и мало ОП. В противном случае для начала задайте размер swap-раздела таким образом, чтобы объем вирту-

альной памяти был не менее 128 Мбайт. А если у вас больше 128 Мбайт оперативной памяти, то этот раздел вообще может оказаться ненужным.

Все остальные части Linux и работающее под ней программное обеспечение, в принципе, могут размещаться в одном разделе. Однако имеет смысл подумать о том, чтобы разместить файловую систему Linux в нескольких отдельных разделах. А. Федорчук, например, рекомендует выделить для файловой системы Linux три раздела. Первый из них (на мой взгляд, под него достаточно отвести раздел размером в один гигабайт) будет содержать корневую файловую систему (/). Второй раздел отводим для каталога /home. И третий раздел монтируется как каталог /usr. Такое разбиение обосновывается следующими соображениями. Как бы ни была устойчива и надежна ОС Linux, иногда возникает необходимость переустановить ее. Например, вы решили обновить версию дистрибутива или по неопытности разрушили жизненно важные для системы файлы. Если все установлено в один раздел, вы при переустановке теряете все, что наработали и хранили в своем домашнем каталоге. Кроме того, потеряны будут и установки программных пакетов, которые вы компилировали из исходных кодов, или другим способом устанавливали уже после установки системы. Большая часть таких пакетов по умолчанию инсталлируется в каталог /usr. Если же отвести для этих каталогов отдельные разделы на диске и при переустановке системы не форматировать эти разделы, все наработанное можно будет (может быть после небольших дополнительных настроек) сохранить и использовать после переустановки самой ОС. В разрабатываемом сейчас стандарте на файловую систему Linux (подробнее об этом будет рассказано в *гл. 4)* тоже имеется рекомендация о размещении каталога /usr в отдельном разделе диска.

Мне кажется, что этих рекомендаций вполне достаточно для того, чтобы спланировать разбиение в случае одного небольшого диска. Рассмотрим теперь случай диска с числом цилиндров более 1024.

Из того, что было сказано в предыдущих разделах, следует, что программы-загрузчики должны располагаться в пределах первых 1024 цилиндров. Между прочим, NT Loader может располагаться не обязательно в NTFS-разделе, как и вообще не в том разделе, где расположены остальные файлы ОС. Как сказано выше, для Linux тоже можно расположить корневой каталог вместе с подкаталогом /boot в "нижних" цилиндрах, а остальное — где угодно.

Поэтому в этом случае мои предложения сводятся к следующему:

- загрузочные части всех систем от Microsoft поместить в первый первичный раздел диска, который отформатировать в системе FAT16 (в DOS);
- следующий первичный раздел выделить для корневого каталога Linux (/), размер которого сделать равным примерно 1 Гбайт;
- выделить swap-раздел для Linux (рекомендации о его размерах даны выше);
- все остальное дисковое пространство сделать расширенным разделом;

- в расширенном разделе создать логические разделы для каждой из устанавливаемых ОС: Windows 98, Windows NT или 2000, а также для файловых систем /home и /usr ОС Linux (в /home будут располагаться личные файлы пользователей, а в /usr устанавливаются все приложения).

Конечно, если у вас стоит только Windiws 95 с FAT16, то можете оставить ее в первом разделе. Если же у вас была установлена Windows NT или FAT32, то наличие небольшого раздела с FAT16 будет не лишним. Во-первых, даже в случае любого краха системы вы всегда сможете загрузиться с загрузочной дискеты DOS и хотя бы увидеть, что жесткий диск работоспособен (в принципе). А во-вторых, файловая система FAT16 видна из-под любой ОС, в том числе Linux, так что этот раздел может служить для обмена файлами между разными системами. Но делать этот раздел большим не стоит — FAT16 очень нерационально использует дисковое пространство. Поэтому отведите под него, скажем, 256 или 512 Мбайт.

Эти рекомендации формулировались в предположении, что у вас всего один жесткий диск. Если у вас их 2, то все остается в силе, разве что swap-раздел Linux лучше расположить не на том физическом диске, где расположены остальные разделы, отведенные Linux. Говорят, что от этого повышается быстродействие в Linux (оно и понятно, считывающие головки меньше бегают).

2.5.2. Программы для разбиения диска

После того как план разбиения составлен, осталось подобрать инструмент, с помощью которого это разбиение можно осуществить на практике. Наиболее известной программой разбиения диска является уже упоминавшаяся программа fdisk, варианты которой имеются во всех операционных системах. И ничего другого, может быть, и не требовалось бы, если бы речь шла о разбиении девственно чистого диска. Но мы рассматриваем случай, когда какая-то ОС на диске уже имеется и надо обеспечить переразбиение диска без потери информации на нем. Программа fdisk для таких операций не подходит.

В составе дистрибутивов Red Hat и BlackCat (вероятно, и в других тоже) имеется программа fips, которая служит для переразбиения диска. Однако отзывы, которые я слышал, не воодушевили меня на использование этой программы. Поэтому мой вам совет — если хотите переразбить диск без потери информации, найдите программу Partition Magic фирмы Power Quest (www.powerquest.com) и воспользуйтесь ею.

Во-первых, она позволяет произвести переразбиение диска без потери информации (то есть все ваши предыдущие установки и настройки будут сохранены). При этом можно не только создать новый раздел на свободном месте на диске, но и как угодно переместить ранее существующие разделы.

Во-вторых, эта программа (даже в варианте для DOS) предоставляет вам удобный графический интерфейс, так что все осуществляемые действия вы

воспринимаете наглядно. Никакого сравнения с интерфейсом командной строки программы fdisk. Я пользовался 5-ой версией этой программы. Она отказывается работать под Windows 2000 и Windows NT, однако в ее дистрибутиве имеется возможность изготовить две дискеты, которые служат для загрузки компьютера в режиме DOS (вариант Caldera DR-DOS) и последующего запуска программы. Эти дискеты прекрасно позволяют переразбить и диск, на котором установлена ОС Windows NT или 2000.

При создании разделов необходимо следить за тем, чтобы границы разделов не пересекались.

Я думаю, что приведенных данных достаточно для того, чтобы спланировать и осуществить разбиение диска на разделы. Поэтому перейдем к рассмотрению конкретных вариантов установки двух ОС на одном компьютере.

2.6. Windows NT и Linux: загрузка через OS Loader от NT

В этом разделе, говоря о Windows NT, я все время буду иметь в виду и Windows 2000, т. к. в части установки Linux между ними различий нет. Будем предполагать, что Windows NT установлена в разделе /dev/hda2 (как вы помните, /dev/hda1 предлагалось отвести под FAT16-раздел). Раз ОС Windows NT на вашем компьютере уже установлена, значит и загрузчик OS Loader тоже уже установлен, а, следовательно, вполне логично использовать его и для загрузки Linux. Я надеюсь, что вы сделали резервную копию всей ценной для вас информации с жесткого диска(ов). После этого можно перейти к установке Linux. Этапы установки можно описать следующим образом.

1. Если вы не сделали этого ранее, то до начала каких-либо действий по инсталляции Linux изготовьте загрузочные дискеты для загрузки и восстановления Windows NT. Для создания загрузочной дискеты достаточно скопировать на отформатированную дискету файлы ntldr, ntdetect.com и boot.ini из корневого каталога загрузочного диска. Программу создания дискеты аварийного восстановления для Windows 2000 можно запустить из панели управления (команда **Архивация данных),** а в Windows NT 4.0 я ее сумел найти только с помощью поиска в справочной системе (ищите "Создание диска аварийного восстановления").
2. С помощью программы Partition Magic освободите часть дискового пространства и создайте на свободном месте раздел типа ext2 (файловая система Linux) и swap-раздел. О том, что надо при этом учесть, было рассказано выше.

3. Проведите процедуру инсталляции Linux, следуя рекомендациям, прилагаемым к имеющемуся у вас дистрибутиву. В ходе процедуры инсталляции учтите следующее:

   - во-первых, в процессе установки надо обязательно изготовить загрузочную дискету, для чего при запросе о создании загрузочного диска выбрать команду **Yes, make a BOOT DISK** (Создать загрузочный диск). Этот диск понадобится вам на одном из следующих этапов. Кроме того, в последующем можно будет просто использовать эту дискету для загрузки Linux. Это тоже вариант загрузки, тем более что, в отличие от загрузочной дискеты DOS, после загрузки система уже о дискете не вспоминает, ее можно убрать из дисковода (даже размонтировать не требуется) и использовать дисковод для работы с обычными дискетами. Однако этот метод все же не совсем удобен, так что я не предлагаю вам использовать его постоянно. Только как запасной! Но это потом, а при настройке многовариантной загрузки он будет просто *необходим;*
   - во-вторых, при установке Linux надо установить LILO не в главный загрузочный сектор диска (Master Boot Record), а в первый сектор того раздела, который вы отвели для корневого раздела Linux. Для определенности предположим, что Linux устанавливается в первый сектор раздела /dev/hda3.

   В принципе, если вы установите LILO в MBR, то и это не смертельно, конечный результат (загрузка через NT Loader) может быть достигнут и в этом случае, но потребуется больше усилий. Дело в том, что формат MBR, создаваемого LILO и Windows (DOS), различен. Поэтому если вы поставите LILO в MBR, вам придется потом восстановить MBR от Windows. На всякий случай я позже расскажу, как это делается, но лучше будет, если вы сразу поставите LILO в первый сектор его собственного раздела.

4. После завершения инсталляции загрузите Linux с помощью загрузочной дискеты (если вы установили LILO в раздел Linux и не трогали MBR, то другой возможности загрузить Linux у вас пока нет).

5. Скопируйте загрузочный сектор Linux в файл; он понадобится для того, чтобы загрузчик Windows NT/2000 мог запускать Linux. Для этого надо смонтировать чистую дискету, например, так:

   ```
   [root]# mount -t vfat /dev/fda1 /mnt/floppy
   ```

   перейти в каталог /mnt/floppy

   ```
   [root]# cd /mnt/floppy
   ```

   и выполнить команду

   ```
   [root]# dd if=/dev/hda3 of=/mnt/floppy/bootsect.lnx bs=512 count=1
   ```

   которая позволяет записать содержимое загрузочного сектора диска /dev/hda3 в файл /mnt/floppy/bootsect.lnx.

Примечание

Хочу заметить, что если диск C: (/dev/hda1) отформатирован в системе FAT, то можно сразу создать файл bootsect.lnx в корневом каталоге диска C:. Отмечу, что я не знаю (не пробовал), можно ли будет загружаться без дискеты, если первый раздел загрузочного диска отформатирован в NTFS. Но думаю, что проблем и здесь не будет, кроме необходимости переноса boot-сектора через дискету, поскольку пока что поставляемые в дистрибутиве варианты ядра не поддерживают NTFS.

6. Далее необходимо перезагрузиться, чтобы запустить Windows NT, для чего даем в Linux команду:

```
[root]# shutdown -h now
```

Поскольку главная загрузочная запись не была изменена, должна загрузиться Windows NT. После завершения загрузки необходимо перенести файл /mnt/floppy/bootsect.lnx в корневой каталог диска C:, точнее — в корневой каталог того раздела, с которого загружается Windows NT. В зависимости от того, как вы устанавливали Windows NT, это может быть как FAT16-раздел, так и NTFS-раздел. Признаком нужного раздела является наличие в нем файлов ntldr и boot.ini (эти файлы могут быть скрытыми!). Файлу bootsect.lnx можно присвоить атрибут read-only.

7. После завершения загрузки NT найдите файл boot.ini в корневом каталоге и добавьте в него следующую строчку:

```
C:\bootsect.lnx="LINUX"
```

(естественно, что в кавычках вы можете поставить все, что вам угодно).

8. Осталось перезапустить компьютер еще раз, причем при загрузке вы уже будете иметь возможность выбрать Linux для загрузки. После этого будет запущен LILO, который загрузит Linux.

Теперь отдельно рассмотрим случай, когда вы (по ошибке или намеренно) установили LILO в главную загрузочную запись диска (Master Boot Record, MBR). В этом случае загрузочная запись Windows NT (или 2000) будет затерта, и загрузить Windows NT (см. шаг 6 выше) уже будет невозможно. Если вы все еще намерены пользоваться загрузчиком OS Loader от NT, а не LILO, последовательность действий несколько изменяется: вместо шага 6 необходимо проделать следующее.

1. Загрузите Windows NT с загрузочных дискет (см. шаг 1 выше). При этом необходимо выбрать в меню загрузчика команду **Recover,** а затем — режим **Command mode.** При запросе следует зарегистрироваться с учетной записью администратора системы (Administrator).
2. Восстановите главную загрузочную запись диска. Для этого можно дать команду fdisk /mbr. У меня получалось, хотя в некоторых статьях утверждается, что восстановить таким образом MBR удается не всегда. В Windows 2000 имеются специальные команды fixboot и fixmbr (они

запускаются из консоли восстановления, см. справку). Выполните их обе в указанном порядке. Теперь Windows 2000 снова будет загружаться нормально.

3. Перезапустите компьютер с загрузочной дискеты Linux и зарегистрируйтесь в системе с полномочиями администратора (root).
4. Введите команду `cd` /etc и откройте файл lilo.conf. В начале этого файла есть ссылка на загрузочный раздел по умолчанию, например, /dev/hda.
5. С помощью любого редактора, например, CoolEdit из Midnight Commander, следует заменить это значение указанием на диск и раздел, куда была установлена Linux (точнее, указанием на тот диск и раздел, который монтируется как корневой в Linux). Если Linux установлена в раздел /dev/hdc1, то именно это и следует записать, т. е. поменять /dev/hda на /dev/hdcl. Если вы не помните, куда именно установлена Linux, найдите последнюю переменную файла /etc/lilo.conf — переменную image. В ней хранится нужное значение.
6. Выполнить команду `/etc/lilo` для записи загрузчика в раздел /dev/hdcl (команду `lilo` нужно выполнить без аргументов). Будет выдано предупреждение о том, что раздел не является первым на диске. Именно это нам и нужно, чтобы загрузочная запись Windows 2000 осталась в целости и сохранности.
7. Выполнить шаги 6—8 приведенного выше алгоритма.

Легко догадаться, что только что приведенная сложная последовательность операций с двумя лишними перезагрузками потребовалась только для того, чтобы перенести загрузочный сектор Linux из MBR в первый сектор раздела, отведенного для Linux, и восстановить MBR от Windows.

Все, на этом установка завершена, и вы имеете возможность на этапе загрузки выбирать ту ОС, которая будет осуществлять управление вашим компьютером в данном сеансе работы.

2.7. Использование загрузчика LILO

2.7.1. Установка и настройка загрузчика LILO

Как уже было сказано в разделе о выборе загрузчика, если у вас была установлена Windows 98 с файловой системой FAT16, то наиболее логичным и доступным выбором является использование программы, которая входит в состав всех дистрибутивов ОС Linux, и называется LILO (Linux LOader).

Так же, как в случае с Windows NT, приведем пошаговые инструкции того, как в этом случае настроить процесс загрузки.

1. До начала каких-либо действий по инсталляции Linux изготовьте загрузочную дискету Windows.

2. С помощью программы Partition Magic освободите часть дискового пространства и создайте на свободном месте раздел типа ext2 (файловая система Linux) и swap-раздел. О том, как это сделать, рассказано выше. Если объем вашего жесткого диска превышает 8,4 Гбайт, особенно внимательно прочитайте *разд. 2.3—2.5.*
3. Проведите процедуру инсталляции Linux, следуя рекомендациям, прилагаемым к имеющемуся у вас дистрибутиву. Учтите, что если вы вознамерились использовать загрузчик LILO, вам необходимо при инсталляции системы установить LILO в основную загрузочную запись (Master Boot Record). Изготавливать в процессе установки загрузочную дискету, в принципе, не обязательно, но я очень рекомендую вам это сделать.

Примечание

Загрузчик LILO необязательно устанавливать в главную загрузочную запись диска, он может располагаться в загрузочной записи первичного активного Linux-раздела или даже логического раздела в расширенном разделе. В таком случае в MBR должно быть нечто, способное его загрузить, например, стандартный загрузчик MS-DOS или Windows. Но необходимости применения такого варианта я не вижу (раз уж вы используете LILO в качестве основного загрузчика), поэтому здесь его рассматривать не будем.

4. На следующем шаге нужно заставить LILO загружать ОС по выбору. LILO конфигурируется с помощью файла /etc/lilo.conf и команды `/etc/lilo`. Эта команда устанавливает (или переустанавливает) LILO.

Рассмотрим небольшой пример файла конфигурации LILO. Для примера будем считать, что устройство /dev/hda1 является разделом с DOS/Windows, а раздел /dev/hda2 содержит Linux. В таком случае файл /etc/lilo.conf может иметь примерно такой вид:

```
boot = /dev/hda2
compact
delay = 50
# message = /boot/bootmesg.txt
root = current
image = /boot/vmlinuz-2.2.11-4bc
label = linux
read-only
other = /dev/hda1
table = /dev/hda
label = dos
```

Дадим некоторые пояснения к этому примеру.

Строка boot указывает загрузочное устройство.

Строка compact включает режим сжатия map-файла, содержащего характеристики загрузочных ядер; это ускоряет начальную загрузку.

С помощью команды message можно заставить загрузчик выдавать при загрузке произвольное сообщение.

Начиная со строки image, идут секции конфигурационного файла, соответствующие разным операционным системам, которые должны загружаться по выбору пользователя. В каждой такой секции имеется строка label. В этой строке записывается имя, которое вводится в ответ на приглашение LILO или является командой меню и служит для выбора пользователем загружаемой ОС. Если имя не введено по истечении времени, заданного строкой `delay` (задается в десятых долях секунды), будет загружена ОС, выбираемая по умолчанию. В данном случае по умолчанию будет загружаться Linux, поскольку соответствующая ей секция стоит первой в файле. Можно указать загружаемую по умолчанию систему с помощью строки вида `default=dos` (то есть используя метку из соответствующей строки label).

Строка `table=<device>` содержит указание на устройство, на котором находится таблица разбиения диска. LILO не передает информацию о разбиении загружаемой операционной системе, если эта переменная не задана. (Некоторые операционные системы имеют другие средства для определения того, из какого раздела они загружены.) Не забывайте, что необходимо выполнить команду `/sbin/lilo`, если вы изменили ссылку на таблицу разбиения, задаваемую переменной table.

Если вы задали строку (лучше сказать, секцию) other = `/dev/hda1` в файле /etc/lilo.conf, то в корневом каталоге диска /dev/hda1 (диска C: в терминологии Microsoft) должен находиться вторичный загрузчик. У меня, например, на одном из компьютеров с многовариантной загрузкой там находится NT Loader (поскольку Windows NT была установлена до Linux), и LILO успешно загружает Windows NT. Только надо установить в файле boot.ini задержку времени равной нулю, чтобы не получать собственного меню загрузки NT Loader. Впрочем, если вы хотите по каким-то причинам видеть это меню, то значение timeout в файле boot.ini надо задать отличным от нуля (задается в секундах). Это может понадобиться, например, для обеспечения возможности загружать Windows 98 как еще один вариант ОС.

Если вы хотите грузить Windows непосредственно из LILO, то добавьте в /etc/lilo.conf еще одну секцию:

```
other = /boot/bootsect.dos
label = win,
```

где файл bootsect.dos берется из корневого каталога того диска, на котором стоит NT Loader.

5. После того как вы откорректировали файл /etc/lilo.conf, необходимо выполнить команду /etc/lilo, чтобы изменения вступили в силу. Эта команда (которая в руководстве называется map-installer) устанавливает вторичный загрузчик системы, который будет активизирован во время следующей загрузки машины. Прежде, чем запускать /etc/iilo для модификации загрузочных процедур, выполните эту команду с параметром -t. При этом будет выполнена вся процедура инсталляции загрузчика, кроме изменения map-файла, записи модифицированного загрузочного сектора и изменения таблицы разбиения диска, т. е. выполнен тест нового варианта. Если добавить еще опцию -v, вы получите более подробную информацию о том, что будет делать команда /etc/iilo.

   Когда /sbin/lilo перезаписывает загрузочный сектор, старое содержимое этого сектора автоматически сохраняется в файле. По умолчанию это файл /boot/boot.NNNN, где NNNN соответствует номеру устройства, например, 0300 — это /dev/hda, 0800 — это /dev/sda и т. д. Если такой файл уже существует, он не перезаписывается. Но можно задать альтернативный файл для сохранения загрузочного сектора.

   Файл /boot/boot.NNNN можно использовать для восстановления старого содержимого загрузочного сектора, если более простой метод его восстановления недоступен. Соответствующие команды имеют вид:

   ```
   [root:~#] dd if=/boot/boot.0300 of=/dev/hda bs=446 count=1
   ```

   или

   ```
   [root:~#] dd if=/boot/boot.0800 of=/dev/sda bs=446 count=1
   ```

   (bs=446 потому, что восстанавливаем только программу-загрузчик, и не трогаем таблицу разбиения диска).

   Копию загрузочного сектора лучше сохранить на дискете. В случае, если неприятности произойдут, вы сможете восстановить старую загрузочную запись MBR, выполнив команду (предполагается, что дискета смонтирована в каталог /mnt):

   ```
   [root:~#] dd if=/mnt/MBR of=/dev/hda bs=446 count=1
   ```

   Восстановить старый MBR при необходимости можно также командой /sbin/iilo с опцией -и. Только надо иметь в виду, что эта команда отрабатывает корректно при условии, что каталог LILO (а именно, /boot) не изменялся со времени инсталляции.

   Стандартный MBR от MS-DOS может быть восстановлен также, если воспользоваться загрузочной дискетой системы DOS, восстановив MBR командой fdisk /mbr. Она изменяет только код программы-загрузчика в MBR, не изменяя таблицу разбиения диска.

6. После переустановки загрузчика надо перезагрузить компьютер, опробовав разные варианты загрузки.

В заключение раздела приведем некоторые сведения о том, какие затруднения могут возникать при использовании LILO.

Когда LILO загружается, он выводит на дисплей слово "LILO". При этом вывод каждой буквы обозначает завершение определенного действия или этапа загрузки LILO. Если загрузка сорвется, то по числу выведенных букв можно судить о причине возникновения проблемы.

- **Ничего не выведено** — никакая часть LILO не была загружена. Либо LILO не был установлен, либо раздел, на котором он находится, не является активным.
- L [код ошибки] — первичный загрузчик загрузился и стартовал (на него передано управление), но он не сумел загрузить вторичный загрузчик. Двухзначный код ошибки указывает на конкретную причину проблемы (расшифровку кодов надо искать в технической документации на LILO). Обычно это связано с дефектами носителя или неправильно заданной геометрией диска. Если только LILO не остановился на этом этапе, выдавая бесконечную последовательность кодов ошибки, проблема обычно легко решаема.
- LI — первичный загрузчик сумел загрузить вторичный загрузчик, но не сумел запустить его на выполнение. Это может быть вызвано ошибкой в задании геометрии диска или тем, что файл /boot/boot.b был перемещен без перезапуска `/sbin/lilo`.
- LIL — вторичный загрузчик запустился, но не смог загрузить таблицу дескрипторов из map-файла. Причина обычно состоит в наличии дефектов на диске или в неправильно заданной геометрией диска.
- **LIL?** — вторичный загрузчик был загружен по неправильному адресу. Обычно вызвано ошибкой в задании геометрии диска или тем, что файл /boot/boot.b был перемещен без перезапуска `/sbin/lilo`.
- LIL- — таблица дескрипторов разрушена. Обычно вызвано ошибкой в задании геометрии диска или тем, что файл /boot/boot.b был перемещен без перезапуска /sbin/iilo.
- **LILO** — все части LILO успешно загружены.

2.7.2. Установка других операционных систем после Linux

При инсталляции MS-DOS и Windows 95/98 ее стандартный загрузчик независимо от вашего желания записывается в Master Boot Record (MBR), а признак активности в таблице разделов ставится на раздел MS-DOS (Windows 95/98). А стандартный загрузчик MS-DOS и Windows 95/98 умеет только передавать управление на первый сектор активного раздела. Поэтому, если вы вначале установите Linux, а потом будете ставить Windows 95/98

или MS-DOS, то Linux перестанет загружаться. Говорят, что Windows NT и 2000 загрузчик из MBR не трогают (но сам я этот факт не проверял). Восстановить загрузку LILO можно либо путем перезапуска `/sbin/lilo` (если LILO установлен в MBR), либо сделав активным раздел LILO (если он установлен в первичный раздел).

С проблемами, возникающими при установке после Linux другой ОС, обычно можно справиться, загрузившись в Linux с помощью загрузочной дискеты, откорректировав конфигурационный файл LILO (добавив вызов новой ОС) и запустив /sbin/iilo.

2.7.3. Перенос каталога /boot в DOS-раздел

Последние версии ядра Linux поддерживают возможность размещения файлов, необходимых на этапе загрузки, в файловой системе MS-DOS (или UMSDOS). Поскольку в большинстве случаев разделы DOS занимают как раз те области диска, где связанные с BIOS ограничения не действуют, это позволяет решить многие проблемы больших дисков, возникающие в тех случаях, когда раздел, отведенный для Linux, не может быть использован для размещения в нем каталога /boot.

Для того чтобы реализовать такой вариант загрузки, DOS-раздел монтируется в режиме чтение/запись, создается каталог (например, /dos/linux), в который перемещаются все файлы из каталога /boot и образы ядер Linux, каталог /boot заменяется символической ссылкой на каталог /dos/linux, новое местоположение каталога /boot указывается в файле /etc/lilo.conf, и, наконец, запускается /sbin/iilo.

2.8. Загрузка Linux из MS-DOS с помощью loadlin.exe

Не только загрузочные файлы и образы ядра могут располагаться в DOS-разделе, но и вообще вся загрузка Linux может быть организована из DOS. Для этого используется специальная программа loadlin.exe, разработанная Хансом Лерменом (Hans Lermen, lermen@elserv.ffm.fgan.de). Эта программа используется в таких дистрибутивах, как Red Hat, для организации процедур установки Linux с CD-ROM. Поэтому она имеется на дистрибутивном диске, а, следовательно, всегда имеется в вашем распоряжении.

Loadlin.exe предоставляет вам самый безопасный способ загрузки Linux с жесткого диска, если вы имеете на нем загрузочный (активный) DOS- или Windows-раздел. Этот вариант организации загрузки Linux можно особенно порекомендовать начинающим пользователям Linux. Большинство новичков, устанавливающих Linux, слишком нетерпеливы, чтобы читать очень хорошее, но очень длинное описание загрузчика LILO, прилагаемое к этой

программе (да еще по-английски!). Поэтому они часто используют его некорректно, в результате чего теряют возможность вообще загрузить какую-нибудь операционную систему (я тоже попадал в такую ситуацию). Для таких пользователей гораздо удобнее начать освоение Linux, используя для загрузки loadlin.exe.

Программа loadlin.exe не требует какой-либо установки, надо только разместить саму программу и образы ядра на одном из дисков, доступных в DOS. С помощью этой программы можно загрузить Linux с CD или сетевого диска, не используя загрузочной дискеты. Это делает loadlin.exe великолепным инструментом на те случаи, когда необходимо загрузить Linux после какого-то сбоя в работе загрузчика LILO.

Версия 1.6 loadlin.exe работает практически при любых конфигурациях DOS и имеет очень мало ограничений. Она может использовать расширенную память и может загружать большие ядра (bzImages) и образы виртуальных дисков (initrd) непосредственно в верхние области ОП.

Применение loadlin.exe не означает, что Linux работает под DOS, т. к. эта программа обеспечивает "логическую перезагрузку" вашего компьютера, после чего DOS полностью заменяется на Linux. Если вы хотите вернуться в DOS, вы должны перезагрузить компьютер, например, с помощью команды reboot.

Итак, что же необходимо для того, чтобы воспользоваться программой loadlin.exe.

- На вашем компьютере (конечно, с процессором 386 или выше) должна быть установлена DOS или Windows 95.
- Нужно иметь сжатые образы ядра (zImage, bzImage).

Примечание

zImage — это старый бинарный формат ядра, bzImage — это более новый формат (номер ядра версии больше 1.3.73), который может иметь размер до 1 Мбайт, следовательно, разархивированное ядро может иметь размер до 2 Мбайт. Далее будем говорить только о файлах zImage, хотя вы можете всюду заменить zImage на bzImage.

- Саму программу loadlin.exe, которую вы можете найти на дистрибутивном диске как в разархивированном виде, так и в пакете LODLIN16.TGZ, который содержит, кроме того, руководство по ее использованию DOC\MANUAL.TXT, пример файла задания параметров DOC\TEST.PAR и руководство по заданию параметров DOC\PARAMS.DOC (не забывайте, что все это файлы DOS).

Если вы запустите команду `loadlin` без параметров:

```
C:\LOADLIN> loadlin
```

то получите подсказку по использованию программы. Удобнее может оказаться запустить ту же программу с параметром more (в стиле Linux):

```
C:\LOADLIN> loadlin | more
```

Теперь мы можем рассмотреть последовательность действий по установке Linux в том варианте, когда загружаться она будет с помощью loadlin.exe.

1. Выделите раздел для Linux (как это сделать — *см. разд. 2.5).*
2. Установите Linux в выделенный раздел. При этом LILO установите в первый сектор Linux-раздела, чтобы не перезаписать MBR и не потерять возможность загружаться в Windows.
3. После завершения процедур установки загрузите Linux (если не получается по-другому, то используйте загрузочную дискету). Смонтируйте DOS-раздел (будем считать, что в Linux DOS-раздел именуется как /dev/hda1, а Linux-раздел — как /dev/hda3):

   ```
   root]# mount -t vfat /dev/hdal /mnt/C
   ```

   Создайте каталог /mnt/C/loadlin и разархивируйте в него содержимое файла LODLIN16.TGZ с дистрибутивного CD-диска Linux. Кроме того, поместите туда же файл с образом ядра из каталога /boot. Найти нужный файл образа ядра можно с помощью файла /etc/lilo.conf: найдите в нем строку `"image=..."` и вы увидите нужное имя справа от знака равенства. У меня, например, полное имя этого файла — vmlinuz-2.2.16-3bc, но я при копировании в каталог /mnt/C/loadlin переименовал его в vmlinuz; это имя и буду использовать далее в примерах.
4. Теперь перезагрузите компьютер в DOS. Если у вас есть возможность загрузить непосредственно DOS, то делайте это сразу, а если нет, то загружайте Windows, при появлении сообщения "Загрузка Windows 95" нажимайте клавишу <F8> и выбирайте вариант **Command prompt only.** Если вы не успели нажать на клавишу <F8>, то можно дождаться завершения загрузки Windows 95, после чего воспользоваться кнопкой **Пуск,** выбрать команду **Завершение работы** и далее — команду **Перезагрузить компьютер в режиме эмуляции DOS.**

   После выхода в режим DOS перейдите в каталог C:\LOADLIN (cd\LOADLIN) и выполните Команду

   ```
   C:\LOADLIN> LOADLIN vmlinuz /dev/hda3 ro vga=ask
   ```

   или, если вы хотите загрузить ядро с установкой RAM-диска:

   ```
   C:\LOADLIN> LOADLIN vmlinuz /dev/ram rw initrd=diskimage
   ```

   Можно также записать все аргументы команды loadlin.exe в файл (например, с именем param) и вызвать ту же команду следующим образом:

   ```
   C:\LOADLIN> LOADLIN @params
   ```

Такая возможность особенно полезна для тех случаев, когда вы задаете много аргументов командной строки, и ее длина становится больше 127 символов. Полное описание всех возможных аргументов (параметров) команды loadlin.exe вы сможете найти в файле PARAMS.DOC или в Интернете на сайтах **http://sunsite.unc.edu/mdw/HOWTO/BootProrapt-HOWTO.html и http://rsphyl.anu.edu/~gpgl09/BootPrompt-HOWTO.html.**

Теперь вы можете пользоваться этим способом загрузки Linux. Единственное, что остается, это избавиться от необходимости каждый раз при загрузке вводить команду `loadlin` со всеми параметрами. Для облегчения можно прописать вызов loadlin в файл autoexec.bat или создать командный файл (например, linux.bat), с помощью которого просто запускать Linux, предварительно загрузившись в режиме DOS. Я думаю, что приведенных выше данных вполне достаточно для создания необходимого bat-файла. Если же у вас что-то не получится, загляните в статью А. Московских *(см. [П4.13] приложения),* где этот вопрос рассмотрен более подробно. Там, в частности, отмечается, что если на машине установлены Windows 95 и Linux, то нельзя пытаться загружать Linux из графической оболочки и требуется отключить некоторые опции в скрытом файле C:\MSDOS.SYS (это простой текстовый файл), а именно, добавить в него две строки:

```
BootGUI=0
Logo=0
```

Первая строка отключает загрузку графической оболочки, и выбор команды меню `W95` будет вызывать переход к обычной командной строке DOS. (Чтобы загрузить графическую оболочку, вы должны будете ввести команду `C:>` win).

Logo=o отключает вывод логотипа Windows. Дело в том, что для некоторых графических адаптеров Linux может выдавать после загрузки "пустой" экран, если перед его загрузкой отображался логотип Windows.

В списке литературы в конце книги я привожу небольшой список других источников и ссылок на материалы, имеющих отношение к вопросу установки нескольких ОС на одном компьютере. Так что в случае, если в моей книге рассмотрены не все вопросы, и вы столкнетесь с какими-то сложностями, ищите ответы в указанных материалах.

Глава 3

Первый запуск ОС Linux

3.1. Загрузка ОС Linux

Итак, инсталляция Linux завершена, и вы перезапускаете компьютер. Если Linux — единственная операционная система, установленная на вашем компьютере (а, значит, загрузчик LILO размещается в главной загрузочной записи — MBR), то после обычного тестирования аппаратуры, выполняемого BIOS, ненадолго появится надпись

```
LILO boot:
```

Если не предпринимать никаких действий, то на экран будет выдана масса сообщений, разбирать смысл которых мы пока не будем, и, наконец, появится стилизованное изображение пингвинчика. (Я пока рассматриваю тот случай, когда вы при установке отказались от автоматической загрузки графической оболочки). Ниже изображения пингвина на экране написано:

```
Linux Version 2.0.36, Compiled #1 Tue Dec 29 13:11:11 EST 1998
One Intel 486 DX/2-WB Processor, 16M RAM, 33.28 Bogomips Total
                    localhost.localdomain
Black Cat Linux release 5.2  (Fulcrun)
Kernel 2.0.36 on an i486
localhost login:
```

Я привел здесь то сообщение, которое появлялось у меня при одном из вариантов установки, когда я ставил Black Cat версии 5.2; у вас, конечно, сообщение будет отличаться в некоторых деталях. Если у вас хватит терпения дочитать эту книгу, то вы узнаете, что выдаваемое при загрузке сообщение при желании можно изменить, так что будет выдаваться что-нибудь вроде "Привет, дружище! Сегодня И ноября 2001 года. Сейчас 19 часов 22 минуты. Введи, пожалуйста, свое имя и пароль". Однако пока рано говорить о том, как это делается.

Если вы загружались с дискеты, то загрузка происходит точно так же, только чуть медленнее.

Если Linux не единственная ОС на вашем компьютере, и вы используете LILO для организации многовариантной загрузки, то в тот момент, когда на экране появится надпись LILO `boot:`, вы должны нажать клавишу <Tab> или <?>. Тогда LILO выдаст вам список меток, которые сопоставлены раз-

ным ОС. В версии 21 LILO уже автоматически выводит этот список на экран в виде меню. Необходимо выбрать из меню или ввести (набрать на клавиатуре) одну из этих меток и нажать клавишу <Enter>. Если вы выберете метку, соответствующую Linux, то в конце концов вы все равно должны увидеть слово `login:`, которое в данном случае служит приглашением к вводу вашего пользовательского имени.

3.2. Вход в систему

Как вы понимаете, в ответ на это приглашение необходимо ввести имя пользователя, а потом, по запросу, и пароль для входа в систему. Если это первый вход в систему после ее установки, то входить надо под именем "root". Это единственный пользователь, для которого обязательно заводится счет или бюджет (account) во время инсталляции. Этот пользователь является полным хозяином системы (как сейчас, так и в последующем), т. е. имеет неограниченный доступ к ее ресурсам, может заводить и удалять других пользователей, останавливать систему и т. д. Неосторожное поведение пользователя с такими правами легко может привести к печальным последствиям, вплоть до полного краха системы. Поэтому обычно под этим именем входят в систему только для выполнения административных задач. Но у нас сейчас как раз такой случай, так что в ответ на приглашение login: вводим "root" и нажимаем клавишу <Enter> (или <Return>). Система выдаст запрос на ввод пароля:

```
Password:
```

Очевидно, что в ответ надо вводить пароль того пользователя, имя которого было введено ранее. При первой загрузке надо ввести тот пароль, которой был задан для пользователя root в процессе инсталляции, и нажать <Enter>. Заметим, что если после ввода имени очень долго не вводить пароль, то система снова вернется к запросу имени пользователя. После ввода пароля вы увидите примерно такую надпись:

```
[root@localhost /root]#
```

Такая строка называется приглашением. Появление приглашения означает, что система готова воспринять и выполнить вашу команду. Сейчас это свидетельствует о том, что вы успешно вошли в систему. Вы видите черный экран и приглашение системы к вводу команды — то, что в MS-DOS или Windows принято называть режимом командной строки. Мы будем называть этот режим текстовым (в отличие от графического режима, предоставляемого системой X Window).

В приведенном примере приглашение включает в себя указание имени пользователя (root), имени системы (`localhost`) и текущего каталога (`/root`). Вид приглашения тоже можно изменить. Во всех последующих

примерах мы будем использовать приглашение, состоящее только из имени пользователя.

Прежде чем предложить вам ввести первую команду, надо сказать, что в любой UNIX-системе учитывается регистр символов, т. е. различаются строчные и прописные буквы. Поэтому вводить все команды и их параметры следует именно так, как указано в примерах, учитывая регистр.

Первая команда, которую стоит ввести, — команда useradd. После имени команды надо ввести пробел и имя пользователя, например, jim:

```
[root]# useradd jim
```

После этого система будет знать о существовании пользователя jim (говорят, будет "открыт счет для пользователя jim"). Однако войти в систему (или, как обычно говорят, "логироваться") под этим именем еще невозможно. Для того чтобы система разрешила работать пользователю с именем jim, надо задать ему пароль. Для этого вводим команду

```
[root]# passwd jim
```

Появится строка

```
New UNIX password:
```

Вводите пароль. После того как вы завершите ввод нажатием клавиши <Enter>, система попросит ввести его повторно:

```
Retype new UNIX password:
```

Если вы не ошиблись при вводе (пароль приходится вводить "вслепую", поскольку он не отображается на экране), появится сообщение:

```
passwd: all authentication tokens updated successfully
```

и приглашение системы. Если вы выбрали пароль не очень удачно (слишком короткий или простой), вам будет выдано предупреждение, но система все равно примет пароль и позволит новому пользователю входить с ним в систему.

Таким образом, вы познакомились с первыми двумя командами системы Linux: useradd и passwd. Следующая команда, о которой нужно знать каждому пользователю любой UNIX-системы, — это команда man. Команда man — это система встроенной помощи Linux. Вводить ее надо с параметром — именем другой команды или ключевым словом, например,

```
[root]# man passwd
```

В ответ вы получите описание соответствующей команды или информацию по теме, обозначенной ключевым словом. Поскольку информация обычно не помещается на одном экране, при просмотре можно пользоваться клавишами <PageUp> и <PageDown>, а также клавишей пробела. Нажатие клавиши <Q> в любой момент приводит к выходу из режима просмотра и воз-

врату в режим ввода команд. Попробуйте просмотреть информацию по рассмотренным уже командам login и passwd. Замечу, что точно также можно получить информацию по самой команде *man*. Введите

```
[root]# man man
```

К сожалению, в большинстве случаев информация выдается по-английски. Если вы не читаете по-английски, то терпеливо читайте настоящее руководство или другую подходящую книгу по данной теме.

Вы можете попробовать вводить еще некоторые команды и понаблюдать за реакцией системы. Попробуйте, например, команды, перечисленные в табл. 3.1 (вводите их с приведенными в таблице параметрами).

Таблица 3.1. *Простейшие команды Linux*

Команда	Краткое описание
whoami	Сообщает имя, с которым вы вошли в систему в данном сеансе работы
w или who	Сообщает, какие пользователи работают в данный момент в системе
pwd	Сообщает имя текущего каталога
ls -l	Выдает список файлов и подкаталогов текущего каталога
cd <имя_каталога>	Осуществляет смену текущего каталога
ps ax	Выдает список выполняющихся процессов

Просмотрите описания этих команд с помощью команды man.

Я не буду приводить здесь более подробный список возможных команд. Во-первых, с необходимыми командами можно будет постепенно познакомиться в ходе дальнейшего чтения. Во-вторых, описания основных команд можно найти в любой книге по UNIX.

3.3. Консоль, виртуальные терминалы и оболочка

Итак, вы приобрели первый опыт работы в текстовом, или "консольном", режиме системы Linux. Понятия "терминала" и "консоли", которые встретятся нам еще не раз, требуется, вероятно, дополнительно пояснить.

Когда создавалась система UNIX, компьютеры были большими (мейнфреймами), и пользователи работали на них через множество последовательных интерфейсов для подключения удаленных терминалов. Терминал —

это устройство, которое предназначено для взаимодействия пользователя с компьютером и состоит из монитора и клавиатуры. К вашему персональному компьютеру наверняка не подключены удаленные терминалы, но есть клавиатура и монитор, которые и выполняют роль терминала пользователя (только в его состав добавилась мышь).

У мейнфреймов имелся особый терминал, который предназначался для системного администратора и назывался консолью. Консоль обычно подсоединялась к компьютеру не по последовательному интерфейсу, а через отдельные разъемы (иногда в качестве устройства вывода в ее состав вместо монитора входило печатающее устройство).

Поскольку в UNIX-системах обычно соблюдаются традиции, клавиатура и монитор персонального компьютера ведут себя так же, как ранее консоль. Преимущество такого решения состоит в том, что все старые программы, создававшиеся для администраторов UNIX, без проблем работают и на новом типе системной консоли.

Но, кроме консоли, Linux позволяет подключать к компьютеру и удаленные терминалы и, более того, обеспечивает возможность работы с несколькими виртуальными терминалами с одной консоли. Нажмите комбинацию клавиш <Ctrl>+<Alt>+<F2>. вы снова увидите приглашение `login:`. Однако это не возврат к началу работы с системой — вы просто переключились в другой виртуальный терминал. Здесь вы можете зарегистрироваться под другим именем. Попробуйте войти в систему под именем только что заведенного пользователя. После этого нажмите комбинацию клавиш <Ctrl>+<AltXFl>. Вы вернетесь к первому экрану. По умолчанию Red Hat Linux открывает при запуске 6 параллельных сеансов работы (виртуальных терминалов), и этим иногда очень удобно пользоваться. Для переключения между виртуальными терминалами используются комбинации клавиш <Ctrl>+<Alt>+<F1>—<Ctrl>+<Alt>+<F6>. (Замечу, что при работе в текстовом режиме тот же результат можно получить, используя комбинации <Alt>+<F1>—<Alt>+<F6>, однако в графическом режиме без клавиши <Ctrl> не обойтись, так что лучше сразу привыкать к комбинациям из 3-х клавиш). Кстати, если в процессе работы вы забыли, в каком терминале находитесь в данный момент, воспользуйтесь командой tty, которая выводит имя терминала в следующем формате: /dev/tty2.

Сразу же скажем, что, если вы хотите завершить сеанс работы с системой в одном из терминалов, вы можете сделать это нажатием комбинации клавиш <Ctrl>+<D>. Это не приведет ни к остановке работы компьютера, ни к перезагрузке системы. Не забывайте, что Linux — многозадачная и многопользовательская система. Завершение работы одного пользователя не означает, что надо выключать компьютер. Просто завершается сеанс работы одного из пользователей, и система снова выводит в данном терминале приглашение,

которое вы уже видели. Можно завершить сеанс работы и введя одну из **Команд** logout ИЛИ exit.

Зная теперь как открыть и закрыть сеанс работы в системе, выполните приведенные выше рекомендации, т. е. заведите себя как рядового пользователя (без суперпользовательских прав), завершите все сеансы работы, открытые от имени root, и снова войдите в систему под своим новым именем.

Теперь надо сказать несколько слов об оболочке. Оболочка, или просто shell (это слово часто не переводят, а оставляют в английском написании), — это программа, которая осуществляет все общение с пользователем. Именно оболочка воспринимает все команды, вводимые пользователем с клавиатуры, и организует исполнение этих команд. Поэтому оболочку можно назвать еще командным процессором (более привычный термин для пользователя DOS, не правда ли?). Строго говоря, когда выше говорилось, например, "система выводит приглашение", это неправильно, поскольку приглашение выводит именно оболочка, ожидая ввода пользователем очередной команды. Каждый раз, когда очередной пользователь входит в систему, команда login запускает для него командный процессор — оболочку. Если вы логировались со второго терминала под именем пользователя jim (или под другим выбранным вами именем), то обратите теперь внимание на различие в приглашениях у пользователей root и jim. У пользователя root приглашение оканчивается символом #, а у всех остальных пользователей — символом $.

Оболочку может запускать не только команда login. Вы можете просто ввести команду bash (именно так называется программа-оболочка в системе Red Hat Linux) и тем самым запустить новый экземпляр оболочки. Выходя из него (по команде exit или по комбинации клавиш <Ctrl>+<D>), вы вернетесь к предыдущему экземпляру оболочки.

Оболочка bash является не только командным процессором, но и мощным языком программирования. В ней имеется целый ряд встроенных (внутренних) команд и операторов, а, кроме того, в качестве команды может использоваться любая программа, хранящаяся в виде файла на диске. Список встроенных команд можно получить по команде help. Попробуйте! Детальную информацию по конкретной встроенной команде выдает та же команда help с указанием в качестве параметра имени встроенной команды, например: help cd.

Поскольку оболочка играет очень важную роль в Linux, ей будет посвящена отдельная глава этой книги. Впрочем, аналогичный материал вы найдете в любой книге по UNIX. Стоит только отметить, что для UNIX-подобных систем разработано несколько альтернативных bash оболочек. Их можно использовать и в Linux, но по умолчанию запускается именно bash.

Рассмотрим теперь еще одну команду, которую вам необходимо знать, поскольку все же компьютер у вас персональный (неважно, дома ли он стоит, или на работе). А это значит, что вы и есть суперпользователь данного ком-

пьютера. Но, как уже было сказано выше, входить в систему под именем суперпользователя не рекомендуется, поскольку любое неосторожное действие суперпользователя может привести к нежелательным последствиям. Входя под именем простого пользователя, вы, по крайней мере, не можете по неосторожности удалить или испортить системные файлы. В то же время, имеется ряд действий (например, монтирование файловых систем), выполнить которые может только суперпользователь. Не перезагружать же каждый раз компьютер! Именно в таких ситуациях выручает команда su. Достаточно ввести команду su и текущая оболочка (так и хочется сказать "система") запустит для вас новый экземпляр оболочки, в который вы попадете уже с правами пользователя root. Естественно, что для этого вам придется (в ответ на соответствующий запрос) ввести пароль этого пользователя. Закончив выполнять административные действия, выйдите из оболочки, и вы снова станете непривилегированным пользователем с отведенными ему полномочиями.

Если вы вошли в систему под именем root, то вы можете аналогичным образом запустить новый экземпляр оболочки от имени любого пользователя, пароль которого вы знаете. Но для этого надо указать имя этого пользователя в командной строке, например:

```
[user]$ su jim
```

Когда мы вводим su без указания имени, по умолчанию подставляется имя суперпользователя root.

Но в ОС Linux есть еще одна возможность временно переключаться в бюджет пользователя root для выполнения административных функций. Вспомните, что Linux — это многопользовательская система, в ней одновременно могут работать несколько пользователей. Поэтому в первом виртуальном терминале можно войти под именем root, а в любом другом терминале — под именем простого пользователя. Основную работу вы можете выполнять как простой пользователь, а когда потребуется выполнить административные функции, вы "зовете системного администратора". Для этого достаточно нажать <Ctrl>+<Alt>+<Fl> — и системный администратор уже тут. По завершении операции, которую может выполнить только суперпользователь, вы немедленно должны вернуться в бюджет простого пользователя. В таком случае вы не рискуете нарушить что-либо в системе, пока еще не набрались необходимого опыта.

3.4. Редактирование командной строки. История команд

В предыдущих разделах вам было предложено выполнить несколько команд ОС Linux. Полагаю, что, если в процессе набора возникали ошибки, вы сами догадались, как их исправить. Тем не менее, будет полезно привести для

справки короткую сводку команд (табл. 3.2), позволяющих редактировать командную строку, а также вообще оказывать какое-то влияние на поведение оболочки с помощью клавиатуры (о мышке разговор отдельный).

Примечание

Приводимые ниже описания команд относятся к оболочке GNU bash версии 1.14.7(1). Думается, что они будут верны и для последующих версий этой оболочки.

Таблица 3.2. *Клавиши редактирования командной строки*

Клавиша	Описание реакции системы
<→> или <Ctrl>+<F>	Перемещение вправо по командной строке в пределах уже набранной цепочки символов плюс один символ справа (место для ввода следующего символа)
<←> или <Ctrl>+<B>	Перемещение на один символ влево
<Esc>+<F>	Перемещение на одно слово вправо
<Esc>+<B>	Перемещение на одно слово влево
<Home> или <Ctrl>+<A>	Перемещение в начало набранной цепочки символов
<End> или <Ctrl>+<E>	Перемещение в начало/конец набранной цепочки символов
<Del> или <Ctrl>+<D>	Удаление символа, на который показывает курсор
<Backspase>	Удаление символа в позиции, предшествующей курсору
<Ctrl>+<K>	Удалить правую часть строки, начиная с символа, на который указывает курсор
<Ctrl>+<U>	Удалить левую часть строки, включая символ, который находится слева от курсора
<Enter> или <Ctrl>+<M>	Запуск на выполнение команды, определяемой набранной цепочкой символов
<Ctrl>+<L>	Очистить экран и поместить текущую команду в верхней строке экрана
<Ctrl>+<T>	Поменять местами два символа: символ, на который показывает курсор, и символ слева от курсора, затем, курсор переместить на один символ вправо
<Esc>+<T>	Поменять местами два слова: слово, на которое указывает курсор и слово слева от первого

Таблица 3.2 *(окончание)*

Клавиша	Описание реакции системы
<Ctrl>+<K>	Вырезать часть строки от текущей позиции курсора до конца строки (вырезанная часть строки сохраняется в буфере, ее можно вставить в другое место строки)
<Esc>+<D>	Вырезать часть строки от текущей позиции курсора до конца текущего слова (если курсор указывает на пробел между словами, то вырезается все слово справа от курсора)
<Esc>+<Del>	Вырезать часть строки от текущей позиции курсора до начала текущего слова (если курсор указывает на пробел между словами, то вырезается все слово слева от курсора)
<Ctrl>+<W>	Вырезать часть строки от текущей позиции курсора до предыдущего пробела
<Ctrl>+<Y>	Вставить последний вырезанный текст в позицию курсора
<Esc>+<C>	Символ, на который указывает курсор, заменить на тот же, но заглавный, а курсор переместить на первый пробел справа от текущего слова
<Esc>+<U>	Сделать символы данного слова заглавными, начиная с символа, на который указывает курсор, а курсор установить на пробел справа от слова
<Esc>+<L>	Превратить символы, начиная с символа, на который указывает курсор, до конца данного слова в прописные (маленькие) буквы, а курсор установить на пробел справа от слова
<Shift>+<PgUp> <Shift>+ +<PgDown>	Эти команды позволяют просмотреть несколько страниц экранного вывода (количество зависит от размера видеопамяти); полезны в тех случаях, когда та или иная команда выводит на экран очень много информации, быстро пробегающей по экрану и как бы исчезающей для пользователя; как видите, эта информация не пропадает
<Ctrl>+<C>	Прервать выполнение запущенной команды
<Ctrl>+<D>	Выход из оболочки bash

Примечание

Если вы работаете не просто *в* оболочке bash, а запустили программу Midnight Commander, то такие клавиши, как <→>, <←>, <Home>, <End>, <Del> не могут использоваться так, как указано в приведенной таблице, поскольку они используются для перемещения подсветки в текущей панели. Но указанные выше в скобках комбинации символьных клавиш с клавишами <Ctrl> и <Esc> по-прежнему действуют для редактирования командной строки.

Список возможных команд не ограничивается только приведенными в табл. 3.2, но из-за ограниченности объема книги невозможно привести здесь

полный перечень клавиатурных команд. Для получения дополнительной информации воспользуйтесь командой info bash.

Заметим, что оболочка bash имеет встроенную подпрограмму, предназначенную для облегчения ввода команд в командной строке. Эта подпрограмма вызывается при нажатии клавиши <Tab> после того, как вы уже ввели некоторое число символов. Если эти символы являются началом названия одной из стандартных команд, которые известны оболочке, то возможны два варианта реакции оболочки на нажатие клавиши <Tab>. Если по введенным символам команда определяется однозначно, оболочка просто добавляет окончание команды в командную строку. Если однозначно восстановить имя команды по введенным символам невозможно, то выдается список возможных вариантов продолжения для того, чтобы пользователь мог ввести еще несколько символов, позволяющих однозначно завершить ввод команды нажатием клавиши <Tab>.

Если вы попробуете ввести символ табуляции в пустой командной строке, то после первого ввода вы получите только звуковой сигнал, а после второго — примерно следующее сообщение: "There are 1217 possibilities. Do you really wish to see them all? (y or n)" ("Возможны 1217 вариантов завершения. Вы действительно хотите увидеть их все?").

Если ввести символ табуляции после того, как введена одна из команд и пробел, оболочка предположит, что вы ищете имя файла, который должен вводиться как параметр команды, и выдаст в качестве подсказки список файлов текущего каталога. Если же достаточная часть имени файла введена, то заканчивается ввод этого имени в командную строку. Аналогичным образом можно пытаться угадывать окончания переменных окружения, если вместо клавиши <Tab> воспользоваться комбинацией <Esc>+<$>.

Для практической работы с оболочкой также полезно знать, что оболочка запоминает некоторое число введенных команд (по умолчанию 1000, это значение задается в переменной HISTSIZE; *см. гл. 5)* и позволяет вызывать их путем выбора из списка — так называемой истории команд. Историю команд можно просмотреть, введя в командной строке history (здесь вы сможете воспользоваться комбинациями клавиш <Shift>+<PgUp> и <Shift>+<PgDown>, чтобы просмотреть то, что выдаст эта команда). История команд сохраняется в файле, определяемом переменной HISTFILE (обычно $HOME/.bash_history). Для работы с историей команд в оболочке bash используются следующие комбинации клавиш (табл. 3.3).

Таблица 3.3. *Клавиши для управления историей команд*

Клавиша	Описание реакции системы
<t> или <Ctrl>+<P>	Переход к предыдущей команде в списке (движение назад по списку)

Таблица 3.3 *(окончание)*

Клавиша	Описание реакции системы
<↓> или <Ctrl>+<N>	Переход к следующей команде в списке (движение вперед по списку)
<PgUp>	Переход к (вызов в командную строку) самой первой команде, сохраненной в истории команд
<!>, <N>	Выполняется (без нажатия клавиши <Enter>) *n*-ная команда из списка истории команд
<!>, <->, <N>	Выполняется *n*-ая от конца списка команда
<!>, строка символов	Выполняется команда, имя которой начинается на строку символов (поиск нужной команды осуществляется движением в обратном порядке от конца файла истории и выполняется первая попавшаяся команда, которая начинается на строку символов)
<Ctrl>+<O>	То же, что нажатие клавиши <Enter>, затем отображается очередная команда из файла истории

3.5. Завершение работы системы Linux

Хотя компьютер, работающий под управлением ОС Linux, при выполнении некоторых условий можно оставлять работающим круглосуточно, большинство пользователей персональных компьютеров привыкли выключать их после завершения работы. Если вы работаете с ОС Linux, нельзя выключать компьютер простым отключением питания, как это было под MS-DOS. Дело в том, что в любой момент времени в системе запущено несколько процессов, вы могли видеть это, когда выполняли команду

```
[root]# ps ax
```

(и можете посмотреть еще раз, повторив запуск этой команды). Но более важная причина состоит в том, что некоторые из этих процессов могут работать с файлами, причем система не записывает все изменения файлов на диск сразу после внесения этих изменений пользователем или процессом, а сохраняет их временно в оперативной памяти (кэширует). Если просто выключить питание, эти изменения не будут сохранены и пропадут, что иногда может привести даже к невозможности последующей загрузки системы. Так что надо уметь правильно завершить работу системы перед выключением компьютера. Это делается командой shutdown.

Команда shutdown может быть выполнена только пользователем root, так что вы либо должны были войти в систему под этим именем, либо должны предварительно выполнить команду su, чтобы приобрести соответствующие права.

Команда shutdown имеет следующий синтаксис:

```
[root]# shutdown <options> <time> <warning-massage>
```

Замечание

Существует некоторая вероятность того, что запустив команду, вы получите ответ "command not found". Это значит, что оболочка не знает, где находится файл программы. В таком случае вам необходимо ввести команду с указанием полного пути, в данном случае в виде /sbin/shutdown -h, поскольку для команды shutdown файл программы лежит в каталоге /sbin.

Из опций программы shutdown наиболее часто используются две:

- -h — полная остановка системы (компьютер будет выключен);
- -r — перезагрузить систему.

Параметр time указывает время, когда должна быть выполнена команда (необязательно выполнять ее немедленно). Время можно указать в форме задержки от текущего момента. Например, если вы хотите, чтобы система остановилась через 5 минут, вводите команду

```
[root]# shutdown -r +5
```

что будет означать "остановить систему через 5 минут и перезагрузиться после того, как работа будет корректно завершена". Для вас пока наиболее актуальной формой этой команды будет, скорее всего,

```
[root]# shutdown -h 0
```

когда вы захотите просто выключить компьютер. Эквивалентом команды

shutdown -h O

является команда halt. При нажатии известной комбинации клавиш <Ctrl>+<Alt>+<Del> в Red Hat Linux выполняются действия, аналогичные команде

shutdown -r O

так что таким образом тоже можно выключить компьютер, только надо в момент перезагрузки отключить питание.

3.6. Помощь по работ с Linux

Вы завершили первый сеанс работы с операционной системой Linux, и, надеюсь, понимаете, что вам еще не раз потребуется подсказка в разных ситуациях. Надеюсь также, что настоящая книга сможет служить вам такой подсказкой на первых порах, но она наверняка не решит всех ваших проблем. Поэтому постараюсь сразу указать другие источники информации. При этом я несколько забегаю вперед, рассказывая о получении подсказки в графическом режиме работы, о котором еще ничего не было сказано. Но

это оправдано, поскольку о способах выхода из затруднительных ситуаций лучше знать заранее.

3.6.1. Источники справочной информации

Если вы окажетесь в ситуации, когда не знаете, что предпринять или сделать для достижения желаемой цели, лучше всего начать искать подсказку в самой системе. Дистрибутив Red Hat Linux содержит тысячи страниц документации, представленной в электронном виде, так что ответы на все возникающие вопросы у вас, что называется, "на кончиках пальцев". Существует несколько независимых источников, которые содержат информацию почти по любому аспекту работы в системе Linux:

- страницы интерактивного руководства man;
- гипертекстовое руководство info;
- документация, прилагаемая к пакетам ПО;
- текстовые файлы HOWTO и FAQ проекта Linux Documentation Project;
- команда locate.

Необходимо сразу сказать, что большую часть информации из этих источников вы будете получать на английском языке. Только для русифицированных дистрибутивов часть страниц интерактивного руководства man выдается на русском языке. Можно дополнительно скачать из Интернета имеющиеся там страницы руководства man, переведенные на русский язык *(см. [П5.1] приложения),* и разместить их в соответствующих каталогах. Но все равно, на русский переведено далеко не все. Учитывая это замечание, рассмотрим каждый из перечисленных выше источников информации подробнее.

3.6.2. Страницы интерактивного руководства *man*

Выше уже было вкратце рассказано о команде man, с помощью которой пользователь всегда может в затруднительной ситуации получить подсказку почти по любой команде системы, по форматам файлов и системным вызовам. Это основной способ получения подсказки во всех UNIX-системах. Страницы руководства man в Linux делятся на секции, описанные в табл. 3.4.

Таблица 3.4. Основные секции интерактивного руководства man

Секция	Содержание
1	Команды пользователя
8	Системные команды

Таблица 3.4 (окончание)

Секция	Содержание
2	Системные вызовы
3	Библиотечные вызовы (подпрограммы)
4	Устройства
5	Форматы файлов
6	Игры
7	Разное
9	Ядро (kernel internals)
л	Tcl/Tk commands

Порядок перечисления секций в этой таблице не случаен. Дело в том, что файлы с информацией расположены в подкаталогах каталога /usr/man и команда man ищет нужную информацию, просматривая эти подкаталоги именно в том порядке, который приведен табл. 3.4. Если вы, например, дадите команду

```
[user]$ man swapon
```

то получите справку о команде swapon из секции 8. Поэтому если вы хотите получить справку по системному вызову swapon, надо дать команду

```
[user]$ man 2 swapon
```

указывая номер секции, в которой надо искать информацию.

Страницы man просматриваются с помощью команды less (что дает возможность просматривать информацию поэкранно и перемещаться по этим экранам вперед и назад), так что для управления процессом вывода информации можно использовать клавиши, используемые в программе less. Наиболее употребительные приведены в табл. 3.5.

Таблица 3.5. *Клавиатурныекоманды, используемые при просмотре man-страниц*

Клавиша	Назначение
<Q>	Выход из программы
<Enter>	Просмотр строка за строкой
<Space>	Вывод следующего экрана информации
<B>	Вернуться к предыдущему экрану
</>, за которой следует строка символов и <Enter>	Поиск введенной строки символов
<N>	Повторение предыдущего поиска

Если вы предпочитаете читать текст не с экрана, а с отпечатанной копии, то можете отпечатать соответствующую страницу, воспользовавшись командой

```
[user]$ man имя_команды I lpr
```

или, если у вас postscript-принтер,

```
[user]$ man -t имя_команды | lpr
```

Но для того, чтобы получить необходимую информацию, нужно еще знать, что искать. В таком случае могут помочь команды whatis и apropos. Команда whatis производит контекстный поиск заданного ключевого слова (шаблона) в базе данных, содержащей перечень системных команд с кратким описанием команды. Выводятся только точные совпадения с ключевым словом. Команда apropos производит поиск по фрагментам слов. Аналогично команде apropos работает команда man с параметром -k. Попробуйте, например,

```
[user]$ man -k net.
```

Необходимо, однако, предупредить, что для того, чтобы команды man -k, whatis и apropos работали, следует вначале создать базу данных о системных командах, для чего надо запустить команду makewhatis. В противном случае вы можете на любой запрос получить сообщение "nothing appropriate". Правом запустить команду makewhatis обладает только пользователь root. Если вы не выключаете компьютер на ночь, то лучше всего запускать эту команду как задание для процесса cron.

В заключение хочется сказать, что страницы руководства man, вообще говоря, создавались не для первоначального изучения системы. Они скорее предназначены для опытных пользователей, которым в процессе работы нужно иметь под рукой справку по формату, опциям и синтаксису команд, чтобы не приходилось держать весь этот громоздкий материал в голове.

3.6.3. Команда *info*

Команда info является некоторой альтернативой команде man. Для получения информации по отдельной команде надо задать в командной строке info с параметром, являющимся именем интересующей вас команды, например,

```
[user]$ info man.
```

Информация, которую вы увидите, в большинстве случаев несколько отличается от той, которую дает команда man, причем, по моему мнению, в лучшую сторону. Но самое существенное отличие заключается в том, что выдаваемая info информация представлена в гипертекстовом формате. В силу этого вы получаете возможность просматривать различные разделы помощи, не выходя из оболочки, предоставляемой командой info. Работая в текстовом режиме, вы можете запустить info в одной из альтернативных консолей (помните:

<Ctrl>+<Alt>+<F2>, <Ctrl>+<Alt>+<F3> и т. д.) и переключаться за помощью в случае необходимости. В тех случаях, когда вы не знаете, где именно найти нужную информацию, может оказаться полезным побродить по разным разделам текста с помощью гипертекстовых ссылок, предоставляемых командой info. Эти ссылки обозначены символом звездочки (*), что несколько отличается от способа обозначения гипертекстовых ссылок в широко распространенных браузерах типа Internet Explorer или Netscape Navigator, но от этого не становится менее удобным. Перемещаться по ссылкам можно также с помощью клавиши <Tab>. Достигнув названия нужной темы, нажмите клавишу <Enter>. Нажатие клавиши <P> возвращает вас к предыдущей странице, <N> вызывает переход на следующую страницу, а <U> переводит на один уровень вверх по иерархической структуре страниц документации.

Кроме того, можно вызвать переход по ссылке другим способом, аналогичным системе меню. Для этого надо нажать клавишу <M> и набрать в появившейся внизу экрана строке ввода некоторое число начальных символов названия нужного вам раздела помощи (из числа названий, представленных на отображаемой в данный момент на экране странице, причем даже если не вся страница помещается на экране). Число символов должно быть достаточным для однозначного определения раздела помощи (если недостаточно, то программа попросит дополнить название). Выход из программы — по клавише <Q>.

3.6.4. Команда *help*

Выше уже упоминалась система помощи по встроенным командам оболочки bash — команда help. Если ввести в командной строке help без параметров, вы получите список всех встроенных команд оболочки. Если ввести команду help name, где name — имя одной из этих команд, то вы получите очень краткую справку о применении этой команды.

3.6.5. Документация, поставляемая с дистрибутивом и пакетами ПО

Если в процессе установки системы вы не отказались от установки документации, то после завершения процедур инсталляции в каталоге /usr/doc/ (или /usr/share/doc) вы найдете подкаталоги HOWTO, FAQ, HTML и LDP, содержащие обширнейшую документацию по системе Linux в целом и отдельных аспектах ее применения. Большая часть этой документации представляет собой обычные текстовые ASCII-файлы, которые можно просматривать **ПО командам** more filename **ИЛИ** less filename, **а также С ПОМОЩЬЮ** встроенной программы просмотра, включенной в оболочку Midnight Commander. Просмотр этих файлов был для меня основным источником получения информации при освоении Linux (а, значит, и при подготовке данной книги). Начните с Red Hat Reference Guide (/doc/ref-guide на компакт-диске). Осо-

бенно стоит обратить внимание на главы Package Management with RPM и System Administration. Также полезно прочесть содержимое /usr/doc/initscripts-x.xx. К сожалению, большая часть этой документации написана на английском языке, но я надеюсь, что со временем с русифицированными дистрибутивами эта документация будет поставляться на русском.

Большинство пакетов программного обеспечения поставляются разработчиками с обширной документацией по установке и использованию этих пакетов. Если пакет представлен в формате RPM (а дистрибутивы Red Hat Linux и его клонов типа Black Cat поставляются в этом формате), то эта документация будет развернута в соответствующих подкаталогах каталога /usr/doc. Имена этих подкаталогов соответствуют названию пакета и версии ПО. Например, для графической оболочки KDE версии 1.1.1 создается подкаталог KDE-1.1.1.

Иногда в поиске нужного файла документации может помочь команда locate. Команда locate в некотором смысле аналогична командам whatis и apropos. По этой команде производится поиск всех файлов, имена которых содержат заданный шаблон. Например, по команде locate net будет найдена масса имен файлов, в названиях которых встречается подстрока "net". В шаблоне могут применяться метасимволы *, ? , []. Однако команда locate производит поиск не по каталогам файловой системы, а в специально созданной базе имен файлов, которую надо вначале создать (и иногда **обновлять) Командой** updatedb.

В некоторых дистрибутивах (например, в ALTLinux) вместо locate имеется команда slocate, которая сама создает для себя базу имен файлов (после запуска с соответствующим параметром).

3.6.6. Команда *Xman*

В 6-ой версии дистрибутива Red Hat Linux была включена программа, которая позволяет просмотреть страницы руководства man при работе в графическом режиме. Поиск и вывод на экран страниц руководства вызывается посредством щелчков по кнопкам и меню. В остальном (по выдаваемой информации) Xman аналогична команде man.

3.6.7. Команда *helptool*

По команде helptool появляется графическое окно, имеющее строку ввода, в которой вы сможете задать интересующий вас термин. Команда просматривает все файлы документов (вы можете сконфигурировать, какие документы следует просматривать при поиске). По завершении поиска вам будет выдан список файлов, где встречается данный термин. Если щелкнуть мышкой на элементе списка, то появится дополнительное окно, в котором будет отображаться выбранный вами файл. При этом файл будет отобра-

жаться в том формате, в котором он хранится на вашей машине: страницы info, страницы man и др.

3.6.8. Книги и Интернет

Естественно, что осваивать Linux проще и легче, если под рукой имеется хорошая книга. Прежде всего почитайте руководство к своему дистрибутиву (если таковое есть). Несколько наиболее полезных, на мой взгляд, книг указаны в *приложении* в конце книги. И, конечно, если у вас есть выход в Интернет, то вы можете найти ответ на любой из возникающих вопросов. Просмотрите *приложение* для того, чтобы выбрать начальные точки путешествия по этому безбрежному океану информации. А дальше... остается пожелать вам удачи в ваших поисках.

Если проблемы возникают еще при установке, можно обратиться в службу поддержки производителя дистрибутива (если дистрибутив не пиратский, на нем как правило указываются координаты этой службы), на которую можно выйти через Web-сайт производителя. Учтите, что в случае, если у вас пиратский дистрибутив, причина проблем может быть в том, что он "криво" записан на компакт-диск.

Используйте возможности электронной почты. Полезно подписаться на некоторые списки рассылки (эхо-конференции). Например, чтобы подписаться на список рассылки по дистрибутиву Black Cat Linux, надо отправить письмо, содержащее "subscribe blackcat-list", по адресу **majordomo@geon.donetsk.ua.** Правда, чтение всех писем требует очень большого времени, т. к. придется читать множество вопросов начинающих пользователей (например, "Что такое консоль?"), а также многочисленные ответы на них от тех, кто хоть что-то знает. Поэтому можно просто просматривать с помощью браузера архив списка рассылки: может быть, кто-то уже задавал тот вопрос, ответ на который вы ищете. Весьма вероятно, что ваша проблема уже обсуждалась – если это так, то вы получите более быстрый и полный ответ, вовсе не написав письмо в конференцию.

Ну, а самый эффективный способ — это, не стесняясь своего незнания, задать свой вопрос в конференцию. Вы почти гарантированно получите ответ на него и, если не поймете, можете попросить дать дополнительные пояснения.

Если вы задаете вопрос, связанный с вашей конкретной системой, всегда включайте как можно больше деталей — какой дистрибутив, какая версия дистрибутива, какая версия ядра, с какими именно "железками" у вас проблемы (опять же версии, надписи на микросхемах), какие сообщения выдает система или запускаемая программа, и т. д. Только старайтесь не требовать, чтобы ответ на вопрос был прислан непосредственно на ваш электронный адрес. Как заметил Виктор Вагнер, "написание писем в эху есть самовыражение, написание писем нетмейлом есть техническая поддержка. Первое бесплатно, второе платно." Подумайте об этом!

Глава 4

Знакомство с файловой системой ext2fs

Теперь, когда вы научились запускать Linux и завершать работу с этой системой, надо познакомиться с устройством одной из основных ее частей — файловой системы. Файловая система — это структура, с помощью которой ядро операционной системы предоставляет пользователям (и процессам) ресурсы долговременной памяти системы, т. е. памяти на различного вида долговременных носителях информации — жестких дисках, магнитных лентах. CD-ROM и т. п.

Подобно луне, которая обращена к нам всегда одной стороной, файловая система тоже обращена к пользователю (может быть, лучше сказать — к приложениям) постоянно одной стороной. С этой, видимой для пользователей стороны, файловая система выглядит как логическая структура каталогов и файлов. Но у нее есть и другая сторона, обращенная к носителям, образующая внутреннее (с точки зрения пользователя) устройство файловой системы. Эта невидимая сторона файловой системы устроена далеко не просто. Дело в том, что она реализует механизмы записи файлов на различные носители, алгоритмы доступа (выборки нужной информации) и многое другое.

В настоящей главе мы рассмотрим файловые системы только с той стороны, которая обращена к пользователям. Обратную, невидимую для пользователей, сторону файловой системы мы будем изучать в *гл. 16.* Надо еще, может быть, отметить, что речь пойдет конкретно о файловой системе типа ext2fs, основном на данный момент типе файловых систем для Linux (существуют и другие типы файловых систем, об этом тоже будет сказано в *гл. 16).*

4.1. Файлы и их имена

Компьютер есть не что иное, как инструмент для обработки информации. А информация в любой ОС хранится на носителях в виде файлов. С точки зрения ОС файл представляет собой непрерывный поток (или последовательность) байтов определенной длины. Внутренний формат файла операционную систему не интересует. Но ОС должна дать файлу какое-то имя, с помощью которого пользователь, а точнее, программы-приложения, будут обращаться к файлу. Как организовать это обращение — дело файловой системы, пользователя это чаще всего не интересует. Поэтому с точки зре-

ния пользователя файловая система выглядит как логическая структура каталогов и файлов.

Имена файлов в Linux могут иметь длину до 255 символов и состоять из любых символов, кроме символа с кодом 0 и символа / (слэша). Однако имеется еще ряд символов, которые имеют в оболочке shell специальное значение и которые поэтому не рекомендуется включать в имена. Это следующие символы:

```
! @ # $ & ~ % * ( ) [ ] { } ' " \ : ; > < ` Пробел.
```

Если имя файла содержит один из этих символов (это не рекомендуется, но возможно), то вы должны перед этим символом поставить символ обратного слэша \ (в том числе и перед самим этим слэшем, т. е. повторить его дважды).

```
[user]$ mkdir \\my\&his
```

Можно также заключить имя файла или каталога с такими символами в двойные кавычки. Например, для создания каталога с именем My old files следует использовать команду:

```
[user]$ mkdir "My old files"
```

так как команда

```
[user]$ mkdir My old files
```

создаст каталог с именем му.

Аналогичным образом можно поступать и с другими символами, перечисленными выше, т. е. их можно включать в имена файлов, если имя файла взять в двойные кавычки или отменить специальное значение символа с помощью обратного слэша. Но все же предпочтительнее не использовать эти символы, включая пробел, в именах файлов и каталогов, потому что могут возникнуть проблемы при обращении к таким файлам из некоторых приложений, а также при переносе таких файлов в другие файловые системы.

Но к точке сказанное не относится, и в Linux часто ставят более одной точки в именах файлов, например, This_is.a.forth-chapter_of_my_book.about.Linux. При этом теряет смысл такое понятие (принятое в DOS), как расширение имени файла, хотя все же часто последние части имени, отделенные точками, используют для обозначения файлов каких-то особых типов (например, .tar.gz используется для обозначения сжатых архивов). Но исполняемые и неисполняемые файлы в Linux распознаются не по расширениям имен файлов. Для этого существуют другие признаки, о которых мы скажем чуть позже. Точка имеет особое значение в именах файлов. Если она является первым символом имени, то данный файл считается скрытым для некоторых команд, например, он не показывается при выполнении команды `ls`.

В Linux различаются символы верхнего и нижнего регистра в именах файлов. Поэтому FILENAME.tar.gz и filename.tar.gz вполне могут существовать одновременно и являться именами разных файлов.

Мы привыкли считать, что файл полностью определяется его именем. Однако с точки зрения ОС и файловой системы это немного не так (точнее, совсем не так). Хотя мы будем говорить о внутреннем устройстве файловой системы только в конце книги *(гл. 16)*, кое-что надо сказать уже сейчас.

Каждому файлу в Linux соответствует так называемый "индексный дескриптор" файла, или "inode", (однозначного перевода этого термина на русский язык не существует, в разных книгах эту структуру называют по-разному). Именно индексный дескриптор содержит всю необходимую файловой системе информацию о файле, включая информацию о расположении частей файла на носителе, типе файла и многое другое. Индексные дескрипторы файлов содержатся в специальной таблице (inode table), которая формируется при создании файловой системы на носителе. Каждый логический и физический диск имеет собственную таблицу индексных дескрипторов. Дескрипторы в этой таблице пронумерованы последовательно, и именно номер дескриптора файла является его истинным именем в системе (этот номер мы будем называть индексом файла). Однако для человека такая система имен неудобна. Сможете ли вы вспомнить, что сохранили в файле с номером 56734? Поэтому файлам даются еще "человеческие" имена, и помимо этого файлы группируются в каталоги.

Приведенная выше информация нужна здесь только для того, чтобы сказать, что имя любого файла в Linux является ни чем иным, как ссылкой на индексный дескриптор файла. Поэтому каждый файл может иметь сколько угодно разных имен. Эти имена называют еще "жесткими" ссылками. Когда вы удаляете файл, имеющий несколько разных имен — жестких ссылок, то фактически удаляется только одна ссылка — та, которую вы указали в команде удаления файла. Даже когда вы удаляете последнюю ссылку, это еще может не означать удаления содержимого файла — если файл еще используется системой или каким-то приложением, то он сохраняется до тех пор, пока он не "освободится".

Для того чтобы дать файлу (или каталогу) дополнительное имя (создать жесткую ссылку), используется команда `ln` в следующем формате:

```
ln имя_существующего_файла новое_имя
```

Пример:

```
[user]$ ln /home/howto/font-HOWTO-ru/Font-HOWTO.html ~/fonts.html
```

(специальный символ ~ здесь и вообще в системе означает домашний каталог пользователя, о котором будет сказано чуть дальше). Теперь можно вместо ДЛИННОГО имени `/home/howto/font-HOWTO-ru/Font-HOWTO.html` ИСПОЛЬЗовать просто `~/fonts.html`. Подробнее о команде `ln` вы можете прочитать на странице интерактивного руководства man.

Число жестких ссылок на файл (то есть разных имен файла) можно узнать, выполнив команду `ls` с параметром -l. Сразу за перечислением прав доступа к файлу следует число, которое и обозначает число жестких ссылок на файл:

```
[user]# ls -l
total 9
drwxr-xr-x  2  user  users   1024  Jul  1  2000  Autostart
-rw-r--r--  1  user  users    230  Sep 14  1999  Printer.kdelnk
-rw-r--r--  1  user  users    159  Sep 15  1999  Red Hat
```

4.2. Каталоги

Если бы файловая структура не позволяла использовать ничего кроме просто имен файлов, даже сколь угодно длинных (то есть все файлы располагались бы в одном общем списке), то обращаться к ним было бы чрезвычайно трудно. Вообразите себе список из нескольких тысяч файлов! Поэтому файлы группируются в каталоги, которые, в свою очередь, могут быть включены в другие каталоги. В результате получается иерархическая структура каталогов, начинающаяся с корневого каталога. Каждый (под)каталог может содержать как отдельные файлы, так и подкаталоги.

Иерархическую структуру каталогов обычно иллюстрируют рисунком "дерева каталогов", в котором каждый каталог изображается узлом "дерева", а файлы — "листьями". В MS Windows или DOS каталоговая структура строится отдельно для каждого физического носителя (то есть имеем не отдельное "дерево", а целый "лес") и корневой каталог каждой каталоговой структуры обозначается какой-нибудь буквой латинского алфавита (отсюда уже возникает некоторое ограничение). В Linux (и UNIX вообще) строится единая каталоговая структура для всех носителей, и единственный корневой каталог этой структуры обозначается символом "/". В эту единую каталоговую структуру можно подключить любое число каталогов, физически расположенных на разных носителях (как говорят, "смонтировать файловую систему" или "смонтировать носитель").

Имена каталогов строятся по тем же правилам, что и имена файлов. И, вообще, каталоги в принципе ничем, кроме своей внутренней структуры (до которой ОС уже есть дело), не отличаются от "обычных" файлов, например, текстовых.

Полным именем файла (или путем к файлу) называется список имен вложенных друг в друга подкаталогов, начинающийся с корневого каталога и оканчивающийся собственно именем файла. При этом имена подкаталогов в этом списке разделяются тем же символом /, который служит для обозначения корневого каталога. Например, на моем компьютере /home/kos/ve/book/filesysteml.htm является полным именем того файла, в котором я сохранил первый вариант данного текста.

В каждый момент времени пользователь работает с одним экземпляром оболочки shell и эта оболочка хранит значение так называемого "текущего" каталога, т. е. того каталога, в котором пользователь сейчас работает. Имеется специальная команда, которая сообщает вам значение текущего каталога — pwd.

Примечание

Если быть более точным, то следует сказать, что текущий каталог — это понятие, относящееся к каждому запущенному в системе процессу (в частности к оболочке); поэтому иногда запуск какой-то программы в shell может привести к тому, что после завершения работы этой программы текущий каталог сменится.

Кроме текущего каталога для каждого пользователя определен еще его "домашний каталог" — каталог, в котором пользователь имеет все права: может создавать и удалять файлы, менять права доступа к ним и т. д. В каталоговой структуре Linux домашние каталоги пользователей обычно размещаются в каталоге /home и имеют имена, совпадающие с именем пользователя. Например, /home/jim. Каждый пользователь может обратиться к своему домашнему каталогу с помощью значка ~, т. е. например, пользователь jim может обратиться к каталогу /home/jim/doc как к ~/doc. Когда пользователь входит в систему, текущим каталогом становится домашний каталог данного пользователя.

Для изменения текущего каталога служит команда cd. В качестве параметра этой команде надо указать полный или относительный путь к тому каталогу, который вы хотите сделать текущим. Понятие полного пути уже было пояснено, а понятие относительного пути требует дополнительного пояснения. Относительным путем называется перечисление тех каталогов, которые нужно пройти в "дереве каталогов", чтобы перейти от текущего каталога к какому-то другому каталогу (назовем его целевым). Если целевой каталог, т. е. каталог, который вы хотите сделать текущим, расположен ниже текущего в структуре каталогов, то сделать это просто: вы указываете сначала подкаталог текущего каталога, затем подкаталог того каталога и т. д., вплоть до имени целевого каталога. Если же целевой каталог расположен выше в каталоговой структуре, или вообще на другой "ветви" дерева, то ситуация несколько сложнее. Конечно, можно было бы пользоваться полным путем, но тогда придется записывать очень длинные маршруты.

Эта трудность преодолевается следующим образом. Для каждого каталога (кроме корневого) в дереве каталогов однозначно определен "родительский каталог". В каждом каталоге имеются две особых записи. Одна из них обозначается просто точкой и является указанием на этот самый каталог, а вторая запись, обозначаемая двумя точками, — указатель на родительский каталог. Эти имена из двух точек и используются для записи относительных путей. Чтобы сделать текущим родительский каталог, достаточно дать команду

```
[user]$ cd ..
```

А чтобы перейти по дереву каталогов на два "этажа" вверх, откуда спуститься в подкаталог kat1/kat2, надо дать команду

```
[user]$ cd ../../kat1/kat2
```

Команда `ls` служит для вывода на экран списка имен файлов и подкаталогов текущего каталога. Нужно отметить, что фактически команда `ls` просто выводит содержимое файла, который описывает данный каталог, и не происходит никаких обращений к самим файлам. Любой каталог, как уже говорилось, — это обычный файл, в котором перечислены все файлы и подкаталоги этого каталога.

Примечание

Задумайтесь, кстати: нет никаких особых "ящиков с файлами", есть просто файлы-списки, которые причисляют данный файл к определенному каталогу.

Если дать команду `ls` без параметров, то выводятся только имена файлов текущего каталога. Если нужно просмотреть содержимое не текущего, а какого-то другого каталога, надо указать команде `ls` полный или относительный путь к этому каталогу.

Кроме имени файла (или подкаталога) запись о нем в соответствующем каталоге содержит еще массу информации об этом файле. Для того чтобы получить эту информацию, надо использовать дополнительные параметры команды `ls`. Если дать команду `ls` с параметром -1, то будут выданы не только имена файлов, но также данные о правах доступа к файлу (подробнее о правах будет рассказано ниже), количество жестких ссылок или имен файла (для каталога указывается число дополнительных блоков), имя владельца файла и группы файла, его размер и дата последней модификации. Вот небольшой пример.

```
[user]$ ls -1
итого 1171
drwxrwxr-x    2  kos  kos    1024  Jun 20  22:42  NotR
drwx------    2  kos  kos    1024  Jun 27  21:02  Star
-rw-rw-r--    1  kos  kos   17351  Nov 2   23:59   arch.htm
-rw-rw-r--    1  kos  kos   19847  Dec 11  20:23  contents.htm
-rw-rw-r--    1  kos  kos   48866  Nov 2   23:59  edit.htm
-rw-rw-r--    1  kos  kos   38867  Dec 12  20:58  filesystem1.htm
-rw-rw-r--    1  kos  kos   29545  Dec 11  20:23  first_start.htm
drwxr-xr-x    3  kos  kos    2048  Sep 24  21:33  img
-rw-rw-r--    1  kos  kos   21590  Dec 11  19:42  init.htm
drwxrwxr-x    2  kos  kos    1024  Sep 27  22:35  pic
-rw-rw-r--    1  kos  kos   11084  Nov 8   21:26  preface.htm
```

Если дополнительно задать параметр -i, то в первой колонке будут отображены индексы файлов (номера соответствующих inode). При задании параметра -t сортировка файлов будет производиться не по именам, а по времени модификации файла. Задание параметра -и приводит к тому, что вместо времени модификации файла будет выводиться время последнего доступа к файлу. Параметр `-r` меняет порядок сортировки на обратный (используется вместе с параметрами -1 и -t). Заметим еще, что параметры можно перечислять как отдельно:

```
[user]$ ls -l -i — r
```

так и объединять:

```
[user]$ ls -lir
```

На этом мы закончим краткое описание команды `ls` (подробнее см. соответствующую man- или info-страницу) и перейдем к рассмотрению основных каталогов в каталоговой структуре Linux.

4.3. Назначение основных системных каталогов

Если вы работали, например, с Windows 95, то вы знаете, что, хотя пользователь имеет полную свободу в организации структуры каталогов, некоторые "обычаи" все же сохраняются. Так, системные файлы располагаются обычно в подкаталоге C:\Windows, вновь устанавливаемые программы по умолчанию размещаются в каталоге C:\Program Files и т. д. В Linux типовая структура каталогов выдерживается, пожалуй, даже более строго. Более того, существует даже стандарт на структуру каталогов для UNIX-подобных ОС, так называемый Filesystem Hierarchy Standart (FHS), полный текст которого можно найти по адресу **http://www.pathname.com/fhs/.** Дистрибутив Red Hat в основном придерживается стандарта FHS.

В табл. 4.1 дан краткий перечень основных стандартно создаваемых каталогов той файловой структуры, которая формируется при установке дистрибутива Red Hat (и его последователей).

В левом столбце перечислены подкаталоги корневого каталога, в среднем столбце — некоторые основные (далеко не все!) подкаталоги второго уровня, а в правом столбце даны краткие пояснения о назначении всех этих каталогов. Пояснения по необходимости очень краткие, более подробно с основными каталогами вы можете познакомиться по тексту стандарта FHS (**http://www.pathname.com/fhs/**).

Таблица 4.1. *Структура каталогов Red Hat Linux*

Каталог	Подкаталоги	Назначение
/bin		Этот каталог содержит в основном готовые к исполнению программы, большинство из которых необходимо во время старта системы (или в однопользовательском системном режиме, используемом для отладки). Здесь хранится значительное количество общеупотребительных команд Linux
/boot		Содержит основные постоянные файлы для загрузки системы, в частности загружаемое ядро. Файлы из этого каталога нужны только во время загрузки системы
/dev		Каталог специальных файлов или файлов устройств. О них мы поговорим чуть подробнее в одном из следующих разделов. Можете также заглянуть в man `mknod(1)`
/etc		Этот каталог и его подкаталоги содержат большинство данных, необходимых для начальной загрузки системы, и основные конфигурационные файлы. В /etc находятся, например, файл inittab, определяющий загружаемую конфигурацию и файл паролей пользователей passwd. Часть конфигурационных файлов может находиться и в /usr/etc. Каталог /etc не должен содержать двоичных файлов (их следует перенести в /bin или /sbin). Ниже приводится назначение основных (но далеко не всех!) подкаталогов каталога /etc
	/etc/rc.d	Этот подкаталог содержит файлы, которые используются в процессе начальной загрузки системы. Подробнее о них и вообще о процессе загрузки будет рассказано в *разд. 8.2*
	/etc/skel	Когда создается новый пользователь и account для него, то файлы из этого каталога копируются во вновь созданный домашний каталог пользователя
	/etc/sysconfig	Каталог, содержащий некоторые (но не все) конфигурационные файлы системы
	/etc/X11	Каталог для конфигурационных файлов подсистемы X11 (например, XF86Config)
/home		Обычно в этом каталоге находятся каталоги пользователей
/lib		Этот каталог содержит разделяемые библиотеки функций, необходимых компилятору языка С, и модули (драйверы устройств). Даже если в системе не установлен компилятор языка С, разделяемые библиотеки необходимы, поскольку они используются многими прикладными программами. Они загружаются в память по мере необходимости выполнения каких-то функций, что позволяет уменьшить объем кода программ — в противном случае один и тот же код многократно повторялся бы в различных программах

Таблица4.1 (продолжение)

Каталог	Подкаталоги	Назначение
/lost+found		Этот каталог используется при восстановлении файловой системы командой fsck. Если fsck обнаруживает файл, родительский каталог которого определить невозможно, она помещает такой файл в каталог /lost+found. Поскольку родительский каталог потерян, то таким файлам присваиваются имена, совпадающие с номерами их индексных дескрипторов
/mnt		Это точка монтирования для временно монтируемых файловых систем. Если на компьютере запускается поочередно Linux и MS-DOS, то этот каталог обычно используется, чтобы монтировать файловую систему MS-DOS. Если вы имеете привычку монтировать несколько дополнительных носителей, например, дискеты, CD-ROM, дополнительный жесткий диск и т. д., то можно создать в нем соответственно дополнительные подкаталоги для каждого носителя
/proc		Это точка монтирования для файловой системы proc, которая обеспечивает информацию о выполняющихся процессах, ядре, оборудовании вычислительной установки и т. д. Это псевдофайловая система, подробности о которой можно узнать по команде man 5 proc. Специальные файлы из этого каталога используются для получения и передачи данных ядру
/root		Это домашний каталог суперпользователя. Обратите внимание на то, что он расположен не там, где располагаются личные каталоги остальных пользователей (в /home)
/sbin		Подобно каталогу /bin, содержит в основном исполняемые файлы — программы и утилиты ОС, используемые в процессе загрузки и запускаемые системным администратором. В стандарте FHS говорится, что в этот каталог надо помещать те исполняемые файлы, которые используются после успешного подключения файловой системы /usr. Минимальное содержимое этого каталога включает программы clock, getty, init, update, mkswap, swapon, swapoff, halt, reboot, shutdown, fdisk, fsck.*, mkfs.*, lilo, arp, ifconfig, route
/tmp		Каталог для временных файлов. В любой момент суперпользователь может удалить файлы из этого каталога без большого ущерба для остальных пользователей. Однако не стоит удалять файлы из этого каталога, если вам не стало ясно, что конкретный файл или группа файлов мешают продолжению продуктивной работы на машине. Система сама периодически очищает этот каталог, поэтому не следует хранить тут файлы, которые вам могут понадобиться в дальнейшем

Таблица 4.1 (продолжение)

Каталог	Подкаталоги	Назначение
/usr		Этот каталог огромен и его структура в основном повторяет структуру корневого каталога. В его подкаталогах находятся все основные приложения. В соответствии со стандартом FHS рекомендуется выделять для этого каталога отдельный раздел диска или вообще располагать его на сетевом диске, общем для всех компьютеров в сети. Такой раздел или диск монтируют только для чтения и располагают в нем общие конфигурационные и исполняемые файлы, документацию, системные утилиты и библиотеки, а также включаемые файлы (файлы типа include)
	/usr/bin	Готовые к исполнению программы — утилиты и приложения, которые часто вызывают обычные пользователи. /usr/bin/X11 — обычное место для расположения готовых к исполнению программ из XWindow в Linux. Часто это символическая ссылка на /usr/X11 R6/bin
	/usr/dict	Этот каталог содержит файлы со словарным запасом для программ проверки корректности написания слов
	/usr/etc	Здесь содержатся конфигурационные файлы для группы машин. Однако команды и программы должны смотреть в каталог /etc, в котором должны быть ссылки к файлам в каталоге /usr/etc
	/usr/include	Этот каталог содержит исходный код стандартных библиотек языка С, подставляемый в программы директивой препроцессора include. Поэтому пользователю надо иметь, по крайней мере, право на чтение из этого каталога. Ни в коем случае не следует модифицировать файлы в этом каталоге, потому что они тщательно отлажены разработчиком системы (разве что вы знаете систему лучше разработчика)
	/usr/lib	В данном каталоге содержатся объектные библиотеки подпрограмм, динамические библиотеки, некоторые готовые к исполнению программы, которые не вызываются непосредственно. Сложные программные системы могут иметь свои подкаталоги в этом каталоге. • /usr/lib/X11 — обычное место для помещения файлов, связанных с X Window, а также конфигурационных файлов самой системы X Window. В Linux это обычно символическая ссылка на каталог /usr/X11 R6/lib/X11; • /usr/lib/gcc-lib — содержит готовые к исполнению программы и файлы типа include для компилятора GNU С (дсс); • /usr/lib/groff — файлы для системы форматирования текстов groff;

Таблица 4.1 (продолжение)

Каталог	Подкаталоги	Назначение
		• /usr/lib/uucp — файлы для UUCP; • usr/lib/zoneinfo — файлы для определения временной зоны. Смотрите также страницы руководств по named-xfer (8), tzfile (5), tzselect (8), zdump (8), zic (8)
	/usr/local	Обычно здесь помещают программы и подкаталоги, которые являются локальными (уникальными) для данной машины. • /usr/local/bin — обычно здесь помещают готовые к исполнению программы, которые являются локальными (уникальными) для данной машины; • /usr/local/doc — здесь располагается документация ко всем установленным на Вашем компьютере пакетам прикладного ПО; • /usr/local/etc — конфигурационные файлы для локально установленных программ; • /usr/local/lib — библиотеки и файлы для локально установленных программ и систем; • /usr/local/info — страницы описаний, которые просматриваются посредством программы info, для локально установленных программ; • /usr/local/man — страницы описаний, которые просматриваются посредством программы man, для локально установленных программ; • /usr/local/sbin — локальные программы системного администратора; • /usr/local/src — исходные тексты программ, установленных на данной машине
	/usr/man	Страницы интерактивного руководства man в исходном формате (не подготовленные для просмотра). /usr/man/<locale>/man[1-9] — эти каталоги содержат страницы руководств на различных языках (в зависимости от значения locale). Системы, которые используют один язык и один кодовый набор, могут не использовать подстроку <locale>
	/usr/sbin	Этот каталог содержит готовые к исполнению программы для системного администрирования, которые не используются во время загрузки
	/usr/src	Исходные тексты для различных частей Linux. /usr/src/linux — исходные тексты для ядра Linux

Таблица 4.1 (продолжение)

Каталог	Подкаталоги	Назначение
	/usr/tmp	Еще одно место для хранения временных файлов. Обычно это символическая ссылка на каталог /var/tmp
	/usr/X11R6	Файлы, относящиеся к системе X Window (версии 11, релиз 6). • /usr/X11R6/bin — готовые к исполнению программы системы X Window; • /usr/X11 R6/lib — файлы и библиотеки связанные с системой X Window
/var		Этот каталог содержит файлы, в которых сохраняются различные переменные данные, определяющие конфигурацию некоторых программ при следующем запуске или временно сохраняемую информацию, которая будет использоваться позже в ходе текущего сеанса. Объем данных в этом каталоге может сильно изменяться, поскольку он содержит, например, файлы протоколов (логи), файлы спулинга и блокировки (locking), временные файлы и т. д.
	/var/adm	Содержит учетную и диагностическую информацию, необходимую системному администратору
	/var/backups	Этот каталог используется, чтобы сохранить резервную копию важных системных файлов
	/var/catman/cat[1-9]	Этот каталог используется, чтобы хранить уже сформированные страницы руководств в соответствии с номером главы
	/var/lock	Здесь содержатся управляющие файлы системы, которые используются для резервирования использования тех или иных ресурсов системы
	/var/log	Различные файлы протоколов
	/var/run	Переменные файлы времени выполнения различных программ. Они содержат идентификаторы процессов (PIDs) и записывают текущую информацию (utmp). Файлы в этом каталоге обычно очищаются во время загрузки системы
	/var/spool	Файлы различных программ, поставленные в очередь на обслуживание. • /var/spool/at — файлы заданий, запущенных посредством команды at; • /var/spool/cron — файлы системы cron; • /var/spool/lpd — файлы, ожидающие вывода на печать

Таблица 4.1 (окончание)

Каталог	Подкаталоги	Назначение
		• /var/spool/mail — пользовательские почтовые ящикиж
		• /var/spool/news — файлы системы newsж
		• /var/spool/uucp — файлы системы uucp
	/var/tmp	Временные файлы

4.4. Типы файлов

В предыдущих разделах мы рассмотрели два типа файлов: обычные файлы и каталоги. Но в Linux существует еще несколько типов файлов. С ними мы познакомимся в этом разделе.

Как уже было сказано, с точки зрения операционной системы файл представляет собой просто поток байтов. Такой подход позволяет распространить концепцию файла на физические устройства и некоторые другие объекты. Это дает возможность упростить организацию данных и обмен ими, потому что аналогичным образом осуществляется запись данных в файл, передача их на физические устройства и обмен данными между процессами. Во всех этих случаях используется один и тот же подход, основанный на идее байтового потока. Поэтому наряду с обычными файлами и каталогами, файлами с точки зрения Linux являются также:

- файлы физических устройств;
- именованные каналы (named pipes);
- гнезда (sockets);
- символические ссылки (symlinks).

4.4.1. Файлы физических устройств

Как уже говорилось, с точки зрения ОС Linux, все подключаемые к компьютеру устройства (жесткие и съемные диски, терминал, принтер, модем и т. д.), представляются файлами. Если, например, надо вывести на экран какую-то информацию, то система как бы производит запись в файл /dev/tty01.

Физические устройства бывают двух типов: символьными (или байт-ориентированными) и блочными (или блок-ориентированными). Различие между ними состоит в том, как производится считывание и запись информации в эти устройства. Взаимодействие с символьными устройствами произво-

дится посимвольно, в режиме потока байтов. К таким устройствам относятся, например, терминалы. На блок-ориентированных устройствах информация записывается (и, соответственно, считывается) блоками. Примером устройств этого типа являются жесткие диски. На диск невозможно записать или считать с него один байт: обмен с диском производится только блоками.

Взаимодействием с физическими устройствами в Linux управляют драйверы устройств, которые либо встроены в ядро, либо подключаются к нему как отдельные модули. Для взаимодействия с остальными частями операционной системы каждый драйвер образует коммуникационный интерфейс, который выглядит как файл. Большинство таких файлов для различных устройств как бы "заготовлены заранее" и располагаются в каталоге /dev.

Если вы заглянете в каталог /dev, то увидите там огромное количество файлов физических устройств. ("Заглянуть в каталог" означает выполнить последовательно две команды `cd` и `ls`.) В табл. 4.2 приведена небольшая справка по именам наиболее часто используемых специальных файлов.

Таблица 4.2. Основные специальные файлы

Имя	Значение
/dev/console	Системная консоль, т. е. монитор и клавиатура, физически подключенные к компьютеру
/dev/hd	Жесткие диски с IDE-интерфейсом. Устройство /dev/hda1 соответствует первому разделу на первом жестком диске (/dev/hda), т. е. на диске, подключенном как Primary Master
/dev/sd	Жесткие диски с SCSI-интерфейсом
/dev/fd	Файлы дисководов для гибких дисков. Первому дисководу соответствует /dev/fd0, второму /dev/fd1
/dev/tty	Файлы поддержки пользовательских консолей. Название сохранилось с тех пор, когда к системе UNIX подключались телетайпы в качестве терминалов. В Linux эти файлы устройств обеспечивают работу виртуальных консолей (переключаться между которыми можно с помощью комбинаций клавиш <Alt>+<F1>—<Alt>+<F6>)
/dev/pty	Файлы поддержки псевдотерминалов. Применяются для удаленных рабочих сессий с использованием telnet
/dev/ttS	Файлы, обеспечивающие работу с последовательными портами. /dev/ttSO соответствует COM1 в MS-DOS, /dev/ttS1 - COM2. Если ваша мышь подключается через последовательный порт, то /dev/mouse является символической ссылкой на соответствующий /dev/ttSN
/dev/cua	Специальные устройства для работы с модемами
/dev/null	Это устройство — просто черная дыра. Все, что записывается в /dev/null, навсегда потеряно. На это устройство можно перенаправить вывод ненужных сообщений. Если /dev/null используется как устройство ввода, то оно ведет себя как файл нулевой длины

Каждому типу устройств в системе может соответствовать несколько файлов устройств. Поэтому файлы устройств характеризуются двумя номерами: старшим и младшим. Старший номер устройства говорит ядру о том, к какому драйверу относится данный файл, а младший номер показывает, к какому именно устройству данного типа следует обращаться.

Для файлов устройств команда `ls -l` вместо размера файла выдает старший и младший номера данного устройства.

4.4.2. Именованные каналы (pipes)

Еще один тип специальных файлов — именованные каналы, или буферы FIFO (First In — First Out). Файлы этого типа служат в основном для того, чтобы организовать обмен данными между разными приложениями *(pipe* переводится с английского как труба).

Канал — это очень удобное и широко применяемое средство обмена информацией между процессами. Все, что один процесс помещает в канал, другой может оттуда прочитать. Если два процесса, обменивающиеся информацией, порождены одним и тем же родительским процессом (а так чаще всего и происходит), канал может быть неименованным. В противном случае требуется создать именованный канал, что можно сделать с помощью программы mkfifo. При этом собственно файл именованного канала участвует только в инициации обмена данными.

4.4.3. Доменные гнезда (sockets)

Гнезда — это соединения между процессами, которые позволяют им взаимодействовать, не подвергаясь влиянию других процессов. Вообще гнезда (и взаимодействие программ при помощи гнезд) играют очень важную роль во всех UNIX-системах, включая и Linux: они являются ключевым понятием TCP/IP и соответственно на них целиком строится Интернет. Однако с точки зрения файловой системы гнезда практически неотличимы от именованных каналов: это просто метки, позволяющие связать несколько программ. После того как связь установлена, общение программ происходит без участия файла гнезда: данные передаются ядром ОС непосредственно от одной программы к другой.

Несмотря на то, что другие процессы могут видеть файлы гнезд как элементы каталога, процессы, не участвующие в данном конкретном соединении, не могут осуществлять над файлами гнезд операции чтения/записи. Среди стандартных средств, использующих гнезда, — система X Window, система печати и система syslog.

4.4.4. Символические ссылки (еще раз об именах файлов)

В разделе об именах файлов уже говорилось о том, что файл в Linux может иметь несколько имен или "жестких" ссылок.

Жесткая ссылка является просто еще одним именем для исходного файла. Она прописывается в индексном дескрипторе исходного файла. После создания жесткой ссылки невозможно различить, где исходное имя файла, а где ссылка. Если вы удаляете один из этих файлов (точнее одно из этих имен), то файл еще сохраняется на диске (пока у него есть хоть одно имя-ссылка).

Очень трудно различить первоначальное имя файла и позже созданные жесткие ссылки на него. Поэтому жесткие ссылки применяются там, где отслеживать различия и не требуется. Одно из применений жестких ссылок состоит в том, чтобы предотвратить возможность случайного удаления файла.

Особенностью жестких ссылок является то, что они прямо указывают на номер индексного дескриптора, а, следовательно, такие имена могут указывать только на файлы внутри той же самой файловой системы (то есть на том же самом носителе, на котором находится каталог, содержащий это имя).

Но в Linux имеется другой тип ссылок, так называемые символические ссылки. Эти ссылки тоже могут рассматриваться как дополнительные имена файлов, но в то же время они представляются отдельными файлами — файлами типа символических ссылок. В отличие от жестких ссылок символические ссылки могут указывать на файлы, расположенные в другой файловой системе, например, на монтируемом носителе, или даже на другом компьютере. Если исходный файл удален, символическая ссылка не удаляется, но становится бесполезной. Используйте символические ссылки в тех случаях, когда хотите избежать путаницы, связанной с применением жестких ссылок.

Создание любой ссылки внешне подобно копированию файла, но фактически как исходное имя файла, так и ссылка указывают на один и тот же реальный файл на диске. Поэтому, например, если вы внесли изменения в файл, обратившись к нему под одним именем, вы обнаружите эти изменения и тогда, когда обратитесь к файлу по имени-ссылке. Для того чтобы создать символическую ссылку, используется уже упоминавшаяся команда `ls` с дополнительной опцией -s:

```
ln -s имя_файла_или_каталога имя_ссылки
```

Пример:

```
[user]$ ln -s /home/kos/ve/HOWTO/font-HOWTO-ru/ ~/FONTS
```

После выполнения такой команды в моем домашнем каталоге появился подкаталог FONTS. Если теперь мы просмотрим список файлов в каталоге

/home/kos с помощью команды `ls -l`, то среди прочих увидим такую строку:

```
lrwxrwxrwx 1 kos kos 31 Dec 13 21:13 FONTS -> /home/kos/ve/HOWTO/font-HOWTO-ru/
```

Обратите внимание на самый первый символ в этой строке: он показывает, что данная запись соответствует символической ссылке. Впрочем, это видно и в поле имени, где после нового имени и стрелки указано исходное имя файла (в данном случае — каталога).

Если вы создали в каталоге kat1 символическую ссылку, которая указывает на какой-то другой каталог, то вы можете переместить каталог kat1 куда угодно, символическая ссылка при этом будет оставаться корректной. Точно так же можно перемещать сами символические ссылки. Но остерегайтесь использовать ".." (то есть ссылку на родительский каталог) в полных именах файлов, включающих символические ссылки, поскольку по символической ссылке нельзя проследовать в обратном направлении, а ".." всегда означает истинный родительский каталог данного каталога.

4.5. Права доступа к файлам и каталогам

Поскольку Linux — система многопользовательская, вопрос об организации разграничения доступа к файлам и каталогам является одним из существенных вопросов, которые должна решать операционная система. Механизмы разграничения доступа, разработанные для системы UNIX в 70-х годах (возможно, впрочем, они предлагались кем-то и раньше), очень просты, но они оказались настолько эффективными, что просуществовали уже более 30 лет и по сей день успешно выполняют стоящие перед ними задачи.

В основе механизмов разграничения доступа лежат имена пользователей и имена групп пользователей. Вы уже знаете, что в Linux каждый пользователь имеет уникальное имя, под которым он входит в систему (логируется). Кроме того, в системе создается некоторое число групп пользователей, причем каждый пользователь может быть включен в одну или несколько групп. Создает и удаляет группы суперпользователь, он же может изменять состав участников той или иной группы. Члены разных групп могут иметь разные права по доступу к файлам, например, группа администраторов может иметь больше прав, чем группа программистов.

В индексном дескрипторе каждого файла записаны имя так называемого владельца файла и группы, которая имеет права на этот файл. Первоначально, при создании файла его владельцем объявляется тот пользователь, который этот файл создал. Точнее — тот пользователь, от чьего имени запущен процесс, создающий файл. Группа тоже назначается при создании файла — по идентификатору группы процесса, создающего файл. Владельца и группу

файла можно поменять в ходе дальнейшей работы с помощью команд chown и chgrp (подробнее о них будет сказано чуть позже).

Теперь давайте еще раз выполним команду `ls -l`. Но зададим ей в качестве дополнительного параметра имя конкретного файла, например, файла, задающего саму команду `ls`. (Обратите, кстати, внимание на эту возможность команды `ls` -1 — получить информацию о конкретном файле, а не о всех файлах каталога сразу).

```
[user]$ ls -l /bin/Is
-rwxr-xr-x   1   root   root   49940   Sep 12  1999   /bin/Is
```

Вы видите, что в данном случае владельцем файла является пользователь root и группа root. Но нас сейчас в выводе этой команды больше интересует первое поле, определяющее тип файла и права доступа к файлу. Это поле в приведенном примере представлено цепочкой символов -rwxr-xr-x. Эти символы можно условно разделить на 4 группы.

Первая группа, состоящая из единственного символа, определяет тип файла. Этот символ в соответствии с возможными типами файлов, рассмотренными в предыдущем разделе, может принимать такие значения:

- O - — обычный файл;
- O `d` — каталог;
- O ь — файл блочного устройства;
- ❒ c — файл символьного устройства;
- ❒ s — доменное гнездо (socket);
- ❒ p — именованный канал (pipe);
- ❒ 1 — символическая ссылка (link).

Далее следуют три группы по три символа, которые и определяют права доступа к файлу соответственно для владельца файла, для группы пользователей, которая сопоставлена данному файлу, и для всех остальных пользователей системы. В нашем примере права доступа для владельца определены как rwx, что означает, что владелец (root) имеет право читать файл (`r`), производить запись в этот файл (w), и запускать файл на выполнение (x). Замена любого из этих символов прочерком будет означать, что пользователь лишается соответствующего права. В том же примере мы видим, что все остальные пользователи (включая и тех, которые вошли в группу root) лишены права записи в этот файл, т. е. не могут файл редактировать и вообще как-то изменять.

Вообще говоря, права доступа и информация о типе файла в UNIX-системах хранятся в индексных дескрипторах в отдельной структуре, состоящей из двух байтов, т. е. из 16 битов (это естественно, ведь компьютер оперирует битами, а не символами `r`, w, x). Четыре бита из этих 16-ти отве-

дены для кодированной записи о типе файла. Следующие три бита задают особые свойства исполняемых файлов, о которых мы скажем чуть позже. И, наконец, оставшиеся 9 битов определяют права доступа к файлу. Эти 9 битов разделяются на 3 группы по три бита. Первые три бита задают права пользователя, следующие три бита — права группы, последние 3 бита определяют права всех остальных пользователей (то есть всех пользователей, за исключением владельца файла и группы файла).

При этом, если соответствующий бит имеет значение 1, то право предоставляется, а если он равен 0, то право не предоставляется. В символьной форме записи прав единица заменяется соответствующим символом (r, w или x), а 0 представляется прочерком.

Право на чтение (r) файла означает, что пользователь может просматривать содержимое файла с помощью различных команд просмотра, например, командой more или с помощью любого текстового редактора. Но, подредактировав содержимое файла в текстовом редакторе, вы не сможете сохранить изменения в файле на диске, если не имеете права на запись (w) в этот файл. Право на выполнение (x) означает, что вы можете загрузить файл в память и попытаться запустить его на выполнение как исполняемую программу. Конечно, если в действительности файл не является программой (или скриптом shell), то запустить этот файл на выполнение не удастся, но, с другой стороны, даже если файл действительно является программой, но право на выполнение для него не установлено, то он тоже не запустится.

Вот мы и узнали, какие файлы в Linux являются исполняемыми! Как видите, расширение имени файла тут ни при чем, все определяется установкой атрибута "исполняемый", причем право на исполнение может быть предоставлено не всем!

Если выполнить ту же команду `ls -l`, но в качестве последнего аргумента ей указать не имя файла, а имя каталога, мы увидим, что для каталогов тоже определены права доступа, причем они задаются теми же самыми символами `rwx`. Например, выполнив команду `ls -l /`, мы увидим, что каталогу bin соответствует строка:

```
drwxr-xr-x   2   root   root    2048    Jun 21  21:11    bin
```

Естественно, что по отношению к каталогам трактовка понятий "право на чтение", "право на запись" и "право на выполнение" несколько изменяется. Право на чтение по отношению к каталогам легко понять, если вспомнить, что каталог — это просто файл, содержащий список файлов в данном каталоге. Следовательно, если вы имеете право на чтение каталога, то вы можете просматривать его содержимое (этот самый список файлов в каталоге). Право на запись тоже понятно — имея такое право, вы сможете создавать и удалять файлы в этом каталоге, т. е. просто добавлять в каталог или удалять из него запись, содержащую имя какого-то файла и соответствующие ссылки. Право на выполнение интуитивно менее понятно. Оно в данном случае оз-

начает право переходить в этот каталог. Если вы, как владелец, хотите дать доступ другим пользователям на просмотр какого-то файла в своем каталоге, вы должны дать им право доступа в каталог, т. е. дать им "право на выполнение каталога". Более того, надо дать пользователю право на выполнение для всех каталогов, стоящих в дереве выше данного каталога. Поэтому в принципе для всех каталогов по умолчанию устанавливается право на выполнение как для владельца и группы, так и для всех остальных пользователей. И, уж если вы хотите закрыть доступ в каталог, то лишите всех пользователей (включая группу) права входить в этот каталог. Только не лишайте и себя такого права, а то придется обращаться к суперпользователю!

После прочтения предыдущего абзаца может показаться, что право на чтение каталога не дает ничего нового по сравнению с правом на выполнение. Однако разница в этих правах все же есть. Если задать только право на выполнение, вы сможете войти в каталог, но не увидите там ни одного файла (этот эффект особенно наглядно проявляется в том случае, если вы пользуетесь каким-то файловым менеджером, например, программой Midnight Commander). Если вы имеете право доступа в каком-то из подкаталогов этого каталога, то вы можете перейти в него (командой cd), но, как говорится "вслепую", по памяти, потому что списка файлов и подкаталогов текущего каталога вы не увидите.

Алгоритм проверки прав пользователя при обращении к файлу можно описать следующим образом. Система вначале проверяет, совпадает ли имя пользователя с именем владельца файла. Если эти имена совпадают (то есть владелец обращается к своему файлу), то проверяется, имеет ли владелец соответствующее право доступа: на чтение, на запись или на выполнение (не удивляйтесь, суперпользователь может лишить некоторых прав и владельца файла). Если право такое есть, то соответствующая операция разрешается. Если же нужного права владелец не имеет, то проверка прав, предоставляемых через группу или через группу атрибутов доступа для остальных пользователей, уже даже не осуществляется, а пользователю выдается сообщение о невозможности выполнения затребованного действия (обычно что-то вроде "Permission denied").

Если имя пользователя, обращающегося к файлу, не совпадает с именем владельца, то система проверяет, принадлежит ли владелец к группе, которая сопоставлена данному файлу (далее будем просто называть ее группой файла). Если принадлежит, то для определения возможности доступа к файлу используются атрибуты, относящиеся к группе, а на атрибуты для владельца и всех остальных пользователей внимания не обращается. Если же пользователь не является владельцем файла и не входит в группу файла, то его права определяются атрибутами для остальных пользователей. Таким образом, третья группа атрибутов, определяющих права доступа к файлу, относится ко всем пользователям, кроме владельца файла и пользователей, входящих в группу файла.

Для изменения прав доступа к файлу используется команда `chmod`. Ее можно использовать в двух вариантах. В первом варианте вы должны явно указать, кому какое право даете или кого этого права лишаете:

```
[user]$ chmod wXp имя-файла
```

где вместо символа w подставляется:

- либо символ u (то есть пользователь, который является владельцем);
- либо g (группа);
- либо o (все пользователи, не входящие в группу, которой принадлежит данный файл);
- либо a (все пользователи системы, т. е. и владелец, и группа, и все остальные).

Вместо x ставится:

- либо + (предоставляем право);
- либо - (лишаем соответствующего права);
- либо = (установить указанные права вместо имеющихся).

Вместо p — символ, обозначающий соответствующее право:

- `r` (чтение);
- w (запись);
- x (выполнение).

Вот несколько примеров использования команды chmod:

```
[user]$ chmod a+x file_name
```

предоставляет всем пользователям системы право на выполнение данного файла.

```
[user]$ chmod go-rw file_name
```

удаляет право на чтение и запись для всех, кроме владельца файла.

```
[user]$ chmod ugo+rwx file_name
```

дает всем права на чтение, запись и выполнение.

Если опустить указание на то, кому предоставляется данное право, то подразумевается, что речь идет вообще обо всех пользователях, т. е. вместо

```
[user]$ chmod a+x file_name
```

можно записать просто

```
[user]$ chmod +x file_name
```

Второй вариант задания команды chmod (он используется чаще) основан на цифровом представлении прав. Для этого мы кодируем символ `r` цифрой 4, символ w — цифрой 2, а символ x — цифрой 1. Для того чтобы предоставить

пользователям какой-то набор прав, надо сложить соответствующие цифры. Получив, таким образом, нужные цифровые значения для владельца файла, для группы файла и для всех остальных пользователей, задаем эти три цифры в качестве аргумента команды chmod (ставим эти цифры после имени команды перед вторым аргументом, который задает имя файла). Например, если надо дать все права владельцу (4+2+1=7), право на чтение и запись – группе (4+2=6), и не давать никаких прав остальным, то следует дать такую команду:

```
[user]$ chmod 760 file_name
```

Если вы знакомы с двоичным кодированием восьмеричных цифр, то вы поймете, что цифры после имени команды в этой форме ее представления есть не что иное, как восьмеричная запись тех самых 9 битов, которые задают права для владельца файла, группы файла и для всех пользователей.

Выполнять смену прав доступа к файлу с помощью команды chmod может только сам владелец файла или суперпользователь. Для того чтобы иметь возможность изменить права группы, владелец должен дополнительно быть членом той группы, которой он хочет дать права на данный файл.

Чтобы завершить рассказ о правах доступа к файлам, надо рассказать еще о трех возможных атрибутах файла, устанавливаемых с помощью той же команды chmod. Это те самые атрибуты для исполняемых файлов, которые в индексном дескрипторе файла в двухбайтовой структуре, определяющей права на файл, занимают позиции 5—7, сразу после кода типа файла.

Первый из этих атрибутов — так называемый бит смены идентификатора пользователя. Смысл этого бита состоит в следующем.

Обычно, когда пользователь запускает некоторую программу на выполнение, эта программа получает те же права доступа к файлам и каталогам, которые имеет пользователь, запустивший программу. Если же установлен бит смены идентификатора пользователя, то программа получит права доступа к файлам и каталогам, которые имеет владелец файла программы (таким образом, рассматриваемый атрибут лучше называть битом смены идентификатора владельца). Это позволяет решать некоторые задачи, которые иначе было бы трудно выполнить. Самый характерный пример — команда смены пароля passwd. Все пароли пользователей хранятся в файле /etc/passwd, владельцем которого является суперпользователь root. Поэтому программы, запущенные обычными пользователями, в том числе команда passwd, не могут производить запись в этот файл. А, значит, пользователь как бы не может менять свой собственный пароль. Но для файла /usr/bin/passwd установлен бит смены идентификатора владельца, каковым является пользователь root. Следовательно, программа смены пароля passwd запускается с правами root и получает право записи в файл /etc/passwd (уже средствами самой программы обеспечивается то, что пользователь может изменить только одну строку в этом файле).

Установить бит смены идентификатора владельца может суперпользователь с помощью команды

```
[root]# chmod +s file_name
```

Аналогичным образом работает бит смены идентификатора группы.

Еще один возможный атрибут исполняемого файла — это бит сохранения задачи, или sticky bit (дословно — "бит прилипчивости"). Этот бит указывает системе, что после завершения программы надо сохранить ее в оперативной памяти. Удобно включить этот бит для задач, которые часто вызываются на выполнение, т. к. в этом случае экономится время на загрузку программы при каждом новом запуске. Этот атрибут был необходим на старых моделях компьютеров. На современных быстродействующих системах он используется редко.

Если используется цифровой вариант задания атрибутов в команде chmod, то цифровое значение этих атрибутов должно предшествовать цифрам, задающим права пользователя:

```
[root]# chmod 4775 file_name
```

При этом веса этих битов для получения нужного суммарного результата задаются следующим образом:

- 4 — бит смены идентификатора пользователя;
- *2* — бит смены идентификатора группы;
- 1 — бит сохранения задачи (sticky bit).

Если какие-то из этих трех битов установлены в 1, то несколько изменяется вывод команды `ls -l` в части отображения установленных атрибутов прав доступа. Если установлен в 1 бит смены идентификатора пользователя, то символ x в группе, определяющей права владельца файла, заменяется символом s. Причем, если владелец имеет право на выполнение файла, то символ x заменяется на маленькое s, а если владелец не имеет права на выполнение файла (например, файл вообще не исполняемый), то вместо x ставится s. Аналогичные замены имеют место при задании бита смены идентификатора группы, но заменяется символ x в группе атрибутов, задающих права группы. Если равен 1 бит сохранения задачи (sticky bit), то заменяется символ x в группе атрибутов, определяющей права для всех остальных пользователей, причем x заменяется символом t, если все пользователи могут запускать файл на выполнение, и символом T, если они такого права не имеют.

Таким образом, хотя в выводе команды `ls -l` не предусмотрено отдельных позиций для отображения значений битов смены идентификаторов и бита сохранения задачи, соответствующая информация выводится. Вот небольшой пример того, как это будет выглядеть:

```
[root]# ls -l prim1
-rwSrwsrwT  1   kos    root     12 Dec 18 23:17    prim1
```

4.6. Команды для работы с файлами и каталогами

В предыдущих разделах мы уже упоминали некоторые команды для работы с файлами и каталогами: pwd, cd, ls, ln, chmod. В этом разделе рассмотрим (очень кратко) еще несколько часто используемых команд.

4.6.1. Команды *chown* и *chgrp*

Эти команды служат для смены владельца файла и группы файла. Выполнять смену владельца может только суперпользователь, смену группы может выполнить сам владелец файла или суперпользователь. Для того чтобы иметь право сменить группу, владелец должен дополнительно быть членом той группы, которой он хочет дать права на данный файл. Формат этих двух команд аналогичен:

```
[root]# chown vasja имя-файла
[root]# chgrp usersgrp имя-файла
```

4.6.2. Команда *mkdir*

Команда mkdir позволяет создать подкаталог в текущем каталоге. В качестве аргумента этой команде надо дать имя создаваемого каталога. Во вновь созданном каталоге автоматически создаются две записи: . (ссылка на этот самый каталог) и .. (ссылка на родительский каталог). Чтобы создать подкаталог, вы должны иметь в текущем каталоге право записи. Можно создать подкаталог не в текущем, а в каком-то другом каталоге, но тогда необходимо указать путь к создаваемому каталогу:

```
[user]$ mkdir /home/kos/book/glava5/partl
```

Команда mkdir может использоваться со следующими опциями:

- -m mode — задает режим доступа для нового каталога (например, -т 755);
- -p — создавать указанные промежуточные каталоги (если они не существуют).

4.6.3. Команда *cat*

Команда cat часто используется для создания файлов (хотя можно воспользоваться и командой touch). По команде cat на стандартный вывод (то есть на экран) выводится содержимое указанного файла (или нескольких файлов, если их имена последовательно задать в качестве аргументов команды).

Если вывод команды cat перенаправить в файл, то можно получить копию какого-то файла:

```
[user]$ cat file1 > file2
```

Собственно, первоначальное предназначение команды cat как раз и предполагало перенаправление вывода, т. к. эта команда создана для конкатенации, т. е. объединения нескольких файлов в один:

```
[user]$ cat file1 file2 ... fileN > new-file
```

Именно возможности перенаправления ввода и вывода этой команды и используются для создания новых файлов. Для этого на вход команды cat направляют данные со стандартного ввода (то есть с клавиатуры), а вывод команды — в новый файл:

```
[user]$ cat > newfile
```

После того как вы напечатаете все, что хотите, нажмите комбинацию клавиш <Ctrl>+<D> или <Ctrl>+<C>, и все, что вы ввели, будет записано в `newfile`. Конечно, таким образом создаются, в основном, короткие текстовые файлы.

4.6.4. Команда *cp*

Хотя для копирования файлов иногда пользуются командой cat, но в Linux существует для этого специальная команда cp. Ее можно применять в одной из двух форм:

```
[user]$ cp [options] source destination
[user]$ cp [options] source_directory new_directory
```

В первом случае файл или каталог source копируется, соответственно, в файл или каталог destination, а во втором случае файлы, содержащиеся в каталоге source_directory, копируются в каталог new_directory. Для копирования надо иметь права на чтение файлов, которые копируются, и права на запись в каталог, в который производится копирование.

Если в качестве целевого указывается существующий файл, то его содержимое будет затерто, поэтому при копировании надо соблюдать осторожность. Впрочем, можно использовать команду cp с опцией -i, тогда перед перезаписью существующего файла будет запрашиваться подтверждение (очень рекомендую вам всегда использовать эту опцию!).

У команды cp имеется еще несколько полезных опций (табл. 4.3).

Таблица 4.3. *Основные опции команды cd*

Опция	Значение
-p	Сохраняет время модификации файла и максимально возможные полномочия. Без этой опции для нового файла задаются полномочия, соответствующие полномочиям запустившего команду пользователя
-R или -r	Если source — каталог, то копируется как он, так и все входящие в него подкаталоги, т. е. сохраняется исходная форма дерева каталогов
-d	Если задать эту опцию, то символические ссылки будут оставаться ссылками (а иначе вместо ссылки копируется файл, на который дается ссылка)
-f	Перезаписывать файлы при копировании (если такие уже есть) без дополнительных предупреждений

4.6.5. Команда *mv*

Если вам необходимо не скопировать, а переместить файл из одного каталога в другой, вы можете воспользоваться командой mv. Синтаксис этой команды аналогичен синтаксису команды cp. Более того, она сначала копирует файл (или каталог), а только потом удаляет исходный файл (каталог). И опции у нее такие же, как у cp.

Команда mv может использоваться не только для перемещения, но и для переименования файлов и каталогов (то есть перемещения их внутри одного каталога). Для этого надо просто задать в качестве аргументов старое и новое имя файла:

```
[user]$ mv oldname newname
```

Но учтите, что команда mv не позволяет переименовать сразу несколько файлов (используя шаблон имени), так что команда mv *.xxx *.yyy не будет работать.

При использовании команды mv, также как и при использовании cp, не забывайте применять опцию -i для того, чтобы получить предупреждение, когда файл будет перезаписываться.

4.6.6. Команды *rm* и *rmdir*

Для удаления ненужных файлов и каталогов в Linux служат команды rm (удаляет файлы) и rmdir (удаляет пустой каталог). Для того чтобы воспользовался этими командами, вы должны иметь право записи в каталоге, в котором расположены удаляемые файлы или каталоги. При этом полномочия на изменение самих файлов необязательны. Если хотите перед удалением

файла получить дополнительный запрос на подтверждение операции, используйте опцию -i.

Если вы попытаетесь использовать команду rm (без всяких опций) для удаления каталога, то будет выдано сообщение, что это каталог, и удаления не произойдет. Для удаления каталога надо удалить в нем все файлы, после чего удалить сам каталог с помощью команды `rmdir`. Однако можно удалить и непустой каталог со всеми входящими в него подкаталогами и файлами, если использовать команду rm с опцией `-r`.

Если вы дадите команду rm *, то удалите все файлы в текущем каталоге. Подкаталоги при этом не удалятся. Для удаления как файлов, так и подкаталогов текущего каталога надо тоже воспользоваться опцией `-r`. Однако всегда помните, что в Linux нет команды восстановления файлов после их удаления (даже если вы спохватились сразу же после ошибочного удаления файла или каталога)!

Так что дважды подумайте до удаления чего-либо и не пренебрегайте опцией -i.

4.6.7. Команды *more* и *less*

Команда cat позволяет выдать на стандартный вывод (на экран) содержимое любого файла, однако она используется для этих целей очень редко, разве что для вывода очень небольших по объему файлов. Дело в том, что содержимое большого файла мгновенно проскакивает на экране, и пользователь видит только последние строки файла. Поэтому cat используется в основном по ее прямому назначению — для конкатенации файлов, а для просмотра содержимого файлов (конечно, текстовых) используются команды more и `less` (или текстовые редакторы).

Команда-фильтр more выводит содержимое файла на экран отдельными страницами размером как раз в целый экран. Для того чтобы увидеть следующую страницу, надо нажать на клавишу пробела. Нажатие на клавишу <Enter> приводит к смещению на одну строку. Кроме клавиш пробела и <Enter> в режиме паузы еще некоторые клавиши действуют как управляющие (например, клавиша <B> возвращает вас на один экран назад), но мы здесь не будем приводить полного их перечня, как и перечня опций команды. Вам для начала надо еще только запомнить, что выйти из режима просмотра можно с помощью клавиши <Q>, т. к. если вы этого не знаете, то вам придется долго и нудно нажимать пробел, пока вы не доберетесь до конца длинного файла. Обо всех опциях команды more вы можете прочитать в интерактивных руководствах man или info.

Утилита less, разработанная в рамках проекта GNU, содержит все функции и команды управления выводом, имеющиеся в программе more, и некоторые дополнительные, например, позволяет использовать клавиши управления

курсором (<↓>, <t>, <PgUp>, <PgDown>) для перемещения по тексту. Вспомните, мы уже говорили об этом, когда рассматривали интерактивную подсказку man.

Команды more и less позволяют производить поиск подстроки в просматриваемом файле, причем команда less позволяет производить поиск как в прямом, так и в обратном направлении. Для организации поиска строки символов string надо набрать в командной строке программы в нижней части экрана (там, где двоеточие) /string. Если искомая строка будет найдена, будет отображен соответствующий кусок текста, причем найденная строка будет находиться в самом верху экрана.

4.6.8. Команда *find* и символы шаблонов для имен файлов

Еще одной часто используемой командой для работы с файлами в Linux является команда поиска нужного файла find. Команда find может искать файлы по имени, размеру, дате создания или модификации и некоторым другим критериям.

Общий синтаксис команды `find` имеет следующий вид:

find `[список_каталогов]` критерий_поиска

Параметр `список_каталогов` определяет, где искать нужный файл. Проще всего задать в качестве начального каталога поиска корневой каталог /, однако, в таком случае поиск может затянуться очень надолго, т. к. будет просматриваться вся структура каталогов, включая смонтированные файловые системы (в том числе сетевые, если таковые есть). Можно сократить объем поиска, если задать вместо одного корневого каталога список из нескольких каталогов (естественно, тех, в которых может находиться искомый файл):

```
[user]$ find /usr/share/doc /usr/doc /usr/locale/doc -name instr.txt
```

Началом "критерия_поиска", определяющего, что именно должна искать программа find, считается первый аргумент, начинающийся на один из символов: -`(),!`. Все аргументы, предшествующие "критерию_поиска", трактуются как имена каталогов, в которых надо производить поиск. Если не указано ни одного пути, поиск производится только в текущем каталоге и его подкаталогах.

Чаще всего поиск проводится по именам файлов, как это показано в предыдущем примере, т. е. "критерий_поиска" задается как -name `имя_файла`. Вместо опции -name можно использовать опцию -path, тогда команда будет искать совпадения в полном имени файла, с указанием пути. Например, команда

```
[user]$ find . -path './sr*sc'
```

найдет в текущем каталоге подкаталог ./src/misc. Вместо полного имени файла или каталога в этом примере использован так называемый "шаблон имени". Поскольку шаблоны имен файлов могут использоваться не только с командой find, но и со многими другими командами (включая уже рассмотренные Команды chmod, chown, chgrp, cp, rm, cat, mv), ТО Правилам СОставления шаблонов стоит уделить некоторое внимание.

Чаще всего шаблоны имен файлов строятся с помощью специальных символов * и ?. Символ * используется для замены произвольной строки символов. В Linux:

- * — соответствует всем файлам, за исключением скрытых;
- .* — соответствует всем скрытым файлам (но также текущему каталогу . и каталогу уровнем выше ..: не забывайте об этом!);
- *.* — соответствует только тем файлам и каталогам, которые имеют . в середине имени, или оканчиваются на точку;
- p*r — соответствует и peter, и piper;
- *c* — **соответствует И** picked, И peck.

Значок ? заменяет один произвольный символ, поэтому index?.htm будет **соответствовать именам** index0.htm, index5.htm **И** indexa.htm.

Кроме * и ? в Linux при задании шаблонов имен можно использовать квадратные скобки [], в которых дается либо список возможных символов, либо интервал, в который должны попадать возможные символы. Например, [abc]* соответствует всем именам файлов, начинающимся с символов а, ь, с; *[I-N1-3] соответствует файлам, имена которых оканчиваются на I, J, к, L, M, N, 1, 2, 3.

А теперь вернемся к команде find и расскажем подробнее о том, какие критерии поиска возможны. Несколько примеров простых критериев поиска перечислены в табл. 4.4.

Таблица 4.4. *Критерии поиска для команды find*

Опция	Значение
-name шаблон	Ищет файлы, имена которых соответствуют шаблону
-group имя	Ищет файлы, принадлежащие указанной группе
-size число [c]	Ищет файлы, размером в число 512-байтных блоков. Если после числа стоит символ c, значит размер указан в байтах (символах)
-mtime число	Ищет файлы, которые в последний раз изменялись указанное число дней назад
-newer образец	Ищет файлы, которые изменялись после изменения файла, указанного в образце

Таблица 4.4 *(окончание)*

Опция	Значение
-type тип_файла	Ищет файлы указанного типа. Тип задается одним из символов b (блок-ориентированные устройства), c (байт-ориентированные устройства), d (файл каталога), f (обычный файл), p (именованный канал) либо l (символическая ссылка)

Другие простые критерии вы можете узнать, если просмотрите man-страницу о команде `find`. Здесь же надо только сказать, что из простых критериев можно строить более сложные с помощью логических операций and, or или операции отрицания, знаком которой служит восклицательный знак. Например, если вы хотите найти все файлы, имена которых оканчиваются на .txt и .doc, то критерий можно записать как `(-name *.txt -or -name *.doc)`. Можно комбинировать таким образом любое число критериев (и не только простых!). Если операция не указана явно, то подразумевается -and, т. е. вместо `(-name *.txt -and -name *.doc)` можно записать просто `(-name *.txt -name *.doc)`. Если применяется только одна операция -and или `!`, то скобки обычно можно опустить, а с операцией `-or` и в сложных выражениях скобки необходимы. Перед скобкой нужно поставить обратную косую черту, а после скобки — пробел. Например, если вы хотите найти каталог по его имени, то можно сделать это командой

```
[user]$ find /usr -name doc -type d
```

или (с соблюдением правил построения сложных критериев)

```
[user]$ find /usr \( -name doc -and -type d \)
```

В следующем примере мы ищем файлы по такому критерию: либо имя файла оканчивается на `*.tmp`, либо размер файла больше 100 Кбайт.

```
[user]$ find /home/kos \( \( -name *.tmp \) -or \( -size +200 \) \)
```

В последнем примере стоит обратить внимание на то, что перед значением размера стоит знак "+". Такой знак можно использовать с любым числовым параметром в критериях поиска команды `find`. Он означает, что нужно искать файлы, у которых значение параметра больше заданного. Соответственно, знак "-" означает, что надо искать файлы, у которых значение параметра меньше заданного. Если знаки "+" или "-" отсутствует, ищутся файлы, у которых значение параметра равно заданному.

Чтобы закончить рассмотрение команды `find`, надо сказать еще о том, что после критерия поиска в этой команде можно сразу же задать операцию, которая будет применяться ко всем файлам, найденным по указанному критерию. Простейшим примером использования такой возможности является указание команды -print.

```
[user]$ find /home/kos -name *.tmp -print
```

по которой выдается на экран список имен всех найденных файлов с указанием полного пути к файлу. Эта операция применяется по умолчанию, т. е. когда никаких операций вообще не указано (как это было во всех приведенных выше примерах). Другим примером операции, применяемой ко всем найденным файлам, может служить операция

```
-exec cmd {} \;
```

где `cmd` — произвольная команда оболочки shell. То есть в этом случае ко всем найденным файлам (их именами заменяются поочередно фигурные скобки) применяется команда cmd. За cmd `{}` в этом случае должна следовать точка с запятой, экранированная обратной косой чертой.

Например, если вы хотите удалить в текущем каталоге все файлы, к которым пользователи не обращались в течение 30 дней, дайте команду:

```
[root]# find . -type f -atime +30 -exec rm {} \;
```

Вместо -exec можно поставить -ok, тогда перед выполнением указанной команды cmd применительно к каждому файлу будет запрашиваться подтверждение.

В общем, команда find является очень мощным, полезным и чрезвычайно адаптируемым инструментом поиска в файловой системе. Все ее возможности здесь не перечислены, изучайте соответствующую man-страницу. И будьте очень осторожны с применением таких возможностей команды, как вызов других команд, применяемых ко всем найденным файлам. Помните, что изменения часто необратимы!

4.6.9. Команда *split*— разбиваем файл на несколько частей

Иногда бывает необходимо разбить один большой файл на несколько файлов меньшего объема. Для примера рассмотрим ситуацию, когда вы хотите перенести на свой домашний компьютер файл song.mp3 формата MP3, размером в 4 894 425 байтов. Возможности перекачать этот файл по сети у вас нет, и единственно возможный способ переноса — воспользоваться дискетами. Но, поскольку на одну дискету файл не помещается, требуется разбить его на несколько маленьких файлов, а потом "собрать" снова. Для решения этой задачи можно воспользоваться командой split.

Команда split копирует файл, разбивая его на отдельные файлы заданной длины. В качестве аргументов ей надо указать имя исходного файла и префикс имен выходных файлов. Имена выходных файлов будут составляться из этого префикса и двух дополнительных букв aa, ab, ac и т. д. (без пробелов и точек между префиксом и буквами). Если префикс имен файлов не задан, то по умолчанию используется x, так что выходные файлы будут называться xaa, xab и т. д.

Кроме аргументов можно задать опцию -ь, определяющую размер выходных файлов в байтах. Вслед за -b должно стоять число, а за ним — буква k (показывающая, что размер выходного файла указан в килобайтах) или m (значит размер задан в мегабайтах). Если опция не задана, то по умолчанию размер выходных файлов принимается равным 1 Мбайт. Таким образом, чтобы перенести на дискетах файл song.mp3, надо вначале дать команду

```
[user]$ split -b1400k song.mp3 song
```

скопировать полученные файлы song.aa, song.ab, song.ac, song.ad, song.ae на отдельные дискеты, перенести их на домашний компьютер, скопировать в какой-нибудь каталог и восстановить исходный файл с помощью команды

```
[user]$ cat song.* > song.mp3
```

после чего можно удалить временные файлы song.xx.

4.6.10. Сравнение файлов и команда *patch*

Вы не замечали, что задача сравнения содержимого двух различных файлов возникает при работе с компьютером удивительно часто? Конечно, ведь так легко скопировать файл, а потом забываешь, какая же из версий новее или качественнее (по одному автору ведомым критериям). Так что инструменты для сравнения файлов просто необходимы, и Linux такие инструменты предоставляет.

Простейший из них — команда стр. Эта команда просто сравнивает содержимое двух файлов побайтно:

```
[user]$ cmp file1 file2
```

Если файлы полностью совпадают, она молча заканчивает свою работу (происходит возврат к командной строке без каких-либо дополнительных сообщений), а если файлы различаются, выдаются номер строки и номер байта в строке, где имеет место первое различие.

Конечно, информации, выдаваемой командой cmp, маловато для того, чтобы принять, например, решение о том, какой из двух файлов нам более ценен. Поэтому стоит воспользоваться командой diff для получения полного отчета о том, каковы же различия в интересующих нас файлах. Для получения отчета достаточно указать команде, какие именно файлы сравнивать:

```
[user]$ diff paper.old paper.new
```

Отчет о выявленных различиях будет выдан на стандартный выход. Естественно, его лучше перенаправить в файл:

```
[user]$ diff paper.old paper.new >paper.diff
```

Для оценки версий одного и того же файла более удобной может оказаться команда sdiff, которая выдает результат сравнения в виде двух столбцов,

разделенных пробелами. Если строки с одинаковыми номерами в файлах различаются, то в выводе команды sdiff они разделены вертикальной чертой |. Если строка имеется только в первом файле, она отмечена знаком <. Соответственно, строка, встречающаяся только во втором файле, помечена знаком >.

Существует также команда diff3, которая позволяет сравнить сразу 3 файла.

Но все же наиболее часто используется традиционная для UNIX-систем команда diff. Эта ее востребованность объясняется тем, что создаваемый ею отчет о различиях двух файлов может быть использован командой patch. Чаще всего эти возможности используются при распространении обновлений программного обеспечения. Предположим, что некоторое программное приложение было разослано пользователям в виде файла program.c, содержащего исходный код программы на языке C. После этого разработчик внес в программу некоторые исправления и сохранил текст в виде файла program.c.new. Требуется довести исправленный текст программы до пользователей. Очевидно, что пользователям достаточно сообщить только исправления, т. е. отчет об изменениях, создаваемый по команде

```
[user]$ diff program.c program.c.new > program.c.diff
```

Естественно, объем файла program.c.diff существенно меньше объема файла program.c.new, так что можно было бы получить существенную экономию на передаче файлов, если отправлять пользователям только файл program.c.diff (ведь объемы современных программных приложений составляют десятки мегабайтов). Однако пользователи должны иметь возможность внести эти исправления в имеющуюся у них версию программы. Эту задачу позволяет решить команда patch. Имея файлы program.c и program.c.diff, пользователь может дать команду

```
[user]$ patch program.c program.c.diff > program.c.new
```

в результате выполнения которой он получит файл program.c.new.

4.7. Команды архивирования файлов

При работе с Linux вы, может быть, еще не скоро встретитесь с необходимостью работать с большинством консольных команд, поскольку имеются такие оболочки, как Midnight Commander или графические оболочки типа KDE. Но с командами архивирования (точнее, разархивирования) вам работать придется обязательно, хотя бы потому, что вы будете часто встречать архивированные файлы в Интернете.

Основным средством архивирования в UNIX (а, следовательно, и в Linux) является комплекс из двух программ — tar и gzip. Хотя никто не запрещает пользоваться arj, pkzip, lha, rar и т. д. — версии этих программ для Linux

общедоступны. Просто уж исторически сложилось, что пользователи UNIX чаще применяют именно tar и gzip, и именно в таком формате распространяется большая часть программного обеспечения для UNIX. Поэтому овладеть работой с tar и gzip — дело чести любого пользователя Linux.

4.7.1. Программа tar

У читателя, привыкшего к архиваторам типа arj, которые собирают файлы в единый архив и сразу "сжимают" их, может возникнуть вопрос "А зачем использовать две программы?" Все дело в том, что tar расшифровывается как Tape ARchiver, он не сжимает данные, а лишь объединяет их в единый файл с последовательным доступом для последующей записи на ленту. По умолчанию этот архивный файл создается на ленточном накопителе, точнее на устройстве /dev/rmt0. Если вы хотите создать архивный файл на диске, то необходимо использовать команду tar с опцией -f, после которой указывается имя архивного файла.

У программы tar имеется 8 опций, отличающихся от остальных тем, что при вызове программы должна обязательно задаваться одна из этих опций. Эти опции определяют основные функции программы. Перечень их приведен в табл. 4.5.

Таблица 4.5. *Основные опции программы tar*

Опция	Значение
-A, --catenate, --concatenate	Добавляет файлы в существующий архив
-c, --create	Создает новый архив
-d, --diff, --compare	Находит различия между архивом и файловой системой
--delete	Удаляет из архива (не может использоваться с магнитной лентой!)
-r, --append	Дописывает файлы в конец архива
-t, --list	Выводит список файлов архива
-u, --update	Добавляет только файлы, которые новее, чем имеющаяся в архиве копия
-x, --extract, --get	Извлекает файлы из архива

Если вы работаете с файлами архивов на дисках, а не с ленточным устройством, то, очевидно, обязательной будет и опция -f. Другие опции не являются обязательными, они служат только для конкретизации задания про-

грамме. Например, опция -v заставляет программу выводить список обрабатываемых файлов.

Однобуквенные опции программы tar могут перечисляться друг за другом (вы увидите это в приводимых ниже примерах).

Я не буду давать здесь описание всех опций команды tar, просто приведу несколько командных строк для выполнения самых необходимых действий с архивами.

Чтобы создать один tar-архив из нескольких файлов, используется команда:

```
[user]$ tar -cf имя_архива файл1 файл2 ...
```

где опция -c сообщает программе, что необходимо создать (create) архив, а опция -f говорит о том, что архив должен создаваться в виде файла (имя которого должно следовать сразу за этой опцией).

В именах файлов, которые сохраняются в архиве, можно использовать шаблоны имен файлов, в том числе просто символы-заместители * и ?. Благодаря этому можно очень короткой командой отправить в архив сразу много файлов. Например, для того, чтобы создать архив, содержащий все файлы одного из подкаталогов (пусть это будет sub_dir) текущего каталога, достаточно дать команду

```
[user]$ tar -cvf имя_архива ./sub_dir/*
```

или даже просто

```
[user]$ tar -cvf имя_архива sub_dir
```

По этой команде в архиве будут сохранены не только файлы, расположенные непосредственно в подкаталоге sub_dir, но и рекурсивно все файлы из подкаталогов каталога sub_dir. При этом в архиве сохраняется вся структура подкаталогов каталога sub_dir.

Заметим, что если в только что приведенном примере вместо * поставить *.*, то будут сохранены только те файлы, которые расположены непосредственно в подкаталоге sub_dir, а подкаталоги каталога sub_dir архивированы не будут. Если в том же примере не указать имя подкаталога, то будут архивироваться все файлы (и подкаталоги) текущего каталога. Но если вы дадите команду следующего вида

```
[user]$ tar -cvf имя_архива ./.*
```

то в архиве будут сохранены не только все файлы (и подкаталоги) текущего каталога, но и файлы из родительского каталога, а хотели ли вы этого?

Теперь вы знаете, как создать архив, а для того, чтобы распаковать (извлечь) файлы из архива, нужно дать команду:

```
[user]$ tar -xvf имя_архива файлы
```

Получить список файлов архива можно командой

```
[user]$ tar -tf имя_архива | less
```

Программа tar является удобным средством для создания резервных копий файлов. Конечно, существуют специальные утилиты резервного архивирования, но даже если вы о них еще не знаете, то, по меньшей мере, вы можете сделать следующее:

```
[user]$ tar -Mcvf /dev/fdOH1440 /каталог
```

Такая команда создаст на дискетах архив с содержимым каталога, разбивая его на тома. Монтировать дискеты перед запуском команды не нужно, программа просто пишет на устройство потоком (в данном случае на дискету по секторам). При этом никакой файловой системы на дискете не создается. После заполнения дискеты вам будет выдан запрос на смену дискеты. Только, прежде чем запускать такую команду на выполнение, приготовьте достаточное число свободных дискет (помните, что tar не сжимает файлы), которые лучше всего соответствующим образом пометить и обязательно пронумеровать. Кроме того, имейте в виду, что вся информация на дискетах будет молча уничтожена.

Чтобы восстановить сохраненные данные, воспользуйтесь командой:

```
[user]$ tar -Mxpvf /dev/fd0H1440
```

Если вы ошибетесь в порядке вставляемых дискет, программы сообщит вам об этом и попросит заменить том.

В заключение раздела заметим, что всегда можно получить подсказку по использованию программы tar, дав команду

```
[user]$ tar -help
```

При этом, если вы используете русифицированный дистрибутив Linux, например, Black Cat 6.02, то подсказка будет выдаваться по-русски.

4.7.2. Программа gzip

Хотя программа tar создает архивы, она, как было сказано, не сжимает архивы, а просто соединяет отдельные файлы в единый архивный файл. Для сжатия этого файла часто применяют команду gzip. В простейшем случае она вызывается в следующем формате:

```
[user]$ gzip файл
```

В командной строке можно указать сразу несколько имен файлов или шаблон имени файла. Но в этом случае каждый из указанных файлов будет заархивирован отдельно (общий архив не создается).

Для того чтобы распаковать архив, используйте команду

```
[user]$ gzip -d файл_архива
```

или

```
[user]$ gunzip файл_архива
```

Исходные файлы после сжатия удаляются, остается только архивный файл (файлы перемещаются в архив), а при разархивации удаляется архив.

В табл. 4.6 перечислены другие полезные опции программы gzip.

Таблица 4.6. *Основные опции программы gzip*

Опция	Значение
-h, --help	Вызов краткой помощи по использованию программы
-l, --list	Выдает имя файла, содержащегося в архиве, его объем и степень сжатия
-L, --license	Отображает номер версии и лицензию на программу
-N, --name	Сохраняет (или восстанавливает) исходное имя и время создания файла
-n, --no-name	Не сохраняет (не восстанавливает) исходное имя и время создания файла
-q, --quiet	Подавляет выдачу на экран предупреждающих сообщений
-r, --recursive	Рекурсивно обрабатывает подкаталоги (используется в случае, когда задан шаблон имен обрабатываемых файлов)
-S.suf, --suffix .suf	Добавить суффикс .suf к имени сжатого файла (вместо добавляемого по умолчанию суффикса .gz; но учтите, что при разархивации файлов с суффиксами, отличными от .gz, программа вас не поймет)
-t, --test	Тестирует архивный файл
-v, --verbose	Выдает дополнительные сообщения в процессе работы программы
-V, --version	Отображает версию программы
-1, --fast	Быстрое сжатие
-9, --best	Более высокая степень сжатия

Поскольку программа gzip не умеет сохранять в одном архиве несколько файлов, то обычно ее применяют для сжатия архивов, созданных программой tar. Более того, среди опций программы tar имеется специальная опция -z, позволяющая сразу после создания сжать его с помощью программы

gzip. Для выполнения такого сжатия надо использовать команду tar примерно следующим образом:

```
[user]$ tar -czf имя_архива шаблон_имен_файлов (или имя_каталога)
```

Только имейте в виду, что в этом случае суффикс .gz не добавляется автоматически к имени создаваемого архива, поэтому лучше сразу задать имя архива с указанием обеих суффиксов: имя.tar.gz.

4.7.3. Программа bzip2

В последнее время все чаще вместо программы gzip используется архиватор bzip2, который обеспечивает более высокую степень сжатия и работает несколько быстрее. Программа bzip2 обычно не устанавливается автоматически при инсталляции Linux. Но она имеется на дистрибутивном диске в виде rpm-пакета и ее легко установить. (Как это сделать, *см.* в *гл. 10.)*

Работает bzip2 примерно так же, как команда gzip, т. е. замещает каждый файл, имя которого задано в командной строке, сжатой версией, добавляя к имени файла суффикс .bz2.

Сжатый файл имеет то же самое время модификации, права доступа и, по возможности, того же владельца, что и исходный файл, что дает возможность восстановить эти атрибуты при извлечении файлов из архива.

В некоторых случаях сжатый файл может оказаться даже больше по размеру, чем исходный. Это происходит, например, для файлов длиной менее 100 байтов, потому что механизм сжатия использует заголовок длиной около 50 байтов. Для файлов, представляющих собой случайную последовательность символов (в том числе для выходных файлов большинства файловых архиваторов) длина файла увеличивается примерно на 0,5%.

Команда bunzip2 (или bzip2 -d) разархивирует указанные в командной строке файлы. Если эти файлы не были созданы программой bzip2, они не будут разархивироваться, будет выдано соответствующее предупреждение. При разархивации bzip2 пытается угадать имя разархивируемого файла по следующим правилам:

- filename.bz2 заменяется на filename;
- filename.bz заменяется на filename;
- filename.tbz2 заменяется на filename.tar;
- filename.tbz заменяется на filename.tar;
- любое другое "имя" заменяется на "имя.out".

Опции командной строки для bzip2 очень похожи на опции команды gzip, но все же они не идентичны. В табл. 4.7 приведена краткая сводка наиболее необходимых в работе опций.

Таблица 4.7. *Основные опции программы bzip2*

Опция	Значение
-d, --decompress	Принудительная разархивация. Эта опция необходима в силу того, что bzip2, bunzip2 и bzcat — это на самом деле одна и та же программа, которая сама по расширению имени файла принимает решение о том, какое действие надо выполнить над указанным файлом. Опция -d отключает этот механизм и заставляет программу разархивировать указанные файлы
-z, --compress	Принудительная архивация
-t, --test	Проверка целостности указанного файла(ов) без разархивации
-f, --force	Перезапись существующего файла. По умолчанию bzip2 не перезаписывает существующие файлы. Если вы хотите перезаписать существующий файл, надо задать опцию -f
-k, --keep	Сохранение (не удалять) исходные файлы при архивации или разархивации
-s, --small	Снижение требования к объему используемой оперативной памяти за счет снижения скорости архивации. Эту опцию рекомендуется применять на компьютерах с малым объемом ОЗУ (8 Мбайт и меньше)
-q, --quiet	Не выводить малосущественные сообщения
-v, --verbose	Выводить дополнительную информацию в процессе работы (представляет интерес в диагностических целях)
-L, --license, -V, --version	Отобразить версию программы и лицензионное соглашение

Аргументы командной строки, которым предшествует двойное тире и пробел, трактуются как имена файлов, даже если они начинаются с тире. Например,

```
[user]$ bzip2 -  -myfilename
```

Я думаю, что приведенных данных достаточно для квалифицированного применения архиваторов tar, gzip и bzip2. За дополнительными сведениями, как всегда, обращайтесь к интерактивной подсказке man. По утилите bzip2 имеется пособие Bzip2-HOWTO, которое даже переведено на русский язык *(см. [П6.4] приложения).*

4.8. Создание и монтирование файловых систем

В предыдущих разделах мы кратко рассмотрели основные команды для работы с уже сформированной файловой системой. Теперь надо остановиться на вопросе о том, как создать файловую систему и модифицировать ее.

Общее дерево файлов и каталогов системы Linux формируется из отдельных "ветвей", соответствующих различным физическим носителям. Часто говорят, что оно формируется из отдельных файловых систем. Говорить так позволяет тот факт, что в UNIX нет понятия "форматирования диска" (и команды форматирования), а используется понятие "создание файловой системы". Когда мы получаем новый носитель, например, жесткий диск, мы должны создать на нем файловую систему. То есть каждому носителю ставится в соответствие отдгльная файловая система. Чтобы эту файловую систему использовать для записи в нее файлов, надо ее вначале подключить в общее дерево каталогов ("смонтировать"). Вот и получается, что можно говорить о монтировании файловых систем или о монтировании носителей (с созданными на них файловыми системами).

Можно еще в виде предисловия отметить, что обычно жесткий диск предварительно разбивается на разделы (особенно современные диски, имеющие емкость, исчисляемую десятками гигабайтов). Создание разделов облегчает выполнение резервного копирования, решение задач разграничения полномочий, повышает производительность и ограничивает потенциальный ущерб, наносимый вышедшими из-под контроля программами. Поэтому в дальнейшем будем говорить о создании файловой структуры в одном разделе (диск, не имеющий разделов, можно тоже рассматривать как один раздел).

Еще один момент, существенный в контексте этого раздела, связан с тем, что Linux может работать с разными типами файловых систем. "Родной" файловой системой для него в настоящее время является "вторая расширенная файловая система" (second extended fflesystem) ext2fs. Но в Linux можно работать и с 16-разрядной файловой системой FAT, создаваемой в MS-DOS, и с 32-разрядной FAT32, разработанной для MS Windows 95, и с файловой системой ISO9660, используемой для записи информации на CD-ROM, и с другими типами файловых систем (в число которых пока, правда, не входит NFTS). То есть при рассмотрении вопросов создания и монтирования файловых систем надо постоянно помнить о том, что типы файловых систем на разных носителях могут различаться.

Итак, вначале рассмотрим случай, когда требуется создать в каком-то разделе диска файловую систему. Будем предполагать, что создается файловая система типа ext2fs (создание файловых систем других типов — тема для книг, посвященных другим операционным системам). Создание файловой системы типа ext2fs подразумевает создание в данном разделе на диске су-

перблока, таблицы индексных дескрипторов и совокупности блоков данных. Делается все это все с помощью команды mkfs. В простейшем случае достаточно дать эту команду в следующем формате:

```
[root]# mkfs -t ext2 /dev/hda5
```

где /dev/hda5 надо, естественно, заменить указанием на соответствующее устройство или раздел. Например, если вы хотите создать файловую систему на дискете, то команда примет вид

```
[root]# mkfs -t ext2 /dev/fd0
```

(Можно сказать, что мы "отформатировали дискету", но учтите, что в DOS или Windows такие дискеты не читаются. Для создания под Linux дискет, которые бы читались в DOS или Windows, служат специальные утилиты.)

После выполнения команды mkfs в указанном разделе будет создана файловая система ext2fs (еще раз повторю, что подробнее об этом вы узнаете в *гл. 16).* В новой файловой системе автоматически создается один каталог с именем lost+found. Он используется в экстренных случаях программой fsck, поэтому не удаляйте его. Для того чтобы начать работать с новой файловой системой (например, переписать какие-то файлы на новый носитель), необходимо подключить ее в общее дерево каталогов, что делается с помощью команды mount.

В качестве параметров команде mount надо, как минимум, указать устройство и "точку монтирования". Точкой монтирования называется тот каталог в уже существующем и известном системе дереве каталогов, который будет теперь служить корневым каталогом для подключаемой файловой системы.

Пример:

команда `[root]# mount /dev/hdb1 /mnt/disk2`

подсоединяет файловую систему первого раздела на втором жестком диске к каталогу /mnt/disk2 (этот каталог должен существовать!).

Отметим, что после монтирования файловой системы в каталог /mnt/disk2 прежнее содержимое этого каталога станет для вас недоступно (так же, как информация о прежнем владельце и правах доступа к этому каталогу) до тех пор, пока вы не размонтируете вновь подключенную файловую систему. Прежнее содержимое не уничтожается, а просто становится временно недоступным. Поэтому в качестве точек монтирования лучше использовать пустые каталоги (заранее заготовленные).

В той простейшей форме, которую мы использовали в приведенном выше примере, команда mount будет работать только при условии, что все недостающие ей для выполнения параметры она найдет в файле /etc/fstab. Если же такого файла не существует, или он не содержит необходимых данных, надо применять полную форму команды mount:

```
[root]$ mount -t type device path
```

где type задает тип файловой системы, device указывает, на каком устройстве (в каком разделе) она находится, а path задает точку монтирования.

Конфигурационный файл /etc/fstab используется в основном для того, чтобы обеспечить автоматическое монтирование файловых систем в процессе загрузки. Каждая строка этого файла содержит описание одной файловой системы и состоит из 6 полей, разделяемых пробелами (для удобочитаемости поля обычно выравнивают, но делать это необязательно):

- имя устройства. В качестве имени может использоваться как имя локального устройства, например, /dev/hda5, так и путевое имя сетевой файловой системы NFS, например, pc21:/home/jim, что указывает на каталог /home/jim на машине с именем pc21;
- точка монтирования (полное имя каталога, в который будет монтироваться файловая система);
- тип файловой системы;
- опции монтирования (по умолчанию подразумевается rw — чтение, запись);
- уровень дампа. Это поле используется программой dump, предназначенной для создания резервных копий. Если файловая система должна участвовать в процессе резервного копирования, то здесь должно стоять число 1, если нет — 0. Возможны и другие значения (см. руководство к программе dump);
- порядок (приоритет) проверки файловых систем программой fsck. Системы с меньшими значениями этого поля проверяются раньше. Системы с одинаковыми номерами проверяются, если это возможно, параллельно.

В настоящее время Linux поддерживает следующие типы файловых систем (см. страницу man fs, где дано их краткое описание): minix, ext, ext2, xia, msdos, umsdos, vfat, proc, nfs, iso9660, hpfs, sysv, smb, ncpfs. Вместо типа файловой системы в поле "тип файловой системы" (и в опции -t команды mount) можно задать значение auto. В таком случае команда mount попытается самостоятельно определить тип монтируемой файловой системы. Однако это во многих случаях приводит к ошибкам, поэтому лучше указать тип явно. Можно перечислить несколько типов (через запятую). В команде mount можно также вначале задать список типов файловых систем, которые не надо монтировать. Этот список задается с помощью флага no. Такая возможность может оказаться полезной в том случае, когда используется команда mount с аргументом -a. По этой команде производится монтирование всех файловых систем, перечисленных в файле /etc/fstab. С помощью дополнительного аргумента -t type в этом случае можно ограничиться монтированием файловых систем только определенного типа, а с помощью флага no можно указать типы, которые не надо монтировать. Например, команда

```
[root]# mount -a -t nomsdos,ext
```

монтирует все файловые системы, за исключением тех, которые относятся к типам msdos И ext.

Когда монтируется файловая система, упомянутая в файле /etc/fstab, то в команде монтирования достаточно указать только один аргумент — либо имя устройства, либо точку монтирования. Все остальные параметры команда mount возьмет из файла /etc/fstab.

Обычно монтировать файловые системы может только суперпользователь, но если в поле опций монтирования файла /etc/fstab указать опцию user, то соответствующую файловую систему смогут смонтировать все пользователи. Так, если в /etc/fstab имеется строка

```
/dev/cdrom /cd iso9660 ro,user,noauto,unhide
```

то любой пользователь сможет смонтировать файловую систему на своем CD-ROM, используя команду

```
[user]$ mount /dev/cdrom
```
или
```
[user]$ mount /cd
```

В табл. 4.8 приведены еще некоторые опции, которые могут использоваться в команде mount и в файле /etc/fstab.

Таблица 4.8. *Основные опции команды mount*

Опция	Значение
async	Весь ввод/вывод в файловую систему должен производиться асинхронно
auto	Файловая система может монтироваться командой mount с опцией -a
defaults	Использовать набор опций, задаваемый по умолчанию: rw, suid, dev, exec, auto, nouser, async
dev	Файлы байт-ориентированных и блок-ориентированных устройств в файловой системе интерпретируются как специальные файлы
noauto	Файловая система может монтироваться только явно. Опция -a не приведет к монтированию файловой системы
exec	Разрешает выполнение двоичных файлов
remount	Позволяет перемонтировать уже смонтированную файловую систему. Обычно используется для изменения опций монтирования файловой системы, особенно для того, чтобы расширить права доступа (вместо прав только на чтение установить права на чтение/запись)
ro	Монтирует файловую систему только на чтение
rw	Монтирует файловую систему для чтения и записи
suid	Позволяет задействовать биты смены идентификатора пользователя и смены идентификатора группы

Таблица 4.8 *(окончание)*

Опция	Значение
sync	Весь ввод/вывод в файловую систему должен производиться синхронно
user	Позволяет непривилегированному пользователю монтировать файловую систему. Для таких пользователей монтирование всегда выполняется с опциями noexec, nosuid, nodev
nodev	Файлы байт-ориентированных и блок-ориентированных устройств в файловой системе не интерпретируются как специальные файлы
nosuid	Но позволяет задействовать биты смены идентификатора пользователя и смены идентификатора группы
nouser	Запрещает непривилегированному пользователю монтировать файловую систему

Команды mount и umount поддерживают в актуальном состоянии таблицу (или перечень) смонтированных файловых систем. Этот перечень сохраняется на диске в виде файла /etc/mtab. Этот файл можно просмотреть непосредственно, или вывести на экран командой mount без аргументов.

Если вы хотите монтировать какую-то систему только для чтения из нее, то в соответствующей строке файла /etc/fstab надо либо указать опцию -r (read only, по умолчанию подразумевается rw, т. е. и чтение, и запись), либо использовать команду mount с параметром -r.

Перед тем как отключить от компьютера носитель, на котором расположена файловая систгма (чаще всего это требуется делать с дискетами в дисководе, но иногда и с носителями других типов), необходимо "размонтировать" файловую систему (другими словами, "размонтировать носитель"). Эта операция выполняется с помощью команды umount (замечание для тех, кто знает английский язык: имя команды не является правильным английским словом, так что не вставляйте в него лишнюю букву "n"). В качестве аргумента команде umount надо дать либо имя устройства, либо точку монтирования.

Демонтировать файловую систему может только тот пользователь, который ее смонтировал (и суперпользователь, конечно). Для того чтобы операцию демонтирования мог выполнить любой пользователь, в поле параметров соответствующей строки файла /etc/fstab надо указать опцию users (вместо user).

Демонтирование файловой системы возможно только тогда, когда в ней нет открытых файлов (в частности, не должно быть запущено программ, файлы которых расположены в данной системе) и в системе нет процессов, использующих згу файловую систему (то есть демонтируемая файловая система не должна быть занятой).

Надо признать, что, в сравнении с DOS или Windows, работа с дискетами и другими сменными носителями (CD-ROM, Iomega Zip и т. п.) в ОС типа UNIX менее удобна. Для того чтобы обратиться к дискете, вы должны вначале смонтировать ее в какой-то каталог в файловой структуре. А для смены дискеты требуется вначале размонтировать предыдущую дискету, а затем смонтировать новую. То же самое с другими сменными носителями. Поскольку с дискетами и другими сменными носителями так или иначе приходится работать, лучше сразу заготовить в файловой структуре стандартные каталоги для монтирования всех имеющихся в вашем распоряжении типов сменных носителей. Например, создать каталог /mnt, а в нем подкаталоги floppy, cdrom, zip и т. д.

Пожалуй, это все, что нужно знать начинающему пользователю системы Linux о файловых системах. Еще раз повторю, что здесь была рассмотрена только одна сторона файловой системы ext2fs, обращенная к пользователям, т. е. в основном, логика построения структуры каталогов и файлов. Обратная сторона файловой системы, ее внутреннее устройство, а также различные типы файловых систем, в настоящем разделе только упоминались по мере необходимости. Всем этим вопросам будет посвящена *гл. 16.*

Глава 5

Оболочка bash

В этой главе мы рассмотрим работу *с* системой Linux в текстовом режиме, другими словами, с консоли или терминала. Начинающему пользователю может казаться, что он никогда не будет работать в этом режиме, поскольку существует графический режим. Однако это мнение ошибочно, ибо многие действия оказывается быстрее и удобнее выполнять именно в текстовом режиме. Даже находясь в графическом режиме, вы часто будете открывать окно эмулятора терминала и выполнять необходимые действия в этом окне. Ведь текстовый режим ОС Linux — это совсем не то, что текстовый режим однозадачной MS-DOS. Поскольку Linux — это многозадачная ОС, уже в текстовом режиме обеспечивается возможность работы в нескольких окнах (о переключении терминалов мы рассказали в *разд. 3.3).* А для редактирования простого текста или HTML-файла вовсе не обязательно запускать достаточно тяжеловесную и медленную (особенно на "слабых" компьютерах) графическую оболочку.

5.1. Что такое оболочка?

Как уже упоминалось выше, хотя мы часто говорим, что "пользователь работает с операционной системой", фактически это не верно, поскольку на деле взаимодействие с пользователем организует специальная программа. Существует два вида таких программ — оболочка, или shell, для работы в текстовом режиме (интерфейс командной строки) и графический интерфейс пользователя GUI (Graphical User Interface), организующий взаимодействие с пользователем в графическом режиме.

Сразу надо сказать, что в принципе любая программа в Linux может быть запущена как через оболочку (если запущен X-сервер, *см. гл. 6),* так и через графический интерфейс пользователя. Запуск программ из оболочки эквивалентен (двойному) щелчку мышкой по иконке программы в GUI. Передача аргументов программе в текстовом режиме аналогична тому, что мы "бросаем" что-то на иконку программы в графическом. Но, с другой стороны, некоторые программы не приспособлены для запуска через GUI и, соответственно, могут быть исполнены только из командной строки.

Собственно говоря, название "оболочка" не выдерживает критики. На мой взгляд, правильнее было бы называть эту программу командным процессором, как называют command.com в MS-DOS, или интерпретатором команд.

Но так уж повелось во всех UNIX-системах, что интерпретатор команд для текстового режима называют оболочкой.

Когда-то (в первых UNIX-системах) это была программа с именем sh, которое было сокращением от *shell.* Потом были разработаны несколько ее улучшенных вариантов, в частности, *Bourne shell* — расширенная версия sh, написанная Стивом Борном (Steve Bourne). В рамках проекта GNU (проект Р. Столлмана по разработке свободного ПО, см. www.gnu.org) была создана оболочка bash, название которой расшифровывается как *Bourne-again shell,* т. е. "снова оболочка Борна". По-английски в этом названии просматривается еще и игра слов, связанная с тем, что Bourne звучит как borne (рождаться, рожденный), и получается "заново рожденная shell". Оболочка bash была написана Брайеном Фоксом (Brian Fox — основной разработчик) и Четом Рэми (Chet Rarney). Именно bash мы и будем далее рассматривать, и всюду ниже, где говорится об оболочке вообще, вы смело можете считать, что речь идет о bash.

Сама по себе оболочка bash не выполняет никаких прикладных задач. Но она обеспечивает выполнение всех приложений: нахождение вызываемых программ, их запуск и организацию ввода/вывода. Кроме того, оболочка отвечает за работу с переменными окружения и выполняет некоторые преобразования (подстановки) аргументов. Но главное свойство оболочки, которое делает ее мощным инструментом пользователя — это то, что она включает в себя простой язык программирования. Как давно доказано в математике, любой алгоритм можно построить из пары-тройки основных операций и одного условного оператора. Реализацию условных операторов (а также операторов цикла) и берет на себя оболочка. Она использует все остальные утилиты и программы (и те, которые имеются в составе операционной системы, и те, что устанавливаются отдельно) как базовые операции поддерживаемого ею языка программирования, обеспечивает передачу им аргументов, а также передачу результатов их работы другим программам или пользователю. В результате получается очень мощный язык программирования. И в этом основная сила и одна из существенных функций оболочки.

Прежде чем читать дальше этот раздел, вернитесь ненадолго к *разд. 3.4 и* еще раз просмотрите основные комбинации клавиш, используемые для управления вводом в текстовом режиме. Вспомните по крайней мере то, как пользоваться клавишами <Ctrl>+<C>, <Ctrl>+<D>, <Tab> и клавишами со стрелками.

5.2. Специальные символы

Оболочка bash использует несколько символов из числа 256 символов набора ASCII в специальных целях либо для обозначения некоторых операций, либо для преобразования выражений. В число таких символов входят:

```
` ~ ! @ # $ % ^ & * ( ) _ — [ ] { } : ; ' " / \ > <
```

а также символ с кодом 0, символ возврата каретки (генерируемый клавишей <Enter>) и пробел. В зависимости от ситуации эти специальные символы могут трактоваться либо в их специальном значении, либо в буквальном, т. е. как литералы. Но мы в основном будем предполагать, что все эти символы зарезервированы и не должны использоваться в качестве литералов. Это касается в первую очередь использования их в именах файлов и каталогов, о чем мы уже говорили в *гл. 4.* Однако символы _ , - и . (знак подчеркивания, дефис и точка) часто используются в именах файлов, так что именно этот пример показывает, что специальное значение эти символы имеют не всегда. В именах файлов только символы точки (.) и слэша (/) имеют специальное значение. Символ слэша служит для разделения имен отдельных каталогов, а точка имеет специальное значение только если она является первым символом в имени файла (что означает, что файл является "скрытым").

Давать сейчас точное определение того, какое специальное значение и в каких ситуациях имеет тот или иной специальный символ, нецелесообразно. Мы будем рассматривать их постепенно в следующих разделах, по мере того, как они потребуются. Однако есть три символа, которые имеют особое значение и которые поэтому необходимо рассмотреть в первую очередь.

Символ \ (обратный слэш) можно назвать "символом отмены специального значения" для любого из специальных символов, который стоит сразу вслед за \. Например, если мы хотим использовать символ пробела в имени файла, мы должны вместо простого пробела поставить \. Например, возможна следующая команда:

```
[user]$ cp two_words two\ words
```

Символы ' и " (одинарные и двойные кавычки) могут быть названы "символами цитирования". Любой из этих символов всегда используется в паре с его копией для обрамления какого-то выражения, совсем как в обычной прямой речи. Если какой-то текст взят в одинарные кавычки, то все символы внутри этих кавычек воспринимаются как литералы, никаким из них не придается специального значения. Если вернуться к тому же примеру с пробелами в имени файла, то можно сказать, что для того, чтобы дать файлу имя "two words" надо взять имя в кавычки:

```
[user]$ cp two_words 'two words'
```

Различие в использовании символов ' и " состоит в том, что внутри одинарных кавычек теряют специальное значение все символы, а внутри двойных кавычек — все специальные символы кроме $, ' и \ (знака доллара, одинарных кавычек и обратного слэша).

5.3. Выполнение команд

Как было отмечено выше, одна из основных функций оболочки состоит в том, чтобы организовать исполнение команд пользователя, вводимых им в командной строке. В частности, оболочка предоставляет пользователю два специальных оператора для организации задания команд в командной строке: ; и &.

5.3.1. Оператор ;

Хотя чаще всего пользователь задает команды в командной строке по одной, имеется возможность задать в одной строке несколько команд, которые будут выполнены последовательно, одна за другой. Для этого используется специальный символ — оператор ;. Если не поставить этот разделитель команд, то последующая команда может быть воспринята как аргумент предыдущей. Таким образом, если написать в командной строке что-то вроде

```
[user]$ commandl ; command2
```

то оболочка вначале запустит на выполнение команду command1, дождется, пока ее выполнение завершится, после чего запустит command2, дождется ее завершения, после чего снова выведет приглашение командной строки, ожидая следующих действий пользователя.

5.3.2. Оператор &

Оператор & используется для того, чтобы организовать исполнение команд в фоновом режиме. Если поставить значок & после команды, то оболочка вернет управление пользователю сразу после запуска команды, не дожидаясь, пока выполнение команды завершится. Например, если задать в командной строке "commandl & command2 &", ТО оболочка запустит команду coiranandl, сразу же затем команду command2 и затем немедленно вернет управление пользователю.

5.3.3. Операторы && и //

Операторы && и || являются управляющими операторами. Если в командной строке СТОИТ commandl && command2, ТО command2 выполняется В ТОМ И ТОЛЬКО в том случае, если статус выхода из команды commandl равен нулю, что говорит об успешном ее завершении. Аналогично, если командная строка имеет вид commandl || commanda, то команда commanda выполняется тогда и только тогда, когда статус выхода из команды commandl отличен от нуля.

Сама техника организации запуска команд на выполнение не является предметом нашего рассмотрения. Можно только кратко сказать, что обо-

лочка должна найти код команды, загрузить его в память, передать команде аргументы, заданные в командной строке, а после завершения выполнения соответствующего процесса передать каким-то образом пользователю или другому процессу результаты выполнения данной команды. Эти этапы мы кратко и рассмотрим.

Итак, первый этап — поиск кода команды. Команды бывают встроенные (те, код которых включен в код самой оболочки) и внешние (код которых расположен в отдельном файле на диске). Встроенную команду оболочка всегда найдет, а для поиска внешней команды пользователь, в принципе, должен указать оболочке полный путь до соответствующего файла. Однако для облегчения жизни пользователей оболочка умеет искать внешние команды в каталогах, которые перечислены в специально заданных "путях поиска". Только если она не находит нужных файлов в таких каталогах, она решает, что пользователь ошибся при вводе имени команды. О том, как включить каталог в пути поиска, будет сказано ниже, а сейчас рассмотрим, как оболочка организует передачу данных исполняемой команде и выдачу результатов пользователю.

5.4. Стандартный ввод/вывод

5.4.1. Потоки ввода/вывода

Когда программа запускается на выполнение, в ее распоряжение предоставляются три потока (или канала).

- **Стандартный ввод** (standard input или stdin). По этому каналу данные передаются программе.
- **Стандартный вывод** (standard output или stdout). По этому каналу программа выводит результаты своей работы.
- **Стандартный поток сообщений об ошибках** (standard error или stderr). По этому каналу программы выдают информацию об ошибках.

Из стандартного ввода программа может только читать, а два других потока могут использоваться программой только для записи.

По умолчанию входной поток связан с клавиатурой, а выходной поток и поток сообщений об ошибках направлены на терминал пользователя. Другими словами, вся выходная информация запущенной пользователем команды или программы, а также все сообщения об ошибках выводятся в окно терминала. Однако, как мы увидим чуть ниже, можно перенаправить выходные сообщения (например, в файл).

Для того чтобы продемонстрировать, как работает стандартный поток ошибок, выполните команду `ls` с неверным аргументом, например, задав в качестве аргумента имя несуществующего файла. В таком случае `ls` выведет со-

общение об ошибке в стандартный поток ошибок. Для нас, однако, в данном случае стандартный поток ошибок неотличим от выходного потока, поскольку сообщение об ошибке мы видим в окне терминала.

Работу со стандартными входным и выходным потоками лучше всего проиллюстрировать на примере команд echo и cat.

5.4.2. Команда *echo*

Команда echo предназначена для выдачи на стандартный вывод строки символов, которая задана ей в качестве аргумента. После этого она выдает сигнал перевода строки и завершается. Попробуйте выполнить команду

```
[user]$ echo 'Еривет, дружище!'
```

и, думаю, дальнейших пояснений не потребуется (только используйте именно одиночные кавычки, иначе результат может быть несколько иным).

5.4.3. Команда *cat*

Мы уже рассматривали кратко команду cat в *гл. 4.* В данном разделе эта команда интересует нас в основном потому, что чаще всего она работает как раз с входным и выходным потоками. По умолчанию выход команды cat направляется в выходной поток. Чтобы убедиться, что эта команда по умолчанию воспринимает входной поток, запустите команду cat без аргументов. В результате курсор переместится в новую строку, и более как будто ничего не будет происходить. В это время команда ожидает поступления символов во входном потоке. Введите любой символ, и вы увидите, что он сразу же появился на экране, что говорит о том, что программа сразу же направила его в выходной поток. Можно продолжить ввод символов, и они также появятся на экране.

Обычно клавиатура настроена на построчный ввод, поэтому если вы нажмете клавишу <Enter>, последняя набранная строка передается команде cat, которая вновь выводит данные на монитор через стандартный вывод. Таким образом, каждая строка будет показана дважды: один раз при наборе и второй раз — командой cat.

Если нажать комбинацию клавиш <Ctrl>+<D>, которая служит командой окончания процедуры ввода, вы вновь вернетесь к подсказке в командной строке. Можно также использовать комбинацию клавиш <Ctrl>+<C>, которая является в оболочке командой завершения работы запущенной программы.

Если команде cat в качестве аргумента задать имя файла, это будет означать, что содержимое файла будет направлено во входной поток, откуда его примет команда cat и выдаст в выходной поток. Но это только частный

случай перенаправления ввода, очень полезного механизма оболочки, который, безусловно, заслуживает более подробного рассмотрения.

5.5. Перенаправление ввода/вывода, каналы и фильтры

Хотя обычно, как было сказано, ввод/вывод программы связан со стандартными потоками, в оболочке существуют специальные средства для перенаправления ввода/вывода.

5.5.1. Операторы >, <и >>

Для обозначения перенаправления используются символы ">", "<" и ">>". Чаще всего используется перенаправление вывода команды в файл. Вот соответствующий пример:

```
[user]$ ls -l > /home/jim/dir.txt
```

По этой команде в файле /home/jim/dir.txt будет сохранен перечень файлов и подкаталогов того каталога, который был текущим на момент выполнения команды ls; при этом если указанного файла не существовало, то он будет создан; если он существовал, то будет перезаписан; если же вы хотите, чтобы вывод команды был дописан в конец существующего файла, то надо вместо символа > использовать >>. При этом наличие пробелов до или после символов > или >> несущественно и служит только для удобства пользователя.

Вы можете направить вывод не только в файл, но и на вход другой команды или на устройство (например, принтер). Так, для подсчета числа слов в файле /home/jim/report.txt можно использовать следующую команду:

```
[user]$ cat /home/jim/report.txt > wc -w
```

а для вывода файла на печать — команду

```
[user]$ cat /home/jim/report.txt > lpr
```

Как видите, оператор > служит для перенаправления выходного потока. По отношению к входному потоку аналогичную функцию выполняет оператор <. Приведенный выше пример команды для подсчета числа слов в определенном файле можно переписать следующим образом (обратите внимание на отсутствие команды cat):

```
[user]$ wc -w < /home/jim/report.txt
```

Этот вариант перенаправления часто используется в различных скриптах применительно к тем командам, которые обычно воспринимают ввод (или ожидают ввода) с клавиатуры. В скрипте же, автоматизирующем какие-то рутинные операции, можно дать команде необходимую информацию из

файла, в который заранее записано то, что нужно ввести для выполнения этой команды.

В силу того, что символы <, > и >> действуют на стандартные потоки, их можно использовать не только тем привычным образом, как это делается обычно, но и несколько по-другому. Так, следующие команды эквивалентны:

```
[user]$ cat > file
[user]$ cat>file
[user]$ >file cat
[user]$ > file cat
```

Однако сам пс себе (без какой-либо команды, для которой определены стандартные потоки) символ перенаправления не может использоваться, так что нельзя, например, введя в командной строке

```
[user]$ file1 > file2
```

получить копию какого-то файла. Но это не уменьшает значения данного механизма, ведь стандартные потоки определены для любой команды. При этом перенаправить можно не только стандартный ввод и вывод, но и другие потоки. Для этого надо указать перед символом перенаправления номер перенаправляемого потока. Стандартный ввод stdin имеет номер 0, стандартный вывод stdout — номер 1, стандартный поток сообщений об ошибках stderr — номер 2. То есть полный формат команды перенаправления имеет вид (напомним, что пробелы возле > не обязательны):

```
command N > M
```

где N и м — номера стандартных потоков (О, 1, 2) или имена файлов. Употребление в некоторых случаях символов <, > и >> без указания номера канала или имени файла возможно только потому, что вместо отсутствующего номера по умолчанию подставляется 1, т. е. стандартный вывод. Так, оператор > без указания номера интерпретируется как 1 >.

Кроме простого перенаправления стандартных потоков существует еще возможность не просто перенаправить поток в тот или иной канал, а сделать копию содержимого стандартного потока. Для этого служит специальный символ &, который ставится перед номером канала, на который перенаправляется поток:

```
command N > &M
```

Такая команда означает, что выход канала с номером N направляется как на стандартный вывод, так и дублируется в канал с номером м. Например, для того, чтобы сообщения об ошибках дублировались на стандартный вывод, надо дать команду 2>&1, в то время как 1>&2 дублирует stdout в stderr. Такая возможность особенно полезна при перенаправлении вывода в файл, т. к. мы тогда одновременно и видим сообщения на экране, и сохраняем их в файле.

5.5.2. Оператор /

Особым вариантом перенаправления вывода является организация программного канала (иногда называется трубопроводом или конвейером). Для этого две или несколько команд, таких, что вывод предыдущей служит вводом для следующей, соединяются (или разделяются, если вам это больше нравится) символом вертикальной черты -- |. При этом стандартный выходной поток команды, расположенной слева от символа |, направляется на стандартный ввод программы, расположенной справа от символа |. Например:

```
[user]$ cat myfile | grep Linux | wc -1
```

Эта строка означает, что вывод команды cat, т. е. текст из файла myfile, будет направлен на вход команды grep, которая выделит только строки, содержащие слово "Linux". Вывод команды grep будет, в свою очередь, направлен на вход команды wc -l, которая подсчитает число таких строк.

Программные каналы используются для того, чтобы скомбинировать несколько маленьких программ, каждая из которых выполняет только определенные преобразования над своим входным потоком, для создания обобщенной команды, результатом которой будет какое-то более сложное преобразование.

Надо отметить, что оболочка одновременно вызывает на выполнение все команды, включенные в конвейер, запуская для каждой из команд отдельный экземпляр оболочки, так что как только первая профамма начинает что-либо выдавать в свой выходной поток, следующая команда начинает его обрабатывать. Точно так же каждая следующая команда выполняет свою операцию, ожидая данных от предыдущей команды и выдавая свои результаты на вход последующей. Если вы хотите, чтобы какая-то команда полностью завершилась до начала выполнения последующей, вы можете использовать в одной строке как символ конвейера |, так и точку с запятой ;. Перед каждой точкой с запятой оболочка будет останавливаться и ожидать, пока завершится выполнение всех предыдущих команд, включенных в конвейер.

Статус выхода (логическое значение, возвращаемое после завершения работы программы) из канала совпадает со статусом выхода, возвращаемым последней командой конвейера. Перед первой командой конвейера можно поставить символ !, тогда статус выхода из конвейера будет логическим отрицанием статуса выхода из последней команды. Оболочка ожидает завершения всех команд конвейера, прежде чем установить возвращаемое значение.

5.5.3. Фильтры

Последний из приведенных выше примеров (с командой grep) можно использовать для иллюстрации еще одного важного понятия, а именно, программы-фильтра. Фильтры — это команды (или профаммы), которые воспринимают входной поток данных, производят над ним некоторые

преобразования и выдают результат на стандартный вывод (откуда его можно перенаправить куда-то еще по желанию пользователя). К числу команд-фильтров относятся уже упоминавшиеся выше команды cat, more, less, wc, cmp, diff, а также команды, приведенные в табл. 5.1.

Таблица 5.1. *Команды-фильтры*

Команда	Краткое описание
grep, fgrep, egrep	Ищут во входном файле или данных со стандартного ввода строки, содержащие указанный шаблон, и выдают их на стандартный вывод
tr	Заменяет во входном потоке все встречающиеся символы, перечисленные в заданном перечне, на соответствующие символы из второго заданного перечня
comm	Сравнивает два файла по строкам и выдает на стандартный вывод 3 колонки: в первой — строки, которые встречаются только в 1-ом файле, во второй — строки, которые встречаются только во 2-ом файле, и в третьей — строки, имеющиеся в обоих файлах
pr	Форматирует для печати текстовый файл или содержимое стандартного ввода
sed	Строковый редактор, использующийся для выполнения некоторых преобразований над входным потоком данных (берется из файла или со стандартного ввода)

Особым фильтром является команда tee, которая "раздваивает" входной поток, с одной стороны направляя его на стандартный вывод, а с другой — в файл (имя которого вы должны задать). Легко видеть, что по своему действию команда tse аналогична оператору перенаправления `1>&file`.

Возможности фильтров можно существенно расширить за счет использования регулярных выражений, позволяющих организовать, например, поиск по различным, зачастую очень сложным, шаблонам.

О перенаправлении и фильтрах можно было бы говорить очень много. Но этот материал имеется в большинстве книг по UNIX и Linux, например у Петерсена *(см. [П1.4] приложения)* и Келли-Бутла *(см. [П1.8] приложения)*. Поэтому ограничимся сказанным, и перейдем к рассмотрению так называемой среды или окружения, создаваемого оболочкой.

5.6. Параметры и переменные. Окружение оболочки

Понятие параметра в оболочке bash подобно понятию переменной в обычных языках программирования. Именем (или идентификатором) параметра может быть слово, состоящее из алфавитных символов, цифр и знаков под-

черкивания (только первый символ этого слова не может быть цифрой), а также число или один из следующих специальных символов: *, @, #, ?, - (дефис), $, !, 0, _ (подчеркивание).

Говорят, что параметр задан или установлен, если ему присвоено значение. Значением может быть и пустая строка. Чтобы вывести значение параметра, используют символ $ перед его именем. Так, команда

```
[user]$ echo name
```

выдаст на экран слово name, а команда

```
[user]$ echo $name
```

выдаст значение переменной name (если таковое, конечно, задано).

5.6.1. Разновидности параметров

Параметры разделяются на три класса: *позиционные параметры, специальные параметры* (именами которых как раз и служат перечисленные только что специальные символы) и *переменные оболочки*.

Имена (идентификаторы) *позиционных параметров* состоят из одной или более цифр (только не из одиночного нуля). Значениями позиционных параметров являются аргументы, которые были заданы при запуске оболочки (первый аргумент является значением позиционного параметра 1, и т. д.). Изменить значение позиционного параметра можно с помощью встроенной команды set. Значения этих параметров изменяются также на время выполнения оболочкой одной из функций (об этом будет рассказано в *разд. 5.8)*.

Специальные параметры являются шаблонами, замена (подстановка) которых производится следующим образом (табл. 5.2).

Таблица 5.2. *Специальные параметры.*

Параметр	Правила замены
*	Заменяется позиционными параметрами, начиная с первого. Если замена производится внутри двойных кавычек, то этот параметр заменяется на одно-единственное слово, составленное из всех позиционных параметров, разделенных первым символом специальной переменной IFS (о ней будет сказано ниже). То есть "$*" эквивалентно "$1c$2c. . .", где с— первый символ в значении переменной IFS. Если IFS присвоено пустое значение или ее значение не установлено, параметры разделяются пробелами
@	Заменяется позиционными параметрами, начиная с первого. Если замена производится внутри двойных кавычек, то каждый параметр заменяется отдельным словом. Так, "$@" эквивалентно "$1" "$2" ... Если позиционных параметров нет, то значение не присваивается (параметр @ просто удаляется)

Таблица 5.2 *(окончание)*

Параметр	Правила замены
#	Заменяется десятичным значением числа позиционных параметров
?	Заменяется статусом выхода последнего из выполнявшихся на переднем плане программных каналов
- (дефис)	Заменяется текущим набором значений флагов, установленных с помощью встроенной команды set или при запуске самой оболочки
$	Заменяется идентификатором процесса (PID) оболочки
!	Заменяется идентификатором процесса (PID) последней из выполняющихся фоновых (асинхронно выполнявшихся) команд
0	Заменяется именем оболочки или запускаемого скрипта. Если bash запускается для выполнения командного файла, $0 равно имени этого файла. В противном случае это значение равно полному пути к оболочке
_ (подчеркивание)	Заменяется последним аргументом предыдущей из выполнявшихся команд (если это параметр или переменная, то подставляется ее значение)

Специальные параметры, перечисленные в приведенной выше таблице, отличаются тем, что на них можно только ссылаться; присваивать им значения нельзя.

Переменная с точки зрения оболочки — это параметр, обозначаемый именем. Значения переменным присваиваются с помощью оператора следующего вида

```
[user]$ name=value
```

где name — имя переменной, а value — присваиваемое ей значение (может быть пустой строкой). Имя переменой может состоять только из цифр и букв и не может начинаться с цифры. Значением может быть любой текст. Если значение содержит специальные символы, то его надо взять в кавычки. Присвоенное значение этих кавычек не содержит, естественно. Если переменная задана, то ее можно удалить, используя встроенную команду оболочки unset.

Набор всех установленных переменных оболочки с присвоенными им значениями называется окружением (environment) или средой оболочки. Вы можете просмотреть его с помощью команды set без параметров (только, может быть, следует организовать конвейер set | less). В выводе этой команды все переменные окружения перечисляются в алфавитном порядке. Для того чтобы просмотреть значение одной конкретной переменной, вместо команды set (в выводе которой нужную переменную еще искать и искать) можно воспользоваться командой

```
[user]$ echo $name
```

(правда, в этом случае вы должны знать имя интересующей вас переменной).

Среди переменных, которые вы увидите в выводе команды set, встречаются очень интересные переменные. Обратите, например, внимание на переменную RANDOM. Если вы несколько раз подряд выполните команду

```
[user]$ echo $RANDOM
```

вы каждый раз будете получать новое значение. Дело в том, что эта переменная возвращает случайное целое из интервала 0—32 768.

5.6.2. Приглашения оболочки

Одна из очень важных переменных имеет имя PS1. Эта переменная задает вид приглашения, которое bash выводит, когда ожидает ввода очередной команды пользователем. По умолчанию этой переменной присвоено значение "\s-\v\$". Вообще-то в оболочке bash существует четыре приглашения, которые используются в разных ситуациях. Переменная PS1 задает вид строки приглашения, которая выдается тогда, когда оболочка ждет ввода команды. Вторичное приглашение, задаваемое переменной PS2, появляется тогда, когда оболочка ожидает от пользователя ввода еще каких-то данных, необходимых для продолжения работы запущенной команды или программы. По умолчанию переменная PS2 имеет значение ">". Вы уже имели возможность видеть это приглашение, когда запускали команду cat для ввода данных с клавиатуры в файл. Другой пример — команда ftp, после запуска которой приглашение тоже принимает такой вид.

Приглашение, задаваемое переменной PS3, используется в команде select. Приглашение, задаваемое переменной PS4, выводится перед каждой командой в то время, когда bash отслеживает процесс выполнения. Значение по умолчанию - "+".

Если у вас есть такое желание, вы можете изменить вид переменных PS1 и PS2. При этом можно использовать как любые символы, вводимые с клавиатуры, так и некоторое число специальных символов, которые при формировании строки приглашения декодируются в соответствии с табл. 5.3 (приводим только некоторые из них, для примера; полный список см. в man-странице по утилите bash).

Таблица 5.3. *Специальные символы для формирования приглашения*

Символ	Его значение
\a	Звуковой сигнал (ASCII-код 07)
\d	Дата в формате "День, месяц, число", например, Срд, Окт, 17
\h	Имя хоста (hostname) до первой точки
\н	Полное имя хоста

Таблица ***5.3*** *(окончание)*

Символ	Его значение
\t	Текущее время в 24-часовом формате: HH:MM:SS (часы:минуты:секунды)
\T	Текущее время в 12-часовом формате: HH:MM:SS
\@	Текущее время в 12-часовом формате AM/PM
\u	Имя пользователя, запустившего оболочку
\w	Полное имя текущего рабочего каталога (начиная с корня)
\W	Текущий рабочий каталог (без указания пути)
\$	Символ #, если оболочка запущена суперпользователем, и символ $, если оболочка запущена обычным пользователем
\nnn	Символ, имеющий восьмеричный код nnn
\n	Новая строка (перевод строки)
\s	Имя оболочки
\#	Текущий номер команды
\\	Обратный слэш (a backslash)
\[	Начало последовательности непечатаемых символов (этот символ может быть использован для того, чтобы включить в текст подсказки последовательность управляющих символов терминала)
\]	Конец последовательности непечатаемых символов
\!	Порядковый номер данной команды в истории команд

Текущий номер команды (порядковый номер выполняемой команды в рамках текущей сессии) может отличаться от номера данной команды в списке истории команд, поскольку последний включает в себя команды, которые были сохранены в файле истории команд.

После того как значение переменной, определяющей подсказку, прочитано оболочкой, в нем могут быть произведены подстановки в соответствии с правилами расширения параметров, подстановок в именах команд и арифметических выражениях, а также разбиения слов (word splitting). Об этих правилах будет рассказано чуть ниже, в *разд. 5.7.*

Например, после выполнения команды (поскольку в строке имеется пробел, кавычки обязательны)

```
[root]# PS1="[\u@\h \W]\$"
```

в стандартном приглашении будет выводиться квадратная скобка, имя пользователя, символ @, имя компьютера, пробел, название текущего каталога (без указания пути), закрывающая квадратная скобка и символ $ (если в

оболочке работает простой пользователь) или § (если оболочка запущена от имени пользователя root).

5.6.3. Переменная *PATH*

Еще одна очень важная переменная имеет имя PATH. Она задает перечень путей к каталогам, в которых bash осуществляет поиск файлов (в частности, файлов с командами) в тех случаях, когда полный путь к файлу не задан в командной строке. Отдельные каталоги в этом перечне разделяются двоеточиями. По умолчанию переменная PATH включает каталоги /usr/local/bin, /bin, /usr/bin, /usr/X11R6/bin, т. е. имеет вид:

```
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:
```

Для того чтобы добавить каталог в этот список, нужно выполнить следующую команду:

```
[root]# PATH=$PATH:new_path
```

При осуществлении поиска оболочка просматривает каталоги именно в том порядке, как они перечислены в переменной PATH.

Отметим, что можно включить в этот список и текущий каталог, добавив в переменную PATH точку. Однако этого не рекомендуется делать из соображений безопасности: злоумышленник может положить в общедоступный каталог команду, имя которой совпадает с одной из часто выполняемых суперпользователем команд, но выполняющую совершенно другие действия (особенно если текущий каталог стоит в начале перечня путей поиска).

5.6.4. Переменная /FS

Эта переменная задает разделители полей (Internal Field Separator), которые используются при операции разделения слов при преобразованиях командной строки, выполняемых оболочкой перед тем, как запустить командную строку на исполнение. Значение этой переменной по умолчанию — "<Пробел><Tab><Символ_новой_строки>".

5.6.5. Текущий и домашний каталоги

Имя текущего каталога сохраняется в переменной окружения (с именем PWD), и значение этой переменной изменяется при каждом запуске программы cd (а также при смене текущего каталога любым другим способом, например, через Midnight Commander).

Аналогичным образом полное имя (с указанием пути) домашнего каталога пользователя, запустившего данный процесс, сохраняется в переменной HOME.

5.6.6. Команда *export*

Когда оболочка запускает на выполнение какую-то программу или команду, она передает *им* часть переменных окружения. Для того чтобы переменная окружения передавалась запускаемому из оболочки процессу, ее нужно задавать с помощью специальной команды export, т. е. вместо

```
[user]$ name=value
```

надо записать

```
[user]$ export name=value
```

В таком случае все запускаемые из оболочки программы (в том числе вторичные экземпляры самой оболочки) будут иметь доступ к заданным таким образом переменным, т. е. могут вызывать их значения по именам.

5.7. Раскрытие выражений

Когда оболочка получает какую-то командную строку на выполнение, она до начала выполнения команды осуществляет "грамматический разбор" полученной командной строки. Одним из этапов такого "разбора" является раскрытие или подстановка выражений (expansion). В оболочке bash имеется семь типов подстановки выражений:

- раскрытие скобок (brace expansion);
- замена знака тильды (tilde expansion);
- подстановка параметров и переменных;
- подстановка команд;
- арифметические подстановки (выполняемые слева направо);
- разделение слов (word splitting);
- раскрытие шаблонов имен файлов и каталогов (pathname expansion).

Все эти операции выполняются именно в том порядке, как они здесь перечислены. Рассмотрим их последовательно.

5.7.1. Раскрытие скобок

Раскрытие скобок проще всего пояснить на примере. Предположим, что нам нужно создать сразу несколько подкаталогов в каком-то каталоге, или поменять владельца сразу у нескольких файлов. Эти действия можно выполнить с помощью следующих команд:

```
[user]$ mkdir /usr/local/src/bash/{old,new,dist,bugs}
[root]# chown root /usr/{ucb/{ex,edit},lib/{ex?.?*,how_ex}}
```

В первом случае в каталоге /usr/local/src/bash/ будут созданы подкаталоги old, new, dist и bugs. Во втором случае владелец будет изменен у файлов:

- /usr/ucb/ex
- /usr/lib/ex?.?*
- /usr/ucb/edit
- /usr/lib/ex?.?*

O /usr/ucb/ex
- /usr/lib/how_ex
- /usr/ucb/edit
- /usr/lib/how_ex

То есть для каждой пары скобок генерируются несколько отдельных строк (их число равно числу слов, стоящих внутри скобок) путем приписывания к каждому слову из скобок (спереди) того, что стоит перед скобкой, и приписывания в конец каждого полученного слова того, что стоит после скобки. Еще один пример: строка `a{d,c,b}e` при раскрытии скобок превращается в **три слова** ade ace abe.

Раскрытие скобок выполняется до выполнения других видов подстановок в командной строке, причем все специальные символы, встречающиеся в командной строке, в том числе внутри скобок, сохраняются неизменными (они будут интерпретированы на следующих этапах анализа строки).

5.7.2. Замена тильды

Если слово начинается с символа тильды (~), все символы до первого слэша (или все символы, если слэша нет) трактуются как имя пользователя (login name).

Если это имя есть пустая строка (то есть вслед за тильдой идет сразу слэш), то тильда заменяется на значение переменной `HOME`. Если значение переменной `HOME` не задано, тильда заменяется на полный путь к домашнему каталогу пользователя, запустившего оболочку.

Если вслед за знаком тильды (до слэша) стоит слово, совпадающее с именем одного из легальных пользователей, тильда и имя пользователя заменяются полным именем домашнего каталога этого пользователя. Если слово, следующее за тильдой, не является именем пользователя (и не пусто), то оно остается неизменным.

Если вслед за знаком тильды стоит +, эти два знака заменяются на полное имя текущего каталога (то есть значение переменной `PWD`). Если за знаком тильды следует -, подставляется значение переменой `OLDPWD`.

5.7.3. Подстановка параметров и переменных

Символ $ используется для обозначения операций подстановки параметров, подстановки команд и подстановок арифметических выражений. Выражение или имя, следующее за $, может быть заключено в скобки; что необязательно, но удобно, т. к. позволяет отделить заменяемое выражение от следую-

щих за ним слов или символов. Таким образом, чтобы в командной строке вызвать значение параметра (в частности, любой переменной), нужно вставить выражение вида `${parameter}`.

Скобки необходимы только в том случае, если имя параметра состоит из нескольких цифр, или когда за именем следует символ, который не должен интерпретироваться как часть имени.

Все значения переменных подвергаются подстановке знака тильды, раскрытию параметров и переменных, подстановке команд, подстановкам арифметических выражений, а также удалению специальных символов цитирования (см. ниже). Разделение слов не производится, за исключением случая `"$@"` (объяснение см. в табл. 5.3). Раскрытие шаблонов имен файлов и каталогов не производится.

5.7.4. Подстановка команд

Подстановка команд является очень мощным инструментом bash. Она заключается в замене имени команды на результат ее выполнения. Существует две формы подстановки команд:

`$(command)` и `'command'`

Если применяется вторая из этих форм, то обратный слэш внутри кавычек трактуется как литерал, кроме тех случаев, когда за ним следует $, ', или \. Если же используется форма `$(command)`, все символы внутри скобок составляют команду и ни один из них не считается специальным символом.

Если подстановка производится внутри двойных кавычек, то в результатах подстановки не осуществляется разделение слов и раскрытие шаблонов имен файлов и каталогов.

5.7.5. Арифметические подстановки

Арифметические подстановки позволяют вычислить значение арифметического выражения и подставить вместо него результат. Существует две формы задания арифметических подстановок:

```
$[expression]
$((expression))
```

где expression трактуется так, как если бы оно было заключено в двойные кавычки, но встречающиеся в expression двойные кавычки трактуются как простой литерал. Внутри expression выполняются подстановки параметров и команд.

Синтаксис выражения expression подобен синтаксису арифметических выражений в языке C, подробнее об этом можно прочитать в разделе

ARITHMETIC EVALUATION man-страницы по команде bash. Например, команда

```
[user]$ echo $(( 2 + 3 * 5 ))
```

в качестве результата выдает "17".

Если выражение некорректно, bash выдает сообщение об ошибке.

5.7.6. Разделение слов

После завершения подстановок параметров, команд и арифметических выражений оболочка снова анализирует командную строку (в том виде, который она приобрела к этому моменту) и осуществляет разделение слов (word splitting).

Эта операция заключается в том, что в командной строке ищутся все вхождения символов-разделителей, определенных в переменой IFS, и в соответствующих местах строки разделяются на отдельные слова. Если значение IFS равно пустой строке, разделение слов не производится.

Если в командной строке не производилось никаких подстановок, то разбиение на слова не производится.

5.7.7. Раскрытие шаблонов имен файлов и каталогов

Подстановки имен путей и файлов (pathname expansion) используются для того, чтобы с помощью краткого образца или шаблона указать несколько имен файлов (или каталогов), соответствующих данному шаблону. После разделения слов, если не была задана опция `-f`, bash производит поиск в каждом слове командной строки символов *, ? и `[`. Если будет найдено слово с одним или несколькими вхождениями таких символов, то это слово рассматривается как шаблон, который должен быть заменен словами из лексикографически упорядоченного списка имен путей, соответствующих данному шаблону. Если имен, соответствующих шаблону, не найдено, и переменная `nullglob` не задана, слово не изменяется. Если эта переменная установлена, а путей, соответствующих шаблону, не найдено, слово удаляется из командной строки.

Специальные символы шаблонов имеют следующее значение (табл. 5.4).

Таблица 5.4. *Символы шаблонов*

Символ	Правила замены
*	Соответствует произвольной строке символов, включая пустую строку. Например, my*.txt будет заменено на myday.txt, myweek.txt и mymonth.txt (если такие файлы существуют), а *.jpg соответствует всем файлам с расширением jpg в указанном каталоге

Таблица 5.4 *(окончание)*

Символ	Правила замены
?	Соответствует любому одиночному символу. Например, вместо шаблона file?.txt будут подставлены имена file1.txt и filex.txt, но не file10.txt
[...]	Соответствует любому символу из числа символов, указанных в скобках. Пары символов, разделенные знаком минуса, обозначают интервал; любой символ стоящий лексически между этими двумя символами, включая и символы, задающие интервал, соответствует шаблону. Если первым символом внутри скобок является ! или ^, то считается, что шаблону (в данной позиции) соответствуют все символы, не указанные в скобках

Шаблоны имен файлов очень часто применяются в командных строках, содержащих команду `ls`. Представьте себе, что вы хотите просмотреть информацию о содержимом каталога, в котором находится огромное количество разных файлов различных форматов, например, файлов с изображениями форматов GIF, JPEG, AVI и т. д. Чтобы получить только список файлов формата JPEG, вы можете использовать команду

```
[user]$ ls *.jpg
```

Если в каталоге имеется множество файлов, имена которых представлены четырехзначными номерами, то следующей командой можно вывести только список файлов с номерами от 0200 до 0499:

```
[user]$ ls -l 0[2-4]??.*
```

5.7.8. Удаление специальных символов

После того как все подстановки в командной строке сделаны, из нее еще удаляются все вхождения символов \, ' и ", которые служили для отмены специального значения других символов.

5.8. Shell как язык программирования

Как уже говорилось выше, для построения произвольных алгоритмов необходимо иметь операторы проверки условий. Оболочка bash поддерживает операторы выбора if then else и case, а также операторы организации циклов for, while, until, благодаря чему она превращается в мощный язык программирования.

5.8.1. Операторы *if* и *test* (или *[ ]*)

Конструкция условного оператора в слегка упрощенном виде выглядит так:

```
if list1 then list2 else list3 fi
```

где list1, list2 и list3 — это последовательности команд, разделенные запятыми и оканчивающиеся точкой с запятой или символом новой строки. Кроме того, эти последовательности могут быть заключены в фигурные скобки: {list}.

Оператор if проверяет значение, возвращаемое командами из list1. Если в этом списке несколько команд, то проверяется значение, возвращаемое последней командой списка. Если это значение равно 0, то будут выполняться команды из list2; если это значение не нулевое, будут выполнены команды из list3. Значение, возвращаемое таким составным оператором if, совпадает со значением, выдаваемым последней командой выполняемой последовательности.

Полный формат команды if имеет вид:

if list then list [elif list then list] ... [else list] fi

(здесь квадратные скобки означают только необязательность присутствия в операторе того, что в них содержится).

В качестве выражения, которое стоит сразу после if или elif, часто используется команда test, которая может обозначаться также квадратными скобками [ ]. Команда test выполняет вычисление некоторого выражения и возвращает значение 0, если выражение истинно, и 1 в противном случае. Выражение передается программе test как аргумент. Вместо того, чтобы писать

```
test expression
```

можно заключить выражение в квадратные скобки:

```
[ expression ] .
```

Заметьте, что test и [— это два имени одной и той же программы, а не какое-то магическое преобразование, выполняемое оболочкой bash (только синтаксис [требует, чтобы была поставлена закрывающая скобка). Заметьте также, что вместо test в конструкции if может быть использована любая программа.

В заключение приведем пример использования оператора if:

```
if [ -e textmode2.htm ] ; then
     ls textmode*
else
     pwd
fi
```

Об операторе test (или [...]) надо бы поговорить особо.

5.8.2. Оператор *test* и условные выражения

Условные выражения, используемые в операторе test, строятся на основе проверки файловых атрибутов, сравнения строк и обычных арифметических сравнений. Сложные выражения строятся из следующих унарных или бинарных операций ("элементарных кирпичиков").

П `-a file`

Верно, если файл с именем file существует.

❒ `-b file`

Верно, если file существует и является специальным файлом блочного устройства.

П `-c file`

Верно, если file существует и является специальным файлом символьного устройства.

П `-d file`

Верно, если `file` существует и является каталогом.

П `-e file`

Верно, если файл с именем `file` существует.

❒ `-f file`

Верно, если файл с именем `file` существует и является обычным файлом.

П `-g file`

Верно, если файл с именем file существует и для него установлен бит смены группы.

П `-h file` **ИЛИ** `-L file`

Верно, если файл с именем file существует и является символической ссылкой.

П `-k file`

Верно, если файл с именем `file` существует и для него установлен "sticky bit".

П `-p file`

Верно, если файл с именем file существует и является именованным каналом (FIFO).

П `-r file`

Верно, если файл с именем file существует и для него установлено право на чтение.

П `-s file`

Верно, если файл с именем `file` существует и его размер больше нуля.

- `-t fd`

  Верно, если дескриптор файла fd открыт и указывает на терминал.

- `-u file`

  Верно, если файл с именем file существует и для него установлен бит смены пользователя.

- `-w file`

  Верно, если файл с именем file существует и для него установлено право на запись.

- `-x file`

  Верно, если файл с именем `file` существует и является исполняемым.

- `-O file`

  Верно, если файл с именем file существует и его владельцем является пользователь, на которого указывает эффективный идентификатор пользователя.

- `-G file`

  Верно, если файл с именем file существует и принадлежит группе, определяемой эффективным идентификатором группы.

- `-S file`

  Верно, если файл с именем `file` существует и является сокетом.

- `-N file`

  Верно, если файл с именем file существует и изменялся с тех пор, как был последний раз прочитан.

- `file1 -nt file2`

  Верно, если файл file1 имеет более позднее время модификации, чем file2.

- `file1 -ot file2`

  Верно, если файл file1 старше, чем file2.

- `file1 -ef file2`

  Верно, если файлы file1 и file2 имеют одинаковые номера устройств и индексных дескрипторов (inode).

- `-o optname`

  Верно, если задействована опция оболочки optname. Пояснения см. на странице man bash.

- `-z string`

  Верно, если длина строки равна нулю.

П -n string

Верно, если длина строки не равна нулю.

П string1 == string2

Верно, если строки совпадают. Вместо == может использоваться =.

П stringl !== string2

Верно, если строки не совпадают.

П stringl < string2

Верно, если строка stringl лексикографически предшествует строке string2 (для текущей локали).

П stringl > string2

Верно, если строка stringl лексикографически стоит после строки string2 (для текущей локали).

П arg1 OP arc2

Здесь OP — это одна из операций арифметического сравнения: -eq (равно), -ne (не равно), -lt (меньше чем), -le (меньше или равно), -gt (больше), -ge (больше или равно). В качестве аргументов могут использоваться положительные или отрицательные целые числа.

Из этих элементарных условных выражений можно строить сколь угодно сложные с помощью обычных логических операций ОТРИЦАНИЯ, И и ИЛИ.

П !(expression)

Булевский оператор отрицания.

П expression -a expression2

Булевский оператор AND (И). Верен, если верны оба выражения.

П expression -o expression2

Булевский оператор OR (ИЛИ). Верен, если верно любое из двух выражений.

Такие же условные выражения используются и в операторах while и until, которые мы рассмотрим чуть ниже.

5.8.3. Оператор *case*

Формат оператора case таков:

case word in [[(] pattern [| pattern] ...) list ;;] ... esac

Команда case вначале производит раскрытие слова word и пытается сопоставить результат с каждым из образцов pattern поочередно. После нахождения первого совпадения дальнейшие проверки не производятся, выполня-

ется список команд, стоящий после того образца, с которым обнаружено совпадение. Значение, возвращаемое оператором, равно 0, если совпадений с образцами не обнаружено. В противном случае возвращается значение, выдаваемое последней командой из соответствующего списка.

Следующий пример использования оператора `case` заимствован из системного скрипта /etc/rc.d/rc.sysinit.

```
case "$UTC" in
   yes|true)
    CLOCKFLAGS="$CLOCKFLAGS -u";
    CLOCKDEF="$CLOCKDEF (utc)";
   ;;
   no|false)
    CLOCKFLAGS="$CLOCKFLAGS --localtime";
    CLOCKDEF="$CLOCKDEF (localtime)";
   ;;
esac
```

Если переменная принимает значение yes или true, то будет выполнена первая пара команд, а если ее значение равно по или false — вторая пара.

5.8.4. Оператор *select*

Оператор select позволяет организовать интерактивное взаимодействие с пользователем. Он имеет следующий формат:

```
select name [ in word; ] do list ; done
```

Вначале из шаблона word формируется список слов, соответствующих шаблону. Этот набор слов выводится в стандартный поток ошибок, причем каждое слово сопровождается порядковым номером. Если шаблон word пропущен, таким же образом выводятся позиционные параметры. После этого вьщается стандартное приглашение `PS3`, и оболочка ожидает ввода строки на стандартном вводе. Если введенная строка содержит число, соответствующее одному из отображенных слов, то переменной name присваивается значение, равное этому слову. Если введена пустая строка, то номера и соответствующие слова выводятся заново. Если введено любое другое значение, переменной name присваивается нулевое значение. Введенная пользователем строка запоминается в переменой REPLY. Список команд `list` выполняется с выбранным значением переменной name.

Вот небольшой скрипт:

```
#!/bin/sh
echo "Какую ОС Вы предпочитаете?"
```

```
select var in "Linux" "Gnu Hurd" "Free BSD" "Other"; do
        break
done
echo "Вы бы выбрали $var"
```

Если сохранить этот текст в файле, сделать файл исполняемым и запустить, на экран будет выдан следующий запрос:

```
Какую ОС Вы предпочитаете?
1) Linux
2} Gnu Hurd
3) Free BSD
4) Other
#?
```

Нажмите любую из 4 предложенных цифр (1, 2, 3, 4). Если вы, например, введете 1, то увидите собщение:

```
Вы бы выбрали Linux
```

5.8.5. Оператор *for*

Оператор for работает немного не так, как в обычных языках программирования. Вместо того, чтобы организовывать увеличение или уменьшение на единицу значения некоторой переменной при каждом проходе цикла, он при каждом проходе цикла присваивает переменной очередное значение из заданного списка слов. В целом конструкция выглядит примерно так:

```
for name in words do list done
```

Правила построения списков команд (list) такие же, как и в операторе if.

Пример.

Следующий скрипт создает файлы foo_1, foo_2 и foo_3:

```
for a in 1 2 3 ; do
      touch foo_$a
done
```

В общем случае оператор for имеет формат:

```
for name [ in word; ] do list ; done
```

Вначале производится раскрытие слова word в соответствии с правилами раскрытия выражений, приведенными выше. Затем переменной name поочередно присваиваются полученные значения, и каждый раз выполняется список команд list. Если in word пропущено, то список команд list выполняется один раз для каждого позиционного параметра, который задан.

В Linux имеется команда seq, которая воспринимает в качестве аргументов два числа и выдает последовательность всех чисел, расположенных между за-

данными. С помощью этой команды можно заставить for в bash работать точно так же, как аналогичный оператор работает в обычных языках программирования. Для этого достаточно записать цикл for следующим образом:

```
for a in $( seq 1 10 ) ; do
     cat  file_$a
done
```

Эта команда выводит на экран содержимое десяти файлов: "file_1",... ..., "file_10".

5.8.6. Операторы *while* и *until*

Оператор while работает подобно if, только выполнение операторов из списка list2 циклически продолжается до тех пор, пока верно условие, и прерывается, если условие не верно. Конструкция выглядит следующим образом:

```
while list1 do list2 done
```

Пример:

```
while [ -d mydirectory ] ; do
     ls -1 mydirectory >> logfile
     echo -- SEPARATOR — >> logfile
     sleep 60
done
```

Такая программа будет протоколировать содержание каталога mydirectory ежеминутно до тех пор, пока каталог существует.

Оператор until **аналогичен оператору** while:

until listl do list2 done.

Отличие заключается в том, что результат, возвращаемый при выполнении списка операторов list1, берется с отрицанием: list2 выполняется в том случае, если последняя команда в списке list1 возвращает ненулевой статус выхода.

5.8.7. Функции

Синтаксис

Оболочка bash позволяет пользователю создавать собственные функции. Функции ведут себя и используются точно так же, как обычные команды оболочки, т. е. мы можем сами создавать новые команды. Функции конструируются следующим образом:

```
function name () { list }
```

Причем слово function не обязательно, name определяет имя функции, по которому к ней можно обращаться, а тело функции состоит из списка команд list, находящегося между (и }. Этот список команд выполняется каждый раз, когда имя name задано как имя вызываемой команды. Отметим, что функции могут задаваться рекурсивно, так что разрешено вызывать функцию, которую мы задаем, внутри нее самой.

Функции выполняются в контексте текущей оболочки: для интерпретации функции новый процесс не запускается (в отличие от выполнения скриптов оболочки).

Аргументы

Когда функция вызывается на выполнение, аргументы функции становятся *позиционными параметрами (positional parameters)* на время выполнения функции. Они именуются как $n, где n — номер аргумента, к которому мы хотим получить доступ. Нумерация аргументов начинается с 1, так что $1 — это первый аргумент. Мы можем также получить все аргументы сразу с помощью `$*`, и число аргументов с помощью $#. Позиционный параметр о не изменяется.

Если в теле функции встречается встроенная команда return, выполнение функции прерывается и управление передается команде, стоящей после вызова функции. Когда выполнение функции завершается, позиционным параметрам и специальному параметру # возвращаются те значения, которые они имели до начала выполнения функции.

Локальные переменные

Если мы хотим создать локальный параметр, можно использовать ключевое слово local. Синтаксис ее задания точно такой же, как и для обычных параметров, только определению предшествует ключевое слово local: `local name=value`.

Вот пример задания функции, реализующей упоминавшуюся выше команду `seq`:

```
seq()
  {
    local I=$1;
    while [ $2 != $I ] ; do
      {
        echo -n "$I ";
        I=$(( $I + 1 ))
      };
```

```
    done;
    echo $2
}
```

Обратите внимание на опцию -n оператора echo, она отменяет переход на новую строку. Хотя это и несущественно для тех целей, которые мы здесь имеем в виду, это может оказаться полезным для использования функции в других целях.

Функция вычисления факториала *fact*

Еще один пример:

```
fact()
  {
    if [ $1 = 0 ]; then
      echo 1;
    else
      {
        echo $(( $1 * $( fact $(( $1 - 1 )) ) ))
      };
    fi
  }
```

Это функция факториала, пример рекурсивной функции. Обратите внимание на арифметическое расширение и подстановку команд.

5.9. Скрипты оболочки и команда *source*

Скрипт оболочки - это просто файл, содержащий последовательность команд оболочки. Подобно функциям, скрипты можно выполнять как обычные команды. Синтаксис доступа к аргументам такой же, как и для функций.

В общем случае при запуске скрипта запускается новый процесс. Для того, чтобы выполнить скрипт внутри текущей сессии bash, необходимо использовать команду source, синонимом которой является просто точка ".". Скрипт оболочки служит просто аргументом этой команды. Ее формат:

```
source filename [arguments]
```

или

```
. filename [arguments]
```

Эта команда читает и выполняет команды из файла с именем filename в текущем окружении и возвращает статус, определяемый последней командой из файла filename. Если filename не содержит слэша, то пути, пере-

численные в переменной PATH, используются для поиска файла с именем filename. Этот файл не обязан быть исполняемым. Если в каталогах, перечисленных в PATH, нужный файл не найден, его поиск производится в текущем каталоге.

Если заданы аргументы, на время выполнения скрипта они становятся позиционными параметрами. Если аргументов нет, позиционные параметры не изменяются. Значение (статус), возвращаемое командой source, совпадает со значением, возвращаемым последней командой, выполненной в скрипте. Если ни одна команда не выполнялась, или файл filename не найден, то статус выхода равен 0.

5.10. Команда *sh*

Вы всегда можете запустить новый экземпляр оболочки bash, дав команду bash или sh. При этом можно заставить новый экземпляр оболочки выполнить какой-то скрипт, если передать имя скрипта в виде аргумента команды bash. Так, для выполнения скрипта myscript надо дать команду sh myscript.

Если вы заглянете в какой-нибудь файл, задающий скрипт (таких файлов в системе очень много), вы увидите, что первая строка в нем имеет вид: #!/bin/sh. Это означает, что когда мы запускаем скрипт на выполнение как обычную команду, программа /bin/sh будет выполнять ее для нас. Можно заменить эту строку ссылкой на любую программу, которая будет читать файл и исполнять соответствующие команды. Например, скрипты на языке Perl начинаются со строки вида #!/bin/perl.

Отметим также, что символ # служит для выделения в скриптах комментариев. Все, что стоит в текущей строке после этого символа и до символа конца строки, оболочка будет считать комментариями и игнорировать (то есть оболочка не рассматривает этот текст как команды). Если хотите убедиться в действии этого символа, введите в командной строке любую команду, поставив перед ней символ I, например, # ls, и вы увидите, что команда игнорируется оболочкой.

На этом мы завершим сокращенное описание оболочки bash. Конечно, за рамками этого описания остались многие важные вопросы, например, управление процессами, описания встроенных команд, история команд, описание библиотеки readline, сигналы и т. д. Часть этих вопросов будет отражена в последующих главах, а остальное вы должны искать в других руководствах или на странице man bash.

Глава 6

Программа Midnight Commander

6.1. Установка программы Midnight Commander

Хотя для управления файловой системой и вообще для работы с файлами можно использовать такие команды операционной системы, как pwd, `ls`, `cd`, `mv`, `mkdir`, rmdir, cp, `rm`, cat, more **И Т. Д., Гораздо удобнее делать большую** часть работы по обслуживанию файловой системы с помощью программы Midnight Commander, которая наглядно представляет все выполняемые действия, облегчая тем самым выполнение указанных операций.

Midnight Commander (или просто тс) — это программа, которая позволяет просмотреть структуру каталогов и выполнить основные операции по управлению файловой системой. Другими словами, это файловый менеджер. Если вы имеете опыт работы с Norton Commander (nc) в MS-DOS или с FAR в Windows, то вы легко сможете работать и с тс, поскольку даже основные комбинации "горячих" клавиш у них совпадают. В этом случае для того, чтобы работать с Midnight Commander, вам достаточно бегло просмотреть приводимый ниже материал. Для тех же, кто незнаком с пс или FAR (да есть ли такие?), рекомендую внимательно проработать эту главу, потому что Midnight Commander существенно облегчает работу с операционной системой.

Примечание

Приводимое в данной главе описание составлено применительно к версии 4.5.30 программы, хотя может использоваться и для других версий.

Примечание

Данное описание полностью применимо только в тех случаях, когда программа запущена с терминала. Когда работа производится через эмулятор терминала в графическом режиме, некоторые положения описания могут не соответствовать реакции программы, по-видимому, потому, что нажатия на клавиши вначале перехватываются графической оболочкой. Наиболее часто такое несоответствие будет встречаться там, где речь идет о "горячих" клавишах.

В большинстве дистрибутивов программа Midnight Commander не устанавливается автоматически при инсталляции системы. Но соответствующий

rpm-пакет, как правило, имеется на дистрибутивном диске, и установка Midnight Commander из rpm-пакета проходит без каких-либо сложностей (о том, как произвести установку ПО из rpm-пакета, *см. в разд. 10.2)*. А поскольку наличие этой программы существенно облегчит вашу дальнейшую жизнь, я настоятельно рекомендую вам установить ее сразу же, как только вы произвели установку ОС.

6.2. Внешний вид экрана Midnight Commander

Для того чтобы запустить Midnight Commander, надо набрать в командной строке оболочки tc и нажать клавишу <Enter>. Если программа не запустилась, надо найти, где расположен исполняемый файл с именем tc, воспользовавшись командой `find / - name tc`, после чего указать в командной строке полный путь, например, у меня это `/usr/bin/mc`. После запуска вы увидите голубой экран, очень напоминающий экран программы Norton Commander для MS-DOS или программы FAR Е. Рошаля, которая широко используется в DOS-окне под Windows.

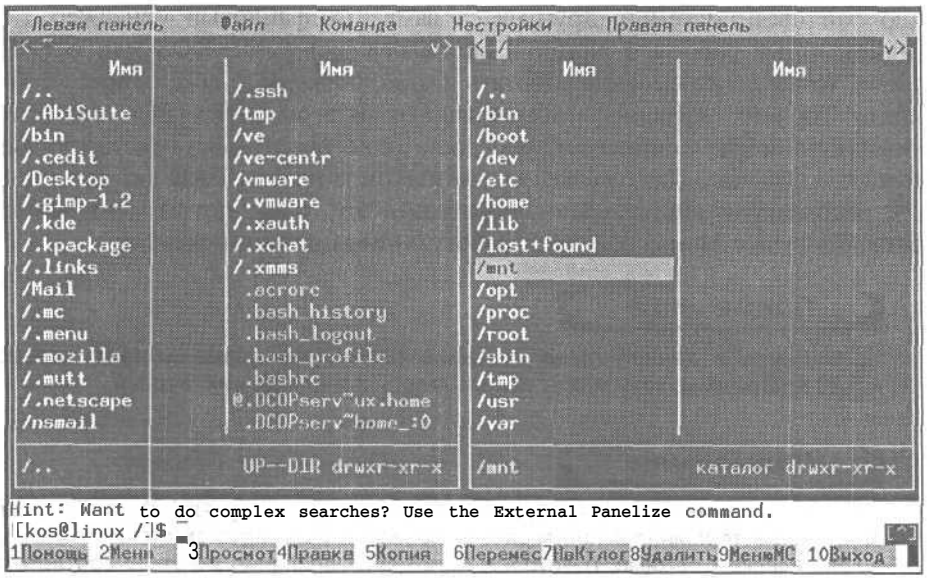


Рис. 6.1. Внешний вид экрана при работе с Midnight Commander

Почти все пространство экрана при работе с Midnight Commander занято двумя "панелями", отображающими списки файлов двух каталогов. Над панелями расположена строка меню, причем к выбору команд в этом меню

можно переключиться по клавише <F9> или с помощью мыши (если сразу после запуска тс вы не видите строки меню, не огорчайтесь — видна или нет строка меню, определяется настройками программы).

Самая нижняя строка представляет собой ряд экранных кнопок, каждая из которых ассоциирована с одной из функциональных клавиш <F1>—<F10>. Можно считать эту строку подсказкой по использованию функциональных клавиш, а можно и непосредственно запускать соответствующие команды, щелкая мышкой по экранной кнопке. Отображение строки с экранными кнопками можно отключить, если вы хотите сэкономить пространство экрана (об этом будет рассказано позже, когда будем говорить о настройках программы). Такая экономия оказывается оправданной по двум причинам. Во-первых, вы достаточно быстро запомните назначение этих 10 клавиш, и подсказка вам станет не нужна (а мышкой по этим кнопкам щелкать не всегда удобно). Во-вторых, если даже вы забыли, какая именно клавиша вам нужна для выполнения необходимого в данный момент действия, вы всегда можете воспользоваться меню **Файл** главного меню программы (только помните, что выход в главное меню осуществляется по клавише <F9>). Через меню **Файл** обеспечивается возможность выполнения любой операции из числа ассоциированных с функциональными клавишами, кроме <F1> и <F9>.

Вторая снизу строка на экране (на черном фоне) — это командная строка программы Midnight Commander (точнее — командная строка текущей оболочки shell), где можно ввести и выполнить любую команду системы. Выше нее (но под панелями) может отображаться поле "полезных советов" (tips), которое можно убрать, соответствующим образом отрегулировав настройки программы.

Каждая панель состоит из заголовка, списка файлов какого-либо каталога и строки мини-статуса (последняя может быть невидна, это тоже задается настройками программы). В заголовке панели указан полный путь к каталогу, содержимое которого отображается в панели, а также три экранных кнопки — <, v и >, которые используются для управления программой с помощью мыши (эти кнопки не работают, если вы запустили тс в эмуляторе терминала). В строке "мини-статуса" отображаются некоторые данные о том файле или каталоге, на который в данный момент указывает подсветка (например, размер файла и права доступа к нему).

Одна из панелей является текущей (активной), о чем свидетельствует подсветка имени каталога в заголовке панели и подсветка одной из ее строк. Соответственно, в той оболочке, из которой была запущена программа Midnight Commander, текущим является каталог, отображаемый в активной панели. В этом каталоге и выполняются почти все операции. Операции типа копирования (клавиша <F5>) или переноса файла (клавиша <F6>) используют каталог, отображаемый на второй панели, в качестве целевого каталога (в который осуществляется копирование или перенос).

В активной панели одна строка выделена подсветкой. Подсветку можно перемещать с помощью клавиш управления перемещением. Встроенная программа просмотра файлов, программа просмотра подсказки и программа просмотра каталогов используют один и тот же программный код для управления перемещением. Следовательно, для перемещения используются одни и те же комбинации клавиш (но в каждой подпрограмме имеются и комбинации, применяющиеся только в ней). В табл. 6.1 перечислены общие клавиши управления перемещением.

Таблица 6.1. *Общие клавиши управления перемещением*

Клавиша	Выполняемое действие
<t> или <Ctrl>+<P>	Перемещение на одну строку назад или вверх
<↓> или <Ctrl>+<N>	Перемещение на одну строку вперед
<PageUp> или <Alt>+<V>	Перемещение на одну страницу назад
<Page Down> или <Ctrl>+<V>	Перемещение на одну страницу вперед
<Home>	Перемещение к началу
<End>	Перемещение к концу

6.3. Получение помощи

При работе с программой Midnight Commander практически в любой момент можно обратиться к интерактивной подсказке, вызов которой осуществляется нажатием клавиши <F1>. Подсказка организована как гипертекст, т. е. в ее тексте встречаются гипертекстовые ссылки на другие ее разделы. Такие ссылки выделены голубым фоном.

Для перемещения в окне просмотра подсказки вы можете использовать клавиши перемещения курсора (стрелки) или мышь. Кроме общих комбинаций управления перемещением, приведенных в табл. 6.1, программа просмотра помощи воспринимает также комбинации, используемые в подпрограмме просмотра файлов (табл. 6.2).

Таблица 6.2. *Управление перемещением при просмотре файлов*

Клавиша	Выполняемое действие
<B> или <Ctrl>+<B> или <Ctrl>+<H> или <Backspace> или <Delete>	Перемещение на одну страницу назад
<Пробел>	Перемещение на одну страницу вперед

Таблица 6.2 *(окончание)*

Клавиша	Выполняемое действие
<U> (<D>)	Перемещение на половину страницы назад (или вперед)
<G>	Перемещение к началу или к концу

Кроме уже перечисленных комбинаций клавиш могут быть использованы еще некоторые, работающие только при просмотре подсказки (они приведены в табл. 6.3).

Таблица 6.3. *Управление перемещением при просмотре подсказки*

Клавиша	Выполняемое действие
<Tab>	Переход на следующую ссылку
<Alt>+<Tab>	Переход на предыдущую ссылку
<↓>	Переход на следующую ссылку или смещение текста на одну строку вверх
<t>	Переход на следующую ссылку или смещение текста на одну строку вниз
<→> или <Enter>	Переход по текущей ссылке
<←> или <L>	Возврат к ранее просмотренным разделам подсказки
<F1>	Помощь по использованию самой подсказки
<N>	Переход к следующему разделу помощи
<P>	Переход к предыдущему разделу помощи
<C>	Переход к оглавлению подсказки
<F10>, <Esc>	Выход из окна просмотра подсказки

Вы можете использовать клавишу пробела для перехода к следующей странице подсказки и клавишу <B> для перехода к предыдущей странице. Программа запоминает последовательность переходов по ссылкам и позволяет вернуться к ранее просмотренным разделам, воспользовавшись клавишей <L>.

Если включена поддержка мыши *(см. разд. 6.4),* то при просмотре подсказки можно пользоваться мышью. По щелчку левой клавиши мыши происходит переход по ссылке или перемещение по тексту подсказки. Щелчок правой кнопкой мыши используется для перехода к ранее просмотренным разделам.

6.4. Поддержка мыши

Программа M:.dnight Commander обеспечивает поддержку мыши. Это свойство реализуется, если запущен драйвер мыши дрт, независимо от того, работаете ли вы на консоли Linux или программа Midnight Commander запущена через терминал xterm (даже если вы используете соединение с удаленной машиной ИЗ xterm Через telnet, rlogin ИЛИ ssh).

Щелчком левой кнопки можно переместить подсветку на любой файл любой из панелей. Для того чтобы отметить (выделить) любой файл, достаточно щелкнуть правой кнопкой мыши на имени файла. Для снятия отметки используется та же правая кнопка.

Двойной щелчок левой кнопкой мыши на имени файла означает попытку запустить файл на исполнение (если это исполняемая программа); либо, если файл расширений содержит программу, ассоциированную с данным расширением, запускается эта программа и ей передается на обработку выбранный файл.

Щелчком мыши по функциональной кнопке можно также вызвать программу, ассоциированную с функциональными кнопками. Щелчок по команде верхнего меню вызывает выпадающее подменю.

Если щелкнуть мышью по верхней рамке панели, отображающей очень длинный список файлов, происходит перемещение списка на одну колонку назад. Щелчок по нижней рамке панели приводит, соответственно, к перемещению по списку на целую колонку вперед. Этот метод перемещения работает также при просмотре встроенной подсказки и просмотре окна Дерево каталогов.

Если Midnight Commander запущен с поддержкой мыши, вы можете производить копирование и вставку блоков текста, если будете удерживать нажатой клавишу <Shift>. Для этого нужно нажать клавишу <Shift> и, удерживая ее нажатой, выделить мышью нужный кусок текста, затем отпустить клавишу <Shift>, перевести курсор в нужное место, снова нажать клавишу <Shift> и щелкнуть правой кнопкой мыши. Отметим, что это свойство не работает в окне эмулятора терминала.

6.5. Управление панелями

Панели программы Midnight Commander чаще всего отображают содержимое каталогов файловой системы (поэтому называются иногда панелями каталогов). Однако на панель может быть выведена и некоторая другая информация. В настоящем разделе будет рассказано, как изменить вид панели или способ представления информации на панели.

6.5.1. Форматы отображения списка файлов

Вид панелей, в которых отображаются списки файлов и подкаталогов, может быть изменен через команды выпадающих меню левой и правой панелей **Левая панель** (Left) и **Правая панель** (Right) главного меню. Если вы хотите изменить формат представления списка файлов в панели, вы можете воспользоваться командой **Формат списка** соответствующей (левой или правой) панели. Имеется возможность выбрать один из 4-х вариантов представления списка файлов: **Стандартный** (Full), **Укороченный** (Brief), **Расширенный** (Long) и **Определяемый пользователем** (User).

- В "стандартном" формате отображаются имя файла, его размер и время последней модификации.
- В "укороченном" формате отображаются только имена файлов, за счет чего на панели умещаются две колонки (и видно вдвое больше имен).
- В "расширенном" формате содержимое каталога представляется так, как это делает команда `ls -l`. В этом формате панель занимает весь экран.
- Если вы выберете формат "определяемый пользователем", вы должны будете задать структуру отображаемой информации.

При задании структуры вначале указывается размер панели: "half (половина экрана) или "full" (весь экран). После размера панели можно указать, что на панели должно быть две колонки. Это делается добавлением цифры 2 в строку задания формата. Далее надо перечислить имена полей с необязательным параметром ширины поля. В качестве имен полей могут использоваться следующие слова:

- name — отображать имя файла;
- `size` — отображать размер файла;
- bsize — отображать размер в альтернативной форме, при которой выводятся размеры файлов, а для подкаталогов выводится только надпись `SUB-DIR` или UP-DIR;
- type — отображать односимвольное поле типа. Этот символ может принимать значения из следующего подмножества символов, выводимых командой `ls` с параметром `-F`:
  - * (asterisk) — для исполняемых файлов;
  - / (slash) — для каталогов;
  - @ (at-sign) — для ссылок (links);
  - = (знак равенства) — для сокетов (sockets);
  - - (дефис) — для байт-ориентированных устройств;
  - + (плюс) — для блок-ориентированных устройств;

- | (pipe) — для файлов типа FIFO;
- ~ (тильде.) — для символических ссылок на каталоги;
- ! (восклицательный знак) — для оборванных (stalled) символических ссылок (ссылок, указывающих на отсутствующий файл);

❒ mtime — время последней модификации файла;

❒ atime — время последнего обращения к файлу;

О ctime — время создания файла;

❒ perm — строка, показывающая текущие права доступа к файлу;

❒ mode — восьмеричное представление текущих прав доступа к файлу;

❒ nlink — число ссылок на данный файл;

П ngid — идентификатор группы (GID) в цифровой форме;

❒ nuid — идентификатор пользователя (UID) в цифровой форме;

П owner — владелец файла;

П group — группа, имеющая права на файл;

❒ inode — номер inode-файла.

Вы также можете использовать следующие имена полей для организации вывода информации на дисплей:

П space — вставить пробел при выводе на дисплей;

П mark — вставить звездочку (asterisk), если файл помечен, пробел — если не помечен;

П | — вставить вертикальную линию при выводе на дисплей.

Для того чтобы задать фиксированную ширину поля, нужно добавить двоеточие :, после которого указать число позиций, которое отводится под это поле. Если после числа поставить символ +, то указанное число будет интерпретироваться как минимальная ширина поля, и, если экран позволяет, поле будет расширено.

Например, "стандартный" формат вывода задается строкой:

```
half type,name,|,size,|,mtime
```

а "расширенный" — строкой:

```
full perm, space, nlink, space, owner, space, group, space, size, space, mtime, space, name
```

А вот пример формата, определяемого пользователем:

```
half name,|,size:7,|,type,mode:3
```

Отображение списка файлов в любой из панелей может производиться в соответствии с одним из восьми порядков сортировки:

- ❒ по имени;
- ❒ по расширению;
- П по размеру файла;
- П по времени модификации;
- П по времени последнего обращения к файлу;
- П по номеру узла (mode);
- П без сортировки.

Порядок сортировки вы можете задать, выбрав в меню соответствующей панели команду **Порядок сортировки.** При этом появляется диалоговое окно (рис. 6.2), в котором кроме желаемого порядка сортировки можно указать, что сортировка производится в обратном порядке (поставив с помощью клавиши пробела отметку в скобках возле слова **Обратный** (Reverse)) и с учетом регистра символов.

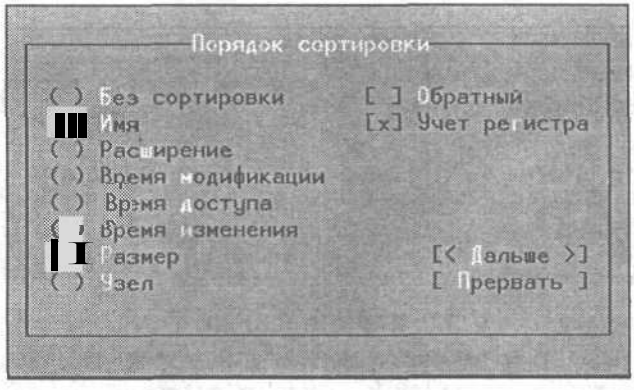


Рис. 6.2. Диалоговое окно задания порядка сортировки

По умолчанию подкаталоги отображаются в начале списка, но это можно изменить, проставив отметку возле опции **Смешивать файлы/каталоги** (Mix all files) команды **Конфигурация** меню **Настройки.**

Можно выводить на панель не все списки файлов данного каталога, а только соответствующие определенному шаблону. Команда **Фильтр** в меню любой панели позволяет задать шаблон, которому должны соответствовать имена файлов, отображаемых на панели (например, "*.tar.gz").

Имена подкаталогов и ссылки на подкаталоги отображаются всегда, независимо от шаблона.

В меню каждой из панелей имеется команда **Перечитать** (аналог которой в других программах обычно называется **Обновить).** Команда **Перечитать** (комбинация "горячих" клавиш <Ctrl>+<R>) обновляет список файлов, отображаемый на панели. Это бывает полезно в тех случаях, когда другие процессы создают или удаляют файлы. Если вы выполнили команду меню **Критерий панелизации** (перенаправление вывода команды на панель, о нем будет рассказано в *разд. 6.10*), и вследствие этого на панели находятся результаты работы какой-то программы, по команде **Перечитать** на панель вновь будет выведено содержимое каталога.

6.5.2. Другие режимы отображения

Помимо того, что может задаваться формат вывода на панель списка файлов, любую панель можно перевести в один из следующих режимов.

- **Режим "Информация".** В этом режиме (рис. 6.3) на панель выводится информация о подсвеченном в другой панели файле и о текущей файловой системе (тип, свободное пространство и число свободных индексных дескрипторов — mode).

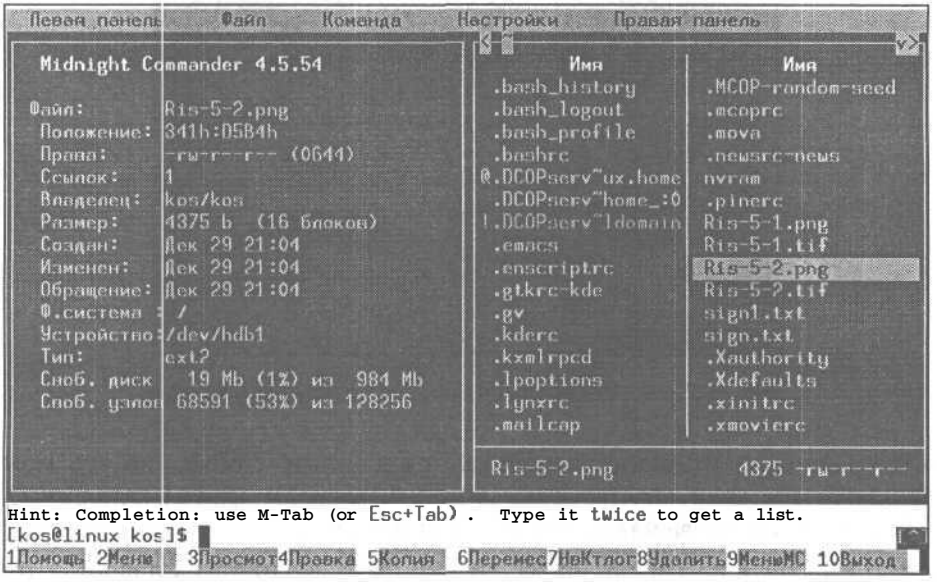


Рис. 6.3. Режим "Информация"

- **Режим "Дерево". В** режиме отображения дерева каталогов на одной из панелей выводится графическое изображение структуры дерева каталогов (см. пример на рис. 6.4). Этот режим подобен тому, который вы увидите,

выбрав команду **Дерево каталогов** из меню **Команды,** только в последнем случае изображение структуры каталогов выводится в отдельное окно.

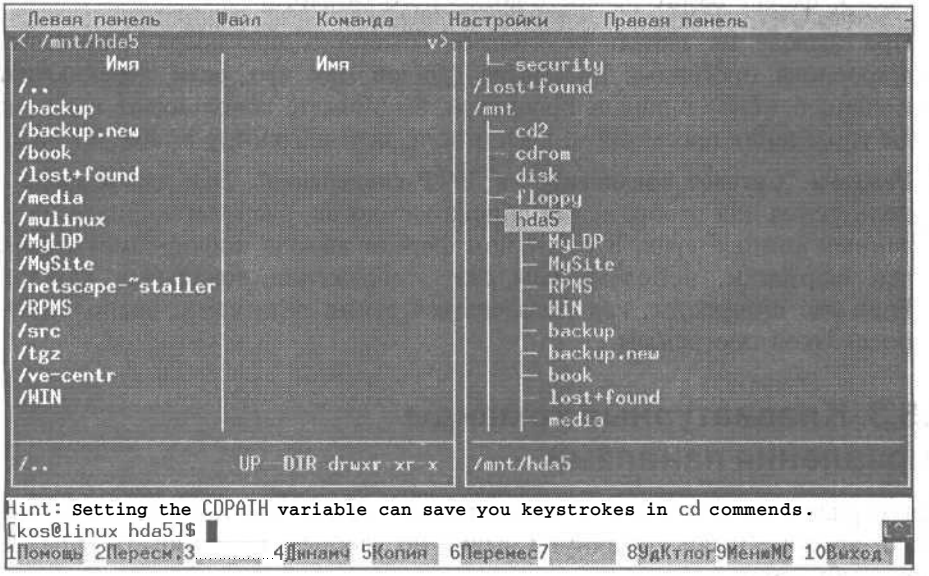


Рис. 6.4. Режим отображения дерева каталогов

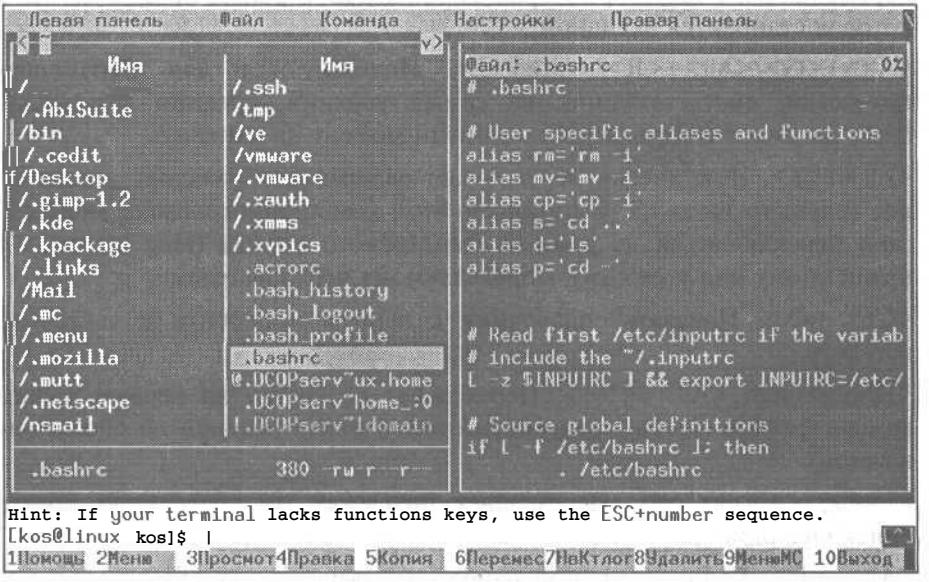


Рис. 6.5. Режим "Быстрый просмотр"

- **Режим "Быстрый просмотр"** ("Quick View"). В этом режиме панель переключается на отображение содержимого файла, подсвеченного в другой панели. Для примера на рис. 6.5 приведен вид экрана при быстром просмотре файла .bashrc из моего домашнего каталога.

  При выводе на панель результатов "быстрого просмотра" используется встроенная программа просмотра файлов, так что, если переключиться клавишей <Tab> в панель просмотра, вы сможете использовать все команды управления просмотром, в частности, перечисленные в табл. 6.1—6.2.

- **Режимы "Сетевое соединение" и "FTP-соединение".** Эти два режима используются для отображения списка каталогов, располагающихся на удаленных компьютерах. В остальном формат вывода информации аналогичен формагам, используемым для отображения локальных каталогов. Если вас интересует, как пользоваться этими режимами, воспользуйтесь подсказкой программы.

6.5.3. Клавиатурные команды управления панелями

Для управления режимами работы панели можно пользоваться командами меню, которые были упомянуты выше, но удобнее использовать управляющие комбинации клавиш.

- <Tab> или <Ctrl>+<I>. Сменить текущую (активную) панель. Подсветка перемещается с панели, которая была активной ранее, на другую панель, которая становится активной.
- <Alt>+<G>/<Alt>+<R>/<Alt>+<J>. Используются для перемещения подсветки, соответственно, на самый верхний, средний или нижний файл из числа отображаемых в данный момент на панели.
- <Alt>+<T>. Циклически переключает режимы отображения списка файлов текущего каталога. С помощью этой комбинации клавиш можно быстро переключаться из режима стандартного вывода (long listing) к сокращенному или к режиму, определяемому пользователем.
- <Ctrl>+<\>. Показать справочник каталогов и перейти к выбранному каталогу.
- П <Home> или <Alt>+<"<">. Перемещает подсветку на первую позицию списка файлов (здесь я вынужден отойти от соглашения об обозначении клавиш).
- П <End>, <Alt>+<">">. Перемещает подсветку на последнюю позицию списка файлов.
- П <Alt>+<O>. Если в активной панели подсвечено имя каталога, а на второй панели отображается список файлов, то вторая панель переходит в

режим отображения файлов подсвеченного каталога. Если на активной панели подсветка указывает на файл, то на второй панели отображается содержимое каталога, родительского по отношению к текущему.

- <Ctrl>+<PageUp>, <Ctrl>+<PageDown>. Только если tc запущен с консоли Linux: выполняется, соответственно, переход (`chdir`) к родительскому каталогу ("..") или к выделенному подсветкой каталогу.
- <Alt>+<Y>. Перемещение к предыдущему каталогу из истории перемещения по каталогам; эквивалентно нажатию мышкой на символ < в верхнем углу панели.
- <Alt>+<U>. Перемещение к следующему каталогу из истории перемещения по каталогам; эквивалентно нажатию мышкой на символ >.

6.6. Функциональные клавиши и меню *Файл*

До сих пор мы рассказывали только о внешнем виде экрана программы Midnight Commander и о том, как изменить этот вид. Теперь пора рассказать и том, как работать с файлами с помощью этой программы.

Наиболее часто выполняемые в Midnight Commander операции привязаны к функциональным клавишам <F1>—<F10>. Приведем сводку в виде табл. 6.4.

Таблица 6.4. *Функциональные клавиши*

Функциональная клавиша	Выполняемое действие
<F1>	Вызывает контекстно-зависимую подсказку
<F2>	Вызывает меню, создаваемое пользователем
<F3>	Просмотр файла, на который указывает подсветка в активной панели
<F4>	Вызов встроенного редактора для файла, на который указывает подсветка в активной панели
<F5>	Копирование файла или группы отмеченных файлов из каталога, отображаемого на активной панели, в каталог, отображаемый на второй панели. При копировании одного файла можно поменять его имя. Можно также указать имя каталога, куда будет производиться копирование (если надо скопировать в каталог, отличный от каталога, отображаемого на второй панели)
<F6>	Перенос файла или группы отмеченных файлов из каталога, отображаемого на активной панели, в каталог, отображаемый на второй панели. Как и при копировании, можно поменять имя файла или целевого каталога

Таблица 6.4 (окончание)

Функциональная клавиша	Выполняемое действие
<F7>	Создание подкаталога в каталоге, отображаемом на активной панели
<F8>	Удаление файла (подкаталога) или группы отмеченных файлов
<F9>	Вызов основного меню программы (отображаемого над панелями)
<F10>	Выход из программы

Перечисленные в этой таблице операции (команды) можно выполнять не только путем нажатия соответствующей функциональной клавиши, но и с помощью щелчка мыши по экранным кнопкам или используя соответствующие команды меню **Файл.**

Прежде чем выполнять какую-то из операций, указанных в табл. 6.4 или задаваемых командами меню, надо выбрать файл или группу файлов, которые будут объектами операции. Для выбора только одного файла достаточно переместить на него подсветку на активной панели (конечно, вначале надо перейти в соответствующий каталог). Если же вы хотите выполнить какую-то операцию сразу над группой файлов, эти файлы надо отметить. Чтобы отметить файл, на который указывает в данный момент подсветка, используйте клавишу <Insert> или комбинацию <Ctrl>+<T>. При этом имя файла в панели выводится другим цветом. Для снятия отметки с файла используются те же комбинации.

Выделить группу файлов для последующей обработки можно также с помощью команды **Отметить группу** меню **Файл.** Эта команда используется для отметки группы файлов по заданному шаблону. Midnight Commander выдаст строку ввода, з которой надо задать регулярное выражение, определяющее желаемую группу имен. Если включена опция **Образцы в стиле shell** *(см. разд. 6.11)*, регулярные выражение строятся по тем же правилам, которые действуют в оболочке shell *(см. гл. 5).* Если опция **Образцы в стиле shell** отключена, то пометка файлов производится по правилам обработки нормальных регулярных выражений (см. руководство man ed).

Если выражения начинаются или оканчиваются слэшем (/), то пометка будет ставиться на каталоги, а не на файлы.

"Горячей" клавишей для операции отметки группы файлов является клавиша <+> на цифровой клавиатуре.

Операция "Снять отметку" ("горячие" клавиши — <–> или <\> на цифровой клавиатуре) является обратной по отношению к операции отметки группы файлов и использует те же правила формирования шаблонов. Используется для снятия отметки с группы файлов.

Операция "Инвертировать отметку" (<*>) используется для того, чтобы снять отметки со всех помеченных файлов текущего каталога, одновременно отметив все файлы, которые не были помечены.

Если текущий каталог содержит много файлов (так что все они не умещаются на панели), то прежде чем отметить файл, его нужно еще отыскать. В таких случаях удобно пользоваться комбинациями клавиш <Ctrl>+<S> и <Alt>+<S>. После нажатия одной из этих комбинаций инициируется режим поиска имен файлов в текущем каталоге по первым символам имени. В этом режиме вводимые символы отображаются не в командной строке, а в строке поиска. Если режим **Показ мини-статуса** (Show mini-status) включен, эта строка отображается на месте строки мини-статуса. При этом в процессе ввода символов линия подсветка перемещается к следующему файлу, название которого начинается с введенной строки символов. Клавиши <Backspace> или <Del> могут использоваться для исправления ошибок. Если комбинация <Ctrl>+<S> нажата снова, осуществляется поиск следующего подходящего файла. Надо отметить, что если в текущем каталоге нет файлов с именами, начинающимися на вводимые символы, эти символы не отображаются в строке мини-статуса, что как раз и говорит вам, что вы здесь не найдете искомого файла.

После выбора и отметки файлов для обработки достаточно нажать одну из функциональных клавиш, чтобы выполнить нужную операцию с файлами, например, скопировать файлы, переместить или удалить. Особенно удобно в сравнении с работой из командной строки осуществлять с помощью Midnight Commander просмотр или редактирование файлов, поскольку в Midnight Commander имеются встроенные программы для этих целей. Но пока мы не будем рассматривать встроенный редактор, отложив его описание до главы, посвященной работе с текстовыми файлами.

Как уже говорилось, вовсе не обязательно использовать функциональные клавиши для вызова ассоциированных с ними команд. Любую из этих команд можно выполнить через меню **Файл.** Кроме команд, ассоциированных с функциональными клавишами, меню **Файл** содержит еще несколько команд (в скобках указаны соответствующие "горячие" клавиши).

- **Права доступа** (<Ctrl>+<X>,<C>). Позволяет изменить права доступа к выделенному или помеченным файлам.
- **Владелец/группа** (<Ctrl>+<X>,<O>). Позволяет выполнить команду `chown`.
- **Права (расширенные).** Позволяет изменить права доступа и владения файлом.
- **Жесткая ссылка** (<Ctrl>+<X>,<L>). Создает жесткую ссылку на текущий файл.

- **Символическая ссылка** (<Ctrl>+<X>,<S>). Создает символическую ссылку на текущий файл. О ссылках мы подробно говорили в *разд. 4.4.* Программа Midnight Commander указывает символические ссылки, выводя знак @ перед именем такой ссылки (кроме ссылок на подкаталоги, которые обозначаются знаком тильды ~). Если на экран выводится строка мини-статуса (опция **Показывать мини-статус** включена), то в ней отображается имя того файла, на который указывает ссылка.

а **Быстрая смена каталога** (<Alt>+<C>). Используйте эту команду меню, если вы знаете полный путь к каталогу, в который хотите перейти (который хотите сделать текущим).

- **Просмотр вывода команды** (<Alt>+<!>). По этой команде на экране появляется строка ввода, в которой вы можете ввести любую команду *с* параметрами (по умолчанию предлагается использовать в качестве параметра имя подсвеченного файла). Вывод этой команды будет отображаться на экране через встроенную программу просмотра.

Как видите, меню **Файл** содержит все наиболее употребительные команды, которые нужны нам для обычных операций обработки файлов текущего каталога. Естественно, что операции, используемые чаще всего, связаны с функциональными клавишами. Поэтому вернемся к описанию этих команд, чтобы дать некоторые дополнительные пояснения.

6.7. Маски файлов для операций копирования/переименования

При выполнении операций копирования и перемещения (или переименования) файлов вы имеете возможность изменить имена копируемых или перемещаемых файлов. Для этого вы должны задать маску для имен файлов-источников и маску для имен файлов, которые будут созданы (файлы-приемники). Обычно эта вторая маска представляет собой несколько символов замены (wildcards) в конце строки, определяющей место назначения создаваемых файлов. Задание масок осуществляется в строках ввода, отображаемых в окне, появляющемся после обращения к командам копирования/переноса (рис. 6.6).

Все файлы, удовлетворяющие маске источника, будут переименованы (скопированы или перемещены с новыми именами) в соответствии с маской файла-приемника. Если имеются помеченные файлы, то копируются (перемещаются) только помеченные файлы, удовлетворяющие заданной маске для файлов-источников.

Есть еще несколько опций, которые влияют на выполнение операций копирования/перемгщения файлов, и которые устанавливаются в том же окне

запроса, где задаются маски имен файлов, либо через команду меню **Настройки | Конфигурация.**

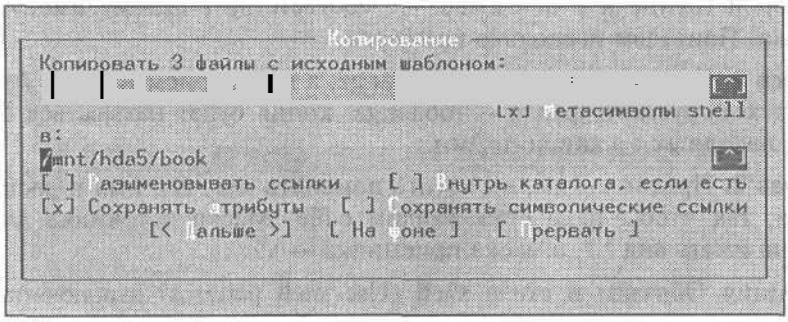


Рис. 6.6. Диалоговое окно для переименования файлов

Опция **Разыменовывать ссылки** (Follow links) определяет, будут ли при копировании жестких или символических ссылок в каталоге-приемнике (и рекурсивно в подкаталогах) создаваться такие же ссылки или будут копироваться файлы (и подкаталоги), на которые эти ссылки указывают.

Опция **Внутрь каталога, если есть** (Dive into subdirs) определяет, что делать, если в каталоге-приемнике уже существует подкаталог, имя которого совпадает с именем файла (каталога), который копируется (источника). По умолчанию (опция отключена) содержимое каталога-источника копируется в каталог-приемник. Если опция включена, то в каталоге-приемнике будет создан новый подкаталог с тем же именем, в который и будет осуществляться копирование.

Лучше показать это на примере. Пусть вы хотите скопировать содержимое каталога one в каталог /two/one, который уже существует. Обычно (опция отключена) тс будет просто копировать все файлы из one в /two/one. Если опцию включить, копирование файлов будет производиться в /two/one/one.

Опция **Сохранять атрибуты** (Preserve attributes) определяет, будут ли при копировании/перемещении сохранены атрибуты исходного файла: права доступа, временные метки и, если вы пользователь root, UID и GID исходного файла. Если опция отключена, атрибуты будут установлены в соответствии с текущим значением опции umask.

На процедуры копирования и перемещения файлов оказывает также влияние установка опции **Образцы в стиле shell** в меню **Настройки | Конфигурация.** Когда эта опция включена, вы можете использовать символы замены (wildcards) * и ? в маске источника. Они обрабатываются аналогично тому, как это делается в shell. В маске приемника разрешается использовать только * и последовательность \ + цифра. Первый символ * в маске приемника

соответствует первой группе символов замены в маске источника, второй символ * соответствует второй группе и т. д. Аналогично, символ замены \1 соответствует первой группе символов замены в маске источника, символ \2 — второй группе и т. д. Символ \0 соответствует целому имени файла-источника. Приведем несколько примеров.

- **Пример 1.** Если маска источника *.tar.gz, а маска приемника — /two/*.tgz, и имя копируемого файла — foo.tar.gz, копия будет называться foo.tgz и будет находиться в каталоге /two.
- **Пример** 2. Предположим, вы хотите поменять местами имя и расширение файла, так чтобы file.c стал файлом c.file. Маска источника для этого должна иметь вид *.*, а маска приемника — \2.\1.

Когда опция **Образцы в стиле shell** (Use shell patterns) выключена, mc не осуществляет автоматической группировки. Для указания групп символов в маске источника, которые будут соответствовать символам замены в маске приемника, вы должны в этом случае использовать скобки \(...\). Этот способ более гибкий, но требует больше усилий при вводе. Снова приведем два примера

- **Пример 1.** Если маска источника имеет вид ^\(.*\)\.tar\.gz$, копирование производится в /two/*.tgz и копируется файл foo.tar.gz, то результатом будет файл /two/foo.tgz.
- **Пример** 2. Предположим, что вы хотите поменять местами имя файла и его расширение, так чтобы имена вида file.c приняли вид c.file.

  Маска источника для этого - ^\(.*\)\.\(.*\)$, а маска приемника - \2.\1.

При выполнении операций копирования/перемещения вы можете также преобразовать регистр символов в именах файлов. Если вы используете \u или \l в маске приемника, то следующий символ имени будет образован в верхнем (заглавные символы) или нижнем (строчные) регистре соответственно.

Если использовать в маске приемника \U или \L, то к соответствующему регистру будут преобразованы все последующие символы, вплоть до следующего вхождения \L или \U, или же до конца имени файла.

Применение \u и \l обеспечивает более широкие возможности, чем \U и \L.

Например, если маска источника есть * (опция **Образцы в стиле shell** включена) или ^\(.*\)$ (опция **Образцы в стиле shell** выключена), а маска приемника есть \L\u*, имена файлов будут преобразованы таким образом, что первые буквы имени будут заглавными, а все остальные — строчными.

Символ \ в масках используется для отмены специальной интерпретации отдельных символов. Например, \\ означает просто обратный слэш (как литерал) и * означает просто звездочку (asterisk).

6.8. Сообщения Midnight Commander при выполнении операций копирования и перемещения файлов

Когда вы выполняете операции копирования, перемещения или удаления файлов, Midnight Commander отображает на экране диалоговое окно, в котором показано, какой(ие) файл(ы) в данный момент обрабатывается и как идет процесс обработки. Для отображения процесса обработки на экран выводится до трех диаграмм-полосок (progress bars). Первая (file bar) показывает, какая часть текущего файла обработана (например, скопирована). Вторая (count bar) показывает, какая доля помеченных файлов обработана на текущий момент. Третья (bytes bar) показывает долю выполненных работ в процентах от суммарного объема (в битах) помеченных файлов. Если отключена опция **Детали операций** (см. команду меню **Настройки | Конфигурация),** две последних диаграммы не отображаются.

В нижней части этого диалогового окна имеются две кнопки. Нажатие на кнопку **Пропустить** приведет к тому, что будет пропущена обработка текущего файла. Нажатие на кнопку **Прервать** завершает выполнение заданной операции, все оставшиеся файлы будут пропущены.

В процессе выполнения файловых операций вы можете увидеть окна еще трех видов.

- **Окно ошибок** информирует об ошибке и предлагает три варианта продолжения. Обычно вы выбираете либо вариант **Пропустить** для того, чтобы исключить из обработки файл, на котором споткнулась программа, либо **Прервать,** чтобы вообще отменить выполнение заданной операции. Третий вариант, **Повторить,** выбирается тогда, когда вам удалось устранить причину сбоя (например, воспользовавшись другим терминалом).
- **Окно подтверждения** перезаписи появляется тогда, когда вы пытаетесь перезаписать существующий файл, т. е. в каталоге, в который производится перемещение или копирование, уже есть файл с заданным именем В окне подтверждения отображаются время создания и размер файла-источника (переносимого или копируемого файла) и файла-приемника (который в случае перезаписи будет уничтожен).

  Ниже выведены два вопроса. На первый вопрос ("Переписать этот файл?") предлагается три варианта ответа:

  - согласиться (кнопка Да);
  - отказаться, т. е. пропустить перезапись текущего файла (кнопка **Нет);**
  - дописать содержимое файла-источника в конец файла-приемника (кнопка **Дописать в конец**).

Ответ на второй вопрос ("Переписать все файлы?") позволяет принять одно из 4-х возможных решений по всем выбранным для обработки файлам (чтобы окно запроса не появлялось каждый раз):

- Все — перезапишутся без дополнительных вопросов все выбранные файлы;
- **Устаревшие** — перезапишутся (затрутся) только файлы, которые созданы раньше файла-источника;
- **Ни одного** — не перезаписывать файлы (но если не существует файла-приемника, то копирование источника будет произведено);
- **Различающиеся по длине.**

Вы можете отказаться от выполнения операции, если нажмете кнопку **Прервать** в нижней части окна запроса. Выбор нужной экранной кнопки производится клавишами со стрелками или клавишей табуляции.

- **Окно запроса** на рекурсивное удаление появляется в том случае, когда вы пытаетесь удалить непустой каталог. По кнопке Да каталог будет удален вместе со всеми файлами, Нет означает отказ от удаления каталога, Все надо выбирать, если вы отметили группу подкаталогов для удаления и уверены в своем выборе, **Ни одного** — чтобы пропустить все непустые каталоги из числа помеченных, **Прервать** означает отказ от выполнения операции удаления. Выбирайте Да или Все только в том случае, когда вы действительно уверены, что хотите удалить каталог со всеми вложенными подкаталогами.

  Если вы пометили для обработки группу файлов, то после выполнения операции будет снята отметка только с тех файлов, которые успешно обработаны. Пропущенные файлы останутся помеченными.

6.9. Командная строка оболочки

Как было сказано при описании внешнего вида экрана Midnight Commander, в нижней части экрана всегда присутствует командная строка оболочки.

Для того чтобы во время работы с Midnight Commander запустить любую команду операционной системы, вы должны либо набрать имя соответствующей программы в командной строке, либо выбрать его в одной из панелей (переместив подсветку на имя файла программы), а затем нажать клавишу <Enter>. Если вы нажимаете клавишу <Enter> в тот момент, когда подсветка указывает на имя файла, не являющегося исполняемым, Midnight Commander сравнивает расширение выбранного файла с расширениями, прописанными в "файле расширений" ~/mc.ext. Если в файле расширений найдется подраздел, задающий процедуры обработки файлов с данным расширением, то обработка файла производится в соответствии с заданными в этом подразделе командами. Перед обработкой выполняются простые макроподстановки.

Зачастую ввод команд требует ввода большого числа символов (особенно с учетом того, что надо указать параметры команды и, в том числе, полные, с

указанием путей, имена обрабатываемых файлов). Для облегчения ввода в Midnight Commander существуют несколько клавиатурных команд, которые позволяют сократить число нажатий на клавиши во время ввода и редактирования команд в командной строке.

- <Alt>+<Enter>. Копирует подсвеченное имя файла или каталога в командную строку.
- <Ctrl>+<Enter>. То же самое, что <Alt>+<Enter>, но работает только в консоли.
- <Alt>+<Tab>. Пытается выполнить операцию "Завершение ввода" (completion) имени файла, названия команды, переменной, имени пользователя или имени хоста (в зависимости от того, что вы начали набирать и какой элемент команды вводите), т. е. пытается угадать еще не набранные вами символы исходя из списка файлов текущего каталога, списка команд и т. д.
- <Ctrl>+<X>, <T>. Копирует в командную строку имена помеченных файлов (или подсвеченное имя, если нет помеченных) из активной панели.
- <Ctrl>+<X>, <Ctrl>+<T>. Копирует в командную строку имена помеченных файлов из пассивной панели.
- <Ctrl>+<X>,<P>. Копирует в командную строку имя текущего каталога (то есть каталога, отображаемого в активной панели).
- <Ctrl>+<X>, <Ctrl>+<P>. Копирует в командную строку имя каталога, отображаемого в пассивной панели.
- <Ctrl>+<Q>. Вставляет символы, которые каким-то образом интерпретируются самой программой Midnight Commander (например, символ +).
- <Alt>+<P>. Вызывает перемещение на команду назад по списку ранее запускавшихся команд (истории команд).
- <Alt>+<N>. Перемещает на одну команду вперед в истории команд.
- <Alt>+<H>. Выводит историю текущей строки ввода (для командной строки — историю команд).

Строки ввода — это не только командная строка оболочки shell, но и строки ввода в диалоговых окнах различных подпрограмм. Во всех случаях, когда на экране появляется строка ввода, можно пользоваться управляющими комбинациями клавиш, перечисленными в табл. 6.5.

Таблица 6.5. *Команды управления строкой ввода*

Комбинация клавиш	Выполняемое действие
<Ctrl>+<A>	Перемещает курсор в началу строки
<Ctrl>+<E>	Перемещает курсор в конец строки

Таблица 6.5 *(окончание)*

Комбинация клавиш	Выполняемое действие
<Ctrl>+<B> или <←>	Перемещает курсор на одну позицию влево
<Ctrl>+<F> или <→>	Перемещает курсор на одну позицию вправо
<Alt>+<F>	Перемещает курсор на одно слово вперед
<Alt>+<B>	Перемещает курсор на одно слово назад
<Ctrl>+<H> или <Backspace>	Удаляет символ, предшествующий курсору
<Ctrl>+<D> или <Delete>	Удаляет символ в позиции курсора
<Ctrl>+<@>	Устанавливает метку для того, чтобы вырезать (скопировать в буфер) часть текста
<Ctrl>+<W>	Копирует текст, расположенный между курсором и меткой, в буфер, удаляя текст из строки ввода
<Alt>+<W>	Копирует текст, расположенный между курсором и меткой, в буфер (без удаления из строки ввода)
<Ctrl>+<Y>	Вставляет содержимое буфера в строку ввода перед позицией курсора
<Ctrl>+<K>	Удаляет текст от курсора до конца строки
<Alt>+<P> и <Alt>+<N>	Эти комбинации используются для перемещения по истории команд. <Alt>+<P> перемещает к предыдущей команде, <Alt>+<N> — к следующей
<Ctrl>+<Alt>+<H> или <Alt>+<Backspace>	Удаляет предшествующее слово
<Alt>+<Tab>	Пытается выполнить завершение ввода (completion) имени файла, команды, переменной, имени пользователя или имени хоста

6.10. Меню *Команды*

Выпадающее подменю **Команды** главного меню позволяет выполнить еще ряд операций по управлению файловой системой, а также выполнить некоторые команды, изменяющие вид панелей Midnight Commander и отображаемую в панели информацию.

При обращении к команде меню **Дерево каталогов** выводится окно, отображающее структуру каталогов файловой системы.

Дерево каталогов может быть вызвано двумя способами: через команду **Дерево каталогов** из меню **Команды** и команду **Дерево** из меню правой или левой панелей.

Чтобы избавиться от долгих задержек во время создания дерева каталогов, Midnight Commander создает дерево путем просмотра только небольшого подмножества всех каталогов. Если каталог, который вам нужен, не отображен, перейдите в его родительский каталог и нажмите комбинацию клавиш <Ctrl>+<R> или клавишу <F2>. Если каталог не содержит вложенных подкаталогов, ничего не произойдет. В противном случае развернется еще один уровень дерева подкаталогов.

Существует два режима отображения дерева каталогов. В статическом режиме перемещения для выбора каталога (то есть перемещения подсветки на имя другого каталога) используются только клавиши <t> и <↓>. Показываются все известные программе на данный момент подкаталоги. В динамическом режиме клавиши <T> и <↓> используются для перехода на соседний каталог того же уровня. Для перехода в родительский каталог используется клавиша <←>, а по клавише <→> происходит переход к потомкам текущего каталога, т. е. на один уровень ниже. При этом отображаются только вышележащие каталоги (включая родительский и выше), соседние каталоги того же уровня и непосредственные потомки. Вид дерева каталогов динамически изменяется после каждого перемещения по дереву.

Для управления просмотром дерева каталогов могут использоваться следующие комбинации клавиш.

- Работают все клавиши управления перемещением (см. табл. 6.1).
- <Enter>. В окне просмотра дерева каталогов нажатие этой клавиши вызывает выход из режима просмотра и отображение списка файлов выбранного каталога на активной панели. При отображении дерева каталогов на одной из панелей при нажатии клавиши <Enter> соответствующий каталог отображается на второй панели, а на текущей панели остается дерево.
- <Ctrl>+<R> или <F2>. Перечитать содержимое каталога. Используется в тех случаях, когда дерево каталогов не соответствует реальной структуре: некоторые подкаталоги не показаны или показаны более не существующие.
- <F3>. Удалить текущий каталог из дерева. Эта команда используется *только* для удаления ошибочно отображаемых ветвей дерева. Если вы попытаетесь удалить существующий каталог, будет выдано сообщение об ошибке. Именно поэтому в строке, поясняющей назначение функциональных клавиш, клавиша <F2> обозначена словом **Забыть.**
- <F4> (Static/Dynamic). Переключение между статическим (применяемым по умолчанию) и динамическим режимами перемещения по дереву.
- <F5>. Копировать подкаталог (появляется строка ввода, в которой надо указать, куда копировать).
- <F6>. Переместить подкаталог.
- <F7>. Создать подкаталог в текущем каталоге.

П <F8>. Удалить подсвеченный каталог из файловой системы.

❒ <Ctrl>+<S> или <Alt>+<S>. Найти следующий каталог, соответствующий заданному шаблону поиска. Если такого каталога не существует, происходит просто смещение подсветки на одну строку вниз.

П <Ctrl>+<H> или <Backspace>. Удаляет последний символ в строке (шаблоне) поиска.

П <Любой другой символX Этот символ добавляется в шаблон поиска и производится перемещение на имя следующего каталога, удовлетворяющее шаблону. В режиме просмотра дерева каталогов вначале необходимо активизировать режим поиска клавишами <Ctrl>+<S>. Шаблон поиска будет отображаться в строке мини-статуса.

П Следующие действия возможны только в окне просмотра дерева каталогов и не поддерживаются при просмотре дерева в одной из панелей.

П <F1> (Help). Вызов подсказки с отображением раздела помощи об окне дерева каталогов.

П <Esc> или <F10>. Выход из окна дерева каталогов. Смены текущего каталога не происходит.

В окне просмогра дерева каталогов поддерживается мышь. Двойной щелчок аналогичен нажатию клавиши <Enter>.

Команда **Поиск файла** (горячие клавиши <Meta>+<?> или <Esc>, <?>) выпадающего меню **Команда** позволяет вам найти на диске файл с заданным именем. После выбора этой команды меню вначале запрашивается имя искомого файла и имя каталога, с которого необходимо начинать поиск. Нажав кнопку **Дерево,** вы можете выбрать начальный каталог поиска из дерева каталогов. В поле **Содержание** (Contents) можно задать регулярное выражение по правилам команды egrep. Это значит, что перед символами, имеющими специальное значение для egrep, необходимо вставить символ \, например, если вам нужно найти строку strcmp (, вы должны указать шаблон поиска в виде strcmp \((без двойных кавычек). Для того чтобы начать поиск, нажмите кнопку **Дальше.** Во время поиска его можно приостановить кнопкой **Остановить** и продолжить по кнопке **Продолжить.**

Список найденных файлов можно просматривать, перемещаясь с помощью клавиш <↑> и <↓>. Кнопка **Перейти** используется для перехода в каталог, в котором находится подсвеченный файл. Кнопка **Повтор** служит для задания параметров нового поиска.

Кнопка **Выход** служит для выхода из режима поиска.

Нажатие на кнопку **Панелизация** приведет к тому, что результаты поиска будут отображены на текущую активную панель, так что вы можете производить с выбранными файлами еще какие-то действия (просматривать, копировать,

перемешать, удалять и т. д.). После вывода на панель можно нажать комбинацию клавиш <Ctrl>+<R> для возврата к обычному списку файлов.

Кроме того, имеется возможность задать список каталогов, которые команда **Поиск файла** будет пропускать в ходе поиска (например, вы знаете, что искомого файла нет на CD-ROM или не хотите искать в каталогах, подключенных через NFS по очень медленному каналу). Каталоги, которые надо пропустить, должны быть указаны в переменной `find_ignore_dirs` в секции Misc вашего файла ~/.mc/ini.

Имена каталогов разделяются двоеточиями следующим образом:

```
[Misc]
find_ignore_dirs=/cdrom:/nfs/wuarchive:/afs
```

Вы можете использовать перенаправление вывода на панель (см. ниже **Критерий панелизации)** для выполнения некоторых усложненных последовательностей действий, в то время как **Поиск файла** позволяет выполнять только простые запросы.

Команда **Переставить панели** (<Ctrl>+<U>) меняет местами содержимое правой и левой панелей.

По команде **Отключить панели** (<Ctrl>+<O>) показывается вывод последней из выполнявшихся команд shell. Эта команда работает только через xterm и на консоли Linux.

По команде **Сравнить каталоги** (<Ctrl>+<X>, <D>) сравнивается содержимое каталогов, отображаемых на левой и правой панелях.

Существует три метода сравнения. При быстром методе сравниваются только размер и дата создания файлов с одинаковыми именами. В результате в обоих каталогах будут подсвечены файлы, отсутствующие во втором каталоге, или более новые версии соответствующих файлов. После этого вы можете воспользоваться командой **Копировать** (<F5>) для того, чтобы сделать содержимое каталогов одинаковым.

При побайтном методе сравнивается содержимое файлов (побайтно). Этот метод недоступен, если машина не поддерживает системный вызов mmap(2). При сравнении по размеру сравниваются только размеры соответствующих файлов, а дата создания не проверяется.

Команда меню **Критерий панелизации** (которую правильнее было бы назвать "Перенаправление вывода на панель") позволяет вам выполнить внешнюю программу, сделав ее вывод содержимым текущей активной панели (характерный пример — панелизация вывода команды `find`). Например, если вы хотите выполнить какое-то действие над всеми символическими ссылками текущего каталога, вы можете использовать команду **Критерий панелизации** для запуска следующей команды:

```
[user]$ find . -type l -print
```

После выполнения этой команды в текущей панели будет отображено не содержимое соответствующего каталога, а только все символические ссылки, в нем расположенные.

Если вы захотите, то можете сохранить часто используемые команды панелизации под отдельными информативными именами, чтобы иметь возможность их быстро вызвать по этим именам. Для этого нужно набрать команду в строке ввода (строка "Команда") и нажать кнопку **Добавить.** После этого вам потребуется ввести имя, по которому вы будете вызывать команду. В следующий раз вам достаточно будет выбрать нужное имя из списка, а не вводить всю команду заново.

Команда меню **История команд** выводит окно со списком ранее выполнявшихся команд. Подсвеченную строку из истории можно скопировать в командную строку оболочки (перемещение подсветки — клавишами <t> и <↓>, копирование — по клавише <Enter>).

Доступ к истории команд можно получить также по комбинациям клавиш <Alt>+<P> или <Alt>+<N>, однако в этом случае вы не видите окна с перечнем команд. Вместо этого в командную строку выводится одна команда из списка и по комбинации <Alt>+<P> происходит смена этой команды на предыдущую, а по комбинации <Alt>+<N> — на следующую команду из истории команд.

Команда меню **Справочник каталогов** (<Ctrl>+<\>) позволяет создать список каталогов, которые часто используются, и обеспечить быстрый переход к нужному каталогу из этого списка. Для этого создается список меток (условных имен), присвоенных наиболее часто используемым каталогам. Этот список можно использовать для быстрого перехода в нужный каталог. Пользуясь диалоговым окном справочника каталогов, вы можете добавить новую метку в список или удалить ранее созданную пару метка/каталог. Для добавления метки можно также использовать комбинацию клавиш (<Ctrl>+<X>, <H>), по которой текущий каталог добавляется в справочник каталогов. Программа выдаст запрос на ввод метки для этого каталога.

Команда меню **Фоновые задания** позволяет вам управлять фоновыми заданиями, запущенными из Midnight Commander (такими заданиями могут быть только операции копирования и перемещения файлов). Используя эту команду меню или "горячие" клавиши <Ctrl>+<X>, <J>, вы можете остановить, возобновить или снять любое из фоновых заданий.

После выбора команды меню **Файл расширений** вы получаете возможность редактировать файл mc.ext, в котором можете связать с определенным расширением файла (окончанием имени после последней точки) программу, которая будет запускаться для обработки (просмотра, редактирования или выполнения) файла с таким расширением. Запуск выбранной программы будет осуществляться после установки подсветки на имя файла и нажатия клавиши <Enter>.

Команда **Файл меню** используется для редактирования пользовательского меню (которое появляется после нажатия клавиши <F2>).

6.11. Настройка программы Midnight Commander

Программа Midnight Commander имеет ряд установок (опций), каждая из которых может быть включена или выключена, для чего служат несколько диалоговых окон, доступных через меню **Настройки.** Опция включена, если поставлена (с помощью клавиши пробела) звездочка или знак "x" в скобках перед названием опции. Рассмотрим последовательно команды меню **Настройки,** через которое включаются/отключаются эти опции.

Команда **Конфигурация.** При выборе этой команды меню появляется диалоговое окно, изображенное на рис. 6.7. Параметры конфигурации, задаваемые в этом окне, делятся на три группы: **Настройки панелей, Пауза после исполнения и Прочие настройки.**

В поле **Настройки панелей** вы задаете значения следующих параметров.

- **Показывать резервные файлы.** По умолчанию программа Midnight Commander не показывает файлы, имена которых заканчиваются на ~ (подобно опции -в команды `ls`).
- **Показывать скрытые файлы.** По умолчанию Midnight Commander показывает все файлы, в том числе файлы, имена которых начинаются точкой (как `ls -a`).

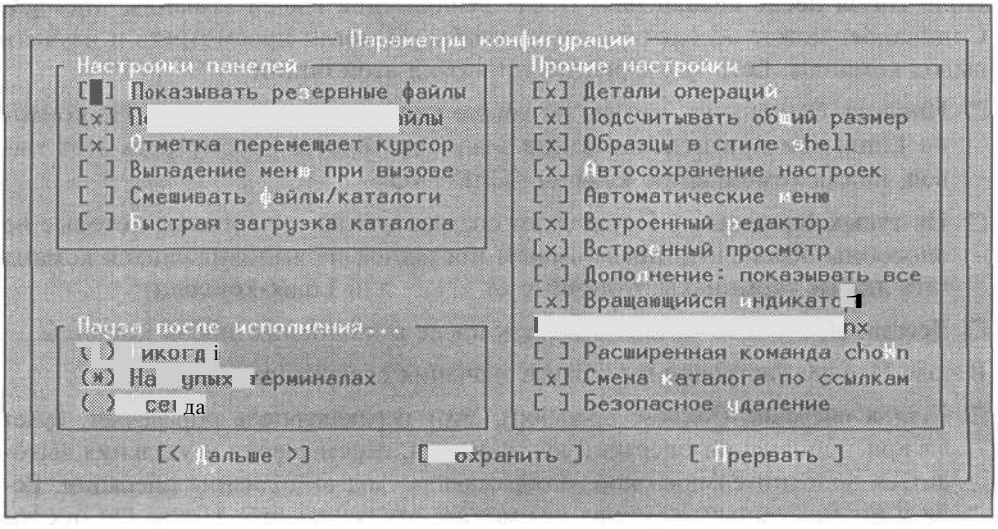


Рис. 6.7. Окно настроек параметров конфигурации

❐ **Отметка перемещает курсор.** Когда вы отмечаете файл (клавишами <Ctrl>+<T> или <Insert>), то по умолчанию подсветка на имени файла смещается на одну строку вниз.

П **Выпадение меню при вызове.** Если эта опция включена, то при вызове главного меню нажатием клавиши <F9>, будет сразу отображаться перечень команд меню (выпадающее меню). В противном случае активизируются только команды главного меню и вы должны (после выбора одного из них клавишами стрелок) нажать клавишу <Enter>, либо выбрать нужную команду по первой букве названия, и только после этого получите возможность выбрать команду выпадающего меню.

П **Смешивать файлы/каталоги.** Если эта опция включена, имена файлов и каталогов отображаются вперемежку. Если опция отключена, каталоги (и ссылки на каталоги) показываются в начале списка, а имена файлов — после имен всех каталогов.

П **Быстрая загрузка каталога.** По умолчанию эта опция выключена. Если вы активизируете ее, Midnight Commander будет использовать для вывода содержимого каталога следующий трюк: содержимое каталога перечитывается только в том случае, если изменилась запись в индексном дескрипторе каталога, т. е. если в каталоге создавались или удалялись файлы; если изменялись только записи в индексном дескрипторе файлов каталога (изменялся размер файла, режим доступа или владелец и т. п.), содержимое панели не обновляется. В этом случае (если опция включена) вы должны обновлять список файлов вручную (комбинацией клавиш <Ctrl>+<R>).

Поле **Пауза после исполнения.** После выполнения вашей команды Midnight Commander может обеспечить паузу, чтобы вы могли просмотреть и изучить вывод команды:. Есть три варианта установки этой опции.

П **Никогда.** Это значит, что вы не хотите видеть вывод команды. На консоли Linux или при использовании `xterm` вы можете просмотреть этот вывод, нажав комбинацию клавиш <Ctrl>+<O>.

П **На тупых терминалах.** Пауза будет создаваться на терминалах, которые не способны обеспечить показ вывода последней из выполнявшихся команд (это любые терминалы, отличные от xterm или Linux-консоли).

О **Всегда.** Программа обеспечит паузу после выполнения любой команды.

В поле **Прочие настройки** вы задаете значения следующих параметров.

П **Детали операций** (Verbose operation). Этот переключатель определяет, будет ли при выполнении операций копирования, перемещения и удаления выводиться дополнительное окно, отображающее ход выполнения операции. Если у вас медленный терминал, вы можете отключить этот вывод. Он отключается автоматически, если скорость вашего терминала меньше 9600 bps.

- **Подсчитывать общий размер** (Compute totals). Если эта опция включена, Midnight Commander перед выполнением операций копирования, перемещения и удаления подсчитывает общее число обрабатываемых файлов и их суммарный размер и показывает ход выполнения операции над этими файлами в виде диаграммы-полоски (правда, это слегка замедляет выполнение операций). Эта опция не работает, если отключена опция **Детали операций.**

- **Образцы в стиле shell** (Shell Patterns). По умолчанию команды отметки группы файлов (Select), снятия отметки (Unselect) и вывода списка файлов по фильтру (Filter) используют регулярные выражения, которые строятся по правилам, действующим в оболочке. Для того чтобы достичь такого эффекта, выполняются следующие преобразования: * заменяется на .* (ноль или больше символов); ? заменяется на . (в точности один символ) и . заменяется на обычную точку (literal dot). Если опция отключена, то регулярные выражения должны строиться так, как описано в руководстве man 1 ed.

ID **Автосохранение настроек.** Если эта опция включена, то при выходе из программы Midnight Commander значения всех настраиваемых параметров сохраняются в файле ~/.mc/ini.

- **Автоматические меню.** Если эта опция включена, пользовательское меню будет автоматически вызываться на экран при запуске программы. Это бывает полезно, если на компьютере работают неопытные пользователи (операторы), которые должны выполнять только стандартные операции.

П **Встроенный редактор.** Если эта опция включена, то для редактирования файлов вызывается встроенный редактор. Если опция выключена, то будет использоваться редактор, указанный в переменной окружения EDITOR. Если такой редактор не задан, будет вызываться vi.

О **Встроенный просмотр.** Если эта опция включена, для просмотра файлов вызывается встроенная программа просмотра. Если опция выключена, вызывается программа, указанная в переменной окружения PAGER. Если такая программа не задана, используется команда view.

П **Дополнение: показывать все.** В процессе ввода команд Midnight Commander может выполнять команду **Завершение ввода** при нажатии на клавиши <Alt>+<Tab>, пытаясь угадать окончание вводимой команды. По умолчанию при первом нажатии клавиш <Alt>+<Tab> он ищет все возможные варианты завершения и, если завершения неоднозначны (имеется много разных вариантов), то только издает звуковой сигнал. При втором нажатии клавиш <Alt>+<Tab> отображаются все возможные завершения. Если вы хотите видеть все возможные варианты завершения после первого нажатия клавиш <Alt>+<Tab>, включите эту опцию.

- **Вращающийся индикатор.** Если эта опция включена, Midnight Commander отображает в верхнем правом углу вращающуюся черточку, как индикатор того, что выполняется какое-то задание (операция).

- **Навигация в стиле lynx.** Если эта опция включена, вы имеете возможность использовать клавиши <→> для перехода в подсвеченный в данный момент каталог и <←> для перехода в родительский по отношению к текущему каталог (при условии, что командная строка пуста). По умолчанию опция отключена.
- **Расширенная команда chown.** Если эта опция включена, то при обращении к командам chmod или chown вместо них будет вызываться Расширенная команда chown.
- **Смена каталога по ссылкам.** Установка этой опции приводит к тому, что Midnight Commander будет следовать логической цепочке подкаталогов при выполнении команд смены каталога как в панели, так и по команде cd. Так же ведет себя по умолчанию bash. Если же опция не включена, Midnight Commander будет при выполнении команды cd следовать реальной структуре каталогов, так что если вы вошли в текущий каталог по ссылке на него, то по команде cd .. вы окажетесь в его родительском каталоге, а не в том каталоге, где расположена ссылка.
- **Безопасное удаление.** Если эта опция включена, непреднамеренно удалить файл будет сложнее. В диалоговом окне подтверждения удаления предлагаемая по умолчанию кнопка изменяется с Да на Нет и операция удаления непустого каталога должна будет подтверждаться путем выбора кнопки Да. По умолчанию эта опция отключена.

Диалоговое окно **Внешний вид** дает вам возможность изменить некоторые параметры отображения главного окна программы Midnight Commander на экране монитора (рис. 6.8).

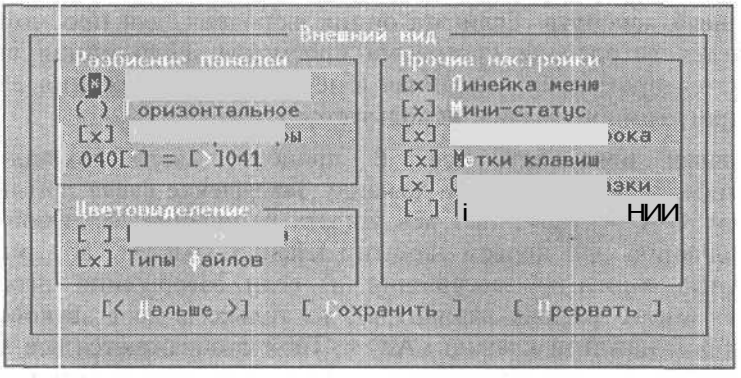


Рис. 6.8. Настройки внешнего вида программы

Вы можете вывести на экран или отменить вывод строки главного меню, командной строки, строки подсказок, мини-статуса, строки с подсказкой по

функциональным клавишам. На Linux- (или SCO-) консоли можно задать число строк, которые будут оставлены для отображения вывода команды.

Можно также указать, должны ли панели располагаться горизонтально или вертикально, изменить размеры панелей.

По умолчанию вся информация отображаются одним цветом, но вы можете сделать так, что права доступа и типы файлов были выделены другими цветами. Если включено выделение цветом прав доступа, поля perm и mode в форматах вывода, показывающие права пользователя, запустившего программу Midnight Commander, выделены цветом, определенным ключевым словом selected в секции [Colors] инициализационного файла ~/.mc/ini. Если включено выделение цветом типов файлов, то разными цветами выделяются каталоги, дампы памяти (файлы core), исполняемые файлы и т. д.

Если включена опция **Мини-статус,** в нижней части каждой панели выводится строка информации о выделенном подсветкой файле или каталоге каждой панели.

В диалоговом окне **Отображение символов** (рис. 6.9) вы указываете, в каком формате ваш терминал будет обрабатывать (вводить и отображать на дисплее) информацию, представленную байтами (например, записанную в файле).

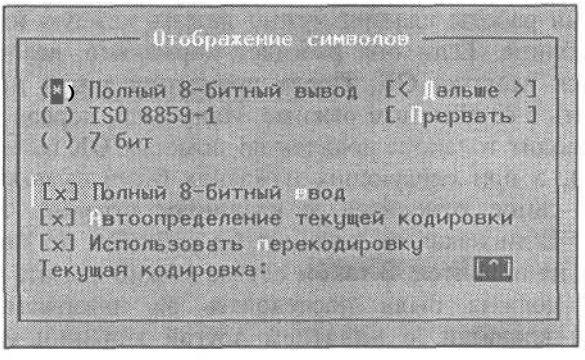


Рис. 6.9. Настройка вывода

Если терминал поддерживает только 7-битный вывод, то нужно выбрать установку 7 бит. Выбрав **ISO-8859-1,** вы получите все символы из таблицы ISO-8859-1, а выбор установки **Полный 8-битный вывод** имеет смысл на тех терминалах, которые могут отображать все 8-битные символы. Чтобы при просмотре или редактировании файлов в Midnight Commander видеть на экране кириллицу, установите опции **Полный 8-битный вывод и Полный 8-битный ввод** (как на рис. 6.9) и нажмите кнопку **Дальше.**

Установка опции **Полный 8-битный вывод** позволяет просматривать на экране файлы, набранные в кодировке KOI8-R. Если вам необходимо работать и с

другими кодировками (например, cp1251), то имейте в виду, что Валерий Студенников написал чудесную заплату (patch), которая позволяет выбирать кодовые страницы при просмотре и редактировании файлов в Midnight Commander (см. **http://www.linux.zp.ua:8100/mc/** или **http://www.sama.ru/~despaire/mc/my-patches.html**). Только вам придется самому скомпилировать версию tc с этой возможностью. Однако это не так уж сложно *(см. разд. 10.3)*.

Используя команду меню **Подтверждения,** вы можете сделать так, чтобы перед выполнением операций удаления, перезаписи и запуска файла на выполнение, а также перед выходом из программы tc, выдавался дополнительный запрос на подтверждение (либо отменить такие запросы, что несколько рискованно).

Команда **Распознавание клавиш** (Learn keys) вызывает диалоговое окно, в котором вы можете протестировать работу некоторых клавиш (<F1>—<F20>, <Home>, <End>), которые работают не на всех типах терминалов. В диалоговом окне появляется таблица с названиями клавиш, которые подлежат тестированию. Вы можете перемещать подсветку по названиям, используя клавишу <Tab> или клавиши, применяемые в редакторе vi (<H> — влево, <J> — вниз, <K> — вверх, <L> — вправо). Если один раз нажать на клавиши со стрелками, после чего возле их названий в таблице появится пометка **OK,** то их тоже можно будет использовать для управления перемещением.

Для тестирования работы клавиш нужно нажать каждую клавишу из перечисленных в таблице. Если она работает нормально, возле ее названия в таблице появится пометка **OK.** После появления такой пометки клавиша начинает работать в ее обычном режиме. Например, первое нажатие на клавишу <F1> приводит только к появлению пометки **OK** (если клавиша работает нормально), а при следующих нажатиях будет вызываться окно подсказки. То же самое относится к клавишам стрелок. Клавиша <Tab> работает всегда. Если какая-то клавиша не работает, то после нажатия на нее пометка **OK** не появится. В таком случае можно связать ту реакцию системы, которая должна была последовать за нажатием неработающей (отсутствующей) клавиши, с нажатием другой клавиши или комбинации клавиш. Для этого надо переместить подсветку на название неработающей клавиши (используя мышь или клавишу <Tab>) и нажать клавишу <Enter> или клавишу пробела. Должно появиться окно красного цвета, в котором вас просят нажать клавишу, которая будет использоваться вместо неработающей. Для отмены операции нажмите клавишу <Esc> и дождитесь, пока красное окно исчезнет. Либо выберите и нажмите ту комбинацию клавиш, которая будет служить заменой для неработающей клавиши (и тоже дождитесь закрытия окна). Когда закончите тестирование и настройку всех клавиш, вы можете сохранить эти настройки в секции `[terminal:TERM]` вашего файла ~/.mc/ini (где TERM — название используемого терминала) либо отказаться от запоминания изменений. Если все клавиши работают нормально, то сохранение, конечно, не требуется.

Команда **Виртуальные ФС** вызывает диалоговое окно, в котором вы можете задать значения некоторых параметров, связанных с использованием виртуальных файловых систем.

Команда **Сохранить настройки** обеспечивает сохранение выбранных значений параметров в ini-файле программы.

Я надеюсь, что приведенный в этом разделе материал позволит вам начать работать с программой Midnight Commander. Но возможности этой программы гораздо шире, чем это описано в настоящей главе. В *гл. 12* вы еще прочитаете о встроенном редакторе CoolEdit программы Midnight Commander. Кроме того, можно воспользоваться встроенной подсказкой, вызываемой по клавише <F1>. Перевод файла подсказки на русский язык (в виде bzip2-архива) вы можете найти на сайте **http://linux-ve.chat.ru**. Разархивируйте его, поместите в каталог /usr/lib/mc заменив расположенный там файл mc.hlp, и вы будете при нажатии клавиши <F1> получать подсказку на русском языке.

Глава 7

Графический интерфейс

Хотя Linux представляет собой очень мощную и развитую операционную систему, но, если работать с ней только через интерфейс командной строки, она довольно трудна в обращении и "недружелюбна" к пользователю. Все необходимые операции выполняются путем запуска отдельных команд, перечень которых огромен, и которые надо помнить наизусть.

Широко известной альтернативой интерфейсу командной строки является так называемый графический интерфейс, который обеспечивает дополнительные удобства для пользователя, в частности, возможность запуска программ в отдельных окнах, обозначения программ (или других обьектов) в виде маленьких картинок (пиктограмм, значков, иконок), возможность оперировать с обьектами с помощью мыши, а также гораздо большую плотность информации на том же пространстве экрана.

Естественно, что для ОС Linux существуют средства, обеспечивающие дружественный к пользователю графический интерфейс. На первый взгляд он очень похож на широко известный графический интерфейс Microsoft Windows, но его внутреннее устройство принципиально отличается. В этой главе мы рассмотрим, как этот интерфейс работает и как его настроить.

7.1. XFree86 и его составные части

Графический интерфейс в Linux строится на основе стандарта X Window System (заметьте, что Window, а не Windows) или просто "X" (в просторечии — "иксы"), первоначальный вариант которого был разработан в 1987 г. в Массачусетском технологическом институте. Начиная со второй версии этот стандарт поддерживался консорциумом X, созданным в январе 1988 г. с целью унификации графического интерфейса для ОС UNIX. С 1997 г. консорциум X преобразован в X Open Group (**http://www.x.org**). В настоящее время действует версия 11 выпуск 6 стандарта на графическую подсистему для UNIX-систем, которая кратко обозначается как X11R6.

Свободно распространяемая реализация стандарта X11R6 для UNIX-систем с процессорами 80386/80486/Pentium (в том числе для ОС Linux) была создана группой программистов, которую вначале возглавлял Дэвид Вексельблат (David Wexelblat). Эта реализация известна как XFree86 (http://www.xfree86.org), и может использоваться не только в Linux, но и в System V/386, 386BSD, FreeBSD и других версиях UNIX для систем на базе

процессоров Intel x86. В настоящее время выпущена уже 4-ая версия XFree86, однако и 3-я версия еще широко используется и входит в состав основных дистрибутивов Linux.

Система X Window построена на основе модели клиент/сервер. Правда, модель эта в данном случае используется как бы в "перевернутом" виде. Дело в том, что X-сервер работает на компьютере пользователя (а не на каком-то удаленном "сервере") и обеспечивает вывод изображения на экран монитора. X-сервер работает непосредственно с "железом": видеосистемой, устройствами ввода и динамиком. Может быть, для некоторых читателей назначение этой программы будет понятнее, если назвать ее драйвером видеосистемы. Эта программа захватывает оборудование и предоставляет его возможности другим программам как ресурсы (собственно, именно поэтому она и считается сервером) по особому протоколу, который называется X-протокол, или протокол сетевой связи (X Network Protocol). Кстати, специализированный компьютер, на котором исполняется исключительно X-сервер, называется (аппаратным) X-терминалом.

Но сам X-сервер изображение не формирует, он только "доставляет" графику видеодрайверу. Если запустить только X-сервер, вы увидите просто серый экран с характерным крестиком курсора посредине. С помощью мыши этот крестик можно перемещать по экрану. И все! На нажатие кнопок мыши и клавиш никакой видимой реакции не следует. И невидимой тоже — сервер готов передавать эти сигналы своим клиентам, а клиенты пока не запущены. Хотя на самом деле некоторые комбинации клавиш X перехватывает и обрабатывает. Эго <Ctrl>+<Alt>+<Backspace> — завершение работы сервера (если эта возможность не запрещена при конфигурации), <Ctrl>+<Alt>+<+> и <Ctrl>+<Alt>+<–> — "горячее" переключение доступных видеорежимов, и <Control>+<Alt>+<F#> -- переключение в другую виртуальную консоль.

Чтобы получить на экране какие-то более содержательные изображения, одного X-сервера недостаточно, надо запустить менеджер окон и хотя бы одну программу-клиент, которая будет формировать изображение. В роли "клиентов" X-сервера выступают приложения, работающие с X Window, например графический редактор GIMP, текстовый редактор Corel WordPerfect, эмулятор терминала xterm и др.

Между клиентами и сервером стоят еще два очень важных компонента графического интерфейса: библиотека графических функций X-Lib и менеджер окон (рис. 7.1). X-Lib содержит графические функции, которые обеспечивают выполнение низкоуровневых операций с графическими образами. Менеджер окон вызывает функции из X-Lib для управления дисплеем и выполнения любых преобразований изображений в окнах.

Когда X-приложение запускается, оно передает управление менеджеру окон. Менеджер оксн обеспечивает выполнение всех операций с окнами: прори-

совку рамок, меню, иконок, полос прокрутки и других элементов окна, а также предоставляет возможность изменять вид и положение окна в процессе работы в соответствии с потребностями пользователя.

Менеджер окон также вызывает соответствующие функции для программ-клиентов в тех случаях, когда пользователь взаимодействует с приложением с помощью клавиатуры и мыши. Именно поэтому при настройке XFree86 необходимо задать не только видеокарту, но и мышь и клавиатуру. Оконному менеджеру нужно знать характеристики этих устройств, чтобы выполнять свои задачи.

Расширенные графические среды типа Motif, CDE, KDE, GNOME, GNUStep и т. д. не замещают перечисленные выше компоненты системы X Window, а расширяют и дополняют их. KDE, например, добавляет библиотеку графических функций Qt в дополнение к X-Lib. Motif имеет собственный набор графических функций. GNOME использует библиотеку GTK+, которая составляет основу GIMP. Кроме того, в GNOME используется также CORBA (The Common Object Request Broker Architecture — универсальная архитектура посредничества при запросе объектов) и библиотека Imlib для дальнейшего расширения возможностей графической подсистемы.

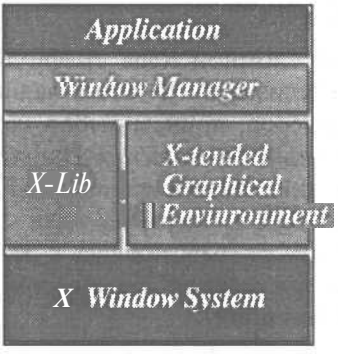


Рис. 7.1. Архитектура графической системы в Linux[1]

Поскольку клиент и сервер являются отдельными процессами, они могут работать на разных компьютерах, а взаимодействовать по сети. Приложения можно запустить, например, на мейнфрейме, а картинка будет выводиться на экран персонального компьютера. Эта очень мощная особенность системы X Window является одним из основных отличий ее от MS Windows.

Кстати, в Linux (и вообще в UNIX) нет жесткого деления между графическими и текстовыми программами, как в MS Windows или OS/2. С точки

[1] Рисунок заимствован из статьи J. Barr в LinuxWorld (**http://www.linuxworld.com/lw-1999-04/lw-04-vcontrol.html**).

зрения системы нет разницы между программой, работающей в графике, и обычной. Программы для графического режима запускают как обычные программы, т. е. из командной строки, из Midnight Comander и т. п. Единственным необходимым условием для их работы является то, что должен работать X-сервер. При необходимости программа сама обращается к X-серверу (через TCP/IP в общем случае, и через локальный сокет в частном, когда X-сервер и X-клиент работают на одной машине). Делает она путем вызова библиотечных функций из X-Lib. Все, что ей надо для работы, это знать, где искать X-сервер (для этого используется либо переменная окружения DISPLAY, либо опция в командной строке). Более того, существуют даже программы, которые умеют работать и с X-сервером и с обычным текстовым терминалом (например, emacs) и сами разбираются при старте, как именно им работать в данном случае.

Конечно, приведенный выше рис. 7.1 очень схематичен. К примеру, как мы уже упоминали, взаимодействие библиотеки X-Lib с сервером в общем случае осуществляется по протоколам TCP/IP. Так что следовало бы еще отобразить программное обеспечение, реализующее протоколы TCP/IP. Еще одним важным ресурсом графической подсистемы являются шрифты. Оперировать со шрифтами может как непосредственно X-сервер, так и специальная программа, которая называется Сервер шрифтов, и которую также следовало включить в рисунок.

Для каждого из типовых компонентов графической системы существует множество конкретных реализаций. В состав пакета XFree86 версии 3 входят несколько различных серверов, причем выбор конкретного сервера зависит от того, какие у вас видеоплата и монитор. Например, сервер XF86_Mono – это сервер для монохромных видеорежимов; XF86_S3 — сервер для карт, основанных на S3; XF86_S3V – сервер для S3 ViRGE и ViRGE/VX; XF86_SVGA — сервер для карт, работающих в режимах Super-VGA. Последнюю версию с писка поддерживаемых X-сервером видеокарт вы можете найти на сайте **http://www.xfree86.org/**.

В 4-й версии XFree86 уже один X-сервер для большинства видеоадаптеров, называется он XFree86 и располагается в каталоге /usr/X11R6/bin/. Обычно на него делается ссылка с именем X, так что запустить X-сервер можно просто введя в командной строке букву X.

Различных менеджеров окон тоже существует очень много, и вы можете использовать любой из них по своему выбору. Вот несколько примеров.

- **fvwm** — Free Virtual Window Manager.
- **fvwm2** — улучшенный вариант fvwm, позволяющий, в частности, использовать различные темы рабочего стола и динамические меню. Официальный сайт разработчиков fvwm и fvwm2 — **http://www.hpc.uh.edu/fvwm/.**
- **fvwm95** — менеджер окон с графическим интерфейсом в стиле Windows 95.

- **IceWM (http://berta.fri.uni-lj.si/~markom/icewm)** — это оконный менеджер, который может эмулировать различные стандарты оконных систем (в частности, OS/2 Workplace Shell), имеет, по некоторым отзывам, довольно маленький объем и быстро работает.
- **Enlightenment (http://www.EnIightenment.org)** — один из самых развитых менеджеров окон.

Можно упомянуть также Motif — коммерческий продукт, используемый во многих UNIX-системах, или его бесплатный аналог LessTif **(http://www.lesstif.org),** а также Xview — Linux-версию интерфейса OpenLook фирмы Sun.

Описание нескольких оконных менеджеров (Window Maker, IceWM, FLWM, FailSafe) и их сравнительный анализ вы можете найти в книге А. Федорчука *(см. [П1.6] приложения).*

Разные менеджеры окон могут обеспечивать различный вид окон за счет использования различных рамок и оконных меню. Но все они используют одну и ту же базовую графическую утилиту X Window — X-сервер.

Итак, вы теперь в общих чертах представляете, из каких основных частей формируется графический интерфейс в Linux. Как видите, строится он по модульному принципу и вполне возможно "собрать" его самостоятельно из отдельных компонентов. Но для начинающего пользователя этот путь не самый легкий. К счастью, и не обязательно этим путем идти, поскольку разработаны (и включены практически во все дистрибутивы) так называемые интегрированные графические среды. Наиболее известными представителями таких сред являются KDE **(http://www.kde.org)** и GNOME **(http://www.gnome.org).**

Основу интегрированной графической среды KDE (K Desktop Enviroment) образует расширенная библиотека графических функций Qt фирмы Troll Tech **(http://www.trolltech.com/).** С использованием этой библиотеки построены собственный оконный менеджер kwm, файл-менеджер kfm, центр управления KDE (аналог панели управления Windows) и множество других компонентов, вплоть до собственного офисного пакета KOffice. Сказанное не означает, что все эти компоненты разрабатываются какой-то одной фирмой. Просто различные и независимые разработчики используют единый подход, единую идеологию создания и оформления программных продуктов. В результате получается функционально полный пакет программ, в рамках которого могут быть решены практически все задачи управления компьютером и программной средой (если не сказать, что вообще все задачи, решаемые на компьютере).

В состав версии 2.1 KDE включен файловый менеджер Konqueror, который предоставляет уникальные возможности доступа к файлам. Кроме того, что он позволяет просматривать файлы большинства известных форматов на

локальных дисках, он является и интернет-браузером, по своим возможностям вполне сравнимым с Internet Explorer или Netscape Navigator.

Другой графической средой того же класса, что и KDE, является пакет GNOME (GNU Network Object Model Environment), который разрабатывается в рамках проекта GNU, а значит, относится к классу свободно распространяемого ПО (KDE до недавнего времени не полностью соответствовал этому понятию, потому что библиотека Qt распространялась не на условиях GPL; хотя сейчас ситуация изменилась и KDE тоже является свободно распространяемым). GNOME строится на основе библиотеки графических функций GTK+.

Существуют и другие разработки интегрированных графических сред, которые, однако, пока не достигли той степени развития, как KDE или GNOME, например, Xfce **(http://www.xfce.org),** dfm **(http://www-c. informatik.uni-hannover.de/~kaiser/dfm/dfm.html) и др.**

До недавнего времени установка и настройка графического интерфейса Linux представляла собой довольно непростую задачу. Однако программы инсталляции последних версий дистрибутивов Linux (например, русифицированой версии Red Hat Linux 7.1) уже вполне успешно с ней справляются, автоматически определяя тип монитора и видеоадаптера и выбирая подходящее разрешение экрана. Но я все же приведу здесь некоторые рекомендации по настройке X Window для начинающих пользователей, имея в виду, что с некоторыми вариантами аппаратного обеспечения настройки все же придется подбирать вручную. А для того, чтобы подходить к процессу настройки осознанно, надо в общих чертах представлять, как работает видеосистема вашего компьютера.

7.2. Как работает видеосистема компьютера

Вы должны понимать, что этот раздел не является руководством для специалистов по созданию видеосистем. Изложение работы видеосистемы здесь дается очень поверхностное, только с той целью, чтобы читатель понимал значение параметров, которые ему придется как-то изменять при настройке графического режима. В основу этого раздела положена статья И. Николаева "FAQ по настройке монитора в Xfree" **(http://knot.pu.ru/faq/xfaq.html).**

Видеосистема компьютера состоит из видеоадаптера и монитора (когда мы будем говорить о мониторах, будут иметься в виду мониторы, построенные на электронно-лучевых трубках).

Как вы знаете, изображение на экране монитора состоит из отдельных точек. Точки формируются электронным лучом и располагаются в виде строк. Монитор делает две независимых вещи: развертку луча и высвечивание от-

дельных точек, из которых строится изображение на экране. Управление монитором заключается в том, чтобы задать число точек в строке (разрешение по горизонтали), число строк на экране (разрешение по вертикали) и интенсивность каждого из трех основных цветов в каждой точке.

Функцию управления монитором осуществляет видеоадаптер. Видеоадаптер передает в монитор три сигнала: видеосигнал (RGB), строчную синхронизацию (HS) и кадровую синхронизацию (VS). По сигналу горизонтальной (строчной) синхронизации происходит возврат луча с конца каждой строки к началу следующей, а сигнал вертикальной (кадровой) синхронизации определяет момент возврата луча из правого нижнего угла экрана в верхний левый. Частоты генерации этих двух сигналов (измеряемые числом импульсов в секунду) необходимо знать для правильной установки и настройки X-сервера. Значения частот горизонтальной и вертикальной синхронизации должны быть указаны в документации к вашему монитору. Частота вертикальной синхронизации (обозначим ее VSF) обычно указывается в герцах (Гц) и находится в пределах 50—180 Гц. Частота горизонтальной синхронизации (HSF) задается в килогерцах (кГц) и принимает значения в диапазоне от 31 до 135 кГц.

Современные мониторы обычно являются многочастотными, т. е. допускают выбор частот вертикальной и горизонтальной синхронизации из определенного диапазона допустимых значений. Некоторые мониторы (особенно дешевые) могут иметь несколько фиксированных значений допустимых частот. Целесообразно выбирать максимально возможные значения частот синхронизации из числа допустимых для получения наилучшего разрешения. Однако будьте осторожны: установка значений, превышающих допустимые, может повредить монитор!

Есть еще одна важная частота — число точек, которые могут быть отображены на экране за одну секунду. Электронный луч перемещается по экрану с конечной скоростью (которая ограничена не скоростью перемещения самого луча, а параметрами микросхем видеоадаптера). В английской документации эта частота называется "the card's dot clock". Будем называть эту частоту тактовой частотой развертки и обозначать как DCF (dot clock frequency).

Следующий существенный параметр - это частота обновления экрана (которую будем обозначать как SRR — screen refresh rate). Чем выше эта частота, тем меньше устают глаза при работе с компьютером, потому что уменьшается мерцание экрана. Не рекомендуется выбирать частоту обновления экрана меньше 60 Гц — частоты мерцания флуоресцентных ламп. Стандарт VESA рекомендует выбирать для нее значение не менее 72 Гц.

Но задать очень большое значение частоты обновления экрана невозможно, потому что она не может быть больше, чем тактовая частота развертки (DCF), поделенная на произведение числа точек в строке и числа строк на

экране. На самом деле частота обновления экрана еще меньше, потому что на перемещение электронного луча от конца строки в начало следующей и из левого нижнего угла экрана в правый верхний требуется некоторое время, т. е. дополнительные такты работы тактового генератора видеоадаптера. Кроме того, для создания четких границ изображения на экране электронный луч обычно перемещается на несколько точек левее и правее видимой части изображения, а также пробегает одну-две строки выше и ниже изображения (это темные полосы, окружающие изображение на экране). Поэтому фактически число необходимых для вывода изображения точек в строке (обозначим его HFL — Horizontal Frame Length — размер фрейма по горизонтали) и число необходимых строк на экране (Vertical Frame Length – VFL) больше, чем размеры видимого изображения. Точнее

HFL = (число невидимых точек слева) + (число видимых точек в строке) + + (число невидимых точек справа) + (число тактов, необходимых для перевода луча к началу новой строки).

Аналогичное соотношение действует для размера фрейма по вертикали – VFL и поэтому имеет место формула:

$$SRR = DCF / (HFL \times VFL).$$

Именно эта частота и является частотой обновления экрана (SRR = VSF) и должна лежать в границах, заданных для частоты вертикальной синхронизации вашего монитора. Таким образом, частота вертикальной синхронизации монитора и тактовая частота видеоадаптера накладывают ограничения на предельное разрешение изображения на экране.

Еще одно ограничение накладывается обьемом памяти на плате видеоадаптера. В цветном мониторе каждая точка высвечивается тремя электронными лучами, которые одновременно направлены в данную точку. Цвет каждой отдельной точки на экране формируется путем смешивания трех основных цветов: красного (R), зеленого (G) и голубого (B). Интенсивность каждого из этих трех цветов определяется уровнем сигнала в соответствующем луче. Изображение дня вывода на экран формируется в видеопамяти, которая физически расположена на плате видеоадаптера, но входит в общее адресное пространство оперативной памяти компьютера. Изображение хранится в памяти в цифровом виде, и преобразование его в аналоговый сигнал RGB является одной из основных задач видеоадаптера, для чего на плате видеоадаптера обычно ставится цифро-аналоговый преобразователь — ЦАП.

Количество возможных цветов для каждой точки, очевидно, ограничено тем, сколько различных уровней сигнала для каждого луча может сформировать видеоадаптер, *Е.* также обьемом видеопамяти. Но обычно выходные сигналы видеоадаптера обеспечивают число уровней, достаточное для отображения картинки, записанной в видеопамять, так что определяющим параметром становится именно ее обьем.

Формула, определяющая ограничения на разрешение экрана и количество воспроизводимых цветов, очень проста: *(объем видеопамяти в байтах)* должен быть не меньше, чем *(число видимых точек в строке)* x *(число видимых строк на экране)* x *(число байтов на точку)*.

Например, если вы хотите выбрать разрешение экрана 1024x768 и иметь 16 миллионов цветов для каждой точки (4 байта на точку), то вам надо иметь 1024x768x4=3 145 728 байт памяти. Если же у вас всего 2 Мбайт памяти, то придется либо выбрать меньшее разрешение, либо меньшее количество цветов. Вы легко можете сами составить таблицу соответствий между различными комбинациями значений видеопараметров и объемом необходимой памяти.

О том, как вся эта теория применяется при настройке графического режима, будет рассказано в следующих разделах.

7.3. Конфигурирование X-сервера

Рассмотрим ситуацию, когда после инсталляции Linux вы либо не можете выйти в графический режим, либо недовольны тем, как выводится изображение в графическом режиме. Если вы при инсталляции задали автоматический выход в графический режим, то вам может показаться, что система вообще отказывается загружаться. Именно поэтому я рекомендовал вам при инсталляции отказаться от автоматической загрузки в графику. Если вы все же попали в такую ситуацию, то попробуйте выйти в однопользовательский режим. О том, как это сделать, рассказано в *разд. 8.2.5*.

Итак, в графический режим вы выйти не можете, но текстовый режим вам доступен, и, значит, можно воспользоваться некоторыми программами или командами ОС Linux для настройки графического режима. Будем предполагать, что пакет XFree86 у вас установлен и все файлы, упоминаемые ниже, на диске имеются. Если это не так, то сначала установите пакет XFree86, (о том, как устанавливаются программные пакеты, рассказано в *гл. 10)*. Если эти предварительные условия выполнены, я надеюсь, что приводимые ниже инструкции по настройке графического режима позволят вам осуществить эту настройку.

Предупреждение

Согласно документации к Xfree86 для некоторых типов мониторов, особенно старых моделей, задание недопустимых значений частот вертикальной и горизонтальной развертки может повредить монитор. Поэтому автор не может гарантировать, что, следуя приведенным ниже советам, вы получите желаемый результат и не повредите вашему компьютеру! Все, что вы будете делать, вы будете делать на свой страх и риск!

7.3.1. Сбор необходимых данных

Для настройки вам потребуются некоторые данные, а именно:

- названия фирм — производителей видеоадаптера и монитора (берутся из документации; если нет, можно обойтись и без них);
- тип набора микросхем, применяемых в видеоадаптере (по нему определяется тип X-сервера, который должен работать у вас);
- объем имеющейся видеопамяти;
- допустимые интервалы частот горизонтальной и вертикальной синхронизации для нашего монитора (берутся из документации на монитор; эти две частоты надо узнать обязательно).

К числу необходимых для установки X Window сведений относятся также тип вашей мыши и клавиатуры. Работать в графическом режиме без мыши довольно неудобно, а без клавиатуры и вовсе нельзя, так что X-сервер должен быть настроен на использование имеющихся у вас типов этих устройств.

Кое-какую информацию можно получить с помощью программы SuperProbe (ее вывод можно перенаправить в файл, например, sprobe.txt):

```
[root]# SuperProbe > sprobe.txt
```

Заглянув в этот файл (воспользуйтесь клавишей <F3> в программе Midnight Commander), вы узнаете тип набора микросхем и обьем имеющейся у вас видеопамяти. У меня, например, SuperProbe выдала в одном случае

```
Chipset: S3 Trio64 (Port Probed)
Memory: 1024 Kbytes
RAMDAC: Generic 8-bit pseudo-color DAC
    (with 6-bit wide lookup tables (or in 6-bit mode))
```

в другом случае

```
Chipset: Trident 3DImage985 (PCI Probed)
Memory: 1096 Kbytes
RAMDAC: Trident Built-in 15/16/24-bit DAC
    (with 6-bit wide lookup tables (or in 6-bit mode))
```

Тип микросхем видеоадаптера необходимо знать в том случае, если у вас 3-я версия XFree86. Вообще говоря, программа инсталляции Linux автоматически определяет, какой сервер у вас должен стоять и инсталлирует его. Одновременно в каталоге /etc/X11/ формируется ссылка с именем X на этот сервер примерно такого вида:

```
@X -> /usr/X11R6/bin/XF86_SVGA      (для 3-й версии XFree86).
@x -> /usr/X11R6/bin/XFree86        (для 4-й версии XFree86).
```

Загляните в каталог /etc/X11 и если обнаружите, что файла-ссылки с именем X там нет, создайте такую ссылку командой

```
[root]# ln -s /usr/X11R6/bin/XFree86 X
```

(естественно, файл /usr/X11R6/bin/XFree86 должен существовать).

7.3.2. Структура файла /etc/X1 1/XF86Config

Конфигурация X-сервера определяется файлом /etc/X11/XF86Config (для 3-й версии XFree86) или /etc/Xl1/XF86Config-4 (для 4-й версии XFree86), поэтому самый правильный способ настройки X-сервера состоит в прямом редактировании этого файла.

Примечание

Если быть точным, то X-сервер при запуске ищет конфигурационный файл в нескольких местах, а именно:

- /etc/X11/XF86Config-4
- /etc/XF86Config
- /etc/X11/XF86Config
- /usr/X11R6/etc/XF86Config
- <XRoot>/lib/X11/XF86Config,

где <XRoot> означает каталог, в котором была установлена система XFree86 (обычно это /usr/X1 1 R6).

Создавать файл XF86Config полностью с нуля не стоит. Если вы не отказались от установки X Window в процессе инсталляции системы (именно от установки, а не от автоматического запуска, не путайте!), то такой файл у вас уже есть. Если по каким-то причинам вы устанавливали XFree86 отдельно от инсталляции Linux, то запустите одну из программ, которые позволяют такой файл сформировать. Когда у меня стояла 3-я версия XFree86, я пользовался для создания этого файла программой Xconfigurator. В 4-й версии для создания конфигурационного файла можно воспользоваться утилитами `xf86config` или `xf86cfg`, или же использовать опцию `-configure` X-сервера. Для этого дайте такую команду:

```
[root]# /usr/X11R6/bin/XFree86 -configure
```

Эта команда молча отрабатывает, сообщая в конце, что создала новый конфигурационный файл /root/XF86Config.new. Ниже приводится файл /root/XF86Config.new, который сформировался у меня (в дальнейшем речь пойдет в основном о версии 4 XFree86, так что особых упоминаний об этом делать не будем).

```
Section "ServerLayout"
        Identifier     "XFree86 Configured"
```

```
        Screen        0 "Screen0" 0 0
        InputDevice   "Mouse0" "CorePointer"
        InputDevice   "Keyboard0" "CoreKeyboard"
EndSection

Section "Files"
        RgbPath "/usr/X11R6/lib/X11/rgb"
        ModulePath "/usr/X11R6/lib/modules"
        FontPath "/usr/X11R6/lib/X11/fonts/misc/"
        FontPath "/usr/X11R6/lib/X11/fonts/Speedo/"
        FontPath "/usr/X11R6/lib/X11/fonts/Type1/"
        FontPath "/usr/X11R6/lib/X11/fonts/CID/"
        FontPath "/usr/X11R6/lib/X11/fonts/75dpi/"
        FontPath "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection

Section "Module"
        Load "extmod"
        Load "xie"
        Load "pex5"
        Load "glx"
        Load "dri"
        Load "GLcore"
        Load "dbe"
        Load "record"
EndSection

Section "InputDevice"
        Identifier "Keyboard0"
        Driver     "keyboard"
EndSection

Section "InputDevice"
        Identifier "Mouse0"
        Driver     "mouse"
        Option     "Protocol"  "PS/2"
        Option     "Device"  "/dev/mouse"
EndSection
```

```
Section "Monitor"
        Identifier "Monitor0"
        VendorName "Monitor Vendor"
        ModelName  "Monitor Model"
EndSection

Section "Device"
        Identifier  "Card0"
        Driver      "trident"
        VendorName  "Trident"
        BoardName   "3DImage985"
        BusID       "PCI:1:0:0"
EndSection

Section "Screen"
        Identifier "Screen0"
        Device     "Card0"
        Monitor    "Monitor0"
        Subsection "Display"
                Depth 8
        EndSubSection
        Subsection "Display"
                Depth 15
        EndSubSection
        Subsection "Display"
                Depth-16
        EndSubSection
        Subsection "Display"
                Depth 24
        EndSubSection
EndSection

Section "DRI"
EndSection
```

Как видите, файл XF86Config состоит из нескольких секций, имеющих следующую структуру:

```
Section "Название_секции"
        Identifier      "Name"
```

```
    SectionEntry
    ...
EndSection
```

Строки, начинающиеся символом #, являются комментариями. Секции могут быть расположены в файле в произвольном порядке. Могут присутствовать следующие секции:

- ServerLayout — общие установки;
- InputDevice — описания устройств ввода;
- Screen — конфигурация экрана;
- Device — описания графических карт;
- Monitor — описания монитора;
- Modes — описания видеорежимов;
- Files — пути к файлам;
- ServerFlags — опции сервера;
- videoAdaptor — описание адаптера Xv (как сказано в интерактивном руководстве man, никто не говорит, что это такое, даже если знает!);
- Module — динамически загружаемые модули;
- DRI — конфигурация DRI;
- Vendor — установки для оборудования конкретных поставщиков (Vendor-specific configuration).

Причем наличие всех секций не обязательно, но могут существовать несколько разных секций одного типа (только они должны иметь уникальные идентификаторы).

Секция ServerLayout имеет наивысший приоритет, т. е. именно с нее начинается анализ файла XF86Config при загрузке X-сервера. Эта секция определяет, какие устройства ввода/вывода будут использоваться в X-сессии. Устройства ввода — это клавиатура и мышь, которые задаются в секциях InputDevice (такие секции создаются отдельно для каждого устройства). Устройства вывода обычно состоят из нескольких независимых компонентов (графический адаптер и монитор), которые связываются воедино в секции Screen, на которую указывает ссылка в секции serverLaycut. Таким образом, секция ServerLayout должна содержать, как минимум, следующие строки:

```
Section "ServerLayout"
        Identifier      "MainSection"
        Screen          0   "Screen0"   O   O
        InputDevice       "Mouse0"          "CorePointer"
```

```
    InputDevice     "Keyboard0"     "CoreKeyboard"
EndSection
```

Секция Screen определяет используемые монитор и видеоадаптер и задает режимы работы экрана. Она может иметь примерно такой вид:

```
Section " Screen"
    Identifier  "Screen0"
    Device      "Trident 3DImage985 (generic)"
    Monitor     "ViewSonic|ViewSonic G771"
    DefaultColorDepth 24
  Subsection  "Display"
    Depth       8
    Modes        "1024x768"  "800x600"  "640x400"
    ViewPort    O O
  EndSubsection
  Subsection  "Display"
    Depth       16
    Modes        "1024x768"  "800x600"  "640x480"
    ViewPort    O O
  EndSubsection
  Subsection  "Display"
    Depth       24
    Modes        "1024x768"  "800x600"  "640x480"
    ViewPort    O O
  EndSubsection
  Subsection  "Display"
    Depth       32
    Modes        "1024x768"  "800x600"  "640x480"
    ViewPort    O O
  EndSubsection
EndSection
```

Как видите, она содержит указания на используемый графический адаптер (задается в секции Device с идентификатором "Matrox Millenium G200") и монитор (задается в секции Monitor с идентификатором "Monitor0").

Секция screen может содержать несколько подсекций (Subsection) Display, по одной такой подсекции на каждую глубину цвета. В каждой такой подсекции вы должны прописать те режимы монитора, которые будете использовать. Режимы задаются в строке Modes. Они указываются путем перечисления их наименований, взятых из секции Monitor (в точности в том виде,

как эти названия указаны после слова Modeline). В одной строке можно перечислить любое число таких имен режимов. Первый из указанных режимов будет запускаться по умолчанию, в остальные можно будет переключаться (циклически), нажимая комбинацию клавиш <Ctrl>+<Alt>+<+> или <Ctrl>+ +<Alt>+<–> (используются клавиши <+> и <-> на цифровой клавиатуре).

В этой подсекции указывается также размер виртуального экрана, который будет использоваться сервером. Соответствующая строка имеет вид "Virtual xdim ydim", где xdim и ydim — размерности виртуального экрана. Например, вы можете иметь дисплей с разрешением 800x600, а размер виртуального экрана задать равным 1024x768. Тогда в каждый момент времени вы будете видеть на дисплее только часть полного изображения. Надо, однако, учитывать, что видеопамять должна хранить изображение, равное по размеру виртуальному экрану, а также то, что нежелательно занимать всю память хранением виртуального экрана, поскольку в этом случае не остается резерва на кэширование, что может повлечь потерю 30—40% производительности сервера.

Секция Monitor обычно начинается тремя строками, в которых указываются производитель монитора и его модель, однако если у вас только одна секция Monitor, то эти строки вполне могут иметь вид:

```
Identifier      "Unknown"
VendorName      "Unknown"
ModelName       "Unknown"
```

Далее идут две очень важные строки, определяющие допустимые значения частот вертикальной и горизонтальной синхронизации. Для современных мульти-частотных мониторов эти строки могут иметь примерно такой вид:

```
HorizSync       30-70
VertRefresh     50-180
```

Для мульти-частотных мониторов с фиксированными частотами:

```
HorizSync       31.5, 35.2
VertRefresh     60, 65
```

Для мульти-частотных мониторов с несколькими интервалами допустимых частот:

```
HorizSync       15-25, 30-50
VertRefresh     40-50, 80-100
```

Частоты горизонтальной синхронизации задаются в килогерцах, частоты вертикальной синхронизации (обновления экрана) — в герцах.

Внимание

Обязательно проверьте, что здесь указаны значения, соответствующие характеристикам вашего монитора, приведенным в документации.

Секций Monitor в файле может быть несколько, они различаются уникальными идентификаторами (первая строка в секции). В 4-й версии XFree86 перечисление режимов монитора в этой секции является необязательным, поскольку X-сервер использует встроенный список стандартных VESA-режимов. Однако явное задание режима в секции Monitor и не возбраняется, причем, если заданный вами режим получит такое же название, как один из стандартных режимов, будут использованы ваши установки. Встроенные режимы с именами, не встречающимися в секции Monitor, будут использоваться сами по себе.

Видеорежимы могут быть заданы также в секции Modes. Таких секций в файле тоже может быть несколько. Каждая из них задает некоторый набор видеорежимов, на который можно сослаться из секции Monitor, используя ключевое слово `UseModes`.

Режимы могут задаваться в двух эквивалентных форматах: в виде одной строки или несколькими строками. Вот пример задания одного и того же режима в этих двух форматах:

```
Modeline "640x480example" 25.175 640 664 760 800 480 491 493 525 -HSync
+VSync
Mode "640x480example"
    DotClock      25.175
    Htimings      640 664 760 800
    VTimings      480 491 493 525
    Flags         "-HSync +VSync"
EndMode
```

Первое слово в каждом формате (Modeline или Mode), **как и слово** EndMode, являются служебными. По ним X-сервер определяет, что данная строка или группа строк служат для задания видеорежима. Следующее слово (в кавычках) является названием данного режима. Вы можете выбрать это название по своему вкусу (хотя, как пишет Игорь Николаев *(см. [П9.1] приложения)*, "я бы не стал использовать в названии ничего, кроме цифр и букв латинского алфавита").

Следующее число (оно может быть дробным) задает тактовую частоту развертки (частоту вывода точек на экран) в мегагерцах.

Далее следуют четыре целых числа (группа Htimings), определяющих параметры строки, а, значит, разрешение по горизонтали. Первое из этих чисел задает число видимых точек в строке. Чтобы пояснить значение следующих трех чисел, надо иметь в виду, что началом строки на экране считается не самая левая точка в строке (она попадает в невидимую, затемненную область экрана), а левая граница видимого изображения. Поэтому первое число в этой группе определяет число точек от начала до конца видимой части строки. Второе число — число точек от начала строки до начала импульса

горизонтальной синхронизации (в этот момент начинается перемещение электронного луча с правой границы экрана на левую).

Третье число — число тактов развертки от начала изображения до того момента, когда электронный луч будет готов к выводу следующей строки изображения. То есть разность третьего и второго чисел задает длительность интервала, отводимого на перевод электронного луча от правой (с точки зрения пользователя) границы экрана к левой границе. И, наконец, четвертое число — общее число тактов, затрачиваемых на формирование одной строки изображения (с учетом невидимой части и времени, необходимого для перевода луча на новую строку).

Аналогично формируются следующие 4 числа (VTimings) в описании формата, только в данном случае речь будет идти о числе строк на экране (видимых и невидимых) и импульсе вертикальной синхронизации (началом экрана считаем верхнюю границу видимой части изображения).

О последней группе параметров в описании видеорежима (группа Flags) мы поговорим позже, когда рассмотрим вопрос о том, как правильно сформировать значения числовых параметров.

В большинстве случаев нет необходимости в наличии секции Modes и строки UseModes, т. к. достаточно встроенного в сервер набора видеорежимов, соответствующих стандарту VESA. При запуске X-сервера он автоматически выбирает из возможных режимов лучший вариант, поддерживаемый монитором и видеокартой (решение принимается на основе значений, указанных в строках HorizSync и vertRefresh секции Monitor, так что очень важно, чтобы они были заданы корректно).

Следующая секция, Device, описывает вашу видеоплату. По-видимому, самая важная строка в этой секции — указание на драйвер. У меня эта строка имеет вид:

```
Driver      "mga"
```

В этой секции могут быть также указаны следующие данные:

- тип набора микросхем (Chipset);
- количество видеопамяти (указывается в килобайтах);
- допустимые значения тактовой частоты развертки (dot-clocks) или наименование используемого генератора тактовой частоты;
- тип RAMDAC (RAMDAC — это часть графического контроллера, отвечающая за преобразование изображения из цифровой в аналоговую форму).

Эти параметры были, по-видимому, критичны для старых видеоплат, но для современных они перестали играть существенное значение. Большинство современных видеоплат имеют программируемые генераторы частот, которые могут задавать тактовую частоту развертки в очень широких пределах.

Только в том случае, если у вас какой-то экзотический тип видеоплаты (особенно если вы знаете, что она поддерживает только ограниченное число частот генератора), надо позаботиться о корректном задании режимов в этой секции. В этом случае вначале воспользуйтесь программой SuperProbe для того, чтобы узнать нужные параметры, а затем пропишите полученные значения в секции Device.

Две секции inputDevice определяют мышь и клавиатуру (в 3-й версии XFree86 вместо секций inputDevice имелись секции: Keyboard и Pointer). Во-первых, здесь нужно указать тип клавиатуры. Помните, что обычная клавиатура для персональных компьютеров обозначается как "Generic 102-key PC (intl)" вместо предлагаемого по умолчанию варианта "Generic 101-key PC". Однако наиболее часто сейчас встречаются 104-клавишные клавиатуры, так что секция, определяющая конфигурацию клавиатуры, должна содержать следующие строки:

```
Option "XkbRules"    "xfree86"
Option "XkbModel"    "pc104"
Option "XkbLayout"   "ru"
Option "XkbOptions"  "grp:ctrl_shift_toggle"
```

Последняя из этих строк определяет переключатель, который будет менять раскладку клавиатуры с американской на русскую (в данном примере – <Ctrl>+<Shift>).

Кроме того, здесь можно указать протокол для работы с клавиатурой (Xqueue или Standart), скорость повтора (the repeat rate), а также переопределить значения некоторых клавиш. Для начала, пожалуй, не стоит тут слишком экспериментировать. Подробнее о настройке клавиатуры будет рассказано в *разд. 9.3.*

В отдельной секции inputDevice задаются параметры работы мыши. Самые существенные в этой секции 2 строки (для примера привожу строки из своего файла):

```
Option          "Protocol"      "PS/2"
Option          "Device"        "/dev/mouse"
```

определяющие протокол и файл устройства. Если ваша мышь подключена к специальному ("мышиному") разъему, то в качестве параметра Protocol должно быть указано либо PS/2, либо более длинное слово, оканчивающееся на PS/2. Остальные протоколы используются для мышей, подключаемых через последовательный порт. Обратите внимание на то, что не всегда протокол совпадает с названием фирмы-производителя. Так, некоторые из мышей фирмы Logitech используют либо протокол MouseMan, либо Microsoft. Если ваша мышь выпущена относительно недавно, можно поставить **Auto** в графе **Protocol.**

Секция Files содержит несколько строк, в которых прописаны пути к файлу с базой цветов (это таблица, задающая соответствия между цифровым и словесным заданием цветов, эту строку не стоит редактировать) и каталогам с файлами шрифтов. Путей к каталогам со шрифтами может быть задано несколько. Убедитесь только, что все указанные каталоги существуют и действительно являются каталогами шрифтов (там должен быть специальный файл fonts.dir, создаваемый командой `mkfontdir`; этот файл не рекомендуют создавать вручную). Если при запуске сервер будет выдавать сообщение типа `Can't` open default font `'fixed` или что-то еще в таком роде, это свидетельствует о том, что в секции Files ошибка в указании пути к файлу шрифтов (нужный шрифт не найден).

В этой же секции могут быть заданы пути к каталогам, содержащим загружаемые модули. Какие именно модули загружаются, и параметры загрузки определяются в секции Module, о которой мы сейчас говорить не будем, как **И О секциях** ServerFlags, VideoAdaptor, `DRI`, Vendor. **Оставьте ЭТИ** секции **В** том виде, как они созданы программой конфигурации.

7.3.3. Настройка /etc/X11/XF86Config

Теперь, когда вы знаете, как устроен файл XF86Config, определяющий настройки X-сервера, и где какую настойку задавать, давайте подробно разберем, как его (X-сервер) настроить, редактируя файл /etc/X11/XF86Config. Точнее, нашей целью пока что будет только получение на экране серого прямоугольника с крестиком-курсором посередине, а процедура запуска графической среды в целом будет рассматриваться позже.

Как было оказано выше, создать первоначальную версию файла /etc/X11/XF86Config можно с помощью программы Xconfigurator в 3-й версии XFree86 или командой:

```
[root]# /usr/X11R6/bin/XFree86 -configure
```

в 4-й версии XFree86. В последнем случае создается конфигурационный файл /root/XF86Config.new. Его можно опробовать с помощью команды

```
[root]# XFree86 -xf86config /root/XF86Config.new
```

явно указывая X-серверу, какой конфигурационный файл использовать. Но проще сразу перенести его в каталог /etc/X11 под именем XF86Config-4 и работать стандартным образом.

Итак, вы имеете какой-то вариант файла /etc/X11/XF86Config-4. Попробуйте просто запустить графический режим командой x (еще раз напомню, что это просто ссылка на сервер и если команда не работает, то ссылка должна быть создана, либо нужно вместо x указывать имя используемого сервера, может быть, даже с указанием полного пути). Если вам удалось запустить

X-сервер, и вы увидели курсор посредине экрана, то, значит, вы можете пропустить оставшийся текст данного подраздела и перейти к *разд. 7.4.*

Если вы увидели черный экран или какие-то мелькающие полоски, то вернитесь в текстовый режим, нажав комбинацию <Ctrl>+<Alt>+<Backspace> (эта комбинация может выручить вас в случае многих затруднений с графическим режимом, так что рекомендую запомнить, что она позволяет завершить работу X-сервера, а, значит, и графического режима в целом). Однако не слишком торопитесь воспользоваться этой магической комбинацией, поскольку переход в графический режим требует определенного времени; я вначале несколько раз прерывал загрузку комбинацией клавиш <Ctrl>+<Alt>+<Backspace> и не мог понять, почему у меня графика не запускается, хотя вся причина заключалась в моей поспешности.

Однако, поскольку существует опасность повредить монитору неправильным заданием параметров, лучше не заниматься экспериментами с непредсказуемыми последствиями. Целесообразней вначале протестировать тот вариант файла, который был создан, и посмотреть, нет ли в нем явных ошибок. Для этого нужно воспользоваться командой

```
[root]# X -probeonly > probe.log 2>&1
```

Эта команда запускает X-сервер в тестовом режиме и выдает протокол в файл probe.log. После запуска этой команды на экране что-то помелькает, и вновь появится командная строка оболочки, а в текущем каталоге появится файл probe.log.

В этом файле много очень полезной информации, которую надо использовать при настройке X Window (отметим, что в файле протокола /var/log/XFree86.0.log содержится еще больше информации, но нам достаточно и того, что есть в probe.log). В первых строках файла probe.log содержатся сообщения о версии XFree86 и версии ядра Linux. Затем идет сообщение о том, куда записывается протокол работы (обычно это /var/log/XFree86.0.log), и какой конфигурационный файл используется. (Помните, выше было сказано, что программа ищет этот файл в нескольких местах? Так что не лишне будет узнать, какой именно файл используется.)

```
(==} Log file: "/var/log/XFree86.0.log", Time: Mon Feb 12 17:20:25 2001
(==) Using config file: "/etc/X11/XF86Config-4"
```

Далее в probe.log идет небольшое пояснение тех служебных пометок, которые используются в этом файле в начале информационных строк:

- (--) — означает, что соответствующие значения получены путем тестирования;
- (**) — означает, что установки взяты из конфигурационного файла (в нашем случае — из /etc/X11/XF86Config);
- (++) — означает, что установки взяты из командной строки запуска;

- (==) — означает, что используются установки по умолчанию;
- (II) — за таким значком следует информационное сообщение;
- (WW) — за таким значком следует предупреждение;
- (EE) — за таким значком следует сообщение об ошибке.

Ошибки надо, естественно, постараться исправить путем соответствующей модификации файла /etc/X11/XF86Config-4. Но мы пока продолжим рассмотрение файла probe.log, в котором далее идет несколько чисто информационных строк (отметим, что в них отображена взаимосвязь секций конфигурационного файла):

```
(**) |-->Screen "Screen0" (0)
(**)  |       |-->Monitor "Monitor0"
(**)  I       |-->Device "Card0"
(**)  |-->Input Device "Mouse0"
(**)  |-->Input Device "Keyboard0"
(**) FontPath set to "unix/:-1"
(**) RgbPath set to "/usr/X11R6/lib/X11/rgb"
(**) ModulePath set to "/usr/X11R6/lib/modules"
(--) using VT number 7
```

и сообщения о том, какие модули загружаются, вроде следующего:

```
(II) Loading /usr/X11R6/lib/modules/fonts/libbitmap.a
(II) Module bitmap: vendor="The XFree86 Project"
     compiled for 4.0.2, module version = 1.0.0
```

После исправления тех ошибок, о которых сообщают строки, помеченные значком (EE), беремся за строки с предупреждениями. В частности, вы наверняка увидите строки следующего вида:

```
(WW) MGA(0): MonitorO: Using default hsync range of 28-33kHz
(WW) MGA(0): MonitorO: using default vrefresh range of 43-72Hz
```

за которыми последует множество указаний на отбраковку каких-то видеорежимов монитора:

```
(WW) MGA(0): Default mode "1856x1392" deleted (hsync out of range)
```

Строки эти являются сообщениями о тех режимах, которые X-сервер не в состоянии отображать. О важности правильного выбора частот вертикальной и горизонтальной синхронизации было уже сказано неоднократно. Тем не менее, здесь появляется повод вспомнить об этом снова. Проверьте правильность их задания еще раз.

Снова выполнив после этого команду

```
[root]# X -probeonly > probe.log 2>sl
```

и заглянув в файл probe.log, вы увидите, какие режимы для вас недоступны (обычно это режимы с высоким разрешением, типа 1600x1200).

Давайте теперь отыщем в файле probe.log две строки следующего вида (цифры у вас, конечно, могут быть другими):

```
(--) MGA(0): Virtual size is 640x480 (pitch 640)
(**) MGA(0): Default mode "640x480": 25.2 MHz, 31.5 kHz, 60.0 Hz
```

а также строку вида (она находится где-то отдельно от двух предыдущих)

```
(**) MGA(0): Depth 24, (--) framebuffer bpp 24
```

В совокупности эти три строки указывают на то, что у вас (точнее, у меня) используется глубина цвета 24 и разрешение экрана 640x480 с частотой обновления картинки 60 Гц. Такие параметры совершенно неудовлетворительны, поэтому снова обращаемся к корректировке /etc/X11/XF86Config-4. Вначале надо задать нужную глубину цвета. Естественно, побольше. Предельно допустимое значение определяется объемом видеопамяти. О том, как его рассчитать, было сказано в предыдущем подразделе.

В файле /etc/X11/XF86Config-4 для каждой глубины цвета задается своя подсекция Display в секции Screen. Выбор нужной подсекции определяется строкой вида

```
DefaultColorDepth       24
```

секции screen. Чтобы изменить разрешение экрана, например, на значение 1024x768, вставляем во все подсекции Display секции Screen строку

```
Modes "1024x768"
```

(разрешение выбираете по своему вкусу из стандартного ряда 640x350, 640x400, 640x480, 800x600, 1024x768, 1152x864, 1280x1024, 1600x1200 и т. д.). Можно ограничиться добавлением только одной такой строки в ту подсекцию Display, которая задана строкой Defaul'tDepth (ИЛИ DefaultColorDepth). После этого снова выполняем команду

```
[root]# X -probeonly > probe.log 2>&1
```

чтобы убедиться, что нет грубых ошибок, и, если таковых нет, можем рискнуть и запустить графический режим командой x.

Надо иметь в виду, что отсутствие указаний на ошибки еще не гарантирует успешный запуск графического режима. Если результатом этой команды снова будет черный экран, то выйдите из графического режима с помощью комбинации <Ctrl>+<Alt>+<Backspace> и снова внимательно проанализируйте вывод команды

```
[root]# X -probeonly > probe.log 2>&1
```

Если графический режим не запускается, попытайтесь задать меньшее разрешение или меньшую глубину цвета. Обычно таким образом удается до-

биться успешного запуска графического режима при желаемом разрешении 1024x768.

Но может быть и иначе. У меня, например, на одном из компьютеров никак не удавалось запустить графический режим с разрешением 1024x768, графическая оболочка нормально загружалась только при разрешении 800x600, а при более высоком — загрузка осуществлялась, но экран дрожал и трясся. С этой проблемой мне удалось справиться путем явного задания строки Modeline в секции Monitor и подбора указанных в ней значений параметров. Делал я это следующим образом. Вначале, не обращая внимания на мерцание и дрожание экрана, запустил графический режим и программу xvidtune (о ней будет рассказано ниже). С ее помощью (кнопка **Show)** определил, что по умолчанию используется режим, имеющий следующие параметры:

```
# 1024x768 @ 85 Hz, 68.31 kHz hsync
Modeline "1024x768" 94.5 1024 1072 1168 1376 768 769 772 808 +hsync
```

Я прописал эти две строки явным образом в секции монитор и стал менять различные параметры строки Modeline. В конце концов, удалось определить, что причиной дрожания изображения являлась слишком высокая частота тактового генератора (DotClock) — 94,5 МГц. Уменьшение ее до 94 МГц привело к стабилизации изображения. Частота обновления экрана при этом снизилась до 84,55 Гц, но это, на мой взгляд, несущественно. Впрочем, при желании ее можно даже повысить, если еще поэкспериментировать с программой xvidtune. Я же этим ограничился, а окончательную корректировку изображения провел с помощью аппаратных средств монитора.

О программе xvidtune надо рассказать отдельно. Эта программа очень полезна для окончательного выбора оптимального видеорежима. Но запустить ее можно только после того, как запустится графический режим. Выйдите в графический режим командой startx и запустите xvidtune в окне эмулятора терминала. Вы увидите вначале суровое предупреждение (рис. 7.2).

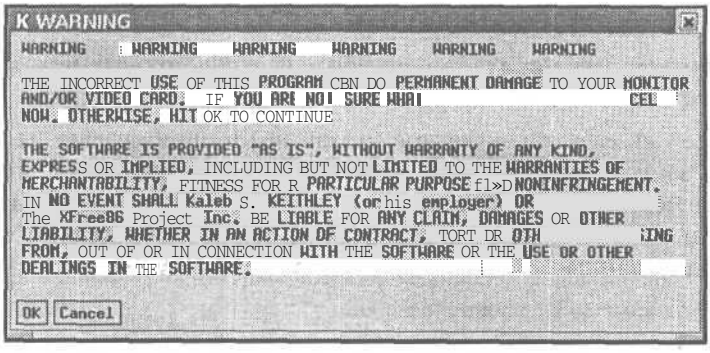


Рис. 7.2. Первое сообщение программы xvidtune

В этом предупреждении сообщается, что ни разработчик программы, ни XFree86 Project Inc. не несут никакой ответственности за последствия ее применения. И если вы не понимаете, что делаете, то лучше не беритесь! Говорят, что на мониторах старых типов неправильной установкой параметров можно добиться того, что ваш монитор начнет дымиться (за счет неправильной установки частот синхронизации, насколько я понимаю). Но я надеюсь, что вы будете достаточно осторожны и семь раз подумаете, прежде чем применить какое-то значение режима. Если так, то вперед!

Закрываем окно с предупреждением (кнопка OK) и видим основное окно программы xvidtune (рис. 7.3).

Теперь вы можете начать настраивать изображение на мониторе. Если вы хотите сместить его вправо, то щелкните мышкой по кнопке **Right** (обратите внимание на изменение некоторых цифр в окне программы), а затем по кнопке **Apply.** Вы увидите, что изображение сдвинется вправо. Аналогично можно подвинуть его влево (кнопка **Left),** вверх (Up) и вниз **(Down).** Можно также увеличить размер изображения по горизонтали **(Wider)** или вертикали **(Taller)** и, наоборот, уменьшить (соответственно **Narrower и Shorter).**

Запомните, что первоначальные установки можно всегда вернуть нажатием кнопки R или щелчком по кнопке **Restore.** Эта возможность особенно полезна в тех случаях, когда необходимо восстановить стабильный режим работы после слишком кардинальных изменений, когда изображение на экране пропадает вовсе или начинает мелькать. Кнопка **Fetch** служит для запроса текущих значений установок режима работы монитора.

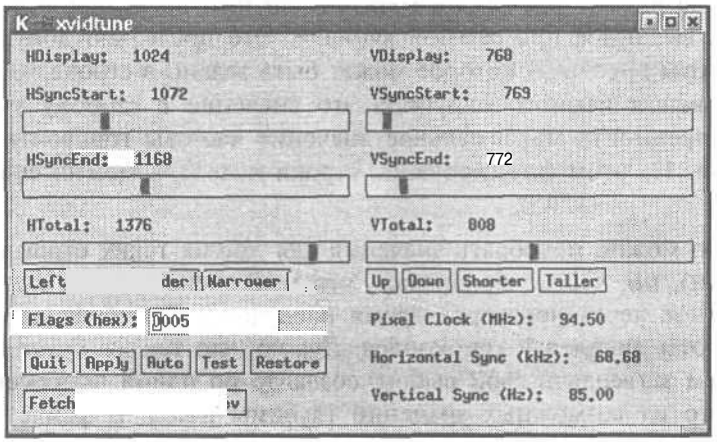


Рис. 7.3. Основное окно программы xvidtune

Кнопка **Auto** является переключателем, т. е. в режиме Auto (изображение кнопки инвертируется, т. е. отображается белыми буквами на черном фоне)

нажатие на кнопки **Up, Down, Right, Left и Wider, Narrower, Shorter, Taller** приводит к немедленному (без щелчка по кнопке **Apply)** изменению размера или положения изображения. Кнопка **Test** временно переключает установки в указанные значения. Кнопка **Show** служит для того, чтобы вывести выбранные значения на стандартный вывод (практически — в окно эмулятора терминала, из которого была запущена программа). Строка выводится в формате "Modeline", т. е. в том виде, как она должна быть записана в файле XF86Config. Кнопка **Next** переключает X-сервер в следующий видеорежим, а кнопка **Prev** — в предыдущий видеорежим (как они заданы в строке Modes).

После того как вы подобрали оптимальные с вашей точки зрения значения параметров видеорежима, запишите эти значения на бумагу. Не забудьте записать и приведенные в правом нижнем углу значения частот вертикальной и горизонтальной синхронизации. Формально они не нужны, но обычно пишутся в комментарии к строке Modeline. На всякий случай нажмите кнопку **Show** и завершите работу программы нажатием кнопки **Quit.** После этого осталось проверить, правильно ли вы записали значения, и вписать новую строку Modeline в файл XF86Config.

Теперь еще обратите внимание на то, что как бы мы не изменяли положение изображения, остается неизменным значение **Pixel Clock** в правом нижнем углу окна программы xvidtune. Этот параметр не поддается изменению в рамках программы xvidtune. Но после того, как выбраны размеры и положение изображения на экране, можно и нужно рассчитать наилучшее значение этого параметра. Принцип расчета прост: берем максимально допустимое для вашего монитора значение частоты горизонтальной синхронизации и делим его на размер фрейма по горизонтали (это последнее число в группе `HTimings`), выданное программой xvidtune. Это предельно допустимое значение частоты `DotClock`, которое может быть задано в строке Modeline. Для перестраховки я немного округляю это значение в сторону уменьшения, чтобы не превысить максимальное значение частоты горизонтальной синхронизации. На этом формирование строки Modeline можно считать завершенным.

Аналогично можно подобрать значения для других таких строк (для других разрешений), но, честно признаюсь, что мне эксперименты по изменению видеорежимов через некоторое время наскучили. Я выбрал для себя один режим работы дисплея и успокоился. Думаю, что так же будет и у вас. Но прежде, чем затвердить свой выбор, создайте по одной подсекции `Display` для каждого из возможных значений глубины цвета. В файле XF86Config имеется как минимум одна такая подсекция, соответствующая глубине цвета 8 битов. Таких подсекций в секции screen может быть несколько, поэтому не изменяйте существующую, а скопируйте ее (можно несколько раз) и подкорректируйте новые экземпляры в соответствии со своими пожеланиями. А выбор той подсекции, которая будет использоваться, осуществляется

заданием нужной глубины цвета в параметре DefauitcoiorDepht секции Screen, таким вот примерно образом: DefauitcoiorDepht 24. Пример того, как может выглядеть секция screen в итоге, был уже приведен выше, когда описывалась структура файла XF86Config, так что здесь не будем приводить его повторно.

На этом рассказ о настройке X-сервера закончен и можно перейти к вопросу о настройке графического режима в целом и как можно на этот процесс повлиять.

7.4. Запуск системы X Window

Запустить X Window можно несколькими способами. Иногда при инсталляции ОС соглашаются с предложением инсталлятора запускать их автоматически. Если при инсталляции все прошло нормально, то вы будете сразу после запуска ОС попадать в графический режим. Однако, судя по моему опыту (особенно по опыту работы с 3-й версией XFree86), система X далеко не всегда устанавливается автоматически, поэтому в разделе об инсталляции ОС я советовал отказаться от автоматического запуска. Как же запустить графическую оболочку?

Из предыдущего текста вы должны знать, что вначале необходимо запустить X-сервер. Это можно сделать, непосредственно запустив на выполнение подходящий сервер из каталога /usr/X11R6/bin. Вы уже знаете, что в результате вы увидите на экране серый прямоугольник с крестиком курсора мыши посредине. Но дальше вы вряд ли чего-нибудь добьетесь, поскольку не запущен менеджер окон и ни одной программы-клиента. Поэтому просто нажмите комбинацию клавиш <Ctrl>+<Alt>+<Backspace> для того, чтобы завершить работу X-сервера.

Несколько более успешный (но все еще не самый правильный) способ выхода в графический режим состоит в том, что вы даете команду `xinit`.

Команда `xinit` (она расположена в каталоге /usr/X11R6/bin) предназначена для запуска сервера системы X Window и хотя бы одной программы-клиента.

Если в командной строке не указано, какой именно X-сервер запускать, xinit ищет в домашнем каталоге пользователя файл .xserverrc, чтобы выполнить содержащийся в нем скрипт запуска сервера. Если такого файла нет, xinit по умолчанию выполняет следующий скрипт:

```
X :0
```

т. е. запускает программу с именем X на дисплее с номером 0. При этом предполагается, что в одном из каталогов, перечисленных в путях поиска, найдется программа с именем X. Как вы уже знаете, это должна быть ссыл-

ка на подходящий сервер. Используя скрипт .xserverrc, удостоверьтесь, что по команде exec в нем запускается существующий X-сервер. В противном случае загрузка будет происходить очень медленно и завершится немедленным выходом.

Если в командной строке запуска xinit не указана клиентская программа, которую надо запускать, программа xinit ищет в домашнем каталоге пользователя файл .xinitrc, чтобы выполнить его как скрипт, запускающий клиентские программы. Если такого файла не существует, xinit по умолчанию выполняет вместо этого скрипта команду:

```
xterm -geometry +1+1 -n login -display :0
```

Если вы после установки Red Hat Linux еще не создали свой файл .xinitrc, и просто запустите команду xinit из командной строки, вы увидите почти пустой рабочий стол с единственным окном терминала. Поскольку менеджера окон нет, вы ничего не можете сделать с этим окном (переместить, изменить размер и т. д.), но вы можете в этом окне запустить другие программы, в том числе менеджер окон. Перейдите, например, в каталог /usr/X11R6/bin и дайте команду fvwm или twm (один из этих оконных менеджеров обычно по умолчанию установлен). После этого вид экрана несколько изменится, вы сможете перемещать окна (обычным способом, захватывая мышкой заголовок окна), а по щелчку левой кнопкой по пустому полю рабочего стола получите выход в меню.

Если остановиться на таком способе вызова графического интерфейса, то каждый раз при его запуске придется повторять одну и ту же последовательность команд (не считая других минусов этого метода). Естественно, что пользователю стоит воспользоваться возможностью создания скрипта .xinitrc для автоматизации этой рутинной работы.

Ниже приведен пример скрипта .xinitrc, который запускает часы, несколько терминалов и оставляет менеджер окон в качестве "последнего" клиента.

```
#!/bin/sh
xrdb -load $HOME/.Xresources
xsetroot -solid gray &
xclock -g 50x50-0+0 -bw 0 &
xload -g 50x50-50+0 -bw 0 &
xterm -g 80x24+0+0 &
xterm -g 80x24+0-0 &
twm
```

Важно отметить, что программы, запускаемые из .xinitrc, должны запускаться в фоновом режиме, если только они не завершаются немедленно. Иначе эти программы будут препятствовать запуску других программ. Однако одна из запущенных программ (обычно менеджер окон или эмулятор терминала)

должна выполняться не в фоновом режиме, а на переднем плане, чтобы работа скрипта не завершалась (завершением работы этой программы пользователь сообщает программе xinit, что закончил работу и что сама она должна завершиться). В приведенном примере, если менеджер окон правильно сконфигурирован, то для завершения работы в X-сессии достаточно выбрать команду Exit в меню менеджера twm.

Аргументы, заданные в командной строке вызова xinit, позволяют обойти выполнение скриптов .xinitrc и .xserverrc. В командной строке может быть указана альтернативная программа-клиент и/или альтернативный сервер. Клиентская программа должна быть первым аргументом в командной строке вызова xinit. Для того чтобы вызвать конкретный X-сервер, добавьте двойное тире (после указания программы-клиента и ее аргументов), после которого укажите имя нужного сервера.

Имена программы-сервера и программы-клиента должны начинаться со слэша (/) или точки (.). В противном случае они воспринимаются как аргументы, добавляемые в командную строку вызова соответствующей (предыдущей) программы. Таким образом, можно добавлять аргументы (например, задавать цвета фона и текста), не вводя заново всю командную строку.

Если конкретное имя сервера не указано и следом за двойным тире идет двоеточие с последующей цифрой, xinit будет воспринимать это число как номер дисплея вместо предполагаемого по умолчанию нуля. Вообще все следующие за двойным тире аргументы добавляются к командной строке вызова сервера.

Вот несколько примеров командной строки вызова программы xinit.

- `[user]$ xinit`

  Этой командой будет запущен сервер, на который указывает ссылка x, и выполнен пользовательский скрипт .xinitrc, если таковой существует, а иначе просто запушена программа xterm.

- `[user]$ xinit — /usr/X11R6/bin/Xqdss :1`

  Таким образом, можно запустить какой-то конкретный сервер на альтернативном дисплее.

- `[user]$ xinit -geometry =80x65+10+10 -fn 8x13 -j -fg white -bg navy`

  По этой команде будет запущен сервер, на который указывает ссылка x, и запускаемой по умолчанию программе xterm будут переданы аргументы, перечисленные в командной строке. Скрипт .xinitrc будет проигнорирован.

- `[user]$ xinit -e widgets -- .Xsun -1 -c`

  В этом случае для запуска сервера используется команда `.Xsun -1` -c, а запускаемой по умолчанию программе-клиенту xterm будут переданы **аргументы** -e widgets.

Поскольку пользователям-новичкам обычно не хватает квалификации для создания собственного варианта скрипта .xinitrc, администраторы сайтов могут помочь им в вызове графического интерфейса, создав общедоступный скрипт, выполняющий эту функцию. Такие скрипты обычно называются x11, xstart, или startx и являются удобным способом создания простого интерфейса для пользователей-новичков. Вот пример простейшего скрипта такого вида:

```
tt!/bin/sh
xinit /usr/local/lib/site.xinitrc — /usr/X11R6/bin/X bc
```

При инсталляции стандартной версии Red Hat Linux создается более сложный вариант скрипта startx, который расположен в каталоге /usr/X11/bin (вы можете его просмотреть). Для него существует и man-страница, в которой говорится, что этот скрипт создается просто как образец для администраторов сайтов и предназначен для создания собственных вариантов такого скрипта.

Если просмотреть стандартный вариант скрипта startx, мы увидим, что практически он сводится к выполнению всего-навсего трех команд:

```
xauth add $display . $mcookie
xauth add `hostname -f`$display . $mcookie
xinit $clientargs -- $display $serverargs
```

То есть, в конечном итоге, startx вызывает уже рассмотренную нами команду xinit, только предварительно формирует нужные значения аргументов командной строки для нее. Первый аргумент — имя файла xinitrc, причем если в домашнем каталоге пользователя есть файл .xinitrc, то берется он (с указанием пути), а если в домашнем каталоге нет такого файла, то берется общесистемный файл /etc/X11/xinit/xinitrc, т. е. "clientargs" = "/etc/X11/xinit/xinitrc".

Аналогично формируется значение переменной serverargs: если существует файл .xserverrc в домашнем каталоге пользователя, то переменная serverargs будет указывать на него. Если такого файла нет, то serverargs укажет на /etc/X11/xinit/xserverrc. Переменной display присваивается значение 0. Далее в скрипте startx производится анализ аргументов, которые были заданы в командной строке при его вызове (эту часть мы пока не будем детально разбирать, поскольку для начала будем вызывать скрипт без параметров) и, наконец, в конец строки вызова xinit добавляется -auth $HOME/.Xauthority. Таким образом, сразу после установки системы (пока пользователь не создал файлов .xinitrc и .xserverrc в своем домашнем каталоге) xinit будет вызываться в следующем виде:

```
xinit /etc/X11/xinit/xinitrc — :0 /etc/X11/xinit/xserverrc -auth
$HOME/.Xauthority
```

Команда xauth и опция -auth $HOME/.Xauthority, передаваемая X-серверу, служат для авторизации пользователя, запускающего графический режим. Механизмы авторизации нас пока не интересуют, так что рассматривать эту часть не будем (если интересно, см. интерактивное руководство man с параметром Xsecurity).

7.5. Выбор и настройка менеджера окон

Если вам удалось добиться того, что X Window работает, у вас имеется масса возможностей для дальнейшей настройки. Конкретный набор этих возможностей зависит от того, какой менеджер окон вы используете. Менеджеров окон существует много, и можно организовать выбор менеджера окон на этапе загрузки графического режима. Вот пример, показывающий как можно сделать это с помощью файла .xinitrc:

```
#!/bin/sh
# $HOME/.xinitrc
usermodmap=$HOME/.Xmodmap
xmodmap $usermodmap
xset s noblank          I отключаем хранитель экрана
xset s 300 2            t запуск хранителя экрана через 5 мин.
xset m 10 5             # установка ускорения мыши
rxvt -cr green -ls -bg black -fg white -fn 7x14 \ -geometry 80x30+57+0 &
if [ "$1" = "" ] ; then WINMGR=wmaker      § по умолчанию WINMGR=wmaker
else WINMGR=$1
fi
$WINMGR
```

Хотя это и не обязательно, можно сделать этот файл исполняемым с помощью команды

```
[user]$ chmod +x .xinitrc.
```

Этот вариант .xinitrc позволяет вам выбрать менеджер окон, попробуйте, например:

```
[user]$ startx startkde
```

Если администратор хочет создать одинаковое начальное окружение для всех пользователей, можно сделать так, чтобы по умолчанию для пользователя создавался скрипт .xinitrc, который ссылается на общий стартовый скрипт:

```
#!/bin/sh
. /usr/local/lib/site.xinitrc
```

7.6. Графическая среда KDE

Каждый пользователь так или иначе формирует на своем персональном компьютере некоторую рабочую среду, которая для него наиболее удобна (точнее, которую он таковой считает, исходя из своих знаний и привычек). Можно формировать эту среду, подобрав один из оконных менеджеров (который больше всего нравится) и затем подбирая отдельные программы для выполняемой работы.

Однако имеет смысл вначале просмотреть возможности одной из разработанных в последние годы интегрированных графических сред. В английской терминологии часто используют сокращение GUI — Grafical User Interface, т. е. графический интерфейс пользователя. Будем его использовать в качестве сокращения, причем без перевода, именно как GUI. GUI представляет собой уже подобранную и проверенную совокупность программ для работы в графическом режиме, включающую в себя и менеджер окон и набор других программ, обладающих единообразным интерфейсом. Пожалуй, можно сказать, что именно единообразие интерфейса является ключевым моментом, определяющим преимущества использования GUI вместо создания собственной среды.

Существует уже несколько графических сред, как свободно распространяемых, так и коммерческих. Из свободно распространяемых наибольшую известность приобрели KDE и GNOME. Мне лично больше нравится KDE. Возможно, это определяется тем обстоятельством, что именно эта среда была первой, с которой я начал работать, и притом вполне успешно. В то время как первая встреча с GNOME была несколько неудачной. Говорят, что в последнее время коллектив разработчиков GNOME сделал большой шаг вперед, но я пока остаюсь приверженцем KDE.

В *гл. 15* попытаюсь ближе познакомить вас с графической средой KDE. Здесь же я хочу рассказать только о том, как добиться, чтобы оболочка KDE запускалась у вас по команде startx. После инсталляции некоторых дистрибутивов (например, Black Cat Linux версии 6.02) по этой команде по умолчанию запускается GNOME. Справиться с этой проблемой очень просто. Если вы внимательно прочитали предыдущий раздел, то знаете, что startx просто вызывает команду xinit с нужными параметрами, одним из которых является файл /etc/X11/xinit/xinitrc (если подобного файла нет в вашем домашнем каталоге). А в файле /etc/Xll/xinit/xinitrc производится вызов либо файла Xclients из домашнего каталога пользователя, либо общесистемного файла /etc/X11/xinit/Xclients:

```
if [ -f $HOME/.Xclients ]; then
        exec $HOME/.Xclients
elif [ -f /etc/Xll/xinit/Xclients ]; then
        exec  /etc/Xll/xinit/Xclients
```

Просмотрев, в свою очередь, файл /etc/X11/xinit/Xclients, вы увидите, что выбор графической среды определяется тем, что прописано в файле /etc/sysconfig/desktop.

Если при установке такой файл не был создан, создайте его сами и запишите в него одно слово: KDE (создать такой файл можно командой cat > /etc/sysconfig/desktop). После перезапуска графической оболочки вы получите желаемый результат: будет запущена графическая оболочка KDE. А уж о том, как настроить ее, читайте в *гл. 15* или на русской версии сайта KDE **(http://www.kde.ru).**

7.7. Использование менеджера дисплея

Систему X Window можно запускать автоматически при включении компьютера, используя программу, которая называется менеджером дисплея (X Display Manager — xdm). В этом случае пользователь сразу видит привлекательную графическую среду, и нет необходимости специально запускать графический интерфейс командой `startx`. При этом сохраняется возможность переключиться в текстовую консоль, нажав <Ctrl>+<Alt>+<F#>, а потом вернуться обратно в графическую среду, используя комбинацию <Ctrl>+<Alt>+<F7>.

Для того чтобы запускать xdm при загрузке ОС, надо отредактировать файл /etc/inittab. В этом файле имеется строка вида

```
id:3:initdefault:
```

определяющая уровень запуска по умолчанию (об уровнях запуска можно почитать в *разд. 8.2)*. Замените эту строку строкой следующего вида:

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

Такое изменение заставляет Linux при запуске переходить на 5-й уровень. А в конце того же файла /etc/inittab обычно прописана строка

```
x:5:respawn:/usr/bin/X11/xdm -nodaemon
```

которая означает, что на этом уровне запуска должен запускаться менеджер дисплея xdm.

Если вы решили запускать xdm при старте и хотите использовать, например, глубину цвета 24 бита на пиксел вместо применяемой по умолчанию 8 bpp (и ваша видеокарта и монитор поддерживают ее), вы должны изменить файл /etc/Xl 1/xdm/Xservers (в нем всего одна строка) следующим образом:

```
:0 local /usr/X11R6/bin/X -bpp 24
```

Если вы установили KDE, то вместо xdm, вероятно, запускается kdm. После установки Black Cat Linux, например, строка в /etc/inittab, определяющая менеджер дисплея, имеет вид:

```
x:5:respawn:/etc/X11/prefdm -nodaemon,
```

а /etc/Xl 1/prefdm есть ссылка на /usr/bin/kdm.

Очень важное примечание

Имейте в виду, что команда respawn в только что приведенной строке означает, что при попытках перезапуска системы будет происходить перезапуск менеджера дисплея. В частности, нажатие "магической" комбинации <Ctrl>+ +<Alt>+<Del> будет повторно запускать систему в той же конфигурации. Поэтому если вы после установки xdm будете как-то менять системные настройки и в результате ошибочных действий нарушите хрупкое равновесие системы X Window, вы попадете в очень затруднительную ситуацию. У меня, например, после одного из опытов с редактированием файла /etc/X11/XF86Config экран после загрузки стал черным и дальше компьютер ни на что не реагировал, кроме <Ctrl>+<Alt>+<Del>, а по этой комбинации происходила просто перезагрузка и выход в ту же ситуацию. На этот случай надо помнить, как перевести систему в однопользовательский режим, а прочитать об этом вы можете в следующей главе (точнее, *в разд.* 8.2).

Глава 8

Основы администрирования системы

Поскольку мы с самого начала считаем, что речь в книге идет о персональном компьютере, надо уделить некоторое внимание задачам администрирования системы. Ведь у вас не будет системного администратора, к которому можно обратиться, столкнувшись с какой-нибудь проблемой. Хочется только с самого начала напомнить, что, в большинстве случаев, для конфигурирования системы необходимо иметь права суперпользователя root.

И еще одно предварительное замечание, которое поможет вам легче понять и освоить принципы администрирования Linux: любые настройки этой ОС могут быть выполнены путем редактирования файлов сценариев (или скриптов) и конфигурационных файлов, которые читаются скриптами. Причем и те, и другие (то есть и скрипты, и конфигурационные файлы) являются простыми текстовыми файлами. Конечно, в Linux существуют различные специальные утилиты конфигурирования и администрирования системы (типа linuxconf или printtool), однако результаты работы этих программ все равно записываются в тех же конфигурационных файлах. Образно выражаясь, про Linux (и UNIX вообще) можно сказать "это почти целиком обработчик текста". Если с самого начала помнить об этой особенности, можно значительно легче освоить вопросы системного администрирования Linux.

Кстати, если вы хотите облегчить себе работу по редактированию конфигурационных файлов, сразу после инсталляции ОС Linux установите программу Midnight Commander. Это существенно облегчит вам поиск и редактирование конфигурационных файлов, т. к. можно будет пользоваться встроенным редактором этой программы (не говоря уж о том, что поиск нужного файла тоже сильно облегчается).

8.1. Основные задачи системного администрирования. Процессы и их идентификаторы

К обязанностям системного администратора обычно относят следующие задачи:

- подключение и настройка аппаратных устройств;
- установка и обновление программного обеспечения;

- ❒ запуск и настройка общесистемных сервисов (конфигурирование системы);
- О управление пользователями;
- ❒ управление процессами;
- ❒ распределение ресурсов;
- О обеспечение безопасности.

Вопросы подключения и настройки аппаратных средств, а также процедуры установки и обновления программного обеспечения мы рассмотрим в двух последующих главах. Остальные задачи системного администрирования будут кратко рассмотрены в настоящем разделе. Начнем с рассмотрения того, как происходит процесс загрузки ОС. Дело в том, что этот этап во многом определяет режим последующей работы системы и ее конфигурацию. Если вы умеете влиять на процесс загрузки, значит, вы уже сможете добиться желаемой конфигурации системы после загрузки.

Но для понимания процедуры начальной загрузки необходимо иметь самое общее представление о том, что такое процесс в системе, поскольку это понятие будет постоянно использоваться в дальнейшем.

В самом первом приближении можно считать, что процесс — это загруженная в оперативную память программа. Но это не совсем точно, правильнее было бы сказать, что "процесс выполняет программу". Дело в том, что в Linux вначале запускается процесс, который загружает в оперативную память программу из указанного ему файла и начинает ее выполнять. Это означает, что каждый процесс должен быть запущен (как говорят – "порожден") каким-то другим процессом. То есть для каждого процесса однозначно определен его "родитель" (или "предок"), для которого данный процесс является "дочерним" (или "потомком"). Если вы хотите увидеть "дерево" запущенных в вашей системе процессов, выполните команду pstree. Вывод этой команды позволяет увидеть, что "отцом" всех процессов в системе (или "корнем дерева процессов") является процесс init, который первым запускается после загрузки ядра.

Каждый процесс в системе имеет уникальный идентификатор — PID, назначаемый процессу при запуске. Процесс с идентификатором 1 выполняет программу init. Именно по этим идентификаторам система различает процессы. Каждый запущенный процесс в любой момент времени находится в одном из следующих состояний: активен (R), приостановлен (T) или "спит" *(S)*. Текущее состояние процесса называют статусом процесса.

Кроме идентификатора и статуса для каждого процесса в специальных структурах ядра сохраняются следующие данные (приводимый ниже перечень является далеко не полным):

- О полная командная строка запуска выполняемой процессом задачи;
- ❒ информация об отведенном процессу адресном пространстве;

- ❒ ссылка на текущий рабочий каталог и корневой каталог процесса (последний служит для ограничения доступа процесса к файловой структуре);
- ❒ таблица открытых процессом файлов;
- О так называемое окружение процесса, т. е. перечень заданных для данного процесса переменных с их текущими значениями;
- ❒ атрибуты, определяющие права и привилегии процесса;
- ❒ таблица обработчиков сигналов;
- О указание на родительский процесс;
- П пользовательская маска (umask) или маска доступа — указание на то, какие права надо удалить при создании нового файла или каталога из стандартного набора прав, присваиваемых файлу (каталогу).

Поскольку Linux — система многозадачная, одновременно может быть запущено много процессов. Впрочем, слово "одновременно" здесь применено не совсем корректно, поскольку на самом деле в каждый момент времени выполняется только один процесс. *(Для точности следует заметить, что в многопроцессорных системах, на которых Linux тоже может работать, одновременно могут выполняться несколько процессов, но мы рассматриваем только однопроцессорные системы).* Планировщик процессов выделяет каждому процессу небольшой квант времени и по истечении этого кванта передает управление следующему процессу. Кванты времени, выделяемые каждому процессу, так малы, что у пользователя создается иллюзия одновременного выполнения многих процессов. А для того, чтобы некоторые, наиболее важные процессы, получали больше процессорного времени, для каждого процесса установлен приоритет.

Пользователи могут "общаться" с процессами путем посылки им сигналов. Процессы тоже общаются друг с другом посредством сигналов. Когда мы нажимаем комбинацию клавиш <Ctrl>+<C>, чтобы завершить выполнение какой-то программы, мы фактически посылаем соответствующему процессу сигнал "Завершить работу". Завершаясь, процесс посылает родительскому процессу сигнал о своем завершении. Но бывают случаи, когда родительский процесс завершается раньше дочернего. Процессы, не имеющие родителя, называются "сиротами". "Сироты" автоматически усыновляются процессом init, который и принимает сигналы об их завершении. Если процесс-родитель по каким-то причинам не может принять сигнал о завершении дочернего процесса, то процесс-потомок превращается в "зомби" и получает статус Z. Процессы-зомби не занимают процессорного времени (то есть их выполнение прекращается), но соответствующие им структуры ядра не освобождаются. Уничтожение таких процессов — одна из обязанностей системного администратора. Наконец, процесс может надолго "впасть в сон", прервать который не удается. Статус таких процессов обозначается символом D. Уничтожить их удается только при перезагрузке системы.

Особым видом процессов являются демоны. Вообще-то в них нет ничего особого. Это просто процессы, выполняющиеся в фоновом режиме, без вывода каких-либо данных на терминал. Демоны обычно используются для выполнения сервисных функций, обслуживания запросов от других процессов, причем не обязательно выполняющихся на данном компьютере.

Надо еще упомянуть, что процессы могут запускать ("внутри себя") отдельные нити (thread), или потоки. Нити — это параллельно выполняемые части одной программы, которые в Linux реализованы как процессы, запускаемые со специальным флагом. С точки зрения системы они отличаются от других процессов только тем, что для них не создается отдельное окружение, они выполняются в среде родительского процесса.

Приведенных данных о процессах нам пока достаточно (к рассмотрению процессов мы еще вернемся в *разд. 8.4)*.

8.2. Процедура загрузки ОС Linux

Для начала надо отметить, что все, о чем будет рассказано в этом разделе, относится к дистрибутиву Red Hat и его аналогам. В других дистрибутивах (например, Debian) процедуры загрузки могут быть организованы иначе

8.2.1. Процесс init и файл /etc/inittab

Как вы знаете, после включения питания компьютера и завершения тестирования аппаратной части BIOS считывает из первого сектора загрузочного диска короткую программу-загрузчик. Эта программа запускает основной системный загрузчик (например, LILO), который, в свою очередь, загружает в память ядро системы, которое обычно хранится в файле vmlinuz-x.y.z-a в каталоге /boot. Здесь x.y.z — это номер версии ядра, а вместо символа "a" часто стоит указание на какие-то конкретные модификации ядра. Впрочем, название файла ядра может быть и другим, для загрузчика это не имеет значения, только это имя надо указать в конфигурационном файле загрузчика.

Сразу после зафузки ядро монтирует корневую файловую систему и запускает процесс init. Процесс init — это профамма, которая ответственна за продолжение процедуры зафузки, и перевод системы от начального состояния, возникающего после зафузки ядра, в стандартное состояние обработки запросов многих пользователей. Процесс init выполняет еще массу различных операций, необходимых для дальнейшей работы системы: проверку и монтирование файловых систем, запуск различных служб (демонов), запуск процедур регистрации, оболочек пользователей на различных терминалах и т. д.

Точный список этих операций зависит от так называемого уровня выполнения (run level). Уровень выполнения определяет перечень действий, выполняемых процессом init, и состояние системы после зафузки, т. е. конфигу-

рацию запущенных процессов. Уровень выполнения идентифицируется одним символом. В ОС Linux существует 8 основных уровней выполнения:

- 0 — остановка системы;
- 1 — однопользовательский режим (для специальных случаев администрирования);
- 2 — многопользовательский режим без NFS (то же, что и 3, если компьютер не работает с сетью);
- 3 — полный многопользовательский режим;
- 4 — использование не регламентировано;
- 5 — обычно используется для запуска системы в графическом режиме;
- 6 — перезагрузка системы;
- S (или s) — примерно то же, что и однопользовательский режим, но S и s используются в основном в скриптах.

Как видите, уровни О, 1 и 6 зарезервированы для особых случаев. Относительно того, как использовать уровни со 2 по 5, единого мнения не существует. Некоторые системные администраторы используют разные уровни для того, чтобы задать разные варианты работы, например, на одном уровне запускается графический режим, на другом работают в сети и т. д. Вы можете сами решить, как использовать разные уровни для создания разных вариантов загрузки. Но для начала проще всего воспользоваться тем способом определения разных уровней, который был задан при установке.

Первым делом после старта процесс init считывает свой конфигурационный файл /etc/inittab. Этот файл состоит из отдельных строк. Если строка начинается со знака # или пуста, то она игнорируется. Все остальные строки состоят из 4 полей, разделенных двоеточиями:

```
id:runlevels:action:process
```

где:

- id — идентификатор строки. Это произвольная комбинация, содержащая от 1 до 4 символов. В файле inittab не может быть двух строк с одинаковыми идентификаторами;
- runlevels — уровни выполнения, на которых эта строка будет задействована. Уровни задаются цифрами или буквами без разделителей, например 345;
- process — процесс, который должен запускаться на указанных уровнях. Другими словами в этом поле указывается имя программы, вызываемой при переходе на указанные уровни выполнения;
- action — действие.

В поле action стоит ключевое слово, которое определяет дополнительные условия выполнения команды, заданной полем process. Допустимые значения поля action:

- respawn — перезапустить процесс в случае завершения его работы;
- once — выполнить процесс только один раз при переходе на указанный уровень;
- wait — процесс будет запущен один раз при переходе на указанный уровень и init будет ожидать завершения работы этого процесса, прежде, чем продолжать работу;
- sysinit — это ключевое слово обозначает действия, выполняемые в процессе загрузки системы независимо от уровня выполнения (поле id игнорируется). Процессы, помеченные этим словом, запускаются до процессов, помеченных словами boot и bootwait;
- boot — процесс будет запущен на этапе загрузки системы независимо от уровня выполнения;
- bootwait — процесс будет запущен на этапе загрузки системы независимо от уровня выполнения, и init будет дожидаться его завершения;
- initdefault — строка, в которой это слово стоит в поле action, определяет уровень выполнения, на который система переходит по умолчанию. Поле process в этой строке игнорируется. Если уровень выполнения, используемый по умолчанию, не задан, то процесс init будет ждать, пока пользователь, запускающий систему, не введет его с консоли;
- `off` — игнорировать данный элемент;
- powerwait — позволяет процессу init остановить систему, когда пропало питание. Использование этого слова предполагает, что имеется источник бесперебойного питания (UPS) и программное обеспечение, которое отслеживает состояние UPS и информирует init о том, что питание отключилось;
- `ctrlaltdel` — разрешает init перезагрузить систему, когда пользователь нажимает комбинацию клавиш <Ctrl>+<Alt>+<Del> на клавиатуре. Обратите внимание на то, что системный администратор может определить действия по комбинации клавиш <Ctrl>+<Alt>+<Del>, например игнорировать нажатие этой комбинации (что вполне разумно в системе, где много пользователей).

Этот список не является исчерпывающим. Более подробно о файле inittab можно узнать из man-страниц init (8), inittab (5) и getty (8).

Обработка файла /etc/inittab процессом init начинается в однопользовательском режиме (уровень 1), в котором единственным пользователем является пользователь root, работающий с консоли. Первым делом init на-

ходит строку, которая определяет, какой уровень выполнения запускается по умолчанию:

```
id:3:initdefault:
```

Это и будет тот уровень, в котором запустится и будет работать система после загрузки, поэтому естественно, что нельзя указывать в строке `initdefault` уровни О И 6.

Далее init выполняет команды, указанные в строке с ключевым словом sysinit. В стандартной конфигурации здесь выполняется скрипт rc.sysinit из каталога /etc/rc.d. После этого процесс init просматривает файл /etc/inittab и выполняет скрипты, соответствующие однопользовательскому уровню (1 во втором поле строки), всем уровням (строки с пустым вторым полем) и уровню, заданному по умолчанию. В строке, соответствующей уровню по умолчанию, вызывается скрипт rc из каталога /etc/rc.d. Этот скрипт один и тот же для всех уровней (то есть обязательно вызывается, на какой бы уровень выполнения не загружалась система), только в зависимости от уровня выполнения ему передается соответствующее значение параметра вызова, так что, например, для 3-го уровня вызов скрипта осуществляется строкой типа

```
13:3:wait:/etc/rc.d/rc 3
```

Функции, выполняемые скриптами rc.sysinit и rc мы подробно рассмотрим ниже, в *разд. 8.2.2,* а сейчас вернемся к краткому обзору действий процесса init.

Следующая важная функция, которую выполняет этот процесс (на уровнях со 2 по 5) — запуск шести виртуальных консолей (процессов getty), чтобы предоставить пользователям возможность регистрироваться в системе с терминалов. Для этого init порождает процессы, именуемые getty-процессами (от "get tty" — получить терминал), и следит за тем, какой из процессов открывает какой терминал. Каждый getty-процесс устанавливает свою группу процессов, используя вызов системной функции setpgrp, открывает отдельную терминальную линию и обычно приостанавливается во время выполнения функции open до тех пор, пока машина не получит аппаратную связь с терминалом. Когда функция open возвращает управление, getty-процесс исполняет программу login (регистрации в системе), которая требует от пользователей, чтобы они идентифицировали себя указанием регистрационного имени и пароля. Если пользователь зарегистрировался успешно, программа login, наконец, запускает командный процессор shell и пользователь приступает к работе. Этот вызов shell именуется "login shell" (регистрационный shell, регистрационный интерпретатор команд). Процесс, связанный с shell, имеет тот же идентификатор, что и начальный getty-процесс, поэтому login shell является процессом, возглавляющим группу процессов.

Если пользователь не смог успешно зарегистрироваться, программа регистрации завершается через определенный промежуток времени, закрывая открытую терминальную линию, а процесс init порождает для этой линии сле-

дующий getty-процесс, открывающий терминал, вместо прекратившего существование.

После завершения зафузки init продолжает работать в фоновом режиме, отслеживая изменения в состоянии системы. Например, если будет подана команда telinit, позволяющая изменить уровень выполнения, процесс init обеспечит выполнение команд, заданных для нового уровня файлом /etc/inittab. Этот файл прочитывается заново и в случае поступления сигнала HUP; эта особенность избавляет от необходимости перезагружать систему для того, чтобы сделать изменения в начальной конфигурации.

Таким образом, процесс начальной зафузки init постоянно находится в оперативной памяти и при получении соответствующих сигналов повторно выполняет цикл чтения из файла /etc/inittab инструкций о том, что нужно делать, причем этот набор инсфукций различен для разных уровней выполнения.

Когда суперпользователь останавливает систему (командой shutdown), именно init завершает все другие исполняющиеся процессы, размонтирует все файловые системы и останавливает процессор.

Замечание

В приведенном описании опущены многие важные детали. Более подробное описание можно найти в man-страницах по init (8), inittab (5) и getty (8), а также в документах "Linux Documentation Project's Serial HOWTO".

Замечание

Если вы некорректно модифицируете файл /etc/inittab, система может перестать загружаться. Так что перед внесением каких-либо изменений в этот файл по меньшей мере запаситесь загрузочной дискетой и сохраните копию исходного файла на случай фатальных ошибок.

8.2.2. Основные конфигурационные файлы

Если вы прочитали *разд. 8.2.1* (или если смотрели файл /etc/inittab), то представляете, что в обычной ситуации процесс init помимо запуска процессов getty выполняет 2 основных действия:

- запускает скрипт rc.sysinit из каталога /etc/rc.d;
- запускает скрипт rc из того же каталога /etc/rc.d с опцией, равной уровню выполнения (обычно rc 3).

В файле rc.sysinit содержатся команды инициализации системы, в том числе команды установки системных переменных, зафузки таблиц раскладки клавиатуры (командой loadkeys) И системного шрифта (командой consolechars), монтирования и проверки файловых систем, зафузки модулей, задания предпочитаемой фафической оболочки и т. д.

Если вы внимательно прочитали раздел о командном языке интерпретатора команд shell *(см. гл. 5),* то вы легко поймете большую часть скрипта /etc/rc.d/rc.sysinit.

Прежде чем рассматривать функции, выполняемые скриптом rc, надо сказать несколько слов о каталоге /etc/rc.d. Этот каталог вообще играет важную роль в процессе загрузки, поскольку он содержит основные скрипты (программы на языке командного процессора shell), служащие для организации процесса загрузки.

Каталог rc.d содержит следующий набор подкаталогов:

- rc0.d
- rc1.d
- rc2.d
- rc3.d
- rc4.d
- rc5.d
- rc6.d
- init.d

Если вы просмотрите (например, с помощью команды `ls -l`) содержимое подкаталогов rcX.d, то увидите, что в этих подкаталогах содержатся не файлы, а только ссылки на файлы скриптов, находящиеся в других каталогах, а именно (за редким исключением), в каталоге /etc/rc.d/init.d. Названия этих ссылок имеют имена, начинающиеся либо с буквы K, либо с буквы S. Подкаталог init.d содержит по одному скрипту для каждой из возможных в системе служб (NFS, sendmail, httpd и т. п.).

Теперь вспомним, что процесс init после скрипта rc.sysinit запускает скрипт rc с опцией, равной заданному уровню выполнения. Этот скрипт предназначен в общем случае для перевода системы из одного уровня выполнения на другой. В процессе начальной загрузки этот скрипт переводит систему из однопользовательского режима на уровень, задаваемый по умолчанию. Общий алгоритм работы rc состоит в следующем. При переходе на уровень *X* сначала просматривается каталог *rcX.d* и для всех ссылок, которые начинаются на K, вызываются файлы, на которые идет ссылка, с опцией stop, т. е. осуществляется останов соответствующих служб (которые не должны работать на данном уровне выполнения). Затем запускаются службы, которые на данном уровне выполнения должны быть запущены. Это осуществляется путем последовательного просмотра ссылок, которые начинаются с символа S, и запуска соответствующих скриптов с опцией start. Из сказанного ясно, что буквы (символы) S и K, с которых начинаются имена ссылок в подкаталогах rcX.d, происходят от слов start и kill, соответственно. Заметим еще, что после S и K в именах ссылок стоят двузначные номера, которые служат для задания порядка запуска скриптов.

Одной из последних ссылок вида SXXname, используемых скриптом rc на уровнях 2—5, является ссылка на скрипт /etc/rc.d/rc.local. Как сказано в самом этом файле, этот скрипт выполняется после всех других скриптов в процессе инициализации системы, поэтому если вы хотите, чтобы в процессе загрузки были выполнены какие-то дополнительные команды или ваши персональные настройки, то их целесообразно поместить именно сюда.

Тот вариант этого скрипта, который устанавливается из дистрибутива, выполняет очень ограниченные задачи: выводит на экран логотип дистрибутива и формирует файлы /etc/issue и /etc/issue.net, содержащие текст сообщений, выдаваемых пользователю при входе в систему.

8.2.3. Другие файлы, влияющие на процесс загрузки

Кроме файлов /etc/inittab, /etc/rc.d/rc.sysinit, /etc/rc.d/rc, /etc/rc.d/rc.local на процесс загрузки (и, следовательно, формирующуюся в результате конфигурацию системы), оказывают влияние те скрипты и отдельные программы, которые вызываются из только что перечисленных файлов, а также некоторые чисто конфигурационные файлы. Рассмотреть их все невозможно, но о некоторых необходимо упомянуть.

Все важнейшие общесистемные конфигурационные файлы расположены в каталоге /etc и его подкаталогах. Приведем краткий список с указанием на роль некоторых из этих файлов в системе и ссылки на то, где искать более подробную информацию.

- /etc/lilo.conf — файл, определяющий конфигурацию загрузчика LILO (о структуре этого файла было сказано несколько слов в *гл. 2);*
- /etc/modules.conf (или /etc/conf.modules) — файл, определяющий конфигурацию загружаемых модулей ядра (см. man-страницу по modules.conf);
- /etc/fstab — содержит информацию, необходимую для автоматического монтирования файловых систем *(см. разд. 4.8 и разд. 8.3);*
- /etc/passwd — различная регистрационная информация, включая пароли;
- /etc/profile — глобальный файл профилей — устанавливает переменную `$PATH` и другие важнейшие переменные; заглянув в него, вы увидите, что в нем вызываются все файлы из подкаталога /etc/profile.d, в частности, файл, задающий параметры локализации системы;
- /etc/bashrc — глобальный файл конфигурации bash, устанавливает синонимы (алиасы), функции, и т. п.;
- /etc/issue — содержит сообщение, выдаваемое на терминал перед входом в систему (перед запросом имени и пароля); однако редактировать этот файл с целью изменения текста сообщения не стоит, потому что сам он формируется инициализационным скриптом /etc/rc.d/rc.local;
- /etc/motd — устанавливает сообщение, выдаваемое пользователю после входа в систему (после правильного ввода пароля);
- etc/redhat-release — содержит название и номер версии дистрибутива, используется скриптом rc.local.

Перечисленные выше конфигурационные файлы оказывают влияние на процесс загрузки системы и процесс входа в систему любого пользователя. Но существуют и такие файлы, которые влияют только на процедуры входа в систему отдельного пользователя, позволяют создать для него индивидуальную рабочую среду. Такие файлы будут рассмотрены в следующем разделе.

8.2.4. Процессы, происходящие при регистрации пользователя

Последовательность событий при полной регистрации выглядит так.

1. Пользователь вводит регистрационное имя по приглашению login: процесса getty.
2. getty выполняет программу login, используя в качестве аргумента указанное имя.
3. Программа login запрашивает пароль и сверяет имя и пароль с записанными в файле /etc/passwd.
4. Программа login выводит на экран из файла /etc/motd "сообщение дня".
5. Программа login запускает интерпретатор shell, указанный в бюджете пользователя и устанавливает переменную среды TERM.
6. Интерпретатор shell выполняет соответствующие файлы запуска, после чего выводит на экран приглашение и ожидает ввода информации.

О файлах запуска надо сказать несколько слов дополнительно. В домашнем каталоге пользователя находятся несколько личных файлов конфигурации. Если таких файлов в домашнем каталоге нет, то после входа в систему будут прочитаны глобальные файлы, содержащие значения "по умолчанию". Если в качестве оболочки используется Bourne-shell, выполняется файл .profile, если C-shell — .login и .cshrc, если Korn-shell — .profile и .kshrc (мы в дальнейшем рассматриваем только случай оболочки bash).

Если вы хотите установить для себя переменные среды (PATH или другие), отличающиеся от тех, которые по умолчанию задаются для всех пользователей, или вы хотите изменить сообщение, которое будет выдаваться вам после входа в систему, или хотите, чтобы после того, как вы войдете в систему, автоматически запускалась какая-то программа, вы можете сделать это с помощью следующих файлов:

- /home/your_home/.bashrc — устанавливает ваши алиасы (то есть псевдонимы или альтернативные имена команд, удобные для упрощения ввода часто используемых команд, имеющих значительную длину из-за большого количества опций) и функции;
- /home/your_home/.bash_profile или /home/your_home/.profile — устанавливают переменные среды и запускают ваши программы.

Если такие файлы существуют (заметим, что это скрытые файлы), они будут считаны после входа в систему, и команды, записанные в них, будут выполнены.

Если вы хотите, чтобы при входе пользователя в систему выполнялся какой-то скрипт, то можно вызов этого скрипта поместить в файл ~/.profile. Это может сделать и сам пользователь.

Эти команды будут исполняться только при входе пользователя в систему. Можно, например, приветствовать каждого пользователя по имени или посылать индивидуальные сообщения:

```
if test $USER = jim; then
    echo 'Здравствуйте, уважаемый Jim!'
fi
```

8.2.5. Загрузка в однопользовательском режиме

Процесс загрузки ОС, к сожалению, не всегда происходит так, как это задумано. Бывают случаи, когда система отказывается загружаться нормальным образом. Основные причины, приводящие к такой ситуации *(см. [П10.1] приложения):*

- неисправности аппаратных средств;
- дефектные блоки на диске, в частности, блоки, в которых находится программа-загрузчик или ядро системы;
- повреждения файловых систем;
- неверно сконфигурированное ядро (например, при попытках установить самостоятельно скомпилированную или экспериментальную версию ядра);
- ошибки в сценариях запуска (появившиеся, например, из-за того, что вы внесли в эти сценарии какие-то исправления).

Первое, что надо знать пользователю в таком случае, — как войти в контакт с системой, заставить ее воспринимать команды, чтобы попытаться что-то исправить. Один из возможных вариантов действий в этом случае — попытаться запустить систему в однопользовательском режиме, т. е. с уровнем выполнения 1 *(см. разд. 8.2).*

Обычно о необходимости перехода в однопользовательский режим говорит то, что команда fsck не может автоматически восстановить файловую систему при загрузке. В таких случаях бывает необходимо запустить fsck в разделе /usr, для чего требуется, чтобы раздел был размонтирован, а этого нельзя сделать, пока не будут отключены почти все системные службы. Тут-то и требуется перейти в однопользовательский режим, в котором запускается минимум служб и сервисов системы.

Вы можете заставить процесс ink загрузить систему в однопользовательском режиме, если зададите в командной строке загрузки ядра (в ответ на при-

глашение LILO boot:) аргумент single или emergency. Точнее, в тот момент, когда на экране появится сообщение

```
LILO boot:
```

необходимо ввести

```
linux single root=/dev/hda1
```

где вместо /dev/hda1 надо, естественно, подставить имя раздела с корневой файловой системой. Эта команда подключит корневой раздел и переведет систему в однопользовательский режим. В этом режиме в системе работает только один пользователь — администратор и запускается только очень небольшое число самых необходимых системных служб (system services), включая login. (Заметим в скобках, что другим способом перевода системы в однопользовательский режим является применение команды telinit, однако в рассматриваемой ситуации, когда не проходит загрузка, воспользоваться этим способом вряд ли удастся).

Из соображений безопасности нормально сконфигурированная система при загрузке оболочки в однопользовательском режиме запросит пароль пользователя root. Это очевидно, т. к. иначе злоумышленнику было бы очень легко, задав соответствующие аргументы загрузчику LILO, войти в систему как root со всеми вытекающими отсюда последствиями. Чтобы злоумышленники не могли воспользоваться загрузкой в однопользовательском режиме для входа в систему без пароля, в соответствующую секцию файла lilo.conf должны быть добавлены две строки:

```
restricted
password=<password>
```

После выхода в оболочку вы сможете отменить те правки, которые привели к краху, или предпринять какие-то другие действия по выходу из сбойной ситуации. В книге Д. Такета и С. Барнета *(см. [П1.9] приложения)* сказано, что этот способ не работает, если корневой раздел находится на диске SCSI. Однако, возможно, это относится к старым версиям Linux, поскольку один из моих корреспондентов (Р. Сузи) уверяет, что с SCSI-дисками никаких проблем нет, и система грузится с них в любом режиме, лишь бы была доступна программа initrd. Ну, а если загрузиться в однопользовательском режиме все же не получается, можно попробовать загрузить систему с загрузочной дискеты, так что позаботьтесь о том, чтобы такая дискета у вас была.

Кстати, не дожидаясь возникновения чрезвычайных ситуаций, проверьте, как будет у вас проходить загрузка в однопользовательский режим и загрузка с аварийной дискеты. После этого, если неприятности все же возникнут, вы будете чувствовать себя значительно спокойнее.

8.3. Запуск и настройка общесистемных сервисов

Теперь, когда вы знаете, как загрузиться даже в сложной ситуации, можно приступить к экспериментам по изменению конфигурации и настроек системы. И начать изложение этих вопросов надо, конечно, с вопроса организации монтирования необходимых файловых систем.

8.3.1. Редактирование файла /etc/fstab

Файловая система — один из важнейших общесистемных сервисов. Монтирование основных файловых систем осуществляется на этапе загрузки системы. Другие (дополнительные) файловые системы монтируются командой mount, которая была рассмотрена в *разд. 4.8.* Конфигурационным файлом для команды монтирования является файл /etc/fstab, который тоже был рассмотрен в *разд. 4.8.* Поэтому здесь не будем повторяться, а ограничимся тем, что приведем несколько советов, которые позволят несколько снизить трудоемкость процедур монтирования файловых систем.

Одним из неудобств ОС Linux по сравнению с Windows является необходимость монтировать файловую систему при работе с дискетами и вообще сменными накопителями (CD-ROM, Zip фирмы Iomega и т. п.). Каждый раз при смене диска приходится заново монтировать и размонтировать файловую систему. Впрочем, и для получения доступа к некоторым разделам жесткого диска тоже необходимо выполнять команды монтирования, если только не заставить систему делать это автоматически, изменив соответствующим образом файл /etc/fstab.

Для того чтобы не повторять одинаковых действий при каждом перезапуске системы и сократить число необходимых символов, которые приходится вводить с клавиатуры при выполнении операций монтирования, целесообразно выполнить следующее. Сначала создайте точки монтирования (пустые каталоги) для каждого из устройств, или внешних файловых систем, которые вы будете периодически подключать: гибкого диска, CD-ROM, ZIP-диска, сетевых дисков, которые будут подключаться по NFS. Это можно сделать из Midnight Commander или следующими командами:

```
[root]# cd /mnt
[root]# mkdir floppy; mkdir cdrom; mkdir win; mkdir zip; mkdir server
```

Теперь отредактируйте файл /etc/fstab, добавив в него следующие строки, соответствующие тем устройствам, которые имеются в вашей системе (то, что в файле было до вас, лучше не трогать):

```
/dev/fd0        /mnt/floppy    vfat       user,noauto        0 1
/dev/cdrom      /mnt/cdrom     iso9660    ro,user,noauto     1
```

```
/dev/sda4        /mnt/zip      vfat      user,noauto,exec    0 1
/dev/hda1        /mnt/win      vfat      user,noauto         0 1
server:/export   /mnt/server   nfs       defaults
```

Редактирование файла /etc/fstab можно выполнить и с помощью программы linuxconf, о которой будет рассказано ниже (команда File systems | Access local drive).

Если правильно настроен файл /etc/fstab, то обращение к гибкому диску или дискам CD-ROM осуществляется довольно просто. В графической среде KDE чтобы смонтировать диск, надо просто щелкнуть правой кнопкой мыши по соответствующему значку и выбрать в появившемся меню команду **Монтировать.** Чтобы добиться примерно такого же эффекта в программе Midnight Commander, надо добавить в меню этой программы (файл /usr/lib/mc/mc.mnu) команды монтирования и размонтирования дисков. Вот пример таких команд для гибких дисков:

```
m  Смонтировать дискету
     mount /mnt/floppy
d  Размонтировать дискету
     umount /mnt/floppy
```

(предполагается, что в файле /etc/fstab прописана строка, определяющая устройство и тип файловой системы для /mnt/floppy). После этого смена дискеты в Midnight Commander под Linux будет ничем не сложнее аналогичного действия в программе Norton Commander или в FAR под MS Windows: для того, чтобы смонтировать дискету, достаточно будет последовательно нажать клавиши <F2> и <M>, для размонтирования — клавишы <F2> и <D>.

8.3.2. Файлы и разделы подкачки

Выше уже было сказано, что в тех случаях, когда системе Linux не хватает оперативной памяти, имеется возможность выгрузить часть исполняющихся (но временно простаивающих) программ и их данных на жесткий диск. Это называется свопированием (swaping). В Linux существует два варианта организации той области на диске, в которую осуществляется выгрузка данных из ОП:

- в виде файла подкачки (swap-файл);
- в виде отдельного раздела диска (swap partition).

Второй способ несколько эффективнее, поскольку ядру не приходится выяснять через файловую систему, где находится файл подкачки. Еще лучше, если раздел подкачки находится на отдельном жестком диске, ибо в таком случае меньше времени расходуется на перемещение считывающих головок. Однако, поскольку размер раздела подкачки не имеет смысла делать очень

большим, то нецелесообразно отводить под такой раздел отдельный диск. Иное дело, если у вас имеется диск, на котором установлена другая ОС, или диск, используемый для резервного дублирования самых важных для вас данных. В таком случае имеет смысл разместить раздел или файл подкачки на этом диске.

О том, как создавать отдельный раздел подкачки, вы могли прочитать в *гл. 2.* А файл подкачки создается с помощью команды dd:

```
[root]# dd if=/dev/zero of=/swapfile bs=lk count=size
```

где size — размер файла подкачки в килобайтах. Некоторое время назад значение параметра size должно было находиться в пределах от 40 до 131 073 и, следовательно, размер файла подкачки мог быть не более 133 890 048 байт (это чуть меньше 128 Мбайт). То же самое ограничение действовало и для разделов подкачки. Однако последние версии ядра позволяют использовать области подкачки размером до 2 Гбайт (правда, это требует выделения дополнительной памяти для ядра).

Отметим, что для создания файла подкачки нельзя использовать команду ср, поскольку этот файл должен занимать непрерывную область на диске, что не обеспечивается командой копирования.

После создания области подкачки (будь это хоть раздел, хоть файл) на ней необходимо создать соответствующую файловую систему, что делается командой mkswap следующим образом:

```
[root]# mkswap -c swapfile [size]
```

или

```
[root]# mkswap -c /dev/hdb3
```

Опция -c обеспечивает проверку указанной области на наличие плохих блоков. Если таковые обнаружены, сообщается их количество.

Linux использует страничную организацию памяти и области подкачки. Размер страницы можно указать в команде mkswap, используя опцию -p. Типичные значения, указываемые после этой опции, 4096 и 8192. Надо сказать, что команда mkswap отказывается работать с областями подкачки, размер которых менее 10 страниц, чем и объясняется то, что размер файла подкачки должен быть не менее 40 Кбайт.

Команда mkswap подготавливает область подкачки к использованию, но чтобы система могла ее использовать, необходимо эту область инициализировать (эта операция аналогична монтированию обычных файловых систем). Инициализация выполняется с помощью команды swapon. Обычно это делается во время выполнения инициализационного скрипта /etc/rc.d/rc.sysinit. Но если вы создали файл подкачки после запуска ОС, надо выполнить эту команду отдельно. Команда swapoff производит демон-

таж указанных областей подкачки. Эта команда необходима для того, чтобы ядро могло снова считать в память данные из области подкачки.

ОС Linux может одновременно работать с несколькими областями подкачки (до 8). Сколько их задействовано в системе, можно увидеть с помощью команды `swapon -s` или просмотрев файл /proc/swaps.

8.3.3. Запуск демонов

В *разд. 8.2,* посвященном описанию процесса загрузки, было сказано, что запуск системных сервисов осуществляется скриптом /etc/rc.d/rc, который вызывается с параметром, определяющим уровень запуска. В этом скрипте поочередно вызываются на выполнение все программы и скрипты, ссылки на которые содержатся в особом каталоге /etc/rc.d/rcN.d, где N — номер уровня выполнения. Ссылки в каталоге /etc/rc.d/rcN.d имеют имена KNNname и SNNname, где NN — порядковые номера, определяющие последовательность запуска скриптов, а name — имя соответствующей программы (это имя приводится, скорее всего, просто для удобства пользователей, его отсутствие ничего бы не изменило). Скритп /etc/rc.d/rc вначале последовательно (в порядке присвоенных номеров NN) вызывает программы, на которые делаются ссылки KNNname. При этом программы вызываются с аргументом stop, т. е. соответствующие службы останавливаются. Затем так же последовательно перебираются ссылки с именами SNNname и соответствующие программы вызываются с параметром start.

Давайте рассмотрим процесс запуска новой службы на примере запуска Web-сервера Apache. Я выбрал для примера этот сервис только потому, что на изолированном персональном компьютере вы его, скорее всего, не установили при инсталляции системы. Можно было бы для примера взять Samba-сервер или сервер FTP, или любую другую службу. Последовательность действий в любом случае одинакова.

Естественно, что вначале надо установить в системе соответствующий пакет. Находим на дистрибутивном диске или скачиваем из Интернета пакет (пусть это будет apache-1.3.19-3.i586.rpm). Выполняем установку

```
[root]# rpm -Uhv apache-1.3.19-3.i586.rpm
```

При необходимости предварительно устанавливаем требующиеся дополнительные пакеты. После этого надо произвести все необходимые настройки сервера. Мы здесь не рассматриваем вопросы настройки сервера Apache. Пока вы не закончили с его настройкой, можно запускать сервер "вручную", чтобы проверить, как все работает. После того как настройка сервера завершена, можно заняться организацией его автоматической загрузки при запуске системы. Для этого переходим в каталог /etc/rc.d/rcN.d и создаем там ссылку на демон httpd:

```
[root]# ln -s /usr/sbin/httpd SNNapache
```

Значение NN для этой ссылки вы можете выбрать сами (в данном случае желательно запускать этот сервер после запуска других служб, поэтому значение надо брать побольше, например, 98). После этого сервер Apache будет автоматически загружаться при старте системы. Если потом вы захотите отказаться от его автоматического запуска, просто удалите ссылку SNNapache из каталога /etc/rc.d/rcN.d.

Надо сказать, что существует специальная утилита для управления запуском сервисов (демонов) на разных уровнях выполнения. Она называется chkconfig. Если ее запустить с опцией - - list, вы получите полный список доступных сервисов, с указанием того, запускается или нет данный сервис на каждом уровне. Опции - add и - - del служат для создания или удаления соответствующей ссылки в каталоге /etc/rc.d/rcN.d:

```
[root]# /sbin/chkconfig [--add I --del ] name
```

Формат команды для запуска или остановки сервиса следующий:

```
[root]# /sbin/chkconfig [-- level levels ] name [on | off I reset]
```

Так что запуск демона apache можно осуществить командой

```
[root]# /sbin/chkconfig --level 345 httpd on
```

8.3.4. System V Init Editor ksysv

В оболочке KDE имеется очень удобная утилита для управления запускаемыми на разных уровнях службами. Она называется ksysv или System V Init Editor и запускать ее надо от имени суперпользователя, иначе она будет только показывать существующую конфигурацию служб, но не будет позволять ее менять. После запуска этой программы вы увидите окно, изображенное на рис. 8.1.

В правом нижнем углу находятся 7 переключателей, которые позволяют определить, какие уровни исполнения будут отображаться (для уменьшения размера картинки я отключил отображение некоторых уровней). В левом поле отображаются все доступные в системе сервисы. Если поместить указатель мыши на название службы, появится краткое описание данной службы. А если щелкнуть левой кнопкой по имени службы, то это же описание появляется в отдельном окне (рис. 8.2), на двух других вкладках которого (рис. 8.3 и 8.4) можно узнать некоторые характеристики запускаемой программы, владельца и группу программы, а также их права и полномочия.

На вкладке Service имеются 4 кнопки (вы можете видеть их на рис. 8.2), которые позволяют запустить службу, перезапустить ее (Restart), остановить или редактировать соответствующий скрипт. Сообщения, выдаваемые при запуске, перезапуске, остановке службы, отображаются в нижнем поле основного окна программы. В двух рядах полей, отображаемых справа вверху, и соответствующих различным уровням выполнения, показаны службы, останавливаемые и запускаемые при переходе на каждый уровень. С помощью мыши можно удалить ссылку на какой-то сервис или добавить такую ссылку.

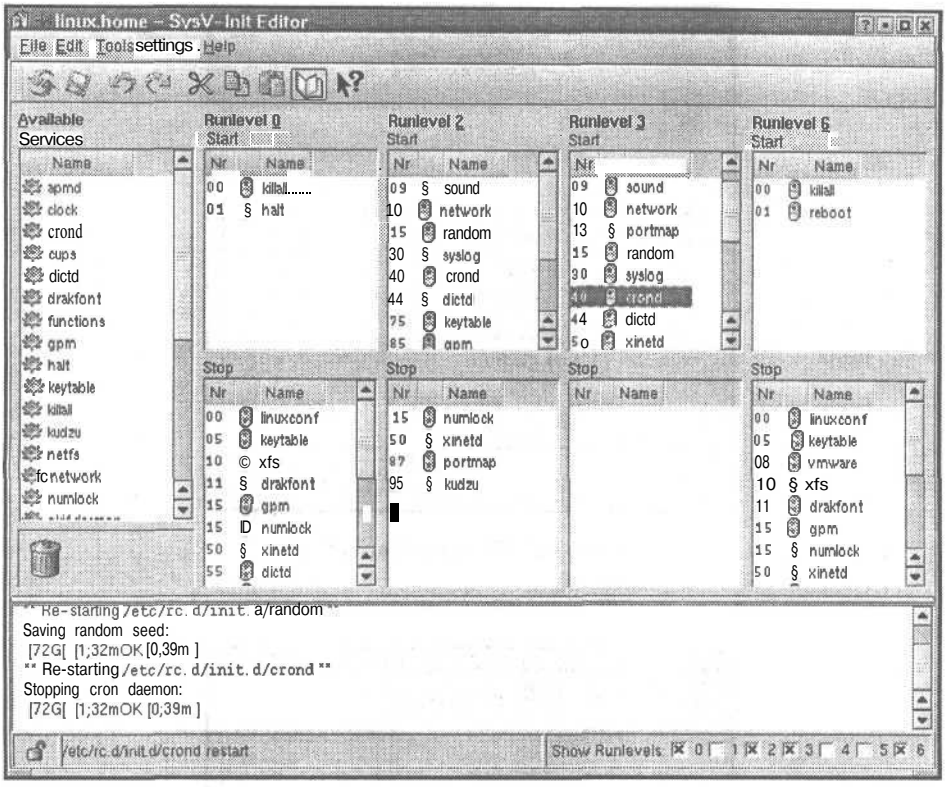


Рис. 8.1. Основное окно программы ksysv

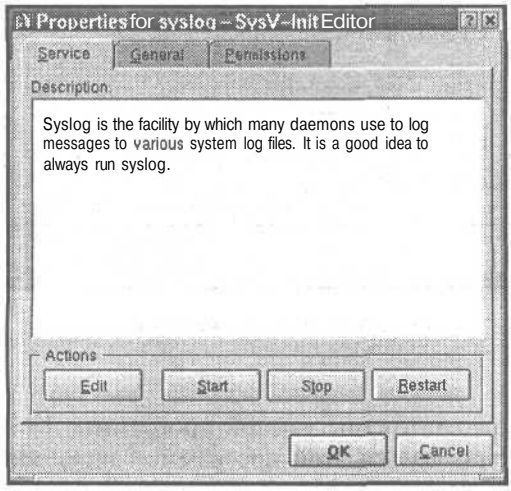


Рис. 8.2. Окно свойств службы

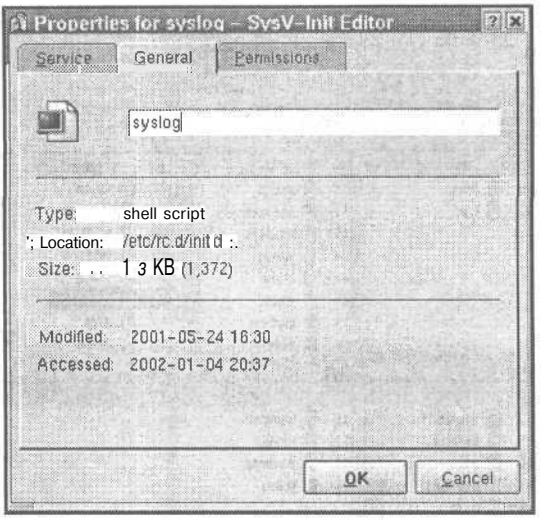


Рис. 8.3. Вкладка **General**

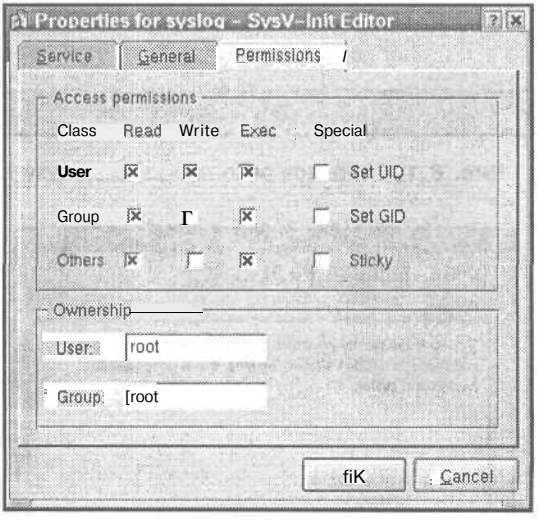


Рис. 8.4. Вкладка **Permossions**

На этом закончим рассмотрение данной программы. Конечно, не имея достаточного опыта пользоваться этой программой надо осторожно. Но, с другой стороны, откуда же опыту взяться, если его не набирать?

8.4. Управление процессами

Первым делом научимся определять, какие процессы в системе запущены. Для этого в Linux (как и во всех UNIX-системах) имеется команда ps. Если ее запустить без всяких параметров, то она выдает список процессов, запущенных в текущей сессии. Если вы хотите увидеть список всех процессов, запущенных в системе, надо задать ту же команду с параметром -ax.

8.4.1. Команда ps

Когда я заглянул в man-страницу, посвященную команде ps, я был поражен, как много у нее разных опций. Как оказалось, GNU-версия этой программы, входящая в состав Linux, поддерживает опции в стиле трех разных типов UNIX. Опции в стиле Unix98 состоят из одного или нескольких символов, перед которым(и) должен стоять дефис. Опции в стиле BSD имеют аналогичный вид, только используются без дефиса. Опции, характерные только для GNU-версии представляют собой слово, перед которым должно стоять два дефиса. Их нельзя объединять, как однобуквенные опции двух предшествующих типов. Таким образом, существует три равноправных формата задания этой команды:

```
ps [-опции]
ps [опции]
ps [--длинное_имя_опции [--длинное_имя_опции]...]
```

При этом опции разных типов нельзя употреблять в одной команде. Дадим краткую характеристику наиболее важных опций.

Первая группа опций регулирует вывод команды. Независимо от наличия опций этой группы команда ps выдает для каждого процесса отдельную строку, но содержимое этой строки может быть разным. В зависимости от заданных опций могут присутствовать следующие поля:

- USER — имя владельца процесса;
- PID — идентификатор процесса в системе;
- PPID — идентификатор родительского процесса;
- %CPU — доля времени центрального процессора (в процентах), выделенного данному процессу;
- %MEM — доля реальной памяти (в процентах), используемая данным процессом;
- vsz — виртуальный размер процесса (в килобайтах);
- RSS — размер резидентного набора (количество 1K-страниц в памяти);
- STIME — время старта процесса;

- TTY — указание на терминал, с которого запущен процесс;
- s или STAT — статус процесса;
- PRI — приоритет планирования;
- NI — значение nice (см. описание команды nice ниже);
- TIME — сколько времени центрального процессора занял данный процесс;
- CMD или COMMAND — командная строка запуска программы, выполняемой данным процессом.

Есть и другие поля, полный список которых приведен на man-странице, посвященной команде ps.

Значения, выводимые в большинстве этих полей, вы поймете без дополнительных пояснений. В поле **Статус процесса,** как уже говорилось выше, могут стоять следующие значения:

- R — выполнимый процесс, ожидающий только момента, когда планировщик задач выделит ему очередной квант времени;
- s — процесс "спит";
- D — процесс находится в состоянии подкачки на диске;
- т — остановленный процесс;
- z — процесс-зомби.

Рядом с указателем статуса могут стоять дополнительные символы из следующего набора:

- W — процесс не имеет резидентных страниц;
- < — высокоприоритетеный процесс;
- N — низкоприоритетный процесс;
- L — процесс имеет страницы, заблокированные в памяти.

Вторая группа опций регулирует то, какие именно процессы включаются в вывод команды. Чтобы получить список всех процессов, надо использовать команду ps с опциями ax или -A. Вывод в этих двух случаях отличается только в поле CMD: в первом случае выдается полная командная строка запуска программы, а во втором — только имя запущенной программы.

Описание всех опций команды ps здесь привести невозможно. Поэтому приведем только несколько примеров ее применения, которые покажут, как пользоваться этой командой в типичных ситуациях.

Для того чтобы увидеть все процессы в системе, используя стандартную форму вывода:

```
[user]$ ps -e
```

можно к той же команде добавить опцию -o, после которой указать через запятую, какие именно поля вы хотите видеть в выводе команды:

```
[user]$ ps -eo pid,user,and
```

Для того чтобы увидеть все процессы в системе, используя форму вывода BSD-систем:

```
[user]$ ps ax
```

Для того чтобы увидеть все процессы в системе, с применением графического отображения отношения "предок-потомок":

```
[user]$ ps -ef
```

Впрочем, для того, чтобы увидеть "лес" деревьев "предок-потомок", лучше воспользоваться очень интересным аналогом команды ps -ef — командой pstree.

Для того чтобы увидеть, сколько % ЦПУ и памяти занимают запущенные вами процессы:

```
[user]$ ps -u
```

Чтобы узнать приоритет процесса и значение nice, воспользуйтесь опцией `-l`:

```
[user]$ ps -l
```

8.4.2. Команда *top*

Команда ps позволяет сделать как бы "моментальный снимок" процессов, запущенных в системе. В отличие от ps команда top отображает состояние процессов и их активность "в реальном режиме времени". На рис. 8.5 изображено окно терминала, в котором запущена программа top.

Как видите, в верхней части окна отображается астрономическое время, время, прошедшее с момента запуска системы, число пользователей в системе, число запущенных процессов и число процессов, находящихся в разных состояниях, данные об использовании ЦПУ, памяти и свопа. А далее идет таблица, характеризующая отдельные процессы. Число строк, отображаемых в этой таблице, определяется размером окна: сколько строк помещается, столько и выводится. Графы таблицы обозначены так же, как поля вывода команды ps *(см. разд. 8.4.1),* так что дополнительных пояснений здесь не требуется.

Содержимое окна обновляется каждые 5 секунд. Список процессов может быть отсортирован по используемому времени ЦПУ (по умолчанию), по использованию памяти, по PID, по времени исполнения. Переключать режимы отображения можно с помощью команд, которые программа top воспринимает. Это следующие команды (просто нажимайте соответствующие клавиши, только с учетом регистра):

- <N> — сортировка по PID;
- <A> — сортировка процессов по возрасту;

- <P> — сортировка процессов по использованию ЦПУ;
- <M> — сортировка процессов по использованию памяти;
- <T> — сортировка по времени выполнения.

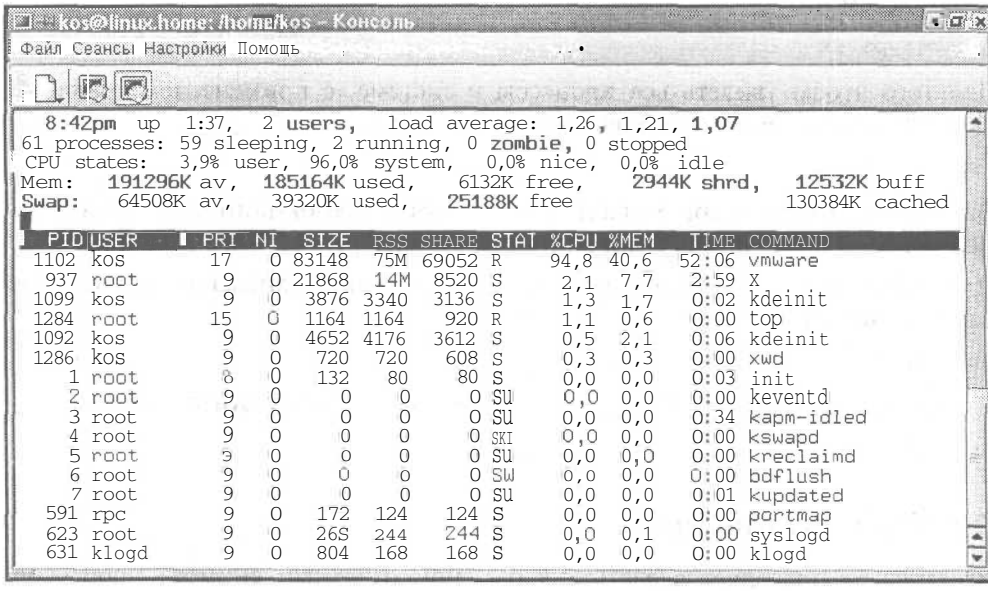


Рис. 8.5. Вывод команды top

Кроме команд, определяющих режим сортировки, команда top воспринимает еще ряд команд, которые позволяют управлять процессами в интерактивном режиме. С помощью клавиши <K> можно завершить некоторый процесс (его PID будет запрошен), а с помощью клавиши <R> можно переопределить значение nice для некоторого процесса. Таким образом, эти две команды аналогичны командам kill и renice, которые рассматриваются в *разд. 8.4.3* и *8.4.4*.

8.4.3. Приоритеты, значение nice и команда *renice*

О том, что такое приоритет, мы уже кратко говорили в начале этой главы. Но некоторые факты надо изложить дополнительно. Приоритет для каждого процесса устанавливается в тот момент, когда процесс порождается. Приоритет процесса определяется так называемым "значением nice", которое лежит в пределах от +20 (наименьший приоритет, процесс выполняется только тогда, когда ничто другое не занимает процессор), до -20 (наивысший приоритет).

Значение nice устанавливается для каждого процесса в момент порождения этого процесса и при обычном запуске команд или программ принимается равным приоритету родительского процесса. Но существует специальная команда nice, которая позволяет изменять значение nice при запуске программы. Формат использования этой программы:

```
nice [- adnice] command [args]
```

где adnice — значение (от -20 до +19), добавляемое к значению nice процесса-родителя. Полученная сумма и будет значением nice для запускаемого процесса. Отрицательные значения может устанавливать только суперпользователь. Если опция adnice не задана, то по умолчанию для процесса-потомка устанавливается значение nice, увеличенное на 10 по сравнению со значением nice родительского процесса. Очевидно, что если вы не суперпользователь, то применять эту команду имеет смысл только тогда, когда вы хотите запустить некий процесс с низким значением приоритета.

Другая команда, renice, служит для изменения значения nice для уже выполняющихся процессов. Ее формат таков:

```
renice priority [[-p] PID] [[-g] grp] [[-u] user]
```

Например, команда

```
[root]# renice -1 987 -u daemon -p 32
```

увеличивает на 1 приоритет процессов с PID 987 и 32, а также всех процессов пользователя daemon.

Суперпользователь может изменить приоритет любого процесса в системе. Другие пользователи могут изменять значение приоритета только для тех процессов, для которых данный пользователь является владельцем. При этом обычный пользователь может только уменьшить значение приоритета (увеличить значение nice), но не может увеличить приоритет, даже для возврата значения nice к значению, устанавливаемому по умолчанию. Поэтому процессы с низким приоритетом не могут породить "высокоприоритетных детей".

8.4.4. Сигналы и команда *kill*

Сигналы — это средство, с помощью которого процессам можно передать сообщения о некоторых событиях в системе. Сами процессы тоже могут генерировать сигналы, с помощью которых они передают определенные сообщения ядру и другим процессам. С помощью сигналов можно осуществлять такие акции управления процессами, как приостановка процесса, запуск приостановленного процесса, завершение работы процесса. Всего в Linux существует 63 разных сигнала, их перечень можно посмотреть по команде

```
[user]$ kill -l
```

Сигналы принято обозначать номерами или символическими именами. Все имена начинаются на SIG, но эту приставку иногда опускают: например, сигнал с номером 1 обозначают или как SIGHUP, или просто как HUP.

Когда процесс получает сигнал, то возможен один из двух вариантов развития событий. Если для данного сигнала определена подпрограмма обработки, то вызывается эта подпрограмма. В противном случае ядро выполняет от имени процесса действие, определенное по умолчанию для данного сигнала. Вызов подпрограммы обработки называется *перехватом* сигнала. Когда завершается выполнение подпрограммы обработки, процесс возобновляется с той точки, где был получен сигнал.

Можно заставить процесс игнорировать или блокировать некоторые сигналы. Игнорируемый сигнал просто отбрасывается процессом и не оказывает на него никакого влияния. Блокированный сигнал ставится в очередь на выдачу, но ядро не требует от процесса никаких действий до разблокирования сигнала. После разблокирования сигнала программа его обработки вызывается только один раз, даже если в течение периода блокировки данный сигнал поступал несколько раз.

В табл. 8.1 приведены некоторые из часто встречающихся сигналов.

Таблица 8.1. *Сигналы*

№	Имя	Описание	Можно перехватывать	Можно блокировать	Комбинация клавиш
1	HUP	Hangup. Отбой	Да	Да	
2	INT	Interrupt. В случае выполнения простых команд вызывает прекращение выполнения, в интерактивных программах — прекращение активного процесса	Да	Да	<Ctrl>+<C> или <Del>
3	QUIT	Как правило, сильнее сигнала Interrupt	Да	Да	<Ctrl>+<\>
4	ILL	Illegal Instruction. Центральный процессор столкнулся с незнакомой командой (в большинстве случаев это означает, что допущена программная ошибка). Сигнал отправляется программе, в которой возникла проблема	Да	Да	
8	FPE	Floating Point Exception. Вычислительная ошибка, например, деление на ноль	Да	Да	

Таблица 8.1 *(окончание)*

№	Имя	Описание	Можно перехватывать	Можно блокировать	Комбинация клавиш
9	KILL	Всегда прекращает выполнение процесса	Нет	Нет	
11	SEGV	Segmentation Violation. Доступ к недозволенной области памяти	Да	Да	
13	PIPE	Была предпринята попытка передачи данных с помощью конвейера или очереди FIFO, однако не существует процесса, способного принять эти данные	Да	Да	
15	TERM	Software Termination. Требование закончить процесс (программное завершение)	Да	Да	
17	CHLD	Изменение статуса порожденного процесса	Да	Да	
18	CONT	Продолжение выполнения приостановленного процесса	Да	Да	
19	STOP	Приостановка выполнения процесса	Нет	Нет	
20	TSTR	Сигнал останова, генерируемый клавиатурой. Переводит процесс в фоновый режим	Да	Да	<Ctrl>+ +<Z>

Как видите, некоторые сигналы можно сгенерировать с помощью определенных комбинаций клавиш. Но такие комбинации существуют не для всех сигналов. Зато имеется команда `kill`, которая позволяет послать заданному процессу любой сигнал. Как уже было сказано, с помощью этой команды можно получить список всех возможных сигналов, если указать опцию `-l`. Если после этой опции указать номер сигнала, то будет выдано его символическое имя, а если указать имя, то получим соответствующий номер.

Для посылки сигнала процессу (или группе процессов) можно воспользоваться командой `kill` в следующем формате:

```
[user]$ kill [-сигн] PID [PID..]
```

где сигн - это номер сигнала, причем если указание сигнала опущено, то посылается сигнал 15 (TERM — программное завершение процесса). Чаще всего используется сигнал *9* (KILL), с помощью которого суперпользователь может завершить любой процесс. Но сигнал этот очень "грубый", если можно так выразиться, поэтому его использование может привести к нарушению поряд-

ка в системе. Поэтому в большинстве случаев рекомендуется использовать сигналы TERM или QUIT, которые завершают процесс более "мягко".

Естественно, что наиболее часто команду kill вынужден применять суперпользователь. Он должен использовать ее для уничтожения процессов-зомби, зависших процессов (они показываются в листинге команды ps как <exiting>), процессов, которые занимают слишком много процессорного времени или слишком большой объем памяти и т. д. Особый случай — процессы, запущенные злоумышленником. Но обсуждение этого особого случая выходит за рамки данной книги.

8.4.5. Перевод процесса в фоновый режим

Если вы запускаете какой-то процесс путем запуска программы из командной строки, то обычно процесс запускается, как говорят, "на переднем плане". Это значит, что процесс "привязывается" к терминалу, с которого он запущен, воспринимая ввод с этого терминала и осуществляя на него вывод. Но можно запустить процесс в фоновом режиме, когда он не связан с терминалом. Для запуска процесса в фоновом режиме в конце командной строки запуска программы добавляют символ &.

В оболочке bash имеются две встроенные команды, которые служат для перевода процессов на передний план или возврата их в фоновый режим. Но прежде, чем рассказывать об этих командах, надо рассказать о команде jobs. Она всегда вызывается без аргументов и показывает задания, запущенные из текущего экземпляра shell. В начале каждой строки вывода этой команды указывается порядковый номер задания в виде числа в квадратных скобках. После номера указывается состояние процесса: stopped (остановлен), running (выполняется) или suspended (приостановлен). В конце строки указывается команда, которая исполняется данным процессом. Один из номеров выполняющихся заданий помечен знаком +, а еще один — знаком -. Процесс, помеченный знаком +, будет по умолчанию считаться аргументом команд fg или bg, если они вызываются без параметров. Процесс, помеченный знаком -, получит знак +, если только завершится по какой-либо причине процесс, который был помечен знаком +.

А теперь можно рассказать и о командах fg и bg, которые служат для перевода процессов на передний план или возврата их в фоновый режим. В качестве аргумента обеим этим командам передаются номера тех заданий, которые присутствуют в выводе команды jobs. Если аргументы отсутствуют, то подразумевается задание, помеченное знаком +. Команда fg переводит указанный в аргументе процесс на передний план, а команда bg — переводит процесс в фоновый режим. Одной командой bg можно перевести в фоновый режим сразу несколько процессов, а вот возвращать их на передний план необходимо по одному.

8.4.6. Команда *nohup*

Предположим, вы запустили из оболочки bash несколько процессов, часть из них в фоновом режиме. И по каким-то причинам завершили текущую сессию работы в оболочке. При завершении сессии оболочка посылает всем порожденным ею процессам сигнал "отбой", по которому некоторые из порожденных ею процессов могут завершиться, что не всегда желательно. Если вы хотите запустить в фоновом режиме программу, которая должна выполняться и после вашего выхода из оболочки, то ее нужно запускать с помощью утилиты nohup. Делается это так:

```
nohup команда &
```

Запускаемый таким образом процесс будет игнорировать посылаемые ему сигналы (если это возможно, см. табл. 8.1). Стандартный выходной поток и стандартный поток ошибок при таком запуске команд перенаправляются в файл nohup.out или $HOME/nohup.out.

Команда nohup имеет побочный эффект, заключающийся в том, что значение nice для запускаемого процесса увеличивается на 5, т. е. процесс выполняется с более низким приоритетом.

8.5. Управление пользователями

Задача управления пользователями имеет большое значение для истинно многопользовательских систем. Для персонального компьютера, о котором идет речь в этой книге, эта задача не так актуальна. Тем не менее, некоторые вопросы отразить необходимо, раз уж Linux по своей природе многопользовательская система.

Во-первых, еще раз следует повторить, что не стоит работать в системе от имени суперпользователя (кроме случаев выполнения сугубо административных задач). Следовательно, как минимум два пользователя у вас должны быть заведены, назовем их root и user. Но обходиться только двумя пользователями удается далеко не всегда. Даже на домашнем компьютере приходится давать доступ детям или другим членам семьи, а на служебном компьютере может потребоваться дать ограниченный доступ другим сотрудникам организации. Кроме того, в системе всегда автоматически заводится ряд пользователей для выполнения служебных задач. Поэтому задача управления актуальна и для персонального компьютера. Задача эта состоит из нескольких подзадач, а именно:

- заведение новых пользователей;
- распределение пользователей по группам;
- задание прав и полномочий для нового пользователя;

О установление для него квот;

❒ блокирование, при необходимости, бюджета пользователя.

Рассмотрим кратко хотя бы основные из этих задач.

Для начала надо отметить, что список всех известных системе пользователей находится в файле /etc/passwd. Каждая строка этого файла соответствует одному пользователю и состоит из семи полей, разделенных двоеточиями:

❒ регистрационное имя пользователя;

❒ зашифрованный пароль;

❒ UID (идентификатор пользователя);

П GID (идентификатор группы);

П информация о пользователе (обычно полное имя, должность и телефоны);

П домашний каталог пользователя;

П регистрационный shell.

С точки зрения системы поля GID и информация о пользователе не имеют никакого значения, а имя пользователя используется только на этапе входа пользователя в систему (при логировании). Дальше система идентифицирует пользователя по его UID. Идентификатор группы включается в этот файл только по традиции, потому что включение пользователей в различные группы определяется теперь файлом /etc/group. Второе поле каждой строки тоже уже не содержит зашифрованного пароля, теперь в этом поле стоит просто звездочка (*), а зашифрованные пароли переместились в файл /etc/shadow. Дело в том, что с увеличением производительности компьютеров появилась возможность определять открытый пароль по зашифрованному методом простого перебора всех возможных комбинаций символов. А поскольку файл etc/passwd доступен по чтению всем пользователям, безопасность многопользовательской системы ставится под удар. Файл же /etc/shadow доступен только суперпользователю, что уменьшает, если не снимает совсем, эту угрозу.

Уже из описания файла /etc/passwd вы могли заключить, что заведение нового пользователя не заканчивается тем, что администратор прописывает в этом файле дополнительную строку. Процесс подключения нового пользователя состоит из следующих этапов:

П занесение информации в файл /etc/passwd;

П задание исходного пароля для нового пользователя;

П создание для него домашнего каталога;

П копирование в этот каталог стандартных вариантов файлов запуска;

П установка адреса электронной почты и почтовых псевдонимов;

П включение пользователя в необходимые группы;

П установка квот и ограничений.

Конечно, можно все эти этапы выполнять и вручную, но все же проще и удобнее воспользоваться имеющимися в системе специальными программными средствами. Как уже было описано в *гл. 3,* заводить в системе нового пользователя удобнее всего командой useradd. Однако перед тем, как заводить пользователя, надо подготовить типовые файлы конфигурации, которые используются при вводе новых пользователей, и располагаются в каталоге /etc/skel. Один полезный совет: не заводите много новых пользователей, пока вы не настроили конфигурационные файлы и не поместили образцы в /etc/skel/*. И сразу после того, как вы отредактировали какой-то конфигурационный файл, скопируйте его в /etc/skel/.

Команда useradd заводит бюджет нового пользователя, создает для него домашний каталог, копирует в него файлы конфигурации из каталога /etc/skel. В качестве аргумента команде должно быть указано имя пользователя, которое потом будет использоваться им для входа в систему. Кроме того, с помощью дополнительных опций можно задать:

- данные о пользователя (имя и т. д.), записываемые в поле комментария в файле /etc/passwd (опция -с);
- имя или номер группы, к которой будет отнесен пользователь (опция -д);
- список групп, в которые будет включен данный пользователь (опция -G);
- UID пользователя, назначаемый вместо UID, задаваемого системой (опция -и);
- какая оболочка назначается пользователю (опция -s)

и еще некоторые параметры. С помощью опции -D можно изменять значения параметров, которые назначаются вновь создаваемому пользователю. Однако рассматривать эти возможности мы здесь не будем.

После ввода нового пользователя надо задать ему первоначальный пароль, что делается командой passwd login_name. После первого входа в систему пользователь должен будет поменять свой пароль с помощью той же команды (только login_name указывать ему не требуется).

Команда usermod имеет те же опции, что и useradd, только используется для изменения параметров существующего пользователя, причем на момент применения этой команды суперпользователем данный пользователь не должен быть логирован в системе.

Каждый пользователь может быть включен в произвольное число групп. Включение пользователя в различные группы может быть осуществлено путем "ручного" редактирования файла /etc/group суперпользователем, а может быть выполнено С ПОМОЩЬЮ команд groupadd И groupmod.

В процессе работы пользователь может сменить имя, с которым он вошел в систему (поработать в системе от имени другого пользователя). Как уже говорилось, для этого используется команда su. В качестве аргумента команде

в простейшем случае передается имя или идентификатор пользователя, под которым мы хотим работать в системе. При этом, если перед именем пользователя стоит дефис, то для пользователя открывается новая сессия (происходит как бы "перелогирование"), а если дефиса нет, то изменяется только имя пользователя, а окружение (например, текущий каталог) не изменяется. В любом случае выводится запрос на ввод пароля того пользователя, под чьим именем мы хотим дальше работать.

Команда `su` чаще всего используется для того, чтобы временно получить доступ к системе с правами суперпользователя. Параметр в этом случае указывать не требуется.

После того как вы поработали под чужим именем, достаточно выполнить команду exit, чтобы вернуть себе свое обычное имя.

Команда sg аналогична команде su, но используется для смены группы. Доступ предоставляется в том случае, если пользователь является членом указанной группы. В результате выполнения этой команды все вновь созданные файлы и запускаемые процессы получат новый идентификатор группы.

На этом мы ограничим рассуждения о процедурах управления пользователями, поскольку, как уже говорилось, вопросы эти не очень актуальны для персонального компьютера. Единственное, что можно еще добавить, — это то, что существуют два файла, с помощью которых можно передавать "приветы" пользователям в процессе их логирования. Это файлы:

- /etc/issue — сообщение, выдаваемое системой до приглашения "`login:`";
- /etc/motd — сообщение, выдаваемое системой после входа пользователя в систему.

Если вы хотите сделать эти сообщения более приветливыми, можете немного подкорректировать их. После корректировки можно переключиться в другую консоль и несколько раз войти в систему и выйти из нее, чтобы насладиться плодами своего труда. Но учтите, что скрипт /etc/rc.d/rc.local может перезаписывать файлы /etc/issue и /etc/motd при каждом перезапуске системы (у меня, например, Red Hat перезаписывает файл /etc/issue и не трогает /etc/motd). То есть корректировать, возможно, надо не сами эти файлы, а скрипт /etc/rc.d/rc.local.

8.6. Управление ресурсами

В этом разделе мы рассмотрим только один аспект управления ресурсами: как сэкономить тот или иной ресурс, точнее, как поступить в случае, если какого-то ресурса недостаточно. Основными ресурсами компьютера являются память и дисковое пространство. Того и другого, как известно, всегда не хватает. Вопросы экономии оперативной памяти уже вкратце рассмотрены,

поскольку мы уже рассмотрели вопросы управления процессами и swap-файлами *(см. разд. 8.3, 8.4)*. Так что осталось только рассмотреть вопрос о том, как освободить место на жестком диске.

8.6.1. Сколько осталось места на диске?

При установке новых пакетов очень часто возникает одна проблема, хорошо знакомая всем пользователям компьютеров: недостаток дискового пространства. Поэтому перед установкой нового пакета надо вначале ответить на вопрос о том, достаточно ли места на диске для размещения данного ПО?

Команда `rpm` позволяет определить, сколько места потребуется для установки пакета: для этого надо дать запрос вида `rpm -qpi имя_пакета` и в строке **Size** будет выдано, сколько байтов займет пакет. Осталось узнать, есть ли столько свободного места на диске.

Для определения объема свободного пространства на диске вы можете воспользоваться командой `df`. Если дать эту команду без аргументов, то она сообщит, каков объем дискового пространства во всех смонтированных файловых системах, сколько используется и сколько еще свободно. Единицей измерения при этом служит 1-килобайтный блок. Если вы хотите получить сведения об объеме свободного пространства в более привычных мегабайтах, дайте команду с параметром `-h`:

```
[user]$ df -h
```

Сведения о количестве свободного пространства на конкретном диске можно получить, если задать в качестве параметра имя файла устройства:

```
[user]$ df -h /dev/hda2
```

Если вместо имени файла устройства указать полное (с указанием пути) имя произвольного файла или каталога, то вы получите данные о количестве используемого и свободного места в файловой системе, содержащей указанный файл (каталог).

Если место еще есть, то можно перейти к установке пакета. Хуже владельцам компьютеров с дисками маленького объема: тут каждый раз надо думать, как бы освободить место для новой программы, другими словами, что уже можно с диска удалить. Удалять можно отдельные файлы, но, конечно, с точки зрения освобождения пространства эффективнее удалять целыми каталогами или пакетами.

Файлы (каталоги) удаляются в том случае, если они (размещенные в них файлы) вам более не нужны. Естественно, что кандидатами на удаление в первую очередь рассматриваются каталоги (или файлы) самого большого объема, и тут оказывается полезной команда du (disc usage).

Команда du позволяет узнать, сколько места занимает конкретный файл или подкаталог. Для этого надо дать команду следующего формата (в примере мы узнаем объем каталога /usr/lib)

```
[user]$ du -ks /usr/lib
```

Результатом выполнения данной команды будет примерно такая строка

```
91418 /usr/lib
```

которая означает, что каталог /usr/lib занимает 91 418 Кбайт (опция k указывает, что объем должен выдаваться в килобайтах). Опция s задана для того, чтобы выводился только суммарный объем каталога. Если вы ее опустите, то получите данные об объеме каждого подкаталога и файла в указанном каталоге, а это очень много информации. Впрочем, последней строкой все равно будет выведен суммарный объем каталога.

Если маленькую s заменить на большую *s,* то выводиться будет только информация об объеме подкаталогов (но не файлов), что иногда тоже полезно. О других опциях указанных команд вы можете узнать на соответствующих man-страницах или по команде info.

8.6.2. Освобождение дискового пространства

Теперь вы знаете, как определить, сколько места займет на диске устанавливаемый пакет и каков объем свободного пространства на диске. Рассмотрим, что можно сделать, если свободного места недостаточно. Надо заметить, что при стандартной инсталляции ОС Linux на диске образуется большое количество файлов, которые вам никогда не понадобятся. В то же время довольно трудно в огромной массе файлов найти те, которые вам не нужны (тем более, что они спрятаны глубоко в дебрях структуры каталогов). Поэтому я расскажу вкратце о своем опыте поиска ненужных файлов, надеясь, что этот опыт окажется полезен читателю.

Однажды мне пришлось устанавливать Linux (Black Cat 6.0) на 486-ой компьютер с жестким диском объемом 350 Мбайт. Хотя при установке я старался выбрать минимально возможную конфигурацию ПО, все равно после завершения установки диск оказался заполнен более чем на 90%. Перечислю вкратце те действия, которые позволили мне освободить достаточно места на диске для установки новых пакетов.

1. Первым делом стоит подумать об удалении части пакетов ПО, установленных при инсталляции системы. Для того чтобы решить, какой пакет или пакеты можно удалить, дайте команду

   ```
   [root]# rpm -qa > packages
   ```

   Полученный файл packages будет содержать список всех установленных в системе пакетов ПО. Этот список можно проанализировать с целью выяв-

ления ненужных вам пакетов и удалить таковые с помощью команды rpm с параметром -e. При этом будут удалены все созданные при инсталляции пакета файлы и каталоги. Этот способ самый безопасный, поскольку программа rpm предварительно проверит, не используется ли данный пакет какой-либо другой программой, и при наличии такой зависимости выдаст соответствующее предупреждение, а удалять что-либо откажется. Отмечу только, что удаление пакетов надо выполнять с правами пользователя root.

У меня, например, в полученном таким образом файле packages встретилось упоминание пакета AfterStep-APPS-990329-2. Выполнив команду

```
[root]# rpm -qi AfterStep-APPS
```

я выяснил, что этот пакет содержит некие апплеты к оконному менеджеру AfterStep. Поскольку такой менеджер я не использовал, я удалил этот пакет, а также и сам пакет AfterStep, с помощью команд

```
[root]# rpm -e AfterStep
[root]# rpm -e AfterStep-APPS
```

Это освободило около 5800 Кбайт. Воспользовавшись таким приемом, вы можете удалить все пакеты, которые не используете. Я, например, удалил такие пакеты, как gnome-core, gnome-libs, gnome-audio, поскольку в качестве графической среды использую не GNOME, а KDE. Отмечу, что когда я попытался удалить gnome-core, программа rpm отказалась это сделать, сообщив, что часть этого пакета используется пакетом xmms-gnome. Еще больше неудовлетворенных зависимостей образуется при попытке удалить gnome-libs. Однако поскольку все упоминавшиеся в этих сообщениях пакеты так или иначе были связаны с GNOME, я рискнул удалить их все, а также другие пакеты, которые были с ними связаны. В результате освободилось около 7800 Кбайт дискового пространства.

2. Заглянув в каталог /usr/man, я обнаружил два подкаталога со страницами руководства man на французском и испанском языках. После их удаления освободилось еще около 100 Кбайт.

3. В каталоге /usr/lib/kbd/keymaps можно, по-видимому, удалить подкаталоги, в которых хранятся таблицы раскладок клавиатуры, предназначенные для других типов микропроцессоров, а в /usr/lib/kbd/keymaps/i386 — подкаталоги для раскладок, отличных от qwerty.

   Аналогичным образом удаляются раскладки клавиатуры для различных языков, которыми вы не пользуетесь (скажем, для китайского или тайского). Эти шрифты расположены в каталоге /usr/lib/kbd/keymaps/i386/qwerty. Можно попробовать удалить ненужные шрифта из /usr/lib/kbd/consolefonts, однако если в начале таблицы раскладки клавиатуры указывается, для какого языка она служит, то о назначении файла шрифта приходится судить по его названию.

Поскольку по умолчанию в системе устанавливаются средства локализации для различных стран (которые вам, по-видимому, не потребуются), то можно удалить из каталога /usr/share/locale все подкаталоги, соответствующие ненужным вам языкам. Всего там около 16 Мбайт, так что удаление ненужного может дать около 15 Мбайт освобожденного пространства. Еще более 2 Мбайт может дать очистка каталогов /usr/share/i18n/locales и /usr/share/il8n/charmaps.

4. В каталоге /usr/share/doc/HTML/ имеются подкаталоги с документацией на разных языках, значительная часть которой вам, по-видимому, ненужна. Я оставил в этом каталоге только три подкаталога en, ru, default, причем последний является просто ссылкой на подкаталог en, так что фактически там осталось только 2 подкаталога. Удаление этой документации освободило около 500 Кбайт.

Я думаю, что, проведя более детальный анализ содержимого своего жесткого диска, вы найдете еще много файлов, которые можно безопасно удалить. В заключение хочется только сказать, что, если после такого удаления вы попытаетесь проверить с помощью той же программы rpm корректность установки некоторых пакетов, вы можете получить сообщения об ошибках. Но это не страшно, если только вы удаляли пакеты после основательных раздумий.

8.7. Программные средства для конфигурирования системы

Как вы уже, наверное, поняли, самый эффективный способ настройки системы в целом и отдельных служб состоит в редактировании конфигурационных файлов. Однако, для начинающего пользователя этот метод не самый лучший, поскольку надо иметь достаточно большой объем знаний по ОС Linux, чтобы правильно отредактировать даже простейшие из этих файлов. К счастью, в состав дистрибутива Red Hat Linux входят специальные программные средства для конфигурирования системы, существенно облегчающие выполнение этих функций. Пожалуй, наиболее часто употребляемой программой такого сорта является linuxconf (**http://www.solucorp.qc.ca/linuxconf**). Эта программа может работать как в текстовом, так и в графическом режиме.

На рис. 8.6 приводится вид окна, которое появляется при запуске программы linuxconf в графическом режиме. Первоначально правое поле окна пустое.

В левой части окна отображается древовидная структура групп конфигурируемых параметров. Если данная группа параметров содержит более мелкие подгруппы, это обозначается крестиком перед названием группы. Если щелкнуть мышкой по этому крестику, то развернется список подгрупп. Если крестик перед названием группы отсутствует, щелчок мыши по этому названию приводит к появлению в правой части окна списка параметров

данной группы. Например, группа параметров **User accounts** (Счета пользователей) выглядит так, как изображено на рис. 8.7.

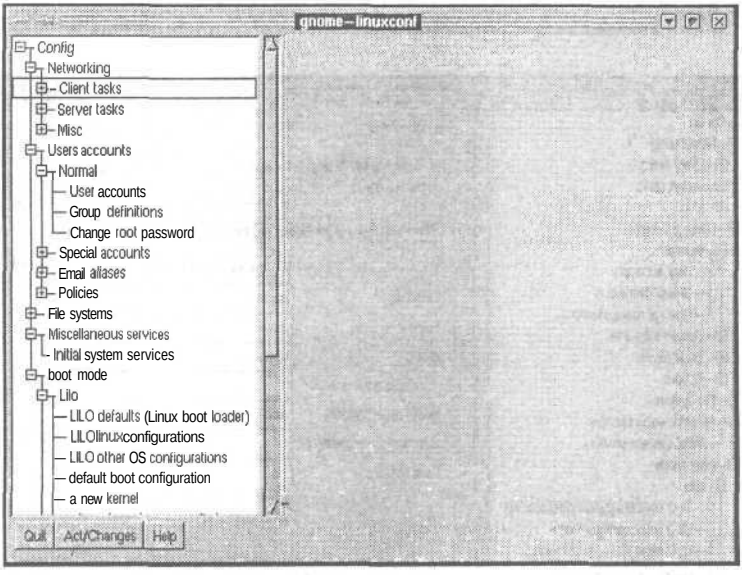


Рис. 8.6. Основное окно программы linuxconf

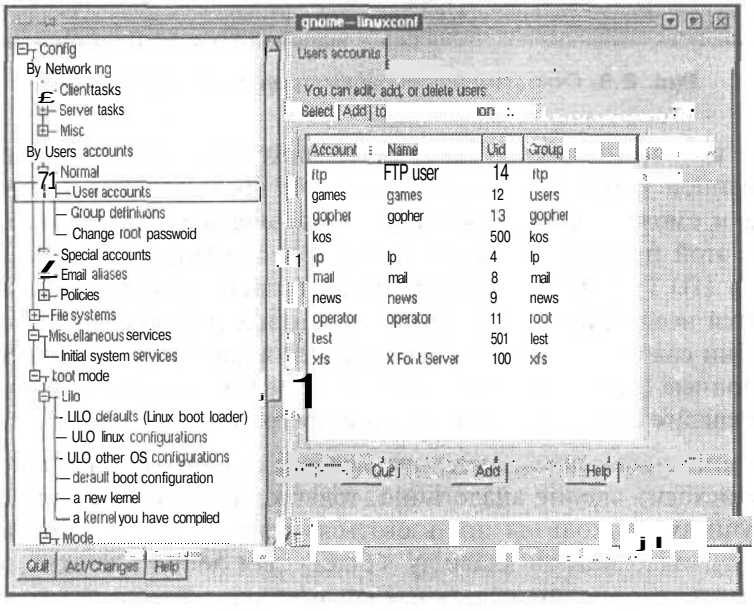


Рис. 8.7. Окно управления счетами пользователей

С помощью экранной кнопки Add можно добавить нового пользователя, а щелкнув мышкой по имени уже существующего, вы получите возможность изменять значения отдельных параметров счета данного пользователя (см. рис. 8.8).

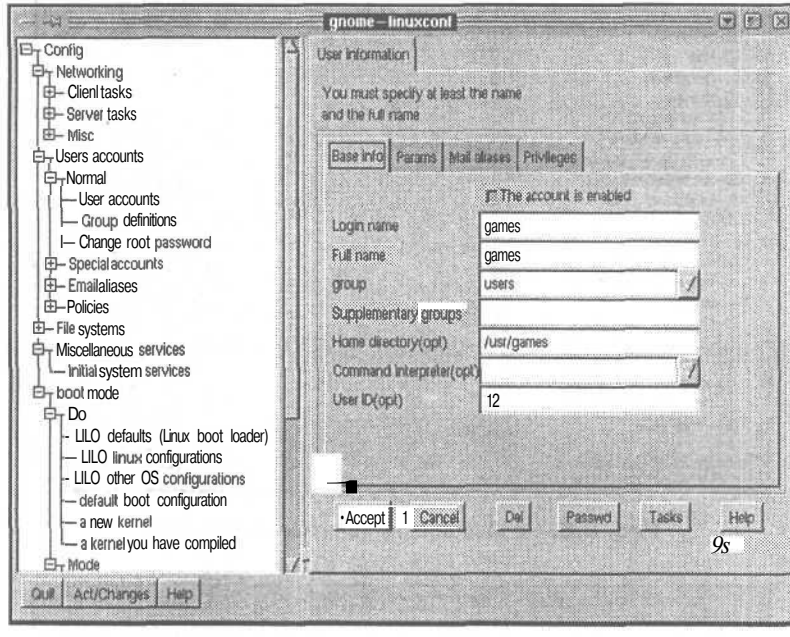


Рис. 8.8. Окно изменения параметров счета пользователя

Описание всех возможностей программы linuxconf заняло бы слишком много места в данной книге. Я надеюсь, что общее представление о программе вы получили и сможете освоиться с ней самостоятельно. Кроме того, неплохое описание этой программы можно найти в нескольких книгах по Linux, например, в [П1.3] *(см. приложение)*. Надо только сказать, что в Интернете встречаются неодобрительные отзывы некоторых пользователей об этой программе. Они советуют вместо ее использования напрямую редактировать конфигурационные файлы. Однако я не наталкивался на случаи ее некорректной работы. Решайте сами, пользоваться этой программой или нет.

В заключение нужно еще сказать, что работа с программой в текстовом режиме происходит вполне аналогично, только вместо мышки для перемещения по группам приходится пользоваться клавиатурой, а, выбрав конкретную группу, надо нажать клавишу <Enter> для доступа к окну, в котором можно изменять значения отдельных параметров.

8.8. Настройка окружения пользователя

Как вы уже знаете, при входе любого пользователя в систему для него запускается особый экземпляр оболочки — login shell. В процессе запуска в качестве login shell bash ищет следующие файлы:

- /etc/profile
- ~/.bash_profile
- ~/.bash_login
- ~/.profile

(в указанном порядке) и выполняет содержащиеся в них команды. Если bash запускается повторно из командной строки в интерактивном режиме (то есть не для выполнения какой-то одиночной команды), то он находит файл ~/.bashrc и выполняет содержащиеся в нем команды. Впрочем, в дистрибутиве Mandrake файл ~/.bashrc вызывается и для login shell, а из него вызывается еще и общесистемный файл /etc/bashrc. Так что, как видите, тут возможны варианты.

Но какова бы ни была последовательность вызова этих скриптов, с их помощью для каждого сеанса работы пользователя создается так называемая "пользовательская среда" или окружение, представляющая собой набор переменных с установленными для них значениями. Эти значения считываются некоторыми программами и утилитами, и в соответствии с их значениями изменяется поведение системы в тех или иных ситуациях.

Файлы /etc/profile и /etc/bashrc определяют общесистемные настройки пользовательской среды, а остальные перечисленные файлы определяют индивидуальную среду конкретного пользователя. Сравнительно небольшие добавления или исправления в индивидуальных файлах настройки, изменяющие значения, заданные по умолчанию, могут сделать значительно более приятной для вас работу в системе (о вкусах, как известно, не спорят, и вряд ли люди, которые определяли настройки по умолчанию, угадали ваши предпочтения). Поэтому давайте кратко рассмотрим основные переменные пользовательской среды и то, каким образом вы сами можете их изменять.

Вначале посмотрите переменные окружения, которые заданы по умолчанию. Как мы уже говорили в *разд. 5.6,* это можно сделать с помощью команды set (и аналогичной ей команды typeset) или env. Значение, присвоенной отдельной переменной, можно просмотреть с помощью команды echo `$name`, где name — **ИМЯ переменной.**

Из всех переменных, которые вы увидите по команде set, обычно меняют вид приглашения `PS1` и перечень путей поиска PATH. О том, как поменять значения этих переменных, было подробно рассказано в *разд. 5.6.* Если вы почему-либо пропустили этот раздел, загляните в него сейчас. Так что оста-

лось только решить, в каком именно скрипте задать этим переменным новые значения. Рассмотрим этот вопрос на примере переменной PATH.

Переменная PATH формируется в двух скриптах: /etc/profile (пути, общие для всех пользователей) и в одном из пользовательских скриптов (например, в ~/.bash_profile), где к ранее сформированному перечню пользователь может добавить пути по своему желанию. Только не стоит делать это в файле ~/bashrc, т. к. последний перезапускается каждый раз при запуске второго, третьего и т. д. экземпляра оболочки. Для добавления пути в переменную PATH надо вписать в выбранный скрипт строку следующего вида (в этом примере в перечень добавляется путь /home/user/bin):

```
PATH=$PATH:/home/user/bin
```

Обратите внимание на то, что двоеточия в конце нет. И имейте в виду, что каталоги просматриваются в поисках нужного файла в том порядке, как они перечислены в переменной PATH.

В отличие от MS-DOS Linux не ищет исполняемый файл в текущем каталоге. Поэтому, если вы хотите, чтобы поиск производился и в текущем каталоге, надо добавить и этот каталог (напомним, что он имеет имя, состоящее из одной точки) в переменную PATH. Но имейте в виду, что с точки зрения безопасности добавлять текущий каталог в перечень путей поиска недопустимо, т. к. злоумышленник может поместить в один из доступных ему по записи каталогов вредоносную программу, названную именем одной из часто используемых системных утилит. И, когда вы запустите эту программу, считая, что запускаете системную утилиту, она нанесет большой вред вашей системе, тем более, если вы запустили ее от имени суперпользователя.

При желании можно слегка "украсить себе жизнь", включив цветной вывод в команде ls (если по умолчанию он черно-белый). Для этого редактируем файл /etc/bashrc, в который добавляем строку:

```
alias ls = "ls -F --color"
```

Кстати, если вы захотите, чтобы Midnight Commander в окне эмулятора терминала был цветным, а не черно-белым, то установите переменную COLORTERM:

```
COLORTERM = ; export COLORTERM
```

Глава 9

Подключение и настройка аппаратных устройств

Существует бесконечное множество аппаратных конфигураций, рассмотреть каждую из которых не представляется возможным. Даже если говорить только о типах устройств, и то рассмотреть их все будет трудновато. А поскольку данная книга предназначена для начинающих пользователей Linux, будем для простоты считать, что компьютер такого пользователя имеет следующую конфигурацию:

- жесткий диск достаточно большого объема, разбитый на три раздела (раздел для DOS/Windows, раздел для Linux, swap-раздел);
- дисковод для гибких дисков на 3,5 дюйма;
- дисковод CD-ROM;
- принтер;
- клавиатура;
- мышь;
- звуковая карта;
- модем;
- внешний дисковод Zip фирмы Iomega, подключаемый через параллельный порт.

Вопросы настройки перечисленных аппаратных компонентов и рассмотрим. Причем, поскольку настройка монитора была разобрана в *гл.* 7, а о подключении и настройке модема будет рассказано в разделе, посвященном работе в Интернете, то нам остается рассмотреть только 7 типов устройств. Но, прежде чем переходить к рассказу об устройствах конкретных типов, необходимо пояснить такие общие понятия, как драйвер устройства и специальный файл устройства.

9.1. Драйверы устройств

Как уже говорилось выше, одной из основных задач операционной системы является управление аппаратной частью. Ту программу или тот кусок программного кода, который предназначен для управления конкретным устройством, и называют обычно драйвером устройства. Необходимость драйверов

устройств в операционной системе объясняется тем, что каждое отдельное устройство воспринимает только свой строго фиксированный набор специализированных команд, с помощью которых этим устройством можно управлять. Причем команды эти чаще всего предназначены для выполнения каких-то простых элементарных операций. Если бы каждое приложение вынуждено было использовать только эти команды, писать приложения было бы очень сложно, да и размер их был бы очень велик. Поэтому приложения обычно используют какие-то команды высокого уровня (типа "записать файл на диск"), а о преобразовании этих команд в управляющие последовательности для конкретного устройства заботится драйвер этого устройства. Поэтому каждое отдельное устройство, будь то дисковод, клавиатура или принтер, должно иметь свой программный драйвер, который выполняет роль транслятора или связующего звена между аппаратной частью устройства и программными приложениями, использующими это устройство.

В Linux драйверы устройств бывают трех типов.

Драйверы первого типа являются частью программного кода ядра (встроены в ядро). Соответствующие устройства автоматически обнаруживаются системой и становятся доступны для приложений. Обычно таким образом обеспечивается поддержка тех устройств, которые необходимы для монтирования корневой файловой системы и запуска компьютера. Примерами таких устройств являются стандартный видеоконтроллер VGA, контроллеры IDE-дисков, материнская плата, последовательные и параллельные порты.

Драйверы второго типа представлены модулями ядра. Они оформлены в виде отдельных файлов и для их подключения (на этапе загрузки или впоследствии) необходимо выполнить отдельную команду подключения модуля, после чего будет обеспечено управление соответствующим устройством. Если необходимость в использовании устройства отпала, модуль можно выгрузить из памяти (отключить). Поэтому использование модулей обеспечивает большую гибкость, т. к. каждый такой драйвер может быть переконфигурирован без остановки системы. Модули часто используются для управления такими устройствами, как SCSI-адаптеры, звуковые и сетевые карты.

Файлы модулей ядра располагаются в подкаталогах каталога /lib/modules. Обычно при инсталляции системы задается перечень модулей, которые будут автоматически подключаться на этапе загрузки. Список загружаемых модулей хранится в файле /etc/modules. А в файле /etc/modules.conf находится перечень опций для таких модулей. Редактировать этот файл "вручную" не рекомендуется, для этого существуют специальные скрипты (типа update-modules).

Для подключения или отключения модулей в работающей системе имеются специальные утилиты.

- `lsmod` — выдает список загруженных в данный момент модулей.
- `insmod` — служит для загрузки или "установки" модуля из командной строки.

Пример:

```
insmod joystick
```

- `rmmod` — служит для выгрузки или "удаления" модуля.

  Пример:

  ```
  rmmod joystick
  ```

- `modprobe` — автоматически загружает модули. Для того, чтобы отобразить текущую конфигурацию всех модулей, можно воспользоваться командой modprobe -c.

Примечание

Хотя файлы модулей имеют суффикс `.o`, при использовании этих команд ссылки на модули указываются без упоминания этого суффикса. Например: при упоминании модуля, файл которого называется "joystick.o", вы должны использовать в командной строке просто "joystick".

И, наконец, для *третьего типа драйверов устройств* программный код драйвера поделен между ядром и специальной утилитой, предназначенной для управления данным устройством. Например, для драйвера принтера ядро отвечает за взаимодействие с параллельным портом, а формирование управляющих сигналов для принтера осуществляет демон печати lpd (который использует для этого специальную программу-фильтр, о чем подробнее будет рассказано ниже, в *разд. 9.6)*. Другие примеры драйверов этого типа — драйверы модемов и X-сервер (драйвер видеоадаптера), о котором шла речь в *гл. 7.*

Но надо специально отметить, что во всех трех случаях непосредственное взаимодействие с устройством осуществляет ядро или какой-то модуль ядра. А пользовательские программы взаимодействуют с драйверами устройств через специальные файлы, расположенные в каталоге /dev и его подкаталогах. То есть взаимодействие прикладных программ с аппаратной частью компьютера в ОС Linux осуществляется по следующей схеме:

устройство ↔ ядро ↔ специальный файл устройства ↔
↔ программа пользователя

Такая схема обеспечивает единый подход ко всем устройствам, которые с точки зрения приложений выглядят как обычные файлы.

9.2. Специальные файлы устройств

Однако, в отличие от обычных файлов, специальные файлы устройств в действительности есть только указатели на соответствующие драйверы устройств в ядре. По сравнению с обычными файлами файлы устройств имеют

три дополнительных атрибута, которые характеризуют устройство, соответствующее данному файлу.

- **Класс устройства. В** ОС Linux различают устройства блок-ориентированные и байт-ориентированные. Блок-ориентированные (или блочные) устройства, например, жесткий диск, передают данные блоками. Байт-ориентированные (или символьные) устройства, например, принтер и модем, передают данные посимвольно, как непрерывный поток байтов. Взаимодействие с блочными устройствами может осуществляться лишь через буферную память, а для символьных устройств буфер не требуется. Кроме этих двух классов устройств имеются еще два — небуферизованные байт-ориентированные устройства и именованные каналы (FIFO).
- **Старший номер устройства,** обозначающий тип устройства, например, жесткий диск или звуковая плата. Текущий список старших номеров устройств можно найти в файле /usr/include/linux/major.h. В табл. 9.1 приведена небольшая выдержка из этого списка.

Таблица 9.1. *Старшие номера некоторых устройств*

Старший номер	Тип устройства
1	Оперативная память
2	Дисковод гибких дисков
3	Первый контроллер для жестких IDE-дисков
4	Терминалы
5	Терминалы
6	Принтер (параллельный разъем)
8	Жесткие SCSI-диски
14	Звуковые карты
22	Второй контроллер для жестких IDE-дисков

Файлы устройств одного типа имеют одинаковые имена и различаются по номеру, прибавляемому к имени. Например, все файлы сетевых плат Ethernet имеют имена, начинающиеся на eth: ethO, eth1 и т. д.

- **Младший номер устройства** применяется для нумерации устройств одного типа, т. е. устройств с одинаковыми старшими номерами.

Если вы заглянете в каталог /dev и выполните команду `ls -l`, вы увидите, что эта команда вместо размера файла в байтах, как для обычного файла, выводит два числа, разделенных запятой. Это и есть старший и младший номера данного устройства. Эти номера задаются в соответствии с таблицей устройств, определенной разработчиками ядра.

Старшие номера известных ядру устройств можно увидеть, выполнив команду

```
[user]$ cat /proc/devices
```

Если вы решили подключить к системе какое-то новое устройство, необходимо вначале проверить, что в каталоге /dev имеется специальный файл (или ссылка на специальный файл) для этого устройства. Специальные файлы устройств создаются с помощью команды `mknod` (но, естественно, использовать команду mknod без необходимости и полного понимания последствий не стоит). Эта команда имеет следующий формат:

```
mknod [опции] имя_устройства тип_устройства старший_номер младший_номер
```

где `тип_устройства` может принимать одно из четырех значений:

- `b` — блок-ориентированное устройство;
- `c` — байт-ориентированное (символьное) устройство;
- `u` — небуферизованное байт-ориентированное устройство;
- `p` — именованный канал.

Для блок-ориентированных и байт-ориентированных устройств (b, c, и) нужны и старший и младший номера, для именованных каналов номера не используются. В следующем примере создается специальный файл для терминала, подключенного к порту COM3, который в Linux обозначается как /dev/ttyS2:

```
[root]# mknod -m 660 /dev/ttyS2 c 4 66
```

(устройства-терминалы представляют собой байт-ориентированные устройства со старшим номером 4 и младшими номерами, которые начинаются с 64).

Но вот о чем стоит подумать, так это о том, как дать пользователям права, необходимые для доступа к устройствам. Эти права устанавливаются через атрибуты специальных файлов. Можно, например, дать всем пользователям полные права (chmod `666`) на доступ к таким устройствам, как /dev/cdrom, /dev/floppy, /dev/modem и т. д. Можете поступить иначе, создав группу "cdrom", сделать /dev/cdrom принадлежащим группе cdrom, а потом добавлять пользователей в эту группу по мере необходимости. Аналогичную процедуру можно применить к другим устройствам.

9.3. Клавиатура

Клавиатура к вашему компьютеру уже, скорее всего, подключена, вопрос может состоять только в том, чтобы настроить ее. Настройка клавиатуры заключается в настройке таких вещей, как:

- раскладка клавиатуры;
- скорость повтора посылаемых клавиатурой сигналов в случае удержания клавиш пользователем;

- длительность интервала задержки от момента нажатия клавиши до того момента, когда клавиатура начинает повторять посылку сигналов.

Два последних параметра (скорость повтора и время задержки) устанавливаются с помощью специальной команды kbdrate.

9.3.1. Команда *kbdrate*

Скорость повтора задается в символах в секунду и может принимать только определенные значения в пределах от 2 до 30 символов в секунду. Но задать (после опции `-r`) вы можете любое значение в этих пределах, программа сама выберет ближайшее допустимое значение. Число после опции -d задает задержку в миллисекундах (допустимы значения от 250 до 1000 с шагом 250). Чтобы не устанавливать эти значения после каждого перезапуска компьютера, можно добавить в файл /etc/rc.d/rc.sysinit сроку следующего вида:

```
/sbin/kbdrate -s -r 16 -d 500
```

где опция -s просто подавляет вывод ненужных в данном случае сообщений. Если эту команду выполнить без указания параметров, для скорости повтора и задержки будут установлены значения по умолчанию: для скорости повтора — 10,9 символов в секунду, а для задержки — 250 миллисекунд.

Еще один вопрос, относящийся к настройке клавиатуры, — это способ изменения положения переключателей NumLock, CapsLock и ScrollLock. Для этого можно воспользоваться командой setleds. Например, для того, чтобы переключатель NumLock был по умолчанию включен, добавьте в файл /etc/rc.d/rc.sysinit следующие строки:

```
for tty in /dev/tty[1-9]*; do
    setleds -D +num < $tty
done
```

Изменение раскладки клавиатуры — это вопрос значительно более сложный. Но, поскольку этот вопрос имеет большое значение как вообще для настройки клавиатуры, так и для решения проблемы русификации, его необходимо рассмотреть подробнее.

И начать придется с краткого изложения проблем кодировки символов.

9.3.2. Таблицы кодировки символов

В человеческом мире информация представляется последовательностями *символов*. Каждый символ имеет каноническое изображение, которое позволяет однозначно идентифицировать данный символ. Шрифты задают разные варианты начертания символов.

В вычислительных машинах для представления информации используются цепочки байтов. Поэтому для перевода информации из машинного представления в человеческий необходимы *таблицы кодировки символов* — таблицы соответствия между символами определенного языка и кодами символов.

Самой известной таблицей кодировки является код ASCII (Американский стандартный код для обмена информацией), который был разработан для передачи текстов по телеграфу задолго до появления компьютеров. Этот код является 7-битным, т. е. для кодирования символов английского языка, служебных и управляющих символов используются только 128 7-битных комбинаций. При этом первые 32 комбинации (кода) служат для кодирования управляющих сигналов (начало текста, конец строки, перевод каретки, звонок, конец текста и т. д.).

При разработке первых компьютеров фирмы IBM этот код был использован для представления символов в компьютере. Поскольку в исходном коде ASCII было всего 128 символов, для их кодирования хватило тех однобайтовых кодов, у которых 8-й бит равен 0. Во второй половине кодовой таблицы (значения байта с 8-м битом равным 1) фирма IBM разместила символы псевдографики, математические знаки и некоторые символы из языков, отличных от английского (немецкие умляуты, французские диакритические знаки, символы греческого алфавита и т. п.). Эту кодовую таблицу стали называть кодировкой IBM.

Когда IBM-совместимые персональные компьютеры стали использовать в других странах, потребовалось обеспечить обработку информации на языках, отличных от английского. Для того чтобы полноценно поддерживать другие языки, фирма IBM ввела в употребление несколько кодовых таблиц, ориентированных на конкретные страны. Так, для скандинавских стран была предложена таблица 865 (Nordic), для арабских стран — таблица 864 (Arabic), для Израиля — таблица 862 (Israel) и т. д. В этих таблицах часть кодов из второй половины кодовой таблицы использовалась для представления символов национальных алфавитов (за счет исключения некоторых символов псевдографики). Для представления символов кириллицы была введена кодировка IBM-866.

Однако с русским языком ситуация развивалась особым образом. Очевидно, что замену символов во второй половине кодовой таблицы можно произвести разными способами. В других европейских странах сумели найти единое решение, а для русского языка появилось несколько разных таблиц кодировки символов кириллицы: IBM-866, CP-1251, KOI8-R, ISO-8859-5. Все они одинаково изображают символы первой половины таблицы (от 0 до 127) и различаются представлением символов русского алфавита и псевдографики во второй половине.

Одна из самых известных кодовых таблиц для кириллицы получила название альтернативной (по отношению к кодировке IBM-866, наверное). Она

была разработана фирмой Microsoft для MS-DOS. При ее разработке постарались сделать так, чтобы результирующая таблица была насколько это возможно совместима с кодировкой IBM. Поэтому альтернативная кодировка — это кодировка IBM, в которой все специфические европейские символы в верхней половине были заменены на кириллицу, оставляя псевдографические символы нетронутыми. Следовательно, это не портило вид программ, использующих для работы текстовые окна, что было очень существенным фактором для работы в среде MS-DOS, основой которой был именно текстовый режим.

Кодировка KOI-8 была разработана изначально с ориентировкой на UNIX. Так как UNIX в своей основе сетевая ОС, то основной идей при создании KOI-8 была идея об обеспечении перемещения кириллической информации по сети. Но для передачи-то использовался 7-битный стандарт ASCII. Разработчики поместили кириллические символы в верхней части таблицы таким образом, что позиции кириллических символов соответствуют их фонетическим аналогам в английском алфавите в нижней части таблицы. Это означает, что, если в тексте, написанном в KOI-8, мы убираем восьмой бит каждого символа, *то мы все еще имеем "читабельный" текст, хотя он и написан английскими символами!* Не удивительно, что KOI8-R быстро стал фактически стандартом для кириллицы в Интернет, что и нашло отражение в RFC 1489 *("Registration of a Cyrillic Character Set")*. Автором этого документа является А. Чернов, который проделал огромный объем работы, чтобы превратить KOI-8 в стандарт Интернета.

Международная организация по стандартизации (ISO) внесла свою лепту в создание различных кодировок кириллицы, когда ввела семейство стандартов, известных как ISO 8859-X. Это семейство есть совокупность 8-битных кодировок, где младшая половина каждой кодировки (символы с кодами О—127) соответствует ASCII, а старшая половина определяет символы для различных языков. Например:

- 8859-0 — новый европейский стандарт (так называемый Latin 0);
- 8859-1 — Европа, Латинская Америка (также известный как Latin 1);
- 8859-2 — Восточная Европа;
- 8859-5 — кириллица;
- 8859-8 — идиш.

Фирма Microsoft еще больше запутала ситуацию с кодировками для русского языка, когда при разработке Windows ввела кодировку CP-1251.

Таблицы кодировок, содержащие 256 символов, стали называть расширенными кодами ASCII (потому что в основе любой из них лежит 128-символьный код ASCII), кодовыми страницами или английским термином *character set* (который часто сокращают до *charset)*.

Но в мире есть языки, такие как китайский или японский, для которых 256 символов в принципе недостаточно. Кроме того, всегда существует проблема вывода или сохранения в одном файле одновременно текстов на разных языках (например, при цитировании). Поэтому была разработана универсальная кодовая таблица UNICODE, содержащая символы, применяемые в языках всех народов мира, а также различные служебные и вспомогательные символы (знаки препинания, математические и технические символы, стрелки, диакритические знаки и т. д.). Очевидно, что одного байта недостаточно для кодирования такого большого множества символов. Поэтому в UNICODE используются 16-битные (2-байтные) коды, что позволяет представить 65 536 символов. К настоящему времени задействовано около 49 000 кодов (последнее значительное изменение — введение символа валюты EURO в сентябре 1998 г.). Для совместимости с предыдущими кодировками первые 128 кодов совпадают со стандартом ASCII. На рис. 9.1 схематично представлено размещение символов разных языков в кодовом пространстве UNICODE.

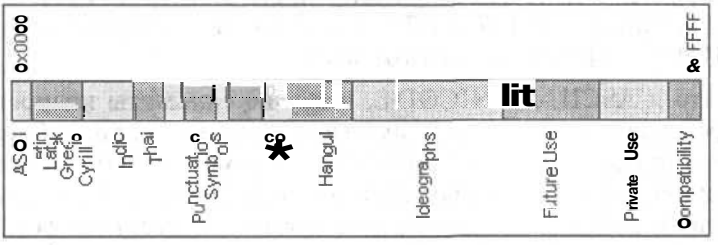


Рис. 9.1. Структура UNICODE

В стандарте UNICODE кроме определенного двоичного кода (эти коды принято обозначать буквой U, после которой следуют знак + и собственно код в шестнадцатеричном представлении) каждому символу присвоено определенное имя. В табл. 9.2 приведено несколько примеров кодов и имен символов из стандарта UNICODE.

Таблица 9.2. Примеры именования кодов UNICODE

Символ	UNICODE	Название символа (Character Name)
A	U+0041	LATIN CAPITAL LETTER A
a	U+0061	LATIN SMALL LETTER A
Ю	U+042E	CYRILLIC CAPITAL LETTER YU
+	U+002B	PLUS SIGN
1	U+0031	DIGIT ONE

Таблица 9.2 *(окончание)*

Символ	UNICODE	Название символа (Character Name)
Ω	U+03A9	GREEK CAPITAL LETTER OMEGA
!	U+2569	BOX DRAWINGS DOUBLE UP AND HORIZONTAL

Еще одним компонентом стандарта UNICODE являются алгоритмы для взаимно-однозначного преобразования кодов UNICODE в последовательности байтов переменной длины. Необходимость таких алгоритмов обусловлена тем, что не все приложения умеют работать с UNICODE. Некоторые приложения понимают только 7-битные ASCII-коды, другие приложения — 8-битные (расширенные) ASCII-коды. Для представления символов, не поместившихся, соответственно, в 128-символьный или 256-символьный набор, такие приложения используют цепочки байтов переменной длины. Алгоритм UTF-7 служит для обратимого преобразования кодов UNICODE в цепочки 7-битных ASCII-кодов, а UTF-8 — для обратимого преобразования кодов UNICODE в цепочки из расширенных 8-битных ASCII-кодов. Подробнее об алгоритмах UTF-7 и UTF-8 и кодировках вообще вы можете прочитать в [П11.3 — П11.5] *(см. приложение).*

Отметим, что и ASCII, и UNICODE, и другие стандарты кодировки символов не определяют изображения символов, а только состав набора символов и способ его представления в компьютере. Кроме того (что, может быть, не сразу очевидно), они еще задают порядок перечисления символов в наборе, который очень важен, т. к. он влияет самым существенным образом на алгоритмы сортировки. Именно таблицу соответствия символов из какого-то определенного набора (скажем, символов, применяемых для представления информации на английском языке, или на разных языках, как в случае с UNICODE) и обозначают термином *таблица кодировки символов* или *charset.* Каждая стандартная кодировка имеет *имя,* например, KOI8-R, ISO_8859-1, ASCII. К сожалению, стандарта на имена кодировок не существует.

9.3.3. Ввод символов с клавиатуры

В процессе ввода символов с клавиатуры можно выделить четыре соответствия или отображения (в математическом смысле этого слова).

1. На клавиатуру нанесены (или наклеены) символы. Это первое соответствие: символ → клавиша.
2. Микропроцессор клавиатуры реализует второе соответствие: комбинация клавиш → скан-код.
3. Далее скан-код клавиатуры преобразуется в код символа, понятный приложению, например, ASCII-код или UNICODE; это третье соответствие: скан-код → код символа, используемый приложением.

4. Для изображения символа на экране или принтере используется четвертое соответствие: ASCII-код → изображение символа.

О наличии этих соответствий полезно помнить при рассмотрении вопросов взаимодействия пользователя с приложениями. А теперь вернемся к вопросу о том, как работает клавиатура.

Управление работой клавиатуры в текстовом режиме осуществляется драйвером терминала, который входит в состав ядра Linux. Драйвер терминала состоит как бы из двух отдельных драйверов: драйвера клавиатуры и драйвера экрана. Драйвер клавиатуры обрабатывает нажатия клавиш пользователем и передает результат прикладной программе, которая, в свою очередь, посылает экранному драйверу символы, которые должны быть отображены на экране.

При каждом нажатии на клавишу микропроцессор клавиатуры генерирует последовательность так называемых скан-кодов, которая представляет собой последовательность из двух или большего числа байтов. Эта последовательность передается драйверу клавиатуры, который может работать в одном из 4 режимов:

- **K_RAW**, когда прикладной программе передается последовательность скан-кодов, сгенерированных клавиатурой. Этот режим используется при работе с приложениями, которые имеют собственный драйвер клавиатуры. Примером такого приложения является система X Window.
- **K_MEDIUMRAW**, когда скан-код клавиши преобразуется в один из 127 возможных кодов, называемых кодами клавиш (keycodes). Каждый код клавиши состоит из кода нажатия клавиши и кода отпускания клавиши. Преобразование скан-кодов в коды клавиш осуществляется в соответствии с внутренней таблицей драйвера клавиатуры. Обычно эта таблица фиксирована, и изменять ее не требуется, хотя в системе существуют команды getkeycodes И setkeycodes, С ПОМОЩЬЮ которых МОЖНО просмотреть ИЛИ изменить некоторые соответствия в этой таблице. Эти команды используются только в том случае, если у вас программируемая клавиатура.
- П **K_XLATE** (или режим ASCII), когда код клавиши преобразуется в ASCII-код символа или некоторую последовательность ASCII-кодов символов в соответствии с таблицей раскладки клавиатуры, которая хранится в виде отдельного файла. Например, для Red Hat Linux 5.2 по умолчанию используется файл defkeymap.map в каталоге /usr/lib/kbd/keymaps/i386/qwerty. Команда `dumpkeys` выводит на экран содержание действующей в данный момент таблицы раскладки клавиатуры, а команда loadkeys загружает в драйвер таблицу раскладки клавиатуры из указанного файла.
- О K_UNICODE, когда скан-коды преобразуются в двухбайтовые коды таблицы UNICODE (этот режим пока используется очень редко).

Выбор режима работы драйвера терминала определяется прикладной программой, которая в данный момент времени выполняется компьютером. Чаще всего используется третий режим, когда код клавиши либо преобразуется в ASCII-код символа или строку таких кодов в соответствии с таблицей раскладки клавиатуры, либо выполняется действие, определенное для конкретной комбинации клавиш в таблице раскладки клавиатуры. Например, нажатие <Ctrl>+<Alt>+<Del> эквивалентно вызову команды shutdown -r о, т. е. приводит к останову системы и перезагрузке компьютера.

Режим работы драйвера клавиатуры можно узнать или изменить с помощью команды kbd_mode. Однако не торопитесь менять режим, т. к. перевод драйвера клавиатуры в режим RAW или MEDIUMRAW может сделать его недоступным для большинства приложений, т. е. легко можно вообще потерять возможность ввода команд.

Не для всех клавиш и комбинаций клавиш процесс обработки проходит так прямолинейно, как это описано выше. Во-первых, имеется несколько особых клавиш, так называемых клавиш-переключателей. Это клавиши <Shift> (левая и правая), <Alt> (левая и правая), <Ctrl> (левая и правая), <Caps Lock>, <Num Lock>, <Ins>. Нажатие на клавишу-переключатель изменяет значение одного из разрядов (битов) в двухбайтном слове, которое хранит состояние клавиш-переключателей. Поэтому драйвер клавиатуры вначале должен проанализировать состояние этого слова, а затем, соответственно, преобразовать коды.

Клавиша <Ins> является единственной из клавиш-переключателей, нажатие которой не только заносит признак в слово состояния переключателей, но и порождает передачу соответствующего кода драйверу терминала.

С помощью ASCII-кодов можно представить 256 различных символов, а, значит, ASCII-коды можно сопоставить 256-ти кодам клавиш. Учитывая наличие клавиш-переключателей, комбинаций клавиш существует гораздо больше. Поэтому некоторые комбинации клавиш драйвер клавиатуры преобразует в цепочки из нескольких байтов, так называемые Escape-последовательности, в которых два первых байта служат признаком Escape-последовательности, а последующие представляют собой собственно значащие байты. Escape-последовательности обычно представляют те комбинации клавиш, которые используются для управления работой программ, таких как стрелки, клавиши <Page Down>, <Page Up>, <Home>, <End>, <F1>—<F12>, <Ins>, <Del> и т. д.

В комплект Red Hat Linux входит программа showkey, которая показывает все три вида кодов, связанных с нажатиями клавиш. Если запустить эту программу с параметром -s, она будет показывать скан-коды нажатий клавиш (чтобы выйти из программы, надо просто выждать 10 секунд, не нажимая в это время ни одной клавиши). Ввод команды showkey -k приводит к выводу на экран кодов клавиш (выходим так же). Ввод команды showkey -m

позволяет просмотреть ASCII-коды, которые выдаются драйвером клавиатуры после того, как скан-код клавиши будет оттранслирован с помощью таблицы раскладки клавиатуры. Попробуйте в этом режиме нажать <Ctrl>+<=> или <Ctrl>+<Esc>, и вы увидите, что не каждая комбинация клавиш порождает ASCII-код (попробуйте также клавиши-переключатели). В новых версиях программы showkey появилась опция -и, при которой отображаются коды UNICODE.

Примечание

Если вы уже перешли в графический режим, то программа showkey может работать некорректно, о чем она вежливо сообщает при запуске. Обратите внимание на эти сообщения! Кстати, showkey неправильно работает не только в графическом режиме, но и при подключении через telnet и т. п.

Поскольку чаще всего используется режим преобразования в ASCII-коды, например, при работе в текстовом режиме и в эмуляторе терминала, таблица раскладки клавиатуры имеет такое большое значение. Возможно, вас не устраивает та раскладка, которая используется по умолчанию. Давайте рассмотрим, как ее поменять.

9.3.4. Изменение раскладки клавиатуры для текстового режима

В дистрибутиве Red Hat загрузка таблицы раскладки клавиатуры и системного фонта производится в файле /etc/rc.d/rc.sysinit. Но лезть в этот файл и корректировать его содержимое для изменения раскладки не требуется. Дело в том, что файлы с различными раскладками находятся в каталогах /lib/kbd/keymaps/i386/qwerty или /usr/lib/kbd/keymaps/i386/qwerty, а выбор конкретного файла раскладки задается файлом /etc/sysconfig/keyboard. Этот файл можно отредактировать вручную, а можно — с помощью программы `kbdconfig`.

Команда `kbdconfig` прописывает новое значение в файл /etc/sysconfig/keyboard и загружает указанную таблицу в оперативную память. Того же эффекта можно добиться, если приписать имя новой таблицы в файл /etc/sysconfig/keyboard и выполнить команду

```
[root]# /etc/rc.d/init.d/keytable start
```

Оба этих варианта позволяют переключиться на новую раскладку "на ходу".

Если же только откорректировать содержимое файла /etc/sysconfig/keyboard, то перезагрузка таблицы произойдет только после перезапуска компьютера или после выполнения команды (в примере загружается раскладка из файла ru-win.map):

```
[root]# loadkeys /usr/lib/kbd/keymaps/i386/qwerty/ru-win.map
```

Впрочем, переключение "на ходу" вряд ли требуется делать, поскольку обычно человек привыкает к одной раскладке, и пальцы сами находят привычные клавиши, так что всякое изменение тут только осложнит работу. Поэтому имеет смысл поэкспериментировать один раз с различными раскладками, выбрать наиболее удобную (считай, привычную) и на этом можно успокоиться.

При установке русифицированных дистрибутивов Linux (по крайней мере Black Cat) обычно выбирается раскладка ru1 (точка на <Shift>+<7>, запятая на <Shift>+<6>). Для тех, кто привык работать в Windows, может оказаться более привычной раскладка как в Windows (в русском регистре точка и запятая находятся рядом с правой клавишей <Shift>). Для таких пользователей имеется раскладка ru_ms. Если вас не удовлетворяют эти варианты, то можете выбрать любую из имеющихся в вашей системе, либо найти что-либо подходящее в Интернете. Предположим, что вы нашли и скачали файл ru_win_ctrl.map.gz от IP Labs (**http://www.iplabs.ru/Linux/ru_win_ctrl.map.gz**). Остается только положить этот файл в каталог /usr/lib/kbd/keytables/i386/qwerty/, запустить kbdconfig и выбрать ru_win_ctrl.

После установки новой таблицы раскладки клавиатуры иногда возникают затруднения в определении того, какая именно клавиша или комбинация клавиш переключает из режима ввода английских символов в режим ввода русских. Гадать тут не надо, достаточно просмотреть файл таблицы раскладки клавиатуры. Обычно в самом начале файла эта комбинация указывается открытым текстом, правда, в большинстве случаев английским. (Если вы забыли, какая именно таблица загружена, то посмотрите файл /etc/sysconfig/keyboard).

9.3.5. Создание собственной раскладки

Если вас не устраивает ни одна из тех раскладок клавиатуры, которые имеются в каталоге /usr/lib/kbd/keytables/i386/qwerty/, можете попробовать подправить ту раскладку, которая ближе всего к вашему идеалу. Попробуем показать, как это делается, на примере выбора клавиши переключения между русской и латинской клавиатурой (этот совет позаимствован у Романа Минакова, **pharao@kma.mk.ua**).

Для переключения между русской и латинской клавиатурой часто используется правая клавиша <Ctrl>, в то время как на любой более-менее современной IBM-клавиатуре есть три клавиши, которые, как правило, в Linux не задействованы. Вот одну из них и приспособим для переключения алфавитов. Для начала надо узнать какой у них код. Запускаем команду showkey с опцией `--keycodes` (запуск showkey, естественно, производится с консоли и необходимо предварительно выйти из оболочки Midnight Commander) и последовательно (слева направо) нажимаем эти три клавиши, чтобы узнать их коды:

```
[root]# showkey --keycodes
kb mode was XLATE
```

```
press any key (program terminates after 10s of last keypress)...
keycode 125 press
keycode 125 release
keycode 126 press
keycode 126 release
keycode 127 press
keycode 127 release
```

Числа 125, 126, 127 и есть коды этих клавиш. Далее переходим в каталог /usr/lib/kbd/keytables/i386/qwerty, находим файл, который используется в данный момент (что-то типа ru1.map, если в каталоге /usr/lib/kbd/keytables/i386/qwerty вы найдете только rul.map.gz, то выполните предварительно разархивацию: `gunzip ru1.map.gz`).

Для того чтобы заставить клавишу работать как временный переключатель с русского на латинский (пока клавиша удерживается), надо придать ей значение AltGr, а чтобы она использовалась как постоянный переключатель – AltGr_Lock. Находим внутри ru1.map:

```
keycode 125 =
keycode 126 =
keycode 127 =
```

и меняем на:

```
keycode 125 =
keycode 126 = AltGr
keycode 127 = AltGr_Lock
```

Далее надо изменить установки тех клавиш, которые ранее использовались для переключения. Например, если в качестве постоянного переключателя использовалась клавиша <Ctrl> (код клавиши 97), находим строку

keycode 97 =

и вписываем:

```
keycode 97 = Control
```

В итоге получаем: клавиша, расположенная возле правой клавиши <Ctrl>, — фиксированный переключатель "рус/лат", а та, что рядом с правой клавишей <Alt> — временный переключатель "рус/лат" (то есть действующий только на то время, пока удерживается в нажатом положении соответствующая клавиша).

После редактирования сохраняем файл под новым именем (например, mymap.kmap) и записываем это имя в /etc/sysconfig/keyboard.

9.3.6. Работа с клавиатурой вграфическомрежиме

В графическом режиме работа с клавиатурой организована значительно сложнее. Подробное описание этого вопроса можно найти в обстоятельном (но, к сожалению, очень трудном для понимания) материале Ивана Паскаля "X Keyboard Extension" *(см. [П11.6] приложения)*. Приведем очень краткий конспект основных положений этого материала.

Как было сказано выше, при работе в системе X Window клавиатура передает этой системе чистые скан-коды. Клавиатурный модуль X-сервера передает сообщение о нажатии (и отпускании) кнопки прикладной программе. В этом сообщении указывается только скан-код нажатой кнопки и "состояние клавиатуры" -- набор битовых "флагов", который отражает состояние клавиш-модификаторов (<Shift>, <Ctrl>, <Alt>, <Caps Lock> и т. п.). "Клиентская" программа должна сама решить — какой код символа, соответствующий скан-коду, надо выбрать при таком сочетании битов-модификаторов. Разумеется, при создании программ никто не пишет каждый раз программу для интерпретации скан-кодов. Для этих целей существуют специальные подпрограммы в библиотеке X-Lib. Процедуры из X-Lib, зная скан-код и "состояние клавиатуры", выбирают подходящий символ в соответствии с таблицей символов, которая хранится в X-сервере и которую они обычно "запрашивают" у X-сервера при старте программы. Эту таблицу можно менять с помощью утилиты xmodmap. Действующая таблица выводится командой xmodmap `-pk`.

9.3.7. Модуль XKB

В последних версиях дистрибутивов Linux устанавливается дополнительный модуль работы с клавиатурой — XKB. Модуль XKB точно также сообщает программе только скан-код и свое "состояние". Но в отличие от "старого" модуля (который называют "core protocol", или "core-модуль") XKB имеет более сложную таблицу символов, другой набор "модификаторов" и некоторые дополнительные параметры "состояния клавиатуры". Поэтому для полноценной работы с XKB библиотека X-Lib должна содержать модифицированные процедуры интерпретации скан-кодов (процедуры, "знающие" о XKB). Естественно, все версии X Window, у которых X-сервер "укомплектован" модулем XKB, имеют и соответствующую библиотеку X-Lib. Таким образом, XKB фактически делится на две части — модуль, встроенный в X-сервер, и набор подпрограмм, входящих в библиотеку X-Lib.

Однако, поскольку существуют программы, которые были рассчитаны на работу со старой библиотекой X-Lib, "не подозревающей" о существовании XKB, возникает "проблема совместимости". То есть, модуль XKB должен уметь общаться как со "своей" X-Lib, так и со "старой" (работающей в соответствии с

"core protocol"). Естественно, "общение" не ограничивается только передачей сообщений о нажатии/отпускании клавиш. Процедуры X-Lib могут обращаться к X-серверу с различными запросами (например — взять таблицу символов) и командами (например, поменять в этой таблице расположение символов). На все эти запросы и команды модуль XKB должен реагировать так, чтобы даже "старая" X-Lib могла работать правильно (насколько это возможно).

При старте X-сервера, модуль XKB зачитывает все необходимые данные из текстовых файлов, которые образуют "базу данных" настроек XKB. Строго говоря, большинство из этих файлов сам модуль XKB не читает. Он вызывает программу xkbcomp, которая переводит содержимое этих файлов в двоичный формат, понятный непосредственно модулю XKB. Но для настройки это не так уж важно, поскольку вызов xkbcomp происходит автоматически, незаметно для пользователя.

База данных, необходимых модулю XKB, находится в каталоге /usr/X11R6//lib/X11/xkb и состоит из 5 компонентов, расположенных в подкаталогах с именами:

- keycodes

  Здесь расположены таблицы, которые просто задают символические имена для скан-кодов. Например

  ```
  <TLDE>= 49;
  <AE01> = 10;
  ```

- types

  Здесь описываются возможные типы клавиш. Тип клавиши определяет как должно меняться значение, выдаваемое клавишей в зависимости от модификаторов (<Ctrl>, <Shift> и т. п.). Так, например, "буквенные" клавиши относятся к типу ALPHABETIC, что означает, что они имеют разное значение в зависимости от состояния <Shift> и <Caps LockX A клавиша <Enter> имеет тип ONE_LEVEL, что означает, что ее значение всегда одно и то же, независимо от состояния модификаторов.

- compat (сокращение от compability)

  Здесь описывается "поведение" модификаторов. В модуле XKB имеется несколько внутренних переменных, которые в конечном счете и определяют, какой символ будет генерироваться при нажатии клавиши в конкретной ситуации. Так вот, в файлах из каталога compat как раз описывается, как должны меняться эти переменные при нажатии различных клавиш-модификаторов. В этом же разделе обычно описывается и поведение "лампочек-индикаторов" на клавиатуре.

- symbols

  Этот каталог содержит таблицы, в которых для каждого скан-кода (задаваемого именем скан-кода, определенным в keycodes) перечисляются

все значения (symbols), которые должна выдавать клавиша. Естественно, количество различных значений зависит от типа клавиши (которые описываются в types), а какое именно значение будет выдано в конкретной ситуации, определяется состоянием модификаторов и их "поведением" (которое описывается в compat).

- geometry

  Здесь описываются варианты "геометрии" клавиатуры, т. е. расположение клавиш на клавиатуре. Эти описания нужны не столько самому X-серверу, сколько прикладным программам, которые рисуют изображение клавиатуры.

Надо сказать, что в каждом из этих каталогов имеется несколько файлов (иногда, довольно много) с разными настройками. Более того, каждый файл внутри себя может содержать несколько блоков (секций, разделов) вида

```
тип_компонента "имя_блока" {. . . . . };
```

Поэтому, для того, чтобы выбрать конкретную настройку, ее обычно указывают в виде имя_файла(имя_блока), например, us(pc104). В то же время, обычно один из блоков в файле (не обязательно самый первый) помечается флагом default. Это означает, что если указать только имя файла, то будет выбран именно этот блок.

Полная конфигурация XKB задается в секции InputDevice, определяющей клавиатуру, файла конфигурирования пакета XFree86, т. е. в файле /etc/X11/XF86Config-4. При этом имеется три способа задания конфигурации клавиатуры в этом файле.

Первый способ задания конфигурации заключается в том, что вы можете указать непосредственно каждый из компонентов, например

```
Option "XkbKeycodes"    "xfree86"
Option "XkbTypes"        "default"
Option "XkbCompat"       "default"
Option "XkbSymbols"     "us(pc104)"
Option "XkbGeometry"    "pc(pc104)"
```

Как легко догадаться, это означает, что:

- описание keycodes берется из файла "xfree86" в подкаталоге keycodes, причем из файла будет выбран тот блок, который помечен в нем флагом default;
- описание types берется из файла "default" в подкаталоге types;
- описание compat берется из файла "default" в подкаталоге compat;
- описание symbols берется из файла "us" в подкаталоге symbols, причем будет выбран блок "pc104";
- описание geometry берется из файла "pc" в подкаталоге geometry, блок "pc104";

Надо заметить, что в любом блоке (в любых компонентах) может встретиться инструкция include "имя_файла(имя_блока)" (естественно, имя_блока может отсутствовать) что означает, что в текущий блок должно быть вставлено другое описание из указанного файла (указанного блока). Поэтому полное описание может неявно включать в себя данные из многих других файлов, кроме тех, которые вы явно укажете в файле конфигурации X-сервера.

Второй способ задания конфигурации клавиатуры заключается в том, что вы можете указать одной инструкцией сразу полный набор настроек. Такие наборы называются keymaps и, так же как и обычные компоненты конфигурации XKB, располагаются в отдельных файлах (которые, тоже содержат в себе несколько именованных блоков) в подкаталоге keymap.

Обычно, в каждом блоке в файлах из keymap просто указывается из каких файлов XKB должен извлечь соответствующие компоненты (хотя, в принципе, там может быть и полное описание всех компонентов), например

```
xkb_keymap "ru" {
  xkb_keycodes        { include "xfree86"          };
  xkb_types           { include "default"          };
  xkb_compatibility   { include "default"          };
  xkb_symbols         { include "en_US(pc105)+ru"  };
  xkb_geometry        { include "pc(pc102)"        };
};
```

Обратите внимание, что в одной инструкции include может быть указано несколько файлов (блоков) через знак +. Понятно, что это означает, что должны быть вставлены последовательно все указанные файлы.

Таким образом, в файле конфигурации X-сервера можно вместо пяти компонентов указать сразу один из готовых наборов keymap, например

```
Option "XkbKeymap"      "xfree86(ru)"
```

Кроме того, эти два способа можно комбинировать. Например, если вы выбрали один из подходящих наборов keymap, но вас не устраивает один из компонентов, например geometry, то в файле конфигурации можно указать

```
Option "XkbKeymap"      "xfree86(ru)"
Option "XkbGeometry"    "pc(pc104)"
```

При этом, в соответствии с первой инструкцией, все компоненты будут взяты из keymap "xfree86(ru)", а вторая инструкция "перепишет" geometry, не затрагивая остальные компоненты.

Третий способ несколько отличается от предыдущих. Набор настроек можно указывать не перечислением компонентов, а с помощью задания "правил" (Rules), "модели" (Model), "схемы" (Layout), "варианта" (Variant) и "опций" (Option).

В этом наборе только Rules представляют собой некий файл (эти файлы тоже находятся в отдельном подкаталоге rules каталога /usr/X11R6//lib/X11/xkb), в котором находится таблица правил - "как выбрать все пять компонентов настроек XKB в зависимости от значений Model, Layout и т. д.". Все остальные параметры представляют собой просто "ключевые слова":

- Model обычно определяет тип "железа" -- клавиатуры;
- Layout — язык или, точнее, алфавит, который "навешивается" на кнопки клавиатуры;
- Variant — различные варианты размещения знаков алфавита (заданных Layout);
- Options — обычно меняет "поведение" или "расположение" модификаторов Control и Group (переключатель групп — это переключатель "языка", например, русский/латинский).

По этим словам модуль XKB при старте ищет в таблицах "правил" подходящие файлы настроек (keycodes, types, compat, symbols и geometry). Другими словами, Rules определяет некоторую функцию, аргументами которой являются Model, Layout, Variant и Options, а значение, которое возвращает эта функция, представляет собой полный набор из компонентов настроек XKB — keycodes, types, compat, symbols и geometry (или полная keymap).

Итак, если вы используете третий способ указания конфигурации XKB, то в файле конфигурации X-сервера надо задать параметры `XkbRules`, `XkbModel`, `XkbLayout` и, если вам нужно что-то не совсем стандартное — xkbvariant и XkbOptions.

Например,

```
Option "XkbRuies"          "xfree86"
Option "XkbModei"          "pc104"
Option "XkbLayout"         "ru"
Option "XkbVariant"        ""
Option "XkbOptions"        "ctrl:ctrl_ac"
```

означает, что модуль XKB должен в соответствии с правилами, описанными в файле ./rules/xfree86, выбрать настройки для клавиатуры типа `"pc104"` (104 кнопки), русского алфавита (английский алфавит будет включен "по умолчанию"), вариант -• "стандартный" (то есть, этот параметр можно было не писать) и, наконец, дополнительные опции для вашей "раскладки клавиатуры" - - `"ctrl:ctrl_ac"`.

Что означают различные опции, а также какие "модели" и "схемы" определены в "правилах" (и что они означают), можно посмотреть в файле xfree86.lst (или другом lst-файле, если вы выбрали "правила", отличные от xfree86), который находится в том же каталоге, что и файл "правил", т. е. в подкаталоге rules.

Небольшое отступление о клавише-переключателе "рус/лат". В первых вариантах модуля XKB раскладка "русской" клавиатуры включала в себя и "переключатель групп" — рус/лат, "подвешенный" на клавишу <Caps Lock>. С одной стороны, это было удобно: в простейшем случае достаточно было выбрать "русскую раскладку" и вы автоматически получали и клавишу для переключения "на русский". Но, с другой стороны, это было неудобно для тех, кто предпочитает в качестве переключателя "рус/лат" другую клавишу (или комбинацию клавиш). Конечно, выбрать другой переключатель не составляло труда, но при этом оставался и переключатель на <Caps Lock>, что многим не нравилось. Для того чтобы убрать его, надо было "залезть" в соответствующий файл и вручную подправлять соответствующую раскладку.

В конце концов (начиная с версии 3.3.4) сами разработчики Xfree86 убрали этот "переключатель" из "русской раскладки". Но, в связи с этим появились и некоторые проблемы — теперь клавишу-переключатель надо явно "заказывать" при конфигурировании XKB.

Несколько практических рекомендаций по настройке модуля XKB

Самый простой способ — использовать программу для автоматической настройки X Window. В XFree86 версии 3 такая программа называется XF86Setup. Она использует третий метод задания конфигурации XKB. При этом "по умолчанию" используются "правила" (`XkbRules`) — xfree86. Вам нужно будет только выбрать "модель" (`XkbModel`), "схему" (xkbLayout) и "способ переключения групп" (переключатель "рус/лат").

Кроме того, при желании вы можете изменить "положение клавиши <Ctrl>". Естественно, в конфигурации это будет выглядеть как соответствующие строчки xkboptions. Итак, запустите программу XF86Setup, выберите раздел Keyboard. В этом разделе выберите нужные варианты из меню **Model** (Тип клавиатуры) и **Layout** (Язык). Не забудьте отметить в отдельных списках (в правой части) подходящий "переключатель групп" и, если хотите — "расположение <Ctrl>". При выходе из программы она запишет соответствующие строчки в файл конфигурации Xfree86 в секции Keyboard.

А теперь рассмотрим то, как можно задать эти настройки путем прямого редактирования секции `InputDevice` (Keyboard) файла /etc/X11/XF86Config.

Прежде всего, надо сказать, что "ключевыми словами" в этих настройках будут:

- xfree86 — название "архитектуры" X Window;
- pc101 (pc104, pc105 и т. п.) — тип клавиатуры (количество кнопок);
- ru — название "раскладки клавиатуры" с русским алфавитом.

Проще всего сразу задать конфигурацию клавиатуры с помощью keymap. В файлах конфигурации есть набор "полных keymap" для архитектуры xfree86, отличающихся "языком". Все они лежат в файле xfree86, а название блока внутри файла отражает название "языка" (точнее — алфавита) – xfree86(us), xfree86(fr), xfree86(ru) и т. д. Полный список keymap-файлов можно посмотреть в файле /usr/X11R6//lib/X11/xkb/keymap.dir.

Для "русифицированной" клавиатуры вполне подойдет

```
Option "XkbKeymap"        "xfree86(ru)"
```

К сожалению, после исключения клавиши <Caps Lock> как переключателя "рус/лат" из русской раскладки (см. замечание в конце предыдущего раздела) получилось так, что "полная keymap" для русского языка осталась вообще без какого-либо переключателя "по умолчанию". Но вы можете добавить его вручную. Для этого придется найти в файле /keymap/xfree86 блок ru. И дописать в строчку `xkb_symbols` ссылку на описание соответствующего переключателя групп. Для клавиши <Caps Lock> это будет – `group(caps_toggle)`. То есть, строка `xkb_symbols` будет выглядеть как

```
xkb_symbols      { include "en_US(pc105)+ru+group(caps_toggle)"};
```

Полный список возможных переключателей групп (то есть возможных переключателей "рус/лат") можно найти в файле /usr/X11R6//lib/X11/xkb/symbols/group (проведите в этом файле поиск по ключевому слову `xkb_symbols`).

Теперь рассмотрим случай, когда для задания конфигурации клавиатуры используется третий способ — через "правила", "модель", "схему" и т. д. Как было сказано выше:

- название "правил" (rules) соответствует "архитектуре" (xfree86);
- "модель" (model) соответствует типу клавиатуры (pc101, pc102 и т. п.);
- "схема" (layout) отражает "язык" (ru).

Поэтому, подходящая конфигурация будет выглядеть примерно так:

```
Option "XkbRules"     "xfree86"
Option "XkbModel"     "pc104"
Option "XkbLayout"    "ru"
```

С помощью строки xkbOptions можно подобрать "поведение" управляющих клавиш. Возможные значения xkbOptions и их смысл можно подсмотреть в файле /rules/xfree86.lst в той части, которая начинается строкой `"! option"`.

Не забудьте, что, как и в предыдущем случае, надо явно выбрать переключатель групп. Для клавиши <Caps Lock> это будет

```
Option "XkbOptions"  "grp:caps_toggle"
```

И, наконец, рассмотрим первый способ — описание отдельных компонентов настройки (keycodes, compat, types, symbols, geometry).

Если вы не знаете с чего начать, подсмотрите соответствующий набор в keymap. Или попробуйте "вычислить" его через rules/model/layout. Чаще всего подойдут следующие значения:

- для keycodes выбрать файл xfree86;
- для types и compat подойдут файлы default ("по умолчанию") или complete ("полная");
- geometry, скорее всего, "pc", а количество кнопок задается названием блока в файле pc — pc(pc101), pc(pc102), pc(pc104). Полный список "геометрий" имеется в файле /usr/X11R6/lib/X11/xkb/geometry.dir.

А вот на symbols обратите особое внимание. Файл symbols/ru описывает только "буквенные" клавиши. Если вы укажете только его, то у вас не будут работать все остальные клавиши (включая <Enter>, <Shift>, <Ctrl>, <Alt>, <F1>—<F12> и т. д.). Поэтому symbols должен состоять по крайней мере из двух файлов — en_US(pcl01) (в скобках — тип клавиатуры) и, собственно, га. Полный список symbols — в файле /usr/X11R6/lib/X11/xkb/symbols.dir.

Сюда же надо добавить и описание подходящего переключателя "рус/лат" (как уже говорилось, их перечень — в файле symbols/group).

Для первого метода список может выглядеть так

```
Options "XkbKeycodes"        "xfree86"
Options "XkbTypes"           "complete"
Options "XkbCompat"          "complete"
Options "XkbSymbols"         "en_US(pc101)+ru+group(alt_shift_toggle)"
Options "XkbGeometry"        "pc(pc101)"
```

Если вам хочется задать дополнительные изменения "поведения" управляющих клавиш (то, что в третьем методе задается XkbOptions), то подсмотрите подходящую "добавку" в rules/xfree86.1st и "приплюсуйте" ее в строчку XkbSymbols. Например,

```
XkbSymbols      "en_US(pc101)+ru+group(shift_toggle)+ctrl(ctrl_ac)"
```

На этом мы ограничим описание методов настройки клавиатуры, а точнее — настройки модуля XKB. Если вы хотите разобраться с этим детальнее, то обратитесь к исходному материалу И. Паскаля *(см. [П11.6]приложения)*.

9.4. Мышь

Существуют два основных типа мышей — подключаемые через последовательный порт (serial mice) и подключаемые к шине (bus mice). Большинство компьютеров оборудуются в настоящее время мышами второго типа. Даль-

нейший текст относится к bus-мышам и основан на Busmouse HOWTO Криса Багвелла (Chris Bagwell), версии 1.91 от 15 июня 1998 г.

9.4.1. Определение типа мыши

Вы должны знать две важных характеристики своей мыши: какой у нее интерфейс и какой она использует протокол.

Интерфейс — это совокупность аппаратных параметров мыши, включающая такие параметры, как используемые мышью прерывания, порты ввода/вывода и количество контактов в разъеме. Ядро Linux поддерживает 4 типа интерфейсов bus-мыши: Inport (Microsoft), Logitech, PS/2 и ATI-XL. Не существует однозначного алгоритма определения типа интерфейса мыши.

Мыши типа Inport обычно подключаются к интерфейсной карте на материнской плате. Если разъем, который подключается к интерфейсной карте, круглый, имеет 9 контактов и желобок (направляющую выемку) с одной стороны, то вполне возможно, что у вас мышь типа Inport. Если только не Logitech, поскольку эти мыши внешне имеют те же характеристики. Различить их можно только если у вас сохранилась упаковка или руководство, в котором указан тип мыши.

Мыши типа PS/2 подключаются не к плате расширения, а к специальному разъему (PS/2 Auxiliary Device port) на контроллере клавиатуры. Этот разъем имеет 6 контактов (6-pin mini DIN connector), и похож на разъем для подключения клавиатуры.

Мыши типа ATI-XL — это вариант мышей типа Inport. Они подключаются к комбинированной карте, являющейся видеоадаптером и контроллером мыши. Если только вы не знаете точно, что у вас видеоадаптер ATI-XL (и следовательно мышь ATI-XL), то, скорее всего, у вас мышь другого типа.

Протокол — это чисто программная характеристика мыши. Большинство мышей Inport, Logitech и ATI-XL используют протокол BusMouse, а мыши типа PS/2 используют протокол PS/2.

9.4.2. Конфликты по прерываниям

Сначала надо определить, какое прерывание использует ваша мышь, и убедиться, что она не конфликтует с каким-нибудь другим устройством. Этот момент очень важен, потому что под Linux мышь не может использовать одно и то же прерывание с каким-либо другим устройством, даже если все прекрасно работает под управлением другой ОС. Так что проверьте документацию на все подключенные у вас периферийные устройства, чтобы знать, какие прерывания они используют!

Список занятых (используемых) на данный момент прерываний можно получить, выполнив команду

```
[user]$ cat /proc/interrupts
```

или просмотрев файл /proc/interrupts.

В большинстве случаев IRQ4 используется первым последовательным портом (/dev/ttyS0), IRQ3 — вторым последовательным портом (/dev/ttyS1, предполагается, что у вас есть такие устройства, если нет — вы можете использовать их IRQ). IRQ5 используется некоторыми SCSI-устройствами, а IRQ12 — некоторыми сетевыми картами. Если ваша сетевая карта использует IRQ12, а ваша мышь — типа PS/2, то у вас будут проблемы, поскольку вы вынуждены будете использовать IRQ12 только для порта PS/2. Для мышей ATI-XL, Inport и Logitech ядро по умолчанию использует прерывание IRQ5, так что если вы не хотите перекомпилировать ядро, вам придется использовать для мыши именно это прерывание. Впрочем, последние версии ядра позволяют задать опции командной строки, определяющие прерывание, которое будут использовать мыши типа Inport и Logitech. Мыши типа PS/2 всегда используют прерывание IRQ12, и не существует способа изменить это, так что в случае конфликтов надо перенастраивать другие периферийные устройства.

9.4.3. Настройка мыши

Далее необходимо проверить настройки в некоторых конфигурационных файлах. Вначале убедитесь, что существует файл /etc/sysconfig/mouse и что в нем записано что-то вроде:

```
MOUSETYPE="Microsoft"
XMOUSETYPE="Microsoft"
XEMU3=yes
```

Естественно, что тип мыши должен соответствовать вашей мыши, у меня, например, это PS/2.

Чтобы вырезать и вставлять куски текста в консоли, должен быть установлен сервер мыши gpm.

Проверьте, что сервер мыши gpm запущен, для чего дайте команду:

```
[user]$ ps -A I grep gpm
```

Если в результате вы получите непустую строку, то драйвер работает. Если же процесс gpm не найден, надо проверить наличие скрипта /etc/rc.d/init.d/gpm, в котором должна найтись строка вызова демона gpm. Эта строка может иметь примерно такой вид:

```
daemon gpm -t $MOUSETYPE -d 2 -a 5 -B 132 # two-button mouse
```

(смысл параметров см. на странице man gpm).

Если сервер gpm работает, то выделять и вставлять куски текста можно следующим образом. Нажмите левую кнопку и выделяйте текст. Когда дойдете до конца нужного куска текста, отпустите кнопку. Потом нажмите правую кнопку в том месте, где вы хотите осуществить вставку. Можно даже в другой виртуальной консоли. То же самое можно проделать в X Window, но для вставки нужно нажимать среднюю клавишу, или обе, если у вас двухкнопочная мышь.

9.5. Жесткий диск

9.5.1. Нумерация

Сначала приведем дополнительные сведения о нумерации жестких дисков в системе Linux (табл. 9.3).

Таблица 9,3. *Нумерация жестких дисков*

Тип жесткого диска	Старший номер	Наименование устройства	Младший номер Диск 1	Диск 2
IDE на 1 контроллере	2	/dev/hda и /dev/hdb	0-63	64-127
IDE на 2 контроллере	33	/dev/hdc и /dev/hdd	0-63	6 –127
SCSI	8	/dev/sd	0-15	16-31

Первые жесткие диски на обоих IDE-контроллерах получают младшие номера 0, а вторые диски — номера 64 (и имена /dev/hda и /dev/hdb). По этим номерам происходит обращение ко всему диску в целом, независимо от количества разделов на этом диске. Разделы на дисках получают, соответственно, следующие по порядку номера: первый раздел на первом диске — номер 1 и имя /dev/hda1, второй раздел — номер 2 и имя /dev/hda2 и т. д.

Более двух дисков на IDE-контроллере не бывает, и более двух контроллеров на персональные компьютеры обычно не ставят, так что всего бывает не более 4 IDE-дисков (/dev/hd[a-d]).

Аналогично, целые SCSI-диски получают младшие номера 0, 16, 32 и т. д., и имена /dev/sd[a-g]. Разделы на этих дисках получают младшие номера, следующие по порядку за младшим номером всего диска: первый раздел на диске /dev/sda получает младший номер 1 (и имя /dev/sda1), первый раздел на диске /dev/sdb получает младший номер 17 (и имя /dev/sdb1), и т. д.

9.5.2. Форматирование жесткого диска

Низкоуровневое форматирование жесткого диска под Linux невозможно. Впрочем, в этом нет особой необходимости, поскольку современные диски выпускаются отформатированными на низком уровне.

Форматирование на высоком уровне заключается в создании на диске разделов и файловой системы. Для создания разделов под Linux используются программы fdisk, cfdisk и sfdisk. Как сообщает man-страница к программе fdisk, программа cfdisk позволяет создать качественную таблицу разделов, но имеет некоторые ограничения. Программа fdisk, хотя и позволяет произвести разбиение диска в большинстве случаев, но содержит несколько ошибок. Ее главное преимущество в том, что она поддерживает разделы DOS, BSD и других систем. Программа sfdisk работает более корректно, чем fdisk, и она гораздо мощнее и fdisk, и cfdisk, но имеет неудобный пользовательский интерфейс. Так что man-страница рекомендует пытаться применять эти программы в следующем порядке: cfdisk, fdisk, sfdisk.

Однако, на мой взгляд, интерфейс любой из этих программ не предназначен для начинающих пользователей. Поэтому для создания разделов на диске я бы рекомендовал использовать программу Partition Magic фирмы Power Qwest. Несколько слов о ней было сказано в *гл. 2,* как и о разбиении диска, поэтому на этом вопросе более не будем останавливаться.

После разбиения диска на разделы надо создать файловую систему в разделах, предназначенных для использования под Linux,. Для этого используется команда `mkfs`. С ее помощью можно создать не только файловую систему типа ext2fs, но и файловые системы других типов. Типичный пример запуска этой команды:

```
[root]# mkfs -t тип /dev/hda3
```

где тип - тип создаваемой файловой системы, например, ext2, а `/dev/hda3` — указание форматируемого раздела диска.

Чтобы использовать mkfs, необязательно иметь права суперпользователя, достаточно иметь право записи в файл соответствующего устройства.

Внимание

Команда mkfs очень опасна! Она перезаписывает область диска, в которой хранятся inodes. Так что если вы ошибетесь в указании раздела диска, вы можете уничтожить ценные для вас данные.

После создания файловой системы ее надо смонтировать в общее дерево каталогов. Делается это с помощью команды mount, которую мы уже рассматривали, так что повторяться не стоит. Единственное, что можно отметить, так это то, что смонтировав первый раз диск или раздел, в котором вы только что создали файловую систему, вы увидите, что она пуста, т. е. не

содержит никаких файлов и каталогов, кроме единственного каталога с именем lost+found. Этот каталог должен существовать в каждой файловой системе, поскольку он выполняет служебную роль: при проверке файловой системы командой fsck в этом каталоге собираются "потерянные" файлы и подкаталоги. О команде fsck мы еще поговорим, но до этого рассмотрим кратко вопрос об оптимизации работы жесткого диска, которая выполняется **с помощью Команды** hdparm.

9.5.3. Команда *hdparm*

Команда hdparm служит для того, чтобы получить или установить некоторые параметры IDE-интерфейса жесткого диска(ов). С помощью этой команды можно попытаться оптимизировать работу с жестким диском. Однако имейте в виду, что команда эта *не безопасна.* Если задать значение параметра, которое не поддерживается аппаратным обеспечением, можно потерять данные на диске.

Данные о том, в каких случаях стоит использовать эту команду, весьма противоречивы. Одни авторы *(см. [П11.8]приложения)* уверяют, что с ее помощью добиться чуть ли не 10-кратного увеличения скорости обмена данными с жестким диском, другие *(см. [П1.5] приложения)* утверждают, что можно потратить много времени на настройку интерфейса жесткого диска, а в результате получить лишь незначительное увеличение скорости работы. По-видимому, получаемый выигрыш определяется, в основном, моделью жесткого диска, так что сами решайте, имеет ли смысл этим заниматься. Но, как утверждает Rob Flickenger*(см. [П11.8]приложения),* если у вас IDE- или EIDE-диск, выпущенный не более 2 лет назад, то заняться оптимизацией стоит!

Экспериментировать с этой командой рекомендуется, получив права суперпользователя, причем лучше, если система запущена в однопользовательском режиме. Поскольку существует опасность потери данных, перед использованием команды hdparm необходимо сделать резервную копию ценной информации, и каждый раз перед ее запуском выполнять команду sync, чтобы сбросить на диск данные, находящиеся временно в буферах.

Формат запуска команды прост:

```
[user]$ hdparm опция устройство
```

Если после указания опции не указывать нового значения для соответствующего параметра, то будет просто выдано его действующее значение. А если запустить команду без указания опций вообще, то будут выведены значения основных параметров, действующих при работе с данным устройством (IDE-диском). Вывод выглядит примерно следующим образом:

```
[user]$ hdparm /dev/hda
/dev/hda:
   multcount    = 0 (off)
```

```
I/O support  =  0  (default 16-bit)
unmaskirq    =  0  (off)
using_dma    =  0  (off)
keepsettings =  0  (off)
nowerr       =  0  (off)
readonly     =  0  (off)
readahead    =  8  (on)
geometry     = 1870/255/63, sectors = 30043440, start = 0
```

Обычно это значения, устанавливаемые по умолчанию. Как видите, большинство возможностей просто отключено. Это и естественно, поскольку разработчики дистрибутивов выбирают такие значения параметров, при которых будут работать любые типы дисков. А уж об оптимизации параметров для вашего диска придется позаботиться вам самим!

Для начала посмотрим, какую еще информацию о диске и параметрах интерфейса можно получить с помощью команды `hdparm`.

- Опция -i позволяет получить информацию о модели жесткого диска, его серийном номере и некоторых других параметрах.
- Опция -д выводит информацию о геометрии диска (число цилиндров/головок/секторов), его размере (число секторов) и начальном смещении (номер сектора, с которого начинается используемое пространство, обычно 0).
- Опция -т позволяет протестировать скорость обмена данными с кэшем (то есть скорость работы подсистемы "память — ЦПУ — буфер кэш").
- Опция `-e` служит для тестирования скорости непосредственно записи на диск (а не в кэш-память).

Поскольку опции в команде можно комбинировать, давайте выполним команду

```
[root]# hdparm -Tt /dev/hda
```

Результатом будут примерно такие строки (для вашего диска цифры, конечно, будут другими):

```
/dev/hda:
 Timing buffer-cache reads:   128 MB in  1.34 seconds = 95.52 MB/sec
 Timing buffered disk reads:   64 MB in 17.86 seconds =  3.58 MB/sec
```

Эти данные позволяют судить о производительности подсистемы ввода/вывода вашего жесткого диска. Если ваш диск не очень давнего выпуска, можно попытаться поднять производительность этой подсистемы, используя следующие опции команды hdparm.

- С помощью опции -с вы имеете возможность воздействовать на формат обмена данными между процессором и жестким диском. По умолчанию ис-

пользуется значение 16 бит, но если после -c указать 1 или 3, то произойдет переключение на один из режимов 32-битной передачи. Режим, имеющий номер 3, считается более надежным, хотя и работает чуть медленнее.

- Используя опцию `-d 1`, можно активизировать режим DMA (Direct Memory Access — прямой доступ к памяти). Однако, чтобы это имело смысл, ядро должно быть скомпилировано с поддержкой DMA.
- Опция -т задает многосекторный режим ввода/вывода. Число после опции указывает максимальное количество секторов, которые можно передать вместе. Это ускоряет передачу больших файлов. Максимально допустимое значение этого параметра для вашей модели жесткого диска указано под ключевым словом `MaxMultSect` в перечне параметров, выдаваемых командой hdparm `-i`.
- С помощью опции -p можно настраивать режим PIO (программируемый ввод/вывод). Чем выше число (можно использовать значения в интервале от 0 до 5), тем быстрее осуществляется передача данных. Повышение этого параметра ничего не дает при работе с медленным диском, но повышает опасность потери данных, так что подходите к его использованию осторожно.
- Включение опции -и приводит к тому, что снимается запрет на обработку других прерываний, установленный на время обработки дискового прерывания. Это означает, что во время ожидания запрошенных с диска данных ядро сможет принять и обработать другие запросы (например, полученные от сетевых устройств или модема). Это должно положительно сказаться на общей производительности системы, но не все аппаратные конфигурации способны работать в таком режиме!

Давайте попробуем осторожно улучшить настройки подсистемы ввода/вывода данных для жесткого диска. Изменяйте значения параметров по одному, и после каждого изменения проверяйте результат с помощью команды

```
hdparm -tT /dev/hda.
```

В случае зависания компьютера или каких-то других неприятностей (вы к ним готовы, не так ли!), перезагружайте компьютер (снова в однопользовательском режиме) и пробуйте другое значение для параметра, изменявшегося последним. Поскольку задаваемые вами установки нигде не зафиксировались, вы будете каждый раз начинать с одной и той же точки. Приведу для иллюстрации тот результат, на котором я решил остановить эксперименты:

```
[root]# hdparm -X66 -d1 -u1 -m16 -c3 /dev/hda
/dev/hda:
    setting 32-bit I/O support flag to 3
    setting multcount to 16
    setting unmaskirq to 1 (on)
    setting using_dma to 1 (on)
```

```
   setting xfermode to 66 (UltraDMA mode2)
   multcount    = 16 (on)
   I/O support  =  3 (32-bit w/sync)
   unmaskirq    =  1 (on)
   using_dma    =  1 (on)
[root]# hdparm -tT /dev/hda
/dev/hda:
 Timing buffer-cache reads:   128 MB in  1.43 seconds =89.51 MB/sec
 Timing buffered disk reads:   64 MB in  3.18 seconds =20.13 MB/sec
```

Как видите, скорость обмена данными возросла у меня примерно в 6 раз!

Как уже было сказано, после выполнения команды hdparm с любым набором опций вновь установленные значения действуют только в текущем сеансе работы системы, а после перезагрузки оптимизированные установки будут потеряны. Поэтому после завершения экспериментов надо еще записать вызов команды с подобранными значениями опций в один из системных скриптов загрузки, например, в /etc/rc.d/rc.sysinit (в ALT Linux Junior 1.0 для настройки IDE-дисков имеется специальный скрипт /etc/rc.d/scripts/idetune). Желательно перед этим убедиться, что система ведет себя стабильно, и даже выполнить команду проверки состояния файловой системы на данном устройстве *(см. разд. 9.5.4)*.

В заключение заметим, что кроме опций, влияющих на производительность подсистемы ввода/вывода (имейте в виду, что далеко не все они были рассмотрены выше), команда hdparm имеет еще ряд опций, позволяющих управлять энергопотреблением и другими характеристиками дисковой подсистемы. Полный список всех опций команды hdparm смотрите на соответствующей man-странице (man 8 hdparm).

9.5.4. Команда *fsck*

Описание команды `fsck`, наверное, правильное было бы отнести к разделу о файловых системах, но, поскольку ее предназначение состоит в том, чтобы по возможности восстановить работоспособность дисковой подсистемы, мы рассматриваем ее в разделе о настройке жесткого диска.

Основная функция команды fsck заключается в восстановлении логической непротиворечивости файловой системы, созданной в разделе жесткого диска. При выполнении этой команды производится поиск следующих ошибок:

- сектора, которые используются одновременно двумя файлами;
- сектора, которые включены в список свободных секторов, хотя они содержат часть какого-то файла;

- сектора, которые не содержат информации, но не включены в список свободных секторов;
- индексные дескрипторы файлов (modes), не указанные ни в одном каталоге;
- неверная общая информация в суперблоке и т. д.

Формат запуска команды следующий:

```
[root]# fsck [опции] [-t fstype] [--fs-options] filesystem
```

где fstype — тип проверяемой файловой системы, а в качестве filesystem можно указать либо имя устройства (например, /dev/hda4), либо точку монтирования (/, /opt, /mnt/wint). В man-странице по fsck сказано, что можно еще использовать метку (label) файловой системы, либо UUID, но, что такое два последних варианта, я пояснить не берусь.

Вообще говоря, команда fsck не является самостоятельной утилитой, она просто предоставляет единый интерфейс вызова специализированных программ для проверки файловых систем разных типов. Эти программы называются fsck.fstype (например, fsck.ext2) и команда fsck при запуске производит поиск соответствующей специфической программы сначала в /sbin, затем в /etc/fs и /etc, и, наконец, в каталогах, перечисленных в переменной PATH. Опции, указанные после двойного дефиса, передаются команде fsck.fstype.

Из собственных опций команды fsck (они указываются сразу после имени) стоит отметить опции -A, -a, -r и -N. Если указать опцию *-a,* то при обнаружении ошибок в файловой системе будет производиться их автоматическое исправление. Указание опции -A приводит к тому, что команда просмотрит файл /etc/fstab и за один прогон проверит все перечисленные в нем файловые системы. Опция -r переключает команду в интерактивный режим работы, т. е. перед тем, как произвести какие-то изменения, будет выдаваться запрос на подтверждение действия. Задание опции -N приводит к тому, что никаких изменений в файловой системе производиться не будет, будет только сказано, что должно быть сделано.

При загрузке ОС Linux в некоторых случаях происходит автоматический запуск команды fsck. Основных причин для выполнения fsck на этапе загрузки системы две: некорректный выход из системы в предыдущий раз (например, резкое отключение питания или неисправность аппаратуры) и достижение заданного порога для количества выполнений операции размонтирования файловой системы. При каждом выполнении операций монтирования/размонтирования файловой системы в ее суперблоке *(см. гл. 16)* делаются специальные отметки. Если последнее размонтирование завершилось корректно, то файловая система помечается как "чистая" (clean) и число операций монтирования/размонтирования увеличивается на единицу. Если корректного размонтирования не было, то файловая система помечена

как "грязная" (dirty). На этапе загрузки ОС проверяется, являются ли все файловые системы "чистыми", и если нет, то для "грязных" систем выполняется команда fsck. Кроме того, даже если все системы "чистые", но достигнут порог для числа операций монтирования/размонтирования, запускается та же команда с опцией -А. Как было сказано выше, при этом производится проверка всех файловых систем, перечисленных в файле /etc/fstab. Порядок, в котором проверяются файловые системы, определяется числами, указанными в последнем поле каждой строки файла /etc/fstab. Файловые системы проверяются в порядке возрастания этих номеров. Первым всегда следует проверять корневой раздел. Если две файловые системы расположены на разных дисках, им может быть присвоен один и тот же номер. Это приводит к тому, что они будут проверяться одновременно, а, значит, сократится общее время проверки. Но если файловые системы расположены на одном и том же физическом устройстве (например, в разных разделах), то совпадение номеров вызовет только замедление процедуры проверки, т. к. головки диска должны будут совершать лишние перемещения.

Иногда требуется запустить fsck и вручную. При этом лучше всего предварительно перевести систему в однопользовательский режим и размонтировать проверяемые файловые системы (или смонтировать их в режиме "только для чтения"). Например, запуск fsck в разделе /usr обычно требуется тогда, когда файловая система разрушена и тогда любые дальнейшие действия в разрушенной системе могут привести к полному краху, а, значит, fsck должна быть запущена как можно скорее. Обычно о необходимости перехода в однопользовательский режим говорит также то, что fsck не может автоматически восстановить файловую систему при загрузке. Такое случается относительно редко, обычно при выходе из строя жесткого диска или при попытках установить какую-либо экспериментальную версию ядра, но все же об этом надо знать, чтобы не растеряться в затруднительной ситуации.

К сожалению, в процессе восстановления файловой системы приходится полностью полагаться на возможности программы fsck. Начинающему пользователю не стоит самостоятельно пытаться произвести какие-то действия в поврежденной файловой системы, потому что вы рискуете перевести ядро в паническое состояние (kernel panic).

Если fsck обнаруживает "потерянные файлы", т. е. такие файлы, которые не указаны ни в одном из каталогов, она помещает их в каталог lost+found на верхнем уровне проверяемой файловой системы. Поскольку имена файлов регистрируются только в родительском каталоге, то в данном случае их "истинные" имена неизвестны, и команда присваивает им имена, совпадающие с номерами их индексных дескрипторов.

9.6. Принтер

9.6.1. Традиционные средства печати UNIX

Исторически для печати в UNIX-системах существовали две системы печати: LPD (Line Printer Daemon) [RFC1179], разработанная для Berkeley UNIX (или BSD-система), и AT&T Line Printer system. Эти системы печати были созданы в 70-х годах для печати текстов на построчно-печатающих (линейных) принтерах. Принимая во внимание, что аппаратные средства печати (проще говоря, принтеры) с тех пор существенно изменились, можно было бы предположить, что существенно переработаны и программные средства для управления печатью. Однако, этого не произошло. Хотя и были созданы различные улучшенные системы печати (LPRng, Palladin, PLP), однако ни одна из этих новых разработок не изменяла фундаментальные возможности этих систем. Впрочем, как показало время и практика, возможности этих систем вполне достаточны и при небольших доработках удовлетворяют и современные потребности.

Как и во всех UNIX-системах, в Linux файл, предназначенный для печати, вначале пересылается во временную область (проще говоря, временный каталог), которая называется областью спулинга. Дело в том, что принтеры являются относительно медленными устройствами, и система заботится о том, чтобы не задерживать работу на время распечатки файла. Фоновый процесс — демон печати — постоянно сканирует область спулинга в ожидании файлов, предназначенных для печати. Для каждого принтера, подключенного к системе, заводится своя область спулинга. Таким образом, область спулинга представляет собой очередь заданий на печать, дожидающихся того момента, когда освободится соответствующий принтер и демон печати отправит данное задание на печать (в фоновом режиме).

В основу подсистемы печати в Linux положена BSD-система — LPD, а точнее, доработанный вариант этой системы LPRng. LPRng состоит из отдельных программ, которые обеспечивают выполнение отдельных функций подсистемы печати.

- lpd — демон системы печати. Обычно запускается на этапе загрузки системы из файла rc, но может быть запущен и пользователем.
- lpr — пользовательская команда печати. Программа lpr принимает подлежащие печати данные и помещает их в буферный каталог, где их находит lpd и выводит на печать. Программа lpr — единственная, которая может ставить новые задания в очередь печати. Другие программы, которым необходимо использовать печать, обращаются для этого к lpr.
- lpq — программа, позволяющая просматривать очередь заданий, ожидающих печати на указанном принтере.

- lpc — команда контроля системы lpd. С помощью lpc можно отключать принтеры, останавливать или переупорядочивать очереди печати и т. п. Некоторые из функций этой команды доступны пользователям, но в основном это средство для администратора.
- lprm — эта программа позволяет удалить одно или несколько заданий из очереди печати. При этом стираются соответствующие файлы данных и из системы печати удаляются все ссылки на них.

Взаимодействие lpr и других программ этого набора с демоном lpd осуществляется с использованием сетевых средств, так что они могут запускаться и на других компьютерах. Рассмотрим вкратце, как осуществляется печать файла в системе LPD.

Когда вызывается программа lpr, она первым делом выбирает принтер, на который будет производиться печать. Этот выбор определяется либо параметром командной строки `-Pprinter`, либо значением переменной окружения PRINTER, либо же используется общесистемный принтер, заданный по умолчанию (это принтер с именем lp). Как только lpr узнает, на какой принтер отправлять текущее задание, она ищет описание этого принтера в базе данных об имеющихся принтерах, которая хранится в файле /etc/printcap. Из этой базы lpr получает имя каталога (области спулинга), в который следует помещать задания для найденного принтера. Обычно этот буферный каталог имеет имя /var/spool/lpd/*printer*. Такой каталог (область спулинга) должен существовать, если вы хотите печатать на принтере с именем *printer*.

Для каждого задания программа lpr помещает в буферный каталог два файла: cfxxx и dfxxx, где xxx — номер текущего задания. Файл cfxxx содержит справочную информацию и информацию об обработке задания. Источником этих сведений являются командная строка запуска программы, переменные среды процесса, который запустил эту программу, и глобальная конфигурация системы. Файл dfxxx содержит данные, подлежащие печати.

После постановки задания в очередь lpr уведомляет демона lpd о появлении задания на печать. Взаимодействие lpr с lpd происходит через именованный сокет /dev/printer. Демон lpd тоже обращается к файлу /etc/printcap, чтобы узнать, какой принтер должен использоваться для печати, и является он локальным или удаленным. Если в /etc/printcap указано, что принтер подключен локально, lpd проверяет наличие демона печати, обрабатывающего соответствующую очередь. Дело в том, что для обработки каждой очереди lpd создает отдельную копию самого себя. Если такой копии еще не имеется, она создается и ей передается обработка очереди. Если соответствующий принтер подключен к другой машине, lpd устанавливает соединение с демоном lpd удаленной машины и пересылает туда файл данных и управляющий файл.

Обслуживание заданий печати осуществляется по правилу "первым пришел — первым обслужен" (FIFO). Системный администратор может при желании изменить порядок печати с помощью программы lpc.

Начиная с ядра 2.1.33 устройство lp является клиентом нового устройства parport. Введение parport решает некоторые проблемы, связанные с lp, — теперь можно разделять параллельные порты с другими драйверами, динамически связывать порты с устройствами, не устанавливая жесткого соответствия между адресами I/O и номером порта и т. д. Подробнее об этом вы можете прочитать в статье *(см. [П11.11] приложения)*.

9.6.2. Файл /etc/printcap

Файл /etc/printcap — это главная база данных системы печати LPD. Принтер будет получать задания на печать только в том случае, если он (принтер) описан в этом файле. Поэтому для получения возможности использовать принтер вы должны добавить очередь печати к lpd, т. е. создать новый буферный подкаталог в каталоге /var/spool/lpd, и добавить соответствующую запись в файл /etc/printcap.

Запись в файле /etc/printcap выглядит примерно так:

```
# ЛОКАЛЬНЫЙ djet500
lp|dj|deskjet:\
        :sd=/var/spool/lpd/dj:\
        :mx#0:\
        :lp=/dev/lp0:\
        :sh:
```

Каждый элемент файла /etc/printcap начинается со строки, задающей имена принтера (их может быть несколько), разделяемые вертикальной чертой. Затем следует ряд параметров конфигурации, разделенных двоеточиями (обычно каждый параметр заключается в двоеточия с обеих сторон). Каждый параметр имеет вид `xx=строка` или `xx#` тесло, где xx — двухсимвольное имя параметра, а *строка* и число — присваиваемые ему значения. Если никакого значения не присваивается, переменная является булевой, и ее присутствие означает "истина". Допускаются пустые операторы: два рядом стоящих двоеточия. Строки, начинающиеся символом #, содержат комментарий. Обратная косая черта в конце строки означает, что в следующей строке идет продолжение текущей строки.

Приведенный выше пример записи определяет принтер, называемый lp, dj, или deskjet, его спул размещается в каталоге /var/spool/lpd/dj, максимальный размер задания не имеет ограничения, печать производится на устройство /dev/lp0 и страница с заголовком (с именем пользователя, который отправил задание на печать, и другой информацией) не добавляется в начало задания печати.

Подробнее о том, как создавать такие записи, можно прочитать на справочной странице для printcap. Но о некоторых самых важных (и обязательных) параметрах стоит рассказать здесь, хотя бы просто для примера.

- `sd=буферный_каталог`. У каждого принтера должен быть свой буферный каталог. Все буферные каталоги должны находиться в одном каталоге (обычно это /var/spool/lpd) и иметь имена, совпадающие с полными именами обслуживаемых ими принтеров. Буферный каталог нужен даже в том случае, если описываемый принтер подключен к другой машине: задания находятся на локальной машине до тех пор, пока они не будут переданы на печать.
- lp=имя_устройства. В таком виде этот параметр должен задаваться только для локального принтера. Если принтер находится на другой машине, вместо имени устройства указывается имя уникального файла, который существует и расположен на локальном диске. Чтобы принтер мог возвращать через указанный файл информацию о своем состоянии, необходимо задать в элементе булеву переменную rw; чтобы устройство было открыто и для чтения, и для записи.
- rm и `rp`. Во многих случаях в качестве печатающего устройства используется сетевой принтер, подключенный к какому-то другому компьютеру в локальной (или даже глобальной) сети. В таком случае на вашей машине в файле /etc/printcap должны присутствовать две переменные rm и `rp`. В переменной rm определяется машина, на которую должны посылаться задания, а переменная `rp` задает имя принтера на этой машине.
- П `mx`. Переменная `mx` используется для задания максимального размера (в байтах) печатаемого файла. Этот параметр может использоваться системными администраторами для того, чтобы предотвратить отрицательные последствия распространенной ошибки начинающих пользователей, состоящей в посылке на печать двоичных файлов (которые обычно имеют большую длину). Поскольку такие файлы могут содержать произвольные символы, в том числе символы, которые служат в качестве управляющих команд для принтера, печать двоичных файлов может приводить, например, к неоправданному расходованию бумаги и другим неприятным последствиям. На локальном принтере персонального компьютера этот параметр можно и не задавать. Я привел его для того, чтобы отметить, что для этого параметра, как и для всех других числовых параметров, значение должно отделяться от имени параметра знаком `#`, например, `mx#0`. Если написать `mx=0`, то такая запись не вызовет сообщений об ошибке, но она не изменяет значения переменной тх.

Еще одна очень существенная группа параметров — это параметры of, if и nf. Но о них надо поговорить особо, что мы и сделаем чуть ниже *в разд. 9.6.4.* Однако вначале рассмотрим программу, которая позволяет осуществить настройку принтера и создать файл /etc/printcap/.

9.6.3. Настройка LPD с помощью программы printconf-gui

Настройка системы LPD состоит в том, что обеспечивается возможность создавать очереди файлов и отправлять задания из этих очередей на принтер. Такую настройку можно произвести вручную, и я надеюсь, что приведенное выше описание позволяет это сделать. Однако в большинстве дистрибутивов имеются специальные утилиты, облегчающие конфигурацию подсистемы печати. В дистрибутиве Red Hat версии 6 такая утилита называлась printtoool, а в версии 7.1 вместо printtool включена утилита printconf-gui, которую можно вызвать из меню оболочки KDE. Эта утилита поддерживает ведение конфигурационного файла /etc/printcap, создание областей спулинга и выбор фильтров. Если в файле /etc/printcap не имеется ни одной записи о принтерах, то главное окно программы printconf-gui выглядит так, как показано на рис. 9.2.

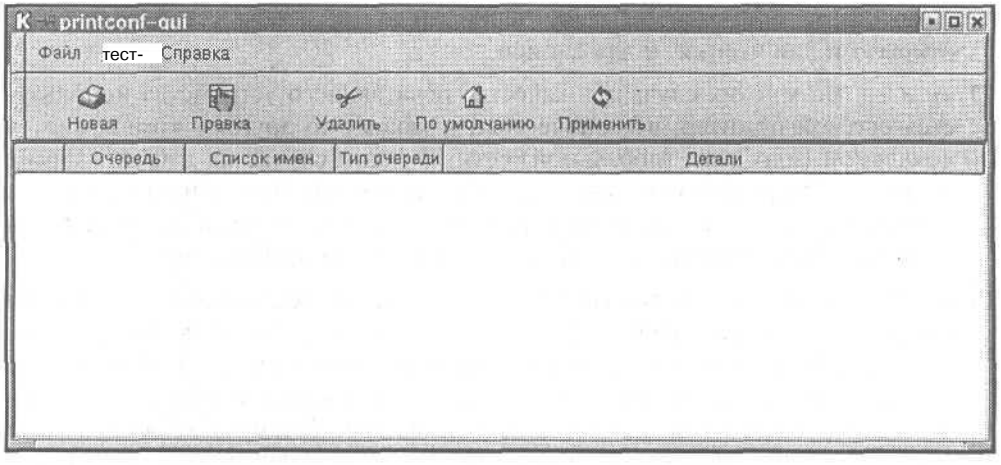


Рис. 9.2. Основное окно printconf-gui

Как видите, в меню программы имеется команда **Новая,** после выбора которой мы видим следующее окно (рис. 9.3), в котором можно ввести имя очереди (которое будет и именем принтера, а поэтому должно быть уникальным), а также его синонимы (aliases).

Щелкнув по команде меню **Тип очереди** в левом поле окна, вы получите возможность выбрать тип принтера и задать имя специального файла устройства. Тип очереди выбирается из выпадающего списка. Можно выбрать один из следующих типов:

- **Локальный принтер** — принтер, непосредственно присоединенный к вашему компьютеру через параллельный или USB-порт;

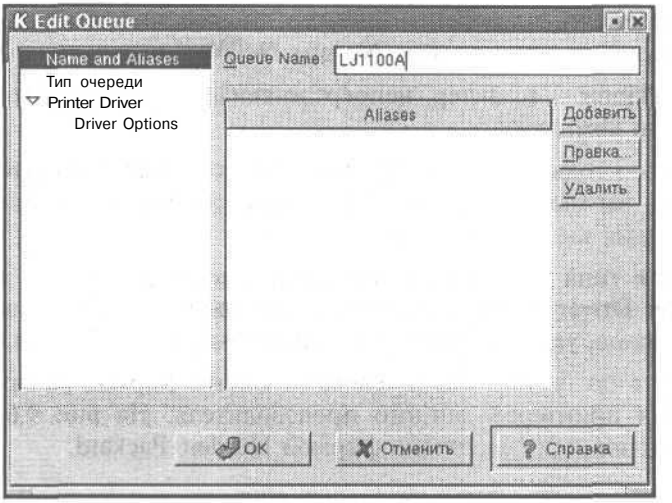


Рис. 9.3. Задание имени очереди

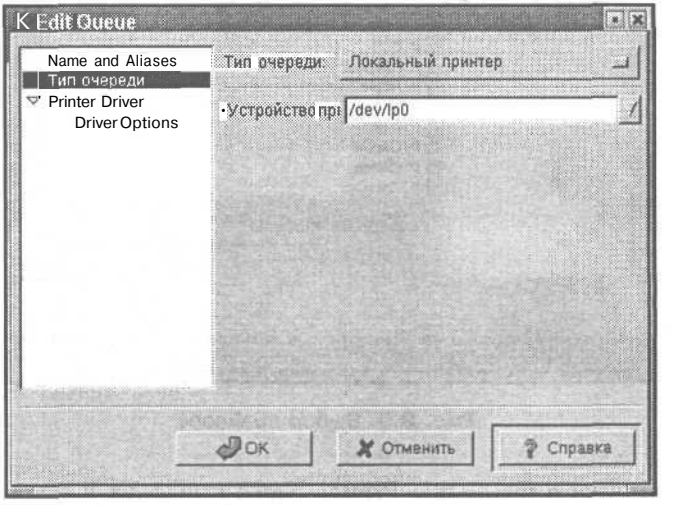


Рис. 9.4. Выбор типа принтера и задание устройства

- **Unix Printer** (lpd Spool) — принтер, присоединенный к другому UNIX-компьютеру, доступ к которому осуществляется по сети TCP/IP (например, принтер, присоединенный к другому Linux-компьютеру);
- **Windows Printer** (SMB Share) — принтер, присоединенный к другой системе, работающей под ОС Windows или имеющей установленный сервер Samba (SMB);

- **Novell Printer** (NCP Queue) — принтер, присоединенный к другой системе, использующей технологию Novell's NetWare и протокол NCP;
- **JetDirect Printer** — принтер, непосредственно подключенный к сети, а не к какому-то компьютеру.

Мы рассмотрим только случай подключения локального принтера, а остальные варианты вы можете изучить с помощью имеющейся в программе подсказки (экранная кнопка **Справка).**

После выбора типа принтера и указания имени устройства щелкните по строке **Printer Driver** в левой колонке, и вы увидите список известных программе драйверов, упорядоченный по фирмам-производителям (рис. 9.5).

Если щелкнуть по треугольничку слева от имени фирмы, раскроется список драйверов для принтеров данного производителя. На рис. 9.6 видна часть списка принтеров, производимых фирмой Hewlett-Packard.

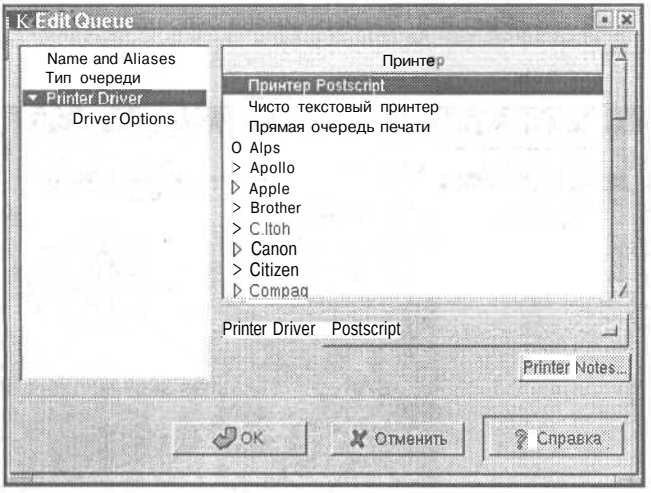


Рис. 9.5. Выбор драйвера

Щелчок по кнопке с надписью **Printer** Notes приводит к появлению окна с краткой информацией о выбранном принтере (рис. 9.7).

Ну, и последнее окно в этом ряду (рис. 9.8) позволяет задать некоторые параметры для выбранного принтера: режим печати (нормальный, экономичный, высокое качество), разрешение печати, размер бумаги, и т. п.

Завершив ввод, нажмите кнопку **OK.** После этого введенные вами данные будут сохранены в базе данных в файле /etc/printcap.

Команда **Правка** основного меню программы позволяет таким же способом отредактировать установки для принтера, который уже был описан ранее.

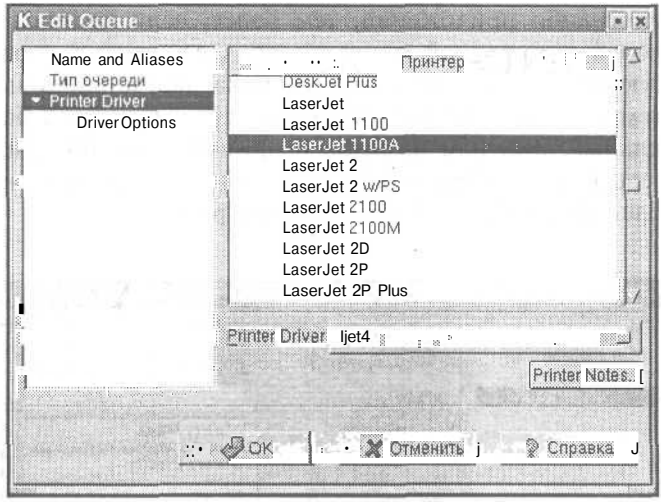


Рис. 9.6. Принтеры фирмы Hewlett-Packard

Рис. 9.7. Заметки о принтере

После того как вы добавили в базу данных новый принтер или отредактировали установки ранее заведенного, обязательно необходимо перезапустить демон lpd. Для этого можно воспользоваться кнопкой **Применить** в основном окне программы.

Теперь можно попытаться распечатать пробную страничку, воспользовавшись командой **Тест** главного меню программы. Вам будет предложено 3 варианта тестовой страницы: Print PostScript Test Page, Print A4 PostScript Test Page, или Print ASCII Test Page. Скорее всего, тестовая страница у вас распечатается без проблем. Но если вы попробуете распечатать страницу

текста из какого-либо приложения, тем более, если используются разные шрифты, да еще с кириллицей, то результат, к сожалению, может оказаться не таким радующим. Дело в том, что мы пока настроили только "нижний" слой подсистемы печати, обеспечивающий передачу потока байтов от ядра ОС в параллельный порт или на сетевой принтер. Но проблема управления самим принтером (а не параллельным портом) пока не решена. В системе LPD эта проблема решается с помощью фильтров.

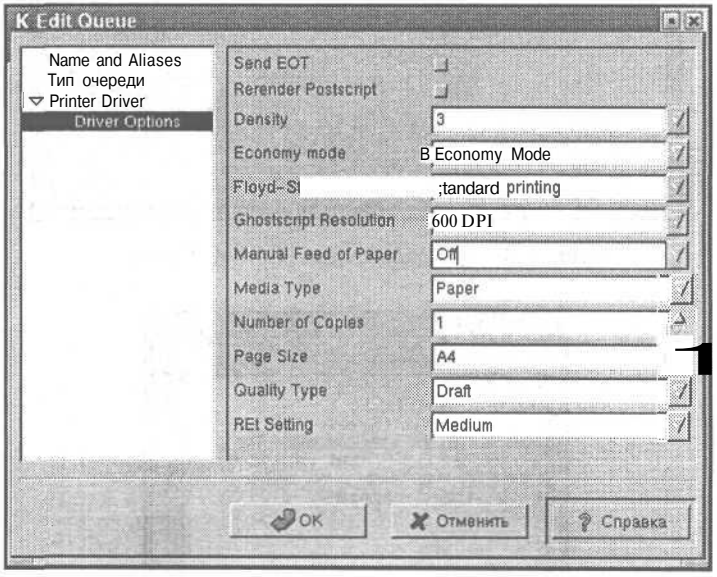


Рис. 9.8. Задание параметров печати

9.6.4. Фильтры

Когда задание на печать дождется своей очереди, lpd создает ряд программных каналов между буферным файлом и печатающим устройством для передачи данных, подлежащих печати. Посередине этой цепочки lpd устанавливает процесс-фильтр, в задачи которого входит просмотр и дополнительная обработка потока данных, направляемых на принтер. Процессы-фильтры могут выполнять над данными различные преобразования, в частности, форматирование и поддержку различных протоколов, которые могут понадобиться для работы с данным принтером.

Фильтр — это, как правило, просто сценарий shell, который вызывает ряд конвертирующих программ. Фильтр можно указать в командной строке вызова программы lpr. Если в командной строке фильтр не указан, то используются фильтры, заданные параметрами if, of и nf соответствующей записи

в файле /etc/printcap. Если в этой записи присутствует переменная if, а параметра of нет, то устройство (принтер) будет открываться один раз для каждого задания, а фильтр будет посылать одно задание на принтер и завершать работу. Если есть of, а if нет, то lpd однократно откроет устройство и вызовет программу-фильтр для посылки сразу всех заданий, стоящих в очереди. Это полезно для печати на тех устройствах, соединение с которыми требует большого времени. Одновременного использования параметров of и if следует избегать, а из двух предыдущих вариантов рекомендуется выбирать использование параметра if. Соответствующий элемент в записи файла /etc/printcap может иметь примерно такой вид:

```
:if=/var/spool/lpd/dj/filter:\
```

Если никакого фильтра вообще не задано, то вывод на печать может выглядеть очень некрасиво. Например, при печати обычного текстового файла вывод может выглядеть примерно так:

```
This is line one.
                This is line two.
                                This is line three.
```

Печать файла в формате PostScript выдаст листинг команд PostScript, напечатанных с этим "лестничным эффектом", а не полезный вывод. В руководстве "Printing HOWTO" приводится следующий пример простого фильтра, предназначенного только для того, чтобы устранить "лестничный эффект":

```
#!perl
# Предыдущая строка должна содержать полный путь к perl
# Скрипт должен быть исполнимым: chmod 755 filter
while(<STDIN>){chop $_; print "$_\r\n";};
# вы можете также добавить в конец прогон страницы: print "\f";
```

Этот текст надо сохранить в виде файла /var/spool/lpd/dj/filter, после чего будут нормально печататься обычные текстовые файлы.

Но печать простых ASCII-файлов — это только частный случай печати. В большинстве случаев в настоящее время печатаются файлы в других форматах, например, PostScript. Проблема вывода таких файлов на печать тоже решается путем использования фильтра, только гораздо более сложного. Таких фильтров разработано уже достаточно много, но самый важный из них — программа Ghostscript.

9.6.5. PostScript и Ghostscript

К сожалению пользователей, фирмы-производители принтеров долгое время не могли достигнуть согласия в вопросе о выборе управляющих сигналов для производимых ими устройств. В результате для каждого принтера до сих

пор необходим особый драйвер. Однако со времен так называемой "революции настольных издательских систем" 80-х годов в качестве стандартного языка управления принтером постепенно утвердился язык PostScript, разработанный фирмой Adobe Systems, Inc. И не только в UNIX-среде, а в издательском деле вообще.

Этот язык представляет собой специальный язык программирования для описания выводимой на печать страницы с текстом или графикой. Adobe Systems, Inc., изначально разработавшая стандарт на PostScript, открыла его для свободного распространения. Отметим еще, что формат PDF (Формат Переносимого Документа Adobe) — это в действительности чуть больше чем несколько преобразованный PostScript в сжатом файле.

Идея, заложенная в основу разработки PostScript, проста: все, что можно напечатать, описывается с помощью специального языка программирования, принтер же должен этот язык понимать. И принтеры, "понимающие" язык PostScript, т. е. имеющие встроенный PostScript-интерпретатор (так называемые PostScript-принтеры), быстро появились. К сожалению, они оказались стабильно дороже обычных принтеров. Тогда были разработаны программные PostScript-интерпретаторы, которые берут данные в формате PostScript и преобразуют в специфический для данного принтера управляющий код. Это дает вам виртуальный PostScript-принтер и позволяет использовать принтеры, не имеющие аппаратного интерпретатора.

Вероятно, одним из лучших программных интерпретаторов языка PostScript является Ghostscript (**http://www.cs.wisc.edu/~ghost/**), или просто gs. Он существует в двух вариантах. Коммерческая версия Ghostscript, называемая Aladdin Ghostscript или AFPL Ghostscript, свободна для персонального использования, но не может распространяться с коммерческими дистрибутивами Linux. В составе последних доступен GNU Ghostscript, представляющий собой тот же gs, только версией ниже и с другим лицензионным соглашением. На сегодняшний день можно загрузить версию AFPL Ghostscript 7.0, тогда как версия GNU Ghostscript — 5.5. В составе Ghostscript имеется внушительный набор фильтров — аппаратно-ориентированных модулей, позволяющих получать изображение на различных устройствах. Устройствах, а не принтерах, поскольку Ghostscript может обеспечить вывод на любое графическое устройство. Именно gs присутствует в качестве фильтра в /etc/printcap — конфигурационном файле lpd. Опции запуска gs в качестве фильтра определяются типом принтера.

9.6.6. Шрифты для Ghostscript

Для пакета Ghostscript разработаны PostScript-шрифты, которые обеспечивают высокое качество печати на Не-PostScript-принтерах. Такие шрифты наверняка найдутся на вашем дистрибутивном диске в виде пакета

ghostscript-fonts. Однако именно со шрифтами и связано большинство проблем, которые возникают при настройке принтера.

Дело в том, что программе Ghostscript надо точно знать, где расположены шрифты для нее. Но поскольку стандарт FHS (Filesystem Hierarhy Standard), о поддержке которого заявили все составители дистрибутивов, пока еще не утвердился окончательно, структура каталогов в Linux меняется от версии к версии даже в пределах одного дистрибутива. Поэтому файлы шрифтов могут оказаться где угодно. Очень часто — не там, где их будет искать Ghostscript. В результате при попытке распечатать какой-либо документ вы можете получить далеко не то, что ожидали: от несоответствия внешнего вида распечатанного документа вашему замыслу до искажения или отсутствия фрагментов текста, требующего, в соответствии с PostScript-файлом, того самого шрифта, который не смог загрузить Ghostscript. Положение усугубляется тем, что, по крайней мере, часть документов включает кириллицу, а некоторые дистрибутивы не имеют в каталогах, сканируемых по умолчанию Ghostscript, шрифтов с кириллицей.

Преодолеть эти трудности в принципе не сложно. Но, прежде чем рассказать, как это сделать, надо сказать, что в Linux имеется программа ghostview (gv), назначение которой — принять вывод ghostscript и вывести изображение на экран. Это дает инструмент, обеспечивающий возможность предварительного просмотра ("print preview") для любого приложения, генерирующего PostScript-файлы. С помощью gv вы сможете определить, связаны ли ваши проблемы с выбором типа принтера или с работой gs в целом. Видите на экране, но не получаете на печати - попробуйте другой фильтр (выберите другой принтер), не видите ничего "путного" — продолжаем разбираться с настройкой ghostscript.

Теперь надо отметить, что программу Ghostscript можно запускать не только в качестве фильтра для LPD, но и из командной строки (для этого надо дать команду gs). Этой возможностью и воспользуемся для целей отладки.

Вначале запустите команду gs с опцией -help. В результате вы получите, во-первых, краткий информативный список опций и доступных драйверов (заметим, что этот список является списком только вкомпилированных, а не всех доступных драйверов), и, во-вторых, перечень путей поиска. Этот список можно, конечно, изменить, но для этого надо перекомпилировать программу. Если же вы не хотите заниматься компиляцией, надо поместить файлы шрифтов именно в эти каталоги.

Но этого еще недостаточно для того, чтобы Ghostscript могла использовать шрифты. Дело в том, что эта программа обращается к шрифтам по именам, записываемым в той нотации, в которой допускается их использование в PostScript-файлах. Соответствие между такими названиями шрифтов и именами реальных файлов шрифтов задается файлом Fontmap (или

Fontmap.GS), который располагается в каталоге /usr/share/ghostscript/N.NN, где N.NN — номер версии программы ghostscript (на данный момент — 5.50). Каждая строка (кроме строк комментариев) этого файла состоит из трех элементов.

- Первым стоит имя, под которым шрифт будет известен программе Ghostscript, причем перед этим именем должен стоять слэш (/), либо имя должно быть заключено в круглые скобки.
- Далее следует имя файла шрифта либо синоним (aliace) имени шрифта. Если указывается имя файла шрифта, то оно должно быть заключено в круглые скобки и записано с указанием расширения (обычно это gsf, но допускаются также pfa и pfb), а также должно соответствовать правилам формирования имен файлов в MS-DOS, т. е. состоять из букв (в нижнем регистре), цифр и знаков подчеркивания. Если же это синоним, то указывается имя одного из уже известных программе Ghostscript шрифтов, причем перед этим именем должен стоять слэш (/).
- Завершает строку точка с запятой, перед которой должен стоять, по крайней мере, один пробел или знак табуляции.

Пути к файлам шрифтов в файле Fontmap не указаны. Если вы не использовали предлагаемые руководством средства принудительной "ориентации" ghostscript (параметры командной строки и переменные окружения), то gs будет использовать "пути по умолчанию", заданные при компиляции. В этих каталогах должны иметься файлы fonts.dir, которые содержат описание шрифтов в данном каталоге (подробнее о файлах fonts.dir вы можете прочитать *в разд. 11.4).*

Таким образом, в зависимости от потребностей вы можете либо внести в Fontmap необходимый шрифт (предварительно поместив соответствующий файл в один из доступных программе каталогов и указав имя файла в добавляемой строке), либо назначить в качестве синонима нужного шрифта имя одного из уже известных программе шрифтов. Например, сделать шрифт /Courier синонимом изначально известного программе шрифта /NimbusMonL-Regu (которому, в свою очередь соответствует файл (n022024l.pfb)). Если задача — печатать в основном файлы, PostScript-содержимое которых вне вашего контроля, — подберите синонимы для нужных шрифтов из числа известных программе. Если PostScript-файл генерируется под вашим контролем — просто выбирайте один из имеющихся в системе шрифтов. Разумеется, не забыв при этом описать его в Fontmap.

После этого выполните команду

```
[user]$ gv filename.ps
```

Если вы при этом увидите на экране весь текст из файла filename.ps, вы можете попытаться отпечатать файл и на принтере. Если же вместо текста увидите

пустой лист или шрифт вам не нравится, продолжайте экспериментировать с настройкой шрифтов. Но предварительно прочитайте статью *(см. [П11.12] приложения),* которая послужила основой для моего рассказа о шрифтах для Ghostscript, и в которой вы найдете несколько дополнительных подсказок. Кроме того, в Интернете имеются два очень полезных ресурса *(см. [П11.13] приложения)* и *(см. [П11.14]приложения),* куда будет не вредно заглянуть.

9.6.7. Печать на удаленный принтер

Если ваш компьютер подключен к локальной сети, то необязательно иметь принтер, непосредственно к нему подключенный, можно пользоваться принтером, подключенным к какому-то другому компьютеру. Настройка такого принтера на вашем компьютере требует только указания того, к какому компьютеру в сети подключен принтер (это делается с помощью задания переменных `rm` и `rp` в файле /etc/printcap, о чем было сказано выше). Если использовать утилиту printconf-gui, то достаточно при выборе типа очереди (см. рис. 9.4) выбрать вариант UNIX **printer (lpd Queue),** если это другой Linux-компьютер. Если принтер подключен к Windows-компьютеру или отдан в сеть через Samba-сервер, то, естественно, надо выбирать тип очереди **Принтер Windows (ресурс Samba).**

На удаленном компьютере должен быть разрешен доступ к этому принтеру. В Linux это делается с помощью файла /etc/lpd.perms (см. соответствующую страницу руководства man).

9.7. Звуковая карта

Если у вас одна из последних версий Red Hat Linux, запустите программу sndconfig. Делать это надо от имени суперпользователя и только в консольном режиме. Первым делом эта программа попытается сама определить наличие у вас звуковой карты (рис. 9.9).

Для этого она вызывает утилиту isapnptools. Если автоматически определить карту не удается, вам придется самому выбрать тип карты из списка (рис. 9.10).

Далее программа спросит вас о таких установках звуковой карты, как адрес порта ввода/вывода (I/O base address), IRQ, DMA, и 16-bit DMA (рис. 9.11). Будьте готовы ответить на эти вопросы. Затем программа попробует воспроизвести 2 музыкальных фрагмента, после каждого из которых спросит вас, слышали ли вы что-нибудь. Если вы ответите положительно, будет создан файл /etc/modules.conf, и вам остается только установить программу-проигрыватель, вроде xmms (о программах воспроизведения чуть позже, в *гл. 15).*

У меня установка звуковой карты и программы x11amp (с дистрибутивного диска) прошли почти без запинки (пришлось только подобрать порт, предлагаемое по умолчанию значение почему-то оказалось неподходящим), после чего я смог прослушивать записи из MP3-файлов так же, как делал это раньше под Windows с помощью winamp. Кстати, программа x11amp теперь называется xmms.

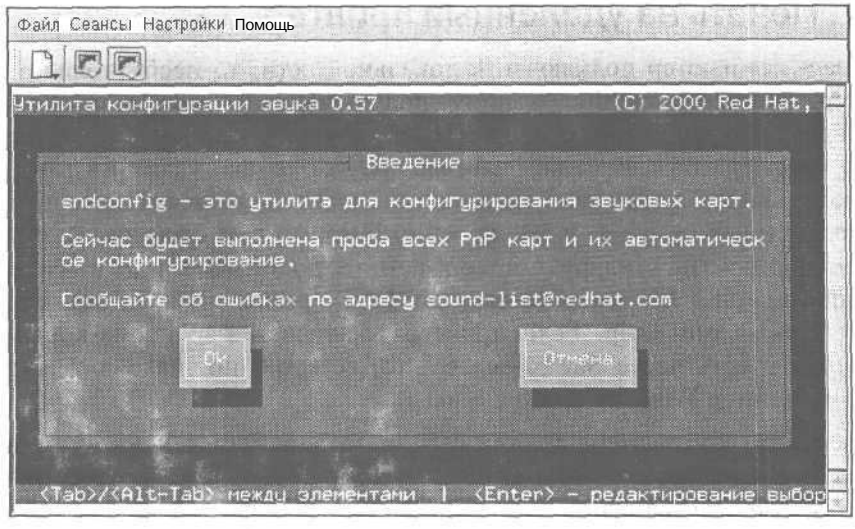


Рис. 9.9. Запуск программы sndconfig

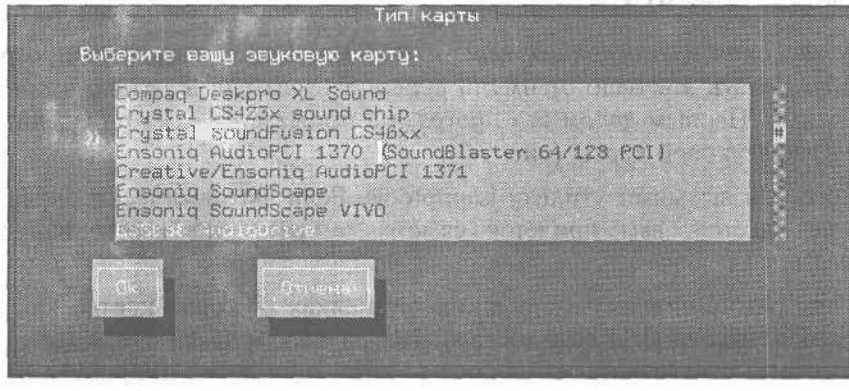


Рис. 9.10. Выбор типа звуковой карты

Если у вас возникнут какие-то затруднения, обратитесь к "Звук в Linux HOWTO" *(см. [П11.15]приложения)*.

Рис. 9.11. Выбор параметров звуковой карты

9.8. Дисковод CD-ROM

Довольно легко заставить Linux проигрывать для вас CD-диски: все, что вам нужно — это дисковод CD-ROM, колонки или наушники, CD-диск с музыкой и немного трудолюбия. Если вы устанавливали Linux на компьютер, в котором уже стоял CD-дисковод, то все необходимые настройки, скорее всего, сделаны автоматически. Если же вы поставили CD-ROM после установки Linux, то надо научить Linux распознавать этот дисковод. Для этого в каталоге /dev должна иметься ссылка (линк) на устройство. Если ее нет, создайте, выполнив следующие команды:

```
[root]# cd /dev
[root]# ln -s hdc cdrom
```

где вместо hdc вы, естественно, должны указать ваш CD-дисковод. Если не знаете, что тут указать, то внимательно просмотрите те сообщения, которые Linux выдает при загрузке. Для этого не требуется перезагружаться, протокол загрузки сохранен в файле /var/log/dmesg и выдается на экран по команде `dmesg`.

После этого можно запустить программу управления проигрыванием CD-дисков, например xplaycd. В графической оболочке KDE имеется простой проигрыватель CD-дисков с названием **CD Player** (рис. 9.12). Вызвать его можно через меню оболочки KDE.

Рис. 9.12. CD Player

9.9. ZIP-дисковод фирмы Iomega для параллельного порта

Для того чтобы использовать ZIP-дисковод, подключаемый к параллельному порту, вы можете использовать драйвер ppa, скомпилированный либо в составе ядра, либо в виде отдельного модуля. В последнем случае необходимо либо добавить строку `insmod ppa` в файл /etc/rc.d/rc.sysinit, либо просто каждый раз подключать нужный модуль той же командой insmod ppa. После этого вы будете иметь возможность смонтировать диск, установленный в ZIP-дисководе фирмы Iomega, с помощью обычной команды

```
mount -t vfat /dev/sda4 /mnt/zip
```

Может оказаться, что команда insmod ppa не срабатывает (например, если не скомпилирован модуль поддержки ppa). В этом случае после выполнения команды insmod ppa появляется сообщение об ошибке, и получить доступ к ZIP-диску этим способом не удается. У меня такая ситуация возникла после перехода с версии 5.2 дистрибутива Black Cat на версию 6.02.

Тем не менее, мне удалось получить доступ к ZIP-дискам. Я нашел пакет Iomega версии 1.0.1 (обратите внимание на название программы — Iomega, а не iomega), созданный Джоном Хоком (John Hawk, e-mail: **visionary@gtemail.net**). Этот пакет дает возможность работать под Linux с дисками ZIP и/или JAZ фирмы Iomega. После обычной процедуры установки tar-gz-пакета (которая выполнялась с правами суперпользователя), мне легко удалось (все еще с правами суперпользователя и при условии, что в дисководе имеется носитель) смонтировать ZIP-диск.

Для того чтобы монтирование дисков могли выполнять обычные пользователи, необходимо (от имени root) выполнить следующие команды:

```
[root]# chown root:root /directory/lomega
[root]# chmod +s /directory/Iomega
```

Помимо обычного способа доступа к диску командами `mount/umount`, можно запустить отдельную программу /usr/local/bin/lomega. Запускается она только в графическом режиме и представляет собой интерфейс для работы с

ZIP-диском. С помощью этой программы можно смонтировать и размонтировать ZIP-диск, защитить диск от записи или снять эту защиту, а также извлечь носитель из дисковода простым щелчком мыши по соответствующей кнопке. Отметим только, что для того, чтобы установить защиту от записи, необходимо размонтировать диск.

В окне программы можно увидеть статус диска (смонтирован, размонтирован, защищен ли от записи), просмотреть содержимое файловой системы и отдельно — файлы на ZIP-диске, а также сделать back-up выбранных файлов. Для этого надо просто отметить нужные файлы в окне программы и нажать экранную кнопку **Back-Up.** После этого будет выведено диалоговое окно, в котором вы можете изменить предлагаемые по умолчанию параметры команды архивации tar. Таким образом, программа Iomega предоставляет удобный интерфейс для архиватора tar.

Щелкнув правой кнопкой мыши в окне программы, вы получите выпадающее меню, с помощью которого можно выполнить много разных команд, в частности, удалить некоторые файлы и каталоги. При этом удаляемые файлы фактически не уничтожаются, а просто перемещаются в "корзину" (trash), откуда их, при необходимости, можно еще восстановить. Однако эта возможность сохраняется только до тех пор, пока вы не опустошили корзину командой **Empty Trash,** которую можно вызвать через то же выпадающее меню.

Программа Iomega использует конфигурационный файл /etc/lomega.conf. Структура этого файла подобна структуре файла /etc/fstab и не требует подробных пояснений.

При запуске программы можно задать в командной строке следующие опции:

- -e — открывает окно файлового менеджера;
- `-v` — просто выводит версию программы;
- `/dev/sda` — указывает имя используемого устройства (должно иметься в файле lomega.conf).

Помимо двух уже описанных, имеется еще один способ получения доступа к ZIP-диску из Linux: с помощью пакета mtools. Этот пакет, конечно, необходимо предварительно установить, а затем добавить следующую строку в файл /etc/mtools.conf:

```
drive z: file="/dev/sda4" exclusive
```

И все же основным способом подключения ZIP-дисковода остается первый из трех описанных — с помощью модуля ppa. Упомянутые выше затруднения возникли у меня только с одной версией дистрибутива. После перехода на Red Hat 7.1 и ALT Linux Junior 1.0 (на двух разных компьютерах) я без затруднений подключаю ZIP-диски стандартным способом.

Глава 10

Установка и обновление программных пакетов

10.1. Два способа установки ПО

Необходимость в установке новых программных пакетов под Linux возникает в двух основных случаях:

- когда появляется новая версия одного из уже установленных у вас пакетов;
- когда возникает желание или необходимость использовать какой-то пакет, еще не установленный в системе.

Во втором случае это может быть один из пакетов, имеющихся на вашем установочном диске, но не установленный в процессе инсталляции. Однако чаще всего новое ПО вы будете находить в Интернете, тем более, что значительная часть этого ПО бесплатна. Как бы то ни было, но рано или поздно вы все равно окажетесь перед необходимостью установить новый пакет.

Для дистрибутивов, основанных на Red Hat Linux, существует две основных формы распространения ПО: в исходных текстах и в виде исполняемых модулей. В первом случае пакет ПО обычно поставляется в виде tar-gz-архива, во втором случае — в виде RPM-пакета (но это необязательно, исполняемые модули также могут распространяться в виде tar-gz-архива).

Проще всего установить ПО, представленное в виде rpm-пакета, содержащего исполняемые файлы, этот способ и рассмотрим первым. Отметим только, что для инсталляции новых пакетов вы должны войти в систему как пользователь root.

10.2. Программа rpm

Название этой программы (или команды) является аббревиатурой от Redhat Package Manager. Такая расшифровка дается в большинстве книг и руководств по Linux и кажется мне более правильной и логичной, хотя в *гл. 6* "The Official Red Hat Linux Reference Guide" говорится: "The RPM Package Manager (RPM), is an open packaging system available for any-one to use, and works on Red Hat Linux as well as other Linux and UNIX systems", т. е. предлагается рекурсивная расшифровка названия RPM, подобная расшифровке GNU - GNU is Not Unix).

Программа rpm в некотором смысле аналогична программам типа setup wizard для MS Windows. Преимуществом использования этой программы по сравнению с установкой tar-gz-архивов является то, что она автоматически проделает все необходимые действия по установке ПО: создаст необходимые каталоги, распределит по ним файлы, создаст ссылки. Кроме того, она может быть использована не только для установки нового пакета, но и для обновления версий ПО, получения перечней установленного ПО и проверки установки, а также для деинсталляции отдельных пакетов (например, если после периода пробной работы с программой вы решили отказаться от ее дальнейшего использования). С помощью той же программы rpm можно самому создать пакет формата RPM, однако для начинающих лучше, наверное, этим не заниматься, а воспользоваться готовыми rpm-пакетами.

Rpm-пакеты — это специальным образом подготовленные архивы, предназначенные для обработки программой rpm. Название RPM-пакетов оканчивается на суффикс .rpm, например, xzip-180-1.i386.rpm или xzip-180-1.src.rpm. Как видите, перед суффиксом .rpm стоит еще один суффикс. Если это .i386, i686 или .i586, то в пакете находятся исполняемые файлы (оптимизированные для соответствующего типа процессора), а если этот суффикс .src, — то в пакете исходные тексты, которые после установки еще надо скомпилировать. Обычно как на установочных компакт-дисках, так и в интернет-каталогах rpm-пакеты с исполняемыми файлами располагаются в каталогах с названием RPMS, а RPM-пакеты с исходными текстами — в подкаталогах SRPMS. Часто встречаются также RPM-пакеты с суффиксом .noarch.rpm, содержащие файлы, которые просто без всякой дополнительной обработки устанавливаются в соответствующие каталоги (например, файлы страниц интерактивного руководства man). И, наконец, если rpm-пакет рассчитан на версию Linux, предназначенную для другой аппаратной платформы (AMD, DEC Alpha, SUN Spare, MIPS, PowerPC), это тоже будет отображено в имени пакета: вместо i386 в суффиксе будет стоять, соответственно, athlon, alpha, spare, mips или ppc.

В Интернете RPM-пакеты можно найти на различных серверах. По моему опыту наиболее удобным сервером в Интернете для поиска RPM-архивов является сервер **http://rufus.w3.org** (недаром он имеет другое имя **http://rpmfind.net).** На нем установлена поисковая система, которая позволяет упорядочивать список пакетов наиболее удобным для вас способом:

- по именам пакетов;
- по дистрибутивам;
- по группам приложений;
- по датам;
- по поставщикам (производителям) ПО.

Общий объем архива RPM-пакетов на этом сервере составляет более 66 Гбайт. Очень богатые архивы хранят также два FTP-сервера в России: **ftp://ftp.chg.ru/pub/Linux и ftp://ftp.nc.orc.ru/.**

Необходимо только заметить, что если для перекачки пакетов из Интернета вы используете компьютер, работающий под Windows 95, то все имена пакетов у вас будут, скорее всего, искажены. Дело в том, что Windows "не любит" имена, в которых несколько точек (например, glib-1.0.6-3.i386.rpm) и заменит "лишние", по его мнению, точки на знаки подчеркивания — glib-1_0_6-3_i386.rpm. Так что после получения пакета (при переносе его на ПК с ОС Linux) желательно эти "исправления" устранить, вернувшись к исходным именам UNIX. Правда, делать это необязательно, поскольку внутри rpm-пакет все равно правильно идентифицирован, но для единообразия и облегчения поиска файлов все же целесообразно.

Итак, вы нашли и скачали RPM-архив с исполняемой версией нужного вам пакета. Если вы ставите совершенно новый пакет (у вас не было на компьютере предыдущих версий этого ПО), то для установки пакета из этого архива достаточно перейти в тот каталог, где находится архив, и дать команду *(для самых нетерпеливых: не спешите выполнять эту рекомендацию, прочитайте еще хотя бы пару абзацев)*

```
[root]# rpm -i имя_rpm-архива
```

Если у вас была установлена предыдущая версия пакета, то в простейшем случае надо дать команду следующего формата:

```
[root]# rpm -U --force имя_rpm-архива
```

Здесь параметр `-U` говорит программе, что надо произвести обновление (upgrade) пакета, а опция `--force` требует безусловно (и без лишних вопросов) обновить все входящие в пакет файлы. Заметьте, что это очень сильное требование, и в некоторых случаях может быть лучше сохранить какие-то (например, конфигурационные) файлы от предыдущей версии. Если установка проходит нормально и никаких дополнительных сообщений не появляется, то после завершения работы программы (после появления приглашения оболочки) вы можете пользоваться вновь установленным пакетом.

К сожалению, не всегда все так просто. Приведу конкретный пример. У меня был установлен Red Hat Linux версии 5.2, причем программа Midnight Commander (mc) была версии 4.1.36. На FTP-сервере я увидел версию 4.5.30 этой программы (пакет mc-4.5.30-12.i386.rpm) и, естественно, решил ее поставить. Однако оказалось, что для этого необходимо установить еще 4 других пакета, о чем rpm мне и сообщила:

```
ошибка: неудовлетворенные зависимости:
redhat-logos нужен для mc-4.5.30-12
libglib-1.2.so.0 нужен для mc-4.5.30-12
```

```
libc.so.6(GLIBC_2.1) нужен для mc-4.5.30-12
libc.so.6(GLIBC_2.0) нужен для mc-4.5.30-12
```

Это не удивительно, если вы вспомните, что и при первоначальной установке Linux программа инсталляции тоже проверяла взаимозависимости пакетов и предлагала установить недостающие. Однако в случае инсталляции с CD-ROM все необходимые пакеты находятся на том же диске, а здесь мне пришлось вначале поискать нужные пакеты. Два пакета (redhat-logos-1.0.5-1.noarch.rpm и glibc-2.1.1-6.i386.rpm) я нашел без труда, после чего rpm перестала просить и GLIBC_2.0. А вот с libglib.so.l вышло сложнее. Во-первых, я никак не мог найти пакета с таким названием. Как оказалось, такого пакета и не существует, файл libglib.so.l входит в состав пакета glib-1.0.6-3.i386.rpm.

Программа rpm позволяет выяснить, какие файлы установит тот или иной пакет. Для этого надо дать следующую команду (только учтите, что текущим должен быть каталог, содержащий интересующий вас пакет):

```
[root]# rpm -qpl имя_rpm-архива
```

А для получения информации о том, для чего служит ПО, содержащееся в rpm-пакете, используйте команду

```
[root]# rpm -qpi имя_rpm-архива
```

Дело в том, что файлы RPM кроме собственно архива файлов содержат информацию о пакете, включая имя, версию и краткое описание. С помощью той же программы rpm вы можете просмотреть эту дополнительную информацию. Например, для пакета glib-1.0.6-3.i386.rpm вывод команды

```
[root]# rpm -qpi glib-1.0.6-3.i386.rpm
```

будет примерно таким:

```
Name : glib Relocations: (not relocateable)
Version : 1.0.6 Vendor: Red Hat Software
Release : 3 Build Date: Суб 10 Окт 1998 04:49:03
Install date: (not installed)
Build Host : porky.redhat.com
Group : Libraries Source RPM: glib-1.0.6-3.i386.rpm
Size : 55305
Packager : Red Hat Software <bug@redhat.com>
Summary : Handy library of utility functions
Description : Handy library of utility functions. Development libs
and headers are in gtk+-devel.
```

Если дать команду:

```
[root]# rpm -qpl glib-1.0.6-3.i386.rpm
```

будет выдан список входящих в пакет файлов с указанием того, куда они будут установлены:

```
/usr/lib/libglib.so.1
/usr/lib/libglib.so.1.0.6
```

RPM также предоставляет мощную систему запросов по установленным в системе пакетам. По команде

```
[root]# rpm -qa
```

вы получите перечень всех установленных в системе пакетов (перечень будет очень большим, так что лучше сразу направить вывод в фильтр more или в файл, который потом просматривать с помощью командой less или встроенной программы просмотра из оболочки Midnight Commander). Вы можете искать информацию об отдельном пакете или об отдельных файлах. Например, вы можете легко найти, какому пакету принадлежит файл и откуда появился. Команда

```
[root]# rpm -qf /etc/bashrc
```

сообщит:

```
bash-1.14.7-16.
```

Если вы беспокоитесь о том, что случайно удалили важный файл из установленного пакета, просто проверьте это:

```
[root]#rpm -Va
```

Вы будете оповещены о любых аномалиях. Потом можно переустановить пакет, если это необходимо. Любые конфигурационные файлы будут сохранены.

Как видите, rpm это очень полезная утилита, и у нее имеется много разных опций. Выше приведено только несколько примеров. Всего rpm имеет 16 основных режимов работы, которые можно объединить в 6 групп (после двоеточия приводится формат команды для соответствующего режима).

- Запросы.
  - **Запрос:** rpm [--query] [queryoptions]
  - Показать метки запросов (Querytags): rpm [--querytags]
- Установка и поддержка установленных пакетов.
  - Установка: rpm [--install] [installoptions] [package_file] +
  - Обновление: rpm [--freshen|-F] [installoptions] [package_file] +
  - Деинсталляция: rpm [--uninstall|-e] [uninstalloptions] [package] +
  - Проверка: rpm [--verify|-V] [verifyoptions] [package] +

- Подписи (пакеты подписываются электронной цифровой подписью в формате PGP, с целью обеспечения неизменяемости и сохранения авторства пакетов).
  - **Проверка ПОДПИСИ:** `rpm [--verify|-V] [verifyoptions] [package]+`
  - **Переподписывание: rpm** `[--resign] [package_file]+`
  - **Добавление ПОДПИСИ: rpm** `[--addsign] [package_file]+`
- Работа с базой.
  - Инициализация базы: rpm `-i [--initdb]`
  - Обновление базы (Rebuild Database): rpm -i `[--rebuilddb]`
- Создание rpm-пакетов.
  - **Создать пакет: rpm** `[-b|t] [package_spec]+`
  - **Перекомпилировать пакет: rpm** `[--rebuild] [sourcerpm]+`
  - Скомпилировать пакет из tar-архива: rpm `[--tarbuild] [tarredsource]+`
- Разное.
  - Показать конфигурацию программы rpm: rpm `[--showrc]`
  - **Задать пользователей: rpm** `[--setperms] [package]+`
  - **Задать группы: rpm** `[--setgids] [package]+`

Подробное описание всех возможностей команды rpm выходит за рамки нашей книги Его вы можете найти в RPM-HOWTO, на страницах man и info. Кроме того, большой раздел о программе rpm имеется в книге *(см. [П1.3] приложения)*.

Примечание

Как и другие программы для Linux, программа rpm постоянно развивается и совершенствуется. При этом при замене версии этой программы могут возникнуть трудности с установкой пакетов, созданных в предыдущих версиях. Так было, например, при переходе с третьей на четвертую версию rpm. Так что надо использовать пакеты, соответствующие установленной у вас версии программы.

Приведенное выше описание программы rpm предполагает, что она запускается с консоли или в эмуляторе терминала. Между тем в разных дистрибутивах имеются графические оболочки для управления rpm-пакетами. В составе графической среды KDE такая оболочка называется kpackage. Вы можете запустить ее либо из командной строки, либо из основного меню KDE. Однако, на мой взгляд, она не дает никаких преимуществ по сравнению с работой из командной строки. Кроме того, она описана в книге А. Федорчука "Офис, графика, Web в Linux" *(см. [П1.6] приложения)*, так что я не буду тратить время на ее рассмотрение.

10.3. Компиляция ПО из исходных текстов

Если rpm-пакеты с необходимым вам программным обеспечением нужно еще поискать (и не всегда можно найти), то tar-gz-архив любого ПО для Linux найдется в Интернете непременно. В некоторых случаях такие архивы содержат исполняемые модули приложений. Тогда установка приложения лишь немного сложнее, чем в случае установки из RPM-пакета: необходимо просто развернуть архив с помощью программ gunzip и tar, перейти в созданный каталог и можно уже запускать полученное приложение. Но чаще всего приложения поставляются в исходных текстах, т. е. в виде программы на языке С. Установить их в этом случае немного сложнее, хотя и тут нет ничего невозможного даже для начинающего пользователя. Давайте рассмотрим как это делается.

10.3.1. Необходимые сведения о программировании на языке С

Начать стоит с того, что операционная система UNIX родилась на свет одновременно с языком программирования С (Си). Более того, язык С был создан специально для разработки этой ОС, значительная часть UNIX была написана на языке С. ОС Linux тоже написана на С. Поэтому, а также в соответствии с принципом свободного распространения исходных кодов, многие приложения для Linux распространяются в виде текстов на С (а в последнее время — и на C++). Естественно, что для установки и запуска такого приложения на исполнение его необходимо предварительно скомпилировать. Для выполнения процедур компиляции обычно используется программа gcc (хотя существуют и некоторые альтернативные разработки).

Примечание

Изначально аббревиатура GCC имела смысл GNU C Compiler, но в апреле 1999 года сообщество GNU решило взять на себя более сложную миссию и начать создание компиляторов для новых языков с новыми методами оптимизации, поддержкой новых платформ, улучшенных runtime-библиотек и других изменений **(http://gcc.gnu.org/gccmission.html).** Поэтому сегодня GCC расшифровывается как GNU Compiler Collection (коллекция компиляторов GNU) и содержит в себе компиляторы для языков С, C++, Objective C, Chill, Fortran, Ada и Java, а также библиотеки для этих языков (libstdc++, libgcj и др.).

GNU-компилятор с языка С gcc, содержит в себе 4 основных компонента, соответствующие четырем этапам преобразования исходного кода в исполняемую программу.

Первый компонент — это препроцессор, который модифицирует исходный код программы перед компиляцией в соответствии с командами препроцессо-

ра, содержащимися в С-программе. В соответствии с этими командами выполняются простые подстановки текста. Второй — собственно компилятор, который обрабатывает исходный код и преобразует его в код на языке ассемблера. Третий компонент — ассемблер, который генерирует объектный код. И, наконец, четвертый компонент — компоновщик, который собирает исполняемый файл из файлов объектного кода. Дело в том, что большие программы обычно пишутся по частям, в виде множества отдельных файлов, содержащих исходный код соответствующей части. Компилятор обрабатывает каждый такой файл отдельно и создает отдельные объектные модули (файлы таких модулей обычно имеют расширение о). Создание единой исполняемой программы из таких модулей и является задачей компоновщика. При таком подходе, если в какой-то модуль программист вносит исправление, нет необходимости заново компилировать всю программу: достаточно откомпилировать исправленный модуль и заново запустить компоновщик.

Для выполнения стандартных операций программист может использовать функции из стандартных библиотек. Самый характерный пример -- это библиотека libc, которая содержит функции, выполняющие такие задачи, как управление памятью и операции ввода/вывода. Программисты могут создать свои собственные библиотеки и использовать их при написании новых программ.

Библиотеки бывают статическими, разделяемыми и динамическими. Статическая библиотека — это библиотека, код которой встраивается в программу при компиляции. Код разделяемой библиотеки не встраивается в программу, а загружается в память одновременно с программой и программа получает доступ к функциям этой библиотеки. Динамические библиотеки – разновидность разделяемых, но библиотечные функции загружаются в память только тогда, когда из программы поступит вызов соответствующей функции. В процессе выполнения программы они могут выгружаться и заменяться другими функциями из той же или другой библиотеки. Имена статических библиотек обычно имеют суффикс .a, а имена разделяемых библиотек — суффикс .so, за которым следует старший и младший номера версии. Имя может быть любой строкой, которая однозначно характеризует библиотеку. Обычно имена библиотек начинаются с lib. Примеры: libm.so.5 — общая математическая библиотека, libX11.so.6 — библиотека для работы с системой X Window. Библиотека libc.so.5 компонуется автоматически, в то время как большинство других библиотек необходимо явно указывать в командной строке при вызове программы gcc. Это делается через опцию -1 за которой следует уникальная часть имени библиотеки, например, для вызова математической библиотеки достаточно указать `-lm`.

Многие системные библиотеки располагаются в системных каталогах, например, в /usr/lib и /lib, но некоторые могут располагаться и в других местах. Список этих каталогов помещается в файл /etc/ld.so.conf. Каждый раз, когда разделяемая библиотека изменяется или инсталлируется вновь, нужно

выполнять команду ldconfig, чтобы обновить файл /etc/ld.so.conf, а также ссылки на него. Если библиотека инсталлируется из RPM-пакета, это обычно делается автоматически, хотя и не всегда.

При компиляции больших программ, использующих фрагменты исходного кода, расположенные в разных файлах, бывает очень трудно отследить, какие файлы нужно перекомпилировать, а какие только компоновать. В таких случаях очень помогает утилита make, которая автоматически определяет, следует ли компилировать файл исходного кода, по дате его последней модификации. Утилита make оперирует файлами, исходя из их зависимости друг от друга. Эти зависимости определяются файлом с именем makefile. Строка файла makefile состоит из трех частей: имени целевого файла, списка файлов, от которых он зависит, и команды. Если какой-либо файл из списка изменился после целевого файла, то выполняется указанная в строке команда. В строке может быть указано несколько команд. Обычно команда — это вызов компилятора для компиляции файла исходного кода или компоновки файлов объектного кода. Строки, определяющие зависимости, отделяются друг от друга пустой строкой.

10.3.2. Инсталляция пакетов ПО из исходных текстов

Теперь, когда мы получили общее представление о компиляции программ на языке C, можно рассмотреть обращение с пакетами программ, распространяемыми в виде исходных кодов. Первое, что надо сказать в этой связи, это то, что для установки таких пакетов вы, естественно, должны иметь в своей системе утилиты gcc и make.

Непосредственно процесс инсталляции пакета состоит из следующих шагов:

1. Перейти (с помощью команды cd) в каталог, содержащий исходные коды устанавливаемого пакета.
2. Выполнить команду ./configure, которая осуществляет конфигурирование пакета в соответствии с вашей системой. Процесс выполнения этой команды занимает довольно длительное время, причем команда выдает на экран сообщения о том, какие именно особенности системы испытываются.
3. Выполнить команду make для того, чтобы скомпилировать пакет.
4. После этого можно выполнить (этот шаг не является обязательным) команду make check, которая вызывает запуск процедур самотестирования, которые поставляются с пакетом.
5. Выполнить команду make install для установки программ, а также файлов данных и документации.

6. Заключительный этап состоит в выполнении команды make clean, которая удаляет промежуточные объектные и двоичные файлы из каталога с исходными кодами. Для удаления временных файлов, которые создала команда configure (после чего пакет можно компилировать для другого типа компьютеров), надо выполнить команду make distclean.

В большинстве случаев выполнения этой последовательности команд достаточно для установки нового пакета.

Основная проблема, с которой приходится сталкиваться при инсталляции программ из исходных кодов, связана с конфликтами версий: для вновь устанавливаемого пакета требуются новые версии каких-то системных утилит, которые пока еще не установлены в вашей системе. Более того, часто возникает целая цепочка (или даже дерево): для программы нужна какая-то новая версия утилиты, для последней нужно обновить еще какие-то утилиты, и т. д. Но если вы не очень давно устанавливали (или обновляли) дистрибутив, то таких проблем не возникает, и обновление пакета пройдет без затруднений. Желаю вам успеха!

Глава 11

Русификация и шрифты

Может возникнуть вопрос: а надо ли подробно разбирать вопрос русификации, не лучше ли просто сразу установить русифицированный дистрибутив? Тем более, что в последних версиях дистрибутивов Red Hat Cyrillic Edition, ASPLinux и AltLinux русификация выполнена на вполне приемлемом уровне. Однако, даже в случае установки русифицированного дистрибутива вы имеете шанс столкнуться с проблемой русификации на последующих этапах. Может получиться так, что новая версия нерусифицированного дистрибутива появится раньше, чем соответствующий русифицированный вариант, и вы захотите его установить. Не всегда хочется дожидаться пока выйдет русская версия. Русификация может нарушиться при обновлении отдельных программных пакетов. У меня, например, что-то случилось с русификацией после установки XFree86 версии 4.0.1. Таким образом, задача русификации может встать перед любым пользователем ОС Linux.

Когда я начинал работать с ОС Linux, самым лучшим материалом по русификации был "The Linux Cyrillic HOWTO" Александра Беликова (Версия 4.2 b2, декабрь 11, 1998) в переводе Е. М. Балдина. Кроме этого HOWTO были доступны только материалы со странички Леонида Кантера. Однако оба этих источника уже в то время существенно устарели, т. к. в Red Hat версии 6 изменились даже команды выбора шрифта. Переводчик "The Linux Cyrillic HOWTO" Е. Балдин в настоящее время создает свой вариант HOWTO по кириллизации *(см. [П13.1]приложения)*. Думаю, что в ближайшее время (когда автор закончит работу над ним) он станет исчерпывающим источником сведений по этому вопросу. Очень полезен также RU.LINUX.FAQ *(см. [П13.16] приложения)*. Настоящая глава во многом следует этим двум основным источникам.

Начать надо с двух замечаний. Во-первых, поскольку способы вывода информации на экран в графическом и текстовом режимах принципиально различны, придется отдельно рассмотреть вопрос о русификации текстового и графического режимов. Во-вторых, в системе Linux существуют два конкурирующих пакета управления консольными шрифтами и клавиатурой:

- **kbd** (**ftp://ftp.win.tue.nl/pub/linux/utils/kbd/** или **ftp://ftp.kernel.org/pub/linux/utils/kbd/**)
- consoletools (**http://lcr.sourceforge.net**).

В разных дистрибутивах применяется или один, или другой. Например, в Red Hat 4.x и *5.x* для русификации консоли применялся пакет kbd. В Red

Hat *6.x* применяется уже другой пакет — consoletools. Приводимое ниже описание ориентировано, в основном, на пакет consoletools.

11.1. Предварительные сведения

В *разд. 9.3* мы уже рассмотрели вопрос о кодировке символов и о работе клавиатуры, а также научились задавать (изменять) раскладку клавиатуры, т. е. вопрос о вводе информации в компьютер. Теперь надо рассмотреть вторую сторону этого вопроса — вопрос о выводе информации для восприятия человеком.

11.1.1. Вывод символов на экран

Обычно (если не считать управляющих комбинаций) код нажатой клавиши либо записывается в файл, либо соответствующий символ отображается на экране. В файл, разумеется, записываются последовательности байтов, а не символы как таковые, но и они в конечном итоге предназначены для прочтения человеком, а человек воспринимает только изображения печатных знаков на экране или в распечатке.

Текстовый режим

Работа экранного драйвера текстового режима основана на использовании 16-битной кодировки символов UNICODE (UCS2). Изображение каждого символа, соответствующего любому двухбайтному коду кодировки UNICODE, представляется матрицей из нулей и единиц размером 8 столбцов на H строк (обычно H принимает значения 8, 14 или 16). Единица в этой матрице соответствует светящейся точке на экране, а ноль — затемненной точке. Каждая строка этой матрицы кодируется одним байтом. Совокупность таких матриц (точнее, их байтовых представлений) для всех символов UNICODE образует таблицу экранного шрифта (Screen Font Map, SFM). Файл, в котором хранится такая таблица, может содержать шрифт одного размера по высоте (H) или шрифты нескольких размеров.

Сам экранный драйвер может работать в одном из двух режимов: режиме UTF или байтовом режиме. Выбор режима определяется приложением, которое обращается к этому драйверу для вывода символов на экран.

В режиме UTF последовательности байтов, получаемые от приложения для отображения на экране консоли, преобразуются по алгоритму UTF в коды UNICODE. После такого преобразования драйвер экрана обращается к загруженной в память таблице экранного шрифта (SFM) за соответствующим данному коду изображением символа.

В байтовом режиме драйвер экрана использует дополнительную таблицу — таблицу перекодировки символов (Application Charset Map, ACM) для пре-

образования получаемых от приложения последовательностей байтов в коды UNICODE. Эта таблица зависит от кодировки символов, применяемой приложением. В дальнейшем драйвер экрана, как и в режиме UTF, обращается к таблице экранного шрифта (SFM) для того, чтобы извлечь из нее изображение нужного символа.

Примечание

Для того чтобы определить, работает ли виртуальная консоль в режиме UTF или в байтовом режиме, можно воспользоваться скриптом vt-is-UTF8, а для переключения режимов работы виртуальной консоли служат два скрипта: unicode_start и unicode_stop.

В ядре Linux отведено место для хранения четырех таблиц перекодировки ACM. Первые три таблицы определяют 437-кодовую страницу IBM (cp437), таблицу для набора символов терминала DEC VT100 (vt100) и таблицу для набора символов ISO latinl (iso01). Эти три таблицы встроены в ядро и никогда не меняются. В качестве четвертой таблицы перекодировки в ядре может быть записана таблица перекодировки, выбранная пользователем.

Консольный драйвер Linux позволяет для каждой виртуальной консоли определить (с помощью команды charset) две ссылки (в документации их называют "сокетами") на таблицы перекодировки ACM. Эти две ссылки обозначаются как GO и G1, причем для каждого виртуального терминала значения, присвоенные этим ссылкам, выбираются независимо. Однако, хотя ссылки GO и G1 задаются независимо для каждого виртуального терминала, выбор таблицы перекодировки, определяемой каждой ссылкой, можно производить только из четырех таблиц, записанных в ядре. Поэтому реально все терминалы используют одну и ту же пользовательскую таблицу ACM. То есть, вы можете задать для tty1 использование G0=cp437 и Gl=vtlO 0, а для tty2 использование G0=iso01 и Gl=userl (определяемая пользователем кодировка), но не можете сделать так, что в одно и то же время ttyl использует таблицу user1, а tty2 использует таблицу user2.

Команда `consolechars` используется для изменения ACM, так же как и для задания шрифта и ассоциированной с ним таблицы SFM. С помощью команды consoiechars можно считать консольный шрифт (таблицу экранного шрифта SFM) 8xH из файла и загрузить его в память, а также сохранить в файле шрифт, загруженный в память. Эта же команда служит для зафузки в ядро таблицы перекодировки, а также позволяет переопределить ссылки GO и G1.

В качестве одной из опций команды consoiechars при загрузке экранного шрифта из файла может быть задан размер шрифта по вертикали H. Значение H должно считываться из файла шрифта. Однако файлы некоторых форматов (в частности, файлы, содержащие только битовые образы символов) не содержат прямого указания на этот размер. В таком случае значение опции -н вычисляется исходя из размера файла (обычно -н 8, -н 14 или -н 16). По-

скольку в настоящее время Linux не позволяет программно переключать режим работы дисплея, то выбор подходящего значения H в зависимости от установленного разрешения экрана полностью возлагается на пользователя.

В заключение отметим еще, что файлы с экранными шрифтами по умолчанию располагаются в каталоге /usr/lib/kbd/consolefonts/, а каталог /usr/lib/kbd/consoletrans/ используется для хранения как таблиц ACM, так и SFM.

Графический режим

В графическом режиме нет разбиения экрана на знакоместа, изображение любого символа можно вывести практически в любую позицию экрана. Изображения символов для конкретного набора символов составляют шрифт. Шрифты хранятся в файлах, которые принято называть файлами шрифтов. Вывод символов того или иного шрифта на экран организуется с помощью специального сервера шрифтов. Поэтому проблема русификации графического режима сводится к выбору русифицированного шрифта. Вопрос о том, что такое шрифты и как работает сервер шрифтов, подробно рассмотрен в *разд. 11.4.*

11.1.2. Локализация

Только организовать ввод и вывод символов национального языка еще недостаточно для того, чтобы можно было считать решенным проблему применения компьютеров в той или иной стране. В разных странах в силу сложившихся традиций имеются различия не только в используемом алфавите, но и в способах представления некоторых конкретных данных. Это касается, например, символов, используемых для обозначения валюты, форматов представления даты и времени, обычаев записи (чтения) текстов слева направо или справа налево и т. д. При создании ПО, рассчитанного на применение в разных странах, приходится учитывать такие местные особенности. Кроме того, большие трудности вызывают такие вопросы, как проверка орфографии на конкретном языке, автоматическая расстановка переносов при вводе текста или перевод на данный язык всех меню и служебных сообщений *в* программном обеспечении.

Способ проектирования ПО (включая ОС), при котором возможность многоязыковой поддержки закладывается с самого начала, принято называть *интернационализацией* (кстати, загадочное i18n — это просто сокращение для слова internationalization: i — потом еще 18 букв — п, аналогично, l10n = localization). При интернационализации программного обеспечения КОД не зависит от национальных особенностей. Все языково-зависимые данные сосредотачиваются в особых "объектах локализации", которые разбиты на функциональные группы: категории локализации. При таком подходе лока-

лизация — это процесс настройки программной системы на особенности конкретной страны.

В стандарте POSIX (Portable Operating System) были определены средства локализации, которые состоят из следующих компонентов:

- набор библиотечных (libc) вызовов (locale API): setlocale(), isalpha(), toupper() и т. д;
- исходные тексты описания средств локализации, в том числе файл(ы) описания кодировки (Character Set Definition File);
- наборы данных для локализации, которые в Linux размещаются в каталогах /usr/share/i18n/* и /usr/share/locale/*;
- утилита для получения информации о средствах локализации: locale;
- утилита для изготовления (компиляции) объектов локализации: localedef;
- переменные окружения, для управления средствами локализации: LANG, LC_ALL, LC_CTYPE, LC_TIME, LC_COLLATE, LC_NUMERIC и LC_MONETARY (они соответствуют категориям локализации).

В табл. 11.1 кратко перечислено, на что именно влияет та или иная из этих переменных (или категорий локализации).

Таблица 11.1. *Категории локализации*

Категория	Описание
LC_CTYPE	Определяет правила классификации и преобразования *одиночных* символов. Позволяет правильно определять вид символа: цифра, буква, значок, заглавная буква или прописная и т. д. Другими словами, включает правильную работу системных вызовов isalnum(), isalpha(), iscntrl(), isdigit() и т. п. для местного алфавита. Вдобавок, включает правильный перевод строчных/прописных букв: toupper() и tolower()
LC_COLLATE	Определяет правила сравнения и преобразования *строк.* Позволяет определять лексикографический порядок символов (порядок сортировки) в местном алфавите. Включает правильную работу strcoll() и strxfrm(). Оказывает непосредственное влияние на работу утилит типа sort и т. д.
LC_TIME	Определяет правила национального представления *времени и даты.* Задает именование дней недели, месяцев и т. п. а также задает *способ* написания даты и времени (12/24). Непосредственно влияет на strftime(), а через нее на утилиты date и т. д.
LC_NUMERIC	Определяет правила национального представления *чисел* с плавающей точкой. Влияет на strtod() и форматы %f и %д printf(), scanf()
LC_MONETARY	Определяет правила национального представления денежных величин

Особая переменная LC_ALL служит для обращения *одновременно* ко всем категориям.

Заметим, что включение средств локализации изменяет также некоторые пути поиска, в частности, пути поиска к файлам помощи (man-страницам), так что вначале команда man ищет переведенные man-страницы и только при их отсутствии выдает информацию на английском.

Подводя краткий итог всему вышеизложенному, можно сказать, что процедура русификации системы состоит из следующих этапов:

- настройка средств локализации;
- русификация текстового режима (консоли);
- русификация графического режима;
- русификация используемого ПО;
- русификация процесса печати.

В последующих разделах каждый из этих этапов рассматривается по отдельности.

11.2. Настройка системных средств локализации

11.2.1. Проверка наличия средств локализации

Современные дистрибутивы Linux (а тем более русифицированные) по умолчанию содержат системные средства локализации, перечисленные в предыдущем разделе.

Чтобы убедиться в этом, проверьте, что у вас имеются каталоги /usr/share/locale/* и /usr/share/i18n/*, а также файл /etc/sysconfig/i18n.

Кроме того, дайте команду

```
[user]$ locale
```

В ответ вы должны получить значения, присвоенные переменным окружения для управления средствами локализации. Я, например, увидел следующее:

```
LANG=ru
LC_CTYPE="ru_RU.KOI8-R"
LC_NUMERIC="ru_RU.KOI8-R"
LC_TIME="ru_RU.KOI8-R"
LC_COLLATE="ru_RU.KOI8-R"
```

```
LC_MONETARY="ru_RU.KOI8-R"
LC_MESSAGES="ru_RU.KOI8-R"
LC_ALL=ru_RU.KOI8-R
```

Можно также дать команду locale с параметром, совпадающим с именем одной из переменных окружения, например:

```
[user]$ locale LC_TIME
```

(но я не берусь объяснить то, что вы увидите).

Если результат этих проверок отрицательный (что маловероятно), то установите пакет локализации для русского языка. Найти его можно в коллекции средств локализации по адресу **http://www.ping.be/linux/locales/** или на **www.kiarchive.ru.**

После этого надо сказать Linux, какой язык вы хотите использовать, для чего необходимо задать значения переменных окружения для управления средствами локализации: LC_CTYPE, LC_TIME, LC_COLLATE, LC_NUMERIC, и LC_MONETARY. Вообще говоря, достаточно задать всего одну переменную LC_ALL, задание которой одновременно определяет значения всех перечисленных выше переменные.

Есть еще две переменных окружения, которые имеют отношение к локализации: LANG и LINGUAS. Они действуют примерно так же, как и LC_ALL, в том смысле, что они определяют значения по умолчанию всех других переменных локализации, но, в отличие от LC_ALL, они не переопределяют значений других переменных LC_*, для которых значения заданы отдельно.

Переменная LINGUAS является GNU-расширением переменой LANG. Не все программы знают о существовании этой переменой (хотя не знают очень немногие, и таких все меньше), но эта переменная обладает тем преимуществом, что она позволяет задать несколько вариантов локализации в порядке предпочтения. В большинстве случаев достаточно задать только одну переменную локализации — LANG. Если вы хотите использовать несколько вариантов локализации, то надо задать LANG и LINGUAS. Остальные переменные необходимо задавать только в особых случаях, которые мы здесь не рассматриваем.

11.2.2. Формат задания значений переменных локализации

В качестве значений переменных локализации используются строки формата

```
ll[ _CC[.EEEE]][@dddd]
```

где

- ll — это двухбуквенный код языка в соответствии со стандартом ISO для названий языков (ISO 639), записываемый в нижнем регистре (строчными латинскими буквами);

- CC — это двухбуквенный код страны в соответствии со стандартом ISO для названий стран (ISO 3166), записываемый в верхнем регистре;
- EEEE — это имя (название) таблицы кодировки, записываемое в верхнем регистре;
- dddd — название диалекта языка, который задается в том случае, если названия кодировки недостаточно для однозначного определения варианта локализации.

Для переменной LINGUAS задаваемые значения разделяются двоеточиями. Пример строки задания значений переменной вы уже видели: ru_RU.KOI8-R.

Как уже говорилось, стандарта на названия кодовых таблиц нет, тем более на название диалектов. То, что в шаблоне дано только 4 символа, ничего не означает — символов может быть и больше, и меньше. Квадратные скобки, как всегда, выделяют необязательные элементы. Обязательным в этом шаблоне является только название языка, но в некоторых источниках рекомендуется для русского языка указывать все три части. Напомню еще, что аббревиатура SU теперь означает Судан, а не нашу страну.

11.2.3. Включение средств локализации

Включение системных средств локализации в Red Hat Linux (а, следовательно, и в других дистрибутивах, основанных на Red Hat) осуществляется из файла /etc/profile.d/lang.sh.

Как известно, при старте любого shell сначала выполняется /etc/profile. В Red Hat в /etc/profile прописаны команды, благодаря которым на исполнение вызываются также *все* файлы /etc/profile.d/*.sh.

Значения переменных локализации в файлах lang.sh задаются путем вызова на выполнение файла /etc/sysconfig/i18n. В файле /usr/doc/initscripts-4.16/sysconfig.txt.rus приведены краткие рекомендации относительно того, как задавать значения переменных в файле /etc/sysconfig/i18n, в частности, предлагается для LANG указать в качестве значения просто двухбуквенный код ISO для желаемого языка; для переменной LINGUAS — список кодов языков, разделенных двоеточием, а переменной LC_ALL присвоить имя в соответствии с приведенным выше описанием формата для переменных локализации. Таким образом, в файле /etc/sysconfig/i18n рекомендуется прописать три строки следующего вида:

```
LANG="ru_RU.KOI8-R"
LINGUAS="ru:en"
LC_ALL="ru_RU.KOI8-R"
```

Заметим, что в RU.LINUX.FAQ написано следующее: "Хотя в принципе допустимо задавать короткое именование, вроде LANG=ru_RU или даже LANG=ru,

лучше использовать полное имя: LANG=ru_RU.KOI8-R. Совершенно недопустимо задавать LANG=ru_SU, такой страны больше нет".

Если все, что было перечислено, сделано, можно считать, что локализация включена.

Правда, это верно только для случая, когда вы имеете права суперпользователя root. Но даже если вы простой пользователь Linux-системы и не можете редактировать файл /etc/sysconfig/i18n, то вы все же можете включить локализацию для себя, но несколько иным способом. А именно, поместите в свой файл $HOME/.profile (или в любой файл, который исполняется в процессе логирования пользователя: $HOME/.Xclients, $HOME/.xinitrc или другой) следующие строки:

```
export LANG=ru_RU.KOI8-R
export LINGUAS=ru_RU:en
export LC_ALL="ru_RU.KOI8-R"
```

Вот и все! (О том, как проверить, что локализация заработала, было сказано выше.)

11.3. Русификация консоли

11.3.1. Что нужно сделать

Если вы внимательно прочитали *разд. 9.3* и *11.1*, то представляете, что для русификации консоли (а консоль — это, грубо говоря, совокупность клавиатуры и дисплея) требуется загрузить в драйвер консоли три таблицы:

- ❒ таблицу раскладки клавиатуры;
- ❒ таблицу экранного шрифта (SFM), в которой хранятся изображения символов;
- О таблицу перекодировки символов (ACM).

Таблицы раскладки клавиатуры находятся в каталоге /usr/lib/kbd/keymaps/i386/qwerty. Выбор конкретного файла раскладки задается файлом /etc/sysconfig/keyboard. Этот файл можно отрегулировать вручную, а можно с помощью программы kbdconfig. В последнем случае нужная таблица с раскладкой клавиатуры загружается в память. При изменении файла вручную /etc/sysconfig/keyboard перезагрузка таблицы произойдет только после перезапуска компьютера или после выполнения команды (в примере загружается раскладка из файла ru-win.map):

```
[root]# loadkeys /usr/lib/kbd/keymaps/i386/qwerty/ru-win.map
```

Второй шаг русификации состоит в задании и загрузке таблицы экранного шрифта (SFM) в драйвер дисплея. Эти таблицы хранятся в виде файлов в каталоге /usr/lib/kbd/consolefonts. Загрузка шрифтов осуществляется немного

по-разному в пакетах kbd и consoletools (соответственно, в версиях *5.2* и *6.x* Red Hat Linux). В версии *5.2* загрузка шрифта осуществлялась с помощью команды setfont. Например, чтобы загрузить кодовую страницу из файла Cyr_a8x16, нужно дать команду

```
[root]# setfont /usr/lib/kbd/consolefonts/Cyr_a8x16
```

В версии 6.0 и последующих надо использовать команду consolechars с опцией -f:

```
[root]# consoiechars -f /usr/lib/kbd/consolefonts/Cyr_a8x16
```

(Отметим, что команда setfont тоже сработает, только выдаст предупреждение о том, что надо пользоваться командой consoiechars.)

Файлы шрифтов являются бинарными файлами размером 256×H байтов, содержащими битовые образы для каждого из 256 символов, по одному байту на каждую линию образа и по H байтов на символ (0 < H <= 32). В этом случае о размере шрифта по вертикали можно судить по длине файла. В качестве файлов шрифтов могут использоваться psf-файлы; они имеют тот же самый формат и, кроме того, заголовок размером 4 байта. Некоторые файлы шрифтов содержат сразу три шрифта разного размера (например, 8x8, 8x14, 8x16), тогда в команде consoiechars надо добавить опцию -н, например: -н 16, для выбора одного из размеров. Поскольку ядро Linux не поддерживает пока переключение режимов работы экрана, consoiechars (как и setfont) не может изменить текущий режим EGA/VGA. Таким образом, пользователь полностью ответственен за выбор шрифта, соответствующего текущему режиму экрана.

Соответствие между символами кода ASCII и образами (или изображениями символов) из файла шрифта можно изменить, используя таблицу перекодировки (ACM — Application Charset Map). Если эту таблицу не загрузить, то, например, в программе Midnight Commander вы можете вместо красивых рамочек увидеть столбцы и строки из непонятных символов. Некоторые файлы шрифтов включают таблицу перекодировки шрифта, и тогда consoiechars загрузит эту таблицу. По умолчанию файлы шрифтов находятся в каталоге /usr/lib/kbd/consolefonts, а таблицы перекодировки — в каталоге /usr/lib/kbd/consoletrans.

Таблицу перекодировки в 6-ой версии Red Hat (то есть в пакете consoletools) можно загрузить отдельной командой consoiechars с опцией -m file:

```
[root]# consoiechars -m /usr/lib/kbd/consoletrans/koi2alt
```

Если таблица перекодировки не включена в файл шрифта и не указана в опции -т, то используется "тривиальная" таблица.

В версии 5.2 для загрузки таблицы перекодировки используется команда mapscrn:

```
[root]# mapscrn /usr/lib/kbd/consoletrans/koi2alt
```

В этом случае драйвер консоли должен быть дополнительно переведен в режим перекодировки, задаваемый таблицей, путем вывода на консоль специальной escape-последовательности. Эта последовательность есть <esc> (к для набора символов GO (GO character set) и <esc>)K для набора символов G1 (G1 character set). Заметим, что активизировать эту таблицу необходимо в каждой консоли. При этом команда loadkeys действует одновременно во всех виртуальных консолях, а вот команда mapscrn действует только в той виртуальной консоли, в которой выполнена команда echo -ne '\033(K'.

Замечание

ESC (к требуется, когда загружается альтернативная кодировка и активизируется таблица перекодировки псевдографики командой mapscrn koi2alt. Если шрифт koi-8, то никаких ESC (K не надо.

Замечание

Все это не действует из-под Midnight Commander!

Существует еще таблица перекодировки для символов UNICODE. Некоторые файлы шрифтов включают эту таблицу, и она будет загружена командой consolechars, если только не задана опция --force-no-sfm. Отдельно загрузить таблицу перекодировки символов UNICODE можно командой consoiechars с опцией -и (см. руководство man).

Итак, для того, чтобы русифицировать консоль, нужно выполнить следующую последовательность команд:

- для версии 5.2 Red Hat:

```
loadkeys /usr/lib/kbd/keytables/i386/qwerty/ru.map
setfont /usr/lib/kbd/consolefonts/Cyr_a8x16
mapscrn /usr/lib/kbd/consoletrans/koi2alt
echo -ne '\033(K'
```

- для версии 6.0 Red Hat (и последующих):

```
loadkeys /usr/lib/kbd/keytables/i386/qwerty/ru.map
consoiechars -f /usr/lib/kbd/consolefonts/Cyr_a8x16
consoiechars -m /usr/lib/kbd/consoletrans/koi2alt
```

Но выполнять эту последовательность команд после каждого перезапуска компьютера, да еще в каждой виртуальной консоли, слишком обременительно. Поэтому рассмотрим вкратце, как русификация выполняется в дистрибутиве Black Cat Linux.

11.3.2. Как это сделано в дистрибутиве Black Cat

Во-первых, в файле /etc/sysconfig/i18n вводится новая переменная: в версии *5.2* это переменная SCRNMAP, а в версии 6.02 — SYSFONTACM. Этой переменной по умолчанию присваивается значение koi2alt. Вот стандартный файл i18n из Black Cat Linux версии 6.02:

```
LANG=ru
LINGUAS=ru
LC_ALL=ru_RU.KOI8-R
SYSFONT=RUSCII_8x16
SYSFONTACM=koi2alt
```

Вызов файла i18n для задания значений переменных осуществляется из скрипта /sbin/setsysfont, из которого вызываются также команды setfont и mapscrn (в версии 5.2) или consolechars (в версии 6.0). Вот этот скрипт из Black Cat Linux версии 5.2:

```
-------------------------------------
#!/bin/sh
if [ -f /etc/sysconfig/i18n ]; then
        . /etc/sysconfig/i18n
fi
if [ -x /usr/bin/setfont ]; then
         if t -n "$SYSFONT" ]; then
                  /usr/bin/setfont $SYSFONT
         fi
if [ -x /usr/bin/mapscrn ]; then
         if [ -n "$SCRNMAP" ]; then
/usr/bin/mapscrn $SCRNMAP
         fi
fi
else
         echo "can't set font"
         exit 1
fi
-------------------------------------
```

Как видно, при вызова скрипта /sbin/setsysfont выполняются команды setfont Cyr_a8x16 и mapscrn koi2alt. После этого, для включения в ядре кодовой таблицы пользователя, необходимо выдать на каждую виртуальную консоль последовательность \033 (к. Это реализовано путем добавления этой

последовательности к файлу /etc/issue, который генерируется при загрузке системы скриптом /etc/rc.d/rc.local и вызывается на исполнение при логировании каждого пользователя. Вот пример скрипта /etc/rc.d/rc.local из версии 5.2:

```
-------------------------------------
#!/bin/sh
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
if [ -f /etc/blackcat-release ]; then
        R=$(cat /etc/blackcat-release)
elif [ -f /etc/redhat-release ]; then
        R=$(cat /etc/redhat-release)
else
R="release 3.0.3"
fi
arch=$(uname -m)
a="a"
case "_$arch" in
_a*) a="an";;
_i*) a="an";;
esac
# This will overwrite /etc/issue at every boot. So, make any changes you
# want to make to /etc/issue here or you will lose them when you reboot.
. /etc/sysconfig/i18n
echo "" > /etc/issue.net
echo "Black Cat Linux $R" >> /etc/issue.net
echo "Kernel $(uname -r) on $a $(uname -m)" >> /etc/issue.net
if [ -n "$SCRNMAP" ]; then
        echo -ne "\033(K" > /etc/issue
else
        echo "" > /etc/issue
fi
if [ -f /usr/bin/linux_logo ]; then
        /usr/bin/linux_logo -n -o 2 >> /etc/issue
        echo "" >> /etc/issue
fi
cat /etc/issue.net >> /etc/issue
echo "" >> /etc/issue
-------------------------------------
```

В версии 6.02 все происходит примерно так же. Просмотрите упомянутые выше файлы и вы убедитесь в этом сами.

11.3.3. Переключение кодировок

Теперь поговорим о том, как "на лету" изменить кодировку символов. Необходимость в этом возникает в тех случаях, когда просматриваешь какой-то файл и вместо читаемого текста видишь непонятную абракадабру. В таких случаях хочется превратить ее в нормальный текст нажатием пары "горячих" клавиш. Можно, конечно, для каждого из необходимых шрифтов создать специальную команду, записав в небольшой файл приведенные выше команды, естественно соответствующим образом модифицированные. Однако ясно, что это не совсем удобно. В этом отношении хочется работать так же комфортно, как в программе FAR Е. Рошаля, где в том случае, когда просматриваешь файл по нажатии клавиши <F3> или редактируешь его по нажатии клавиши <F4>, достаточно набрать комбинацию клавиш <Shift>+ +<F8> и получаешь возможность перекодировать выводимый на экран текст в любую из нескольких кодировок. Можно считать, что это уже роскошь, но что поделаешь, все мы быстро привыкаем к хорошему, и отказываться уже трудно.

2 июля 2000 г. на сайте **linux.ru.net** появилось сообщение о заплате (патче) к Midnight Commander версии 4.5.50, которая позволяет производить перекодировку как в FAR, по нажатии клавиш <Ctrl>+<T>. Тогда это можно было делать только во встроенной программе просмотра. На сегодня автор патча В. Студенников доработал его, так что перекодировка стала возможна и во встроенном редакторе (см. **http://www.linux.zp.ua:8100/mc/ или http://www.sama.ru/~despaire/mc/my-patches.html**).

Это что касается перекодировки "на лету". Как видите, такая перекодировка является функцией отдельных программ просмотра файлов. Если перекодировка "на лету" недоступна, то можно перекодировать сам текстовый файл. Для преобразования потока символов из одной кодовой страницы (charset) в другую, в стандарте POSIX существуют утилита iconv и функция `iconv()`. О других программах перекодировки мы поговорим в *гл. 12.*

11.4. Русификация X Window

Так же, как и в текстовом режиме, в графическом русификация определяется таблицей раскладки клавиатуры и файлами шрифтов. Впрочем, о том, как разобраться с клавиатурой, было достаточно сказано в *разд. 9.3.* Давайте теперь подробно рассмотрим, как организовать вывод на экран символов кириллицы.

11.4.1. Немного о терминологии

Для начала надо пояснить некоторые термины, часть из которых появилась еще в те времена, когда типографии называли словолитнями, а часть — уже в наше время, вместе с широким внедрением компьютеров в типографское дело. Некоторые из этих терминов получены прямым заимствованием из английского языка, например, фонт-сервер, рендеринг, глиф.

Пожалуй, важнейшим типографским термином является *шрифт*. В обычном, или "бытовом", если так можно выразиться, понимании, шрифт — это только набор графических изображений символов, который служит для представления букв на бумаге или другой поверхности (например, экране, причем экране в широком смысле слова - экране кино, телевизора, компьютера). Поскольку можно сказать, что шрифты появились вместе с изобретением книгопечатания, история разработки и создания шрифтов исчисляется уже веками. За это время создание шрифтов превратилось в особый вид деятельности (может быть, даже в отрасль науки) со своей терминологией, инструментарием, традициями, школами и т. д. Новый импульс эта наука получила с переходом на компьютерные технологии.

Субъективное замечание

Вместе с широким распространением компьютеров в русском языке появился новый термин — *фонт.* Первоначально это была простая подмена русскоязычного термина его английским эквивалентом, применявшаяся только в компьютерном сленге. Однако постепенно стало проявляться различие в употреблении этих терминов. Понятие "фонт" стало охватывать понятие "шрифт", но начало трактоваться в более широком смысле, подразумевая не только собственно шрифт, как набор изображений символов, но и способ формирования этих изображений на экране и/или бумаге и способ сохранения этих изображений в файле. Кроме того, словом "фонт" называют просто файл, содержащий закодированные неким образом изображения символов шрифта (например, битовые карты изображений символов). Мне кажется, что в последнем случае правильнее употреблять термин "файл фонта", а термин "фонт" следует употреблять во втором смысле: как совокупность собственно шрифта (как набора изображений символов), способа вывода этих изображений символов на печать или экран и способа сохранения шрифта в виде файла. Таким образом, можно сказать, что шрифт — понятие типографское, а фонт — компьютерное. Впрочем, такая трактовка этих терминов не является пока общепринятой и всюду ниже употребляется единый термин — шрифт.

Существует два основных типа шрифтов: растровые и контурные (их еще называют векторными).

В *растровых шрифтах* изображение символа составляется из отдельных точек и сохраняется в файле в виде двоичной матрицы или "битовой карты" (bitmap). Поэтому такие шрифты можно назвать матричными. Растровые шрифты являются *аппаратно-зависимыми* — они предназначены для получения только определенных разрешений. Матричный шрифт с разрешением

75 точек на дюйм (75 dpi) даст только 75 точек на дюйм даже на принтере, обеспечивающем разрешение 1200 dpi. Для обеспечения возможности изменения размера шрифта приходится либо хранить в файле битовые карты для разных размеров шрифта, либо изображение приходится увеличивать или уменьшать в целое число раз, что приводит к деформации изображений символов. Кроме того, при использовании растровых шрифтов для печати необходимы дополнительные принтерные шрифты, соответствующие экранным.

Матричные экранные шрифты обычно используются в терминальных окнах, в консоли и в простых текстовых редакторах, где отсутствие масштабируемости и тот факт, что они не могут использоваться для печати, не так существенны.

В *контурных* (или векторных) *шрифтах* для вывода изображения символа применяется математический аппарат, основанный на использовании кривых Безье. Как известно, для описания прямой линии нам необходимо и достаточно задать только две точки — начало и конец прямой. Кривая Безье задается своими начальной и конечной точками, и, кроме того, набором промежуточных точек, к которым стремится данная кривая. Регулируя отклонение кривой от этих промежуточных точек, можно достичь гладких изгибов кривой Безье. Каждый символ (точнее говоря, глиф, см. ниже) контурного шрифта представляется набором контуров, задающих границы изображения. Контуры описываются замкнутыми кривыми Безье. При выводе изображения на экран или принтер производится преобразование заданного таким образом изображения символа в матрицу точек (грубо говоря, в растровое изображение соответствующего размера).

Достоинством контурных шрифтов является их независимость от устройства вывода — они одинаково выглядят как на экране, так и на принтере. Каждый символ может иметь различное начертание: курсив, полужирный, подчеркивание и обычное. Кроме того, он может иметь абсолютно любой размер (то есть эти шрифты являются масштабируемыми), что невозможно в случае с растровыми шрифтами.

Отдельным элементом контурного шрифта, выводимым на экран или принтер, является не символ, что казалось бы более естественным для незнакомого с предметом человека, а так называемый *глиф* (этот термин является прямым заимствованием соответствующего английского термина "glyph"). Глиф может представлять либо отдельный символ, либо два или большее число символов (так называемые *лигатуры),* либо часть изображения символа.

Еще одно понятие, которое необходимо для описания и классификации шрифтов — это *сериф.* Серифы — это небольшие выступы у оснований символов и в верхней части изображений символов. Например, буква "i" в таких шрифтах как Times Roman имеет серифы у основания и в верхней части вертикальной черты. Серифы появились в изображениях символов шрифтов относительно недавно и позволили улучшить зрительное восприятие симво-

лов. Шрифты, в которых используются серифы, считаются более читабельными, чем шрифты без серифов. При малых размерах символов серифы обычно не используются, т. к. отсутствие лишних элементов позволяет легче их воспринимать. Например, фирма Microsoft рекомендует шрифт Verdana как наиболее подходящий для очень малых размеров символов при выводе на экран.

Межсимвольное расстояние задает интервал, на который символы шрифта отстоят друг от друга. Бывают шрифты с фиксированным межсимвольным расстоянием и фиксированным размером символов (речь идет о ширине, высота символов у обычных шрифтов и так фиксирована) или *fixed width fonts,* и шрифты переменной ширины — *variable width fonts.* Шрифт фиксированной ширины выглядят как текст, напечатанный на пишущей машинке, поскольку все буквы имеют одну и ту же ширину. Это качество желательно для таких применений, как компьютерная консоль, но нежелательно для больших печатных документов. Шрифты с фиксированной шириной символов часто используются, например, в примерах, иллюстрирующих компьютерные команды.

В шрифтах переменной ширины для повышения читабельности и улучшения внешнего вида текста варьируется как ширина отдельных символов, так и межсимвольное расстояние, для чего используется техника трекинга и кернинга.

Трекинг — это специальные приемы улучшения оптических свойств текста при больших и малых размерах за счет изменения расстояния между символами. Трекинг определяет межсимвольное расстояние как функцию от размера шрифта.

Кернинг улучшает качество восприятия текста, изменяя межсимвольное расстояние в некоторых парах символов, которые называются парами кернинга. Например, при изображении пары символов "To" маленькая буква "o" должна быть приближена к заглавной "T" и даже размещаться частично "под" этой "T", что повышает качество отображения.

Для достижения наилучшего качества отображения символов шрифта при работе с малыми размерами используется такое средство, как *хинтинг* (hinting). При отображении символа в малом размере учитывают, какие части контура символа следует использовать, а какие нет. Для этого в описание контура символа помещаются "хинты" — специальные инструкции, указывающие каким образом надо изменять форму контура для достижения наилучшего качества. Как правило, редакторы шрифтов сами выполняют хинтинг, избавляя пользователя от дополнительной работы над шрифтом.

Совокупность данных о ширине символов, межсимвольном расстоянии, а также информации, необходимой для кернинга и трекинга, называют *метриками шрифта.*

Варианты шрифтов. Каждый шрифт обычно содержит несколько вариантов изображений каждого символа. Большинство шрифтов содержат такие варианты как полужирный (bold), курсив (italic) и полужирный курсив (bold-italic). Некоторые шрифты имеют дополнительные варианты, например, жирный или small caps, когда строчные буквы заменяются уменьшенными заглавными (этот вариант шрифта чаще всего используется для создания заголовков).

11.4.2. Форматы файлов шрифтов

В недавнем прошлом буквально каждый графический редактор или издательская программа использовали свой формат файлов шрифтов и, как правило, одни программы не поддерживали форматы других. Со временем число реально используемых форматов существенно сократилось, т. к. происходит своеобразный "естественный отбор".

Растровые шрифты (Bitmap Fonts)

Существует два типа матричных шрифтов: принтерные матричные шрифты (bitmap printer fonts), такие как шрифты pk, генерируемые программой dvips, и матричные экранные шрифты, используемые в системе X Window и в консольном режиме. Файлы матричных шрифтов обычно имеют расширения bdf или pcf.

Шрифты Type 1

В Linux ведущим форматом шрифтов является Adobe PostScript. Вообще, PostScript — это язык для описания страницы документа, он используется для вывода страницы с текстом и графикой на экран и на принтер (который поддерживает этот язык). Шрифты Adobe Type основаны на данном языке. Существуют две основные разновидности: Type 1 PostScript font и Type 3 PostScript font. Стандартом является Type 1. Шрифт Type 3 используется довольно редко. Шрифты Type 1 применяются в различных графических и издательских программах, например Adobe PageMaker, но они не являются стандартными в Microsoft Windows и поэтому их использование ограничивается программами, специально разработанными для поддержки данных шрифтов.

Язык PostScript традиционно используется для печати в UNIX, и поэтому шрифты Type 1 широко применяются в Linux. Они поддерживаются в графической системе X Window и программе управления печатью ghostscript.

Обычно шрифт Type 1 для UNIX задается двумя файлами: файлом метрик с расширением afm (adobe font metric), и файлом контуров, который имеет расширение pfb (printer font binary) или pfa (printer font ascii). Файл контуров содержит все глифы, а файл метрик содержит метрики шрифта.

Шрифты Type 1 для других платформ могут поставляться в других форматах. Например, шрифты PostScript для Windows часто используют формат PFM для файла метрик.

Шрифты Type 3

Эти шрифты распространяются примерно таким же образом, как шрифты Type 1 -- в виде пары afm-файла метрик и pfa-файла контуров. Хотя они поддерживаются стандартом PostScript, но не поддерживаются в X Window, поэтому имеют ограниченное применение.

Шрифты TrueType

Технология TrueType была создана фирмой Apple для использования в операционной системе MacOS в 1990 году. Позже лицензию на эту технологию приобрела фирма Microsoft. Благодаря гигантской популярности MS Windows шрифты TrueType стали ведущим форматом для компьютерных шрифтов. Шрифты TrueType хранят метрики и информацию о начертаниях символов в одном файле (обычно с расширением ttf). Недавно были разработаны серверы шрифтов, или фонт-серверы, которые сделали шрифты TrueType доступными для X Window. С некоторых пор стандарты postscript и ghostscript тоже поддерживают шрифты TrueType. В силу этого, а также из-за того, что имеется большое количество высококачественных и доступных шрифтов TrueType, они становятся все более популярными в Linux.

Шрифты Type 42

Шрифты Type 42 в действительности являются просто шрифтами TrueType с дополнительным заголовком, что делает возможным обработку их интерпретатором PostScript. Большинство приложений, таких как ghostscript и SAMBA могут работать с этими шрифтами. Однако, если вы имеете PostScript-принтер, может оказаться необходимым явно создать файлы шрифтов типа Type 42.

Сравнение форматов Type 1 и TrueType

Несмотря на длительное противостояние между сторонниками шрифтов Type 1 и TrueType, эти форматы имеют много общего. Оба они представляют масштабируемые контурные шрифты. Разница в том, что шрифты Type 1 используют для построения глифов кривые Безье третьей степени в отличие от квадратичных кривых, на которых строятся шрифты TrueType. Теоретически это является преимуществом, поскольку тем самым Type 1 включает все кривые, которые можно построить с помощью TrueType. На практике, однако, разница очень незначительна.

Шрифты TrueType имеют преимущество, заключающееся в том, что обеспечивают лучшую поддержку хинтинга (шрифты Type 1 тоже поддерживают

хинтинг, но не так эффективно). Это существенно только для устройств с низким разрешением, таких как экраны (улучшение хинтинга не дает заметной разницы на принтерах с разрешением 600 dpi даже при малых размерах точки). Улучшенный хинтинг не имеет большого практического значения еще и потому, что шрифты TrueType с хорошим хинтингом встречаются достаточно редко. Причина этого в том, что пакеты программного обеспечения для создания шрифтов, поддерживающих хинтинг, слишком дороги для большинства дизайнеров шрифтов. Только крупные фирмы, такие как Monotype, создают шрифты с полноценной поддержкой хинтинга.

Основная разница между шрифтами TrueType и Type 1 состоит в доступности и поддержке приложениями. Широкое распространение шрифтов TrueType для Windows привело к тому, что многие страницы на сайтах Интернета создаются в предположении, что определенные шрифты TrueType установлены на компьютере пользователя. Многие пользователи имеют на своих компьютерах большое число шрифтов TrueType, поставляемых с приложениями Windows. Однако, под Linux большинство приложений поддерживают шрифты Type 1, но не поддерживают на том же уровне шрифты TrueType. Кроме того, некоторые из основных производителей шрифтов поставляют свои шрифты в формате Type 1. Например, фирма Adobe выпускает очень мало шрифтов TrueType. Учитывая, что преобразование из одного формата в другой не может быть проведено без определенных потерь качества, надо стараться использовать в каждом приложении те шрифты, на использование которых это приложение рассчитано.

Метафонт

Метафонт был разработан Дональдом Кнутом (Donald E Knuth) как часть его типографской системы TEX. Метафонт — это язык программирования графики (a graphics programming language), подобный PostScript. Сфера применения этого языка шире, чем просто вывод изображений символов. У метафонта имеется ряд очень привлекательных качеств. Одна из очень важных особенностей — то, что масштабирование выполняется очень изящно. Символы метафонта Computer Modern имеют различный вид при размере 20 точек и 10 точек. Изображения символов изменяются при изменении размера, потому что для малых размеров желательно увеличить ширину несколько больше, чем высоту (это делает символы больших размеров более элегантными, а символы маленьких размеров более читабельными).

Файлы метафонтов обычно имеют расширение .mf. При выводе на печать или экран они преобразуются в растровые шрифты, определяемые устройством вывода. Это преобразование, хотя и дает высокое качество, осуществляется медленно, так что эти шрифты не очень удобны для использования в WYSIWYG-приложениях.

11.4.3. Конфигурация X-сервера

Теперь, когда введены все используемые в последующем термины и даны необходимые сведения о форматах шрифтов, можно перейти к рассмотрению того, как настроить ваш Linux на корректную работу со шрифтами. При этом я буду ориентироваться на пользователей дистрибутива Red Hat и его аналогов (я проверял изложенные ниже рекомендации на дистрибутиве Black Cat 6.02).

Выводом изображений и, в частности, изображений символов, на экран занимается X-сервер. Когда приложение обращается к X-серверу с требованием вывести на экран какой-то текст, X-сервер обращается к своему конфигурационному файлу XF86Config, в котором должен быть определен перечень каталогов со шрифтами (FontPath). Откройте файл XF86Config (обычно он находится в каталоге /etc/X11/ или /usr/X11R6/lib/X11/) с помощью любого текстового редактора. Недалеко от начала файла в секции Files вы должны увидеть примерно такие строки:

```
FontPath "/usr/X11R6/lib/X11/fonts/misc/"
FontPath "/usr/X11R6/lib/X11/fonts/Type1/"
FontPath "/usr/X11R6/lib/X11/fonts/Speedo/"
FontPath "/usr/X11R6/lib/X11/fonts/75dpi/"
FontPath "/usr/X11R6/lib/X11/fonts/100dpi/"
```

Это и есть перечень каталогов шрифтов X-сервера, который в англоязычной документации называется коротко FontPath. Порядок перечисления каталогов в этом перечне существенен: когда приложение запрашивает вывод текста на экран определенным шрифтом, X-сервер поочередно просматривает каталоги из FontPath и использует для вывода текста первый шрифт, который соответствует запросу приложения.

Если используются установки, задаваемые по умолчанию, то растровые шрифты с разрешением 75 dpi обычно оказываются размещены в этом перечне перед шрифтами с разрешением 100 dpi. Следствием этого может оказаться то, что на экранах с высоким разрешением символы будут очень маленькими. Если такой эффект у вас проявляется, поменяйте порядок перечисления каталогов в FontPath.

Еще один момент, относящийся к растровым шрифтам, связан с их масштабированием. При отображении символов большого размера с помощью таких шрифтов может оказаться, что изображение символа распадается на отдельные точки. Этот эффект проявляется, например, в навигаторе Netscape при выводе крупных заголовков. Чтобы избежать этого, вы можете указать после имени каталога ключевое слово unsealed (не масштабировать), отделив его двоеточием:

```
FontPath "/usr/X11R6/lib/X11/fonts/misc/:unscaled"
FontPath "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
```

```
FontPath "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
FontPath "/usr/X11R6/lib/X11/fonts/Type 1/"
FontPath "/usr/X11R6/lib/X11/fonts/Speedo/"
FontPath "/usr/X11R6/lib/X11/fonts/misc/"
FontPath "/usr/X11R6/lib/X11/fonts/100dpi/"
FontPath "/usr/X11R6/lib/X11/fonts/75dpi/"
```

При этом можно, как в приведенном выше примере, указать как возможность использования масштабируемых, так и немасштабируемых шрифтов, определив свои предпочтения порядком перечисления строк в XF86Config.

Кстати, раз уж вы заглянули в XF86Config, неплохо заодно проверить и другие установки, определяющие конфигурацию X-сервера. Неправильное задание параметров работы монитора может доставить гораздо больше головной боли, чем неправильный выбор шрифта. В частности, убедитесь, что частота обновления экрана выбрана максимально возможной для вашей аппаратной конфигурации (85 Гц — это великолепно, 75 Гц — неплохо, а 60 Гц — это просто вредно для вашего зрения).

11.4.4. Фонт-серверы

Хотя вывод символов на экран с помощью X-сервера графической подсистемы XFree86 и обеспечивает вывод текста, однако, этот способ имеет два крупных недостатка. Во-первых, шрифты должны располагаться на том же компьютере, на котором запущен X-сервер. Во-вторых, не обеспечивается вывод шрифтов TrueType. Для преодоления этих недостатков были разработаны специальные серверы шрифтов или фонт-серверы (надо отметить, что в версии XFree86 *4.x* второй недостаток уже отсутствует, так что специальный фонт-сервер уже не нужен, если речь идет только о поддержке шрифтов TrueType).

В настоящее время существует три разных фонт-сервера: xfs, xfstt и xfsft.

Фонт-сервер xfs

Если вы пользуетесь дистрибутивом, основанным на Red Hat (Mandrake и т. п.), то фонт-сервер xfs у вас, вероятно, установлен. Сообщение о запуске фонт-сервера xfs появляется на экране монитора в процессе загрузки, а, кроме того, соответствующее сообщение об успешном запуске xfs можно найти в файле /var/log/messages.

В случае применения xfs в файле XF86Config вместо перечня каталогов со шрифтами вы увидите всего одну строку следующего вида:

```
FontPath "unix/:-1"
```

Эта строка является ссылкой на номер порта, который будет использоваться для связи с фонт-сервером.

Использование фонт-сервера не означает, что имеет место полный отказ от перечня каталогов со шрифтами. Только теперь этот перечень переносится в конфигурационный файл программы xfs. По умолчанию это файл /etc/X11/fs/config. В секции catalogue этого файла и перечислены теперь все каталоги со шрифтами. Соответствующая секция файла /etc/X11/fs/config должна выглядеть примерно так:

```
catalogue = /usr/X11R6/lib/X11/fonts/misc:unscaled,
            /usr/X11R6/lib/X11/fonts/100dpi:unsealed,
            /usr/X11R6/lib/X11/fonts/75dpi:unscaled,
            /usr/X11R6/lib/X11/fonts/Type 1,
            /usr/X11R6/lib/X11/fonts/Speedo,
            /usr/X11R6/lib/X11/fonts/misc,
            /usr/XllR6/lib/Xll/fonts/100dpi,
            /usr/XllR6/lib/Xll/fonts/75dpi,
            /usr/local/share/fonts/ttfonts
```

(обратите внимание на отсутствие запятой в последней строке).

Правда, вы можете по-прежнему включить строки с указанием путей к каталогам шрифтов в файл XF86Config (вместе со строкой FontPath "unix/:-1), но они будут обрабатываться не фонт-сервером, а X-сервером.

xfs от Redhat способен обслуживать как шрифты Type 1, так и шрифты TrueType.

Фонт-серверы xfstt и xfsft

Если ваш дистрибутив не включает фонт-сервер xfs, вы можете воспользоваться одним из альтернативных фонт-серверов

- **xfstt** (**http://metalab.unc.edu/pub/Linux/X11/fonts/**);
- **xfsft** (**http://www.dcs.ed.ac.uk/home/jec/programs/xfsft/**).

xfstt — это фонт-сервер, созданный исключительно с целью обслуживания шрифтов TrueType на локальной машине. Поэтому он поддерживает только шрифты TrueType и не может обслуживать шрифты на нескольких машинах в сети. Учитывая это, предпочтительнее пользоваться фонт-сервером xfsft или фонт-сервером xfs от Red Hat, о котором мы уже говорили. Заметим, что фонт-сервер xfsft послужил основой для доработки xfs, и модуль работы со шрифтами в XFree86 версии 4 создан разработчиком xfsft. Наверное, поэтому этот фонт-сервер перестал поддерживаться разработчиком, так что вам лучше сразу ориентироваться на xfs или переходить на XFree86 версии 4.

11.4.5. Ревизия шрифтового хозяйства

Установлен ли фонт-сервер?

Давайте начнем такую ревизию с проверки того, корректно ли установлен фонт-сервер xfs. Вначале запустите команду:

```
[root]# ps ax | grep xfs
```

Если xfs уже установлен в вашей системе (в противном случае установите его), вы должны увидеть строку примерно такого вида:

```
401 ? S 0:04 xfs -droppriv -daemon -port -1
```

По этой строке можно определить, какой порт использует эта программа. Этот же номер должен быть указан в строке вида

```
FontPath "unix/:port_number"
```

в секции Files в конфигурационном файле X-сервера (в xfree86 это файл /etc/X11/XF86Config). Скорее всего секция Files в этом случае вообще содержит только одну строку FontPath, например, такую:

```
FontPath "unix/:-1"
```

Впрочем, можно не искать файл /etc/X11/XF86Config, а выполнить команду:

```
[root]# xset -q
```

в выводе которой вы должны увидеть такие строки:

```
FontPath
        "unix/:-1"
```

Если перечень каталогов шрифтов XFree86 содержит строку типа unix: /port_number, где port_number совпадает с номером порта, используемым программой xfs (вы получили его по команде ps), то сервер xfs у вас установлен корректно. В противном случае вы должны добавить ссылку на него в перечень каталогов шрифтов XFree86 либо с помощью команд:

```
[root]# xset fp+ unix/:port_number
[root]# xset fp rehash
```

либо путем непосредственной корректировки файла /etc/X11/XF86Config и последующего перезапуска X Window.

Для редактирования файла /etc/X11/XF86Config вы должны иметь права суперпользователя. Если вы таких прав получить не можете, то для корректной установки фонт-сервера вы должны обратиться к администратору.

Какие шрифты имеются в вашей системе?

Давайте теперь посмотрим, какие шрифты установлены в системе. Поскольку вы уже знаете (загляните еще раз в FontPath), в каких каталогах находятся

файлы шрифтов, вы можете непосредственно просмотреть эти каталоги. Но одного наличия файла со шрифтом еще недостаточно для того, чтобы шрифт был доступен для X-сервера. Для того чтобы увидеть список шрифтов, известных X-серверу, лучше воспользоваться командой `xlsfonts`, которая выводит на экран перечень таких шрифтов. Если запустить ее с опцией `-lll`, то она дополнительно выдаст массу информации о каждом шрифте. Перенаправьте вывод в файл:

```
[root]# xisfonts > fontlist
```

и вы получите список доступных шрифтов в файле fontlist.

Но для того, чтобы этот список прочитать, надо иметь представление о том, как именуются шрифты. Без этого прочитать полученный файл fontlist будет очень трудно.

Существует стандарт консорциума X (X Consortium) на имена шрифтов для X Window System, который называется *X Logical Font Description Conventions* (обычно упоминаемый как XLFD). Полное описание его дано в [П13.11] *(см. приложение)*. В соответствии с этим стандартом имя шрифта состоит из 14 полей:

- **foundry** (`fndry`) — производитель шрифта (Adobe, Bitstream и т. п.);
- **family** (`fmly`) — название семейства шрифтов (например, Times);
- **weight** (wght) — толщина (bold, demibold, medium);
- **slant** (slant) — наклон (roman, italic, oblique);
- **set width** (swdth) — ширина (normal, condensed, double wide);
- **add style** (`adstyl`) — стиль (serif, sans serif, decorated);
- **pixel size** (`pxlsz`) — размер символа по вертикали (в пикселах, 0 означает масштабируемый шрифт);
- **point size** (ptsz) — размер символа по горизонтали;
- **resolutionX** (resx) — разрешение по горизонтали;
- **resolutionY** (resy) — разрешение по вертикали;
- **spacing** (spc) — ширина символов (пропорциональный, моноширинный);
- **avg width** (`avgWdth`) — среднее значение ширины глифов шрифта;
- **registry** (rgstry) — название стандарта на кодировку символов (koi-8, iso-8859);
- **encoding** (encdng) — язык или кодовая страница (r, и).

Здесь в начале каждой строки указано наименование поля, затем (в скобках) сокращение этого наименования, используемое в программе xfontsel (о ней чуть ниже), после чего дается перевод (приблизительный) наименования поля, В скобках после русского перевода названия поля приводится по не-

сколько примеров возможных значений этого поля, которые поясняют его назначение.

При задании конкретного шрифта поля в его имени принято разделять дефисами. Приведем пару примеров имен в стандарте XLFD:

```
-adobe-times-medium-r-normal-*-14-140-75-75-p-74-iso8859-1
-misc-fixed-medium-i-semicondensed-*-13-120-75-75-c-60-koi8-r
-adobe-courier-bold-o-normal-*-10-100-75-75-M-60-iso8859-1
```

(если какое-то поле не определено, то в соответствующей позиции ставится звездочка; таким образом можно одной строкой задать множество шрифтов).

В качестве параметра команде `xlsfonts` можно указать имя конкретного шрифта или семейства шрифтов (этой возможностью просто необходимо воспользоваться, если вы задали опцию `-lll`, иначе масса полученной информации окажется слишком велика).

Но просмотр списка шрифтов, выдаваемого командой `xlsfonts`, мало информативен в том смысле, что при этом вы не видите изображений символов, которые будет давать на экране данный шрифт. В этом плане гораздо удобнее воспользоваться программой xfontsel, которая работает в графическом режиме и выводит в своем окошке изображения некоторых символов данного шрифта, позволяющих представить, как будет выглядеть выводимый текст.

Эти две команды могут оказаться полезными, как для определения того, какие шрифты уже имеются в системе, так и для проверки того, что новые шрифты успешно установились. Я не буду здесь подробно описывать, как пользоваться этими командами. Воспользуйтесь соответствующими man-страницами или системой info.

На мой взгляд, пользователь обычно руководствуется в выборе шрифта только следующими признаками из перечисленных выше: семейство шрифтов (`fmly`), вариант шрифта — жирный шрифт или обычный (wght), наклон (slant), ширина шрифта (`sWdth`), размер шрифта в пикселах (`pxlsz`), стандарт (rgstry) И ЯЗЫК (encdng).

Попробуйте выбирать разные значения этих параметров в программе xfontsel и вы получите неплохое представление о том, какие шрифты установлены в вашей системе. Для русскоязычных пользователей выбор шрифта для просмотра стоит начать с двух последних полей. Задайте для поля rgstry значение `koi8`, а для поля encdng — значение `r`, и вы увидите сколько русскоязычных шрифтов в кодировке koi8-r у вас установлено. Кириллические шрифты задаются также значениями iso8859-5 в двух последних полях.

Кроме xisfonts и xfontsel существуют еще несколько программ для просмотра установленных в системе шрифтов.

- Чтобы увидеть полный набор символов шрифта, можно воспользоваться командой xfd -fn fontname. В качестве fontname здесь можно использовать как полное имя шрифта, так и строку с символами маскирования (*), а также синонимы имен шрифтов, заданные в файле font.alias.

  Пример:

  ```
  xfd -fn -*-helvetica-medium-r-* &
  ```

- В графической среде KDE имеется менеджер шрифтов, который тоже показывает все установленные шрифты, а также позволяет удалить некоторые (но устанавливать новые, кажется, не умеет!).
- В Gnome имеются утилиты font selector, character picker и gfontview.

Файлы fonts.dir, fonts.alias и fonts.scale

Если вы последуете приведенному выше совету и выполните команду

```
xlsfonts > fontlist
```

а после этого пересчитаете число файлов в каталогах, перечисленных в FontPath, то скорее всего обнаружите, что в fontlist перечислено гораздо больше шрифтов, чем имеется файлов со шрифтами. Чтобы понять, почему это так, надо разобраться с файлами fonts.dir, fonts.alias и fonts.scale. Если вы заглянете в любой каталог со шрифтами, то найдете там по крайней мере файл fonts.dir, а может быть и два других: fonts.alias и fonts.scale. Для чего же они нужны?

Структура файла fonts.dir очень проста и из нее становится ясно, зачем этот файл нужен. Каждая строка файла fonts.dir (кроме первой) содержит имя одного из файлов со шрифтом, находящегося в том каталоге, где расположен данный файл fonts.dir, вслед за которым (после пробела или символа табуляции) указывается имя содержащегося в этом файле шрифта.

Пример:

```
koi12x24.pcf.gz -cronyx-fixed-medium-r-normal--24-170-100-100-c-120-koi8-u
```

Первая строка файла fonts.dir содержит число шрифтов, перечисленных в этом файле (и, соответственно, имеющихся в данном каталоге со шрифтами). Файл fonts.dir совершенно необходим для того, чтобы X-сервер мог работать со шрифтами. По-видимому (я могу судить об этом только на основании проведенных экспериментов, поскольку в литературе такого описания не встречал), при запуске X-сервера или фонт-сервера на основе файлов fonts.dir из каталогов шрифтов в оперативной памяти создается таблица доступных для системы шрифтов.

Файл fonts.scale, по-видимому, задает список масштабируемых шрифтов и необходим некоторым приложениям для корректной работы с такими шрифтами. В большинстве случаев это либо точная копия файла fonts.dir, либо просто ссылка на fonts.dir. Естественно, что в каталогах с растровыми шрифтами мы такого файла не найдем.

Файл fonts.alias — это еще один конфигурационный файл, оказывающий влияние на работу со шрифтами. Уже по названию ("alias" — прозвище, кличка) можно догадаться о его назначении. Строки этого файла имеют следующий вид:

```
синоним XLFD_имя_реального_шрифта
```

При этом каждая строка должна оканчиваться только символом конца строки и владельцем файла должен быть суперпользователь. Вот для примера первые строки из файла /usr/X11R6/lib/X11/fonts/misc/fonts.alias в системе Redhat:

```
fixed -misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
variable -*-helvetica-bold-r-normal-*-*-120-*-*-*-*-iso8859-1
```

Слово fixed здесь является синонимом или ссылкой ("alias"). Каждый раз, когда запрашивается шрифт fixed, будет фактически происходить обращение к шрифту, указанному во второй колонке. Шрифт кажется маловат? Просто поменяйте имя, на которое дана ссылка этим определением. Тот же самый принцип применим ко всем шрифтам, включая TrueType. Более того, если у вас не установлены шрифты TrueType, вы можете использовать этот же трюк для того, чтобы использовать какой-то из шрифтов Type 1 вместо запрашиваемых приложением шрифтов TrueType.

Маленькое ПРЕДОСТЕРЕЖЕНИЕ

Для тех, кто имеет привычку "сильно быстро делать": некоторые синонимы должны быть известны системе в любой момент времени! В первую очередь это относится к синонимам "cursor", "fixed" и "variable" в каталогах /misc. Если таких строк в misc/fonts.alias нет, или они указывают на несуществующий шрифт, то графическая оболочка просто откажется запускаться.

Файл fonts.alias важен для некоторых приложений, которые не могут нормально обрабатывать данные, предоставляемые файлом fonts.scale. Самый характерный пример — навигатор Netscape. Без fonts.alias вы можете столкнуться с тем, что навигатор Netscape будет отображать только шрифты с размером символов от 0 до 12 точек. Создав корректный файл fonts.alias в каталоге со шрифтами TrueType, вы получите возможность выбирать из большего числа вариантов шрифтов. Приведем небольшой пример. Предположим, что в файле fonts.scale имеются строки:

```
arial.ttf -monotype-Arial-medium-r-normal--0-0-0-0-p-0-ascii-0
arial.ttf -monotype-Arial-medium-r-normal--0-0-0-0-p-0-fcd8859-15
arial.ttf -monotype-Arial-medium-r-normal--0-0-0-0-p-0-iso8859-15
arial.ttf -monotype-Arial-medium-r-normal--0-0-0-0-p-0-iso8859-1
```

Это масштабируемые шрифты, так что в их именах не указаны размеры. Поэтому в файле fonts.alias должны, соответственно, присутствовать строки

(в файле они должны быть записаны без переносов, просто в книге строка целиком не умещается в рамках страницы):

```
-monotype-Arial-medium-r-normal--6-60-0-0-p-0-iso8859-1 -monotype-Arial-
medium-r-normal--9-90-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--7-70-0-0-p-0-iso8859-1 -monotype-Arial-
medium-r-normal--9-90-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--8-80-0-0-p-0-iso8859-1 -monotype-Arial-
medium-r-normal--10-100-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--9-90-0-0-p-0-iso8859-1 -monotype-Arial-
medium-r-normal--11-110-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--10-100-0-0-p-0-iso8859-1 -monotype-
Arial-medium-r-normal--12-120-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--11-110-0-0-p-0-iso8859-1 -monotype-
Arial-medium-r-normal--12-120-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--12-120-0-0-p-0-iso8859-1 -monotype-
Arial-medium-r-normal--12-120-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--13-130-0-0-p-0-iso8859-1 -monotype-
Arial-medium-r-normal--13-130-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--14-140-0-0-p-0-iso8859-1 -monotype-
Arial-medium-r-normal--14-140-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--15-150-0-0-p-0-iso8859-1 -monotype-
Arial-medium-r-normal--15-150-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--18-180-0-0-p-0-iso8859-1 -monotype-
Arial-medium-r-normal--18-180-75-75-p-0-iso8859-1
-monotype-Arial-medium-r-normal--24-240-0-0-p-0-iso8859-1 -monotype-
Arial-medium-r-normal--24-240-75-75-p-0-iso8859-1
```

После этого навигатор Netscape Navigator будет корректно масштабировать шрифт Arial.

Обратите внимание на различие в размерах шрифта в правой и левой колонках. Например, в первой строке слева указан размер в 6 точек, а справа — 9 точек. С помощью этого приема удается преодолеть "склонность" навигатора Netscape к использованию слишком маленьких шрифтов. Просто подберите справа цифры в соответствии с вашими вкусами.

Создавать файлы fonts.dir, fonts.scale и fonts.alias вручную — занятие не из простых. Поэтому разработаны специальные программы, которые запускаются в каталоге со шрифтами и создают эти файлы. Файл fonts.dir в каталоге с растровыми шрифтами можно создать с помощью команды mkfontdir. Для создания файлов fonts.dir и fonts.scale в каталогах со шрифтами Type 1 надо воспользоваться утилитой Type 1inst **(ftp://ftp.metalab.unc.edu/pub/Linux/X11/xutils/)**. Это скрипт на языке Perl, который автоматически создает файлы fonts.dir и fonts.scale, необходимые для того, чтобы система X Window могла использовать шрифты. Рекомендации по установке и настройке этого скрипта вы найдете в файле README, который поставляется вместе с пакетом.

Для шрифтов TrueType необходима своя утилита ttmkfdir, которую можно найти на многих сайтах с программным обеспечением для Linux. В Red Hat эта утилита включена в состав rpm-пакета FreeType.

На странице "Some Linux for Beginners" (**http://home.c2i.net/dark/linux.html**) вы можете найти скрипт на языке Python, с помощью которого можно создать файл fonts.alias.

Удаление ненужных шрифтов

Когда я просмотрел с помощью программы xfontsel, какие шрифты установлены в системе, я с удивлением обнаружил, что по умолчанию устанавливаются шрифты с какими-то иероглифами (поскольку я не знаю ни китайского, ни японского, то не могу точно сказать, какому языку эти иероглифы соответствуют). Естественно, что появилось желание удалить эти ненужные шрифты, хотя бы для того, чтобы не занимать зря драгоценное место на диске. Теперь, когда мы знаем, где эти шрифты расположены (см. перечень каталогов шрифтов), а также как шрифты именуются, удалить ненужные довольно просто.

Вначале давайте удалим шрифты с иероглифами (если вы не возражаете!). При работе с программой xfontsel я обнаружил, что иероглифы появляются на экране программы xfontsel тогда, когда в поле registry стоит комбинация `jisx` с какими-то еще цифрами. Перейдя в каталог /usr/X11R6/lib/X11/fonts/misc я просмотрел файл fonts.dir, устанавливающий связь между именами шрифтов и именами файлов, в которых хранятся соответствующие шрифты. С помощью этого файла нужно найти имена файлов с иероглифами и удалить их. После этого надо запустить команду `mkfontdir`, которая подкорректирует файл fonts.dir. Можно, конечно, и просто вручную удалить из fonts.dir строки, соответствующие удаленным файлам. Только не забудьте, что первая строка файла fonts.dir должна указывать число разных вариантов шрифта (не файлов в каталоге, а уменьшенное на единицу число строк в файле fonts.dir).

Кроме шрифтов с иероглифами, которые являются просто самым характерным примером, имеется еще множество шрифтов, которые вы, скорее всего, никогда не будете использовать. Но здесь я не буду давать советов, экспериментируйте, если не боитесь.

11.4.6. Подключение новых шрифтов

Источники шрифтов

Итак, вы удалили ненужные вам шрифты. Теперь предположим, что вы хотите добавить в набор шрифтов вашей системы какие-то новые шрифты (скорее всего кириллические). Вначале возникает вопрос, где их взять.

Очень часто большие коллекции шрифтов поставляются вместе с некоторыми графическими, издательскими или офисными программами. Примером может служить Microsoft Office или CorelDRAW, в состав поставки которого входит громадный набор шрифтов. Если пакет русифицирован, то в этом наборе шрифтов найдутся и кириллические шрифты.

В Интернете тоже существует громадный выбор бесплатных или условно-бесплатных шрифтов, однако не многие из них являются кириллическими. Адреса наиболее крупных и полезных сайтов с кириллическими шрифтами перечислены ниже.

- **Паратайп** (**http://www.paratype.com/**) – сайт отечественной компании, занимающейся созданием кириллических шрифтов. На данном сайте можно заказать как шрифты, так и диспетчеры шрифтов и редакторы шрифтов.
- **Веди** (**http://www.vedi.d-s.ru/obzory/f_art/fart1.htm**) -- сайт независимого центра по разработке и распространению кириллических шрифтов.
- **sunsite.unc.edu** (**ftp://sunsite.unc.edu/pub/Linux/X11/fonts/**) - здесь есть несколько пакетов кириллических шрифтов.
- **Freshmeat** (**http://freshmeat.net/**) — задайте поиск по слову "font" и вы найдете несколько пакетов кириллических шрифтов.
- **www.funet.fi** (**http://www.funet.fi/pub/culture/russian/comp/fonts/**) – архив, предоставляющий довольно неплохой выбор кириллических шрифтов.
- **КиАрхив** (**http://ftp.kiae.su/pub/linux/**) — на КиАрхив тоже проще всего воспользоваться предоставляемой там возможностью поиска. Там имеется, в частности, классический набор шрифтов от Cronyx.
- На странице Д. Болховитянова **CYR-RFX** (**ftp://ftp.inp.nsk.su/pub/BINP/X11/fonts/cyr-rfx/doc/README.ru.html**) вы найдете разработанные им шрифты.

В этот список включены далеко не все сайты, на которых имеются кириллические шрифты.

Кроме того, существует много сайтов со шрифтами для английского и других языков. Конечно, коллекции англоязычных шрифтов гораздо богаче, чем для русского языка. Если вы хотите отыскать какой-то конкретный шрифт для латиницы или просто пополнить свою коллекцию таких шрифтов, начните поиск с одного из следующих сайтов:

- **http://www.007fonts.com/**;
- **http://www.freewarefonts.com/**;
- **http://www.1001freefonts.com/**;
- **http://www.fontfreak.com/**;
- **http://www.freewareconnection.com/fonts.html.**

После того как вы скачали пакет шрифтов, можно приступить к его инсталляции. Процедура инсталляции несколько различается для шрифтов Type 1 и TrueType, поэтому рассмотрим эти два случая отдельно.

Инсталляция растровых шрифтов и шрифтов Type 1

Будем предполагать, что вы можете получить права суперпользователя. В таком случае выполните команду su и проделайте следующее.

Создайте новый каталог и распакуйте в него полученный пакет шрифтов. Кстати, я встречал где-то рекомендацию, что лучше ставить новые шрифты в отдельный каталог, чтобы не нарушить работоспособность ранее установленных шрифтов. Можете последовать этому совету.

Перейдите в новый каталог (если не сделали этого ранее). Если производится инсталляция растровых шрифтов (когда в новом каталоге вы видите файлы с расширением pcf), то выполните в этом каталоге команду

```
[root]# mkfontdir
```

которая создает в каталоге со шрифтами файл fonts.dir.

Если производится инсталляция шрифтов Type 1, то чтобы сделать эти шрифты доступными для X Window, надо воспользоваться утилитой Type linst (**ftp://ftp.metalab.unc.edu/pub/Linux/X11/xutils/**), которая создаст файлы fonts.dir и fonts.scale. После установки утилиты просто перейдите в каталог с новыми шрифтами и запустите Type linst:

```
[root]# cd directory
[root]# Type linst
```

Далее необходимо добавить имя нового каталога со шрифтами к перечню каталогов шрифтов. Если пакет xfs у вас уже запущен, вы можете сделать это путем редактирования конфигурационного файла /etc/X11/fs/config.

Теперь надо заставить шрифт-сервер перечитать перечень каталогов, что можно сделать командой

```
[root]# /etc/rc.d/init.d/xfs restart
```

Ваши новые шрифты должны быть теперь доступны для X.

Если вы не используете шрифт-сервер, то вам необходимо добавить имя каталога, содержащего файлы ваших новых шрифтов, к перечню каталогов шрифтов X-сервера в файле /etc/X11/XF86Config. Это можно сделать в каком-либо текстовом редакторе, а можно с помощью команды

```
[root]# xset fp+ /usr/share/fonts/new
```

(имя каталога будет добавлено в конец списка) или

```
[root]# xset +fp /usr/share/fonts/new
```

(имя каталога будет добавлено в начало списка). После этого надо дать команду

```
[root]# xset fp rehash
```

чтобы X-сервер нашел новые шрифты.

Инсталляция шрифтов TrueType

Мы подробно рассмотрим случай, когда используется дистрибутив, основанный на Red Hat, и фонт-сервер xfs.

Создайте новый каталог и распакуйте в него полученный пакет шрифтов. Для примера будем предполагать, что новые шрифты оказались в каталоге /usr/share/fonts/ttf.

Первым делом надо проверить, что в именах файлов шрифтов не встречаются заглавные буквы и пробелы (это требование xfs). Так что получите права суперпользователя, перейдите в каталог с вновь установленными шрифтами:

```
[root]# cd /usr/share/fonts/ttf
```

и, если в именах файлов встречаются заглавные буквы, преобразуйте все имена в нижний регистр. В [П13.2] *(см. приложение)* приводится небольшой скрипт для автоматического преобразования имен файлов в нижний регистр, однако у меня этот скрипт отказался работать. Но в любом случае преобразовать имена и удалить пробелы из имен файлов можно и вручную.

Далее необходимо создать в каталоге со шрифтами TrueType файлы fonts.scale и fonts.dir. Это, конечно, тоже можно сделать вручную, если вы внимательно прочитали весь предыдущий материал, но я, например, не хотел бы этим заниматься. Тем более, что существует утилита ttmkfdir, которую можно найти на многих сайтах с программным обеспечением для Linux. В Red Hat эта утилита включена в состав rpm-пакета Freetype. Выполните следующие команды:

```
[root]# /usr/sbin/ttmkfdir -o fonts.scale
[root]#  mkfontdir
```

После этого в каталоге с новыми шрифтами TrueType должны появиться файлы fonts.dir и fonts.scale.

К сожалению, команды ttmkfdir и mkfontdir не всегда сообщают об ошибках, поэтому после их запуска необходимо убедиться, что нужные файлы созданы, не пусты и содержат корректно сформированные (то есть соответствующие XLFD, см. выше) наименования шрифтов. Как пишут авторы *(см. [П13.2, П13.13] приложения)*, неприятности могут возникнуть потому, что иногда файлы шрифтов могут не в полной мере соответствовать формату TrueType. Если такое имеет место, то описание этого шрифта, выдаваемое по команде ttmkfdir, будет существенно отличаться от формата описаний

шрифтов, определяемого стандартом XLFD. Возможно, содержащееся в файле описание шрифта некорректно. Файлы шрифтов, вызывающие такой эффект, рекомендуется просто удалить. Поэтому до создания файла fonts.scale надо запустить команду ttmkfdir без параметров. В этом случае вывод идет на экран. Длина выдаваемых строк может отличаться, но их структура должна быть одинаковой и соответствовать стандарту XLFD. Если же какой-то из файлов со шрифтами вызывает появление в выводе чего-то другого, то такой файл лучше удалить. Только после этого можно выполнять команду

```
[root]# ttmkfdir -o fonts.scale
```

Еще одна причина возникновения проблем состоит в том, что ttmkfdir почему-то сортирует имена шрифтов в файле fonts.scale в обратном порядке. Этот факт не вызывает затруднений, если вы используете команду ttmkfdir в указанном выше формате. Но если вы попытаетесь подключить декоративные шрифты, которые часто содержат изображения не для всех возможных символов, то просто дать команду

```
[root]# ttmkfdir -o fonts.scale
```

уже недостаточно. Дело в том, что по умолчанию ttmkfdir допускает отсутствие в шрифте не более 5 символов. Но имеется специальная опция (-m nnn, где nnn — число), которая позволяет увеличить допустимое число отсутствующих изображений. Если запустить ttmkfdir в следующем виде:

```
[root]# ttmkfdir -m 100 -o fonts.scale
```

то созданный в том же каталоге и при тех же файлах шрифтов файл fonts.scale получится гораздо большего объема, т. е. будет содержать больше наименований шрифтов. При этом, как раз из-за обратного порядка перечисления имен, файлы с неполным набором символов окажутся в начале файла fonts.scale. В силу этого приложения могут быть "введены в заблуждение" и "схватить" первый попавшийся (в данном случае — неполный) шрифт. Тогда вместо отсутствующих символов вы увидите просто пробелы. Впрочем, с этой проблемой нетрудно справиться. Просто после создания файла fonts.scale надо изменить порядок строк в нем, для чего после выполнения команды

```
[root]# ttmkfdir -m 100 -o fonts.scale
```

надо сделать следующее:

1. Выполнить команду

   ```
   [root]# tac fonts.scale > fonts.dir
   ```

2. Перенести строку с числом шрифтов из конца полученного таким образом файла fonts.dir в его начало.
3. Убедиться, что файл fonts.dir заканчивается символом конца строки.

Теперь мы имеем корректно сформированный файл fonts.dir! Но список в файле fonts.scale все еще имеет обратный порядок. Однако, поскольку эти два файла (по крайней мере в данном случае) должны быть идентичны, то остается только выполнить команду

```
[root]# cat fonts.dir > fonts.scale
```

или

```
[root]# cp fonts.dir fonts.scale
```

Следующий шаг, вообще говоря, не является обязательным, но позволяет в некоторых случаях повысить читаемость шрифтов на экране. Этот шаг заключается в создании файла fonts.alias в каталоге с TT-шрифтами (или в корректировке существующего файла). О том, как это можно сделать, говорилось выше, в *разд. 11.4.5.* В том же разделе упоминался скрипт, который позволяет (по словам автора интернет-страницы [П13.14]) легко создать файл fonts.alias в каталоге со шрифтами TrueType. Полученный с помощью скрипта файл fonts.alias может оказаться очень большим, особенно если создавали файл fonts.dir (который используется скриптом как основа для создания fonts.alias) с помощью команды `ttmkfdir` с опцией -т 100. Да и без этого в нем окажется масса имен шрифтов, которые вы никогда не будете использовать. Поскольку в типичном случае вполне достаточно только кириллических шрифтов, можно попробовать удалить из fonts.alias все неиспользуемые шрифты с помощью следующей последовательности команд (оставляем только кириллические шрифты):

```
[root]# grep 'iso8859-5"' fonts.alias > newfonts.alias
[root]# grep 'koi8-r"' fonts.alias >> newfonts.alias
[root]# cat newfonts.alias > fonts.alias
```

Возможно, этот же прием стоит применить к файлам fonts.dir и fonts.scale, только предварительно продумав все последствия. Если вы войдете в азарт и захотите провести такие же корректировки не только в каталогах со шрифтами TrueType, но в других каталогах со шрифтами, то, по крайней мере, не забывайте, что нельзя просто удалить шрифты, которые имеют в качестве второго имени (синонима) названия cursor, fixed и variable.

Теперь вы можете добавить новый каталог к перечню каталогов шрифтов xfs. Пользователи дистрибутивов, основанных на Red Hat, могут сделать это с помощью утилиты chkfontpath (она тоже входит в пакет Freetype):

```
[root]# /usr/sbin/chkfontpath --add /usr/share/fonts/ttf
```

Если такой утилиты нет, это можно сделать редактированием конфигурационного файла фонт-сервера xfs, а именно, файла /etc/X11/fs/config.

Совет

Помните, что имена каталогов в перечне НЕ ДОЛЖНЫ иметь слэша (/) в конце!

После этого осталось только перестартовать фонт-сервер xfs. Если вы пользовались утилитой chkfontpath, то она осуществляет рестарт xfs автоматически. Если вы вручную редактировали перечень каталогов со шрифтами, то перестартовать xfs можно командой

```
[root]# /etc/rc.d/init.d/xfs restart
```

После того как вы перезапустили xfs, перезапустите также X-сессию.

На этом все! Вы можете проверить, подключились ли новые шрифты с помощью команды xlsfonts. Например, если среди устанавливаемых шрифтов должны были быть шрифты Arial, можно выполнить команду:

```
[root]# xlsfonts I grep arial
```

(можно также воспользоваться командой `xfontsel`). Если новые шрифты видны через xlsfonts, тогда они доступны и для X Window, и наоборот.

Теперь, когда шрифты TrueType установлены, вы можете попробовать, как они работают, например, в навигаторе Netscape.

1. Запустите навигатор Netscape.
2. Откройте окно **Preferences | Appearence | Fonts** и раскройте выпадающий список **Variable Width Fonts.** Там теперь должны появиться вновь установленные шрифты (я, например, увидел **Verdana (Microsoft),** именно тот единственный ttf-шрифт, который устанавливал). Выберите один из них.
3. Разрешите масштабирование, нажав кнопку **Allow Scaling** рядом со списком **Variable Width Fonts.**
4. Установите опцию Use **my default fonts.**
5. Затем выберите размер 12 в выпадающем списке справа.
6. Щелкните по кнопке **OK.**

Теперь текст в окне навигатора Netscape должен отображаться выбранным вами шрифтом.

11.5. Кириллизация shell и других программ

К сожалению, в Linux нет единой системы работы со шрифтами. Каждую отдельную программу, каждое приложение надо отдельно настраивать для того, чтобы эта программа могла использовать шрифты TrueType, Type 1 или какие-то другие, почему-либо привлекательные для вас. И в каждой программе это может делаться по-своему! Я приведу здесь краткие рекомендации по русификации некоторых наиболее употребительных программ. Эти рекомендации заимствованы из [П13.1] *(см. приложение).* Однако перечень

этот далеко не полон, так что ответы на те вопросы, которые мне не удалось осветить, ищите в источниках, ссылки на которые приведены в *приложении*.

Для полноценной работы с кириллицей в текстовом режиме необходимо, чтобы программы умели интерпретировать значения 8-го бита в коде ASCII (напомним, что первоначально этот код был 7-битным). Вот этого давайте и добьемся.

11.5.1. bash

Хотя для большинства программ вполне достаточно установки `LANG=ru_RU.KOI8-R` чтобы начать распознавать русские буквы, многие программы, основанные на библиотеке readline (например bash) все равно считают символы с кодами больше 128 особыми META-символами (пищит при вводе).

Чтобы "отучить" библиотеку readline от этого, необходимо установить три переменные.

```
set meta-flag on
set convert-meta off
set output-meta on
```

Этого можно добиться разными способами. Поскольку вы являетесь суперпользователем своего компьютера, можно определить переменную `INPUTRC=`, например, создав файл /etc/profile.d/readline.sh следующего содержания:

```
#!/bin/bash
INPUTRC="/etc/inputrc"; export INPUTRC
```

и сделать этот файл исполняемым. Кроме того, прописать в файле /etc/inputrc

```
set meta-flag on
set convert-meta off
set output-meta on
```

После этого библиотека readline (и bash) начнет воспринимать русские буквы.

Еще один вариант: не задавать `INPUTRC=`, а прописать те же значения в файле ~/.inputrc в домашнем каталоге *каждого* пользователя.

См. страницу man readline.

11.5.2. less

Если локализация не настроена (а она обязана быть настроенной), то вывод кириллицы через less можно получить, установив переменную окружения `LESSCHARSET`:

```
export LESSCHARSET=koi8-r
```

Это решение годится для *всех* 8-битных кириллических кодировок.

При правильно *настроенной локализации* указывать LESSCHARSET *не надо.* Более того, в ~/.lesskey надо добавить

```
#env
LESSCHARSET=
```

чтобы программа игнорировала установку LESSCHARSET= другими "глупыми" программами (к примеру, man). После этого надо запустить команду lesskey для получения бинарного файла ~/.less. В противном случае он не будет вызывать setlocale(LC_CTYPE,"") и, как следствие, не будет icase search для русских букв.

11.5.3. man

В последнее время появляется все больше и больше man-статей, переведенных на русский язык, но вот отобразить их не всегда удается. Для исправления этого неудобства следует поправить соответствующие строки в /usr/lib/man.conng, если этот файл есть, или правильно настроить less.

11.5.4. nroff

Для того, чтобы через nroff можно было "пропустить" символы кириллицы, надо использовать его с ключом -Tlatin1. Пропишите где-нибудь в стартовом скрипте (если у вас bash, то в .bashrc).

```
alias nroff='nroff -Tlatin1'
```

Просмотр некоторых файлов в mc запускается через команду nroff, вызов которой осуществляется по расширению имени файла. Поэтому в файле /usr/lib/mc/mc.ext следует в строке вызова nroff изменить параметр вызова C -Tascii на -Tiatini.

11.5.5. ls

Если локализация настроена неправильно, то ls не будет печатать кириллические символы. В этом случае, возможно, поможет одна из следующих команд: ls -N, dir -N ИЛИ ls --show-control-chars.

11.5.6. The Midnight Commander

Чтобы корректно отображался текст кириллицы, необходимо установить флажки в опции **Полный 8-битный вывод и Полный 8-битный ввод** с помощью команды **Настройки | Биты символов.**

11.5.7. Диски Windows 95 и DOS

Чтобы подмонтировать диск в формате файловых систем Windows 95 и DOS с правильной поддержкой русских букв, необходимо воспользоваться командой:

```
[user]$ mount -t vfat -o
umask=002,noexec,gid=500,codepage=866,iocharset=koi8-r /dev/hdb1 /mnt
```

То есть все русские имена на диске FAT сохраняются в Codepage 866! Для работы этой опции ядро (>2.0.36) должно быть пересобрано с поддержкой NLS, кодовыми страницами CP866, NLS KOI8-R и, конечно же, с поддержкой VFAT.

11.5.8. Samba

Чтобы увидеть русские буквы в именах файлов на сетевом диск через пакет Samba, в файл /etc/smb.conf следует добавить строки:

```
[global]
      character set = koi8-r
      client code page = 866
      preserve case = yes
      short preserve case = yes
```

Первые две опции указывают внутреннюю кодировку имен файловой системы (client code page) и внешнюю кодировку пользователя (character set).

Следующие две опции указывают, что надо сохранять регистр длинных и коротких имен файлов.

11.5.9. rlogin

Удостоверьтесь, что shell на месте адресата установлена правильно. Если ваш rlogin не работает как надо по умолчанию, то используйте rlogin -8.

Пропишите в стартовом скрипте (если вы используете bash, то это .bashrc)

```
alias rlogin='rlogin -8'
```

11.5.10. telnet

Если возникают проблемы с вводом русских символов, надо создать файл ~/.telnetrc со следующей строкой:

```
DEFAULT set outbinary
```

Вы можете встретить проблемы при работе в кодировке CP-1251 не передается маленькая русская буква "я" (ASCII-код 0xff). У протокола Telnet

0xff — это первый символ управляющей последовательности. Дабы передать собственно "я", нужно его удваивать: Oxff, Oxff. При использовании KOI8-R такая проблема отсутствует.

11.5.11. Ircll

Добавить в файл конфигурации ~/.ircrc следующие строчки:

```
/set translation russian
/set eight_bit_characters on
```

11.6. Кириллизация печати

До сих пор наше внимание было сосредоточено на выводе текста на экран. Однако когда мы говорим об использовании шрифтов, нельзя совсем уж оставить в стороне вопрос о том, как можно организовать печать на принтере различными шрифтами.

В отличие от других операционных систем (например, Windows и MacOS), Linux, как и другие UNIX-системы, не имеет аппаратно-независимой подсистемы печати. Некоторые приложения сами обеспечивают поддержку печати, но большая часть приложений предполагает, что принтер обладает возможностью интерпретировать язык Adobe PostScript. Но принтеры со встроенными PostScript-интерпретаторами достаточно дороги и довольно редко встречаются в наше время. Для того чтобы можно было использовать другие типы принтеров, была разработана специальная программа, Ghostscript, которая может интерпретировать код PostScript, преобразовывать контурный шрифт в растровый, формировать соответствующие команды для принтера, обеспечивая тем самым печать на большинстве типов принтеров.

Поскольку все версии PostScript и Ghostscript поддерживают шрифты Type 1 и Type 42, а Ghostscript версии 4 и выше поддерживает и шрифты TrueType, то больших трудностей с организацией красивой печати под Linux не возникает. По умолчанию (в Red Hat) шрифты для Ghostscript устанавливаются в каталог /usr/share/fonts/defaults/ghostscript/. Поэтому если вы хотите добавить какие-то шрифты для печати, поместите файлы новых шрифтов в этот каталог или создайте в нем символические ссылки на новые шрифты. Впрочем, можно поместить файлы шрифтов и в другие каталоги, но тогда нужно будет указывать точный путь к шрифтам в файле /usr/share/ghostscript/N.M/Fontmap, который делает шрифты доступными для Ghostscript.

Например, для того, чтобы добавить шрифт Times New Roman и его варианты, скопируйте файлы times.ttf, timesbd.ttf, timesbi.ttf, и timesi.ttf в каталог

/usr/share/fonts/defaults/ghostscript/ и добавьте следующие строки в файл Fontmap:

```
/Times New Roman                (times.ttf);
/Times New Roman Bold           (timesbd.ttf);
/Times New Roman Bold Italic    (timesbi.ttf);
/Times New Roman Italic         (timesi.ttf);
```

Более подробные инструкции вы найдете в документации к программе Ghostscript и в *разд. 9.6.*

Конечно, я рассказал далеко не обо всех моментах, связанных с подключением новых и использованием имеющихся шрифтов для системы X Window. Например, за рамками рассмотрения остались вопросы настройки шрифтов в конкретных приложениях, а также вопросы преобразования шрифтов TrueType в шрифты Type 1. Но все изложить невозможно, поэтому остается только отослать читателя к списку доступной литературы *(см. [П13] приложения).*

Глава 12

Программы для работы с текстом

12.1. Несколько слов о форматах текстовых файлов

Как вы знаете, любой файл, в том числе и файлы, в которых сохранены текстовые документы того или иного вида, представляет собой просто последовательность байтов. Символы текста кодируются разными значениями байта или последовательностей байтов. Информация о том, как именно эти символы должны располагаться на странице, тоже кодируется с помощью неотображаемых управляющих симоволов типа конца строки или символа табуляции. В простейших случаях число управляющих кодов ограничивается 32-мя первыми значениями байта (или кода ASCII), а все остальные значения байта используются для кодирования информационных символов. Именно такие файлы мы и называем ASCII-файлами. Примерами таких файлов могут служить файлы, создаваемые редакторами типа встроенного редактора программы Midnight Commander, файлы, создаваемые программой Notepad в Windows и vi в UNIX.

Со временем появилось желание расширить возможности форматирования текста, а значит потребовалось увеличить число кодов, используемых в качестве управляющих, и в качестве таких кодов стали даже использовать последовательности байтов (символов ASCII). Но существенно то, что эти форматирующие последовательности (почти) не мешают вам прочитать текст, содержащийся в файле, с помощью любого простейшего средства просмотра или простейшего текстового редактора. Примерами таких файлов могут служить файлы, создаваемые редакторами типа Лексикон, файлы в формате HTML.

Третий тип — это файлы, использующие собственный формат для представления текста (в которых символы текста тоже представлены специальными последовательностями). Существеннейшее отличие форматов третьего типа от двух предыдущих заключается в том, что и просматривать и создавать файлы в таких форматах без специальных программ практически невозможно. Например, HTML-файлы можно редактировать с помощью программы Notepad (Блокнот), но невозможно делать то же самое с файлами формата MS Word 97.

Иногда трудно с первого взгляда отнести файл к тому или иному типу. Например, файлы формата PostScript формально относятся ко второму типу, поскольку весь читаемый текст там представлен в кодах ASCII, однако в этих файлах так много форматирующих вставок, что текст можно найти лишь с большим трудом, почти как в файлах третьего типа.

Из вышесказанного следует, что даже для просмотра некоторых типов текстовых файлов (не говоря уж об их редактировании) требуются специальные программные средства. Часто для просмотра файлов пользователь применяет привычный ему текстовый редактор. Но встречаются ситуации, когда информация представлена в незнакомом для этого редактора формате. Самая большая проблема приверженцев Linux — форматы, используемые в продуктах Microsoft. Пока большинство пользователей ПК создают тексты в MS Word, приходится либо изыскивать текстовый редактор, который понимает форматы Word, либо находить программы-переводчики из формата Word в один из открытых стандартных форматов. Впрочем, даже если информация представлена в "простом" коде ASCII, вы, просматривая какой-либо файл, можете столкнуться с "нечитаемым" текстом из-за различия используемых кодировок русского языка. Поэтому сначала давайте рассмотрим вопрос о том, как прочитать (или просмотреть) файлы различных форматов или в различных кодировках.

12.2. Программы для просмотра текстов в разных форматах

Где-то я читал, что в традициях UNIX было создавать отдельную команду для каждого элементарного действия. Это наблюдение хорошо иллюстрируется наличием в Linux целой совокупности отдельных программ для просмотра файлов. Конечно, если вы работаете в каком-либо файловом менеджере, типа Midnight Commander или Konqueror, то вы будете использовать встроенные *в* них средства просмотра файлов. Но в некоторых случаях может оказаться полезным и знание того, как просмотреть содержимое того или иного файла, работая просто в терминале.

12.2.1. Традиционные средства UNIX для просмотра текстовых файлов

Самым простым средством просмотра файла является, наверное, команда cat. Выведя содержимое текущего каталога с помощью команды `ls`, вы можете также вывести на экран содержимое любого из имеющихся файлов командой cat. Правда возникает одно неудобство: если файл большой, то в результате на экране остаются только последние его строки, все остальное "убегает вверх". Можно, конечно, пролистать несколько экранов с помощью

комбинации клавиш <Shift>+<PgUp>, но возможность эта тоже ограничена (некоторой величиной, задаваемой в окружении, по умолчанию — величиной в 1000 строк).

Для организации постраничного вывода существует команда-фильтр more. Ее можно применять в двух эквивалентных вариантах:

```
[user]$ cat file.txt | more
```

или

```
[user]$ more file.txt
```

Команда less представляет собой улучшенный и доработанный вариант команды more, который рекомендуется использовать вместо more во всех случаях. Имейте в виду, что команда less используется для вывода на экран страниц интерактивной подсказки man.

Для просмотра только нескольких последних строк текстового файла существует специальная команда tail, которой в качестве параметра можно указать количество выводимых строк. Можно предположить, что наличие такой команды было очень полезно в те времена, когда текстовые файлы создавались с помощью программы cat, путем прямого ввода с терминала, таким вот примерно образом:

```
[user]$ cat >> file.txt
```

В этом случае после любого перерыва в работе просто необходимо было просмотреть последние из введенных строк, чтобы вспомнить, на чем была остановлена работа.

Однако те времена давно уже прошли. Для создания текстов стали использовать текстовые редакторы, а для сохранения — не простые ASCII-файлы, а специальные, достаточно изощренные, форматы, позволяющие сохранить не только сам текст, но и информацию об абзацах, страницах, стилях, шрифтах и много что еще. И для просмотра таких файлов стали требоваться специальные программы (на сленге компьютерщиков — вьюеры), которые не выводят на экран все специальные символы форматирования, а преобразуют их в соответствующие отступы, выступы, пробелы, шрифты и т. д. Рассмотрим такие просмотрщики для двух распространенных форматов файлов.

12.2.2. Программа Acrobat Reader (версия 4.05)

Программа Acrobat Reader предназначена для просмотра файлов формата Portable Document Format (PDF), который широко распространен в компьютерном мире и используется в разных операционных системах и на разных платформах. В этот формат легко преобразуются документы формата PostScript. Файлы формата PDF очень часто встречаются в Сети. Необязательно иметь возможность создавать такие файлы (программы их создания

распространяются на коммерческой основе), но уметь их читать надо. Как раз для этого и служит программа Acrobat Reader фирмы Adobe Systems Inc. (рис. 12.1), распространяемая бесплатно как в версии для Windows, так и в версиях для UNIX, и в частности для Linux. В настоящее время версия 4.05 этой программы для Linux может быть бесплатно получена на сайте фирмы **http://www.adobe.com** (для Windows уже выложена версия 5.0).

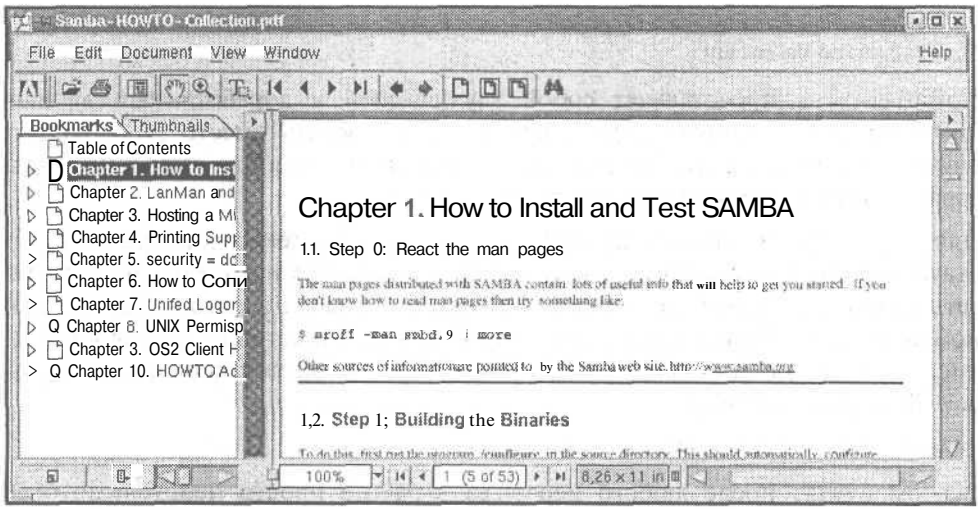


Рис. 12.1. Окно программы Acrobat Reader

Для установки программы Acrobat Reader 4.05 вам потребуется 12 Мбайт дискового пространства, а для работы с программой — 32 Мбайт ОЗУ. Если у вас была установлена версия 3.0 этой программы, ее необходимо предварительно удалить.

Прежде чем начать установку, выберите один из двух возможных вариантов установки: сетевой или локальный (на отдельном компьютере).

Локальная инсталляция гарантирует вам доступ к программе на данном компьютере и может обеспечить значительно большую производительность (особенно в сравнении с работой по загруженной сети). Для локальной инсталляции вы, естественно, должны иметь право записи в тот каталог, куда будет устанавливаться программа.

Сетевая инсталляция облегчает решение задач администрирования и поддержки программы, особенно в тех случаях, когда программу используют многие пользователи. Очевидно, что легче осуществлять обновление одного экземпляра программы на сервере, чем обновлять множество копий на отдельных компьютерах. Сетевая инсталляция должна выполняться администратором сервера.

Однако сетевая инсталляция имеет и некоторые недостатки. Сервер должен быть высоконадежным и инсталлируемые файлы должны быть установлены в каталог, который должен монтироваться одинаковым образом (в одну и ту же точку монтирования) на всех рабочих станциях сети.

По умолчанию программа Acrobat Reader устанавливается в каталог /usr/local/Acrobat4 (проверьте с помощью команды df что на соответствующем разделе диска достаточно свободного места).

Если вы получили файл с программой Acrobat Reader с Web-сайта, требуется вначале разархивировать установочные файлы с помощью программ gunzip и tar. После этого вам необходимо перейти в каталог, содержащий установочный скрипт и запустить его:

```
[root]# ./INSTALL
```

Инсталляционный скрипт выводит на экран несколько вопросов, в частности о том, в какой каталог разместить файлы программы. На каждый вопрос предлагается вариант ответа, который можно принять, просто нажав клавишу <Enter>. Если предлагаемый по умолчанию ответ вас не устраивает, введите собственный вариант ответа. В остальном надо просто следовать инструкциям, выдаваемым программой установки. Если вы где-то ошиблись, можно прервать установку комбинацией клавиш <Ctrl>+<C> и начать все заново.

В конце концов, появляется сообщение о том, что установка успешно завершена (естественно, по-английски, что-то вроде "installed successfully").

После установки в том каталоге, куда вы установили программу (напомню, что по умолчанию это /usr/local/Acrobat4), создаются несколько подкаталогов:

- каталог bin содержит скрипт для запуска программы;
- каталог Reader содержит файлы подсказки и некоторые платформозависимые файлы (смотри примечание ниже);
- каталог Fonts содержит шрифты;
- каталог Browsers содержит скрипт для запуска Acrobat Reader из браузера Netscape.

Примечание

Если вы выбрали сетевой вариант инсталляции, то в каталоге Reader будут установлены несколько вариантов бинарных файлов, и скрипт запуска программы будет выбирать тот вариант бинарного файла, который нужен для рабочей станции, с которой производится вызов программы!

Наконец, в завершение инсталляции, вы можете сделать так, чтобы не было необходимости каждый раз для запуска программы Acrobat Reader сначала переходить в каталог, где расположен скрипт запуска. Для этого надо соз-

дать ссылку на этот скрипт в одном из каталогов, указанных в переменной $PATH. Для примера создадим такую ссылку в каталоге /usr/bin:

```
[root]# ln -s /usr/locale/Acrobat4/bin/acroread /usr/bin/acroread
```

Для того чтобы было удобно запускать программу в графической среде, остается только создать значок ("иконку") на рабочем столе (если вы не помните, как это сделать, смотрите описание используемой вами графической среды) и на этом процесс инсталляции программы можно считать завершенным.

Теперь вы можете запустить Acrobat Reader (естественно, для этого должен быть запущен графический режим), используя команду acroread или щелчком по иконке на рабочем столе.

Если вы не создали ссылку, то надо указывать в командной строке полный путь к скрипту запуска:

```
[user]$ /usr/local/Acrobat4/bin/acroread
```

Для того чтобы просмотреть с помощью Acrobat Reader какой-то конкретный файл или даже несколько, можно сразу указать имена этих файлов в командной строке. Например,

```
[user]$ acroread /user/share/docs/Samba/Collection.pdf
```

Если вы введете команду

```
acroread <filename>
```

когда Acrobat Reader уже запущена, активная копия программы отобразит заданный файл.

Если вы хотите получить подсказку по использованию программы, введите одну из команд

```
[user]$ acroread -help
[user]$ acroread -helpall
```

С помощью программы Acrobat Reader вы можете преобразовать файлы формата PDF в файлы формата PostScript уровней 1 или 2 при условии, что вы делаете это в командной строке графического режима. Для этого надо выполнить примерно следующую команду:

```
[user]$ cat sample.pdf | acroread -toPostScript > sample.ps
```

С помощью аналогичных команд (более подробное описание которых вы сможете найти в документации к программе) можно преобразовывать сразу группу файлов.

Для обратного преобразования PDF-файла в файл формата PostScript уровня 2 надо набрать следующую команду:

```
[user]$ acroread -toPostScript -level2 pdf_file_1
```

Я не буду подробно описывать все меню и возможности программы Acrobat Reader, поскольку она широко распространена и многим читателям знако-

ма. Лучше основимся подробнее на специфических средствах просмотра файлов Linux.

12.2.3. Программа gv

Программа gv (или ghostview) разработана Иоганнесом Плассом (Johannes Plass) и предназначена для просмотра файлов форматов PostScript и PDF (рис. 12.2).

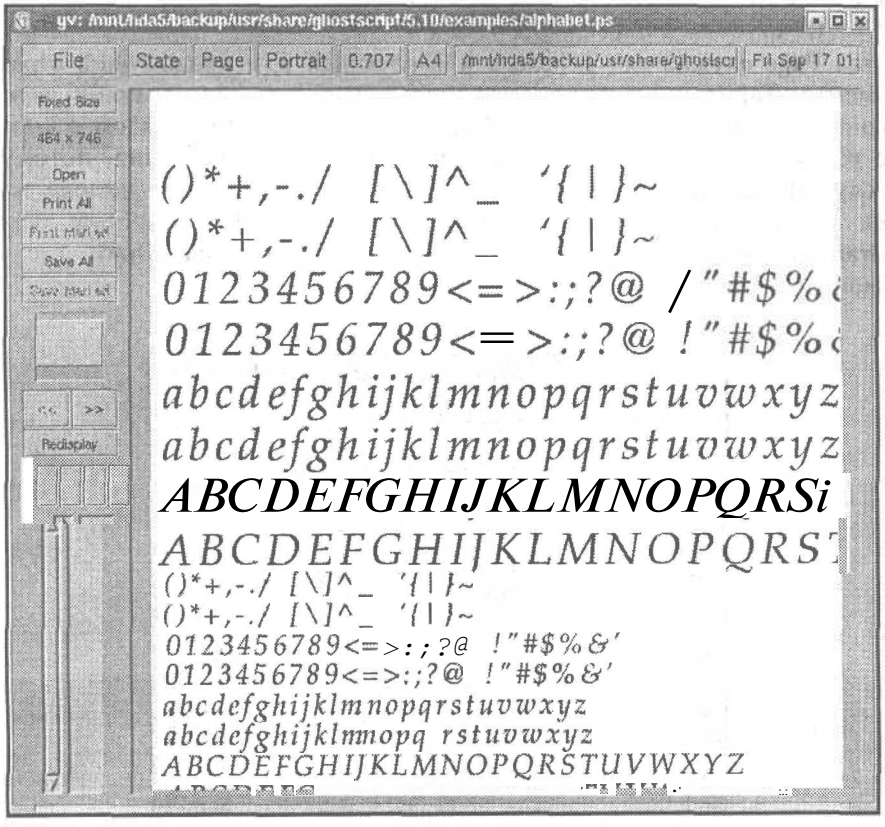


Рис. 12.2. Окно программы gv

После ее запуска без указания имени файла основное окно программы будет пустым. Чтобы открыть какой-то файл, надо щелкнуть по кнопке **Open** в левой колонке. Появится окно выбора файла (рис. 12.3), с помощью которого можно перемещаться по структуре каталогов и выбрать нужный файл. После этого имя файла появится в специальном поле в верхней рамке основного окна программы. Там же отображается несколько кнопок управле-

ния программой, которые образуют своеобразное меню. Только чтобы пользоваться этим меню, по кнопкам надо не просто щелкнуть, а удерживать левую кнопку мыши нажатой.

С помощью кнопки **File** вы получаете доступ к командам **Open** (то же самое, что и упомянутая выше кнопка **Open** в левой вертикальной колонке), **Reopen** (Перечитать), **Print document** (Отпечатать документ), **Save document** (Сохранить документ).

Кнопка **State** (Состояние) позволяет включить или изменить некоторые параметры программы, в частности, получить доступ к отдельному окну настроек программы, изображенному на рис. 12.4.

Кнопка **Page** служит для перехода к следующей или предыдущей странице документа. Такие же переходы можно осуществить при помощи двух небольших кнопок << и >> в левой вертикальной колонке. Рядом с кнопкой **Page** расположена кнопка, не имеющая постоянного названия, потому что она служит для задания одной из 4 возможных ориентации вывода текста. Текст можно выводить на странцу, ориентированную обычным образом **(Portrait),** с поворотом на 90° **(Landscape),** на 180° **(Upside-Down,** т. е. вниз головой), на 270° **(Seascape).**

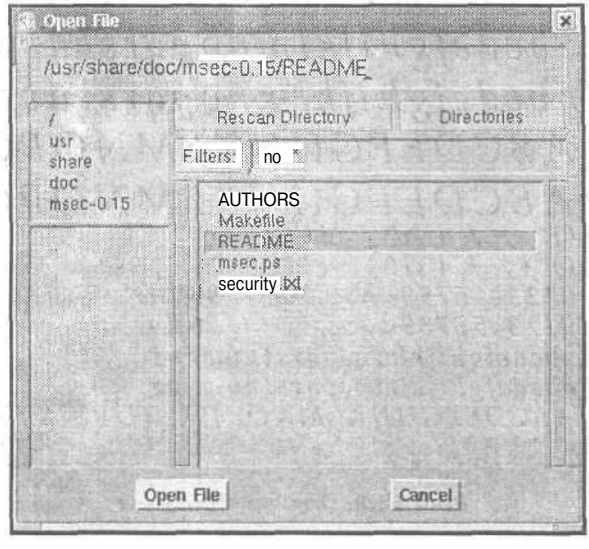


Рис. 12.3. Окно выбора файла для просмотра в gv

Следующая кнопка служит для задания масштаба изображения, который и указывается на этой кнопке. Еще одна кнопка служит для выбора формата бумаги. Это нужно для организации печати, ибо программа gv и есть средство для предварительного просмотра того, что получится на бумаге.

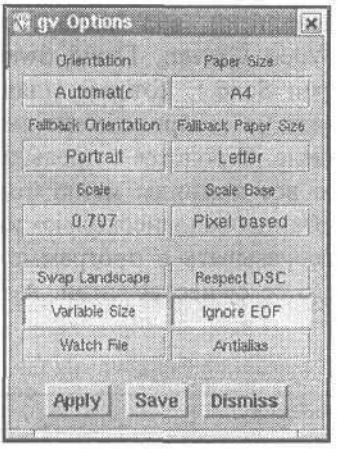


Рис. 12.4. Окно настроек программы gv

В левом столбце имеется еще одна интересная вещь: серый прямоугольник над кнопками >> и <<. Внутри этого прямоугольника имеется еще один прямоугольник, который можно захватить мышкой и подвигать в пределах внешнего прямоугольника. Проделайте это и вы поймете назначение этого элемента: выводить в окно прсмотра различные части страницы.

12.2.4. Программы просмотра файлов PS, PDF и DVI из KDE

Пользовательский интерфейс программы gv несколько непривычен для человека, работавшего только с Windows. Между тем, в составе интегрированной графической среды KDE имеются две программы с более традиционным обликом. Это "Просмотрщик PS/PDF" и "Просмотрщик DVI" (так эти программы именуются в заголовках собственных окон и в меню KDE). Интерфейс у них очень похож и объясняется это тем, что они работают через одну оболочку — KviewShell. Однако в меню KDE они значатся как отдельные программы, и в заголовке окна каждая из них выводит собственное название.

Замечание

Здесь можно попутно отметить, что в UNIX очень часто применяется такой прием: создается "рабочая лошадка" (back-end), к которой затем пишут красивые оболочки (front-end) на разные случаи, например, для текстового и графического режимов или для разных вариантов библиотек. С примерами такого подхода вы еще не раз столкнетесь даже при чтении этой книги, хотя я не буду отдельно выделять эти моменты.

На рис. 12.5 приведен внешний вид окна программы "Просмотрщик PS/PDF" (разработчики Wilco Greven, David Sweet, Mark Donohoe, David Faure, Daniel Duley и Espen Sand.), которая иначе называется Kghostview, что, по-видимому, указывает на ее происхождение от программы, рассмотренной в предыдущем разделе. О родстве с предьщущей профаммой говорит также то, что практически все команды меню повторяют аналогичные команды gv, только меню организовано несколько иначе. Это видно и из рисунка. Более подробно рассказывать о профамме не имеет смысла, достаточно того, что вы знаете о ее существовании. А пользоваться ею вы легко научитесь самостоятельно.

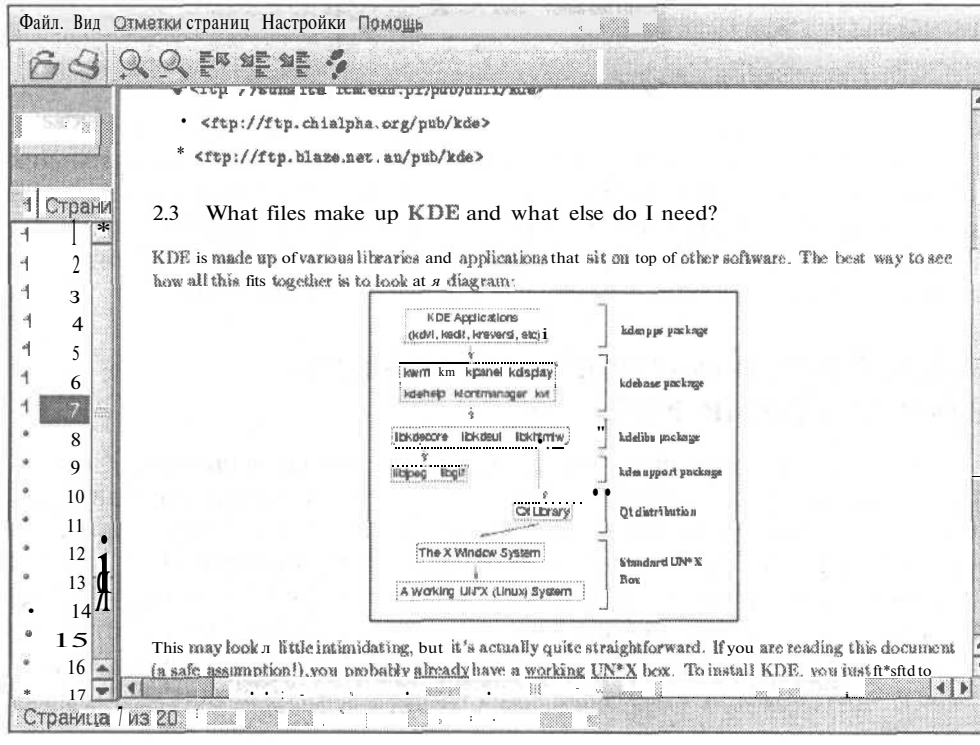


Рис. 12.5. Внешний вид окна программы "Просмотрщик PS/PDF"

О программе "Просмотрщик DVI" я тоже не буду рассказывать, ограничившись уже приведенным упоминанием о ней и сообщением о том, что она предназначена для просмотра файлов формата DVI, создаваемых системой верстки TEX.

12.2.5. Пакет WordViewer

Как уже говорилось выше, одна из самых больших проблем для пользователей Linux — это работа с файлами в форматах MS Word (и других программ из пакета MS Office). Ведь пока что большинство текстов создается именно в этом формате. Наиболее распространенное средство просмотра таких файлов, — это программа (точнее библиотека программ) wv, которая раньше носила более полное название mswordview.

Библиотека wv предназначена для получения доступа к файлам форматов MS Word 6/95/97/2000 из операционных систем типа UNIX, в частности из Linux. В состав дистрибутива ALT Linux Junior 1.0 включена версия 0.6.5 этого пакета, авторами которого являются Dom Lachowicz и Caolan McNamara (первый разработчик). Если у вас этот пакет не установлен, вы можете найти его на сайте **http://www.wvware.com.**

Идея, реализованная в этом пакете, очень проста: раз мы не имеем средств для просмотра файлов в формате MS Word, то давайте преобразуем текст из этих файлов в какой-то открытый формат. В качестве последнего можно выбрать один из следующих форматов: HTML, PS, PDF, LaTex, DVI (формат издательской системы TEX), ABW (формат текстового редактора AbiWord), Wml (формат, используемый в персональных органайзерах PDA и устройствах типа Web-телефонов), ASCII-текст. Вызов отдельных библиотечных процедур может быть использован в других приложениях. Разработчики обещают, что вскоре станет возможным и обратное преобразование: из перечисленных открытых форматов — в формат MS Word.

Пакет состоит из отдельных программ, каждая из которых предназначена для преобразования doc-файла в определенный формат. Этот формат указывается непосредственно в названиях отдельных программ пакета: wvAbw, wvCleanLatex, wvDVI, wvHtml, wvLatex, wvMime, wvPDF, wvPS, wvRTF, wvSimpleCLX, wvText, wvWml. Кроме того, в состав пакета входят две вспомогательных утилиты: wvVersion, которая служит для получения информации о версии документа MS Word, и wvSummary — эта утилита выводит общую информацию о документе, которую в самом MS Word можно ввести через команду меню **Файл | Свойства.** Вот как выглядит вывод этих команд (для использованного мной в этом примере файла служебные данные не были введены; впрочем, если их ввести по-русски, то радости будет не много больше, поскольку вывод получим в кодовой странице CP-1251, так что текст на экране будет нечитаемым).

```
[user@linux tmp]$ wvVersion book-p1.doc
Version: word8, Encrypted: No
[user@linux tmp]$ wvSummary book-p1.doc
The title is B
The subject is
```

```
The author is kos
The keywords are
no comments found
The template was Normal.dot
The last author was kos
The rev # was 12
The app name was Microsoft Word 8.0
PageCount is 1
WordCount is 52757
CharCount is 300716
Security is 0
Codepage is 0x4e3 (1251)
```

Основной утилитой пакета является программа wvWare (или wvConvert), вызов которой осуществляется следующим образом

```
wvWare [OPTION...] filename.doc > filename.html
```

Основные опции:

- -x --config=config.xml

  Указывает на используемый выходной фильтр.

- -c --charset=charset

  Задает кодовую страницу для iconv.

- -p --password=password

  Задает пароль для зашифрованных документов Word.

- -d --dir=dir

  Задает каталог, в котором будет сохранена создаваемая графика.

- -v -version

  Выдает версию пакета wvWare.

- -? -help

  Выводит краткую справку по использованию программы.

Выходной фильтр, указываемый опцией -x или --config= , задает формат выходного файла. Если эта опция не задана, то выходной фильтр ищется в текущем каталоге или (если в текущем не нашли) по месту установки пакета. По умолчанию используется фильтр wvHtml.xml, т. е. doc-файл преобразуется в формат HTML.

После преобразования полученный файл можно просмотреть с помощью соответствующей программы просмотра, например, HTML-файл — с помощью любого Web-браузера. Конечно, при этом некоторые возможности

форматирования, имеющиеся в Word, теряются, и это надо иметь в виду, просматривая полученные файлы.

Насколько я могу судить, именно программы пакета wv используются для открытия файлов MS Word в некоторых текстовых редакторах для Linux, например, в AbiWord.

12.2.6. Программы-перекодировщики кодовых страниц

Как известно, для представления символов русского алфавита существует несколько альтернативных кодировок или кодовых страниц.

В Unix-системах наиболее распространенной является кодировка KOI8-R, Microsoft использует CP-1251 или CP-866 (DOS). Существуют еще ISO8859-5, UTF, UNICODE (подробнее о кодировках можно прочитать в *разд. 9.3).*

Если вы встретили файл, содержимое которого "не читается", то вам потребуется программа перекодировки. Как говорится в RU.LINUX.FAQ: "Перекодировщиков CP-1251 и CP-866 → KOI8-R просто огромное количество. Не надо писать новых ;-". Наиболее широко распространены iconv (входит в поставку glibc) и GNU recode."

Программа iconv запускается следующим образом:

```
[user]$ iconv -f866 -tKOI8-R -o <outfile> infile
```

Если не указать выходной файл (опция -o), то результат будет выдаваться на экран (используя фильтры more или less, можно удобно просмотреть файл). Чтобы получить список всех возможных кодировок (а он огромен!), дайте команду iconv `--list`, а для получения помощи: iconv -? или iconv `--usage`. Между прочим, man-страница не выдается. Впрочем, других опций все равно очень немного: только `--verbose` (сообщать дополнительные сведения), -v и `--version` (обе опции служат для вывода версии программы).

Программу recode можно найти на **http://www.iro.umontreal.ca/~pinard/recode/**.

Запускается она примерно так:

```
[user]$ recode CP1251..KOI8-R winfile.txt
```

Кроме упомянутых команд вы можете воспользоваться программой Russian Anywhere, которая существует как в версии для Windows (где я с ней и познакомился), так и в версии для командной строки Linux (создатели обещают выпустить и графическую оболочку). Эту программу можно скачать (как в исходных кодах, так и в виде исполняемого модуля) с сайта разработчика **http://www.livotov.org/software/**.

Исполняемый модуль программы имеет название re. Его лучше поместить в один из каталогов, указанных в переменной PATH.

Программа вызывается из командной строки. Для того чтобы перекодировать какой-то файл, который "не читается", в кодировку KOI8-R, надо дать команду:

```
[user]$ re <SourceFile> <DestFile> ? K
```

где:

- <SourceFile> — исходный (нечитаемый) файл;
- <DestFile> — перекодированный файл;
- ? — сообщает re, что кодировка исходного файла не известна и re должна проанализировать файл и самостоятельно определить его кодировку;
- к — задает кодировку для результирующего файла (в данном случае KOI-8).

Если вы знаете кодировку исходного файла, вы можете указать ее вместо символа "?". Например, если вы хотите перекодировать файл letter.txt, который был создан в Windows, и вы знаете, что файл сохранен в кодировке CP-1251, то надо дать команду:

```
[user]$ re letter.txt letter-koi.txt W K
```

После этого, просмотрев файл letter-koi.txt, вы увидите вполне читаемый русский текст в KOI8-R.

Полный формат вызова перекодировщика:

```
[user]$ re options filename_from filename_to cp_from cp_to [s/d/f]
                                                          [u/l/s]
```

ГДе options: [-v] [-E|-R|-N][-e|-s]

- -v — выдавать информацию о ходе обработки;
- -п — не выдавать информацию о ходе обработки (задано по умолчанию);
- -E — преобразовывать все символы р, Н из русских в английские;
- -R — преобразовывать все символы р, Н из английских в русские;
- -N — оставлять все р, Н (русские и английские) как в исходном тексте (задано по умолчанию);
- -е — перекодировать все символы 0x80—0xFF;
- -s — перекодировать только 64 символа русского алфавита (задано по умолчанию),

а cp_from и cp_to — любой из следующих символов, обозначающих возможные кодировки (по умолчанию — W, K).

Символы, которыми обозначаются кодировке в программе re, приведены в табл. 12.1.

Таблица 12.1. *Обозначение кодировок в программе re*

Символ	Кодировка	Символ	Кодировка
W	Windows	_ (подчеркивание)	_xxe
D	Dos	%	%hex
K	KOI-8	\\	\'hex
L	Latin	G	Graph_win
I	Iso	<	binhex
H	HEX	+	+UTF7-
S	ShiftKbrd	C	C_MIC
M	Mac	Y	Y_c16
A	AFF	Z	Z_c32
O	Odd(UTF8_1)	F	F(UTF8_2)
B	Base64	P	Pict
E	Express	N	N_Estl
T	T-Html	V	V_Vpp855
U	User	X	X_sp
- (тире)	uue	J	J_diff

Как уже было сказано, если `cp-from="?"`, то программа пытается самостоятельно определить кодировку исходного файла.

Если у вас по каким-либо причинам не оказалось ни одной из указанных программ-перекодировщиков, то для просмотра содержимого файла можно воспользоваться одним из браузеров Интернета, которые изначально ориентированы на работу с разными кодировками. Например, сгодится обычный lynx:

```
[user]$ lynx -assume_local_charset cp866 file.txt
```

Можно также загрузить "нечитаемый" файл в Netscape Navigator, после чего поменять кодировку командой **View | Character Set.**

12.3. Проверка правописания

Прежде чем перейти к рассмотрению текстовых редакторов под Linux, необходимо кратко рассмотреть программу проверки правописания ispell. Дело в том, что проверка правописания — это одна из функций, которую должен иметь современный текстовый редактор, и многие из них подключают для

выполнения этой функции именно ispell (или aspell). Существует русифицированный вариант этой программы, которой был разработан Владимиром Рогановым и Константином Книжником.

Установка ispell состоит из двух этапов: вначале надо установить саму программу, а затем установить словарь русского языка. Для установки самой программы ispell я воспользовался пакетом ispell-3.1.20-23.i386.rpm, а для его русификации — пакетом ispell-russian-3.1.20-23.i386.rpm. Оба пакета входили в состав дистрибутива Black Cat Linux 6.02. Для установки первого пакета достаточно дать команду

```
[root]# rpm -i ispell-3.1.20-23.i386.rpm
```

а для второго — команду

```
[root]# rpm -i ispell-russian-3.1.20-23.i386.rpm.
```

После этого в каталоге /usr/lib/ispell появятся файлы русского словаря russian.aff и russian.hash (остальные словари, например, немецкий), можно удалить, если вы не собираетесь производить проверку правописания на этих языках).

Для проверки текста теперь достаточно дать команду следующего вида:

```
[user]$ ispell -drussian edit.htm
```

Естественно, имя файла edit.htm здесь взято для примера; вы должны подставить имя того файла, который вы хотите проверить, причем файл должен находиться в текущем каталоге, иначе надо дать полное имя файла с указанием пути.

Принцип работы программы ispell очень прост: каждое встречающееся в файле слово должно иметься в словаре программы. Если слово в словаре не найдено, считается, что найдена ошибка, и на экран выводится сообщение, пример которого можно увидеть на рис. 12.6. В самой верхней строке выведено обнаруженное ошибочное слово и имя проверяемого файла. Ниже выведено несколько строк (число можно задать) из этого файла, содержащих обнаруженную ошибку. Если в словаре обнаружены слова, похожие на ошибочное, то они выводятся ниже (с порядковыми номерами). Далее следует строка подсказки и командная строка программы.

В табл. 12.2 приведены клавиши, которыми можно пользоваться при работе с ispell.

Таблица 12.2. *Клавиши, используемые ispell*

Клавиши	Выполняемое действие
<R>	Заменить ошибочное слово (программа предложит набрать правильное слово в нижней строке экрана)
<Пробел>	Пропустить данное вхождение слова

Таблица 12.2 *(окончание)*

Клавиши	Выполняемое действие
<A>	Пропустить все вхождения данного слова в текущей сессии работы с программой
<I>	Пропустить это слово и включить его в персональный словарь (который хранится в файле .ispell_russian в домашнем каталоге пользователя)
<U>	То же самое, только слово записывается в нижнем регистре (маленькими буквами)
<Q>	Немедленный выход из программы (вначале запрашивается подтверждение, а проверяемый файл остается неизмененным; сделанные замены не проводятся)
<X>	Прервать проверку, записать проведенные изменения и выйти из программы
<!>	Временный выход в оболочку shell

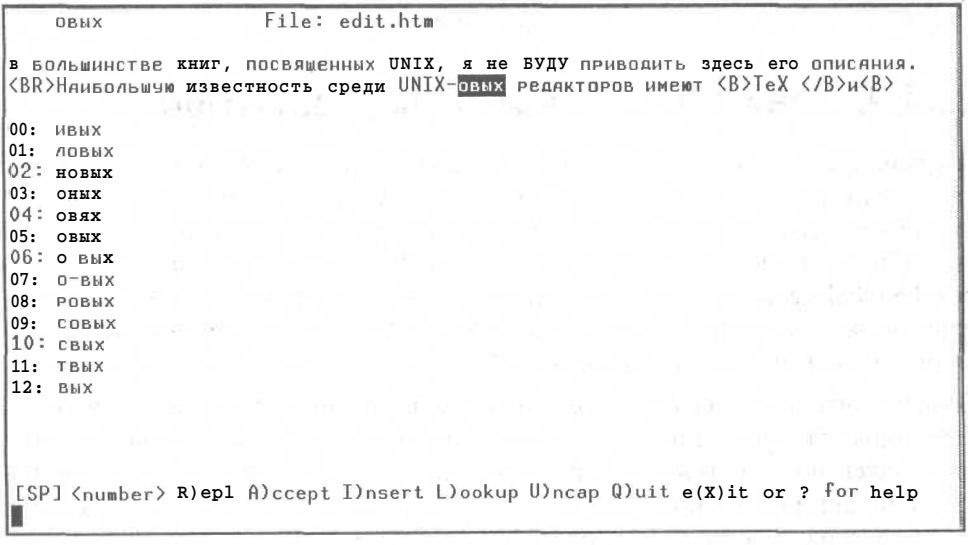


Рис. 12.6. Проверка правописания с помощью ispell

Если в качестве команды ввести порядковый номер одного из предложенных программой вариантов замены, то программа заменит ошибочное слово на слово, соответствующее набранному порядковому номеру варианта замены. Только номера надо вводить в точности так, как они предлагаются программой, т. е. с предшествующими значащим цифрам нулями (если они есть). И набирать номер надо, не нажимая предварительно клавишу <R>, иначе ошибочное слово будет заменено просто на соответствующую цифру.

Программа ispell, как уже упоминалось, используется в качестве модуля проверки правописания во многих текстовых редакторах, например, в Emacs она обеспечивает проверку правописания непосредственно в процессе подготовки текста.

Если подумать о принципе проверки, заложенном в программу, легко понять, что с ее помощью можно проверить только очень ограниченный класс ошибок, а именно, орфографические ошибки, состоящие в неправильном написании слов. Очевидно, что не будут обнаружены никакие ошибки в грамматических конструкциях, согласовании слов и т. д.

Еще один недостаток программы, с которым я столкнулся, проявляется в тех случаях, когда на текущем диске мало свободного места, меньше, чем необходимо для записи исправленного файла. Программа в таком случае записывает только ту часть файла, которая поместилась, и теряет все остальное. Никаких предупреждений при этом не выдается.

Если недостатка дискового пространства нет, то после внесения исправлений программа записывает исправленную версию файла, а исходный файл сохраняет, добавив к его имени расширение bak.

12.4. О трех типах текстовых редакторов

Редактирование текстовых файлов (с текстами на естественном языке, либо с текстами программ) — одна из наиболее часто выполняемых работ на любом компьютере и в любой операционной системе. Может быть, поэтому для Linux разработано уже очень много текстовых редакторов (на **www.linuxlinks.com** перечислены около 100 наименований, и это еще, вероятно, не все). Так что выбрать есть из чего. И стоит уделить некоторое время оптимальному выбору редактора.

Конечно, чтобы такой выбор был обоснован, в идеале надо опробовать все редакторы или большинство из них. Это, очевидно, невозможно, так что приходится положиться либо на случай, либо на мнение кого-то из знакомых или авторов компьютерных книг. (Кстати, неплохой, на мой взгляд, обзор текстовых редакторов содержится в книге А. Федорчука "Офис, графика, Web в Linux" *(см. [П 1.6] приложения)*. Я тоже попробую изложить свои впечатления о некоторых текстовых редакторах для Linux.)

Из всего множества различных текстовых редакторов рядовой пользователь обычно выбирает два-три, с которыми постоянно работает. Он заучивает до автоматизма управляющие комбинации клавиш, привыкает определенным образом, через команды меню или щелчки мышкой, выполнять стандартные операции редактирования и, вообще, приноравливается к среде редактора. Поэтому для смены редактора должны быть достаточно веские причины.

По моему мнению, рядовому пользователю, часто использующему компьютер для редактирования файлов, необходимо освоить, по крайней мере, три редактора.

Один из них — это мощный текстовый процессор, работающий в режиме WYSIWYG, обеспечивающий широкие возможности форматирования текста и массу дополнительных возможностей, отсутствующих в более простых редакторах. К этому типу я бы отнес текстовые редакторы (процессоры) из пакетов StarOffice, Applixware, KOffice, отдельные текстовые процессоры Maxwell и WordPerfect 8, AbiWord, а также издательскую систему TEX. Все редакторы этого типа я буду называть текстовыми процессорами. Правда, некоторые авторы, например, тот же А. Федорчук, все же делят их на просто редакторы и процессоры, однако я не вижу оснований для этого. Собственно говоря, весь вопрос в возможностях форматирования текста, которые предоставляет редактор. И какой-либо точный критерий для деления редакторов этого класса на два подкласса предложить трудно, если вообще возможно.

Второй необходимый редактор — это редактор для создания или правки ASCII-файлов, работающий в графическом режиме. С помощью этого редактора Web-мастер может, например, редактировать HTML-странички, в нем можно написать письмо для последующей отправки по e-mail и т. д. Это должен быть редактор графического режима, потому что во многих случаях в графическом режиме работать легче и удобнее, чем в текстовом. Примерами таких редакторов являются KEdit и KWrite из KDE, nedit.

И все же надо уметь пользоваться и одним из консольных текстовых редакторов, потому что вы, как единственный пользователь (и даже суперпользователь) персонального компьютера, должны уметь отредактировать конфигурационные файлы, причем в любой ситуации, даже тогда, когда графический режим не загружается. Выбор редакторов этого типа очень широк: vi, vim, bvi, Nvi, Elvis, Levee, vile, Wily, joe, aee, Fred, gred, le, lpe, Zed, Emacs, CoolEdit. Давние приверженцы UNIX чаще всего используют vi или его усовершенствованную версию vim, но тем, кто переходит на Linux из среды Windows проще будет освоить CoolEdit, имеющий большое сходство со встроенными редакторами программ Norton Commander и FAR.

Исходя из этих рассуждений нижеследующее изложение разбито на три больших части, каждая из которых посвящена одному из выделенных типов редакторов. Первый вариант раздела с описанием каждого рассматриваемого ниже редактора был создан в том самом редакторе, который в этом разделе описан (эти первые варианты размещались мной на сайте **http://linux-ve.chat.ru).** Я надеюсь, что после чтения этих разделов вы сможете определиться с выбором текстовых редакторов. Конечно, критерии выбора могут у каждого оказаться свои. Но думаю, что для нас, русскоязычных пользователей, немаловажным фактором является возможность вводить и редактировать тексты на русском языке. Желательно также, чтобы команды меню и

сообщения программы тоже были русифицированы (правда, до некоторой степени с английскими терминами здесь можно мириться, особенно если есть хорошее описание программы на русском, потому что число команд меню обычно невелико и их смысл можно просто запомнить).

12.5. Консольные редакторы ASCII-файлов

Начнем с рассмотрения редакторов текстового режима, т. е. работающих в консоли. Говоря о таких редакторах, просто нельзя не упомянуть о редакторахы vi и Emacs, но более основательно будет рассмотрен встроенный редактор оболочки Midnight Commander — CoolEdit.

12.5.1. Редакторы типа vi

Редактор vi (или его несколько доработанные потомки) по умолчанию включается в любую UNIX-подобную систему, в том числе и во все дистрибутивы Linux. Все приверженцы UNIX, имеющие значительный стаж работы с этими ОС, знают и используют этот редактор. Описание редактора vi вы сможете найти если не в любой, то уж точно в большинстве книг, посвященных UNIX. У редактора vi имеется несколько потомков, которые в чем-то его улучшают и усовершенствуют. Это такие редакторы, как Vim, bvi, Nvi, Elvis, Levee, vile, Wily (редактор Vim даже вызывается в Linux по команде vi). Краткий обзор редакторов этого класса вы можете найти в статье А. Фомичева "Текстовые редакторы для ОС UNIX" ("Открытые системы", № 4, 1994 г.). Однако для тех пользователей, которые мигрировали в Linux из среды Windows, все эти редакторы вряд-ли покажутся удобными. Причина в том, что в этих редакторах нет привычных меню и подсказок (насколько можно судить по Vim версии 5.3) и надо запомнить множество клавиатурных комбинаций для ввода команд. При этом работа в редакторе организована в виде двух раздельных режимов — ввода текста и ввода команд, и новичок часто просто путает режимы. В общем без печатного руководства за освоение редакторов этого типа я бы вам браться не рекомендовал. И хотя те, кто освоил vi, часто прибегают к его помощи для редактировании простых файлов (особенно, конфигурационных файлов и скриптов), я считаю, что есть более удобные средства, а поэтому не рассматриваю редакторы типа vi (хотя не упомянуть о них не мог).

12.5.2. Редактор Emacs

Наибольшую известность среди редакторов, используемых в Linux, имеет Emacs. Он существует как в варианте для текстового режима, так и в варианте для графической оболочки. Некоторые даже говорят, что Emacs — это

не редактор, а образ жизни, а в другом источнике его называют религией. Если вы хотите ближе познакомиться с Emacs, я могу рекомендовать вам недавно вышедший русский перевод книги Р. Столлмана о нем; поищите эту книгу, например, в виртуальном магазине "Болеро". Прекрасное вводное описание этого редактора вы найдете в книге А. Федорчука *(см. [П1.6] приложения)*. Я, однако, считаю, что начинающим пользователям для редактирования ASCII-файлов целесообразно использовать CoolEdit — встроенный редактор файлового менеджера Midnight Commander.

12.5.3. CoolEdit — встроенный редактор программы Midnight Commander

CoolEdit — это простая в использовании программа с привычными для большинства пользователей (особенно для тех, кто работал с Norton Commander под DOS или с FAR под Windows) комбинациями управляющих клавиш. Кроме того, надо учитывать, что обычно любая работа с файлом начинается с файлового менеджера, поскольку вначале нужно, как минимум, найти файл. Midnight Commander как раз и является таким файловым менеджером, причем переход к редактированию осуществляется простым нажатием клавиши <F4> после установки подсветки на имя найденного файла. Это мелочь, но удобно. Учитывая изложенные соображения, я начну подробный рассказ о текстовых редакторах именно с CoolEdit.

Встроенный редактор Midnight Commander (рис. 12.7) вызывается во время работы в этой программе нажатием клавиши <F4> при условии, что в инициализационном файле Midnight Commander установлена в 1 опция `use_internal_edit`. Его также можно вызвать независимо от Midnight Commander просто из командной строки командой `mcedit`. Однако его истинное имя все же CoolEdit, в этом вы можете убедиться, если в режиме редактирования нажмете клавишу <F9> и выберете команду меню **Файл | Об авторах.**

Этот редактор обеспечивает выполнение большинства функций редактирования, присущих полноэкранным редакторам текста. С его помощью можно редактировать файлы практически любого размера, поскольку верхняя граница для размера редактируемого файла составляет 16 Мбайт. Обеспечивается редактирование исполняемых (двоичных) файлов без потери данных.

Поддерживаются следующие возможности:

- копирование, перемещение, удаление, вырезание и вставка блоков текста;
- отмена предыдущих операций (по комбинации клавиш <Ctrl>+<U>);
- выпадающие меню;
- вставка файлов;
- макроопределения;

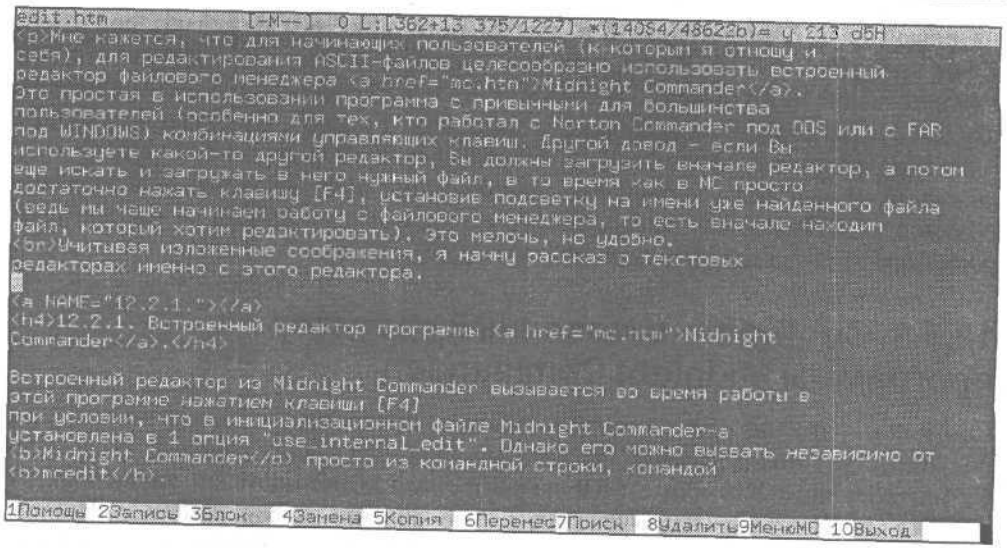


Рис. 12.7. Внешний вид окна редактора CoolEdit

- поиск и замена по регулярным выражениям (другими словами по шаблонам, сформированным с использованием специальных символов) а также собственный вариант операций поиска и замены, основанный на функциях scanf-printf;
- выделение текста комбинацией клавиш <Shift>+<стрелки> в стиле MS Windows - MAC (только для Linux-консоли);
- переключение между режимами вставки/замены символа.

Редактор очень прост и практически не требует обучения (тем более что можно найти версии, в которых основная часть команд меню русифицирована, такая версия была включена, например, в дистрибутив Black Cat 5.2) Для того чтобы узнать, какие клавиши вызывают выполнение определенных действий, достаточно просмотреть выпадающие меню, вызываемые нажатием клавиши **<F9>** в окне редактора (рис. 12.8).

Если вы работаете в консоли Linux, для работы с блоками текста можно использовать следующие комбинации клавиш:

- <Shift>+<клавиши стрелок> - выделение блока текста;
- <Ctrl>+<Ins> - копирует блок в файл cooledit.clip;
- <Shift>+<Ins> производит вставку последнего скопированного в cooledit.clip блока в позицию курсора;
- <Shift>+<Del> — удаляет выделенный блок текста, запоминая его в файле cooledit.clip.

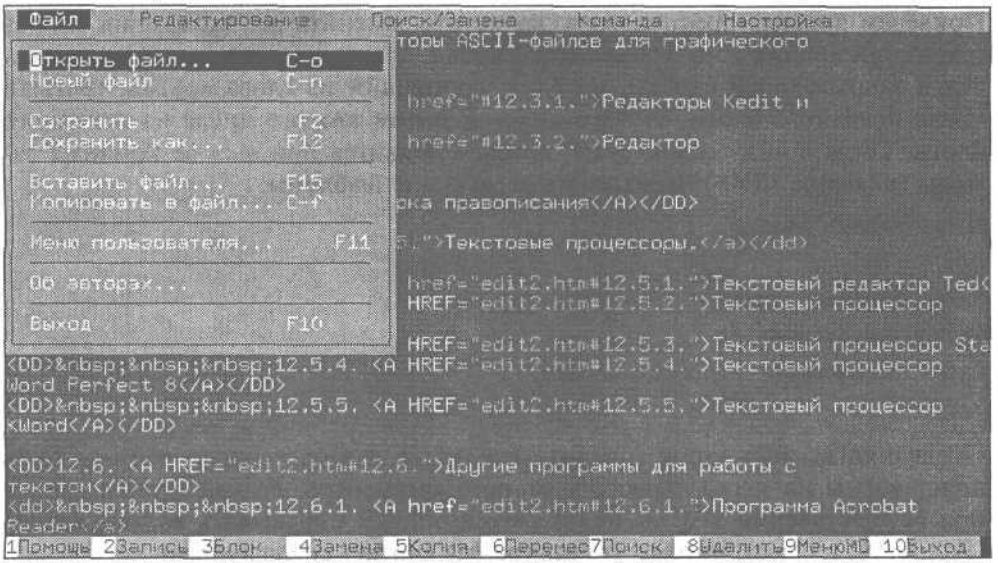


Рис. 12.8. Выход в меню **Cooledit** по клавише <F9>

Если у вас установлена программа gpm — драйвер мыши для консоли, то вы можете нажать на левую кнопку мыши в начале выделяемого блока, перенести курсор в конец блока и отпустить кнопку (тем самым выделить текст), а затем перенести курсор туда, куда надо вставить фрагмент, и нажать на правую кнопку мыши для вставки выделенного текста.

Редактор поддерживает макросы. Для того чтобы определить макрос, нажмите комбинацию клавиш <Ctrl>+<R>, после чего введите строки команд, которые должны быть выполнены. После завершения ввода команд снова нажмите <Ctrl>+<R> и свяжите макрос с какой-нибудь клавишей или комбинацией клавиш, нажав эту клавишу (комбинацию). Макрос будет вызываться нажатием комбинации <Ctrl>+<A> и назначенной для него клавиши. Макрос можно также вызвать нажатием любой из клавиш <Alt>, <Ctrl> или <Esc> и назначенной макросу клавиши, при условии, что данная комбинация не используется для вызова какой-либо другой функции.

Макро-команды после определения записываются в файл cedit/cooledit.macros в вашем домашнем каталоге. Вы можете удалить макрос удалением соответствующей строки в этом файле.

При выполнении операций замены (по функциональной клавише <F4>) вы можете использовать функции поиска и замены scanf для поиска и замены в соответствии с шаблонами формата языка C. Вначале посмотрите man-страницы sscanf и sprintf, чтобы узнать, что такое шаблоны формата и как они работают.

Приведем пример: предположим, вы хотите заменить все вхождения блоков текста, состоящих из открывающей скобки, трех разделенных запятыми чисел и закрывающей скобки, на блок, состоящий из слова apples, третьего числа исходного блока, слова oranges и потом второго числа из исходного блока. Тогда в диалоговом окне, которое появится при вызове команды замены (клавиша <F4>), надо задать следующие шаблоны:

```
Enter search string
(%d,%d,%d)
Enter replace string
apples %d oranges %d
Enter replacement argument order
3,2
```

Последняя из этих строк говорит, что третье и второе число должны быть подставлены на места первого и второго аргументов. Рекомендуется все же при осуществлении замены пользоваться опцией **Спрашивать подтверждение** (Prompt on replace), потому что программа считает совпадениями все случаи, когда число аргументов совпадает с заданным, хотя это не всегда означает полное совпадение. Scanf также не обращает внимания на количество символов пробела.

Встроенный редактор обрабатывает символы из второй половины кодовой таблицы (160+). Но когда редактируете бинарные файлы, лучше установить опцию **Биты символов** (Display bits) из меню **Настройки** в положение 7 бит, чтобы сохранить формат файла.

Для того чтобы описать здесь все функции встроенного редактора, потребовалось бы слишком много места. Да в этом и нет нужды, поскольку для его использования достаточно запомнить, что все основные операции можно выполнить через команды меню, которое вызывается нажатием клавиши <F9> в окне редактирования. Кроме того, можно прочитать man-страницу ПО команде man mcedit ИЛИ info mcedit.

В завершение этой краткой справки по встроенному редактору программы Midnight Commsnder мне хочется рассказать о том, как осуществляется перенос фрагментов текста из одного файла в другой.

Если вы работаете в консоли, то эта задача решается через меню или с помощью следующих операций:

1. Отмечаем начало блока с помощью клавиши <F3>.
2. Перемещаем курсор к концу блока.
3. Отмечаем конец блока с помощью клавиши <F3>.
4. Набираем комбинацию клавиш <Ctrl>+<Insert>.
5. Закрываем этот файл, открываем другой.

6. Ставим курсор туда, куда хотим вставить данный фрагмент, и нажимаем комбинацию клавиш <Shift>+<Insert>.

Все, задача выполнена. Перенос фрагмента текста этим способом может быть произведен из одной виртуальной консоли в другую.

Но все это работает только в консоли. А при работе с тем же редактором в окне графической оболочки та же задача была для меня достаточно долгое время проблемой.

Я не сразу нашел способ ее решения (то есть переноса фрагмента текста), который работает как в консоли, так и в окне графической оболочки. Этот способ состоит в переносе фрагмента текста через другой файл (по умолчанию используется файл ~/.cedit/cooledit.clip). Выделите фрагмент текста, выберите команду меню **Файл | Копировать в файл** (File | Copy to file) и нажмите клавишу <Enter>. Затем переходите в другой файл, ставите курсор туда, куда надо вставить фрагмент, и выбираете команду меню **Файл | Вставить файл** (File | Insert file).

12.6. Редакторы ASCII-файлов для графического режима

Очевидно, что было бы очень удобно, если бы редактирование ASCII-файлов в графическом режиме осуществлялось с помощью тех же редакторов, которые применяются в консольном режиме. Тогда не пришлось бы заучивать другие комбинации клавиш, менять привычную среду. Редактор CoolEdit, о котором было рассказано в *разд. 12.5.3*, может запускаться через эмулятор терминала и использоваться, таким образом, в графическом режиме. Однако, при этом оказывается, что некоторые комбинации клавиш в эмуляторе не работают или работают не так, как в консоли. Кроме того, работа с мышкой в редакторах, изначально ориентированных на графический режим, организована гораздо лучше, чем в консольных редакторах, в силу чего повышается общее удобство работы (хотя, может быть, это чисто мое субъективное ощущение). Поэтому я все же рекомендую не ограничиваться использованием только консольного редактора для ASCII-файлов, а освоить один из многочисленных редакторов, ориентированных на обработку таких файлов в графическом режиме. Рассмотрим три подобных редактора.

12.6.1. Редактор KEdit

Редакторы KEdit и KWrite входят в состав графической среды KDE. Они предназначены для работы в графическом режиме, но работают с ASCII-файлами. Редакторы очень похожи, поэтому я расскажу вначале о KEdit, а затем просто укажу на отличия, имеющиеся во втором редакторе.

Редактор KEdit (рис. 12.9) очень прост в использовании и, если вы вообще когда-нибудь занимались редактированием текста, у вас не возникнет с ним проблем. К тому же, если KDE у вас уже русифицирован, то не возникнет никаких проблем с вводом и отображением кириллических символов. Более того, меню и сообщения программы тоже русифицированы.

Открыть файл для редактирования можно через меню или по комбинации клавиш <Ctrl>+<O>. При этом появляется окно, в котором можно выбрать файл для редактирования. Тут все привычно, и думаю, что дополнительных пояснений вам не потребуется. Кроме этого способа, работая в среде KDE, можно воспользоваться и методом "drag-and-drop". Это значит, что из файлового менеджера Konqueror вы можете прихватить файл мышкой и просто "бросить" его в окно KEdit.

Основные операции редактирования осуществляются с помощью клавиатуры. По клавише <Insert> происходит переключение между режимами вставки и замены символов. Перемещение по тексту осуществляется с помощью клавиш-стрелок и клавиш <Page Down>, <Page Up>, <Home> (переход в начало строки), <End> (переход в конец строки). Выделить блок текста (к сожалению, только линейный) можно с помощью мыши или клавишами стрелок при нажатой клавише <Shift>. Клавиша <Delete> удаляет символ справа от курсора или выделенный блок текста.

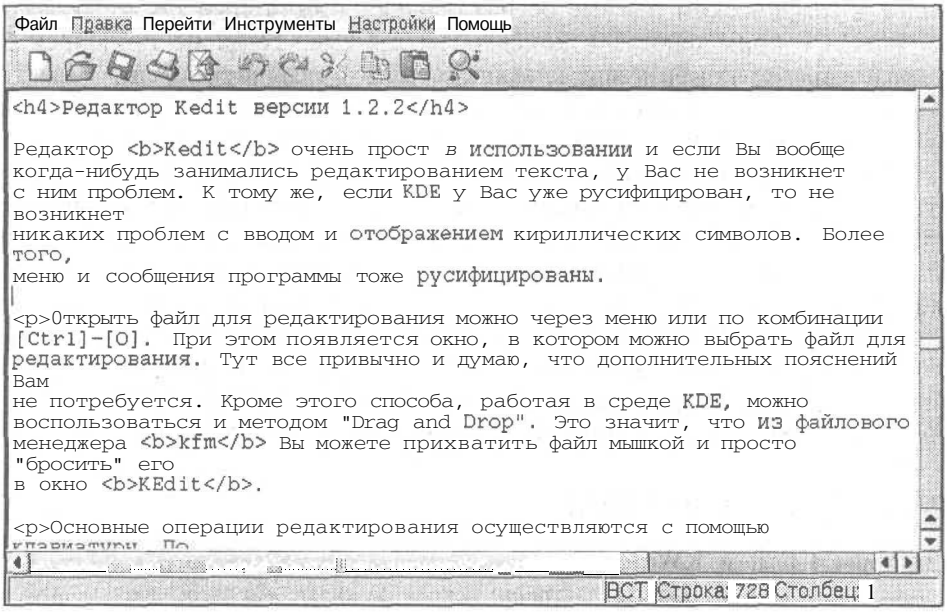


Рис. 12.9. Окно редактора KEdit

Во многих редакторах, и в том числе в KEdit, операции перемещения по тексту и удаления можно выполнять также с помощью "горячих" комбинаций клавиш (возможно, в связи с тем, что на старых терминалах не было клавиш стрелок?):

- <Ctrl>+<A> — переместить курсор в начало строки;
- <Ctrl>+<B> — переместить курсор на один символ вправо;
- <Ctrl>+<E> — переместить курсор в конец строки;
- <Ctrl>+<N> — переместить курсор на одну строку вниз;
- <Ctrl>+<P> — переместить курсор на одну строку вверх;
- <Ctrl>+<D> — удалить символ справа от курсора;
- <Ctrl>+<H> — удалить символ слева от курсора.

Основные команды по переносу блоков текста из одного места в другое осуществляются с помощью тех же комбинаций, которые используются в редакторе из Midnight Commander и многих других редакторах, так что пальцы находят эти комбинации уже автоматически:

- <Ctrl>+<C> - скопировать выделенный текст в буфер обмена (clipboard);
- <Ctrl>+<X> — вырезать выделенный фрагмент текста и поместить его в буфер обмена;
- <Ctrl>+<V> — вставить фрагмент текста из буфера обмена в позицию курсора;
- <Ctrl>+<K> — удалить текст от курсора до конца строки и поместить его в специальный буфер (kill-buffer);
- <Ctrl>+<Y> — вставить фрагмент текста из специального буфера в позицию курсора.

С помощью комбинации <Ctrl>+<J> можно отформатировать текущий абзац текста. Форматирование, впрочем, заключается только в том, что очень длинные строки текста разбиваются на такие, длина которых не превышает величину, заданную в соответствующей опции команды меню **Настройки.**

Комбинация клавиш <Ctrl>+<F> вызывает функцию поиска подстроки.

Для сохранения результатов редактирования в файле можно воспользоваться командой меню **Файл | Сохранить** или комбинацией клавиш <Ctrl>+<S>. Если вы хотите сохранить редактируемый текст в файле с другим именем, то тут надо действовать только через меню **(Сохранить как).**

KEdit поддерживает печать. Вы можете распечатать как весь редактируемый файл, так и выделенный фрагмент текста. Печать осуществляется на принтере, заданном по умолчанию. Дополнительно можно подключить любую из утилит печати, которые во множестве разработаны для UNIX.

Что меня особенно восхитило при первом знакомстве с KEdit, так это встроенная возможность проверки правописания и возможность просмотра и редактирования файлов как в кодировке CP-1251, так и в кодировке KOI8-R.

Для смены кодировки надо поиграть выбором шрифтов в команде меню **Настройки | Шрифт.** К сожалению, не все шрифты поддерживают обе кодировки, но после нескольких попыток удается выбрать нужный шрифт. Для проверки правописания достаточно выбрать команду меню **Редактирование | Проверка правописания.** Естественно, что в системе должна быть установлена программа ispell, и, кроме того, перед началом проверки стоит воспользоваться командой меню **Настройки | Проверка правописания** и выбрать русский словарь и желаемую кодировку символов (см. рис. 12.10).

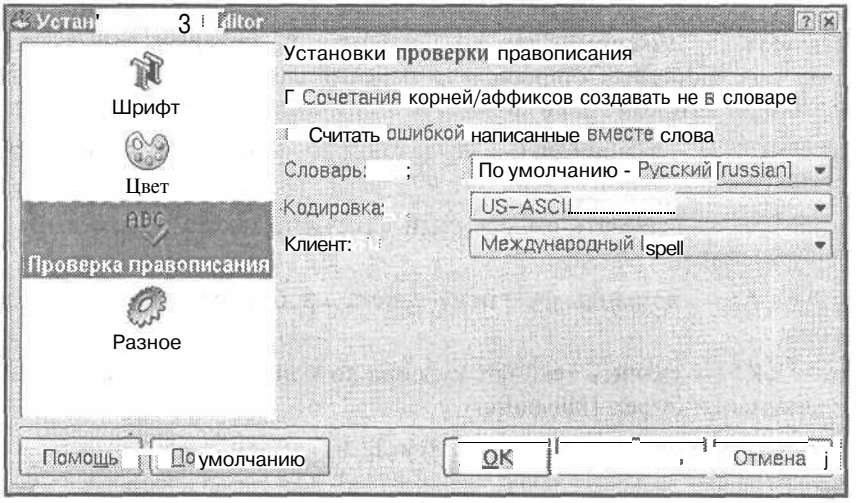


Рис. 12.10. Настройка проверки правописания в Kedit

Кстати, меню **Настройки** вообще желательно повнимательнее посмотреть перед началом редактирования каких-то файлов, выбрать нужные вам установки, после чего сохранить их (команда **Записать установки).**

Еще одна приятная, на мой взгляд, встроенная функция редактора KEdit – возможность вызова почтовой программы прямо из редактора. Я не смог проверить работу этой функции, поскольку не имею возможности посылать письма с домашнего (изолированного) компьютера. Но такой функцией я часто пользуюсь на работе, в MS Word, и она кажется мне очень полезной и удобной.

В общем, этот редактор с первого знакомства произвел на меня самое благоприятное впечатление, так что я рекомендую вам познакомиться с ним

поближе. Возможно, вы даже изберете его для редактирования ASCII-файлов при работе в графическом режиме.

12.6.2. Редактор KWrite

Редактор KWrite, как уже было сказано, очень похож на KEdit. Первое отличие, которое бросается в глаза после загрузки в редактор какого-то файла, — раскраска служебных слов. Раскраска задается командами **Установить раскраску и Раскраска** вложенного меню **Настройки.** Поскольку я часто занимаюсь редактированием HTML-файла, эта особенность показалась мне удобной.

В основном меню появилось одно новое вложенное меню — **Закладки** с командами **Установить закладку** (комбинация клавиш <Alt>+<S>), **Добавить закладку и Очистить закладки** (комбинация клавиш <Alt>+<C>). Остальные элементы главного меню — те же, что и у KEdit.

Появилась функция отката клавишами <Ctrl>+<Z> и возможность выделения вертикального столбца (эта функция включается клавишей <F5>). По комбинации клавиш <Ctrl>+<I> в текущую строку вставляется отступ (добавляется пробел в начало строки). По комбинации клавиш <Ctrl>+<U> отступ убирается (а если отступа не было — молча удаляется первый символ строки).

Пожалуй, это все, что появилось нового в KWrite по сравнению с KEdit. Но, к сожалению, кое-какие полезные функции пропали. В меню **Файл** исчезла команда, касающаяся отправки файла по электронной почте. В меню **Правка** отсутствует возможность вставить дату. Исчезла также возможность выбрать шрифт по команде **Настройка | Настройка KWrite,** что я считаю самым большим недостатком этой программы.

А в остальном эти редакторы практически одинаковы, так что работать можно с любым из них. Я думаю, что со временем все возможности редактора KEdit будут иметься и в KWrite, и он станет "штатным" текстовым редактором оболочки KDE.

12.6.3. Текстовый редактор Nedit версии 5.1.1

Текстовый редактор Nedit (рис. 12.11) создан группой авторов во главе с Марком Эделем (Mark Edel), распространяется на основе лицензии GPL, и его можно получить на Web-сайте **http://nedit.org.** Внешне он очень похож на два редактора, описание которых было дано в предыдущем разделе, но обладает гораздо большими возможностями, чем KEdit или KWrite.

После первого же знакомства с этим редактором я решил отказаться от использования KWrite и перейти на Nedit. А. Федорчук *(см. [П1.6] приложения)* тоже считает, что Nedit приближается к идеалу текстового редактора, и даже называет его "лучшим редактором всех времен и народов".

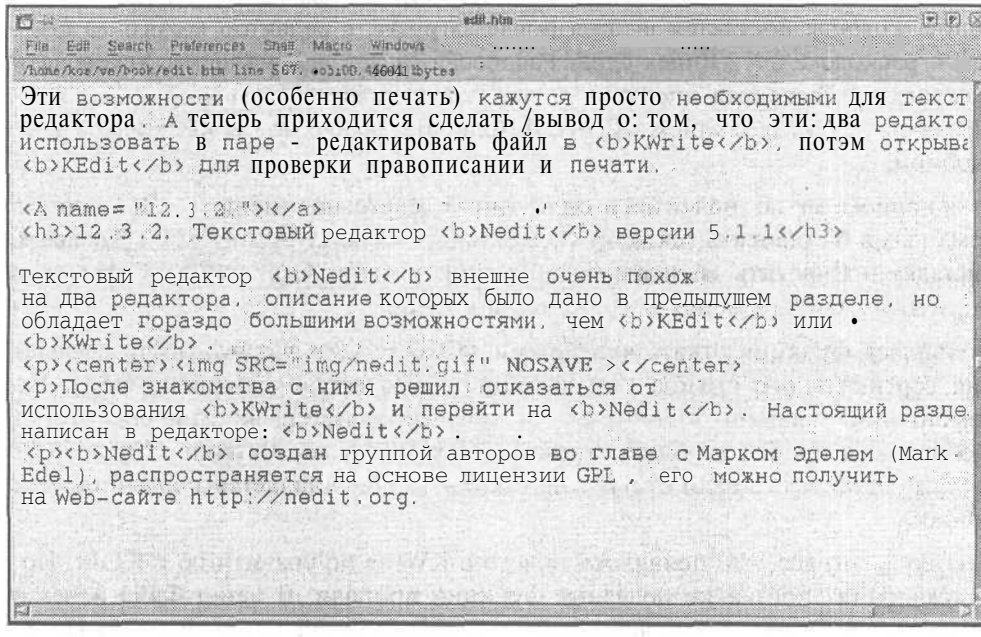


Рис. 12.11. Внешний вид окна редактора Nedit

Пожалуй, единственным (на мой взгляд, не очень существенным) недостатком Nedit по сравнению с KEdit или KWrite можно считать только то, что меню, сообщения и подсказки выдаются на английском языке. Правда, в меню нет такой команды, как отправка почтового сообщения, но зато есть возможность выполнить любую команду оболочки shell или создать макрос, что с лихвой перекрывает отсутствие прямого вызова почтовой программы.

Переключить редактор на ввод русскоязычного текста для меня не составило труда. Для этого нужно войти в команду меню **Preferences | Text Fonts** и задать нужные шрифты в строках ввода соответствующего окна (рис. 12.12). А поскольку запомнить эти длинные строки с именами шрифтов довольно трудно, можно воспользоваться кнопкой **Browse** и выбрать шрифт из выводимого списка (рис. 12.13). Выбирать надо шрифт, имеющий кодировку KOI8-R. Обратите внимание, что для выбора шрифта надо отметить по одному варианту в каждом из трех предлагаемых столбцов, иначе вы, вероятнее всего, получите сообщение о том, что такой шрифт отсутствует.

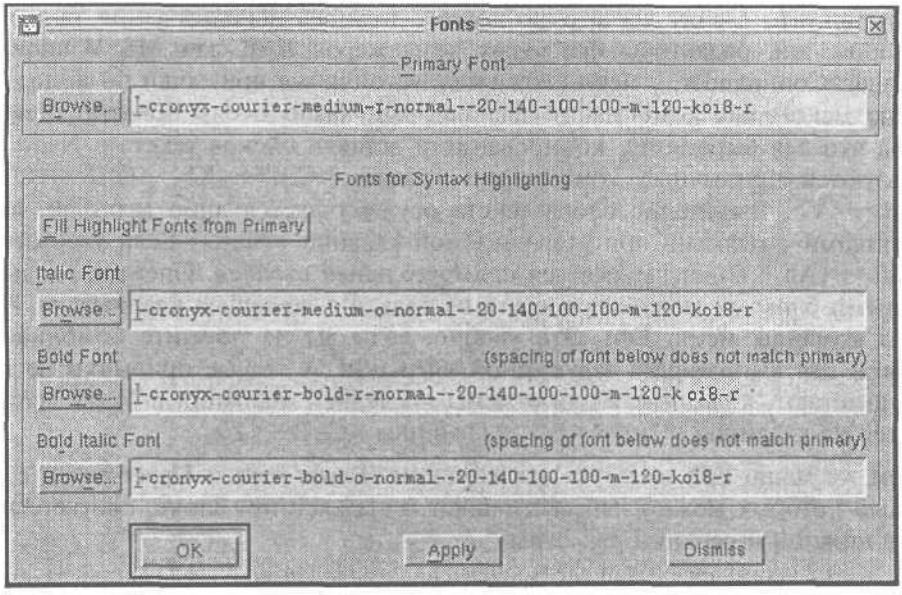


Рис. 12.12. Окно задания шрифтов в Nedit

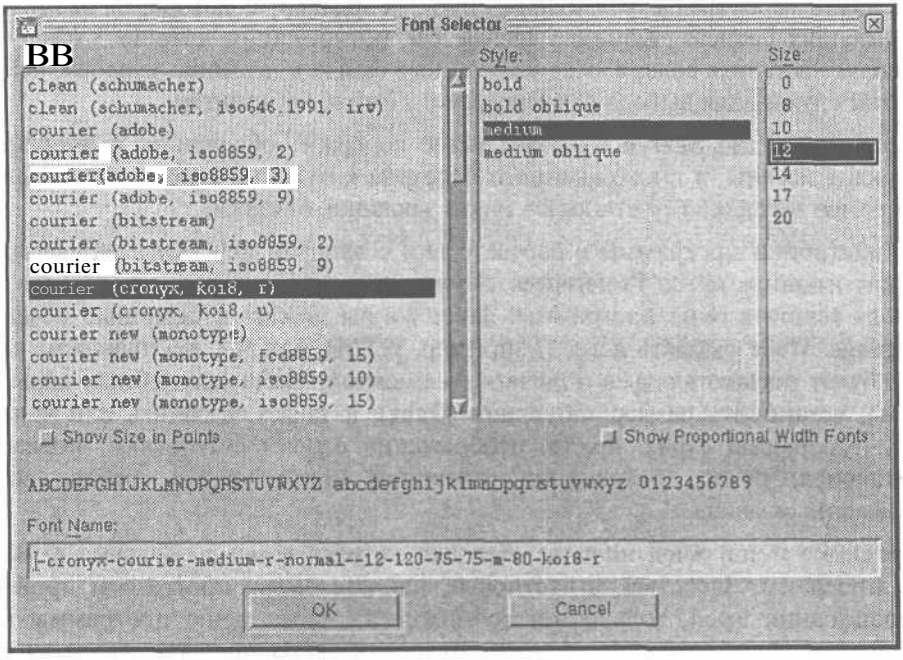


Рис. 12.13. Выбор шрифтов в Nedit

Учитывая, что в силу достаточно долгого периода использования таких редакторов, как редакторы файловых менеджеров FAR для MS Windows и Midnight Commander, у меня сложились устойчивые привычки по использованию некоторых комбинаций клавиш, мне было очень приятно обнаружить, что для вырезания, копирования и вставки блоков текста в Nedit используются привычные комбинации клавиш <Ctrl>+<X>, <Ctrl>+<C> и <Ctrl>+<V>. Выделение блока текста осуществляется либо мышкой, либо клавишами-стрелками при удерживаемой клавише <Shift>. Если удерживать <Shift>+<Alt>, будет выделяться прямоугольный столбец. Операции с выделенными блоками можно проделать не только с помощью клавиатуры, но и через команды меню Edit. Это удобно, если вы не помните комбинацию клавиш для выполнения задуманной операции. А старые привычки не всегда помогают, в частности для отмены последней операции в Nedit используются не клавишы <Ctrl>+<U>, а клавишы <Ctrl>+<Z>.

В том же меню **Edit** имеются две команды **(Lower-case и Upper-case),** с помощью которых можно перевести текст в выделенном блоке, соответственно, в нижний и верхний регистры.

Вложенное меню **File** главного меню содержит команды, с помощью которых выполняются обычные операции открытия и сохранения редактируемых файлов. Обратите внимание на то, что с помощью команды **Open Previous** вы легко можете вернуться к редактированию тех файлов, с которыми работали в предыдущих сеансах. Команда **Include file** (комбинация клавиш <Alt>+<I>) позволяет вставить содержимое выбранного файла в позицию курсора, команда **Print** служит для вывода редактируемого файла на принтер.

Вложенное меню **Search** главного меню содержит обращения к операциям поиска и замены, а также быстрого перехода к нужной строке или к сделанной ранее закладке (и установки такой закладки-отметки).

Для настройки программы в соответствии с вашими вкусами и привычками служит команда меню **Preferences.** Я уже рассказал, как выбрать шрифт для вывода текста в окне программы. Здесь же вы можете с помощью команды **Language Mode** выбрать язык (например, HTML или C), по правилам которого будет осуществляться подсветка элементов текста (служебных слов или тегов), установить размер табуляции (Tabs), а также включить или выключить нумерацию строк текста, отображение строки статистики, подсветку синтаксиса, режим сохранения резервной копии файла, режим вставки/замены символа.

Вложенное меню **Shell** используется для запуска команд оболочки и внешних программ. Здесь вы, в частности, найдете вызов программы проверки правописания ispell, только нет возможности через меню программы подключить русский словарь. Но эта трудность легко преодолевается за счет использования файла ресурсов .nedit в домашнем каталоге пользователя. Достаточно в строке, определяющей вызов программы ispell, добавить пара-

метр -d russian и проверка русскоязычных текстов заработает (в отдельном окне терминала).

Через тот же файл ~/.nedit можно настроить цвета фона и текста, геометрию окна программы и некоторые другие параметры. То же самое можно сделать и в специальной секции [nedit] файла .Xdefaults в вашем домашнем каталоге. Правда, такой секции в этом файле по умолчанию нет, но ее можно создать, как это рекомендуется в книге А. Федорчука.

Но вернемся к меню. Меню **Windows** служит для переключения между окнами (которых можно открыть одновременно несколько). Назначение меню **Help**, я думаю, понятно без дополнительных пояснений. К сожалению, подсказка дается по-английски и отсутствует поиск подсказки по ключевым словам и фразам.

В этом кратком описании я не касался меню **Macro,** а также тех элементов главного меню, которые ориентированы на действия, выполняемые при редактировании исходных текстов программ (например, вставка кодов ASCII, подсветка парных скобок или компиляция программного кода). Но даже и на основании того, что было сказано в данном разделе, можно сделать вывод, что Nedit — это достаточно мощный и удобный редактор для работы с ASCII-файлами в графическом режиме. Если вы хотите узнать о нем побольше, почитайте статьи А. Федорчука в Интернете или его книгу *(см. [П1.6] приложения).*

12.7. Текстовые процессоры

Давайте договоримся, что термином "текстовые процессоры" будем обозначать все программы для редактирования текстовых файлов, работающие в графическом режиме и использующие при сохранении результатов работы в файлах специальные символы или вставки для обозначения элементов форматирования. Эти вставки текстовый процессор не отображает в виде каких-либо символов в процессе редактирования. В этом смысле Nedit не относится к текстовым процессорам, а Netscape Composer — относится, потому что показывает элементы форматирования в графическом виде, а не в виде HTML-тегов.

12.7.1. Возможности текстовых процессоров

К стандартным средствам форматирования текста относятся:

- возможности выбора различных шрифтов для разных частей текста в одном документе;
- задание ширины полей, величины отступов, интервалов;
- организация текста в виде колонок;
- вставка в текст рисунков, таблиц;

- возможности создания надписей под углом к строке или искривленной формы;
- проверка правописания (для языка пользователя)

и т. д.

Для того чтобы текстовый процессор мог считаться мощным и удобным, необходимо, чтобы в нем имелись возможности обработки файлов, созданных в других распространенных текстовых процессорах. (Конечно, в первую очередь имеется в виду MS Word!) Как минимум нужно уметь просматривать документы в форматах других редакторов. Можно, конечно, иметь перекодировщики, но с точки зрения удобства использования — это уже минус.

Имея в виду это неформальное определение, рассмотрим несколько программных продуктов для Linux, претендующих на высокое звание текстового процессора.

Поскольку по своему основному предназначению текстовые редакторы тесно связаны с языком, а мы рассматриваем случай русского языка, при выборе текстового процессора особое внимание приходится уделять тому, имеются ли в нем возможности русификации. Чтобы не повторять одно и то же при описании каждого из рассматриваемых ниже продуктов, замечу сразу, что в моей системе были установлены не только стандартные шрифты Type 1, поставляемые с дистрибутивом Black Cat 6.02, но и шрифты TrueType от Windows *(см. гл. 11)*.

12.7.2. Текстовые процессоры для Linux

Текстовых процессоров для Linux существует множество. Наиболее известны из них StarWriter из пакета StarOffice фирмы Sun и процессор Word Perfect 8. Аналог последнего, разработанный для ОС Windows, долгое время на равных конкурировал с пакетом MS Word, что, конечно, говорит о его высоком качестве. Однако с русификацией Linux-версии этого процессора какие-то проблемы, поэтому мы его здесь не рассматриваем.

Особое место среди текстовых процессоров занимает издательская система TEX и основанные на ней продукты типа Lyx. Что касается TEX, то это скорее язык программирования, чем текстовый редактор. Если вы программируете на двух-трех разных языках, вам, возможно, не составит труда освоить и TEX. Но простому пользователю, которому редактор нужен для написания деловых писем или диссертации по истории паровых котлов, на мой взгляд, вряд ли стоит браться за его изучение.

12.7.3. Текстовый редактор Ted

Редактор Ted задуман автором (Mark de Does) как простой текстовый редактор, работающий под X Window в UNIX/Linux-системах и играющий при-

мерно ту же роль, что и WordPad под MS Windows, только имеющий больше возможностей. Основное преимущество Ted по сравнению с редакторами, которые рассматривались в предыдущих разделах, — это возможность не только вводить и редактировать текст, но и обеспечивать простейшие возможности форматирования текста, изменять шрифты, вставлять в текст таблицы и рисунки. Именно в силу наличия этих возможностей я причислил этот редактор к текстовым процессорам, хотя он и не обладает всеми возможностями развитых текстовых процессоров.

Дистрибутив пакета можно скачать с сайта **ftp://ftp.nluug.nl/pub/editors/ted** или с сервера metalab: **[http,ftp]://metalab.unc.edu/pub/packages/editors/ted**.

Дистрибутив поставляется либо в виде сжатого TAR-архива, либо в виде RPM-пакета. Русифицированный вариант редактора можно найти на сервере ftp://ftp.logic.ru/ (конечно, русификация по версиям чуть отстает).

Я устанавливал и русифицированную версию 2.6 редактора Ted, однако после некоторого периода испытаний вернулся к англоязычной версии 2.7, которая датирована 31 декабря 1999 г., и в которой устранены некоторые недостатки предыдущей версии.

После ее запуска появляется окно, в котором изображена гравюра в стиле Альбрехта Дюрера, указана версия программы, ссылка на сайт разработчиков и имеется всего три вложенных меню: **File, Window и Help.** Очевидно, что для того, чтобы открыть существующий файл для редактирования, нужно выбрать команду **Open** меню **File** и выбрать в появившемся окне нужный файл. Для создания нового файла соответственно выбираем команду New меню **File** или нажимам комбинацию клавиш <Ctrl>+<N>. После открытия файла окно программы приобретает вид, изображенный на рис. 12.14.

Ted поддерживает формат файлов RTF (Rich Text Format), что позволяет обеспечить некоторую степень совместимости с продуктами от Microsoft. А именно, любой файл, созданный в Ted, и сохраненный в формате RTF, будет корректно прочитан программами MS Word под Windows. Обратная совместимость (то есть возможность открыть любой RTF-файл в Ted) не достигнута, поскольку некоторые опции форматирования, используемые в RTF-файлах, программой Ted не воспринимаются. Тем не менее текст открываемого файла будет выведен на экран полностью и может быть сохранен в другом (или том же самом) файле, но уже без той информации о форматировании, которая игнорируется при открытии файла.

Приведу краткий список основных возможностей Ted.

- Редактирование файлов в режиме Wysiwyg. При этом вы можете использовать все шрифты, для которых у вас имеются .afm-файлы и которые доступны для X11. Ted поставляется с .afm-файлами для шрифтов Adobe, которые имеются для систем, основанных на Motif, и могут использоваться со всеми PostScript-принтерами: Times, Helvetica, Courier и Symbol.

Другие шрифты могут быть добавлены с помощью обычных процедур, используемых для этого в X11. Поддерживаются такие свойства шрифтов, как выделение (bold), курсив (italic), подчеркивание, верхние и нижние индексы.

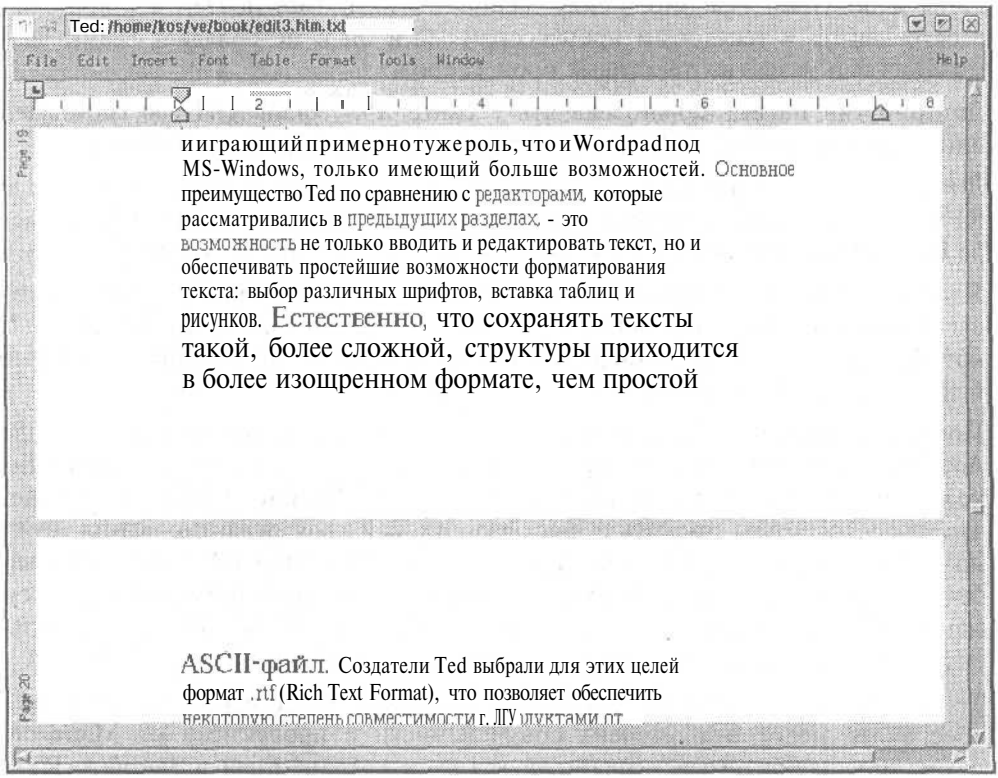


Рис. 12.14. Основное окно редактора Ted

- Вставка рисунков в формате bitmap.
- Печать на PostScript-принтерах.
- Работа с файлами в формате Acrobat PDF.
- Проверка правописания для 12 языков, основанных на латинском алфавите.
- Отправка документов по e-mail не выходя из Ted.
- Операции Cut/Copy/Paste, включая обмен с другими приложениями.
- Поиск/Замена.
- Форматирование абзацев: отступ на первой строке, смещение всего абзаца, использование линеек.

- О Разбиение текста на страницы.
- ❒ Вставка таблиц и операции с таблицами: операции с колонками и строками таблиц, изменение размеров с помощью линеек.
- ❒ Поддерживается вставка символов из разных кодовых таблиц.
- ❒ Гиперссылки и закладки (bookmarks).
- О Сохранение документа в текстовом и HTML-формате.

Ted можно использовать для чтения сообщений электронной почты, посланных с Windows-машин. Можно также сконфигурировать Ted как просмотрщик RTF-файлов для Netscape.

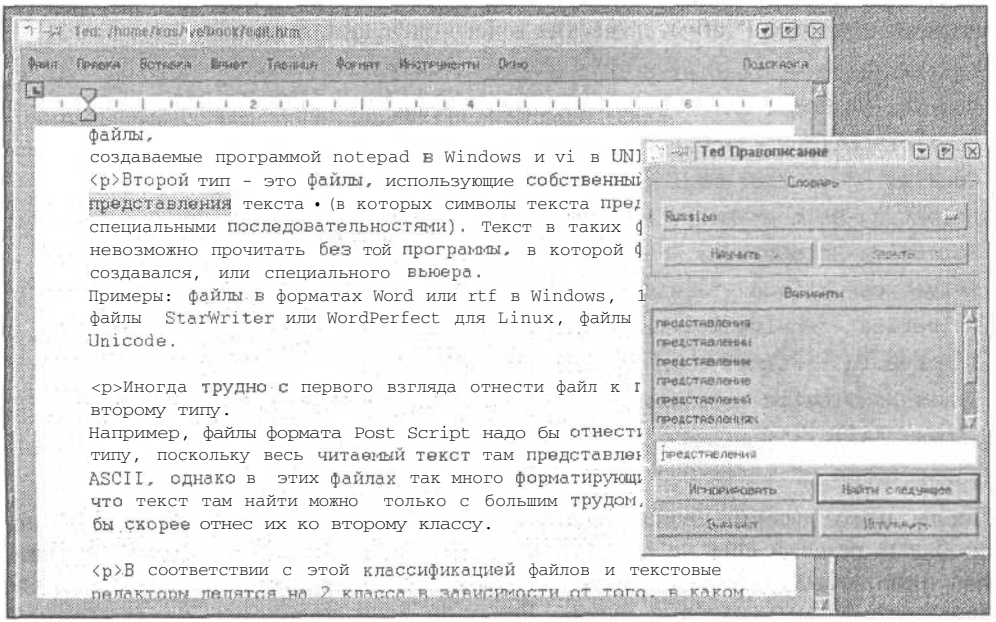


Рис. 12.15. Проверка правописания в редакторе Ted

О том, что Ted может быть полностью русифицирован, я узнал из статьи Виктора Вагнера *(см. [П14.19] приложения)*. Самая интересная возможность русифицированной версии — наличие проверки правописания для русскоязычного текста. Вызов этой функции производится с помощью команды меню **Инструменты | Правописание.** При этом появляется дополнительное окно (рис. 12.15), в котором надо щелкать по кнопке **Найти следующее** для поиска очередной ошибки.

Правда, судя по первому впечатлению, в той версии, которую я видел (версия 2.6), реализация данной функции имеет много недостатков. Во-

первых, все слова, набранные латинским шрифтом, обозначаются как ошибки (хотя даже ispell без проблем игнорирует их). Во-вторых, воспринимаются как ошибки слова, начинающиеся с заглавной буквы (программа предлагает такое слово набрать полностью заглавными буквами). И, что самое непонятное, очень много правильно набранных слов воспринимаются как ошибочные, причем первым же вариантом исправления предлагается в точности то же самое слово.

Впрочем, много и других ошибок. Именно поэтому я вернулся к версии 2.7 и, следуя рекомендациям В. Вагнера, попытался русифицировать ее.

Первым делом надо научить Ted вводить и отображать русские буквы. Для того, чтобы Ted правильно отображал шрифты в кодировке KOI8-R на экране, надо создать файл xfonts.dir в каталоге /usr/local/afm, где Ted держит метрики шрифтов (*.afm), поместив в него следующий текст

```
Courier-Bold -*-courier-bold-r-normal--*-*-*-*-*-*-*-*
Courier-BoldOblique -*-courier-bold-o-normal--*-*-*-*-*-*-*-*
Courier-Oblique -*-courier-medium-o-normal--*-*-*-*-*-*-*-*
Courier -*-courier-medium-r-normal--*-*-*-*-*-*-*-*
Helvetica-Bold -*-helvetica-bold-r-normal--*-*-*-*-*-*-*-*
Helvetica-BoldOblique -*-helvetica-bold-o-normal--*-*-*-*-*-*-*-*
Helvetica-Oblique -*-helvetica-medium-o-normal--*-*-*-*-*-*-*-*
Helvetica -*-helvetica-medium-r-normal--*-*-*-*-*-*-*-*
Times-Bold -*-times-bold-r-normal--*-*-*-*-*-*-*-*
Times-BoldItalic -*-times-bold-i-normal--*-*-*-*-*-*-*-*
Times-Italic -*-times-medium-i-normal--*-*-*-*-*-*-*-*
Times-Roman -*-times-medium-r-normal--*-*-*-*-*-*-*-*
```

После этого русские буквы вводятся и отображаются нормально. Однако проблема решена еще не полностью: после того как поработаешь с **Format tool** или любым диалоговым окном, содержащим строку ввода, Ted снова перестает вводить русские буквы. Решение этой проблемы, предложенное Иваном Паскалем, состоит в том, что создается пустой файл /usr/X11R6/lib/X11/locale/koi8-r/Compose и в файл /usr/X11R6/lib/X11/locale/compose.dir добавляется строчка:

```
koi8-r/Compose ru_RU.KOI8-R
```

Это решение у меня вполне сработало (за исключением того, что клавиша <Delete> перестала удалять символы и стала вставлять пробелы, так что для исправления ошибок приходится пользоваться клавишей <Backspace>).

Теперь рассмотрим вопрос о добавлении новых шрифтов. Вместе со стандартным дистрибутивом Ted поставляются только 4 шрифта: Courier, Helvetica, Symbol и Times. Если вы хотите оформлять свои документы более разнообразно, нужно добавить другие шрифты. В Ted можно использовать

любой шрифт из числа тех, которые видны у вас в графической оболочке, однако для этого нужно кое-что сделать. Чтобы узнать, какие шрифты у вас имеются, запустите программу xfontsel и выберите тот шрифт, который хотите добавить. Предположим, что это Bookman Light.

Для того чтобы Ted увидел новый шрифт, необходимо поместить метрику этого шрифта (метрика — это файл формата AFM) в каталог /usr/local/afm. Метрику можно либо найти готовую, либо создать самому. Ted весьма привередлив к метрикам шрифта. Как пишет В. Вагнер, Ted долгое время отказывался грузить шрифты от kapella, поскольку в AFM-файле отсутствуют упоминания нескольких символов, фигурирующих в Adobe Standard Encoding. Вместе с тем, работа с метриками шрифта — обязательное свойство более-менее продвинутой программы форматирования, поскольку той информации о шрифте, которую предоставляет X-сервер, недостаточно для качественной печати. Если вы попытаетесь работать с Ted используя метрики, не соответствующие вашим шрифтам, результат будет безобразный — слова будут наезжать друг на друга.

Готовые метрики шрифтов можно найти в коллекции Adobe base35, которая находится на FTP-сервере **ftp://ftp.adobe.com,** а инструкцию по их установке — в файле подсказки по редактору Ted.

В заключение замечу, что Ted позволяет осуществить русификацию интерфейса. Достаточно перевести на русский язык файл ресурсов, который находится в каталоге /usr/X11R6/lib/X11/app-defaults/ru и называется Ted. Если такого каталога нет, его нужно создать и складывать туда все русифицированные ресурсы. После этого, будучи запущенным с русской локалью, Ted будет выводить все меню и большую часть сообщений в диалоговых окнах по-русски.

12.7.4. Текстовый процессор AbiWord

AbiWord — это маленький, быстрый, бесплатный и легкий в освоении и использовании текстовый процессор для X. Этот пакет можно скачать из Интернета с сайта **www.abisource.com.** После запуска программы вы увидите основное окно программы (рис. 12.16), которое сильно напоминает знакомое всем окно MS Word.

Процессор AbiWord обладает всеми особенностями и функциями, которые присуши текстовым процессорам, включая проверку правописания, причем работающую непосредственно в процессе ввода текста.

AbiWord позволяет открывать для редактирования и просмотра файлы формата Microsoft Word (как MS Word 95, так и MS Word 97). Однако поскольку для преобразования форматов в нем используется вызов процедур пакета wv, который мы рассматривали выше, сказываются все недостатки последнего.

После редактирования файл можно сохранить в одном из следующих форматов:

- ❒ в собственном формате AbiWord (.abw);
- ❒ в текстовом формате (.txt);
- ❒ в формате RTF (.rtf);
- ❒ в формате HTML;
- O в формате UTF8;
- O в формате LaTex.

Таким образом, в принципе, через формат RTF можно организовать обмен файлами с пользователями MS Word.

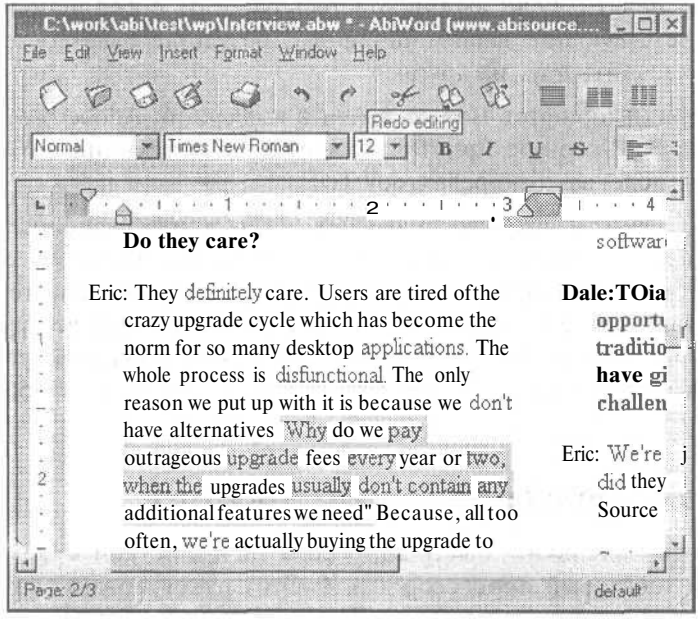


Рис. 12.16. Основное окно редактора AbiWord

К сожалению, пока что этот редактор еще находится в состоянии разработки (я смотрел версии 0.7.8 и 0.7.14). При обращении к некоторым командам меню вместо указанного (или ожидаемого) результата появляется сообщение о том, что данная команда пока не работает. В версии 0.7.8 это относилось к таким функциям, как создание нумерованных списков, стили, нумерация страниц и вставка символов. Вместо выполнения соответствующей функции появлялось сообщение: "пока не реализовано" ("not implemented yet"). И приглашение-просьба: если вы программист, присоединяйтесь к разра-

ботке, а если не программист, то будьте терпеливы. В версии 0.7.14 некоторые из этих недоработок уже устранены, однако часть ошибок все еще не исправлена. В частности, если в диалоговом окне открытия файла просто подсветить какой-то каталог и попытаться перейти в него нажатием клавиши <Enter>, происходит вызов нового экземпляра программы. Правда, текущим каталогом для этого нового экземпляра оказывается тот каталог, куда мы и хотели попасть, но напрасное открытие новых окон раздражает (да и ресурсы зря расходуются). Если же осуществлять переход в каталог двойным щелчком мыши, то нового окна не открывается. Есть и другие ошибки, причем при некоторых редактор просто закрывается, без предупреждений и сохранения результатов предыдущей работы. Так что считать это продукт готовым к применению пока что рановато.

Однако в целом он производит очень приятное впечатление и версия 0.7.14 кажется гораздо более работоспособной, чем 0.7.8. Это свидетельствует о том, что работа над программой продолжается. Поэтому давайте наберемся терпения в надежде, что появление полноценного продукта, реализующего все заявленные в меню возможности, не за горами.

Между прочим, по адресу **http://www.hippo.ru/~hw/abiword/** можно найти русифицированную версию AbiWord.

12.7.5. Текстовый процессор KWord

Текстовый процессор KWord входит в состав офисного пакета KOffice. Однако, в отличие от StarOffice, для его запуска не требуется запуск какой-то общей оболочки или общего офисного Desktop. После запуска программы (проще всего это сделать из главного меню оболочки KDE, выбрав команду **Офис | Текстовый процессор),** появляется непритязательное серое окно (рис. 12.17), на фоне которого тут же возникает окно выбора типа открываемого документа (рис. 12.18).

В этом окне предлагается либо создать новый документ на основе одного из шаблонов, либо открыть существующий документ. Здесь же имеется возможность просмотреть список тех документов, с которыми вы работали в предыдущих сеансах, и быстро вызвать один из них.

Как только вы вызвали на редактирование какой-то документ или открыли новый, вид окна кардинальным образом изменяется (рис. 12.19). Появляются и новые команды меню, и панели инструментов, и линейки форматирования страницы.

Работа с новым текстовым редактором обычно начинается с создания нового документа, поэтому и давайте мы создадим новый документ и попытаемся вводить текст. Для начала надо выбрать шрифт, естественно, русифицированный. В KWord это делается через команду меню **Формат | Шрифт.**

После выбора этой команды меню появится окно выбора шрифта (рис. 12.20), в котором этот самый выбор и осуществляем.

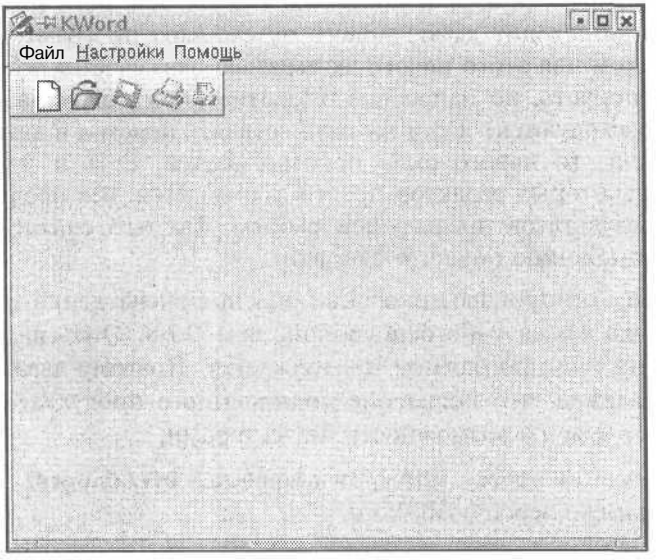


Рис. 12.17. Стартовое окно программы KWord

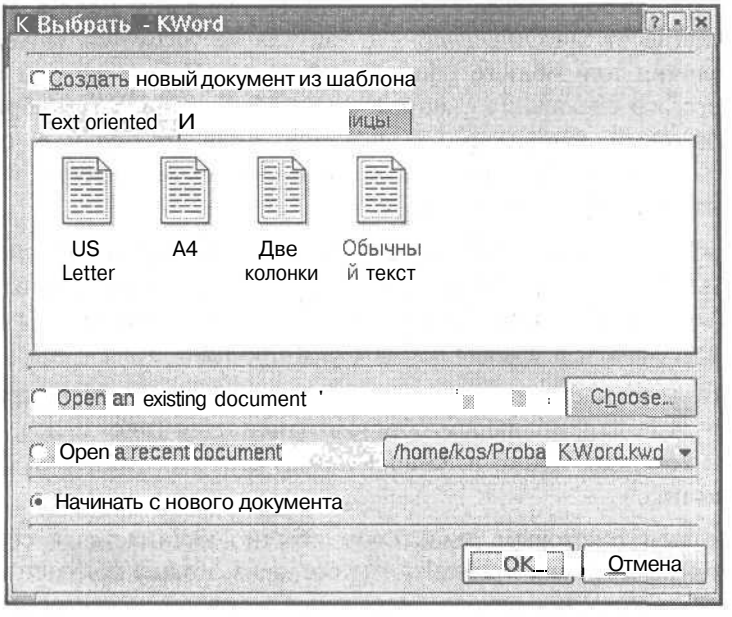


Рис. 12.18. Окно выбора типа открываемого документа

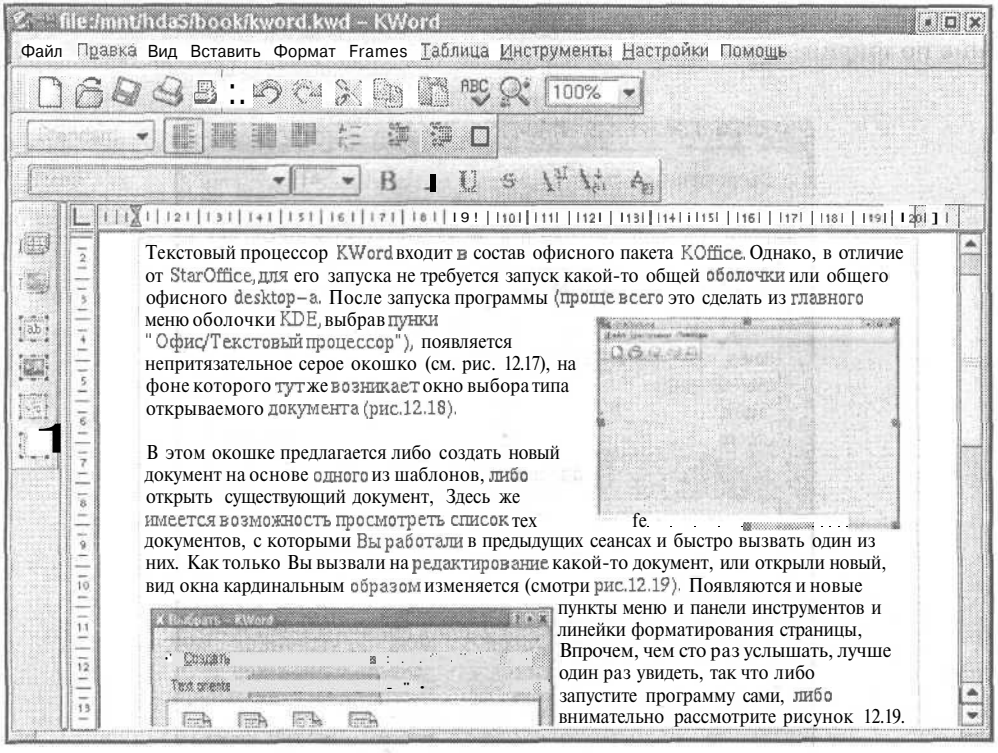


Рис. 12.19. Основное окно программы KWord

В русифицированной версии Linux среди этих шрифтов найдутся и шрифты в кодировках KOI8-R и CP-1251. Выбираем один из них и начинаем вводить текст. Кстати, основные характеристики шрифта можно выбрать и не заглядывая в меню, поскольку на панели инструментов (см. рис. 12.20) имеются выпадающие списки для наименования шрифта, размера символов, значки для задания полужирного, подчеркнутого, зачеркнутого шрифта, а также для изменения цвета символов и превращения части текста в верхние и нижние индексы. Впрочем, значки на панели инструментов можно добавлять или, наоборот, убирать. Но о настройках панелей инструментов поговорим позже, а пока вернемся к набору текста.

Как только введен первый абзац, возникает желание придать ему благообразный вид. Что и сделаем с помощью команды меню **Формат | Абзац.** Появится окно с пятью вкладками, изображенное на рис. 12.21. На первой вкладке можно задать отступы для строк абзаца (отдельно для первой строки и остальных), межстрочные интервалы и интервалы между двумя последовательными абзацами. Величина всех интервалов и отступов задается в миллиметрах. На второй вкладке задается положение строк данного абзаца на

странице: смещение их влево, вправо, по центру или равномерное растяжение по ширине страницы.

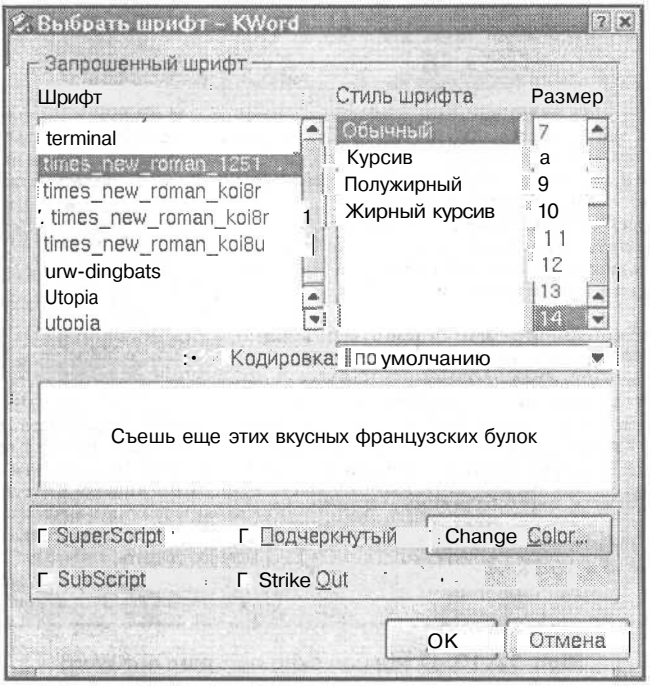


Рис. 12.20. Выбор шрифта в KWord

Вкладка **Рамка** позволяет заключить данный абзац в рамку (например, чтобы как-то выделить данный абзац из общего текста). Две последних вкладки служат для задания формата списков и установки позиций табуляции, но об этом поговорим позже (если выдастся случай).

Следующая команда в меню **Формат** позволяет задать формат целой страницы. После выбора команды **Формат | Страница** появляется окно с тремя вкладками. Первая из них служит для задания ориентации страницы (книга или альбом), размера страницы (например, A4), ширины полей. Во второй вкладке можно задать размещение текста в несколько колонок (столбцов). И третья вкладка определяет свойства верхнего и нижнего колонтитулов. Но нам пока еще не до таких тонкостей, давайте закончим набор первой страницы текста и посмотрим, как же готовая страница будет выглядеть на экране.

Только вначале отметим, что так же, как и для выбора шрифта, для форматирования абзаца необязательно пользоваться соответствующей командой меню. Во-первых, можно использовать панель инструментов **Формат, на**

которой можно выбрать один из стандартных стилей абзаца (правда, поначалу список выбора стиля небогат, но можно создать собственный стиль через команду меню **Формат | Менеджер стилей),** положение строк на странице и сдвинуть абзац вправо. Кроме панели инструментов для форматирования абзаца можно воспользоваться линейками слева и вверху окна ввода текста. Эти линейки позволяют задать поля страницы и отступы абзаца.

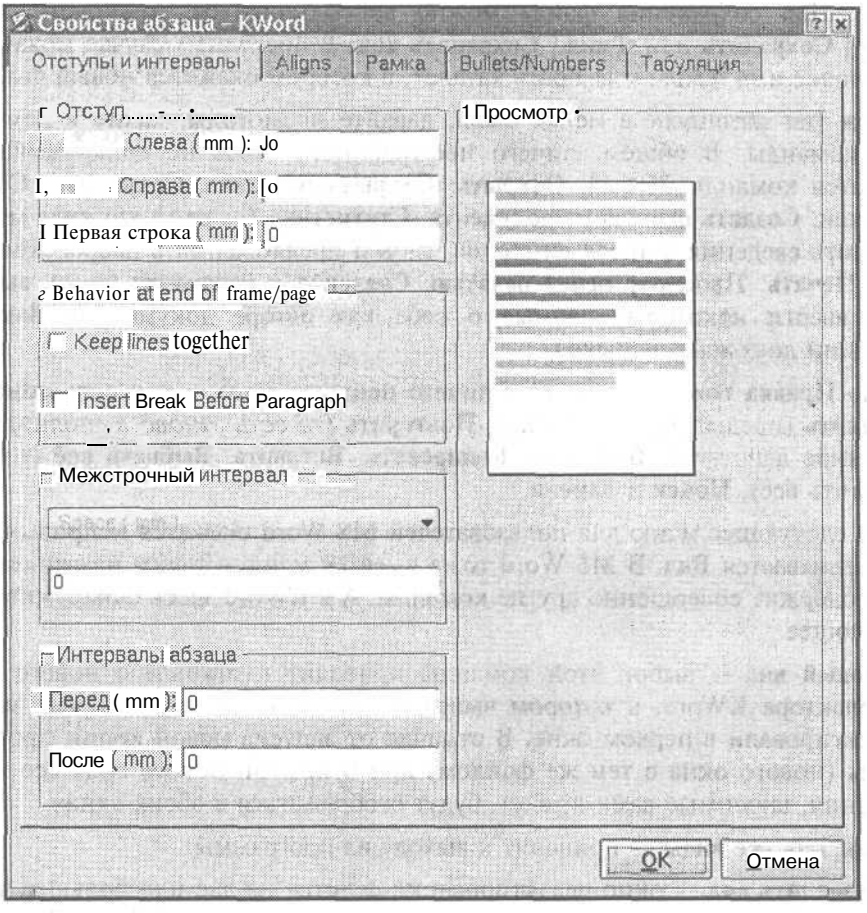


Рис. 12.21. Окно форматирования абзаца

Итак, если вы закончили ввод теста или хотя бы набрали целую страницу, можно посмотреть, как эта страница будет выглядеть отпечатанной. Для этого надо воспользоваться комадой меню **Файл | Предварительный просмотр** или соответствующим значком на панели инструментов **Файл.** В отличие от MS Word для предварительного просмотра страницы вызывается

отдельная программа — "Просмотрщик PS/PDF" (о ней было сказано несколько слов выше). Если что-то в изображении страницы вам не понравится, можно вернуться к форматированию страницы и абзацев, чтобы после внесения изменений снова перейти к предварительному просмотру. Только, чтобы увидеть изменения, просмотрщик придется открыть снова, так что текущий экземпляр лучше закрывать.

Чтобы не потерять нечаянно результаты работы, давайте сохраним файл на диске. Для сохранения файла можно воспользоваться командами меню **Файл | Сохранить** или **Файл | Сохранить как.** В последнем случае можно задать новое имя файла и выбрать каталог, в котором окажется новый файл.

Раз уж мы заглянули в меню **Файл,** давайте посмотрим, какие в нем еще есть команды. В общем, ничего неожиданного здесь не обнаруживается. Имеются команды **Новый, Открыть, Открыть недавние, Сохранить, Сохранить как, Создать шаблон из документа, Статистика** (по этой команде можно получить сведения о числе символов, слов и предложений в набранном тексте), **Печать, Просмотр перед печатью, Сведения о документе** (здесь вы можете ввести некоторые данные о себе как авторе документа), Закрыть (текущий документ) и **Выход.**

Меню **Правка** тоже не содержит ничего неожиданного: в нем есть команды **Отменить** (предыдущее действие), **Повторить** (то есть заново выполнить отмененное действие), **Вырезать, Копировать, Вставить, Выбрать всё** (то есть выделить все), **Поиск и Замена.**

А вот следующее меню для пользователей MS Word окажется непривычным. Оно называется Вид. В MS Word тоже имеется меню с таким названием, но оно содержит совершенно другие команды. А в KWord здесь обнаруживается следующее.

- **Новый вид** — выбор этой команды приводит к открытию нового окна редактора KWord, в котором выведен тот же документ, который вы редактировали в первом окне. В отличие от запуска новой копии программы (нового окна с тем же файлом) при открытии нового вида все изменения, вносимые вами в текст, будут отображаться в обеих окнах.
- **Закрыть все виды** — приводит к выходу из программы.
- **Разделить вид** — окно редактирования делится на две или большее число панелей, в каждой из которых выводится редактируемый файл. То же самое, что и **Новый вид,** только без открытия нового окна. В каждом виде можно перемещаться по тексту независимо.
- **Удалить вид** — закрывает текущий вид, т. е. ту панель или окно, в котором в данный момент находится курсор.
- **Расположение разделителя** — позволяет расположить панели видов вертикально или горизонтально.

Далее идут две команды-переключателя: **Page mode и Preview mode. В** режиме **Page mode** на панели редактирования отображается одна страница текста, как вы видели на рис. 12.19. В режиме **Preview mode** на экран выводятся сразу все страницы документа. Линейки прокрутки позволяют перемещаться по страницам. Это позволяет посмотреть, как будут располагаться рисунки, таблицы и т. д.

Далее идет неактивизированная пока (я работал с версией 1.1 post-beta программы KWord) команда **Непечатаемые символы,** после которого следуют еще три переключателя: **Рамка врезки, Верхний колонтитул и Footer** (в смысле "нижний колонтитул"). Эти три переключателя включают (или, соответственно, выключают) отображение соответствующих элементов оформления страницы (впрочем, **Рамка врезки** не является элементом оформления страницы, она видна только на экране).

Назначение последней команды меню **Вид** очевидно из его названия — **Масштаб.** Впрочем, вместо этой команды удобнее пользоваться выпадающим списком на панели инструментов. Тем более, что задать произвольное значение масштаба невозможно ни тем, ни другим способом, можно только выбрать одно из предустановленных значений.

Раз уж речь зашла о колонтитулах, стоит упомянуть, что на вкладке **Колонтитулы,** которая становится доступной при выборе команды меню **Формат | Страница,** можно задать по три параметра для верхнего и нижнего колонтитулов (рис. 12.22).

В одном из колонтитулов обычно выводят номер текущей страницы. В KWord это можно сделать из следующего меню — **Вставить.** Надо выбрать команду **Переменная** и из появившегося списка выбрать **Page Number.**

А вообще через меню **Вставить** можно вставить картинку, таблицу, формулу, специальный символ, разрыв страницы, содержание, переменную, выражение **(Expression),** текстовый фрейм, фрейм с картинкой, фрейм с объектом **(Object Frame).** Правда с картинками работать достаточно сложно. Щелчок мышкой по картинке не приводит к появлению меню или каких-то символов, служащих для изменения размера. Надо умудриться щелкнуть по рамке картинки в левом верхнем углу, где имеется маленькая область, щелчок по которой приводит к появлению маленьких квадратиков в углах изображения и центрах сторон, уцепившись за которые, можно изменять размер картинки.

Несколько успешнее удается вставлять фрейм с картинкой. Вызвать меню управления фреймом тоже нелегко — надо попасть в ту же маленькую область на границе фрейма правой кнопкой мыши, чтобы вызвать это меню, но все же уже можно что-то сделать. После вызова меню появляется окно, избраженное на рис. 12.23.

В этом окне можно задать взаимоотношения основного текста и врезки (рис. 12.23) и размер врезки, который для фреймов с картинками определяет

и размеры изображения. Однако попытки изменения размеров картинок очень часто приводят к тому, что программа "сваливается".

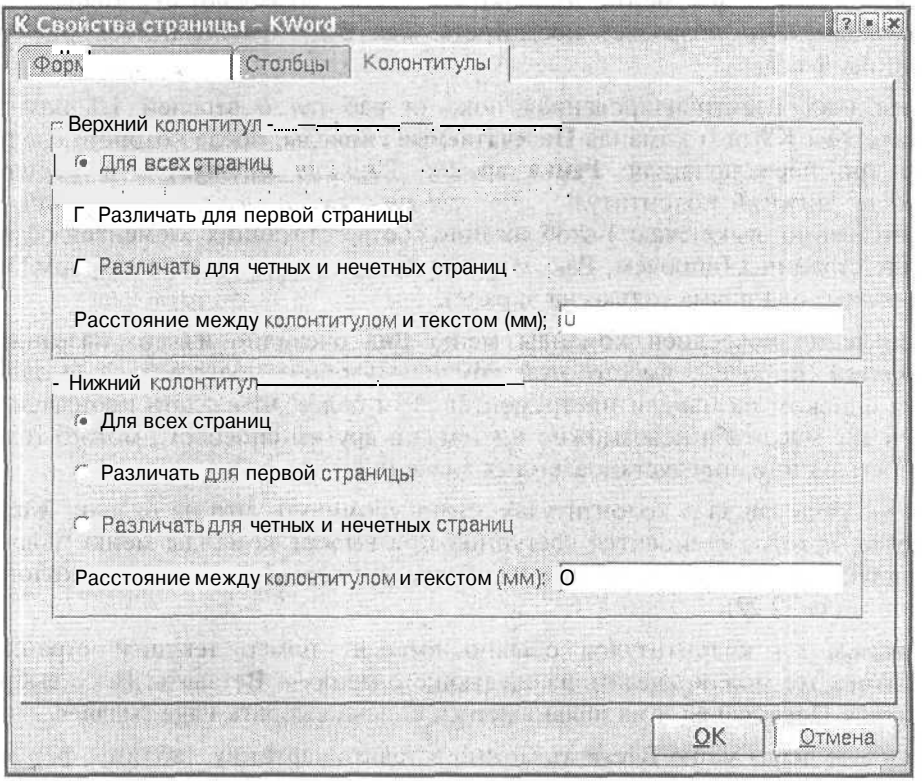


Рис. 12.22. Параметры верхнего и нижнего колонтитулов

Вставка таблиц работает, однако таблицы получаются пока не слишком красивые. Вставка формул — это пока, вообще, что-то неприменимое. Команда **Expression** позволяет вставить в текст несколько фиксированных фраз (привет, прощай, адрес e-mail и т. п.). В качестве вставляемых переменных могут использоваться номер страницы, число страниц в документе, имя файла, имя текущего каталога, текущие дата и время (причем, как я понял, либо время вставки этого параметра, либо время последнего редактирования документа), те данные, которые вы ввели по команде **Файл | Сведения о документе,** а также любые определенные вами значения. По поводу вставки переменных тоже можно высказать некоторые претензии. В частности, после первого сохранения файла вставка времени и даты сохраняется, а после второго сохранения — пропадает.

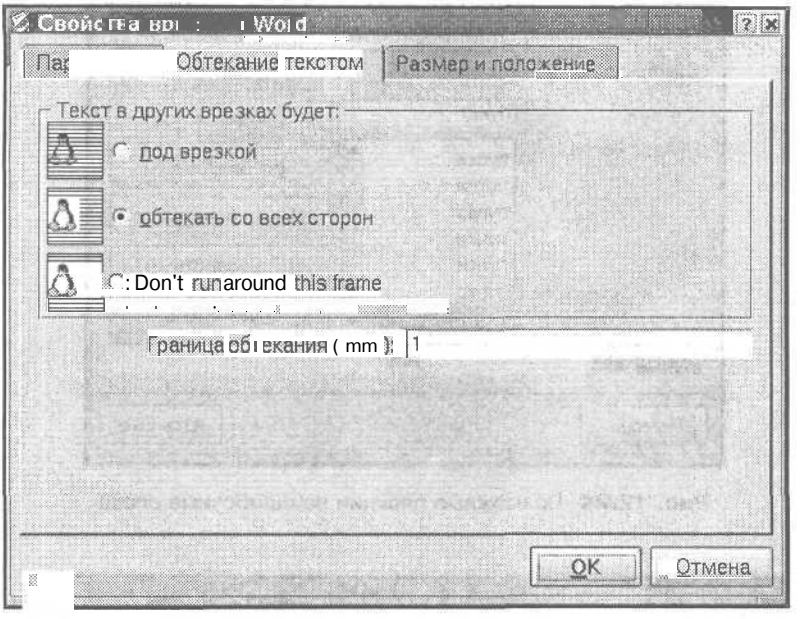


Рис. 12.23. Окно настройки фрейма (врезки)

В меню **Инструменты** мы видим три команды: **Проверка правописания, Пользовательские переменные и Автозамена.** Проверка правописания представляет собой графическую оболочку к встроенному вызову программы ispell. При обнаружении незнакомого слова оно выделяется и выводится окно, изображенное на рис. 12.24. Можно заменить слово одним из предлагаемых вариантов или пропустить его ("игнорировать" ошибку). Можно также напрямую поправить слово в строке предлагаемой замены, после чего нажать кнопку **Заменить.**

Команда **Пользовательские переменные** служит для определения собственных переменных, а команда **Автозамена** — для задания правил автоматического исправления наиболее характерных и часто встречающихся ошибок, типа двух заглавных букв в начале слова. Можно изменить правила замены или задать собственные.

Через меню **Настройки** можно включить или отключить отдельные панели инструментов, настроить вид этих панелей и привязки "горячих" клавиш, а также произвести некоторые настройки самой программы.

Ну и, наконец, последнее вложенное меню, имеющее традиционную для KDE структуру: помощь по программе, сведения о KWord и сведения о KDE. Помощь по программе пока (в версии 1.1) отсутствует, а сведения об авторах видны на рис. 12.25. Во вкладке **Перевод** сообщается, что перевод сделан Андреем Черепановым.

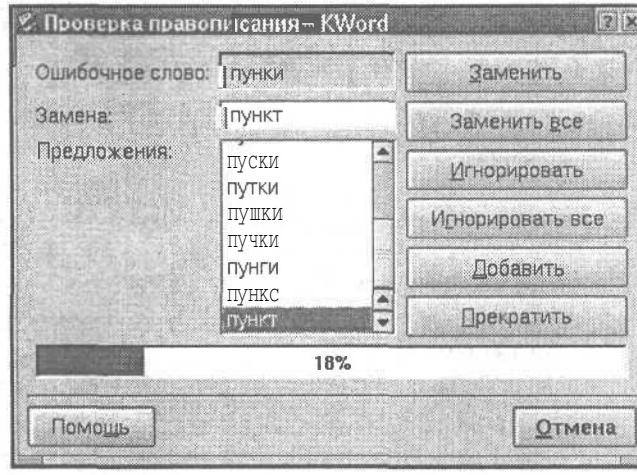


Рис. 12.24. Возможные реакции на ошибочные слова

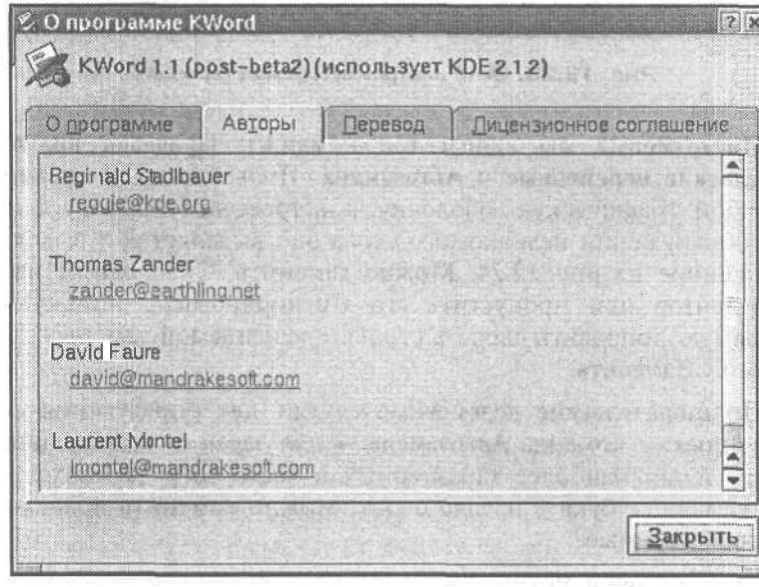


Рис. 12.25. Сведения о разработчиках программы

На этом описание текстового процессора KWord можно считать завершенным. К сожалению, пока что и его нельзя признать полностью функциональным тестовым процессором. Главные претензии — работа с рисунками и таблицами. Но с русским языком процессор работает вполне нормально и даже позволяет выбрать кодировку символов. Как это сделать в MS Word

(если это вообще возможно), мне неизвестно. Так что для создания не очень сложных документов KWord вполне пригоден.

12.7.6. Текстовые процессоры StarWriter и OpenOffice.org Writer

Текстовый процессор StarWriter входит в состав офисного пакета StarOffice фирмы Sun. Домашняя страница: **http://www.sun.com/products/staroffice.** Скачать с домашней страницы можно бесплатно, CD-ROM стоит $ 39,95 (в США).

Первоначально пакет StarOffice был разработан немецкой фирмой StarDivision, которая была основана в середине 1980-х годов. Летом 1999 г. фирма StarDivision была куплена вместе с ее разработкой фирмой Sun. Бета-версия StarOffice 5.1a, выпущенная уже от имени Sun, имела мало принципиальных отличий от тех версий, которые выпускались первоначальным разработчиком. Конечно, появилась эмблема Sun и была значительно (на мой взгляд) облегчена процедура установки, которая стала очень напоминать процедуры установки Windows-продуктов: ответь на несколько вопросов, а дальше сиди и читай сменяющие друг друга рекламные сообщения. Возможно, это явилось следствием того, что пакет StarOffice выпущен также и в варианте для Windows.

В июне 2000 г. Sun выпустила версию 5.2 пакета, причем StarWriter позиционировался как альтернатива Microsoft Word, удовлетворяющая самым придирчивым требованиям. И действительно, StarWriter имеет довольно большой набор файловых фильтров и обладает довольно большим сходством с Microsoft Word по набору основных операций, так что переход от MS Word к StarWriter должен происходить для пользователя достаточно просто. Более того, StarWriter обеспечивает большинство возможностей, которые имеет MS Word. С его помощью можно создавать (в режиме WYSIWYG) документы высокого качества. Легко можно получить текст, разбитый на несколько колонок, вставить в текст рисунки и таблицы, врезки, задать формат абзацев, ширину полей и т. п. Судя по описаниям, имеются возможности, присущие профессиональным издательским системам, такие как управление величиной расстояния между соседними буквами (кернинг), задание стилей для отдельных фрагментов текста (заголовков, цитат и т. п.). С помощью StarWriter можно создавать конверты, наклейки, письма.

Если говорить о том, в каких случаях стоит использовать StarWriter в качестве текстового процессора, то в первую очередь надо упомянуть тот факт, что он неотделим от остальных частей пакета StarOffice. Пакет StarOffice сам по себе представляет собой интегрированную графическую среду, из которой можно запускать все другие приложения, так что он достаточно требователен к ресурсам. Поэтому вряд ли целесообразно использовать StarWriter

в таких случаях, когда надо создать небольшую служебную записку или коротенькое электронное письмо (если, конечно, вы не используете весь пакет StarOffice в качестве своей рабочей среды).

К сожалению, даже версия 5.2 StarWriter не является полностью русифицированной и недостаточно корректно обрабатывает документы, созданные с помощью MS Word, поэтому после недолгих экспериментов с этой программой я отказался от ее использования. Подробнее о пакете StarOffice в целом вы можете прочитать в книге А. Федорчука "Офис, графика, Web и Linux" *(см. [П1. 6] приложения).*

13 октября 2000 года Sun открыла исходные коды пакета StarOffice. Таким образом появился проект OpenOffice.org (обратите внимание на приставку .org — как говорится на Web-странице проекта, именно так следует правильно именовать данный проект, поскольку "OpenOffice" является зарегистрированной торговой маркой какого-то другого продукта).

Исходные коды OpenOffice.org основаны на технологии, которая первоначально была разработана Sun Microsystems для будущих версий пакета StarOffice (TM). В отличие от StarOffice проект OpenOffice.org не содержит интегрирующей оболочки — архитектура пакета предполагает отдельное использование входящих в пакет приложений. Пакет написан на языке C++ и включает в себя все основные офисные приложения, такие как текстовый процессор, электронную таблицу, программу управления презентациями, программу для работы с графикой, средства создания диаграмм и редактирования формул. Но OpenOffice.org не включает в себя клиента электронной почты, календаря и браузера.

Среди других новшеств можно отметить использование формата файлов, основанного на стандарте XML, что облегчает взаимодействие с другими продуктами (и, следовательно, с другими пользователями компьютеров), а также обеспечивает ясные перспективы развития продукта в будущем. В состав продукта включены конверторы для других распространенных файловых форматов, включая форматы всех версий Microsoft Office, и улучшена поддержка азиатских языков (китайского, японского, корейского).

В названиях отдельных исполняемых файлов пакета OpenOffice.org прослеживается его происхождение от StarOffice. Весь пакет вызывается командой "soffice." Отдельные программы пакетов вызываются командами, перечисленными в табл. 12.3.

Таблица 12.3. *Названия программ пакета OpenOffice.org*

Программа	Команда
Текстовый процессор	swriter
Менеджер презентаций	simpress

Таблица 12.3 (окончание)

Программа	Команда
Электронная таблица	scale
Программа работы с графикой	sdraw
Программа обработки формул	smath

В настоящее время локализация пакета OpenOffice.org выполнена для 27 языков. Пакет стабильно работает в операционных системах Solaris, Linux (включая PPC Linux) и Windows. Перенос его на платформы FreeBSD, IRIX и Mac OS X находится в стадии завершения.

Это все были общие сведения. Пора уже рассказать немного о первых впечатлениях от работы с самой программой. Работал я с версией 641b, которая входит в состав дистрибутива Red Hat Linux 7.2 Cyrillic Edition от Урбан-софт.

Установка программы производится достаточно просто — разворачиваем в одном из доступных каталогов архив и запускаем программу установки setup из полученного каталога install. Для начала можно выбрать стандартный вариант установки (есть еще варианты Minimal и Custom). Установка прошла у меня без всяких проблем, после чего в моем домашнем каталоге появился подкаталог OpenOffice.org641. В этом каталоге имеется символическая ссылка с именем soffice, которая позволяет запустить программу. Запускается по этой ссылке программа swriter, т. е. текстовый процессор. Остальные программы надо запускать по их собственным именам.

Внешний вид окна программы, конечно, несколько непривычен для пользователя MS Office. Кроме обычных полосок меню и панелей инструментов в верхней части окна имеется еще вертикальная панель инструментов слева. Но, в общем-то, все элементы окна достаточно понятны.

К сожалению, я познакомился с этой программой на последнем этапе работы над данной книгой, и у меня нет возможности дать подробное описание этого текстового процессора. Отмечу только самое существенное.

Для начала я, естественно попробовал просто вводить произвольный текст на русском языке. Никаких проблем с набором не возникло, но буквы почему-то наезжают друг на друга (проблема, о которой уже сообщалось применительно к StarOffice). Попытки изменить шрифт показали, что наиболее приемлемым является Helvetica. В целом текст стал читабельным, но полностью эффект наезжающих букв не устранился. Вставка картинки (снимка окна программы) и таблицы прошли без проблем (при этом при создании таблицы сразу сформировалась ее шапка).

Далее я попытался сохранить набранный текст в формате MS Word. Для этого выбрал команду **File | Save** as и в выпадающем списке **File Type — ва-**

риант Microsoft Word 97/2000/2002, ввел имя файла и щелкнул по кнопке Save. Перезагрузился в Windows и открыл сохраненный файл. Вначале картинка не появилась, однако после того, как в меню Вид я выбрал команду **Разметка страницы,** все встало на свои места. С таблицей тоже все нормально. Так что сохранение файла в формате Word работает.

Обратное действие — открытие в OpenOffice.org файла, созданного в MS Word — также прошло успешно. Правда, первоначально некоторые строки вывелись целиком квадратиками вместо букв. Но, как оказалось, это следствие разницы в наборах шрифтов (или стилей) — после выделения соответствующей части текста и изменения шрифта текст становится вполне читаемым.

Между прочим, приведенное краткое описание пакета OpenOffice.org набрано в программе swriter.

Не имея сейчас возможности заниматься подробным описанием даже программы swriter, не говоря уже об остальных программах пакета, хочу только сформулировать первый вывод. Самое важное, на мой взгляд, то, что впервые появилась возможность под Linux работать с файлами, созданными в Microsoft Office. Тем самым уже нет необходимости запускать эмуляторы для того, чтобы просмотреть файл формата MS Word, полученный как вложение по электронной почте от любого из приверженцев Microsoft. И не беда, что пока есть претензии к качеству вывода изображения на экран. Все открытые продукты постоянно совершенствуются и развиваются, и недостатки будут со временем устранены. Но появление OpenOffice.org разрывает последнюю нить, привязывавшую приверженцев Linux к продукции Microsoft, и позволяет им полностью перейти в среду Linux.

12.8. Словари и переводчики

Лично я отрицательно отношусь к программам, которые предназначены для автоматического перевода текста с одного языка на другой. Как бы ни старались разработчики таких программ, но добиться сколь-нибудь приемлемого качества перевода им не удается, и, думаю, в ближайшем будущем не удастся. Созданные такими программами переводы могут просто до абсурда изменить смысл написанного. Поэтому сам я такими программами не пользуюсь и вам не советую.

Другое дело словари. Без словаря не могут, как мне кажется, обойтись даже профессиональные переводчики. Даже на родном языке мы не всегда понимаем значение некоторых слов, так что вынуждены пользоваться толковыми словарями и энциклопедиями. Так что иметь электронный словарь просто необходимо. И такие словари существуют, в частности для Linux. Естественно, что большинству пользователей в первую очередь требуются англо-русские словари, чтобы понимать пояснения к программам, которые в абсолютном большинстве случаев даются на английском.

Я не проводил сравнительного анализа электронных словарей, результаты которого мог бы вам здесь предложить. Первой мне попалась программа slovo, разработанная Д. Анисимовым, к которой я уже привык, и она меня вполне устраивает. Эта программа использует широко известный англо-русский словарь Мюллера (хотя могут быть подключены и другие словари) и дает транскрипцию английских слов, так что в обычной практике ее вполне достаточно.

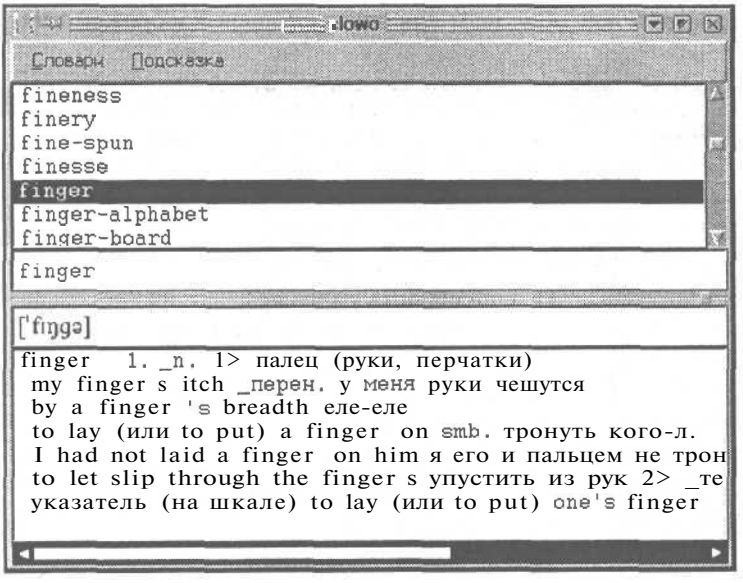


Рис. 12.26. Окно программы slovo

Некоторые затруднения у вас может вызвать получение транскрипции, поскольку для корректной работы этой части программы необходимо установить специальные шрифты (это естественно, обычные шрифты не содержат знаков транскрипции). Однако в руководстве к программе, которое прилагается и в русскоязычном варианте, необходимые действия подробно описаны, так что объяснять это здесь нет нужды. По крайней мере, у меня все получилось, думаю, что и у вас тоже все будет нормально, разве что придется приложить небольшие усилия.

После запуска программы появляется окно, изображенное на рис. 12.26, только первоначально три нижних поля пусты.

Вы можете выбрать требующее перевода слово из списка в верхнем поле, либо прямо ввести его в строке ввода (второе сверху поле). В процессе ввода происходит перемещение подсветки в списке слов на слово, ближайшее к введенной вами комбинации символов. Нажатие на клавишу <Enter> при-

водит к появлению перевода в нижнем поле окна программы и транскрипции в поле, расположенном над переводом. Команда **Словари** позволяет сменить словарь (не все словари позволяют выводить транскрипцию, в моем случае этим свойством обладал только словарь Мюллера).

Надеюсь, что приведенных пояснений вполне достаточно для того, чтобы вы смогли результативно пользоваться этой программой.

Еще один распространенный англо-русский словарь встретился мне в составе дистрибутива ALT Linux Junior 1.0. Это словарь mova Евгения Цимбалюка. Окно графической оболочки для этого словаря изображено на рис. 12.27.

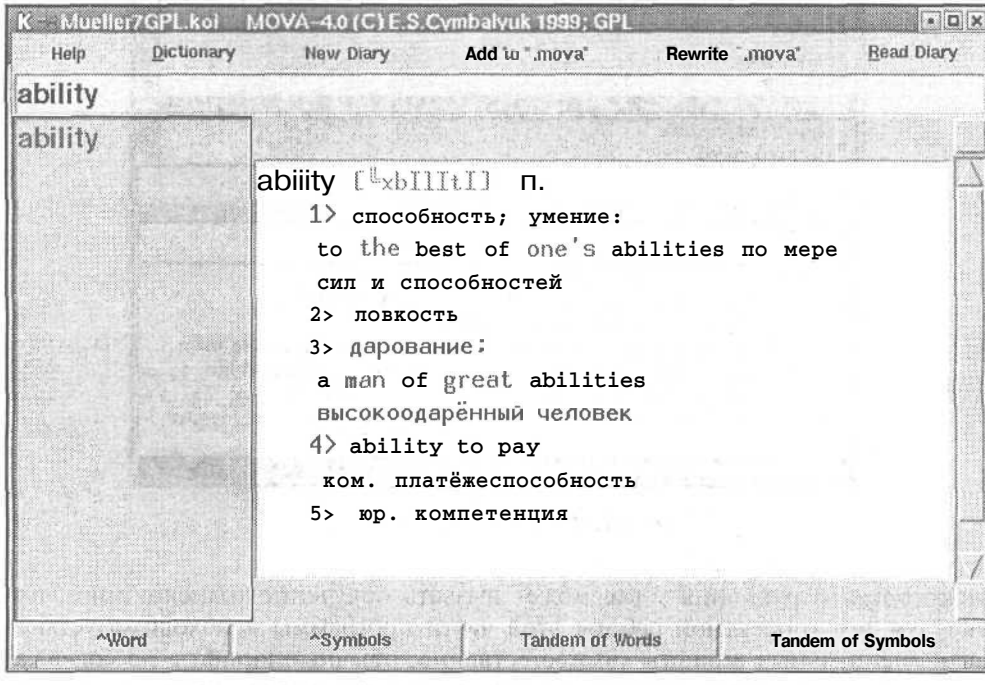


Рис. 12.27. Окно графической оболочки словаря mova

В строке ввода (она выделена желтым цветом) вводим английское слово и нажимаем кнопку **^Word**. Можно просто скопировать неизвестное слово в строку ввода. Инструкции по использованию программы даны в прилагаемом к ней файле Readme_mova_koi.txt, причем на русском языке, так что здесь подробных пояснений давать не требуется.

Глава 13

Выход в локальные сети

Хотя я рассматриваю в данной книге только работу на персональном компьютере, компьютер этот необязательно изолирован от других. Вполне возможно, что персональный компьютер на вашем рабочем месте подключен к локальной сети. Более того, локальные сети становятся реальностью и в домашних условиях.

В данной главе будут рассмотрены вопросы подключения к уже существующей локальной сети. Вопросы создания сети "на голом месте" выходят за рамки настоящей книги. Впрочем, их будет проще понять (и изложить) после того, как будут рассмотрены вопросы подключения отдельного компьютера в сеть.

Нужно сказать для начала, что на русском языке имеется (по крайней мере в электронном варианте, доступном в Интернете) несколько замечательных руководств, в которых подробно разбираются все аспекты подключения вашего компьютера как к локальной сети, так и к Интернету. Самым лучшим руководством является, безусловно, книга О. Кирха *(см. [П1.2] приложения)*. Приводимое ниже описание действий, выполняемых при подключении к локальной сети, представляет собой конспект этих материалов.

13.1. Подготовка к выходу в сеть

13.1.1. Драйверы сетевых устройств в ядре

Естественно, что для того, чтобы работать с локальной сетью, необходимо иметь сетевую карту (плату) и подключение к сети. Я не буду объяснять, как физически подключиться к сети, предполагая, что вы уже установили сетевую карту в ваш компьютер и соединили его (кабелем или витой парой) с существующей сетью.

Заметим, что в каталоге /dev нет специального файла для сетевой карты. В Linux сетевые устройства создаются динамически, и поэтому не требуют наличия соответствующих файлов в каталоге /dev.

Прежде, чем пытаться подключаться к сети, вы должны убедиться, что установленное в вашей системе ядро скомпилировано с поддержкой сетевых возможностей. О. Кирх утверждает, что признаком этого является наличие

каталога /proc/net. Но можно посмотреть и протокол загрузки системы (файл /var/log/dmesg), в котором должны найтись примерно такие строки:

```
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 8192 bind 8192)
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NET4: Ethernet Bridge 008 for NET4.0
```

Далее нужно убедиться, что в состав ядра включен драйвер для вашей сетевой карты. Вообще-то, ядра, включаемые в стандартные дистрибутивы, обеспечивают поддержку большинства распространенных сетевых плат (что, конечно, увеличивает объем ядра). Так что с очень высокой вероятностью нужный драйвер входит в ядро. В процессе загрузки ядра выполняется процедура автоматического обнаружения сетевой карты. Если такое обнаружение было успешным, то в файле /var/log/dmesg вы найдете соответствующие сообщения. У меня они имеют вид:

```
eth0: 3Com PCI 3c905C Tornado at 0x4000, 00:01:02:b4:6c:65, IRQ 9
    product code 'DU' rev 00.11 date 09-02-00
    8K byte-wide RAM 5:3 Rx:Tx split, autoselect/Autonegotiate interface.
    MII transceiver found at address 24, status 782d.
    Enabling bus-master transmits and whole-frame receives.
```

Если карта не обнаружена, то вам придется перекомпилировать ядро (или поменять карту). Перекомпиляция ядра может иметь смысл и в том случае, если вы хотите удалить из ядра ненужные драйверы устройств, которые вы не используете. Но все же в большинстве случаев стандартное ядро успешно решает задачи поддержки сетевых возможностей, так что на вопросе перекомпиляции ядра сейчас мы останавливаться не будем. Рассмотрим только вопрос о динамическом подключении драйвера сетевой карты.

13.1.2. Динамическое подключение драйверов

Для динамического подключения драйвера надо подгрузить модуль ядра, отвечающий за взаимодействие с данным сетевым устройством (например, сетевой картой) и передать ему параметры устройства. Сделать это можно с помощью команды insmod, вызов которой осуществляется следующим образом:

```
[root]# /sbin/insmod [-fkmpsxXv] [-o имя_устройства] файл_драйвера
```

Например, для сетевой карты можно выполнить команду следующего вида:

```
[root]# /sbin/insmod -o eth0 /lib/modules/2.2.x/net/net.o
```

Здесь `2.2.x` надо заменить на версию установленного у вас ядра, а вместо `net.o` надо подставить имя модуля, соответствующего вашей сетевой плате. Поскольку у меня была плата 3Com, я выбрал `3c509.o` (посмотрите перечень в подкаталоге /lib/modules/2.2.x/net/).

Замечание

В последних версиях Linux вместо `insmod` используется `modprobe`.

Ядро именует драйверы Ethernet как eth0, `eth1` и т. д., так что для подключения, например, второй сетевой карты надо в этом примере eth0 заменить на `eth1`.

Кстати, ядро не может автоматически определить наличие двух сетевых адаптеров. В этом случае вам придется произвести некоторые дополнительные настройки. Однако обсуждение этого вопроса выходит за рамки данной книги, ищите ответы в книге О. Кирха и других источниках.

В некоторых случаях в команде требуется дополнительно задать номер порта и номер используемого прерывания, а также некоторые другие опции, но подробнее об этом см. на странице man insmod.

Аналогично, для подключения модуля, обеспечивающего работу с модемом по протоколу PPP, надо дать команду

```
[root]# /sbin/insmod /lib/modules/2.2.11/net/ppp.o
```

(драйвер должен существовать и располагаться в указанном каталоге).

13.1.3. Получение сетевого адреса и установка ПО

Поскольку вы собираетесь устанавливать машину с Linux в уже существующую сеть, то следующим вашим шагом при подключении к сети должно стать обращение к администратору сети за получением сетевого адреса. Точнее, вы должны получить следующую информацию:

- IP-адрес вашего компьютера;
- IP-адрес сети;
- широковещательный IP-адрес;
- имя домена, в который будет включен ваш компьютер;
- маску подсети;
- IP-адрес маршрутизатора (router);
- IP-адрес сервера имен (DNS-сервера).

Заодно согласуйте с администратором сетевое имя, которое вы выбрали для своего компьютера (чтобы избежать его совпадения с именем какого-то из уже включенных в сеть компьютеров).

Далее необходимо проверить, что у вас установлено программное обеспечение, необходимое для подключения к сети, а именно, пакет net-tools. Для проверки воспользуйтесь командой

```
[root]# rpm -qa I grep net
```

или менеджером пакетов (если вы работаете с KDE). Если пакет не установлен, то необходимо установить его с вашего дистрибутивного диска.

13.2. Настройка сетевых интерфейсов

Интерфейсом с точки зрения ОС является устройство, через которое система получает и передает IP-пакеты. Роль интерфейса локальной сети может выполнять одно (или несколько) из следующих устройств: Ethernet-карта, ISDN-адаптер или модем, подключенный к последовательному порту. Каждое устройство (не весь компьютер!) имеет свой IP-адрес. Для выхода в локальные сети используется, как правило, Ethernet-карта, что и будет предполагаться в настоящем разделе. Заодно мы рассмотрим и настройку модемного интерфейса, поскольку настраивается он вполне аналогично, и знания о методах его настройки будут необходимы нам при рассмотрении вопроса о выходе в Интернет в *гл. 14.*

13.2.1. Расположение конфигурационных файлов

Отметим сразу, что все приводимые ниже команды можно выполнять из командной строки, но тогда придется повторять эти операции при каждом перезапуске компьютера. Поэтому может быть удобнее записать их в один из инициализационных файлов, автоматически запускаемых при старте системы. В разных дистрибутивах процесс загрузки организован по-разному. В Linux NET-3-HOWTO приводятся сведения о расположении конфигурационных файлов в разных дистрибутивах (табл. 13.1).

Таблица 13.1. *Расположение конфигурационных файлов в основных дистрибутивах*

Дистрибутив	Настройка интерфейса и маршрутизации	Запуск демонов
Debian	/etc/init.d/network	/etc/init.d/netbase /etc/init.d/netstd_init /etc/init.d/netstd_nfs /etc/init.d/netstd_misc
Slackware	/etc/rc.d/rc.inet1	/etc/re. d/rc.inet2

Таблица 13.1 (окончание)

Дистрибутив	Настройка интерфейса и маршрутизации	Запуск демонов
RedHat	/etc/sysconfig/network-scripts/ifup-<ifname>	/etc/rc.d/init.d/network

Обратите внимание, что дистрибутивы Debian и Red Hat содержат отдельный каталог для скриптов запуска системных сервисов (хотя сами файлы настроек находятся в других местах, например, в дистрибутиве Red Hat они хранятся в каталоге /etc/sysconfig). Для понимания процесса загрузки ознакомьтесь с содержимым файла /etc/inittab и документацией по процессу init.

13.2.2. Команда *ifconfig*

После подключения драйверов вы должны настроить те интерфейсы, которые вы предполагаете использовать. Настройка интерфейса заключается в присвоении IP-адресов сетевому устройству и установке нужных значений для других параметров сетевого подключения. Наиболее часто для этого используется команда ifconfig (ее название происходит от "interface configuration").

Запустите ее без аргументов (или с единственным аргументом -a) и вы узнаете, какие параметры установлены в данный момент для активных сетевых интерфейсов (в частности, для сетевой карты). Кстати, имеет смысл выполнить эту команду еще до подключения модулей: а вдруг у вас поддержка интерфейсов встроена в ядро и необходимые настройки сделаны в процессе инсталляции системы. Тогда вы в ответ можете получить информацию о параметрах вашей Ethernet-карты и так называемого "кольцевого интерфейса" или "обратной петли" — Local Loopback (интерфейс Ethernet при единственной сетевой карте обозначается как eth0, а кольцевой интерфейс — как lo). Если же по этой команде вы ничего не получите, то надо переходить к подключению модулей и настройке, и начинать надо с кольцевого интерфейса.

Настройка локального интерфейса /o

Этот интерфейс используется для связи программ IP-клиентов с IP-серверами, запущенными на той же машине, так что его необходимо настроить даже в том случае, если вы вообще не подключаете никаких сетевых устройств.

Локальный интерфейс настраивается очень просто: командой

```
[root]# /sbin/ifconfig lo 127.0.0.1
```

Теперь, чтобы проверить работоспособность протоколов TCP/IP на вашей машине, дайте команду:

```
[root]# ping 127.0.0.1
```

Настройка интерфейса платы Ethernet локальной сети *(eth0)*

Для того чтобы ваш компьютер вошел в сеть с IP-адресом, полученным вами у администратора (пусть для примера это будет адрес `192.168.0.15`), вы должны запустить команду `ifconfig` примерно следующим образом:

```
[root]# /sbin/ifconfig eth0 192.168.0.15 netmask 255.255.255.0 up
```

Если не указывать маску подсети, то по умолчанию устанавливается маска подсети `255.0.0.0`.

В некоторых случаях необходимо бывает изменить адрес прерывания, используемого сетевой картой, порта ввода/вывода или типа соединения, используемого в сети. Это можно сделать, выполнив следующую команду:

```
root# /sbin/ifconfig eth0 irq 5 io_addr 220 media 10baseT
```

Не все устройства (платы) поддерживают динамическое изменение этих параметров (то есть может потребоваться переустановить переключатели на плате).

Интерфейс для последовательного порта

Последовательный порт используется для подключения модема, через который осуществляется соединение с сетью по телефонной линии. Для настройки интерфейса этого типа тоже можно использовать команду `ifconfig`. Однако такие команды, как pppd и dip, используемые для соединения с сетью по модему, способны автоматически конфигурировать сетевой интерфейс, поэтому обычно для этого случая применять `ifconfig` не требуется.

13.2.3. Настройка маршрутизации

Правила маршрутизации определяют, куда отправлять IP-пакеты. Данные маршрутизации хранятся в одной из таблиц ядра. Вести таблицы маршрутизации можно статически или динамически. Статический маршрут — это маршрут, который задается явно с помощью команды route. Динамическая маршрутизация выполняется процессом-демоном (routed или gated), который ведет и модифицирует таблицу маршрутизации на основе сообщений от других компьютеров сети. Для выполнения динамической маршрутизации разработаны специальные протоколы: RIP, OSPF, IGRP, EGP, BGP и т. д.

Динамическая маршрутизация необходима в том случае, если у вас сложная, постоянно меняющаяся структура сети и одна и та же машина может быть

доступна по различным интерфейсам (например, через разные Ethernet- или SLIP-интерфейсы). Маршруты, заданные статически, обычно не меняются, даже если используется динамическая маршрутизация.

Для персонального компьютера, подключаемого к локальной сети, в большинстве ситуаций бывает достаточно статической маршрутизации командой route. Прежде чем пытаться настраивать маршруты, просмотрите таблицу маршрутизации ядра с помощью команды netstat -n -r. Вы должны увидеть что-то вроде следующего

```
[root]# netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask          Flags MSS Window irtt Iface
10.72.128.101   0.0.0.0         255.255.255.255   UH    0   0      0    eth0
10.72.128.0     0.0.0.0         255.255.255.0      U    0   0      0    eth0
127.0.0.0       0.0.0.0         255.0.0.0          U    0   0      0    lo
0.0.0.0         10.72.128.254   0.0.0.0           UG    0   0      0    eth0
```

Если таблица пуста, то вы увидите только заголовки столбцов. Тогда надо использовать команду route. С помощью команды route можно добавить или удалить один (за один раз) статический маршрут. Вот ее формат:

```
[root]# /sbin/route [-f] операция [-тип] адресат шлюз [dev] интерфейс
```

Здесь аргумент операция может принимать одно из двух значений: add (маршрут добавляется) или delete (маршрут удаляется). Аргумент адресат может быть IP-адресом машины, IP-адресом сети или ключевым словом default. Аргумент шлюз — это IP-адрес компьютера, на который следует пересылать пакет (этот компьютер должен иметь прямую связь с вашим компьютером). Команда

```
[root]# /sbin/route -f
```

удаляет из таблицы данные обо всех шлюзах. Необязательный аргумент тип принимает значения net или host. В первом случае в поле адресата указывается адрес сети, а во втором — адрес конкретного компьютера (хоста).

Как правило, бывает необходимо настроить маршрутизацию по упоминавшимся выше трем интерфейсам:

- локальный интерфейс (lo),
- интерфейс для платы Ethetnet (ethO),
- интерфейс для последовательного порта (PPP или SLIP).

Локальный интерфейс поддерживает сеть с IP-номером 127.0.0.1. Поэтому для маршрутизации пакетов с адресом 127.... используется команда:

```
[root]# /sbin/route add -net 127.0.0.1 lo
```

Если у вас для связи с локальной сетью используется одна плата Ethernet, и все машины находятся в этой сети (сетевая маска 255.255.255.0), то для настройки маршрутизации достаточно вызвать:

```
[root]# /sbin/route add -net 192.168.36.0 netmask 255.255.255.0 eth0
```

Если же вы имеете насколько интерфейсов, то вам надо определиться с сетевой маской и вызвать команду route для каждого интерфейса.

Поскольку очень часто IP-пакеты с вашего компьютера могут отправляться не в одну-единственную сеть, а в разные сети (например, при просмотре разных сайтов в Интернете), то в принципе надо было бы задать очень много маршрутов. Очевидно, что сделать это было бы очень сложно, точнее просто невозможно. Поэтому решение проблемы маршрутизации пакетов перекладывают на плечи специальных компьютеров — маршрутизаторов, а на обычных компьютерах задают маршрут по умолчанию, который используется для отправки всех пакетов, не указанных явно в таблице маршрутизации. С помощью маршрута по умолчанию вы говорите ядру "а все остальное отправляй туда". Маршрут по умолчанию настраивается следующей командой:

```
[root]# /sbin/route add default gw 192.168.1.1 eth0
```

Опция gw указывает команде route, что следующий аргумент, это IP-адрес или имя маршрутизатора, на который надо отправлять все пакеты, соответствующие этой строке таблицы маршрутизации.

После настройки маршрутизации можно проверить, что у вас получилось. Для этого снова дайте команду

```
[root]# netstat -nr
```

Если вывод команды выглядит так, как это было показано выше, но не содержит строки, которая в графе Destination содержит 0.0.0.0, а в графе Gateway указывает на маршрут, используемый для соединений по умолчанию, то вы, вероятно, не задали этот маршрут.

13.2.4. Настройка службы имен

С помощью команды ifconfig вы задали IP-адрес вашего компьютера, но он еще не знает своего имени (при инсталляции системы он получил обезличенное имя localhost). Существует команда hostname, которая позволяет установить (и узнать действующее в данный момент) имя компьютера и имя домена.

Однако установить только имя и только этой командой еще недостаточно, поскольку эта команда меняет имя только на текущий сеанс работы. Поэтому обычно эта команда вызывается в одном из инициализационных файлов, например, /etc/rc.d/rc или /etc/rc.d/rc.local. Вы можете попытаться найти ее там, чтобы изменить должным образом имя компьютера, которое задается в

качестве параметра команды hostname. В таком случае требуется перезагрузиться для того, чтобы изменения вступили в силу.

Другой способ изменения имени компьютера или домена состоит в том, что эти имена прописываются в файле /etc/sysconfig/network в виде двух строчек примерно следующего вида:

```
HOSTNAME="new_host_name.localdomain.upperdomain"
DOMAINNAME=localdomain.upperdomain
```

Тогда в процессе инициализации системы эти имена будут восстанавливаться, потому что файл /etc/sysconfig/network вызывается из /etc/rc.d/rc.sysinit.

Кроме того, имя компьютера должно быть прописано в файле /etc/hosts, который связывает имя компьютера с его IP-адресом. Каждая строка файла /etc/hosts должна начинаться с IP-адреса, за которым следует имя данного узла. Следом за именем можно записать произвольное число псевдонимов этого узла.

Даже если ваш компьютер не подключен к сети, в файле /etc/hosts должна быть прописана хотя бы одна строка следующего вида.

```
127.0.0.1      localhost     localhost.localdomain
```

Если же ваш компьютер подключен к TCP/IP-сети, то в этом файле дополнительно нужно прописать строку вида

```
192.168.0.15      host_name     host_name.localdomain
```

Файл /etc/hosts используется в механизмах разрешения имен. В больших сетях трудно было бы поддерживать в актуальном состоянии файлы /etc/hosts на всех компьютерах, если бы это был основной инструмент для определения IP-адресов по именам. Поэтому обычно для разрешения имен используются серверы DNS. Однако файл /etc/hosts все равно необходим, хотя бы для обращения к серверу DNS. Поэтому в нем имеет смысл указать IP-адреса и соответствующие имена шлюзов и серверов DNS и NIS. А чтобы все приложения использовали этот файл при разрешении имен, должен иметься файл /etc/hosts.conf, содержащий строку

```
order hosts,bind
```

которая говорит, что при разрешении имен сначала должен использоваться файл /etc/hosts, а затем должно происходить обращение к серверу DNS. В большинстве случаев в файле /etc/hosts.conf достаточно иметь две строки:

```
order hosts,bind
multi on
```

Эти параметры указывают системе преобразования имен, что надо просмотреть файл /etc/hosts перед тем, как посылать запрос к серверу, и что следует возвращать все найденные в /etc/hosts адреса для данного имени, а не только первый.

Но настройка механизма разрешения имен не ограничивается редактированием файлов /etc/hosts и /etc/hosts.conf. Необходимо еще указать компьютеру имена серверов DNS. Они прописываются в файле /etc/resolv.conf. Этот файл имеет весьма простой формат. Это текстовый файл, каждая строка которого задает один из параметров системы преобразования имен. Как правило, используются три ключевых слова-параметра:

- domain — задает имя локального домена;
- search — задает список имен доменов, которые будут добавляться к имени машины, если вы не укажете явно имени домена. Это позволяет ограничить область поиска и избежать некоторых ошибок (например, вы ищете компьютер linux.msk.ru, а механизм разрешения имен выведет вас на linux.spb.ru);
- nameserver — этот параметр, который вы можете указывать несколько раз, задает IP-адрес сервера преобразования имен, на который ваша машина будет посылать запросы. Повторяя этот параметр, вы можете задать несколько серверов.

Если вы не собираетесь заводить поддержку сервиса имен для своей сети (что является довольно сложной организационной и технической проблемой), и доверяете ведение своих имен администратору локальной сети или вашему IP-провайдеру, то вам достаточно задать файл /etc/resolv.conf примерно следующего вида:

```
domain abcd.ru
search abcd.ru xyz.edu.ru
nameserver 192.168.10.1
nameserver 192.168.12.1
```

В этом примере машина находится в домене abcd.ru. Если вы зададите имя машины, не указывая домена, например `pc1`, то система преобразования имен попытается сначала найти машину `pc1.abcd.ru`, а в случае неудачи – `pc1.xyz.edu.ru`. Для преобразования имен ваша машина будет обращаться к серверам по адресам `192.168.` и `192.168.12.1`.

13.2.5. Тестирование сетевого соединения

Чтобы проверить, соединяется ли ваш компьютер с сетью, попробуйте дать команду ping, указав ей в качестве параметра IP-адрес одного из компьютеров сети. Пусть, например, вам известно (узнайте реальный номер и имя у администратора сети), что в сети есть компьютер с IP-адресом `192.168.0.2` и именем `pc1`. Тогда вы должны дать команду:

```
[user]$ ping 192.168.0.2
```

или (тут вы одновременно проверяете и работу службы DNS)

```
[user]$ ping pc1
```

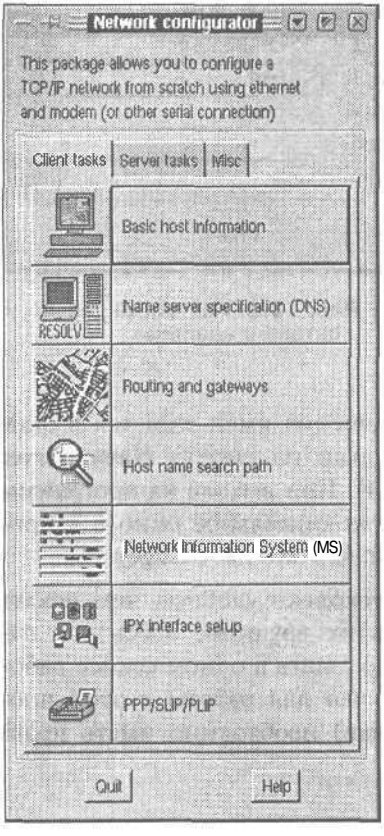
Если соединение с сетью установлено, должны появиться и периодически обновляться строчки примерно такого вида:

```
64 bytes from 192.168.0.2: icmp_seq=0 ttl=32 time=1.2 ms
64 bytes from 192.168.0.2: icmp_seq=1 ttl=32 time=1.0 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=32 time=1.0 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=32 time=1.0 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=32 time=1.1 ms
```

Это означает, что сетевое соединение работает. Для того чтобы прервать тестирование сети, нажмите комбинацию клавиш <Ctrl>+<C>.

13.2.6. Утилита netconf

В предыдущих разделах я попытался подробно и последовательно изложить, каким образом можно настроить выход в сеть путем прямого редактирования конфигурационных файлов. Впрочем, настройки локальной сети можно производить и с помощью специальных утилит netconf или netcfg, которые являются просто составной частью пакета linuxconf. Первая из них работает в графическом режиме, а вторая — в текстовом.



Надо только иметь в виду, что многие опытные пользователи Linux критически относятся к возможностям пакета linuxconf и предпочитают прямое редактирование конфигурационных файлов. Но для новичка эти утилиты могут оказаться удобнее, поэтому дадим их краткую характеристику на примере программы netconf. Запустив ее, вы увидите окно, изображенное на рис. 13.1.

Для того чтобы задать имя вашего компьютера, надо нажать кнопку **Basic host information,** после чего появится еще одно окно с пятью вкладками. На вкладке **Host name** задается имя компьютера (рис. 13.2), а на вкладке **Adaptor 1** — параметры IP-соединения (рис. 13.3).

Рис. 13.1. Главное меню программы netconf

Можно ввести параметры непосредственно в появившиеся строки ввода (если вы знаете, что нужно вводить), а можно выбрать подходящий вариант из списка, который появляется, если щелкнуть мышкой по треугольнику в конце строки. После завершения ввода параметров надо нажать экранную кнопку Accept.

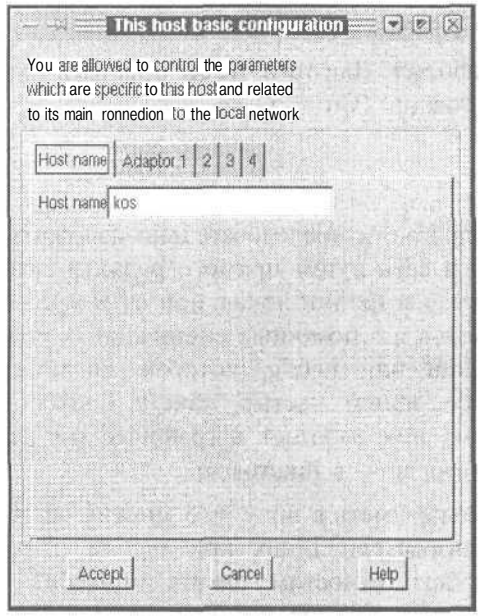


Рис. 13.2. Задание имени через netconf

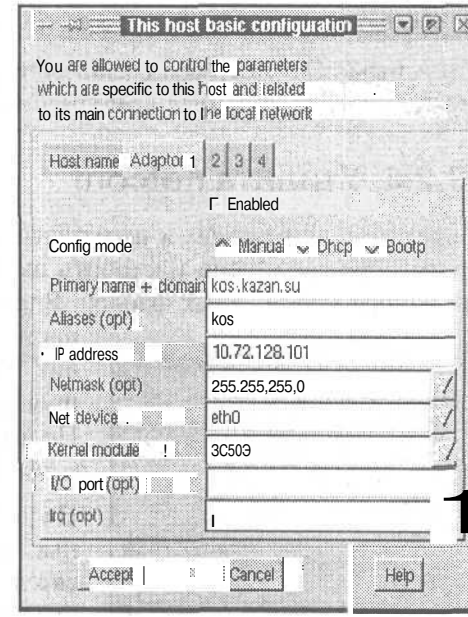


Рис. 13.3. Настройка сетевого адаптера

Адрес сервера DNS и настройка системы разрешения имен задается в окне **Resolver configuration,** появляющемся при нажатии на кнопку **Name server specification (DNS)** программы netconf (рис. 13.4). При выходе из профаммы после завершения редактирования появится дополнительное окно с запросом (рис. 13.5), в котором надо выбрать вариант **Activate the changes.**

Как видите, пользоваться этой профаммой несколько удобнее, чем искать нужные конфигурационные файлы и править их вручную. Здесь все настройки сети собраны в одном месте, и их легко задать в одном сеансе работы с профаммой. Надо только иметь в виду, что для работы с этой профаммой (как и с конфигурационными файлами) необходимо иметь права суперпользователя.

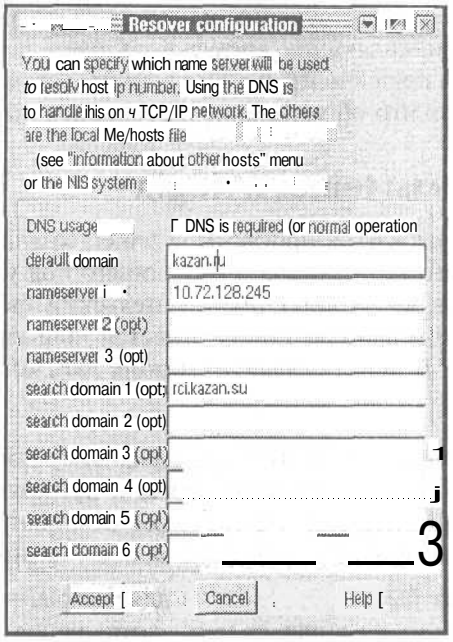


Рис. 13.4. Настройка системы разрешения имен через netconf

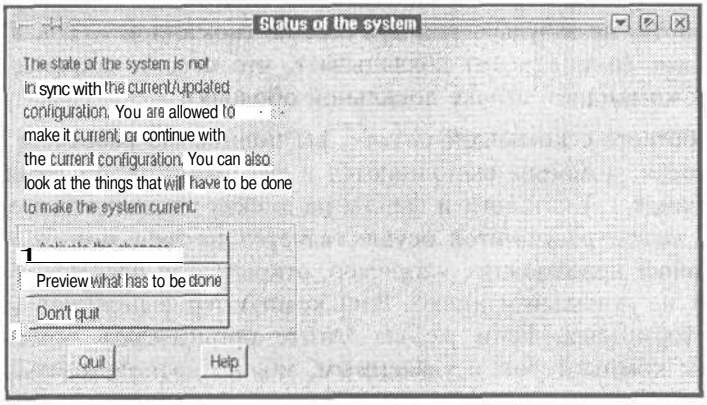


Рис. 13.5. Сохранение изменений, сделанных через netconf

13.3. Программы telnet и ftp

После того как вы убедились, что подключение к сети работает, можно попытаться выйти на какой-либо компьютер в сети по протоколу telnet или ftp. Telnet и ftp — это как имена протоколов для взаимодействия компьюте-

ров, так и названия соответствующих клиентских программ, реализующих эти протоколы и обеспечивающих доступ пользователям к удаленным компьютерам. Программы telnet и ftp по умолчанию устанавливаются при инсталляции системы, так что об установке их говорить не будем.

13.3.1. Программы telnet и rlogin

Для того чтобы воспользоваться программой telnet, вам необходимо знать имя или IP-адрес удаленного компьютера, работающего под управлением ОС типа UNIX, на котором для вас открыт пользовательский бюджет. Предположим для примера, что на компьютере `linux2` имеется пользователь `user5`, пароль которого вам известен. В таком случае вы можете дать команду

```
[user]$ telnet linux2
```

Если программе удалось подключиться к указанному компьютеру, на экране появится сообщение **Connected to server linux2** и приглашение к входу в систему, как если бы вы сидели за терминалом компьютера `linux2`. Вводите имя (`user5`) и пароль и вы будете работать на этом компьютере.

Команда rlogin может быть использована для выхода на удаленный компьютер вполне аналогично команде telnet, хотя лучше сразу указать в командной строке имя пользователя:

```
[user]$ rlogin -l user5 linux2
```

Завершив работу, не забудьте закрыть сессию (командой exit). После этого команда telnet (или rlogin) докладывает, что сессия закрыта, и вы возвращаетесь к командной строке локальной оболочки.

Когда вы работаете с командой telnet, вы полностью работаете на удаленном компьютере: команды выполняются в его оперативной памяти, вы видите (по команде `ls`) каталоги и файлы на дисках удаленного компьютера и т. д. Только вывод результатов осуществляется на ваш монитор. В рамках программы telnet невозможно, например, открыть для просмотра файл, расположенный на локальном диске. Ваш компьютер выполняет только роль удаленного терминала. Если же вы хотите организовать обмен файлами между вашим компьютером и удаленным, можно воспользоваться программой ftp.

13.3.2. Программа ftp

Программа ftp — это пользовательский интерфейс к стандартному протоколу передачи файлов по Интернету — File Transfer Protocol. Программа позволяет передавать файлы на удаленный компьютер и получать файлы с удаленного компьютера. Однако, введя команду ftp, вы запускаете только клиентскую программу. Для того чтобы получить доступ к файлам удален-

ного компьютера, на нем должен быть запущен FTP-сервер. Кроме того, необходимо знать либо имя и пароль пользователя, либо FTP-сервер должен разрешать анонимный доступ. Предположим, что эти условия выполнены и вы запустили программу ftp (без параметров). Вы увидите приглашение интерпретатора команд этой программы:

```
ftp >
```

Если ввести знак вопроса, программа выдаст перечень возможных команд. Первая команда, которую нужно в этом случае ввести, — команда open, после которой надо указать сетевое имя компьютера, на котором запущен FTP-сервер. Если анонимный доступ к этому серверу разрешен, то вы получите запрос на ввод имени и пароля пользователя. По команде pwd можно узнать имя текущего каталога на удаленном компьютере, а по команде `dir` — вывести список файлов и подкаталогов этого каталога. Команда cd `имя_каталога` используется для смены текущего каталога на удаленном компьютере.

В любой момент вы можете повторно ввести команду ? или ее эквивалент help, чтобы получить подсказку по возможным командам. Для получения более подробной подсказки по конкретной команде надо ввести имя интересующей вас команды после help или ?, например, так:

```
ftp > help dir
```

Если вы хотите выполнить какую-то команду на локальном компьютере (например, выяснить имя текущего каталога), надо дать соответствующую команду, перед которой поставить восклицательный знак:

```
ftp >! pwd
```

! — это команда интерпретатора, вызывающая новый экземпляр оболочки shell локального компьютера. Первый аргумент, следующий за символом `!`, должен быть командой оболочки, а все остальные аргументы — аргументами вызываемой команды. Для смены текущего каталога на локальном компьютере имеется специальная команда `lcd` (очень полезная, поскольку часто до запуска ftp забываешь перейти в тот каталог, куда хочешь скопировать файл с удаленного компьютера; не выходить же из-за этого из программы ftp).

Для пересылки файла на удаленный компьютер используется команда

```
ftp > put имя_файла
```

(или ее синоним send), а для копирования файла с удаленного компьютера в текущий каталог на локальном диске — команда

```
ftp > get имя_файла
```

В принципе этих двух команд вполне достаточно для организации обмена файлами с удаленным компьютером, но как же неудобно ими пользоваться! Приходится набирать полностью имена всех пересылаемых файлов. Поэтому

испытываешь воистину большое облегчение, когда узнаешь, что существуют такие команды, как mput и mget. Они позволяют задать шаблон имени пересылаемых файлов и будут дополнительно переспрашивать, надо ли пересылать каждый конкретный файл. Благодаря этому можно (самый крайний случай) заказать пересылку всех файлов:

```
ftp > get *
```

а потом либо подтверждать пересылку очередного файла, либо отказываться. Конечно, когда файлов в каталоге очень много, то и это окажется утомительной процедурой, но ведь можно задать более разумный шаблон! Так что думайте, как облегчить себе работу.

Перед тем как начать пересылку файлов, следует еще выполнить одну из команд, определяющих режим пересылки: ascii или binary. По умолчанию программа использует режим ascii, и это вполне допустимо при пересылке текстовых файлов, но если вы собираетесь передать или получить исполняемый файл, то необходимо задать режим binary. Процесс пересылки файлов можно прервать с помощью комбинации клавиш <Ctrl>+<C>.

Пока вы находитесь в программе ftp, вы можете выполнить некоторые операции с файлами и каталогами на удаленном компьютере (конечно, для этого надо иметь соответствующие права). По команде

```
ftp > rename from_name to_name
```

осуществляется переименование файла или каталога; команда

```
ftp > mkdir name
```

создает каталог, а

```
ftp > delete name
```

удаляет файл или каталог. Еще одна интересная команда — system, позволяет выяснить тип операционной системы на удаленном компьютере. Ну, и наконец, команда close (или disconnect) позволяет завершить сеанс работы с удаленным компьютером, не выходя из программы ftp (то есть предполагается, что после этого вы снова дадите команду open, например, для соединения с другим компьютером). Если же вы хотите вообще выйти из программы, то надо дать команду bye.

Я думаю, что приведенных здесь сведений вполне достаточно для успешной работы с программой ftp (для копирования файлов с удаленного или на удаленный компьютер, на котором запущен FTP-сервер). С другими возможностями этой очень полезной и широко применяемой программы вам придется знакомиться самостоятельно (вернее, с помощью man-страниц и системы info).

13.4. Сетевая файловая система NFS

NFS (эта аббревиатура чаще всего расшифровывается как Network File System, хотя в одном из источников я встретил расшифровку Network File Sharing) — это протокол, разработанный Sun Microsystems для разделения ресурсов (файлов и каталогов) локальной сети. NFS-клиент "монтирует" файловую систему, "экспортируемую" NFS-сервером. Смонтированная таким образом файловая система представляется на клиентском компьютере как часть локальной файловой системы.

Протокол NFS предоставляет, в частности, возможность смонтировать корневую файловую систему в процессе загрузки. Таким образом, можно обеспечить работу бездисковых рабочих станций.

Для того чтобы воспользоваться файловой системой NFS, должны быть выполнены два условия:

- поддержка файловой системы NFS должна быть встроена в ядро Linux или быть доступна в виде модуля;
- в сети должен иметься компьютер, на котором работает NFS-сервер. При этом в файлах настройки этого NFS-сервера должно быть указано, что вашему компьютеру разрешен доступ по NFS.

Пусть NFS-сервер работает на компьютере с именем `serv1`, причем для доступа по NFS открыт каталог /common, и вы хотите смонтировать его в каталог /mnt/serv1 своей файловой системы. Для этого надо (имея права пользователя root) выполнить команду

```
[root]# mount -o rsize=1024,wsize=1024 serv1:/common /mnt/serv1
```

Если в ответ вы получите сообщение, содержащее слова "Permission denied", значит вам не разрешен доступ на сервер.

Размонтировать файловую систему, смонтированную таким образом, можно обычной командой

```
[root]# umount /mnt/serv1
```

Конечно, если вам постоянно необходимо монтировать каталог /common с сервера serv1, то лучше добавить в файл /etc/fstab строку следующего вида:

```
serv1:/common /mnt/serv1 nfs rsize=1024,wsize=1024 0 0
```

Это обычный формат строки файла /etc/fstab:

```
device mountpoint fs-type options dump fsckcrder
```

Подробнее о выборе опций монтирования файловой системы NFS вы можете прочитать в NFS-HOWTO (этот документ имеется в русском переводе, см. [П15.7] *приложения).*

13.5. Подключение к Windows-сети

Я наверняка не ошибусь, если буду утверждать, что большинство компьютеров в вашей локальной сети работают под управлением ОС Windows. В таком случае к тем ресурсам этих компьютеров, которые "отданы" в общее пользование, проще всего подключаться, используя пакет Samba.

13.5.1. Что такое Samba

Samba — это набор приложений, позволяющих ОС Linux взаимодействовать с сетью, построенной на основе MS Windows, причем как в роли клиента сетей MS Windows, так и в роли сервера. Пакет Samba реализует протокол Server Message Block (SMB), который иногда называют также Session Message Block (SMB), протоколом NetBIOS или протоколом LanManager. В этом разделе мы рассмотрим только работу клиентских программ этого пакета, а именно smbclient, smbmount и smbumount. Если вы не найдете этих программ на своем компьютере, то установите пакет Samba (например, на дистрибутивном компакт-диске с Black Cat 6.02 имелся файл samba-client-2.0.5a-2bc.i386.rpm, возможно у вас окажется другая версия).

Программа smbclient предоставляет пользователю FTP-подобный интерфейс для переноса файлов с компьютеров, работающих под ОС Windows (или с компьютеров, на которых запущен сервер Samba). По сравнению с FTP программа smbclient имеет то преимущество, что не требует, чтобы на удаленном компьютере, работающем под Windows, была запущена специальная серверная программа, поскольку Windows поддерживает NetBIOS по умолчанию. Там только должен быть открыт доступ к какому-либо каталогу из сети. Если же вы хотите через Samba получить доступ к UNIX-серверу, то на нем, естественно, должна быть запущена серверная часть пакета Samba.

Итак, предположим, что в вашей сети имеется компьютер с именем PC1, работающий под ОС Windows, и на нем имеется каталог, открытый для доступа из сети, которому присвоено имя ресурса PUBLIC (напомним, что в ОС Windows регистр символов не имеет значения).

Для начала дайте команду

```
[root]# smbclient -L pc1
```

для того, чтобы увидеть доступные из сети ресурсы компьютера PC1. Если компьютер PC1 работает под управлением Windows NT, то надо сразу указать имя пользователя, который имеет права доступа к компьютеру:

```
[root]# smbclient -U user -L pc1
```

и в ответ на запрос программы ввести пароль этого пользователя, иначе вы не увидите даже списка открытых ресурсов.

В ответ на такой запрос вы получите примерно следующую информацию:

```
Domain=[WORKGROUP] OS=[Windows NT 4:0] Server=[NT LAN Manager 4.0]
 Sharename Type Comment
 --------- ---- -------
 ADMIN$         Disk      Remote Admin
 public         Disk
 C$             Disk      Default share
 IPC$           IPC       Remote IPC
 G              Disk
 Server Comment
 --------- -------
 PC2        Samba 1.9.15p8
 PC5
 PC25       Samba 1.9.15p8
 PC1
```

Обратите внимание на то, что вслед за строкой server Comment перечисляются другие SMB-серверы в сети с доступными ресурсами.

Для того чтобы получить доступ к ресурсу на удаленном компьютере, надо дать команду следующего вида:

```
[user]$ /usr/sbin/smbclient servicename -U user [password]
```

где servicename — это имя машины и ресурса, которые должны бы вообще-то иметь вид \\pc1\public, но из-за ограничений оболочки каждый слэш надо удваивать, поэтому команда принимает следующий вид:

```
[user]$ /usr/sbin/smbclient \\\\PC1\\public -U user mypasswd
```

(в той версии Samba, которая стоит у меня, прекрасно работают и обратные слэши, которые к тому же не надо удваивать:

```
[user]$ /usr/sbin/smbclient //PC1/public -U user mypasswd
```

скорее всего и у **вас** будет **то же самое).**

Указывать имя пользователя в опции необходимо только в том случае, если оно не совпадает с именем пользователя, от имени которого вы запустили профамму smbclient. Естественно, что пароль необходим только в том случае, если доступ к ресурсу защищен паролем.

Если доступ к ресурсу дан, вы получите приглашение программы:

```
Server time is Sat Mar 11 15:58:27 2000
Domain=[WORKGROUP] OS=[Windows NT 4.0] Server=[NT LAN Manager 4.0]
smb: \>
```

В ответ на это приглашение вы можете вводить одну из следующих встроенных команд программы smbclient (этот перечень вы можете получить, введя команду h или ?):

```
smb: \> h
```

ls	dir	du	lcd	cd
pwd	get	mget	put	mput
rename	more	mask	del	open
rm	mkdir	md	rmdir	rd
prompt	recurse	translate	lowercase	print
printmode	queue	cancel	quit	q
exit	newer	archive	tar	blocksize
tarmode	setmode	help	?	!

13.5.2. Монтирование файловых систем с помощью Samba

Как видите, команды эти во многом похожи на команды FTP-клиента и работа с программой smbclient не очень удобна. Но в пакет samba-client-2.0.5a-2bc.i386.rpm входят еще 2 программы, которые предоставляют некоторые дополнительные удобства. Эти программы называются smbmount и smbumount. С помощью команды smbmount можно смонтировать сетевой ресурс к локальной структуре каталогов, наподобие того, как монтируются файлы на гибком диске. Формат команды таков:

```
[user]$ /usr/sbin/smbmount //PC1/public /mnt/pcl -U 123 -W 456'
```

(в этом примере сетевой ресурс монтируется в локальный каталог /mnt/pcl, причем владельцем каталога объявляется пользователь 123 и группа 456). При необходимости нужно будет ввести пароль пользователя (тот же, по которому вы получали доступ к ресурсу в команде smbclient).

Команда smbumount позволяет обычным пользователям размонтировать файловую систему, смонтированную командой smbmount (пользователь root может воспользоваться обычной командой umount). Формат команды (используется название точки монтирования из того же примера):

```
[user]$ /usr/sbin/smbumount /mnt/pcl
```

Если после монтирования сетевого ресурса запустить программу Midnight Commander и перейти в каталог /mnt/pcl, то вы увидите файлы каталога public на компьютере PC1. Думаю, вы согласитесь с тем, что теперь с ними работать будет значительно проще, чем через smbclient.

Затруднения

Если что-то не получается, то проделайте следующее:

1. Убедитесь, что между двумя компьютерами имеется связь по сети, для чего дайте команду ping IP-adress с вашего компьютера на тот компьютер, к которому вы хотите получить доступ.
2. Если такая связь есть, проверьте работу DNS, для чего дайте ту же команду ping, но уже с именем удаленного компьютера: ping RemoteName.
3. Если вы подключаетесь к общему ресурсу компьютера, работающего под Windows 95/98, то этого должно быть достаточно; если же вы подключаетесь к Windows NT или 2000, то надо проверить имя пользователя и домена, с которыми ваш компьютер обращается к NT, для чего лучше всего просмотреть журнал событий (аудит входа в систему) на удаленном компьютере; если потребуется, то задать имя пользователя и имя домена (последнее задается в файле /etc/smb.conf).

13.6. Подключение к серверу Novell Netware

Для того чтобы подключаться к серверу Novell, необходимо установить пакет ncpfs. NCPFS — это файловая система, которая понимает протокол NCP (NetWare Core Protocol) фирмы Novell. Другими словами, пакет ncpfs — это клиент сети Netware для Linux. Протокол NCP играет в мире Novell ту же роль, какую в мире TCP/IP играет протокол NFS.

Пока что пакет ncpfs обеспечивает только работу с Novell Netware *3.x* и выше (но не *2.x)* и не поддерживает доступ к NDS, так что для работы с серверами версии *4.x* необходимо, чтобы на сервере была установлена эмуляция bindery.

Прежде, чем заниматься инсталляцией пакета ncpfs, следует убедиться, что ваше ядро поддерживает протоколы IPX и NCP. Вначале загляните в файл /var/log/dmesg и поищите там строки следующего вида:

```
NET4: Linux IPX 0.44 for NET4.0
IPX Portions Copyright (c) 1995 Caldera, Inc.
IPX Portions Copyright (c) 2000 Conectiva, Inc.
```

Если строк, напоминающих эти, не найдется, то ваше ядро не имеет поддержки IPX и вам придется его перекомпилировать. При компиляции ядра (на этапе выполнения команды make config) нужно на вопрос:

```
The IPX protocol (CONFIG_IPX) [N/y/m/?]
```

ответить либо "y", либо "m". На следующий вопрос (о поддержке full internal net) вы можете ответить отрицательно (если не хотите превратить свой Linux-компьютер в полноценный Novell-сервер).

Поддержка протокола SPX в ядре не требуется (если речь идет только о клиентской части), но требуется поддержка протокола NCP (надо ответить утвердительно на вопрос "NCP file system support").

Естественно, что должна также быть обеспечена поддержка сетевых средств вообще и настроен интерфейс сетевой платы *(см. разд. 13.2)*.

Если у вас уже была установлена предыдущая версия пакета ncpfs и вы просто обновляете инсталляцию, то до выполнения процедуры установки надо выполнить (все, что вы будете дальше делать, — надо делать имея права пользователя root) команду

```
[root]# umount -v -a -tncpfs
```

по которой размонтируются все ранее смонтированные через ncpfs ресурсы.

Проверить, установлены ли пакеты ipxutils и ncpfs, можно с помощью команд

```
[root]# rpm -q ipxutils
ipxutils-2.2.0.18-3
[root]# rpm -q ncpfs
ncpfs-2.2.0.18-3
```

Номера версий могут отличаться. Если упомянутые пакеты не установлены — установите (найти их можно в каталоге RPMS инсталляционного диска). Например, на дистрибутивном диске ASPLinux 7.1 нашлась версия 2.2.0.18-3 пакета ncpfs в виде файла ncpfs-2.2.0.18-3.i386.rpm, и пакет ipxutils той же версии, так что установка была выполнена следующим образом:

```
[root]# rpm -Uhv ipxutils-2.2.0.18-3.i386.rpm
[root]# rpm -Uhv ncpfs-2.2.0.18-3.i386.rpm
```

Теперь вы должны выяснить, какой IPX-фрейм "бегает" у вас по сети — *это очень важно* (узнайте это у администратора сервера Nowell). В качестве примера примем, что у вас используется 802.3. Выполните команды

```
[root]# /sbin/ipx_configure --auto_primary=on --auto_interface=off
[root]# /sbin/ipx_interface add -p ethO 802.3
[root]# /usr/bin/slist
```

Последняя команда должна выдать список серверов Nowell Netware в сети, который выглядит примерно так:

```
Known NetWare File Servers        Network       Node Address
-------------------------------------------------------------
TEST                              00034165      000000000001
SOFT                              3123DB21      000000000001
NWSTEND                           00100100      000000000001
```

Если вы не получите подобного списка серверов, не отчаивайтесь, — возможно, еще не все потеряно. Известны случаи, когда установленный пакет `mars_nwe` мешал нормальной работе с Novell-серверами. Удалите его *перед* установкой вышеуказанных пакетов

```
[root]# rpm -e mars_nwe.x.y.z
```

Далее создайте каталог, в который вы будете монтировать каталоги с Novell-сервера (если нужно несколько серверов, то должно быть несколько каталогов — по одному для каждого сервера). Например, для сервера NetWare1 создайте каталог /mnt/nw1. Теперь можно выполнить команду (для bindery-сервера), монтирующую том soft сервера NetWare1 с правами Novell-пользователя `nwuser1` в каталог /mnt/nw1:

```
[root]# /usr/bin/ncpmount -S netware1 -V soft -U nwuseri /mnt/nw1
```

Для NDS имя пользователя пишется по форме

```
cn=username.ou=unit.o=organization
```

Например, если у вас контекст `prog.firm`, имя пользователя nwuseri, то вызов этот выглядит следующим образом:

```
[root]# /usr/bin/ncpmount -S netware1 -U cn=nwuser1.ou=prog.o=firm
/mnt/nw1
```

Если все в порядке, то у вас будет спрошен пароль, и если вы его правильно введете, то каталоги сервера станут видны в указанном каталоге монтирования. Иногда команда ncpmount не срабатывает немедленно из-за задержек с прохождением пакетов. В таком случае подождите около 1 минуты и попробуйте снова.

Для размонтирования достаточно указать только точку монтирования:

```
[root]# /usr/bin/ncpumount /mnt/nw1
```

Однако выполнять все указанные действия необходимо с правами пользователя root. Если вы хотите иметь возможность монтировать серверы Novell от имени обычного пользователя (в домашний каталог, со своими правами доступа к серверам), то в конец файла /etc/rc.d/rc.local нужно вставить команды

```
ipx_configure --auto_primary=on --auto_interface=off
ipx_interface add -p eth0 802.3
chmod +s `which ncpmount`
chmod +s `which ncpumount`
```

Тогда при загрузке Linux они будут выполнены автоматически и обеспечат пользователям возможность выполнять команды ncpmount и ncpumount. После этого пользователи смогут монтировать в свои домашние каталоги серверы Novell командой ncpmount.

Предостережение!

Никогда не вставляйте команды `ncpmount` в файл типа /etc/rc.d/rc.local, это небезопасно с той точки зрения, что если злоумышленник проникнет в вашу машину — то он получит доступ и к серверам Novell. Лучше, если вы будете выполнять монтирование по мере необходимости. И не забывайте размонтировать их по исчерпании необходимости. В команде ncpmount можно задать и пароль в виде параметра команды, но не стоит этого делать: такой вызов сохраняется в истории команд и пароль может стать известным злоумышленнику. Не стоит и сохранять пароль в каком-либо скрипте — это тоже канал возможной компрометации пароля.

Глава 14

Интернет и электронная почта

14.1. Необходимые сведения о протоколах Интернета

ОС Linux в некотором смысле является продуктом, рожденным всемирной сетью Интернет. Создатели Linux использовали Интернет для обмена идеями, исходными кодами, просто опытом. И до настоящего времени Linux чаще всего воспринимается именно как сетевая ОС, наиболее приспособленная для решения серверных задач, как в отдельной организации, так и для обеспечения соединения с другими сетями.

Здесь не место для того, чтобы подробно объяснять устройство Интернета: для этого вам лучше найти специальные книги, посвященные этому вопросу. Но некоторые термины необходимо четко определить, чтобы последующий материал был всем понятен.

Во-первых, надо сказать, что рядовые пользователи обычно подключаются к сети Интернет через провайдера услуг Интернета (Internet Service Provider, ISP). Провайдер занимается эксплуатацией каналов связи и организацией взаимодействия пользователей с Сетью.

Во-вторых, передача информации в сети Интернет осуществляется с помощью набора (или, как часто говорят, "стека") протоколов TCP/IP.

Комплект TCP/IP состоит из нескольких различных протоколов, каждый из которых выполняет в сети определенную задачу. Базовых протоколов два: протокол управления передачей (TCP), который обеспечивает отправку и прием сообщений, и межсетевой протокол (IP), который определяет правила пересылки отдельных пакетов от отправителя получателю (правила маршрутизации). Остальные протоколы выполняют различные вспомогательные и управляющие функции.

Одним из базовых понятий протокола IP является понятие IP-адреса. Каждому сетевому интерфейсу (Ethernet-карте, последовательному порту с модемом и т. д.) на компьютере присваивается IP-адрес. Такой адрес состоит из четырех байтов. Их принято записывать в так называемой "десятично-точечной нотации" (dotted decimal notation). В ней каждый байт представляется десятичным числом в интервале от 0 до 255, байты разделены точками. Возможно использование одного адреса несколькими интерфейсами на одной машине, но, как правило, этого не делают.

Последний байт в адресе считается индивидуальным адресом данного компьютера (точнее, интерфейса, но обычно говорят именно об адресе компьютера). Оставшаяся часть считается "сетевой частью" адреса. Для выделения сетевой части применяют так называемую "маску". Это такая последовательность, наложение которой в двоичной форме на адрес машины (операцией "логическое И") выделяет "сетевую часть". Добавлением к сетевой части нулевого четвертого байта получают адрес сети, к которой принадлежит данный компьютер.

В каждой сети обычно выделяют "широковещательный адрес" — специальный адрес, который "слушают" все машины в сети, кроме своего собственного. Таким образом передается информация о маршрутизации или сетевые сообщения об ошибках. Чаще всего в качестве широковещательного используют наибольший адрес в IP-сети — x.x.x.255. Однако некоторые сети используют в качестве широковещательного адрес x.x.x.O.

Все сказанное иллюстрируется следующим примером:

```
-----------------------------------
Адрес машины 192.168.110.23
Сетевая часть 192.168.110.
Машинная часть .23
-----------------------------------
Маска 255.255.255.0
Адрес сети 192.168.110.0
Широковещательный адрес 192.168.110.255
-----------------------------------
```

Для выхода в Интернет с персональных (в частности, домашних) компьютеров чаще всего используется модем и телефонная линия. Существуют два протокола, которые позволяют передавать IP-сообщения по телефонным линиям. Это протокол SLIP (Serial Line Internet Protocol — межсетевой протокол для последовательного канала) и протокол PPP (Point-to-Point Protocol — протокол "точка-точка"). SLIP по возрасту старше и поэтому не всегда удовлетворяет предъявляемым ныне требованиям, а PPP — более современный универсальный протокол, который завоевывает все большую популярность. Он обеспечивает гораздо более стабильное соединение и может поддерживать работу целого ряда сетевых протоколов помимо IP.

Для установления SLIP-соединения в дистрибутивы Linux обычно включается программа dip, а PPP-соединение устанавливается с помощью программы pppd. Эта программа конфигурирует соединение, устанавливает лимиты MTU и получает IP-адреса. В отличие от dip, pppd не умеет устанавливать соединение с удаленной хост-системой. Чтобы использовать pppd, нужно сначала установить соединение, для чего обычно используется

программа chat. Сначала эта программа устанавливает соединение, а затем pppd конфигурирует его.

Протокол PPP предусматривает возможность использования средств аутентификации пользователя, подключающегося к серверу. Обычно аутентификация осуществляется за счет проверки имени и пароля пользователя. Существует два основных способа аутентификации, определяемых протоколами PAP (Password Authentication Protocol) и CHAP (Challenge Handshake Authentication Protocol).

PAP-аутентификация происходит следующим образом. При установлении PPP-соединения сервер предлагает пользователю использовать аутентификацию по протоколу PAP. Пользователь соглашается и затем передает свое имя и пароль открытым текстом. Если удаленную сторону имя и пароль устраивают, то аутентификация считается успешной.

Имена и пароли для PAP хранятся в файле /etc/ppp/pap-secrets в виде отдельных строк следующего формата: имя пользователя, имя удаленного сервера и пароль. У него должны быть такие права доступа "*rw*-------".

В принципе, в PAP используется только имя и пароль пользователя, а имя удаленного сервера нужно только для того, чтобы можно было определить, какой пароль нужно использовать в случае, когда вы используете одно и то же имя у разных провайдеров, например:

```
username demos pas12345
username citynet pas98765
```

Заметим, что хотя пароль передается в открытом виде, удаленная сторона может хранить пароль в виде результата какой-либо хэш-функции, например, MD5, в качестве параметра которой выступает пароль.

CHAP-аутентификация происходит следующим образом. При установлении PPP-соединения удаленный сервер предлагает пользователю аутентификацию по протоколу CHAP. Пользователь соглашается, и сервер высылает ему ключ (challenge), состоящий из случайной последовательности символов, и свое имя. Пользователь берет свой пароль и присланный ключ, и прогоняет их через алгоритм MD5. Получившийся результат высылает вместе со своим именем серверу. Сервер, зная пароль пользователя и высланный ему ключ, в свою очередь, проделывает то же самое у себя, и, если его результат совпадает с присланным, то аутентификация считается успешной. Таким образом, пароль не передается в открытом виде, но удаленный сервер должен хранить пароли пользователей (обычно в открытом виде).

Имена и пароли для CHAP хранятся у пользователя в файле /etc/ppp/chap-secrets, права доступа у него должны быть такие же, как для PAP: "*rw*-------", и формат строк тоже совпадает.

Протокол PPP позволяет организовать аутентификацию в одних случаях через PAP, а в других — через CHAP. Для этого клиентское ПО при запуске определяет, каким образом можно аутентифицировать себя, исходя из локального имени и имени удаленной стороны, проверяя, есть ли в файлах /etc/ppp/pap-secrets или /etc/ppp/chap-secrets строки с такими именами. И если, скажем, удаленная сторона предлагает CHAP, а в файле /etc/ppp/chap-secrets соответствующего пароля не имеется, то будет запрошен PAP, и если это устраивает удаленную сторону, то аутентификация пройдет по PAP.

Отдельно нужно рассказать о том, как происходит аутентификация в том случае, когда у провайдера установлен Windows NT Remote Access Server (RAS). Windows NT RAS поддерживает два протокола аутентификации: PAP и так называемый MS CHAP 80. Аутентификация через PAP пройдет в том случае, если в Windows NT RAS установлен параметр "Allow any authentication including clear text". Если же администраторы провайдера установили параметры "Require encrypted authentication" или даже "Require Microsoft encrypted authentication", то единственный способ аутентификации — это MS CHAP 80.

Что же такое MS CHAP 80? В CHAP могут использоваться различные методы шифрования и передачи пароля. Описанному выше методу с использованием алгоритма MD5 присвоен номер 05. Microsoft разработал свой метод с использованием алгоритмов MD4 и DES для Windows NT 3.5, 3.51, 4.0 и Windows 95, назвал его MS CHAP и ему был присвоен номер 80. Этот метод описан в RFC 2433. Но видимо, что-то там оказалось не так, и после SP3 был выпущен pptp3-fix (включенный в SP4) и rras30-fix с новым методом за номером 81, названный MS CHAP V2. Предыдущий вариант MS CHAP теперь называется MS CHAP V1.

Для того чтобы из Linux можно было аутентифицироваться по методу MS CHAP 80, нужно собрать пакет pppd с поддержкой этого метода.

14.2. Подготовка к выходу в Интернет

Для того чтобы получить выход в Интернет, первым делом надо организовать физический канал между вашим компьютером и сервером провайдера. Существует три основных способа организации физического соединения: через локальную сеть, ISDN-адаптер или модем, через который осуществляется обмен по протоколу PPP.

Первый случай имеет место тогда, когда ваш компьютер включен в локальную сеть, в которой имеется сервер доступа к Интернету. В этом случае от вас практически не требуется никаких усилий для организации соединения — все заботы берет на себя администратор локальной сети. Вы просто набираете в браузере URL интересующего вас ресурса и получаете к нему доступ. Идеальный вариант!

Соединение через ISDN-адаптер встречается пока не часто, так что мы его не рассматриваем.

Третий способ, соединение с помощью модема по коммутируемой телефонной линии, чаще всего используется для выхода в Интернет с отдельных (в частности, домашних) компьютеров. В этом случае администратора для помощи у вас может и не оказаться, так что все надо делать самому. Поэтому этот случай мы рассмотрим подробнее.

Естественно, что первым делом нужно иметь модем и телефон.

Далее нужно выбрать провайдера услуг Интернет (ISP) и договориться с ним об открытии для вас счета доступа. При этом вам нужно получить у провайдера следующие данные:

- номер телефона, по которому ваш компьютер будет соединяться с модемным пулом провайдера;
- ваши имя пользователя и пароль для доступа в Интернет.

Дополнительная информация, которая может оказаться необходимой для организации полноценного доступа к службам провайдера, включает:

- IP-адрес сервера DNS *(Domain Name Service* — служба доменных имен). При заключении договора с провайдером вы можете получить от него один или несколько адресов серверов DNS;
- адрес шлюза (gateway address): некоторые провайдеры могут требовать указания адреса шлюза в настройках;
- имя домена провайдера;
- информацию о том, какой почтовый протокол используется провайдером (POP3 или ШАР);
- адрес сервера, осуществляющего обработку входящей почты (его имя может иметь вид pop.yourisp.com или mail.yourisp.com), и адрес сервера, осуществляющего обработку исходящей почты (SMTP-сервера). Зачастую обработка исходящей почты осуществляется тем же сервером, на котором обрабатывается входящая почта, и тогда их имена, естественно, совпадают;
- адрес (имя) новостного (NNTP) сервера (что-нибудь вроде news.yourisp.com или nntp.yourisp.com);
- имена proxy-серверов, которые установил ваш провайдер.

Далее необходимо убедиться, что ваша конфигурация Linux имеет необходимые средства поддержки протокола PPP.

Для работы по протоколу PPP у вас должен быть установлен пакет ppp и ваше ядро должно иметь встроенную поддержку протокола PPP (и TCP/IP). Для того чтобы проверить, поддерживает ли ваше ядро протокол PPP, надо ввести команду: `dmesg | more` или просмотреть файл /var/log/dmesg.

Вы должны увидеть такие строки:

```
PPP: version 2.3.0 (demand dialling)
TCP compression code copyright 1989 Regents of the University of California
PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
PPP line discipline registered
```

Большинство руководств, в частности, PPP-HOWTO, рекомендуют в том случае, когда вы таких строк не найдете, перекомпилировать ядро, включив при компиляции поддержку PPP (о том, как перекомпилировать ядро, рассказано в *гл. 17).* Если поддержка протокола PPP в ядре была организована в виде модуля, то нужно подключить модуль поддержки PPP командой

```
[root]#/usr/sbin/insmod ppp
```

Описываемая ниже программа kppp подключает этот модуль автоматически. После успешного запуска kppp вы должны обнаружить приведенное выше сообщение о подключении PPP в файле /var/log/messages.

После того как вы будете уверены, что поддержка протокола PPP обеспечена, можно подключить модем к компьютеру, соединить его с телефонной линией и перейти к настройке соединения вашего компьютера с сервером провайдера. Поскольку соединение происходит по протоколу PPP, именно его надо настраивать.

Есть два способа заставить работать PPP: сконфигурировать вручную и использовать конфигурационную программу.

Ручная конфигурация — дело довольно сложное, требующее редактирования файлов и написания скриптов. Работы, говорят, немного (только надо предварительно прочитать PPP HOWTO или другое руководство), но легко сделать ошибку. Если вы хотите попробовать ручной способ конфигурации, то обратитесь к рекомендациям Игоря Сысоева на его страничке **http://www.nitek.ru/~igor/pppd/. Он,** правда, описывает настройку PPP для операционной ситемы FreeBSD, однако для Linux все делается вполне аналогично.

Но все же для новичков предпочтительнее воспользоваться одним из специальных инструментов, автоматизирующих выполнение этих задач. В составе графической среды KDE для создания и изменения учетных записей PPP имеется превосходная программа kppp. Эта утилита облегчает настройку соединения; в большинстве случаев от вас потребуется только правильно указать учетную информацию.

14.3. Программа kppp

Программа kppp — одна из очень удобных утилит KDE, позволяющая легко установить подключение к Интернету и легко его модифицировать. Она включает функции настройки PPP-интерфейса, обеспечивает "дозвон" по

телефонной линии и совместно с демоном pppd (Point-to-Point Protocol daemon) устанавливает и поддерживает соединение с сервером провайдера.

Чтобы запустить kppp, можно воспользоваться командой меню **Главное Меню KDE | Интернет | Kppp.** Если вы хотите запускать программу kppp от имени обычного пользователя, надо установить suid-бит на исполняемый файл /usr/bin/kppp. В противном случае при запуске пользователем этой программы будет появляться сообщение, изображенное на рис 14.1.

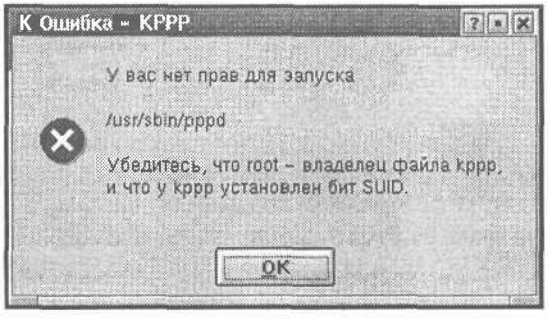


Рис. 14.1. Сообщение о невозможности запуска kppp

Если запускать kppp с правами администратора, то таких сообщений не появляется, открывается главное окно программы, изображенное на рис. 14.2.

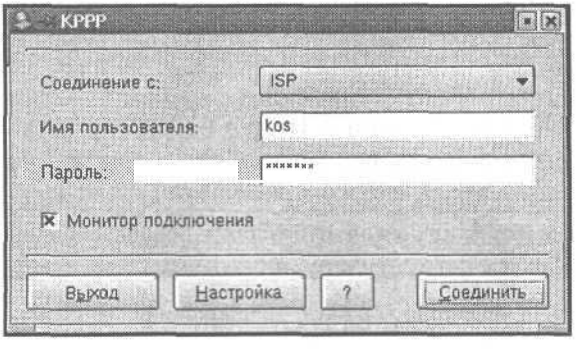


Рис. 14.2. Главное окно программы kppp

14.3.1. Конфигурирование kppp

Чтобы начать конфигурирование, нажмите кнопку **Настройка.** Откроется диалоговое окно конфигурации kppp (рис. 14.3). Начать надо, конечно, с создания новой учетной записи соединения с провайдером услуг Интернета,

для чего щелкнуть по кнопке **Создать.** Появляется окно выбора варианта создания нового соединения, изображенное на рис. 14.4.

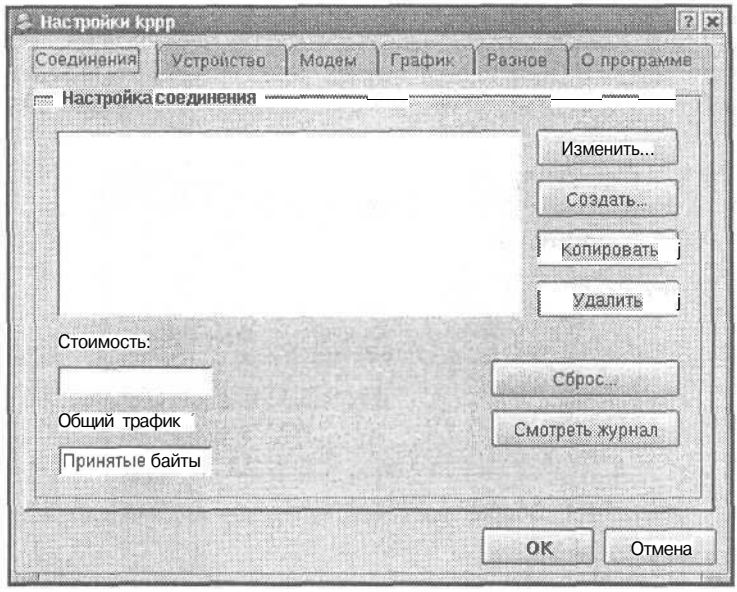


Рис. 14.3. Окно настройки kppp

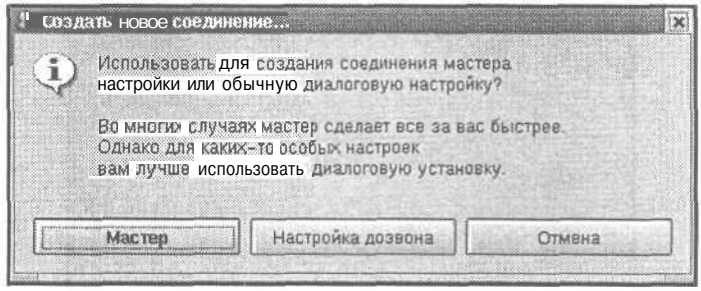


Рис. 14.4. Выбор варианта создания нового соединения

Если воспользоваться кнопкой **Мастер,** то следующие два окна предоставят возможность выбора страны подключения (России в этом списке пока нет) и одного из известных программе провайдеров интернет-услуг в выбранной стране (рис. 14.5). Поскольку я не думаю, что вы будете пользоваться услугами одного из тех провайдеров, о которых известно программе, придется отказаться от использования мастера настройки, и выбрать вариант **Настройка дозвона.**

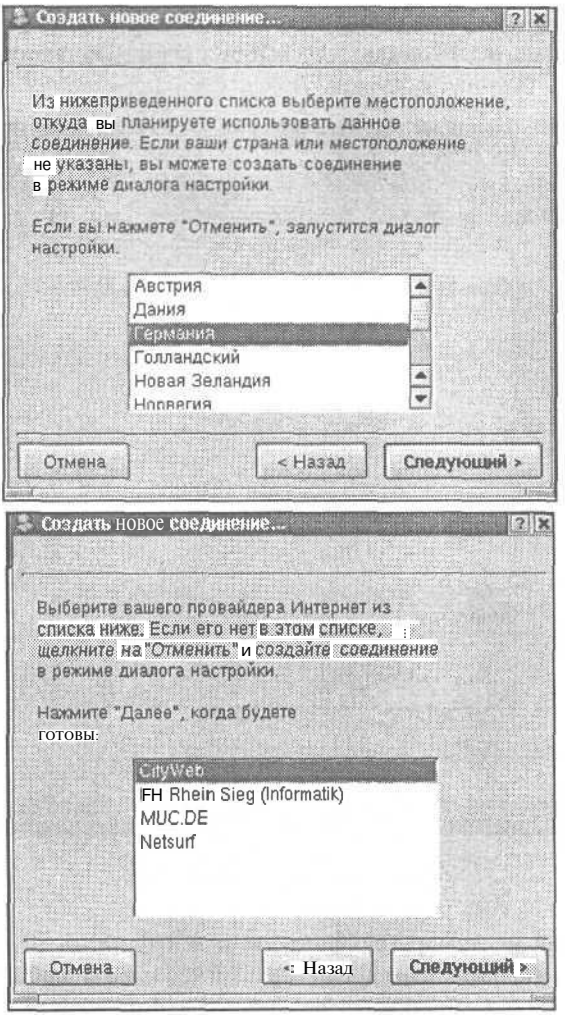


Рис. 14.5. Диалоговые окна, представляемые в варианте "Мастер настройки"

После щелчка по кнопке **Настройка дозвона** появляется окно **Новое соединение,** на котором вы видите 6 вкладок (рис. 14.6). Вообще говоря, вводить все данные, упоминающиеся на этих вкладках, вовсе не обязательно. Но мы рассмотрим их все последовательно, а что именно можно не вводить, вы определите экспериментально.

На вкладке **Дозвон,** которая открывается первой, надо ввести следующие данные:

- **Имя соединения** — название, которое вы хотите дать учетной записи (например, название вашего провайдера). Имя нужно только потому, что

вы можете иметь возможность выхода в Интернет через разных провайдеров и тогда имя необходимо для выбора варианта соединения;

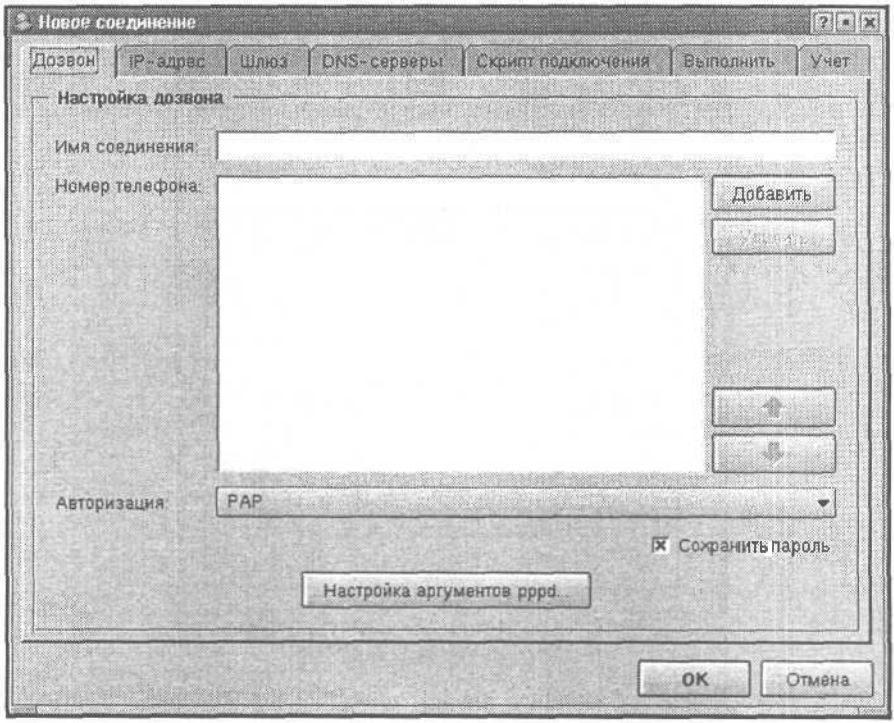


Рис. 14.6. Вкладка **Дозвон**

- **Номер телефона** — номер телефона (или телефонов), по которому вы будете выходить на модемный пул провайдера. Для добавления номера в список надо сначала щелкнуть по клавише **Добавить. В** номерах телефонов можно использовать дефисы (для облегчения чтения номеров). Если несколько номеров перечислены один за другим через двоеточие (1111111:2222222:3333333), kppp будет набирать эти номера один за другим, если получит в ответ сигнал занятости. Кнопка **Удалить** используется для удаления из списка подсвеченного номера, а кнопки со стрелками применяются для изменения порядка номеров в списке;
- **Авторизация** — в этом раскрывающемся списке вы можете выбрать один из следующих способов аутентификации: **Сценарии, PAP, Терминал** или **CHAP.** Это методы проверки вашего имени пользователя и пароля. Вам, возможно, придется выяснить у вашего провайдера, какой способ идентификации следует использовать (чаще всего применяется PAP);

- **Сохранить пароль** — установите этот флажок, если не хотите вводить пароль при каждом соединении. Правда, если вас заботят вопросы безопасности, то лучше этого не делать!

Вы можете также указать параметры, которые нужно передать pppd, нажав кнопку **Настройка аргументов pppd.** При этом откроется отдельное окно **Настройка аргументов pppd** (рис. 14.7). Задаваемые в этом окне аргументы kppp передаст демону pppd. Допустимые значения аргументов вы можете найти на man-странице pppd.

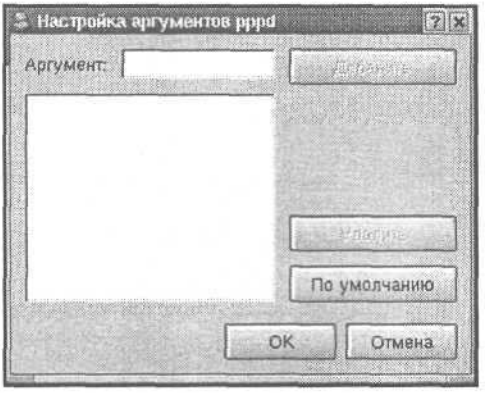


Рис. 14.7. Окно **Настройка аргументов pppd**

На вкладке **IP-адрес** (рис. 14.8) либо введите постоянный IP-адрес, если провайдер выделил вам его, либо укажите, что вы будете использовать динамическое выделение IP-адреса.

На этой же вкладке находится флажок **Автоконфигурация DNS-имени машины по данному IP-адресу.** Руководство по программе kppp рекомендует быть осторожным при задании этой опции и включать ее только в том случае, когда вы абсолютно уверены в необходимости этого.

Если включить автоконфигурацию, то kppp, после успешного установления соединения с сервером, будет запрашивать имя домена и имя, назначаемое вашему компьютеру, у сервера DNS (см. описание вкладки **DNS-серверы** ниже). При этом сервер DNS определяет указанные имена по IP-адресу, который назначен данному сеансу связи по протоколу PPP. Исходное имя машины (hostname) и имя домена, введенное на вкладке **DNS-серверы,** при задании этой опции игнорируются. Они будут восстановлены после завершения сеанса связи по протоколу PPP.

Эта опция полезна в случаях, когда вы хотите использовать такие протоколы, как talk, для которых требуется, чтобы имя машины совпадало с именем, соответствующим IP-адресу машины. Но если вы хотите соединиться с

Интернетом только для обычного просмотра Web-страниц, обмена почтовыми сообщениями или использования chat, то включать опцию не следует. Дело в том, что у нее есть побочный эффект, заключающийся в невозможности новых подключений к X-серверу. Другими словами, вы не сможете запускать какие-то приложения в фафическом режиме, пока не завершите сеанс связи.

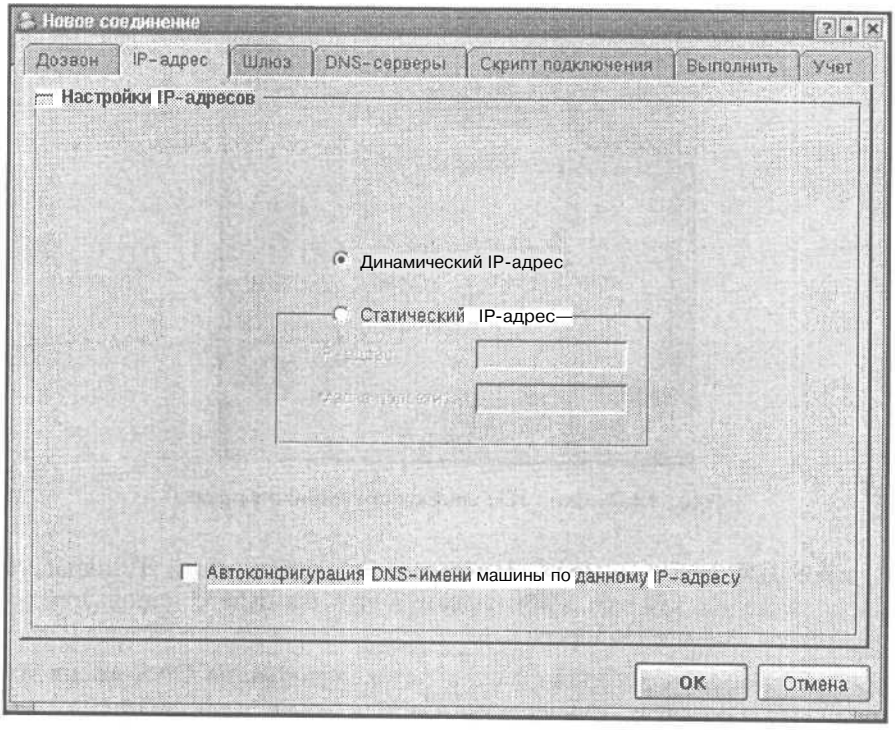


Рис. 14.8. Вкладка **IP-адрес**

На вкладке **Шлюз** (рис. 14.9) выберите либо **Шлюз по умолчанию,** либо, если провайдер сообщил вам IP-адрес статического шлюза, введите его в поле **Статический шлюз.** Нужно только отметить, что по мнению разработчиков программы kppp, опция **Назначить данный адрес шлюзом по умолчанию** должна быть включена в подавляющем большинстве случаев.

Настройка службы имен (вкладка **DNS-серверы**, рис. 14.10) может производиться автоматически (тогда провайдер сам задает адреса серверов DNS, когда начинается сеанс связи) или в ручном режиме. Но в любом случае вы должны ввести имя домена (например, **demos.online.ru**) и хотя бы один IP-адрес сервера DNS для того, чтобы иметь возможность использовать ориентированные на человеческое восприятие интернет-имена, такие, например,

как ftp.kde.org. Адреса DNS-серверов должны указываться в числовой форме, например: 212.24.32.192. Эти адреса будут добавлены во время исполнения команды в файл /etc/resolv.conf. После завершения сеанса связи файл /etc/resolv.conf будет возвращен в исходное состояние.

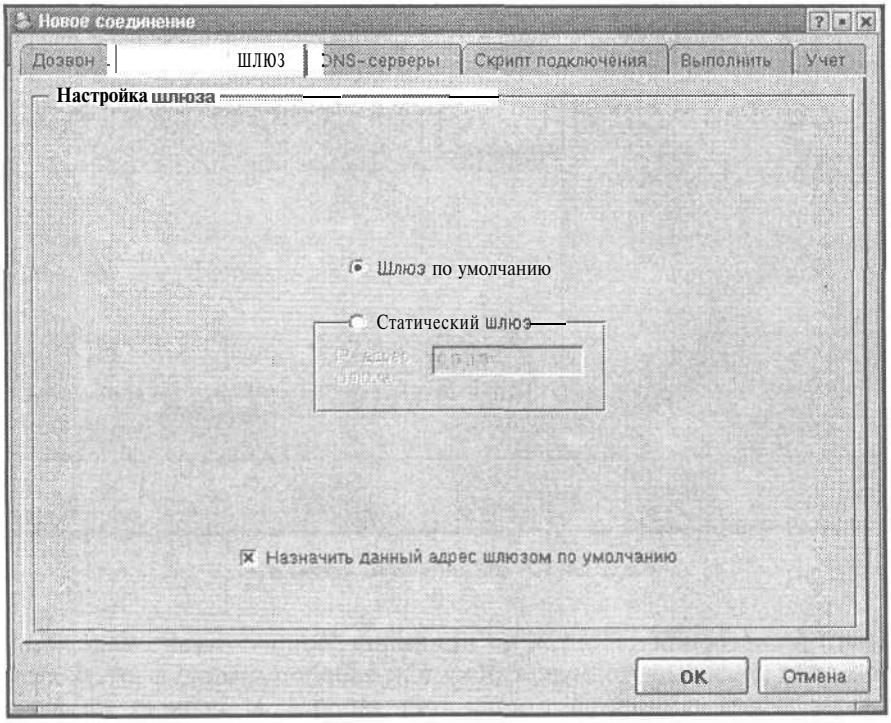


Рис. 14.9. Вкладка **Шлюз**

Когда вы введете цифры, нажмите кнопку **Добавить,** и они попадут в **Список адресов серверов DNS**. (Вы не сможете нажать кнопку **Добавить,** пока не введете полный IP-адрес сервера DNS в поле IP-адреса.)

Если флажок **Закрыть доступ к существующим серверам DNS на время соединения** установлен, то на время сеанса связи будет "отключен" список адресов DNS, заданный в файле /etc/resolv.conf.

Вкладка **Скрипт подключения** (рис. 14.11) позволяет создать сценарий регистрации, который будет выполняться при установлении соединения, если вы выберете вход по сценарию в поле **Аутентификация** на вкладке **Дозвон. Не** все провайдеры требуют использования сценария регистрации; вам необходимо выяснить у своего провайдера, надо ли выполнять какие-либо особенные рекомендации при установке соединения.

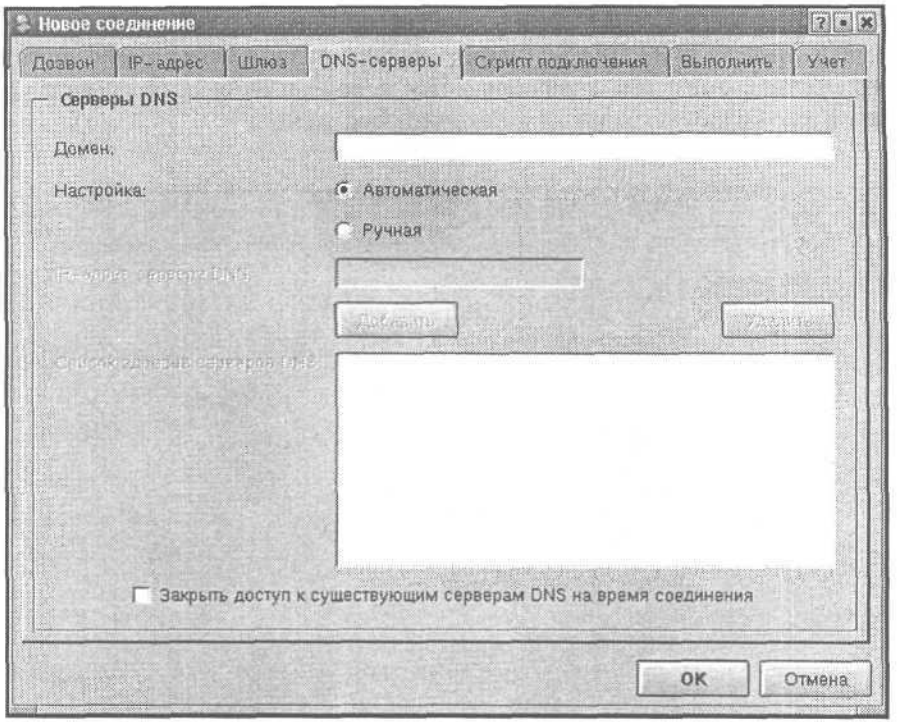


Рис. 14.10. Вкладка **DNS-серверы**

Сценарий регистрации строится по принципу "вопрос-ответ". Ваш компьютер посылает информационную строку или запрос серверу, а затем ожидает получения информационной строки или запроса от сервера провайдера. Кроме команд посылки и ожидания строк в сценариях можно использовать еще некоторые команды. Вот перечень возможных команд:

- Expect -- kppp ожидает получения указанной строки;
- send — kppp будет посылать указанную строку серверу;
- scan — kppp будет просматривать входной поток символов в ожидании указанной строки. После получения этой строки все поступающие символы до первого символа "конец строки" будут сохранены во внутреннем буфере. Пробелы в начале и конце строки игнорируются;
- save — сохраняет полученную строку в указанном регистре. Пока что может использоваться единственный регистр "password";
- Pause — пауза на указанное число секунд;
- Hangup — послать модему сигнал "повесить трубку";
- Answer — перевести модем в режим ожидания ответа;

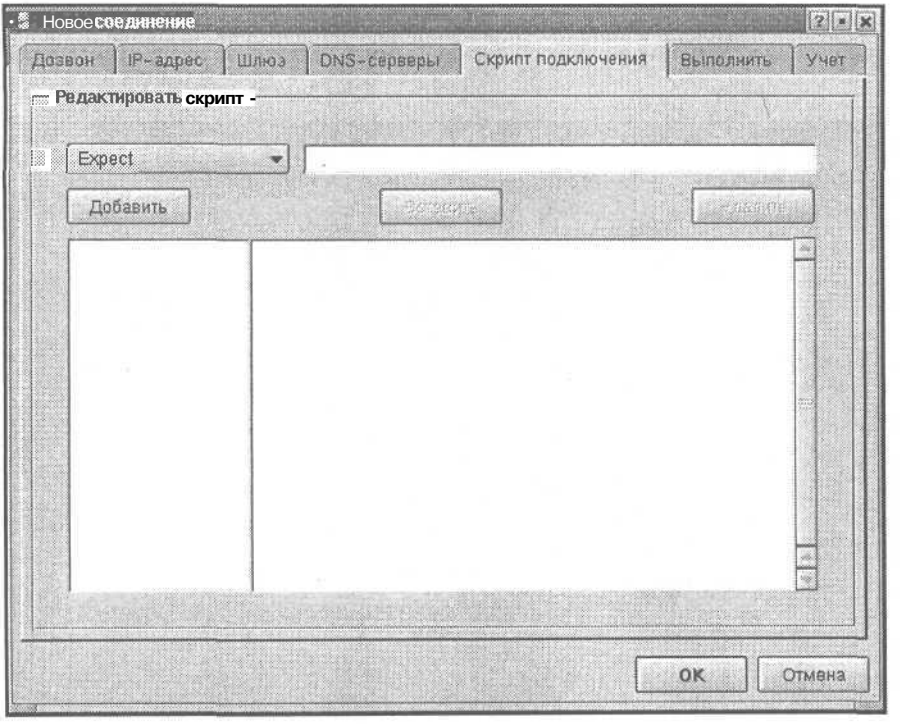


Рис. 14.11. Вкладка Скрипт подключения

- Timeout — изменить заданное по умолчанию значение паузы и установить его в указанное значение (в секундах). Вы можете изменять это значение в скрипте несколько раз, если это необходимо;
- Prompt — выдать пользователю строку запроса, причем в заголовке окна запроса будет приведена подсказка, которую вы указали при задании этой команды. Если в строке подсказки вы вставили ##, эти символы будут заменены на текущее содержимое внутреннего буфера (см. выше команду scan). Вводимые пользователем в ответ на запрос символы будут отображаться в строке ввода;
- PWPrompt • - как и в предыдущей команде пользователю будет выдана строка запроса, однако вводимые им символы не будут отображаться (точнее, будут заменены звездочками). Используется для ввода паролей;
- ID — послать серверу имя пользователя, введенное в главном окне kppp (рис. 14.2). Если имя в главном окне не указано, будет выдан запрос на ввод имени. В качестве заголовка запроса выводится введенная вместе с этой командой строка. Вводимые пользователем символы отображаются на экране. Если запрос имени пользователя встречается в скрипте второй раз,

то введенное в первом запросе имя пользователя сразу же выводится на экран, независимо от того, был ли он указан в главном окне программы;

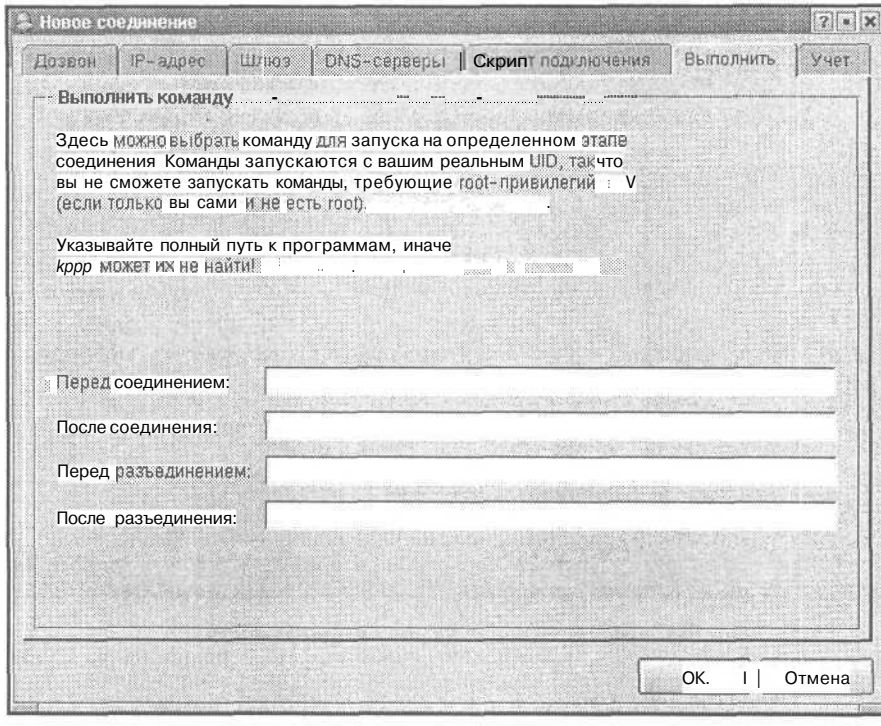


Рис. 14.12. Вкладка **Выполнить**

- Password — послать серверу пароль, введенный в главном окне kppp. Если пароль в главном окне не задан, будет выдан запрос на его ввод. Заголовком запроса служит введенная в правом поле строка. Вводимые пользователем символы пароля отображаются в виде звездочек;
- Loopstart — kppp будет ожидать получения указанной строки. Строка запоминается для использования в команде LoopEnd;
- LoopEnd -- kppp будет ожидать получения указанной строки для выхода из цикла. Если первой получена строка, заданная в соответствующей команде Loopstart, будет осуществлен переход к строке, следующей за командой Loopstart, т. е. возврат к первой строке цикла.

Сценарии регистрации составляются следующим образом.

Сначала используйте раскрывающийся список в левой верхней части окна, чтобы выбрать нужную команду. В строке ввода справа от раскрывающегося

списка вводится строка информации, передаваемая в данной команде (или параметры команды, вроде продолжительности паузы).

После полного определения команды нажмите кнопку **Добавить,** чтобы добавить ее в конец сценария. Таким образом, можно последовательно задавать команды для создания сценария шаг за шагом. Если вам необходимо изменить сценарий, можно удалить какие-то команды или вставить новые. Удалить шаг из сценария можно выделив его и нажав кнопку **Удалить.** Для того чтобы вставить команду в сценарий, сформируйте новую команду указанным выше образом, после чего выделите в сценарии ту команду, после которой надо вставить новую, и щелкните по кнопке **Вставить.**

Пример сценария входа, который ожидает от сервера предложение ввести имя пользователя, выдает пользователю приглашение для ввода имени пользователя, затем такое же приглашение для ввода пароля и, в конце, после небольшой паузы, ожидает от пользователя строку ppp, приведен в табл. 14.1.

Таблица 14.1. *Пример сценария входа*

Команда	Параметр команды	Назначение
Prompt	Нажмите клавишу <Enter>	Отправляет серверу символ конца строки
Expect	Username:	Ожидание приглашения сервера на ввод имени пользователя
ID		Передать имя пользователя из учетной записи
Expect	Password:	Ожидание приглашения с сервера ввести пароль
Password		Передать пароль из учетной записи
Expect	Welcome	Ожидание приветственного сообщения сервера
Pause	3	Устанавливает длительность паузы в 3 секунды

В общем случае использовать процедуру входа по сценарию необязательно; это зависит от требований провайдера. В некоторых случаях достаточно использования PAP или CHAP. Тем не менее, проконсультируйтесь с вашим провайдером.

На следующей вкладке (рис. 14.13) вы можете задать 4 команды (в качестве которых могут, естественно, выступать скрипты, содержащие цепочки команд), которые будут выполняться при запуске и остановке программы kppp.

Примером команды, выполняющейся после установления соединения, может быть команда проверки наличия почты в вашем почтовом ящике.

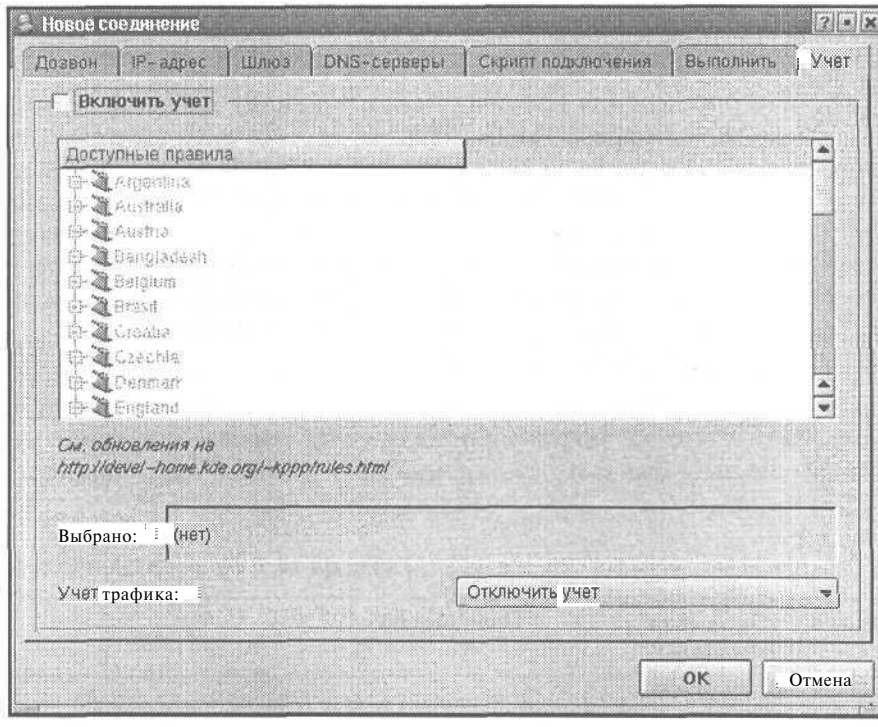


Рис. 14.13. Вкладка Учет

На вкладке **Учет** (рис. 14.13) вы можете указать, где и как kppp будет следить за вашей работой в Интернете. Если вы хотите, чтобы программа вела учет стоимости телефонного соединения, проставьте отметку возле надписи **Включить учет.** Дальше необходимо выбрать правила учета. Скорее всего вы не найдете подходящего правила среди тех вариантов, которые предлагаются программой. Но такие правила можно задать самостоятельно. Образец их задания приведен в подсказке к программе. Файл с правилами учета можно проверить на синтаксическую правильность с помощью команды kppp -r `rule_file`, а затем этот файл надо поместить в каталог ${KDEDIR}/share/apps/kppp/Rules или в каталог ${HOME}/.kde/share/apps/kppp/Rules, после чего вы получите возможность выбрать заданное вами правило на данной вкладке.

В нижней части вкладки имеется выпадающий список **Учет трафика,** с помощью которого можно выбрать один из трех вариантов подсчета числа полученных и отправленных байтов: входящий трафик, исходящий трафик или и то и другое. Можно также и отказаться от учета трафика. Мониторинг трафика может оказаться полезным, если вы платите провайдеру за количество байтов, полученных вами во время соединения — даже, например, при месячной постоянной абонентской оплате.

После того как вы закончили создание учетной записи, нажмите на кнопку **OK.** Теперь в окне **Настройка kppp** появится новая учетная запись. Если вы хотите ее модифицировать, выделите ее одним щелчком мыши, потом нажмите кнопку **Изменить** в окне **Настройка соединения.**

Теперь самое время предоставить системе информацию о типе подключения и скорости вашего модема на вкладке **Устройство** (рис. 14.14).

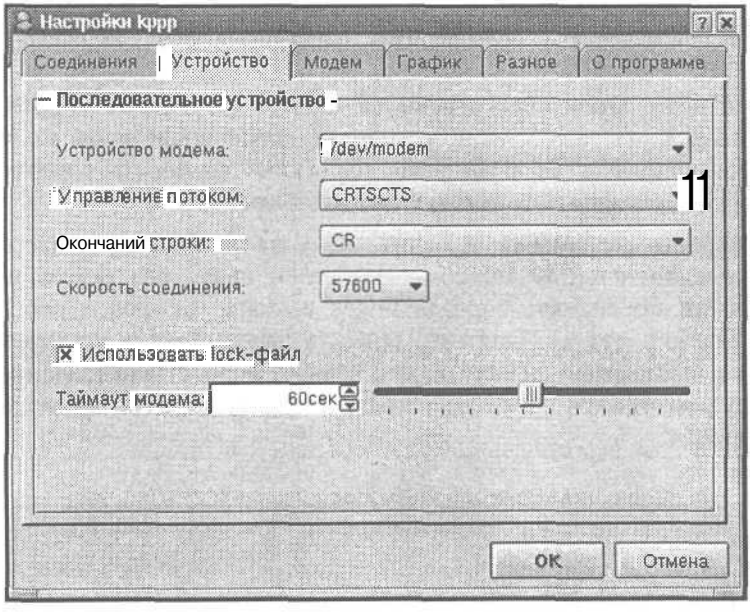


Рис. 14.14. Вкладка Устройство

Здесь вы указываете, к какому порту подключен модем, его скорость, некоторые аппаратные установки и другие настройки.

В раскрывающемся списке **Устройство модема** вы можете выбрать последовательный порт, к которому подключен ваш модем.

Ниже приведен список соответствия названий последовательных портов в MS-DOS (и Windows) и Linux. Если вы знаете, что ваш модем присоединен, например, к порту COM2 в Windows, то в Linux это будет соответственно /dev/ttyS1.

- COM1 = /dev/ttyS0
- COM2 = /dev/ttyS1
- COM3 = /dev/ttyS2
- COM4 = /dev/ttyS3

В некоторых дистрибутивах Linux имеется символическая ссылка /dev/modem, которая указывает на файл реального устройства. В документации по kppp не рекомендуется использовать эту ссылку вместо указания на реальное устройство.

После того как вы выбрали порт для подключения модема, вы можете установить режим **Управление потоком.** Хотя предоставляется возможность выбрать другой режим управления потоком из раскрывающегося списка, рекомендуется оставить значение, заданное по умолчанию — **CRTSCTS,** которое означает аппаратное управление потоком (**XON/XOFF** означает программное управление потоком).

Следующее поле, **Окончание строки,** позволяет установить корректное значение этого параметра для вашего модема. Для большинства модемов подходит значение CR/LF. Если же у вас возникают трудности с опросом модема, попробуйте поменять установку этого параметра.

В поле **Скорость соединения** выберите одну из скоростей, поддерживаемых вашим последовательным портом. Имейте в виду, что последовательный порт способен обеспечить гораздо более высокие скорости передачи данных, чем любой модем. Поэтому вначале здесь можно установить самое большое из возможных значений, а в случае, если с установлением соединения будут возникать проблемы, постепенно переходить на меньшие значения скорости.

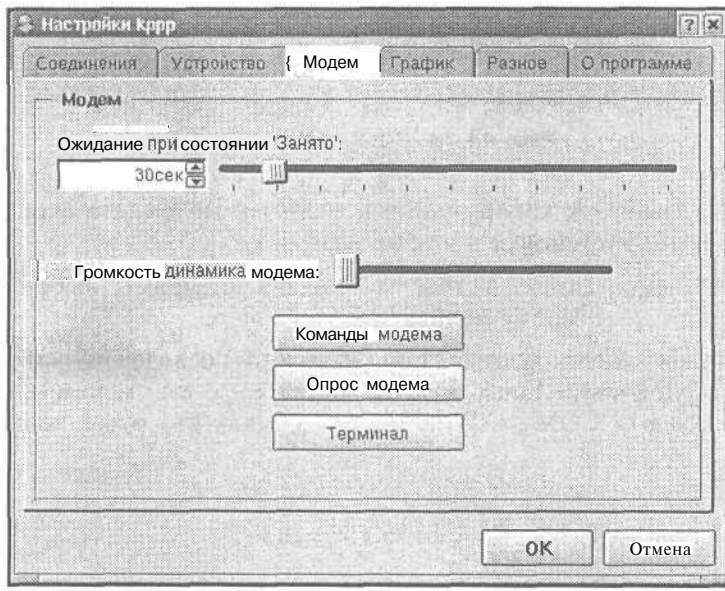


Рис. 14.15. Вкладка **Модем**

Флажок **Использовать lock-файл** (файл блокировки) по умолчанию установлен. Это значит, что kppp будет блокировать устройство при работе в режиме online, предотвращая, таким образом, доступ к модему. Если задана опция **Использовать lock-файл** для kppp, то не должна использоваться аналогичная опция для pppd. В противном случае pppd не сможет запуститься и kppp выдаст сообщение об ошибке.

В поле **Таймаут модема** указывается время в секундах, в течение которого kppp ожидает ответа CONNECT после набора номера. В руководстве к программе kppp рекомендуется установить здесь значение, равное 30 секундам.

На вкладке **Модем** (рис. 14.15) задаются настройки для модема.

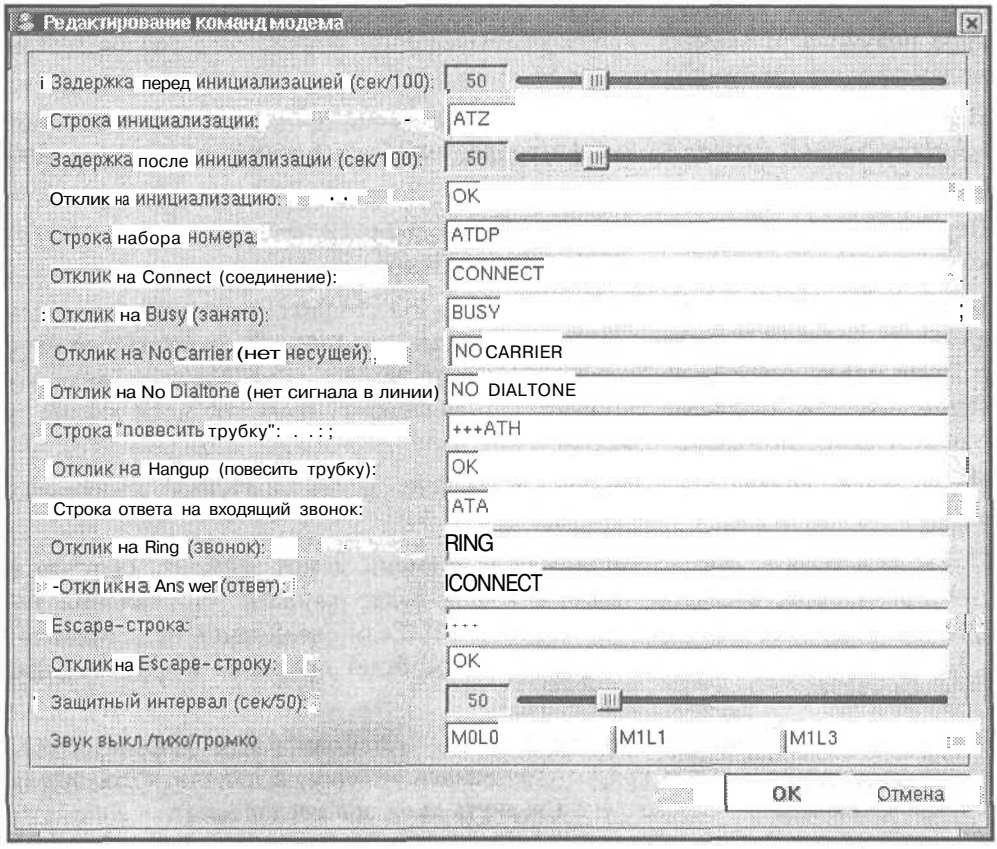


Рис. 14.16. Диалоговое окно **Редактирование команд модема**

Самым важным элементом на этой вкладке является, пожалуй, кнопка **Команды модема.** При нажатии на эту кнопку появится диалоговое окно **Ре-**

дактирование команд модема (рис. 14.16), позволяющее изменять процесс набора номера, установки соединения и другие настройки модема. Если ваш модем является Hayes-совместимым, то вам, скорее всего, не требуется ничего здесь менять.

Внимание

Ознакомьтесь с документацией к вашему модему для настройки строки инициализации, строки набора и других параметров.

Кнопка **Опрос модема** на вкладке Модем служит для того, чтобы программа kppp попыталась идентифицировать модем в вашей системе. Появится окно, в котором отображается индикатор хода опроса. В случае успешного опроса в открывшемся диалоговом окне появятся сведения, которые программа kppp получила от модема. Успех этой операции зависит от того, выдаст ли модем верные идентификационные данные.

При нажатии кнопки **Терминал** kppp откроет мини-терминал, с помощью которого можно проверить конфигурацию модема (но надо знать соответствующие команды).

Перемещая движок **Громкость динамика модема,** вы можете изменять громкость динамика модема во время набора номера и установки соединения.

Вкладка **График** позволяет изменить цвета на графике, отображающем количество байтов, проходящих между вашим компьютером и провайдером в режиме online. Этот график выводится в отдельном окне **Монитор подключения,** которое открывается, если установлена соответствующая опция в главном окне программы (см. рис. 14.2).

На вкладке **Разное** (рис. 14.17) вы можете задать значения следующих опций:

- **Таймаут pppd** — определяет интервал времени между запуском скрипта и стартом pppd, в течение которого kppp будет ожидать установления устойчивого соединения по протоколу PPP. Если соединение не установлено, то по истечении этого времени связь будет прервана и процесс pppd остановлен;
- **Встроить в панель KDE при соединении** — установите этот флажок, чтобы после установки связи kppp отображалась значком в панели. (Эта опция перекрывает действие флажка **Свернуть окно при соединении);**
- **Автодозвон при разъединении** — установите этот флажок, если хотите, чтобы kppp мгновенно восстанавливала соединения при разрыве связи;
- **Показывать время в строке заголовка** — установите этот флажок, чтобы видеть время соединения;

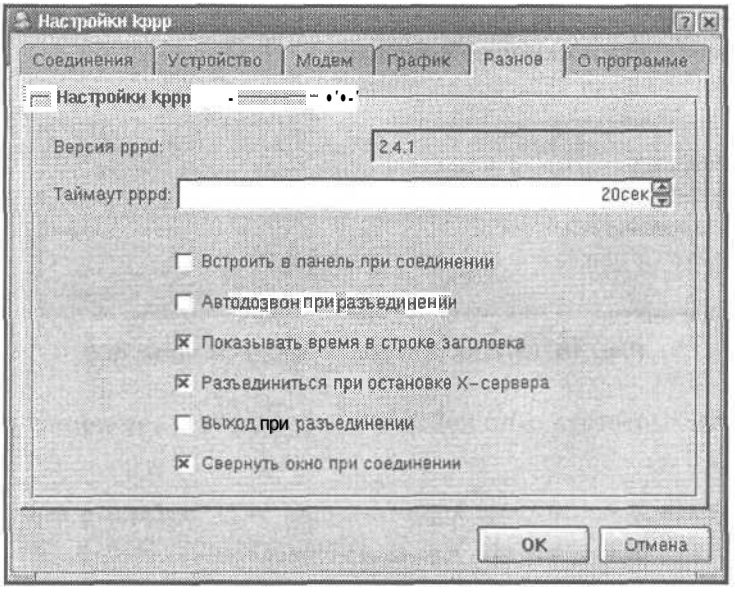


Рис. 14.17. Вкладка **Разное**

- **Разъединить при остановке X-сервера** — если вы выберете эту опцию, kppp будет корректно завершать сеанс связи при выключении X-сервера. Это полезная опция, если вы не хотите терять время на разрыв соединения при выходе;
- **Выход при разъединении** — когда вы разрываете соединение с провайдером, kppp автоматически закрывается. В противном случае вы вернетесь к исходному окну kppp;
- **Свернуть окно при соединении** — сворачивание окна kppp в панель задач при установке соединения.

Ну, и наконец, на вкладке **О программе** представлена некоторая информация о kppp, такая как сведения об авторах, авторских правах и лицензионном соглашении.

На этом описание настроечных параметров программы kppp завершено. Надеюсь, что приведенных сведений достаточно для конфигурирования соединения с провайдером. Рассмотрим вкратце процесс непосредственного подключения к Интернету с помощью программы kppp.

14.3.2. Установка связи с помощью kppp

Для запуска kppp надо выбрать в **Главном меню KDE** команду **Интернет** и щелкнуть на kppp. Можно (и это проще) воспользоваться комбинацией кла-

виш <Alt>+<F2>, ввести в появившейся строке ввода (рис. 14.18) имя программы и щелкнуть по кнопке **Выполнить:**

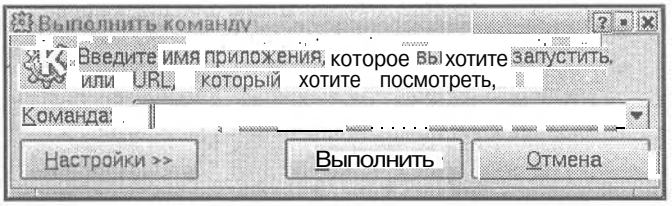


Рис. 14.18. Строка ввода команды оболочки KDE

Можно также запустить окно консоли и выполнить в нем команду

```
[user1]$ kppp &
```

В появившемся главном окне kppp (см. рис. 14.2), выберите вариант соединения, введите имя пользователя и пароль учетной записи для доступа в Интернет.

Если вы хотите отладить конфигурацию или наблюдать за процессом соединения, установите флажок **Монитор подключения.**

Чтобы начать сеанс, нажмите на кнопку **Соединить.** Появятся два окна. В одном из них (рис. 14.19) показан статус соединения, например, так:

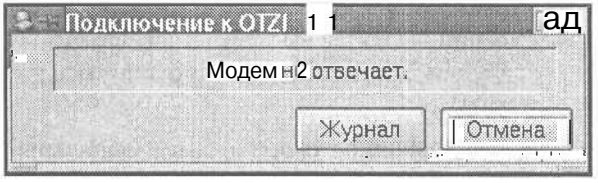


Рис. 14.19. Статус соединения

Второе окно **(Монитор подключения)** отображает команды инициализации модема и процесс обмена данными с сервером провайдера. Если вы не сделали этого раньше, вы можете открыть окно Монитора, воспользовавшись кнопкой **Журнал** в окне статуса, изображенном на рис. 14.19.

После установления соединения с провайдером kppp свернется в кнопку или иконку на панели задач в зависимости от того, какие значения опций вы задали на вкладке PPP (см. выше). Окно монитора тоже закрывается.

Чтобы просмотреть статистику соединения, например, пропускную способность или IP-адреса, откройте окно статуса kppp (рис. 14.20) и нажмите на кнопку **Подробности.** Появится окно статистики соединения (рис. 14.21).

Рис. 14.20. Окно статуса после запуска kppp

Несмотря на то, что kppp вроде бы работает, давайте займемся проверкой того, что все настроено правильно. Первым делом введите (от имени пользователя root) команду

```
[root]# /sbin/ifconfig
```

По этой команде должны быть показаны все работающие ("поднятые") сетевые интерфейсы. Среди них должен быть "кольцевой интерфейс" (Local Loopback) и pppO, причем для pppO в информации, выводимой по этой команде, можно найти присвоенный вам IP-адрес и адрес сервера, с которым вы соединились (эта же информация имеется и в окне статистики, изображенном на рис. 14.21).

Рис. 14.21. Окно статистики соединения

Вот пример вывода команды /sbin/ifconfig:

```
lo Link encap Local Loopback
    inet addr 127.0.0.1 Beast 127.255.255.255 Mask 255.0.0.0
    UP LOOPBACK RUNNING MTU 2000 Metric 1
    RX packets 0 errors 0 dropped 0 overrun 0
    TX packets 0 errors 0 dropped 0 overrun 0
pppO Link encap Point-to-Point Protocol
    inet addr 192.168.17.1 P-t-P 192.168.18.1 Mask 255.255.255.0
    UP POINTOPOINT RUNNING MTU 1500 Metric 1
    RX packets 33 errors 0 dropped 0 overrun 0
    TX packets 42 errors 0 dropped 0 overrun 0
```

Если вы, в частности, не обнаружите в выводе информации о "кольцевом интерфейсе", то вернитесь к *разд. 13.2,* где об этом говорится, и настройте локальный интерфейс.

Теперь дайте команду

```
[root]# ping z.z.z.z
```

где z.z.z.z — IP-адрес сервера DNS (этот адрес вы должны были получить у провайдера и приписать в настройках kppp). Если все корректно работает, то вы увидите строки следующего вида:

```
root:~# ping 212.22.66.70
PING 212.22.66.70 (212.22.66.70): 56 data bytes
64 bytes from 212.22.66.70: icmp_seq=0 ttl=255 time=268 ms
64 bytes from 212.22.66.70: icmp_seq=1 ttl=255 time=247 ms
64 bytes from 212.22.66.70: icmp_seq=2 ttl=255 time=266 ms
^C
--- 212.22.66.70 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 247/260/268 ms
root:~#
```

(Для того чтобы прервать работу команды ping, надо нажать комбинацию клавиш <Ctrl>+<C>.)

Следующий этап проверки состоит в запуске команды

```
[root]# netstat -nr
```

Я в этом случае увидел всего три строки, включая строку заголовка (хотя у вас может оказаться и больше):

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags MSS Window irtt Iface
```

```
192.168.14.1    0.0.0.0         255.255.255.255 UH    0 0 0 pppO
0.0.0.0         192.168.14.1    0.0.0.0         UG    0 0 0 pppO
```

Если вывод команды выглядит подобным образом, но не содержит строки, в которой в графе Destination стоят 4 нуля (`0.0.0.0`) (она указывает на маршрут, используемый для соединений по умолчанию), то вы, вероятно, не включили переключатель **Шлюз по умолчанию** на вкладке **Настройка соединения | Шлюз.**

Далее вы можете попытаться установить соединение с сервером провайдера по одному из протоколов telnet, ftp, finger, используя известные вам цифровые IP-адреса.

Если все эти проверки завершились успешно, вы можете запустить любой интернет-браузер (а в KDE и запускать дополнительно ничего не надо — используйте файловый менеджер Konqueror), задать имя (URL) ресурса и отправиться в путешествие по виртуальному миру. Если запустить почтового клиента, то можно получать и отправлять электронную почту. Подробнее о том, как это сделать, будет сказано в *разд. 14.4* и *14.5,* где кратко описаны браузеры и почтовые клиенты для Linux.

Правда, не всегда все проходит так гладко. Если вам не удается после установления соединения получить выход в Интернет, прочитайте *разд. 14.3.3,* где рассмотрены возможные затруднения.

Чтобы разорвать соединение, если приложение прикреплено к панели, щелкните один раз по его значку. В открывшемся окне статуса соединения нажмите на кнопку **Разъединить.**

Если вы выбрали режим сворачивания в **Панель задач,** разверните приложение и тоже нажмите кнопку **Разъединить.**

14.3.3. Проблемы с настройкой соединения

Если вам не удается установить соединение с провайдером, то надо, естественно, искать причину. Попробую дать несколько советов, как действовать в такой ситуации.

Первый совет, который на этот случай дают "классики": смотрите файлы протоколов!

Во-первых, при запуске kppp установите во включенное состояние переключатель **Монитор подключения** в главном окне kppp. В окне монитора можно увидеть, что ожидает получить провайдер, и какую информацию посылает ваш компьютер. Зачастую можно подправить сценарий соединения, пользуясь только информацией, отображаемой в этом окне.

Во-вторых, надо запустить команду kppp так, чтобы она как можно подробнее протоколировала свои действия. Для этого снова войдите в окно настроек соединения kppp и задайте опцию запуска **debug** на вкладке **Дозвон.**

Кроме того, впишите следующие две строки в файл /etc/syslog.conf:

```
daemon.*    /dev/console
daemon.*    /var/log/kppp.log
```

(обратите внимание на то, что между двумя частями записи в каждой строке должен быть хотя бы один символ табуляции). После внесения изменений в файл /etc/syslog.conf выполните команду kill -HUP <pid>, где <pid> — идентификатор запущенного в это время процесса syslogd. По этой команде syslogd перечитает свой конфигурационный файл. Следствием выполненных вами действий будет то, что pppd будет выдавать сообщения о своих действиях на консоль и записывать эти же сообщения в файл /var/log/kppp.log. Его и смотрите!

Искать в этом файле надо сообщения, начинающиеся на:

- pppd[NNN]: Connected... — означает, что скрипт соединения завершился успешно;
- pppd[NNN]: sent [LCP ConfReq — **Сообщение О том, ЧТО pppd** пытался начать диалог с удаленным сервером;
- pppd[NNN]: rcvd [LCP ConfReq — **Сообщение О том, ЧТО pppd получил** ответ (negotiation frame) от удаленного сервера;
- pppd[NNN]: ipcp up — означает, что pppd дошел до той точки, где, по его мнению, соединение готово для передачи по нему IP-трафика.

Если вы не находите строки с сообщением rcvd, то у вас серьезные проблемы с установлением соединения (например, по каналу не передаются все 8 битов). В этом случае может оказаться полезным записывать в файл протокола весь поток байтов, передаваемых между вашим компьютером и удаленным сервером. Для этого измените две упоминавшиеся выше строки в файле syslog.conf следующим образом:

```
daemon.*,kern.*          /dev/console
daemon.*,kern.*          /var/log/kppp.log
```

и снова перезагрузите демон syslog, как это было сказано выше. Затем запустите pppd с опцией kdebug 25. Теперь все символы, передаваемые по каналу PPP-соединения, будут записываться в файл протокола.

Просмотрите этот файл в поисках сообщений, похожих на следующее:

```
ppp_toss: tossing frame, reason = 4
```

Оно означает, что программа PPP не успевает обрабатывать пакеты от удаленной машины. Это может быть потому, что ваш центральный процессор не способен принимать символы с последовательного порта с той скоростью, с которой они поступают. Если в приведенной выше строке указана причина (Reason), отличная от 4, это может свидетельствовать о других сбоях в работе последовательного порта.

На начальном этапе установления соединения вы можете увидеть одно или несколько сообщений, содержащих строку bad fcs. Это говорит об ошибках в контрольной сумме получаемых PPP-пакетов, и обычно случается в начале сессии, когда удаленная система посылает сообщения типа hello this is the XYZ company. Если такие сообщения продолжают появляться тогда, когда соединение уже установлено, это говорит о шуме в телефонной линии и сбоях в передаче пакетов.

Может оказаться, что на сервере провайдера установлен Remout Access Server (RAS), который настроен на использование алгоритма аутентификации MS CHAP 80. Вы можете определить, запрашивает ли сервер установление подлинности, используя этот алгоритм, при анализе файла протокола pppd. Если сервер запрашивает установление подлинности по MS CHAP, то вы увидите строки типа:

```
rcvd [LCP ConfReq id=0x2 <asyncmap 0x0> <auth chap 80> <magic 0x46a3>]
```

Если все равно не работает (куда обратиться за помощью)

Если, несмотря на все ваши усилия, настроить соединение с провайдером и выйти в Интернет все равно не удается, то необходимо обратиться за помощью к администратору сервера у интернет-провайдера. Кроме того, вы можете обратиться за помощью в один из форумов или списков рассылки, например, на **www.linux.ru.** Но, прежде чем посылать мольбу о помощи, соберите (и сообщите в своем письме) следующую информацию:

- версию используемого вами ядра;
- версию используемого вами пакета PPP;
- какими командами вы запускаете PPP-сессию;
- файл протокола сессии, запускавшейся с опцией debug;
- тип ПО PPP, установленного на сервере провайдера.

И вообще любую информацию, которая, по вашему мнению, может оказаться полезной для решения вашей проблемы.

Теперь еще пара советов в стиле "вопрос-ответ", заимствованных в списке рассылки **blackcat-list@geon.donetsk.ua**:

- Вопрос:

  При подъеме PPP в /var/log/messages появляются такие строчки:

  ```
  modprobe: Can't locate module ppp-compress-21
  modprobe: Can't locate module ppp-compress-24
  modprobe: Can't locate module ppp-compress-26
  ```

О Ответ:

Добавьте в /etc/conf.modules:

```
alias ppp-compress-21 bsd_comp
alias ppp-compress-24 ppp_deflate
alias ppp-compress-26 ppp_deflate
```

❐ Вопрос:

kppp звонит, отдает имя, пароль и сразу отваливается, в протоколах пишет следующее:

```
Sep 12 18:10:34 dimon pppd[1410]: By default the remote system is
required to authenticate itself
Sep 12 18:10:34 dimon pppd[1410]: (because this system has a default
route to the internet)
Sep 12 18:10:34 dimon pppd[1410]: but I couldn't find any suitable
secret (password) for it to use to do so.
Sep 12 18:10:34 dimon pppd[1410j: (None of the available passwords
would let it use an IP address.)
```

Кто виноват и что делать?

❐ Ответ:

- Проверьте правильность ввода имени и пароля.
- Попробуйте в /etc/ppp/options добавить параметр noauth.

На этом закончим рассмотрение программы kppp. Ей было уделено очень много места в этой главе, но я считаю, что это оправдано, поскольку эта программа составляет основу вашего соединения с Интернетом. Только в том случае, когда вам удалось такое соединение установить, вы можете пользоваться браузерами, электронной почтой и другими благами Интернета, о которых я кратко и расскажу в следующих разделах.

14.4. Браузеры Интернета

14.4.1. Путешествия по Интернету с помощью программы lynx

Хотя большинство из нас привыкло путешествовать по WWW с помощью браузеров, работающих в графическом режиме, не стоит окончательно забывать и те программы, которые работают в текстовом. Именно такой программой и является lynx. Несомненными преимуществами этой программы является то, что она может работать и на "слабых" компьютерах, а также ускорение загрузки страниц в силу того, что не загружаются картинки. Конечно, последнее качество не всегда можно считать преимуществом, но если

вы заглядываете на страницу только для оценки ее содержания, то, возможно, стоит воспользоваться lynx. Тем более, если ничто не мешает загрузить ее параллельно с графическим браузером.

Если вы не установили lynx в процессе инсталляции системы, то смонтируйте дистрибутивный диск CD-ROM и установите lynx командами

```
[root]# cd /mnt/cdrom/RedHat/RPMS/
[root]# rpm -i lynx-2.8.2-2.i386.rpm
```

(естественно, цифры у вас могут быть другими, в зависимости от того, какая версия программы находится на диске).

После инсталляции программы можно ее запустить либо в одном из виртуальных терминалов, либо в консольном окне в графической оболочке. При первом запуске появится окно, напоминающее то, которое изображено на рис. 14.22.

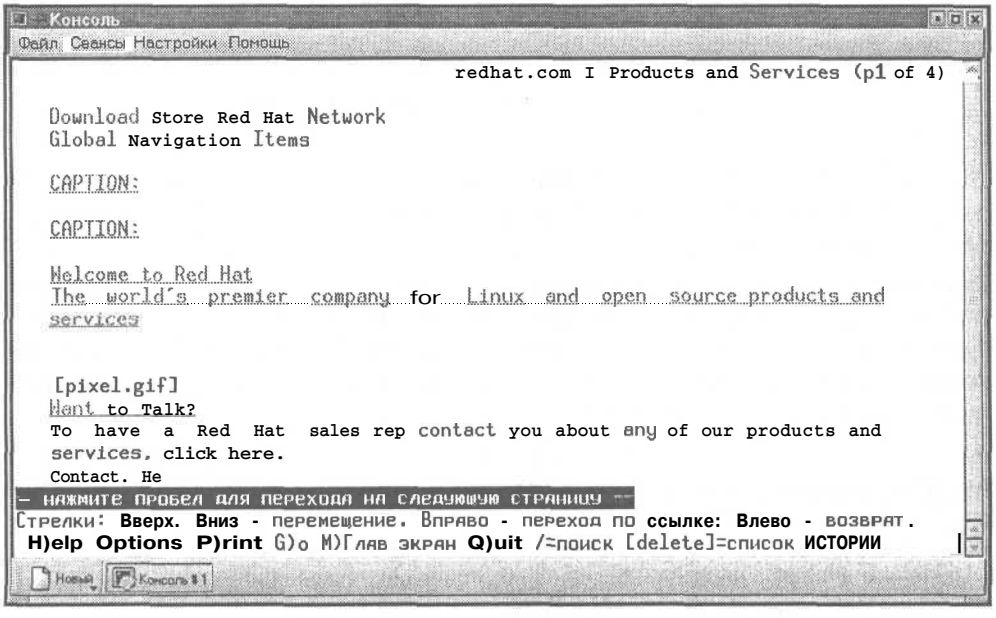


Рис. 14.22. Окно браузера lynx после первого запуска

Обратите внимание на подсказку в двух нижних строках окна (на белом фоне), а также в третьей снизу строке (на синем фоне). Сразу же этой подсказкой и воспользуемся для того, чтобы настроить программу: нажмите клавишу <O> (латинское) для того, чтобы получить доступ к опциям программы. Окно примет вид, изображенный на рис. 14.23.

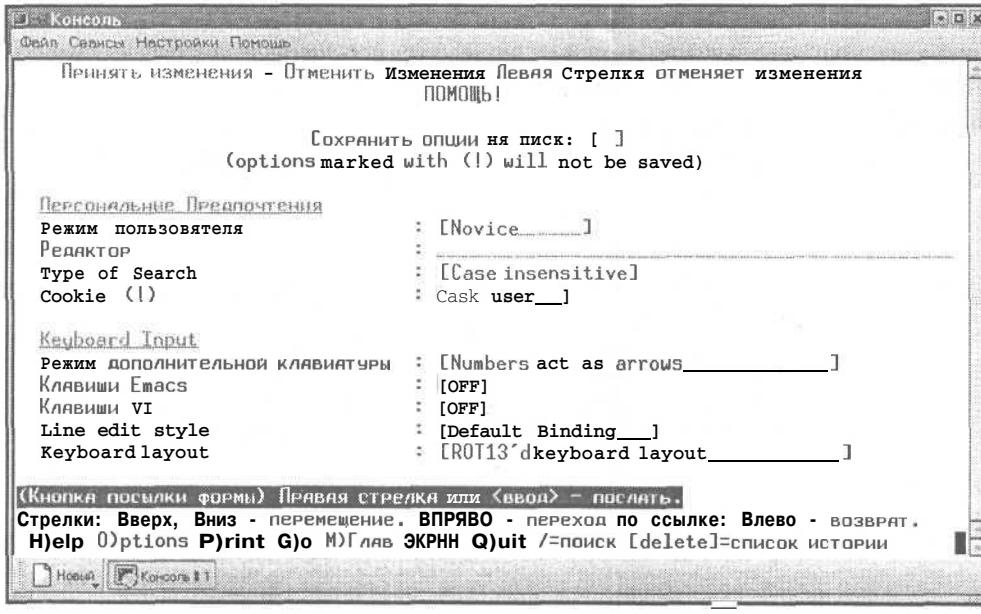


Рис. 14.23. Меню опций программы lynx

Переход от параметра к параметру в этом окне осуществляется по тем же принципам, что и переходы по ссылкам при путешествиях по Интернету, так что запоминайте!

Гипертекстовые ссылки выделены в тексте синим цветом, а текущая (активная) ссылка выделена красным цветом. Клавиши <↑> и <↓> используются для перемещения от ссылки к ссылке, клавиши <Enter> и <→> вызывают переход по активной ссылке, клавиша <←> — возврат к предыдущей странице.

Давайте, для примера, изменим значение параметров **Display character set и Assumed document character set(!)**. Для этого с помощью клавиши <↓> сместим подсветку (красную) на первый из этих параметров и нажмем клавишу <→> или <Enter>. Появится выпадающий список, в котором надо выбрать, например, **Cyrillic (koi8-r)**. Выбор осуществляем теми же клавишами <↓> и <↑>, после чего нажимаем <Enter>. Аналогичным образом параметру **Assumed document character set(!)** присваивается значение **koi8-r.**

Значения некоторых параметров меняются путем прямого редактирования соответствующей строки. Примером такого параметра является **Editor** или **Preferred document language.**

После того как вы задали таким образом значение всех параметров, надо проследовать по ссылке **Accept Changes,** чтобы сохранить эти значения в файле настройки lynx.

Для того чтобы открыть какую-либо интернет-страницу, надо нажать клавишу <G>. В третьей снизу строке окна lynx появится приглашение на ввод URL. Вводите нужный вам адрес, нажимаете клавишу <Enter> и страница открывается. Выглядит она, конечно не так, как в графических браузерах. Вместо картинок даются только имена соответствующих файлов (в квадратных скобках), таблицы преобразованы в последовательный текст, и т. д. Посмотрите для сравнения, как выглядела 15.03.2000 г. главная страница известного сервера IXBT.Hardware.ru в графическом браузере Netscape (рис. 14.24) и в lynx (рис. 14.25).

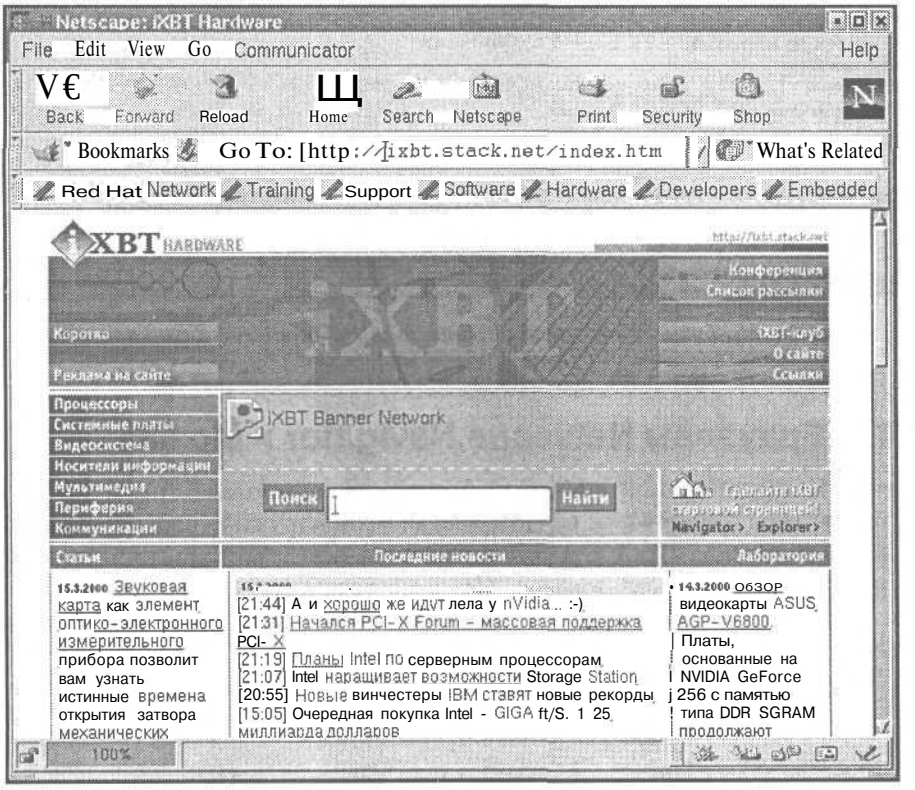


Рис. 14.24. Главная страница сервера IXBT.Hardware.ru в графическом браузере Netscape

Конечно, графический интерфейс и возможность использования таблиц существенно повышают информативность страницы и облегчают ее восприятие. Так что в большинстве случаев вы будете пользоваться каким-либо из графических браузеров (к рассмотрению которых мы сейчас перейдем). Но не забывайте и о существовании lynx, возможно он вам кое-где пригодится!

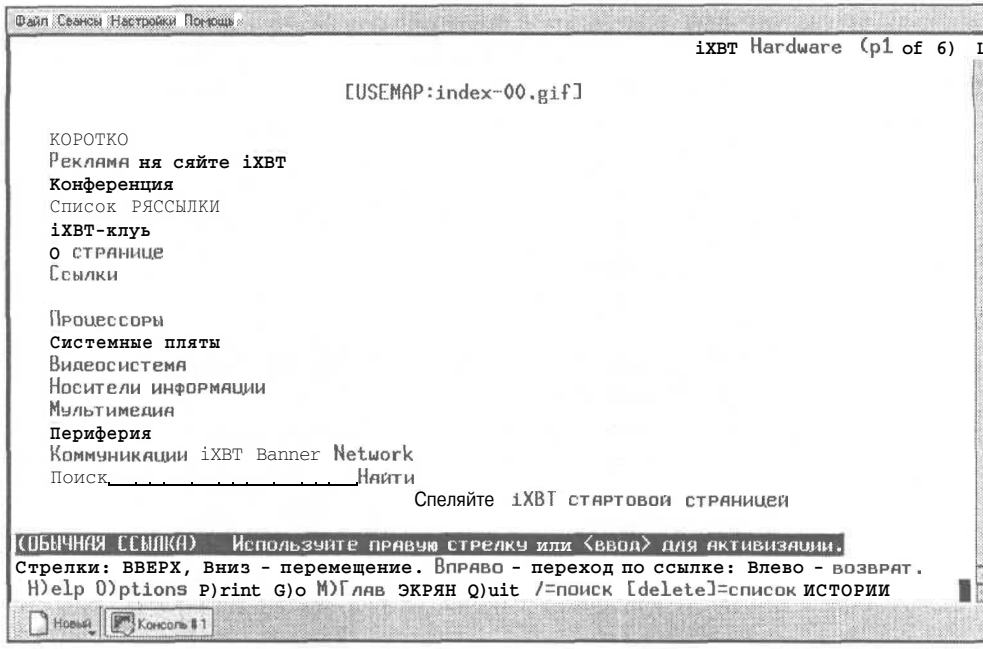


Рис. 14.25. Главная страница сервера IXBT.Hardware.ru в браузере lynx

14.4.2. Браузеры Netscape Navigator и Mozilla

Netscape Navigator — мой любимый браузер. Я пользовался им и тогда, когда работал под Windows, и с удовольствием обнаружил, что он включен в состав дистрибутива Black Cat 5.2, а также во все другие дистрибутивы, которыми я до сих пор пользовался. Поэтому установка его не вызывает проблем: достаточно смонтировать дистрибутивный диск CD-ROM и дать команды

```
[root]$ cd /mnt/cdrom/RedHat/RPMS/
[root]$ rpm -Uhv netscape-common-4.70-1bc.i386.rpm
[root]$ rpm -Uhv netscape-communicator-4.70-1bc.i386.rpm
```

Описанию этого браузера и приемов работы с ним посвящены целые книги, поэтому здесь нет необходимости (да и возможности) приводить подробное описание. На рис. 14.24 вы видите в общем-то привычное (для тех, кто использовал Netscape и раньше) главное окно браузера.

Трудности в использовании Netscape Navigator версии 4.70, насколько мне известно из листа рассылки blackcat-list, были связаны с русификацией программы и подключением Java. На рис. 14.24 видно, что команды меню даны на английском. В общем-то, у большинства пользователей это не вы-

ывает затруднений. Тем более, что при установке последних версий дистрибутивов (я имею в виду ALTLinux Junior 1.0 и Red Hat Linux CE 7.1) инсталлируется уже полностью русифицированная версия 4.74 этого браузера.

Надо сказать, что в последнее время появились очень противоречивые сведения о дальнейшей судьбе этого браузера. С одной стороны, в новостях сообщалось, что браузер перестал поддерживаться и развиваться фирмой Netscape Communication Corporation. С другой стороны, выходят как новые версии ветки 4.*xx* этого браузера, так и уже второй вариант (6.2) шестой версии. Версию 6 (правда, в варианте 6.0 для Windows) я пробовал установить, но она мне показалась настолько тяжеловесной, что я вернулся к версии 4. Под Linux же мои попытки скачать и установить версию 6 окончились неудачей из-за того, что инсталляцию надо было проводить по сети, а связь то и дело срывается. Но пока что в каждом из встречавшихся мне дистрибутивов включена одна из версий Netscape Communicator, и эти версии меня вполне устраивают. Поэтому до сих пор основным браузером под Linux для меня является Netscape Navigator.

Но после установки ALTLinux Junior 1.0 обнаружилось, что кроме Netscape Communicator установлен и браузер Mozilla (на рис. 14.26 вы видите, что он сообщил мне о себе и своей версии). Первое знакомство с этой программой показывает, что она очень похожа на Netscape Navigator. Конечно, некоторые изменения в организации меню имеются (Вы это можете и сами увидеть, сравнив рис. 14.24 и рис. 14.27), но если вы имеете опыт работы с Navigator, то легко освоите и Mozilla.

Сходство Netscape Navigator и Mozilla далеко не случайно. В апреле 1998 года независимая в то время компания Netscape Communications решила переработать свой популярный браузер Navigator, переведя его на механизм отображения HTML-страниц (рендеринга) Gecko и превратив в проект с открытым исходным кодом. Тем самым компания привлекала к разработке браузера широкий круг разработчиков, заинтересованных в модифицировании и распространении исходного кода Netscape. Исходный код Mozilla был опубликован в 1999 году. На Web-сайте **http://www.mozilla.org** все желающие получили доступ к исходным кодам, возможность внесения изменений, участие в группах новостей разработчиков, возможность получения и распространения информации, связанной с браузером.

Проект Mozilla выпустил несколько предварительных версий или выпусков (build) браузера. Каждый такой build отличается от предыдущего в лучшую сторону: добавляются новые функции, исправляются ошибки и пр. Некоторые из них получили особое название — Milestone Builds ("верстовые столбы"). Эти выпуски обозначаются буквой M с добавлением номера очередной "версты" (например, M18) и являются своеобразными вехами в истории развития проекта.

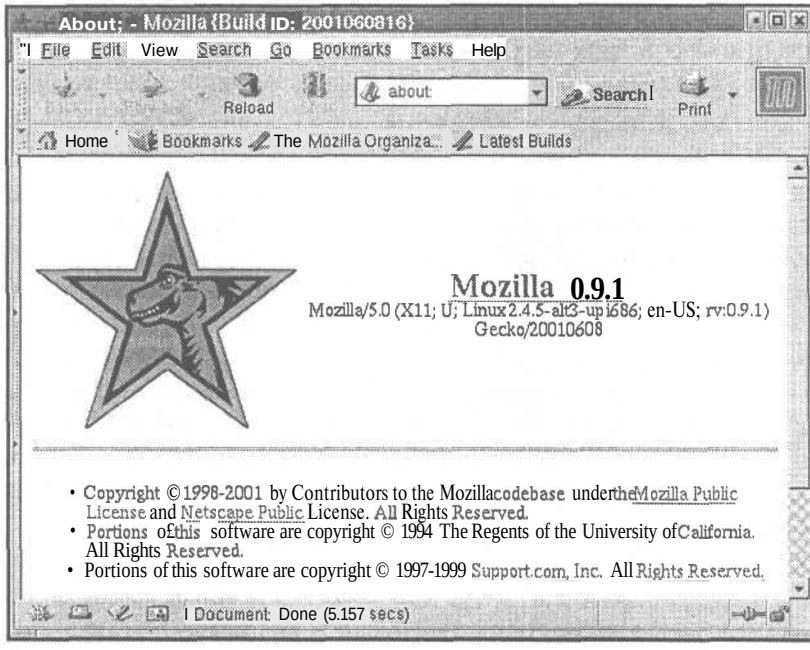


Рис. 14.26. Окно браузера Mozilla

Начиная с M18 проект Mozilla разветвился. Дело в том, что браузер Mozilla, включая механизм рендеринга Gecko, был положен в основу разработки шестой версии Netscape (права на который перешли к компании AOL Time Warner), и в то же время продолжалась разработка самостоятельного браузера Mozila 1.0. Версия 6.0 браузера Netscape 6 на основе Mozilla, которая появилась в октябре 2000 года, подверглась резкой критике, но более поздняя версия 6.1 получила положительные отклики.

В ноябре 2001 года на сайте Mozilla.org появилась бета-версия браузера Mozilla с номером 0.9.6. В Mozilla 0.9.6 исправлено несколько ошибок и добавлены новые функции, такие как отображение значков страниц в поле адреса и поддержка форматов изображений BMP и ICO на разных платформах. В числе других нововведений, произведенных в версии 0.9.6, стоит отметить функцию просмотра страницы перед печатью (Print Preview), расширенные опции настройки вида Web-страниц в версии браузера для MacOS, а также улучшенные контекстные меню. Благодаря новой функции пользователи Mozilla теперь могут выделить любое слово на Web-странице и вызвать поиск по этому слову в поисковой машине. Кроме браузера, в Mozilla входят почтовая программа, программа чтения новостей Usenet и другие компоненты. Дистрибутив программы немного увеличился, его размер составляет 9,1 Мбайт. Скачать его можно с Web-сайтов **http://www.mozilla.org или http://www.mozilla.ru.**

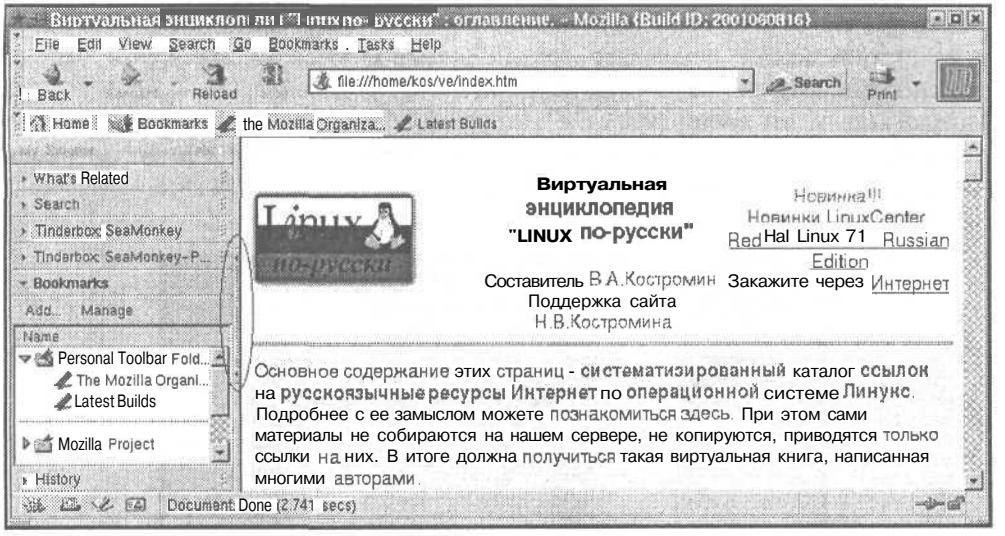


Рис. 14.27. Окно программы Mozilla с отображением **My Sidebar**

Браузер Mozilla уже широко используется, хотя все еще сохраняет статус бета-версии. По словам разработчиков, следующие две бета-версии будут нацелены на повышение производительности и стабильности программ электронной почты и чтения новостей. Но дата выпуска окончательной версии 1.0 до сих пор остается тайной. От разработчиков не удается получить более-менее точной информации о сроках. "Если все будет хорошо, версия 1.0 будет выпущена сразу после выхода версии 0.9.9", — сообщается в "Манифесте", посвященном грядущей версии 1.0, на сайте Mozilla.org. Пока это единственная точная информация.

В то же время исходный код проекта Mozilla уже и сейчас высоко оценен сообществом разработчиков. Механизм рендеринга страниц Gecko, например, применяется в качестве ядра других браузеров, таких как браузер Galeon, разработанный в рамках проекта с открытым исходным кодом GNOME.

Внешне отличия Mozilla от Netscape Navigator не очень существенны. В первую очередь бросается в глаза появление боковой панели, которая называется **My Sidebar.** Вы видите ее на рис. 14.27. При запуске Mozilla панель может не отображаться в окне программы. Но ее всегда можно вызвать (или убрать из вида) простым щелчком мыши по узенькой вертикальной полоске, которая расположена на левой границе панели, отображающей текст просматриваемой странички (на рис. 14.27 она выделена овалом).

My Sidebar представляет собой набор вкладок в левой части окна, каждая из которых содержит "горячую" информацию с какого-либо ресурса Интернета,

обновляемую с некоторой периодичностью. Это могут быть последние новости, прогноз погоды, календарь, ваша адресная книга и многое другое. Другими словами, My **Sidebar** — это настраиваемый пользователем список ссылок (представляющий собой развитие концепции "закладок") на те ресурсы Интернета, к которым вам приходится часто обращаться. Естественно, что имеются механизмы добавления и модификации вкладок (рис. 14.28).

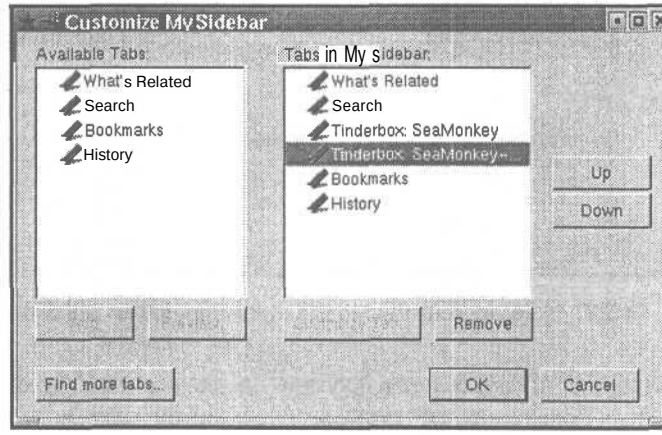


Рис. 14.28. Формирование состава вкладок на панели **My Sidebar**

Между прочим, существует сайт **Sidebar.Ru,** посвященный боковой панели Mozilla & Netscape 6. На этом сайте представлены примеры новых закладок для My **Sidebar.**

Таким образом, хотя браузер Mozilla еще не вышел из стадии бета-тестирования, он кажется очень перспективной разработкой. Не зря компания Red Hat объявила, что когда выйдет версия Mozilla 1.0, она войдет в качестве браузера по умолчанию в состав популярного дистрибутива Red Hat Linux и заменит в этом дистрибутиве браузер Netscape 4.*xx*. Впрочем, Red Hat пока не отрицает и возможности использования Netscape версии 6 (хотя отмечает, что у этого варианта есть некоторые недостатки из-за того, что его код не является полностью открытым), а также рассматривает в качестве возможного варианта браузер Konqueror, разрабатываемый в рамках проекта KDE.

14.4.4. Файловый менеджер Konqueror

Если вы установили графическую среду KDE, то устанавливать отдельную программу-браузер, вообще говоря, нет никакой нужды, поскольку в качестве браузера вполне может служить файловый менеджер Konqueror, по умолчанию установленный в составе KDE. Он позволяет как просматривать со-

держимое каталогов и файлов на локальных дисках (рис. 14.29), так и интернет-ресурсы (рис. 14.30).

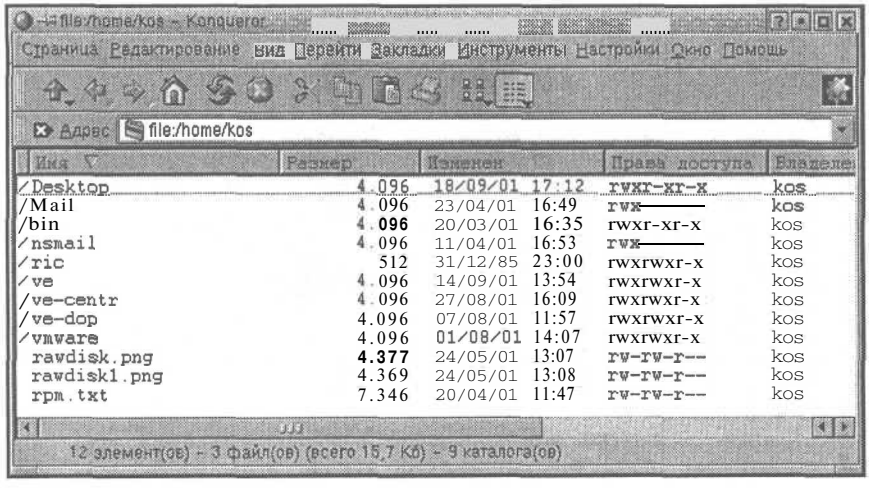


Рис. 14.29. Konqueror при просмотре локального каталога

Для просмотра Web-страничек достаточно ввести в поле **Местоположение** URL ресурса, и вы получите к нему доступ (если, конечно, у вас настроен выход в Интернет). Между прочим, на рис. 14.30 специально приведен вид той же самой Web-страницы, которая использовалась для иллюстрации рассказа о браузерах lynx и Netscape Navigator (см. рис. 14.24 и 14.25). Мне показалось, что с локальными файлами Konqueror работает даже быстрее, чем Netscape Navigator, особенно при загрузке очень больших файлов (то есть при нехватке оперативной памяти).

Еще одним преимуществом этого навигатора для некоторых пользователей может оказаться то, что он полностью русифицирован (вы можете видеть это на рис. 14.29 и 14.30). И позволяет выбрать кодировку просматриваемой страницы из очень большого числа вариантов: посмотрите команду меню **Вид | Кодировка документа.**

Что мне особенно нравится в этом браузере, так это то, что Konqueror позволяет легко и удобно просматривать файлы многих форматов. В первую очередь это касается разных текстовых файлов (ASCII, HTML, PS), но также и наиболее распространенных графических форматов (GIF, JPEG, PNG, TIFF, BMP). Правда, для некоторых форматов файлов для просмотра используется вызов специализированных программ на основе ассоциаций имен файлов с соответствующей программой. Но это естественно и, я думаю, вас не удивит. Более того, если для какого-то типа файлов просмотрщик не определен, Konqueror предлагает пользователю задать нужную ассоциацию.

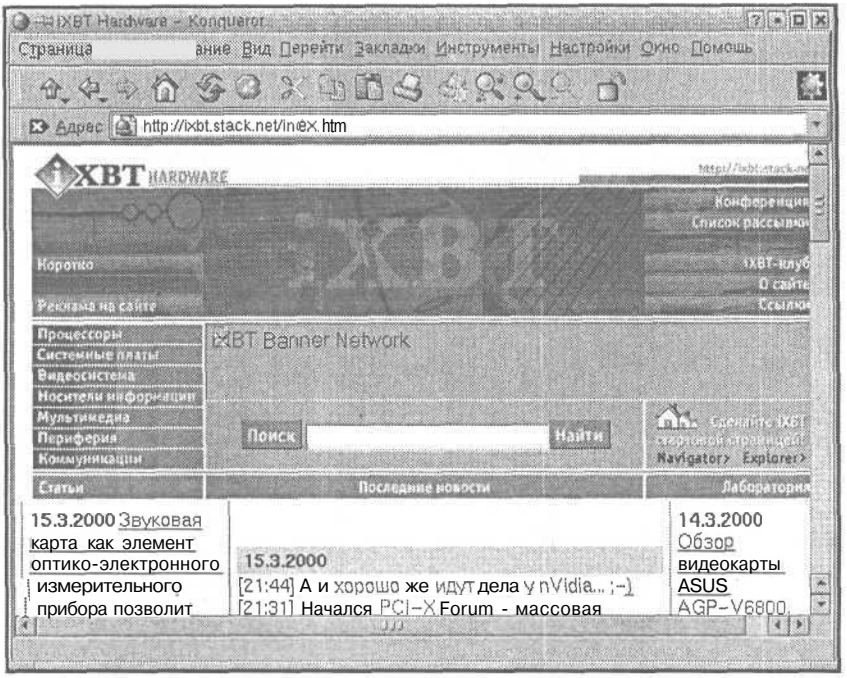


Рис. 14.30. Konqueror при просмотре титульной странички IXBT.Hardware

Вообще говоря, файловому менеджеру Konqueror надо бы посвятить отдельную главу, по объему не меньшую, чем глава, посвященная Midnight Commander. Но пока такой возможности у меня нет, так что придется вам знакомиться с этим браузером самостоятельно (вы можете также прочитать статью [П16.23] *приложения).*

14.5. Электронная почта

Раз уж эта глава посвящена Интернету, то надо уделить соответствующее внимание и программам для работы с электронной почтой, которая представляет собой одно из первых и базовых средств Интернета. Но прежде хочется дать небольшую справку об основных протоколах обработки электронной почты.

В общих чертах работа пользователя с электронной почтой происходит следующим образом: программа-клиент обращается к почтовому серверу, на котором запущено соответствующее серверное приложение, или демон. Демон обрабатывает поступивший запрос, обеспечивая авторизацию пользователя, после чего выполняет требования клиента. Общение демона и программы-клиента может происходить по одному из двух основных протоко-

лов: POP3 (Post Office Protocol, версия 3) и IMAP (Internet Message Access Protocol). Оба эти протокола обеспечивают авторизацию на сервере, считывание списка сообщений, копирование сообщений на компьютер пользователя и удаление их с сервера.

Преимуществом протокола POP3 является его простота, вследствие чего он реализован в большинстве программ-клиентов. Но он имеет и существенные недостатки. Например, если на сервер к моменту выхода на связь поступило несколько писем, то, работая по протоколу POP3, вы не можете предварительно просмотреть список пришедших писем и скачать только некоторые из них. Либо забираете все, либо ничего. К тому же, если в процессе приема почты произошел сбой, то приходится начинать перекачку заново. В отличие от POP3, протокол IMAP обеспечивает выборочный просмотр писем, поиск нужного письма прямо на сервере, позволяет производить манипуляции с удаленными папками, подходит для доступа к отличным от писем данным, например к новостям или документам. Кроме того, в IMAP предусмотрены возможности определения типа данных в письме, выделения определенных частей, поиска по каталогам. Все это уменьшает размер информации, передаваемой с сервера на локальный компьютер, и позволяет обращаться к почтовому серверу с разных компьютеров.

Отметим отдельно, что протоколы POP3 и IMAP не занимаются непосредственно пересылкой почты. Эту процедуру выполняет стандартный протокол SMTP. Он прост и надежен, и с успехом выполняет возложенные на него задачи. Кроме того, при использовании IMAP его работу сопровождает протокол IMSP (Internet Message Support Protocol), который отвечает за конфигурацию работы IMAP.

Клиентских программ обработки электронной почты для Linux существует великое множество. Я до сих пор в основном использовал Netscape Messenger. В составе пакета Mozilla тоже имеется почтовый клиент. Однако я приведу ниже описание не этих программ, а почтового клиента KMail, входящего в состав графической среды KDE. Дело в том, что описание Netscape Messenger можно найти в книгах по Netscape, а Mozilla пока не сильно от него отличается. А вот описаний KMail на русском языке пока нет, хотя программа интенсивно совершенствуется. Я работал с программой KMail версий 1.02 и 1.2 (на момент написания данного текста уже выпущена версия 1.3), и хочется отметить, что с изменением даже во второй цифре номера версии программа существенно повысила свою функциональность. Так что вскоре по своим возможностям она не будет уступать лучшим из клиентских программ обработки электронной почты.

Запустить KMail можно из главного меню среды KDE. В процессе первого запуска программа KMail создает в вашем домашнем каталоге подкаталог Mail, содержащий папки для входящей и исходящей почты, а также "мусорную корзину" (файлы inbox, outbox, sent-mail и trash). Затем KMail

сразу же вызывает окно **Настройка** (рис. 14.31), в котором надо ввести данные, необходимые программе для того, чтобы найти в сети почтовый сервер и добраться до вашего почтового ящика. Окно **Настройка** содержит семь вкладок: **Профиль, Сеть, Внешний вид, Редактор, Заголовки Mime, Безопасность и Разное.** Для того чтобы получить возможность принимать и отправлять почту, вам необходимо ввести данные только на двух вкладках **Профиль и Сеть.**

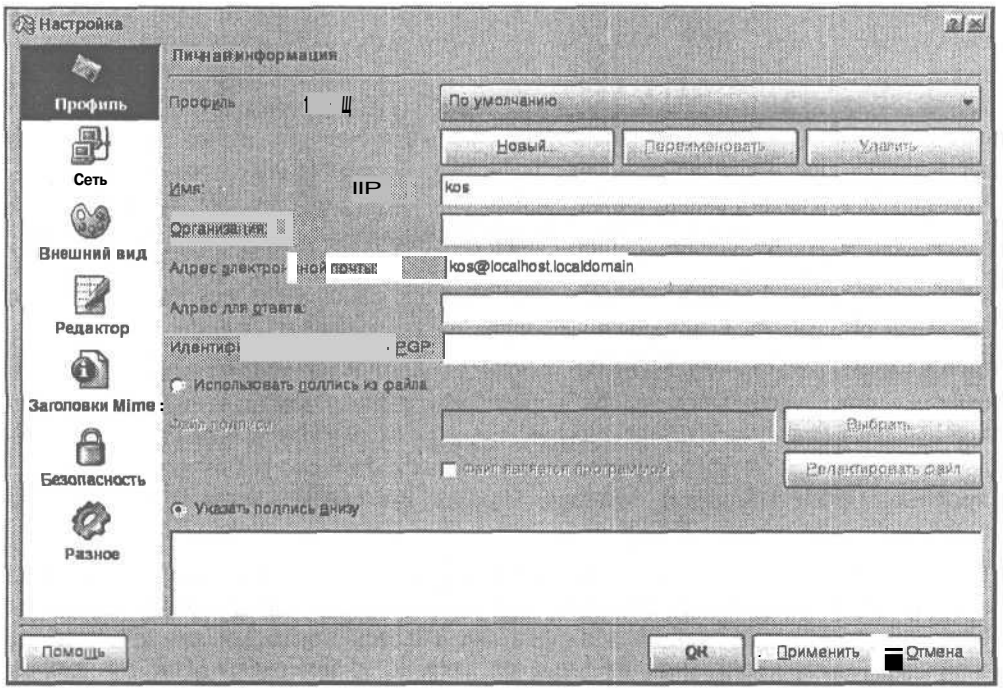


Рис. 14.31. Задание идентификационных данных пользователя

На вкладке **Профиль** вы должны ввести свои идентификационные данные, для того, чтобы KMail была способна правильно получать и отправлять ваши сообщения. Те данные, которые вы введете в поля **Имя** и **Организация,** будут автоматически добавляться в заголовок любого сообщения, которое вы отправляете. В поля **email и Адрес для ответа** введите адрес вашего электронного почтового ящика. В поле **Файл подписи** вы можете указать путь к файлу, содержимое которого будет добавляться в конце каждого отправляемого сообщения. Можно не создавать отдельный файл для сохранения этого текста, а просто ввести нужный текст в поле, имеющееся на вкладке **Профиль.** Имеется также возможность варьировать содержание заключительных

фраз ваших писем (например, вставлять случайно выбранные изречения). В этом случае надо задействовать подпись из файла и установить флажок **Файл является программой.**

Вкладка **Сеть** определяет установки, по которым KMail определяет, как отправлять и где получать ваши сообщения. Эти установки в значительной степени зависят от того, каким образом вы соединяетесь с почтовым сервером.

В поле **Отправка почты** надо указать имя почтового сервера и имя домена, в котором этот сервер находится (рис. 14.32). Если почтовым сервером является программа Sendmail, работающая на вашем компьютере, то вы должны задать ее местоположение. Если же вы отправляете почту на другой почтовый сервер по протоколу SMTP, то требуется указать имя сервера и номер порта. Если вы не знаете, как правильно задать соответствующие настройки, проконсультируйтесь у своего провайдера или системного администратора. Если при настройке выхода в Интернет вы правильно настроили обращение к серверу DNS, то кроме имени сервера никакой другой информации о нем задавать не надо, все остальное будет находиться автоматически. Значение, введенное в поле **Порт,** вам менять, скорее всего, не требуется.

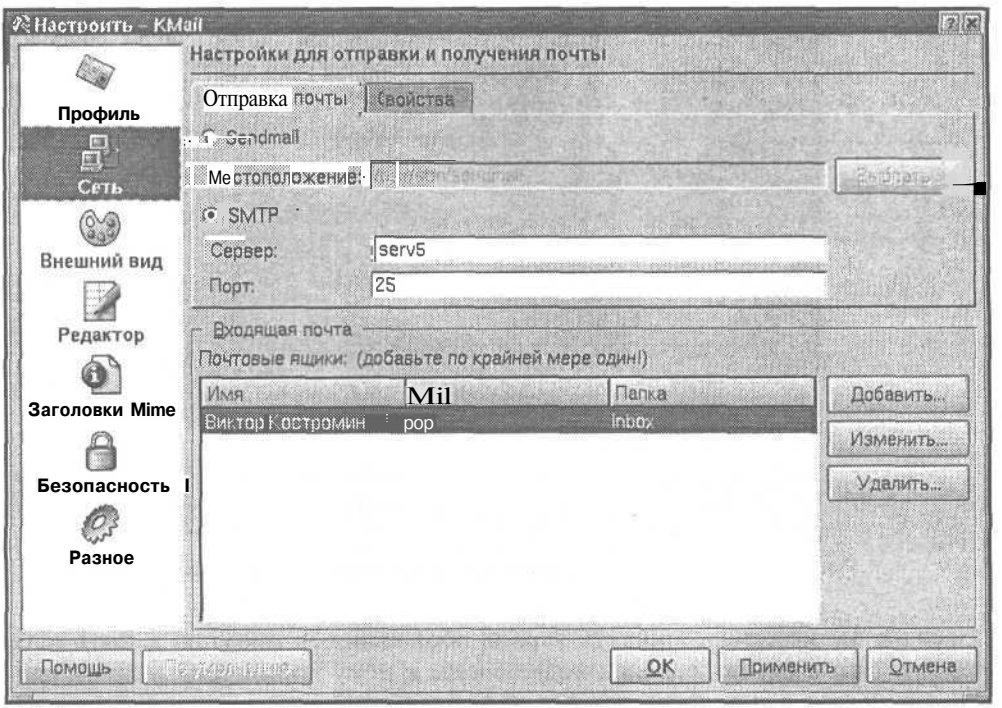


Рис. 14.32. Задание параметров почтового сервера

В поле **Входящая почта** нужно определить хотя бы один почтовый ящик, что делается путем нажатия экранной кнопки **Добавить.** При добавлении почтового ящика программа попросит вас задать его тип: POP3 или локальный (в последнем варианте имеется в виду, что на данном компьютере запущен демон Sendmail и пользователи компьютера обмениваются сообщениями). К сожалению, пока что (в версии 1.2) KMail не может работать с IMAP-серверами. После задания типа ящика программа предложит вам ввести ваше имя, логин, пароль, имя сервера и номер порта. Номер порта (по умолчанию ПО) обычно изменять не стоит. Можно установить опцию сохранения пароля в конфигурационном файле, но делать это не рекомендуется, поскольку пароль там хранится в открытом виде. Зато программа не будет просить ввести его при каждом соединении с сервером. А уж "иметь или не иметь" уверенность в том, что никто ваш пароль не узнает, каждый решает сам. Впрочем, просмотреть конфигурационный файл может только сам пользователь и суперпользователь root, так что в случае персонального компьютера особой опасности здесь не просматривается.

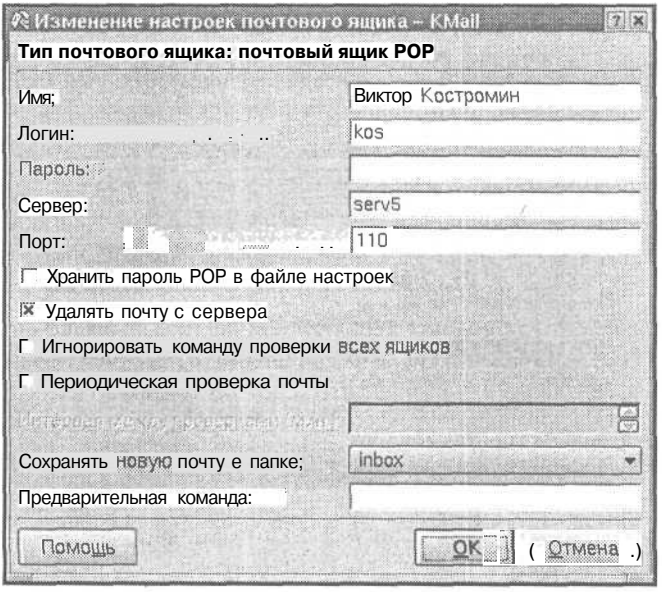


Рис. 14.33. Задание параметров обращения к почтовому ящику

Здесь же вы можете установить период обращения (в минутах) к почтовому серверу (это в случае, если вы подключены к нему постоянно), указать папку, в которую будет поступать вся входящая корреспонденция (по умолчанию -- inbox). Правда, если вы хотите как-то сортировать приходящие письма уже на этапе получения, целесообразнее использовать для этого

фильтры, *а* не простое перенаправление в папку с другим именем. Флажок **Удалять почту с сервера** лучше, на мой взгляд, установить, чтобы ваши письма не занимали зря дисковое пространство сервера. И, наконец, вы можете здесь же задать программу, которую KMail будет запускать перед приемом почты. Только имейте в виду, что необходимо указать полный путь к файлу программы, а также то, что KMail будет ожидать завершения работы этой программы и только затем продолжит свою работу.

Для проверки созданной таким образом конфигурации можно попытаться отправить хотя бы одно тестовое письмо. Давайте закроем окно настроек и вернемся к главному окну программы (рис. 14.34), которое является также окном просмотра почты. Это окно будет в дальнейшем появляться сразу после запуска программы KMail. Оно разделено на три окна (или панели) меньшего размера.

- Окно **Папки** (слева вверху). В этом окне отображается список папок с сообщениями. Для того чтобы выбрать папку, просто щелкните по ней мышкой. Список сообщений, содержащихся в этой папке, будет отображен в окне заголовков.

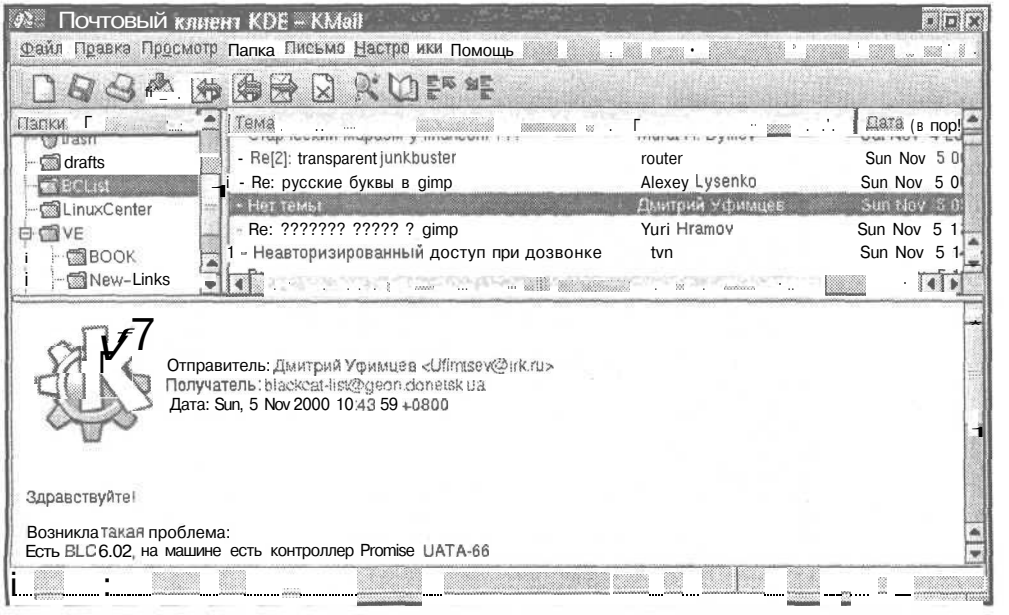


Рис. 14.34. Окно просмотра почты

- Окно заголовков (верхнее правое окно). Это окно содержит информацию (флаг статуса, отправитель, тема и дата отправки сообщения) из заголовков сообщений, содержащихся в выбранной папке. Щелчком мыши по

заголовку вы выбираете сообщение, после чего его содержание отобража ется в окне сообщений. Вы можете отметить сразу несколько сообщений Для этого надо щелкнуть по заголовку первого сообщения, а затем, при нажатой клавише <Shift>, щелкнуть по какому-то другому заголовку. Бу дут выбраны (помечены) эти два сообщения, а также все сообщения, заголовки которых расположены в окне заголовков между первыми двумя.

Порядок, в котором сообщения отображаются в окне заголовков, можно изменить, щелкая мышкой по заголовкам столбцов в окне заголовков сообщений.

- Окно сообщений (нижнее окно). В этом окне отображается заголовок выбранного сообщения и его тело, т. е. собственно содержание сообщения. При просмотре сообщения вы можете пользоваться клавишами <PageUp> и <PageDown> для перехода к следующей странице, или использовать клавиши стрелок для смещения на одну строку.

Однако, если вы пока еще не получали никаких электронных писем, два последних окна у вас пусты. Так что рассмотрение того, как работать с полученными сообщениями, пока отложим и вернемся к отправке нашего первого тестового сообщения. Для того чтобы создать новое сообщение, надо вызвать окно редактора сообщений, что мржно сделать через команду меню **Письмо | Новое письмо,** с помощью соответствующей иконки на панели инструментов или "горячими" клавишами <Ctrl>+<N>. Если вы не задали текст подписи в окне настроек или в файле подписи, появится запрос на ввод имени файла, в противном же случае откроется окно Редактора сообщений, изображенное на рис. 14.35.

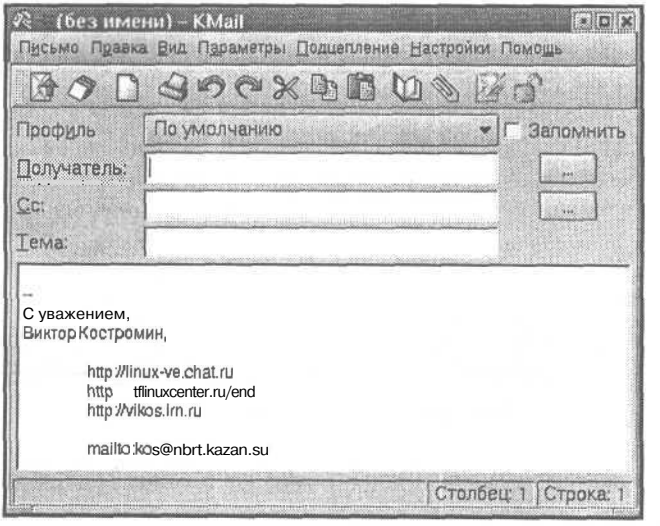


Рис. 14.35. Окно Редактора сообщений

Как видите, вам сразу предоставляется шаблон (или заготовка) нового письма. На рис. 14.35 этот шаблон минимален, он содержит только подпись, но путем соответствующих настроек шаблон можно сделать гораздо изощреннее.

Первым делом введите в верхнем поле (или строке ввода) адрес получателя письма. Если у вас уже имеются записи в Адресной книге, можно вызвать ее с помощью имеющихся рядом с этими полями клавиш с тремя точками, после чего перенести из нее адреса в поля **Получатель** и **Сс** двойным щелчком по нужному адресу. В одно и то же адресное поле можно внести сразу несколько адресов, разделив их запятыми. Рекомендуется указывать полные адреса (вида user@domain.com) даже для локальных пользователей.

После завершения ввода текста отправляем письмо через команду меню **Письмо | Отправить.** Для отправки можно воспользоваться соответствующей кнопкой на панели инструментов (конвертик со стрелкой) или комбинацией клавиш <Ctrl>+<Enter>. Подготовленное письмо можно сохранить в папке черновиков (Drafts), чтобы отправить позже. Если при попытке отправить письмо связи с сервером не было, письмо сохраняется в папке Outbox и будет отправлено в следующем сеансе связи.

К письму можно присоединить один или несколько файлов. Проще всего это сделать с помощью кнопки с изображением канцелярской скрепки на панели инструментов, но можно и через соответствующую команду меню. Появится окно выбора файла. После выбора файла появится еще предложение определить тип файла, вид кодировки и дать краткое описание файла. Когда вы закончите с этим и нажмете кнопку OK, в нижней части окна Редактора сообщений появляется дополнительная панель, отображающая свойства присоединенного файла. Вы еще имеете возможность удалить присоединенный файл, просмотреть его, отредактировать данные о нем или присоединить другие файлы. Все это делается через меню, которое появляется по щелчку правой кнопкой мыши на имени присоединенного файла.

Итак, первое письмо мы отправили. Теперь давайте получим входящую почту. Делается это с помощью команд меню **Проверить почту** или **Проверить почту в ящике** (можно также использовать кнопку на панели инструментов или комбинацию клавиш <Ctrl>+<L>). Перед установлением соединения с сервером программа запросит у вас пароль (рис. 14.36). Если с паролем все в порядке (и других проблем с установлением связи тоже нет), программа осуществляет прием писем. При этом в нижней строке окна программы отображается индикатор процесса загрузки почты с сервера. Если в ходе этого сеанса связи с сервером вы получите то сообщение, которое только что послали сами себе, то вас можно поздравить: почтовый клиент настроен и работает! Если же вы получите сообщение об ошибке, убедитесь, что сетевое подключение работает, перепроверьте установки, которые вы сделали, для чего снова вызовите команду **Настройки | Параметры.**

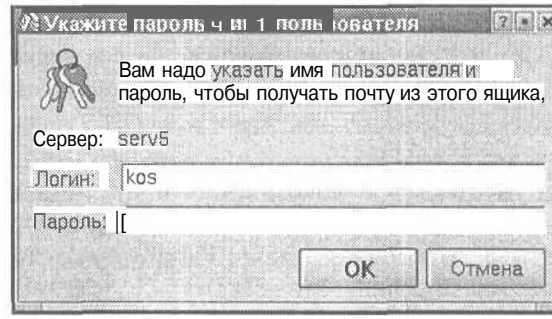


Рис. 14.36. Запрос пароля доступа к почтовому ящику

После получения почты надо ее просмотреть. Для этого заглянем в папку Inbox (Входящие). Для принятых сообщений в ней установлены различные значения флага статуса. Статус может принимать следующие значения:

- **Новое** (красная точка, сообщение выделено красным цветом) — сообщение получено в первый раз и еще не прочитано;
- **Непрочитанное** (зеленая точка, сообщение выделено голубым цветом) — сообщение уже было получено с сервера по крайней мере один раз, но еще не прочитано;
- **Прочитано** (тире) — сообщение прочитано;
- **Отправлен ответ** (голубая стрелка с загнутым хвостом) — на это сообщение был сформирован ответ;
- **В очереди** (конверт) — сообщение было поставлено в очередь в выходном ящике **(outbox)** и будет отправлено позже;
- **Отправлено** (конверт, повернутый под углом к горизонтали) -- сообщение было отправлено.

Вы можете изменить значения статуса произвольным образом через команду меню **Письмо | Установить статус** или аналогичную команду в выпадающем меню.

Переместив подсветку на первое из писем, помеченных как непрочитанное, вы получаете доступ к его содержимому (в нижней панели окна просмотра). Перемещаться от одного письма к другому удобнее всего с помощью мыши, но можно это делать и с помощью клавиатуры, для чего служат следующие "горячие" клавиши:

- <N> — следующее сообщение;
- <P> — предыдущее сообщение;
- <+> — следующее непрочитанное сообщение;
- <–> — предыдущее непрочитанное сообщение.

После прочтения письмо можно удалить. С большинством писем так и случается. Однако некоторые письма вам, возможно, захочется сохранить. Если оставлять их в папке Inbox, то вскоре писем в этой папке скопится так много, что разбираться с ними будет трудновато. Чтобы избежать этой неприятности, стоит с самого начала сортировать сохраняемую корреспонденцию, раскладывая ее по соответствующим архивным папкам. Предварительно надо создать эти папки. KMail поддерживает возможность создания вложенных папок, так что структуру папок лучше продумать заранее (хотя можно и создавать папки по мере необходимости). Для создания папки воспользуйтесь командой меню **Папка | Создать.** Приводить рисунок я не буду, там все интуитивно понятно.

Для того чтобы переместить прочитанное письмо в какую-то из архивных папок, переместите на это письмо подсветку и воспользуйтесь командой меню **Письмо | Переместить в.** Появится выпадающее меню, содержащее список папок. Если какая-то папка (например, Archiv) содержит вложенные папки, то в этом меню будет фигурировать как имя самой папки, так и, дополнительно, "дети папки" (например, "дети Archiv"). Щелчок по команде "дети" вызовет вложенное меню, в котором будут перечислены вложенные папки. Чтобы переместить письмо в выбранную папку (или подпапку), достаточно щелкнуть мышкой по имени нужной папки. Можно перемещать сообщения и методом перетаскивания их мышкой из окна заголовков в окно папок (удерживая нажатой левую кнопку мыши).

Создать в какой-то папке вложенную или изменить название существующей папки можно также, щелкнув по имени папки правой кнопкой мыши и выбрав соответствующую команду из появляющегося меню. Хочется отметить, что выпадающее меню можно вызвать практически для любого элемента окна KMail (как и KDE вообще), что очень удобно. Если, например, вы вызовете такое меню щелчком по имени папки, вы увидите, в частности, команды **Стереть удаленные письма и Очистить.** Последнее означает, что все письма, находящиеся к этому моменту в папке, будут перемещены в мусорную корзину (Trash). Эта операция аналогична операции удаления одиночного письма, только применяется сразу ко всем сообщениям в данной папке. Впрочем, это не совсем так. Дело в том, что при удалении писем с помощью операции **Удалить,** письмо фактически не удаляется из файла, представляющего данную папку (эти файлы находятся в подкаталоге Mail вашего домашнего каталога). Удаляется ссылка на данное сообщение в индексном файле, и письмо более не отображается в окне просмотра заголовков. Поэтому файлы папок со временем могут стать очень большими. Для того чтобы реально удалить сообщение из файла папки и, тем самым, сократить объем этих файлов, необходимо выполнить упомянутую выше операцию **Стереть удаленные письма.** Операция **Очистить** перемещает все письма из папки в "мусорную корзину" и стирает удаленные письма. Если

операцию **Очистить** проделать по отношению к папке Trash, то ее содержимое будет уничтожено безвозвратно.

В программе KMail для файлов папок с сообщениями используется формат MBOX, один из форматов, широко используемых для этого в UNIX-системах. В таком файле сообщения располагаются одно за другим, и начало очередного сообщения определяется по особой строке, начинающейся словом From (не путайте со строкой From: из заголовка сообщения, содержащей адрес отправителя сообщения). Если вы использовали до сих пор какой-то другой почтовый клиент, вы можете использовать специальные конвертеры для преобразования ранее полученной почты в формат MBOX. В составе дистрибутива ALTLinux Junior имеется специальная программа для импорта папок и адресных книг из нескольких почтовых систем (включая Outlook Express, Pegasus-Mail, Eudora Light). Если у вас другой дистрибутив, поищите конвертеры на сайте **http://kmail.kde.org**.

Описание программы KMail я закончу кратким перечислением тех ее возможностей и опций, которые не были упомянуты выше. Приводить их подробное описание здесь нет возможности, и я надеюсь, что разобраться с ними вы сможете самостоятельно. Итак, программа KMail позволяет:

- принимать письма с нескольких почтовых серверов (и) на несколько разных адресов;
- распечатать письмо не покидая программы (команда **Файл | Печать);**
- производить контекстный поиск письма по различным ключевым словам (команда **Правка | Поиск письма);**
- отображать письма в HTML-формате (команда **Папка | Предпочитать html обычному тексту)** и просматривать письма вместе с заголовками в текстовом виде (команда **Письмо | Просмотр источника);**
- изменять (устанавливать) кодировку для отображаемых писем (команда **Письмо | Установить кодировку);**
- производить проверку правописания создаваемых сообщений;
- подписывать создаваемые письма электронной цифровой подписью (по алгоритму PGP) и шифровать отправляемые письма;
- производить автоматическую сортировку входящих писем по заданным фильтрам.

Закончить я хочу теми же словами, которыми начал эту главу: программа KMail очень быстро совершенствуется и в скором времени будет обладать всеми возможностями, присущими лучшим программам этого класса. Именно поэтому я выбрал ее в качестве почтового клиента под Linux.

Глава 15

Обитание в среде KDE

В этой главе я хочу вкратце показать, как создать на компьютере, работающем под управлением ОС Linux, удобную для пользователя рабочую среду. До такой степени удобную, чтобы вообще можно было отказаться от использования Windows и Windows-продуктов. К сожалению, до недавнего времени эта конечная цель была недостижима. Основная причина этого заключалась в отсутствии программ, "понимающих" форматы Microsoft Office. Дело в том, что большинство пользователей персональных компьютеров пока не стремятся перейти на Linux и продолжают работать в Windows-среде. А общаться с ними необходимо (ибо "нельзя жить в обществе и быть свободным от общества"). Поэтому приходится прибегать к таким средствам, как виртуальные машины (которым посвящена *гл. 18),* чтобы организовать такое общение. Невозможность непосредственного общения с "миром Windows" была почти единственной (наряду со сложностью обновления программных продуктов под Linux) преградой для освоения Linux широкими кругами пользователей. С появлением русифицированных версий пакета OpenOffice.org ситуация существенно изменяется. Теперь имеются все необходимые компоненты для создания под Linux полноценной рабочей среды. Это я и попытаюсь показать в настоящей главе.

Под "удобной рабочей средой" я имею в виду набор программных продуктов, позволяющих решать те задачи, которые обычно возникают в ежедневной работе на компьютере, будь то в Windows, Linux или в любой другой операционной системе (в *разд. 15.2* я попытаюсь конкретизировать понятие "удобной рабочей среды"). В основу такой среды я предлагаю положить интегрированную графическую среду KDE. В составе KDE уже имеется большинство компонентов, необходимых в ежедневной работе "среднего" пользователя компьютера. Впрочем, кое-что требуется доустановить. Но, прежде чем говорить о том, чего в KDE не хватает, давайте кратко рассмотрим, что такое KDE само по себе.

15.1. Основы работы с KDE

KDE — это интегрированная графическая оболочка для Linux (и других версий UNIX), которая в настоящее время включает в себя более 100 графических приложений и поддерживает более 40 различных языков. Она разрабатывается в рамках движения Open Source, т. е. распространяется с

открытыми исходными кодами. KDE позволяет совместить современную функциональность, удобство использования и отличный дизайн с технологическими преимуществами операционной системы класса UNIX. На момент написания этого текста разработчики выпустили версию 2.2.2 этой оболочки. Но приводимое ниже описание и все рисунки соответствуют версии 2.1.2, устанавливаемой из дистрибутива Red Hat Linux 7.1 Cyrillic Edition. Конечно, более поздние версии претерпевают некоторые изменения в сторону улучшения, но принципы построения интегрированной среды сохраняются, так что вы вполне можете использовать данный текст для первоначального знакомства с этой оболочкой. Необходимо, кроме того, отметить, что даже если версия самого KDE у вас будет та же самая, возможны некоторые отличия внешнего вида этой оболочки по сравнению с приводимыми ниже рисунками. Дело в том, что в KDE, как и вообще в Linux, все поддается настройке. И каждый производитель дистрибутива делает настройки по-своему. Имейте это в виду и не пугайтесь. Через некоторое время вы и сами сможете полностью изменить вид экрана после запуска KDE.

15.1.1. Внешний вид

Если вы хоть раз запускали KDE, вы уже знакомы с внешним видом экрана после запуска этой оболочки. Для остальных приведу рис. 15.1.

Как видите, экран можно условно поделить на две части.

Панель в нижней части экрана служит для запуска приложений и переключения между рабочими столами. Среди прочих на ней расположен значок с изображением буквы "K". Этот значок (аналог кнопки Start в Windows) служит для вызова иерархического меню, через которое можно запустить любое приложение из числа входящих в состав KDE, даже если значок приложения отсутствует на панели.

Собственно рабочий стол (Desktop) — это вся оставшаяся поверхность экрана, на которой располагается некоторое число значков, которые тоже могут использоваться для запуска соответствующих им приложений.

KDE поддерживает несколько рабочих столов, переключаться между которыми можно с помощью пронумерованных клавиш, расположенных на панели (на рис. 15.1 изображены четыре такие клавиши, хотя это число можно изменить).

Правее переключателя рабочих столов располагается поле, которое называется *панелью* (taskbar). Оно служит для отображения перечня запущенных в данный момент приложений и переключения между ними. Каждому запущенному приложению соответствует кнопка, щелчок по которой переводит данное приложение в активное состояние.

Для тех, кто привык работать в Windows, может показаться непривычным то, что для запуска приложения достаточно щелкнуть по значку только один

раз. Но к этому быстро привыкаешь, а и при желании можно настроить оболочку на два щелчка.

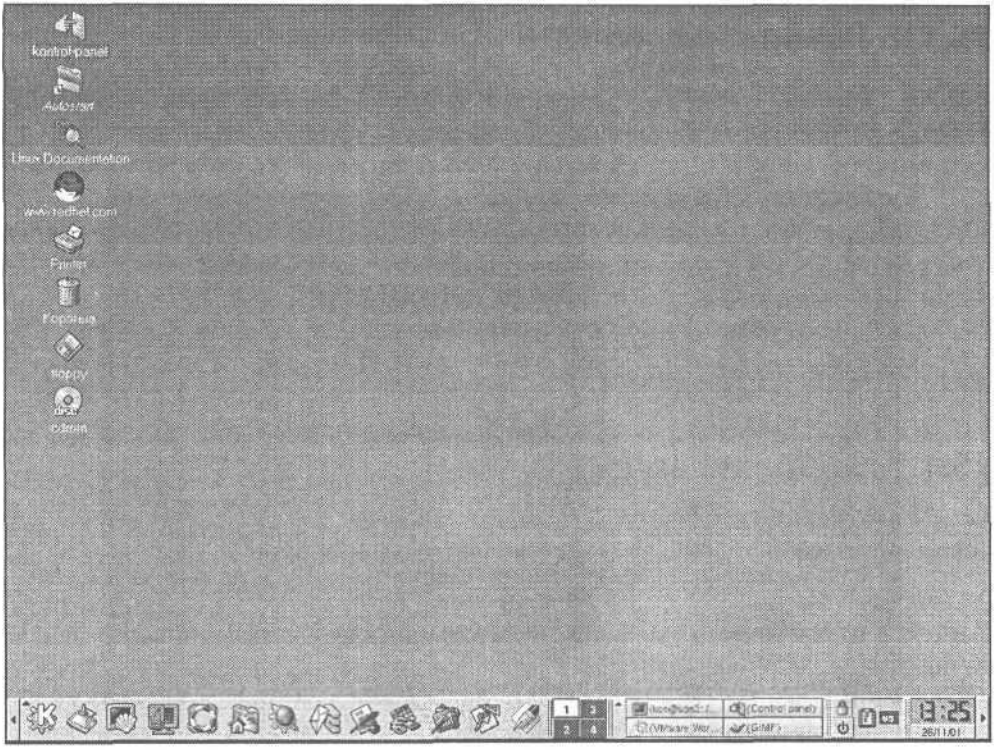


Рис. 15.1. Внешний вид экрана после запуска KDE

В правом конце панели задач находятся часы и небольшая вертикально вытянутая кнопка с треугольником-стрелкой. Такая же кнопка имеется и в левом конце панели. Щелчок по любой из этих кнопок приводит к тому, что панель сворачивается, как бы убегая за границу соответствующей стороны экрана. Видимой остается только такая же кнопка с треугольником. Щелчком по этой кнопке можно вернуть панель на экран.

Если вы подведете указатель мыши к любому значку или кнопке на панели задач и выждете некоторое время, появится подсказка, поясняющая назначение значка или название соответствующей задачи. А если щелкнуть правой кнопкой мыши по любому элементу на экране, в том числе и по пустому полю, появляется меню, в котором можно выбрать одно из действий, применимых к данному элементу. В частности, щелчок по пустому полю приводит к появлению меню настроек рабочего стола.

Панель можно настраивать по своему усмотрению. Но описанием способов настройки мы пока заниматься не будем. Давайте вначале посмотрим на главное меню KDE.

15.1.2. Главное меню KDE

Как вы уже знаете, доступ к главному меню KDE мы получаем, щелкнув по значку с буквой "K" (рис. 15.2).

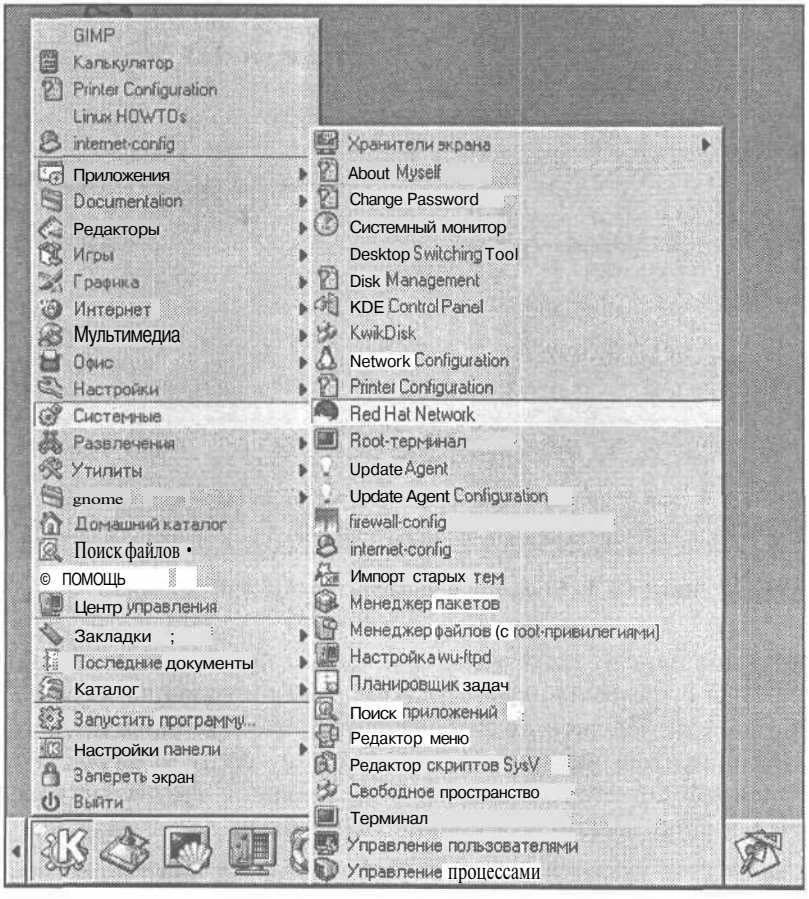


Рис. 15.2. Главное меню KDE

Во-первых, обратите внимание на то, что некоторые элементы меню имеют треугольник после названия. Это говорит о том, что данный элемент вызывает вложенное меню следующего уровня.

Во-вторых, вы можете заметить, что здесь имеются аналоги тех команд, которые вы привыкли видеть в главном меню Windows: **Поиск файлов, Помощь, Запустить программу, Последние документы.** Недаром часто говорят, что из всех интегрированных графических сред KDE наиболее близка к Windows. Что же, хорошим решениям не грех подражать.

Команды **Каталог и Домашний каталог** служат для быстрого перехода в нужный каталог и просмотра его содержимого с помощью файлового менеджера Konqueror (о нем мы уже говорили в *гл. 14* и еще не раз упомянем ниже).

В верхней части главного меню появляется отделенный горизонтальной чертой список часто запускаемых или недавно вызывавшихся команд (естественно, что появляется он не при первом запуске KDE, а после того, как вы немного поработаете в оболочке).

Смысл подавляющего числа команд меню не требует особых пояснений — они вызывают соответствующее приложение, и этим все сказано. Но две из них рассмотреть необходимо: это **Центр управления и Настройки панели.**

15.1.3. Центр управления KDE

Типичный вид окна при работе с Центром управления KDE изображен на рис. 15.3.

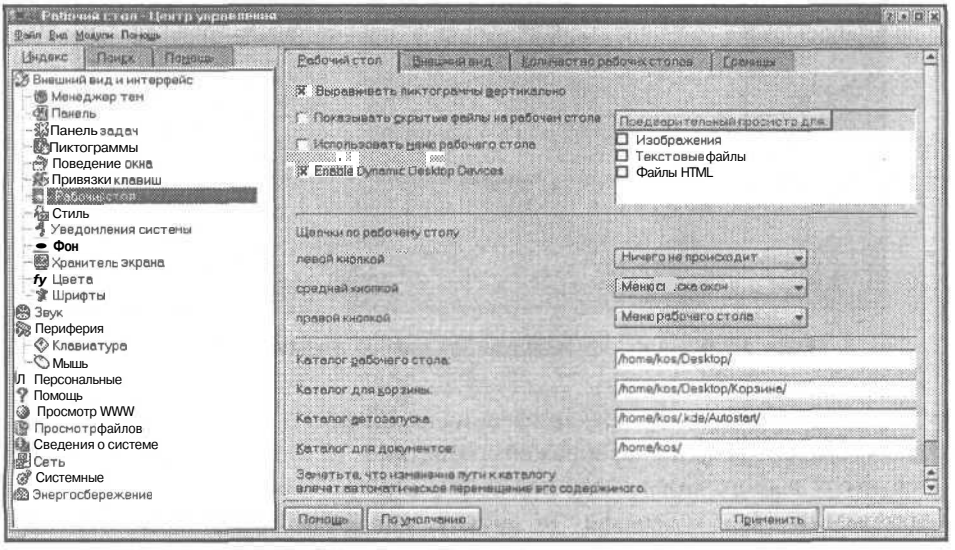


Рис- 15.3. Один из экранов Центра управления KDE

Как видите, слева расположено меню, а справа — поле вкладок, каждая из которых обычно служит для настройки какого-то конкретного элемента

графической среды. Мы рассмотрим несколько таких вкладок, а остальные вы должны будете освоить самостоятельно. Точнее сказать, я просто расскажу вам, как я настраиваю KDE у себя, а вы можете изменить также и те настройки, которых я обычно не касаюсь.

Первое, что я делаю, — настраиваю фон или тему рабочего стола с помощью команд **Фон** и **Менеджер тем** меню **Внешний вид и интерфейс** главного меню Центра управления. На приведенных выше рисунках вы можете видеть результат. Одновременно можно выбрать основные цвета (команда **Цвета)** и стиль оформления графических элементов (команда **Стиль),** однако я обычно оставляю здесь установки по умолчанию.

Далее просто необходимо выбрать (с учетом своих вкусов и, возможно, особенностей зрения) шрифты, которыми будут выводиться различные надписи. На рис. 15.4 вы видите правую панель Центра управления, соответствующую команде **Шрифты.**

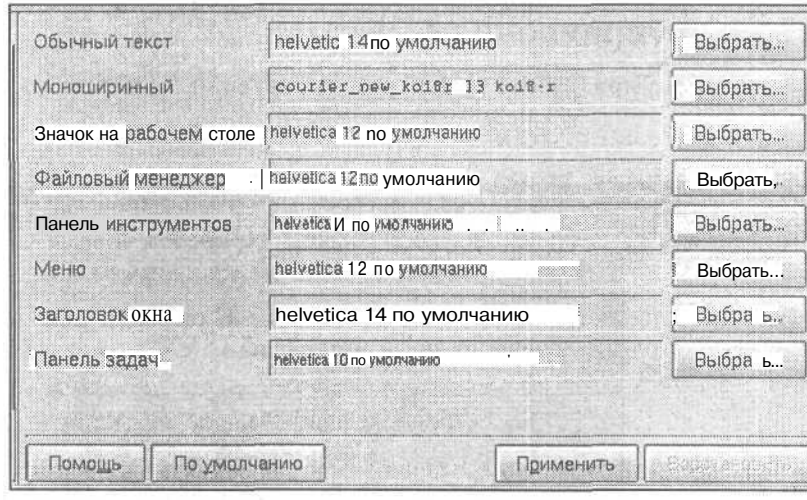


Рис. 15.4. Задание шрифтов для разных элементов экрана

Для изменения какого-либо шрифта надо щелкнуть по экранной кнопке **Выбрать.** Появится окно, изображенное на рис. 15.5, в котором и осуществляется такой выбор. После того как выбор шрифта произведен, вы щелкаете по кнопке **ОК.** Однако это еще не значит, что такой шрифт будет использоваться в KDE. Для того чтобы изменения вступили в силу, надо еще нажать кнопку **Применить** на панели Центра управления (рис. 15.4). Это, кстати, касается не только шрифтов, а любых изменений, производимых с помощью Центра управления.

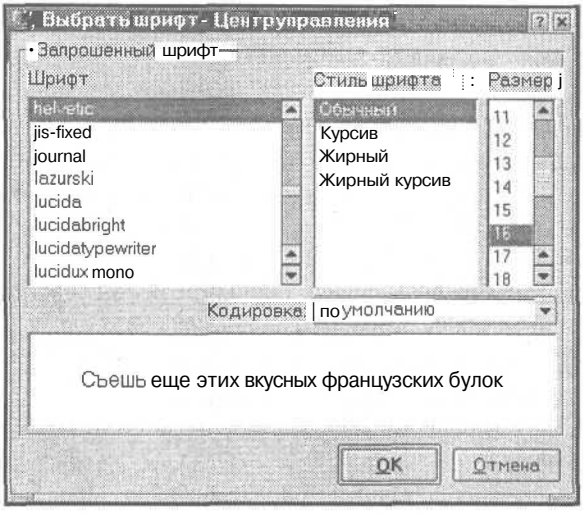


Рис. 15.5. Окно выбора шрифтов

Если вы хотите, чтобы экран гас в то время, когда вы уходите от своего компьютера, а тем более, если вы хотите, чтобы вернуть экран в активное состояние можно было только после ввода пароля, воспользуйтесь вложенной командой **Хранитель экрана.**

Вид и содержание главного меню KDE тоже можно настраивать. Но надо иметь в виду, что главное меню считается элементом панели KDE, а поэтому и команды меню, связанные с его собственной настройкой, надо искать там, где речь идет о настройке панели: в команде **Настройка панели** главного меню или в команде **Внешний вид и интерфейс | Панель** Центра управления KDE.

Добравшись одним из этих способов до вкладки **Меню,** вы увидите картинку, изображенную на рис. 15.6.

Я думаю, что из этого рисунка ясно, какие параметры главного меню можно изменить на этой вкладке. Как видите, состав и содержание элементов меню здесь изменить невозможно. Для этого надо вызвать отдельную программу — Редактор меню KDE, что делается через команду **Настройка панели | Редактор меню** того же главного меню.

Окно программы Редактор меню KDE представлено на рис. 15.7. Здесь можно как создать новую команду меню, так и новое подменю, указать название приложения и имя запускаемого файла, а также назначить запуск программы от имени другого пользователя.

На вкладке **Расширенные** можно назначить "горячую" клавишу, по которой можно будет вызывать приложение, не прибегая к помощи меню и мыши.

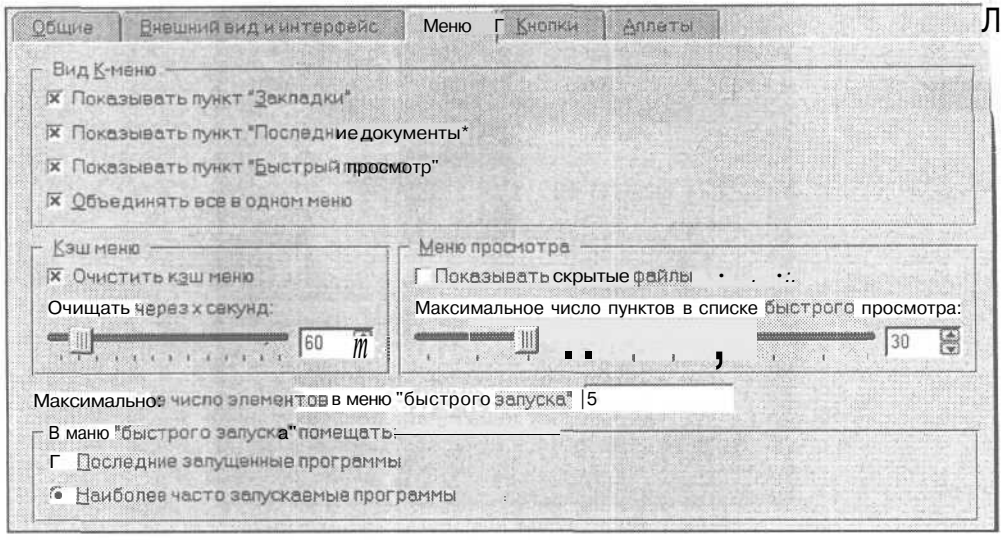


Рис. 15.6. Настройка главного меню

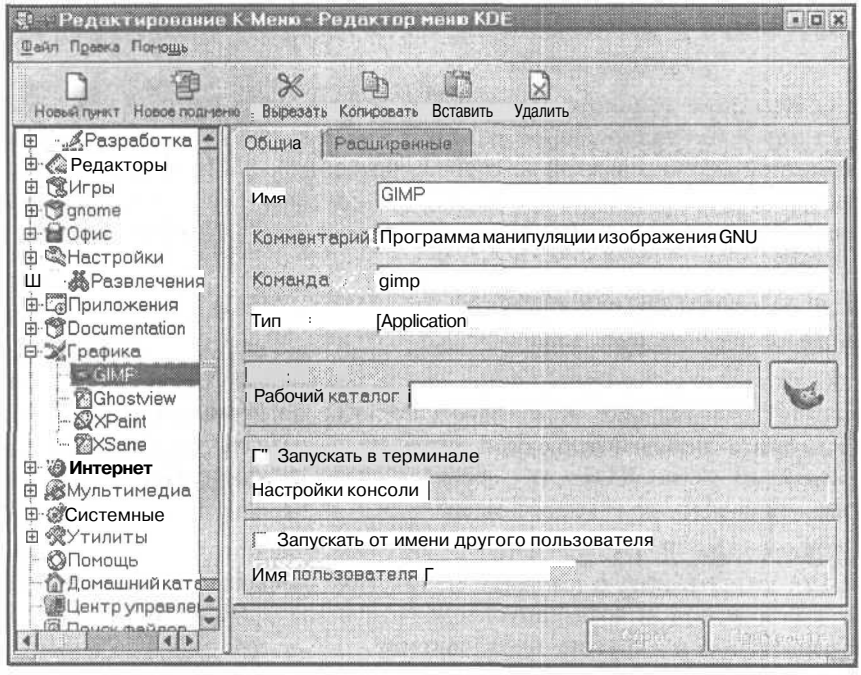


Рис. 15.7. Редактор меню KDE

Ч думаю, что теперь вы сможете при желании перестроить главное меню DE в соответствии со своими вкусами и привычками. Поэтому обратимся вопросу настройки другого важнейшего элемента графического интерфейа KDE — панели.

15.1.4. Настройка панели л значков на рабочем столе

Вы уже знаете, что добраться до настроек панели можно либо через команду **Настройка панели** главного меню, либо через Центр управления KDE. Но имеется и третий вариант. Можно щелкнуть правой кнопкой мыши по свободному полю на панели, и появится выпадающее меню, изображенное на рис. 15.8, в котором тоже имеется команда **Настройки,** и которая по содержанию идентична команде **Настройка панели** главного меню. Если вызвать эту команду, появится окно, изображенное на рис. 15.9.

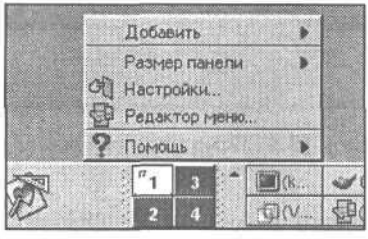


Рис. 15.8. Выпадающее меню для панели

Как видите, с помощью первой вкладки **(Общие)** можно переместить панель к любой границе экрана, изменить ее размер, задать скорость движения при автоскрытии и определить терминальное приложение. На следующей вкладке **(Внешний вид и интерфейс)** задаются еще несколько аналогичных параметров. Поэкспериментируйте! Только не забывайте нажимать кнопку **Применить,** иначе никаких изменений не произойдет. Вкладку **Меню** мы уже рассматривали, а двумя следующими вкладками **(Кнопки и Аплеты) я** не пользуюсь (пробовал поменять фон кнопок и панели, но мне это не понравилось).

Теперь обратите внимание на первую команду **(Добавить)** в меню, изображенном на рис. 15.8. Если переместить на эту команду указатель мыши, то вы увидите, что добавить можно как отдельный элемент (то есть кнопку или иконку) на панель задач, так и четыре вида дополнительных панелей (или четыре вида расширений основной панели). Я не вижу смысла приводить здесь изображения всех этих панелей, расскажу для примера только об одной из них — дополнительной панели задач (на рис. 15.10 она расположена над основной панелью).

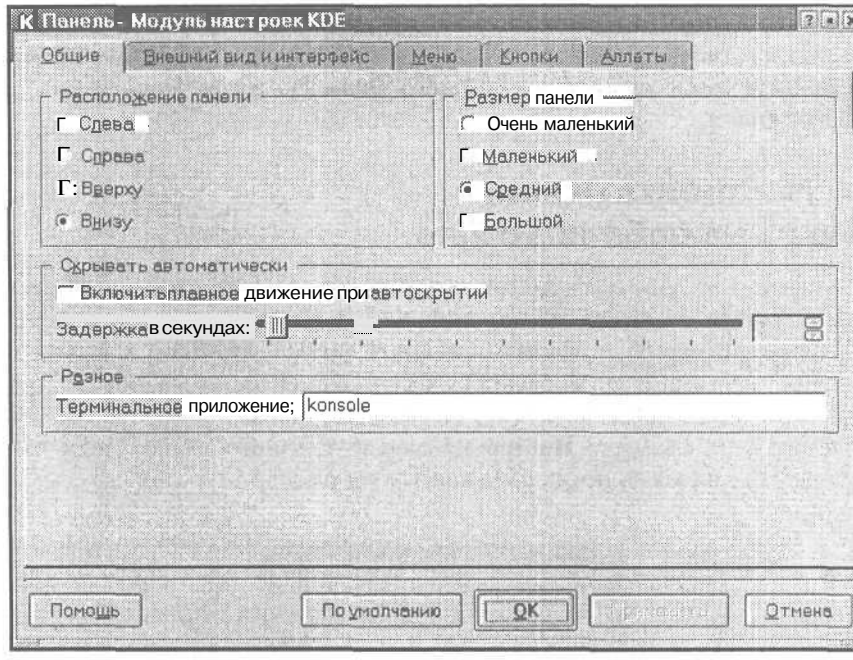


Рис. 15.9. Окно настроек панели

Обратите внимание на серый и как бы ребристый прямоугольник в левом конце этой дополнительной панели. Щелкнув по нему правой кнопкой мыши, вы получите возможность удалить дополнительную панель или добавить на нее что-то (только кнопку мыши надо удерживать). С помощью таких же серых и ребристых прямоугольников можно получить доступ к меню других элементов главной панели.

Рис. 15.10. Дополнительная панель задач

Меню управления кнопками на панели появляются после щелчка правой кнопкой мыши по самой кнопке (рис. 15.11).

Первый элемент в этом меню **(Панель меню)** вызывает уже известное нам меню настроек панели, следующие два служат для перемещения или удаления данной конкретной кнопки, а последний вызывает окно, изображенное на рис. 15.12. В этом окне вы можете изменить некоторые параметры той кнопки, щелчок по которой вызвал появление меню. Изменить можно ри-

сунок на кнопке, вызываемую по ней программу, права, с которыми программа запускается, и список типов файлов, ассоциированных с данным приложением.

Рис. 15.11. Выпадающее меню для кнопки на панели

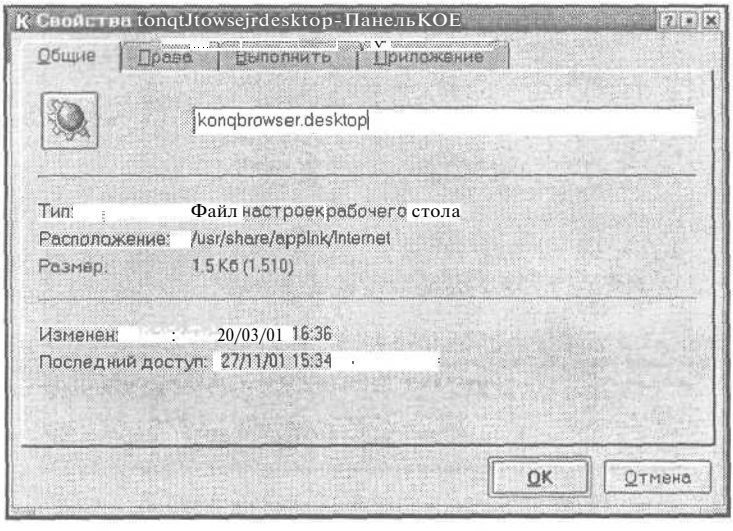


Рис. 15.12. Свойства ссылки

Подробнее со всеми свойствами и возможностями панели вам придется разбираться самостоятельно, а я закончу тем, что расскажу вам "страшную" историю. Во время знакомства с ее возможностями и опциями настройки я умудрился однажды вообще удалить панель с экрана. Как уж это получилось, я тогда даже не понял. Но факт тот, что пропали и сама панель, и главное меню, и спасительная буква "К", а, следовательно, возможность вообще что-либо запустить. Мне долго пришлось искать выход из этой ситуации, но он, конечно же, нашелся. Щелкнув по пустому полю на рабочем столе, я получил выпадающее меню, в котором нашлась команда **Помощь по рабочему столу.** Изучив эту подсказку, я узнал, что панель представляет собой отдельную программу, которая называется kicker. Запустив эту програм-

му с помощью того же выпадающего меню рабочего стола **(Выполнить команду)**, я успешно вернул панель на ее законное место.

В этом разделе нам осталось только сказать пару слов о значках ("иконках") на рабочем столе. Они тоже могут использоваться для запуска приложений или быстрого доступа к некоторым файлам или каталогам. Впрочем, вы вероятно, знакомы с ними по работе в Windows. Мне, как и вам, наверное, неоднократно приходилось видеть экраны мониторов, усыпанные такими значками. Правда, сам я не сторонник такой россыпи картинок на рабочем столе. На мой взгляд, гораздо удобнее пользоваться хорошо структурированным главным меню. Но о вкусах не спорят. Так что, если вам это нравится, щелкайте по пустому пространству рабочего стола, и создавайте новую ссылку на приложение, файл или устройство, как это показано на рис. 15.13.

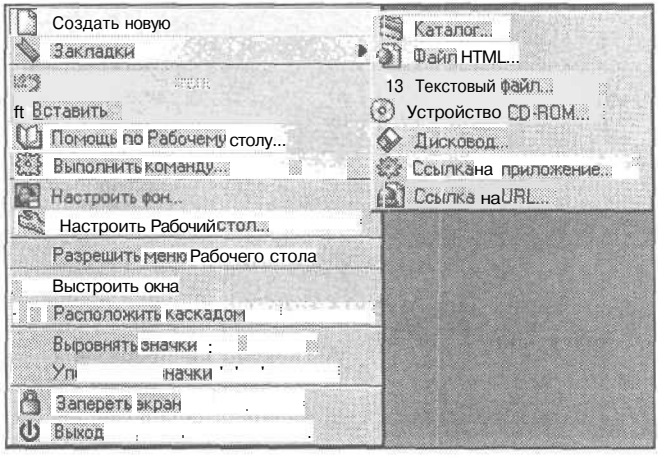


Рис. 15.13. Создание новой ссылки на рабочем столе

Если же хотите знать мое мнение, то из значков на рабочем столе необходимы только значки, ссылающиеся на устройства, и, в первую очередь, на Floppy-дисковод и дисковод CD-ROM. С помощью таких значков очень удобно монтировать и размонтировать сменные носители: достаточно вызвать выпадающее меню (щелчком правой кнопки мыши по соответствующему значку) и выбрать нужную команду.

На этом я закончу свое очень краткое введение в KDE. Точнее, не введение в KDE, а краткое описание программы kicker. Потому что кроме этой программы KDE включает в себя еще массу разнообразных приложений, общим числом более сотни. Описать их все не представляется возможным, поэтому я вернусь к решению той задачи, которую сформулировал в начале этой главы: подобрать набор Linux-приложений, которые необходимы про-

стому пользователю для комфортной работы. Но начать надо с конкретизации самого понятия "удобная рабочая среда".

15.2. Что такое "удобная рабочая среда"

Компьютер и его операционная система, будь то Linux или Windows, — это не самоцель. И то и другое нужно нам только как инструмент, служащий для выполнения каких-то работ, решения определенного круга задач. Задачи эти в большинстве случаев решаются не средствами операционной системы, а путем запуска каких-то приложений. Их совокупность и создает на компьютере то, что мы далее будем называть "удобной рабочей средой". Очевидно, что состав компонентов, которые создают удобную для пользователя рабочую среду, в значительной мере определяется теми задачами, которые этот пользователь решает с помощью компьютера. Однако имеется и некоторый базовый набор приложений, который необходим каждому пользователю.

Если вы работали на компьютере до перехода на ОС Linux, у вас уже имеются какие-то привычки и собственное представление о том, что вам необходимо для комфортной работы. Естественно, что представление о том, что такое удобно, у каждого свое. Я не хочу навязывать вам свои вкусы или предпочтения, но все же расскажу, как я представляю себе удобную рабочую среду, и что, по моему мнению, нужно сделать под Linux, чтобы ее создать. Вы вольны последовать моим советам или сразу отказаться от чтения этого раздела.

Хочу особо подчеркнуть, что речь пойдет именно о персональном компьютере пользователя. Под персональным компьютером понимается как домашний компьютер, так и компьютер, установленный в офисе, на рабочем месте. Суть все равно в слове "персональный", отличающем этот компьютер от разного рода серверов. Итак, перечислю кратко те компоненты, которые, по моему мнению, составляют такую рабочую среду.

- Создание удобной рабочей среды начинается с подбора необходимых утилит для обслуживания аппаратной части, организации печати, работы с файловой системой, организации архивного хранения информации и т. д. К той же категории необходимых служебных утилит я бы отнес средства защиты от вирусов и других вредных воздействий, а также средства "защиты от дурака".
- Следующим по важности компонентом такой среды, безусловно, является набор офисных приложений, начиная с текстового редактора (или процессора). Многим пользователям необходимы также электронные таблицы и какая-либо система управления базами данных (хотя это уже касается далеко не всех).
- Каждый пользователь так или иначе сталкивается и с обработкой графической информации. Большинство из нас не художники, так что рисо-

вать картины нам не потребуется, но просматривать графические файлы разных форматов необходимо всем.

- Следующая задача — организация работы пользователя: ведение деловых дневников, напоминание о задачах, работах, встречах и т. п., т. е. выполнение функций персонального органайзера.
- Затем идет организация взаимодействия с коллегами, что обычно реализуется средствами электронной почты. Можно возложить на компьютер и организацию факсимильной и телефонной связи.
- Если ваш компьютер подключен к сети Интернет, то необходим браузер, FTP-клиент и программные средства для получения новостной информации.
- П Если вы в той или иной мере программируете, то необходимы и какие-то средства разработки, отладки и компиляции программ. Но рассказывать об этой категории продуктов и рекомендовать какой-то продукт я не берусь. Вообще-то эту категорию продуктов можно уже отнести к узкоспециальным. Точно так же математику нужна будет какая-то из математических или статистических программ, бухгалтеру — пакет типа "1С:Бухгалтерия" и т. д. Я упомянул этот класс продуктов в настоящем списке только для того, чтобы отметить необходимость наличия прикладных программ, связанных с профессиональной деятельностью пользователя.
- П Ну, и наконец, компьютер можно использовать не только для работы. Тем более, если речь идет о домашнем компьютере. Значит, в состав "удобной рабочей среды" надо включить средства мультимедиа: программы для прослушивания аудиозаписей, просмотра видео, а также хотя бы минимальный набор игр и развлечений.

На этом краткое описание того, что мы будем понимать под словами "удобная рабочая среда", закончено. Давайте перейдем к рассмотрению того, с помощью каких программных средств такую среду можно создать под Linux.

15.3. Утилиты

Начнем с разных вспомогательных программ. Этот этап создания удобной рабочей среды можно разделить на две стадии. На первой настраивается сама операционная система, причем часть действий по ее настройке должен выполнить администратор. Неважно, что на персональном компьютере это тот же человек, который потом будет выступать в роли пользователя. Важно то, что для выполнения соответствующих действий нужно иметь права суперпользователя. Администратор должен, во-первых, произвести настройку каких-то компонентов самой ОС и, во-вторых, установить и настроить до-

полнительное программное обеспечение, которое облегчит выполнение задач, связанных с обслуживанием и настройкой ОС, а также то программное обеспечение, которое потребуется пользователям для работы.

Вторую часть действий по настройке можно выполнять уже с правами обычного пользователя. Эта часть включает как настройку системных параметров (например, выбор шрифтов или настройку графического интерфейса), так и установку и настройку конкретных приложений. Хочу заметить, большинство из программ, установку которых я возложил на администратора, может установить и обычный пользователь. Просто в многопользовательских системах принято, чтобы такое ПО устанавливал администратор. В общем, поскольку речь идет о персональном компьютере, это всё условности, так что решайте сами. Можете следовать моим советам относительно того, какое ПО должен установить администратор, а какое — пользователь, а можете распорядиться по-своему.

Итак, процедура установки Linux закончена и вам удалось войти в систему с правами суперпользователя, т. е. под именем root. Я не буду говорить о русификации системы (хотя без этого трудно говорить об "удобной рабочей среде"), потому что считаю, что дистрибутив изначально русифицирован, либо с этим вы уже справились. Для начала давайте, еще даже до выхода в графический режим, настроим некоторые компоненты консоли.

Первым делом я рекомендую установить файловый менеджер Midnight Commander *(см. гл. 6),* если он не был установлен при инсталляции дистрибутива. Лично я привык его использовать даже в графическом режиме. После его установки я обычно задаю сокращенный режим отображения в обеих панелях и полный 8-битный ввод и вывод, после чего сохраняю настройки (через команду меню **Настройка | Сохранить настройки).**

Далее стоит создать точки монтирования для гибких дисков, CD-ROM, для разделов, отформатированных в MS Windows, и т. д., а также отредактировать соответствующим образом файл /etc/fstab (об этом было достаточно подробно рассказано в *гл. 4).*

После этого можно дополнить меню пользователя программы Midnight Commander (оно находится в файле /usr/lib/mc/mc.menu) командами монтирования и размонтирования дискеты и CD-ROM. Это сильно облегчит вам операции монтирования/размонтирования в последующем (даже если на рабочем столе KDE имеются соответствующие значки, облегчающие выполнение этих операций, ведь монтировать диски приходится и в консоли). Можно также включить в меню другие часто используемые команды.

Теперь запускаем графический режим и переходим к его настройке. Выберите графическую среду (я рекомендую KDE) и настройте менеджер рабочего стола на автоматический запуск этой среды. Это можно сделать, создав файл desktop в каталоге /etc/sysconfig/, в котором указать, какая графическая

среда запускается по умолчанию. В этом случае указанная графическая среда будет по умолчанию запускаться для всех пользователей.

После первого запуска графическая среда KDE тоже требует некоторых настроек. Я, например, изменяю размеры шрифтов (поскольку стал уже плохо различать мелкие буквы) с помощью Центра управления KDE, а также в консоли и редакторе KWrite, переношу на панель иконки тех программ, которыми часто пользуюсь (консоль и Netscape Communicator) и вообще настраиваю панель под свои привычки, делаю фон значков на рабочем столе прозрачным, и т. д.

Поскольку и в графическом режиме часто приходится выполнять консольные приложения (хотя бы тот же Midnight Commander), проведите настройку эмулятора терминала, пользуясь командой меню **Настройки** окна эмулятора консоли. Я обычно увеличиваю размер окна и размер шрифта. Не забудьте сохранить сделанные настройки как из меню Midnight Commander, так и из меню окна консоли (которое появляется после щелчка правой кнопкой мыши по заголовку окна).

В качестве файлового менеджера в графической оболочке я чаще всего использую тот же Midnight Commander. Просто запускаю его в окне эмулятора терминала и с его помощью как путешествую по файловой структуре, так и провожу все операции с файлами, от создания (запуская команду touch в командной строке) до редактирования и удаления. Однако не стоит забывать и о файловом менеджере Konqueror. Кому-то он может нравиться больше, чем Midnight Commander. Я уже упоминал Konqueror в качестве интернет-браузера в *гл. 14.* Но и в качестве файлового менеджера он, безусловно, заслуживает внимания. Во-первых, хочется сказать, что он многолик. Запустите его и посмотрите на его вид, выбирая различные команды в меню **Окно.** С помощью команды **Загрузить профиль панели,** например, можно сделать окно подобным окну эмулятора терминала с запущенным в нем Midnight Commander (рис. 15.14).

Как видите, есть даже командная строка. Правда, мне лично не хватает привычных кнопок, ассоциированных с клавишами <F1>—<F10> и некоторых других элементов интерфейса, так что я в большинстве случаев предпочитаю пользоваться "настоящим" Midnight Commander. Но в чем Konqueror превосходит Midnight Commander, так это в средствах просмотра файлов. И для этих целей я использую именно Konqueror. В нем в большинство файлов можно "заглянуть" так же удобно, как и в любой каталог. Это касается не только текстовых или HTML-файлов, но и, например, графических файлов разных форматов. В общем, Konqueror, безусловно, является одним из элементов удобной пользовательской среды, а уж в каких случаях им пользоваться — дело вкуса и привычек (или наличия необходимых знаний и соответствующей степени знакомства с продуктом).

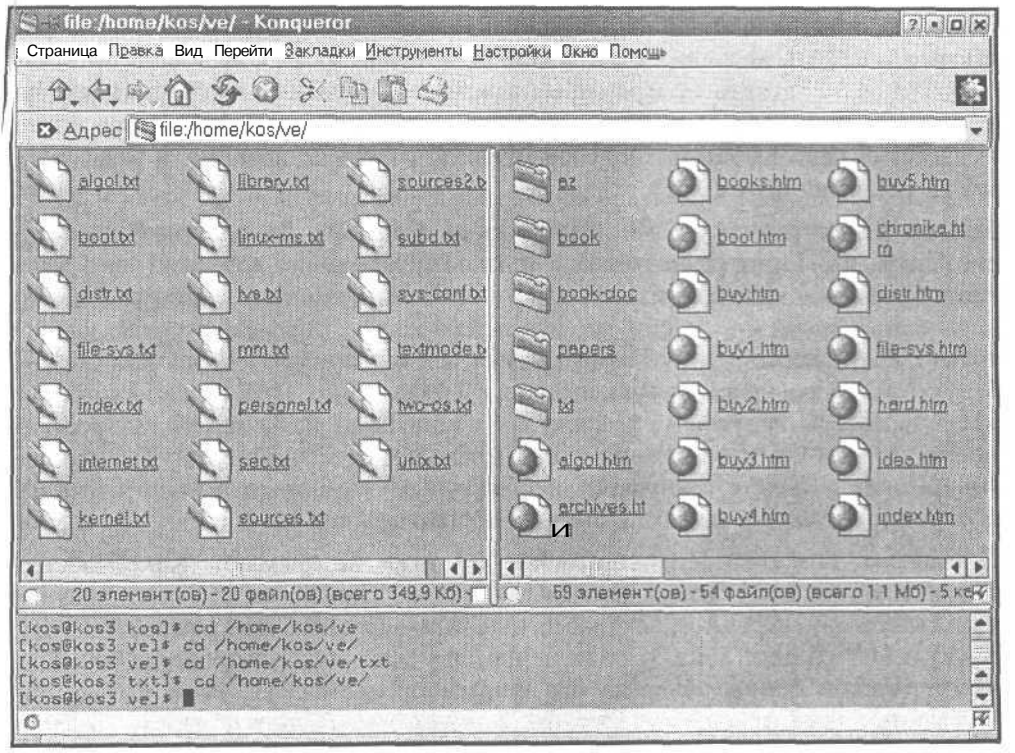


Рис. 15.14. Окно файлового менеджера Konqueror с двумя панелями и командной строкой

Из дополнительно устанавливаемых утилит в первую очередь надо упомянуть архиватор bzip2 (если он не установлен). Его необходимо установить, поскольку архивы в этом формате нередко будут встречаться вам в Интернете.

В составе KDE имеется довольно удобная утилита Архиватор (ark), которая умеет работать с разными типами архивных файлов. Она позволяет просматривать архивы разных форматов (tar, tar.gz, tar.bz2, zip, rar, zoo, lzh, a), а также создавать новые архивы.

О некоторых программах, которые в главном меню KDE первоначально отнесены к группе утилит, я еще напомню вам в следующих разделах. А здесь, в завершение раздела об утилитах, хочу сказать пару слов о двух средствах, необходимых на мой взгляд для любого компьютера при любой операционной системе. Это источник бесперебойного питания с соответствующим программным обеспечением и антивирусный пакет.

В разделе о файловой системе уже было сказано, что выключать компьютер с Linux простым отключением питания недопустимо: это может привести к непоправимым сбоям в файловой системе. Но ведь отключение питания

может произойти и не по вашей воле. Поэтому очень желательно приобрести и установить источник бесперебойного питания, а также найти, инсталлировать и настроить программные средства для выключения системы по сигналу о выключении питания. К сожалению, я не могу дать здесь более подробные рекомендации, но упомянуть об их существовании и необходимости посчитал полезным.

Так же кратко упомяну и об антивирусных средствах. До недавнего времени считалось, что Linux не подвержен опасности заражения вирусами. Это даже преподносилось как одно из достоинств этой операционной системы. Однако в последнее время стало ясно, что это далеко не так. Впрочем, это и не удивительно, если вспомнить, что самый известный из вирусов — червь Морриса - был создан и распространялся именно в UNIX-сетях. Просто до некоторого времени авторы вирусов не уделяли внимания Linux из-за малой распространенности этой ОС и отсутствия у них необходимых знаний о ней. Так что антивирусные утилиты становятся необходимым компонентом программного обеспечения также и на компьютерах, работающих под Linux.

К счастью, создатели антивирусного ПО тоже не дремлют. Как "Лаборатория Касперского" **(www.kaspersky.ru),** так и "Диалог-наука" **(www.dials.ru)** уже выпустили версии своих известных антивирусных программ AVP (или KAV) и DrWeb для Linux. К сожалению, эти продукты распространяются на коммерческой основе. Однако обе фирмы предоставляют всем желающим "урезанные" версии своих программ, которые позволяют обнаружить заражение, но не обладают способностью лечить зараженные файлы.

15.4. Офисные приложения

Вряд ли вы найдете такой персональный компьютер, на котором отсутствовал бы текстовый редактор или процессор. Средства этого класса были подробно рассмотрены в *гл. 12,* так что здесь повторю только основной вывод. Из всего разнообразия текстовых редакторов необходимо выбрать редактор для текстового режима (я рекомендую встроенный редактор CoolEdit программы Midnight Commander), редактор файлов формата ACSII для графического режима (наилучший вариант — Nedit, хотя можно использовать тот же CoolEdit), и, конечно, мощный текстовый процессор. В последнем случае выбор пока неоднозначен. Наиболее вероятным кандидатом на сегодняшний день является текстовый процессор из пакета OpenOffice.org, но есть шанс, что его догонят AbiWord и KWord. Отдельно нужно упомянуть Lyx (Klyx). Этот текстовый редактор ближе к издательским системам, поскольку он является как бы оболочкой к TEX. Но этот продукт очень удобен для научных работников, поскольку позволяет вводить разнообразные формулы. В общем, я пока остановился на связке CoolEdit — Nedit — swriter из пакета OpenOffice.org.

Поскольку редакторы Nedit и пакет OpenOffice.org могут не входить в состав дистрибутива, их надо установить дополнительно. Кроме того, стоит установить программу Acrobat Reader для просмотра файлов формата PDF, программу-перекодировщик кодовых страниц и настроить программу проверки правописания ispell (она или aspell обычно устанавливается с русифицированными дистрибутивами).

Когда говорят об "офисных приложениях", обычно имеют в виду не только работу с текстом, но и электронные таблицы, программу для построения схем, системы подготовки презентаций, а также и какую-либо из систем управления базами данных (хотя последнее уже необходимо далеко не всем). Здесь два основных конкурента в борьбе за место на вашем рабочем столе — пакеты OpenOffice.org и KOffice, хотя нельзя забывать и о программах проекта GNOME.

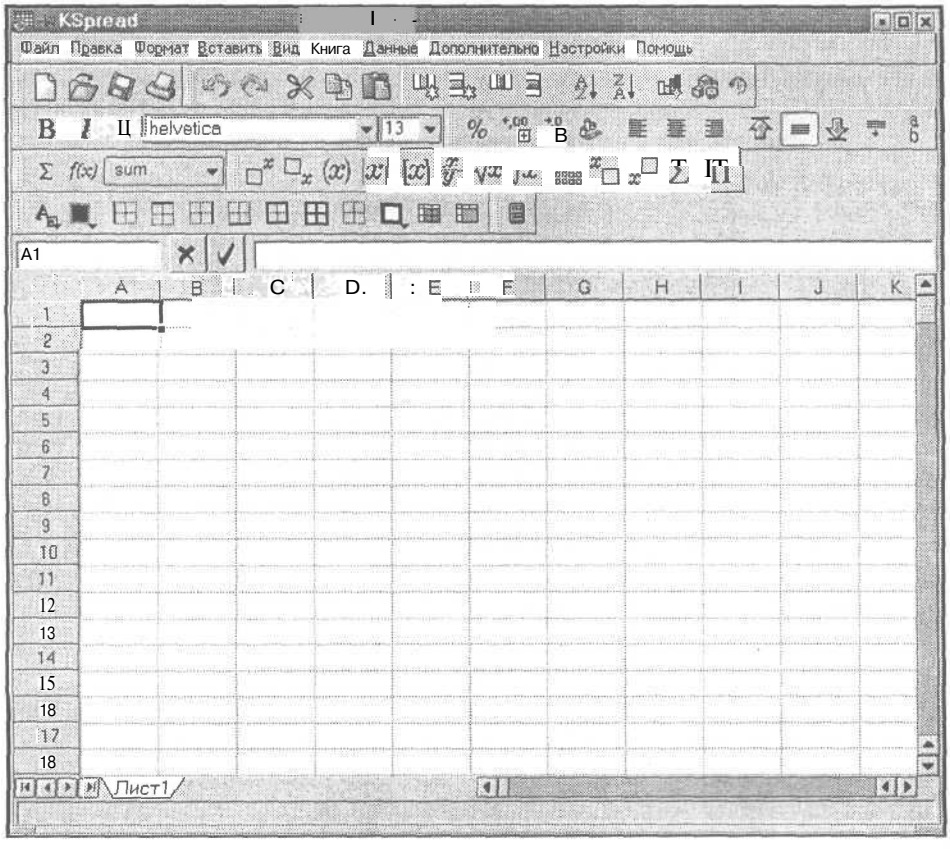


Рис. 15.15. Электронная таблица KSpread

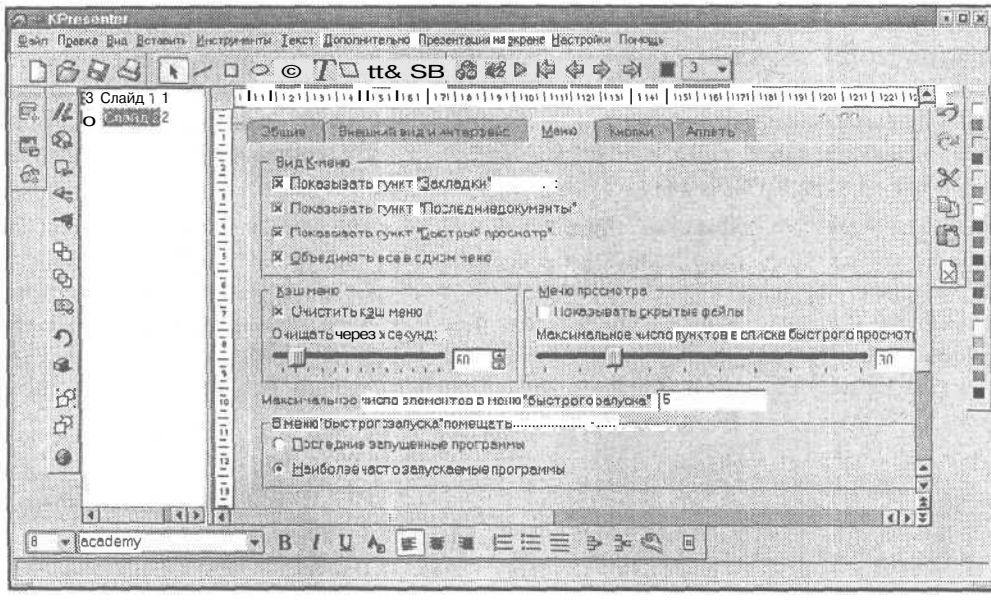


Рис. 15.16. Система подготовки презентаций KPresenter

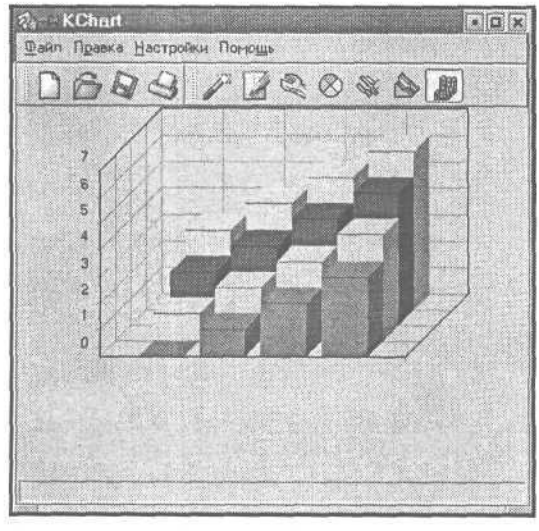


Рис. 15.17. Программа для создания диаграмм KChart

В состав пакета KOffice входит основной набор офисных приложений: электронная таблица KSpread (рис. 15.15), система подготовки презентаций KPresenter (рис. 15.16), программа для создания диаграмм KChart (рис. 15.17) и простенькая "рисовалка" KIllustrator. Все эти компоненты

можно объединить в один рабочий стол KOffice (рис. 15.18), что позволяет легко переключаться между этими приложениями.

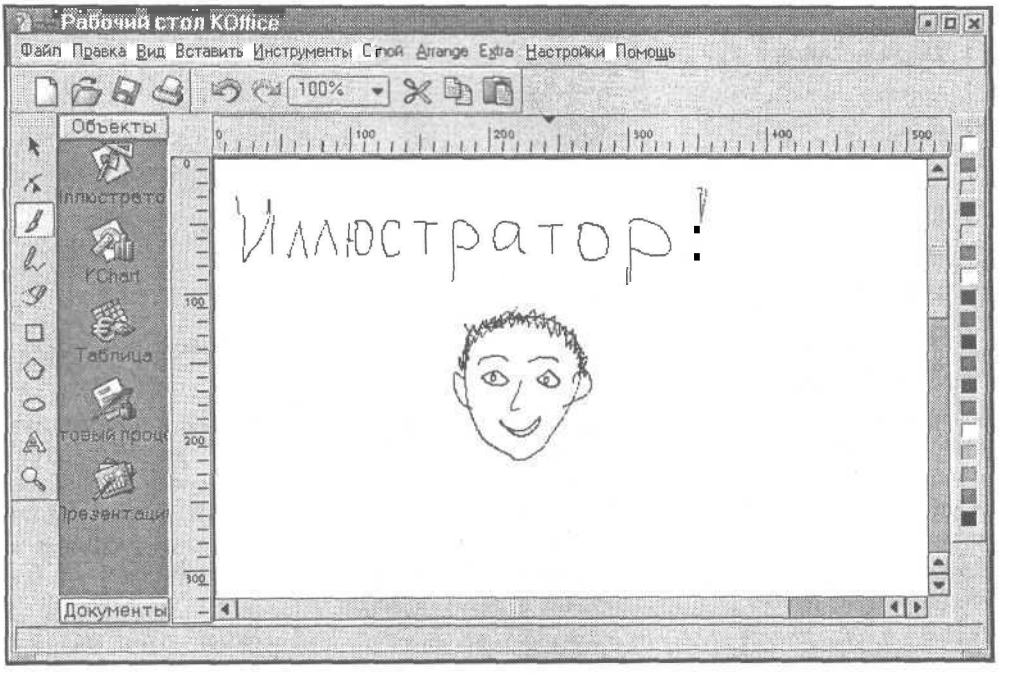


Рис. 15.18. Рабочий стол Koffice с запущенной программой KIllustrator

Я не могу утверждать, что эти приложения обеспечивают все возможности, которые имеются в пакете Microsoft Office, но пакет KOffice интенсивно развивается и совершенствуется, так что если не сейчас, то в ближайшем будущем будет удовлетворять потребности рядового пользователя. Его обновленные версии можно найти на сайте **http://koffice.kde.org/**.

Что касается электронных таблиц, то, по отзывам в Интернете, очень неплохая электронная таблица для Linux разработана в рамках проекта GNOME. Называется она Gnumeric и найти ее можно по адресу **ftp://ftp.gnome.ru/gnumeric/.**

Возможности этой программы (по крайней мере по той информации, которая опубликована на сайте **http://www.gnome.ru)** впечатляют. Она поддерживает 95% встроенных функций Excel и 100% инженерных функций, внутритабличные зависимости, все встроенные в Excel форматы представления данных (числовой, денежный, учетный, даты, времени, процентов, дробный,

научный, текстовый, специальный) и многое другое. В составе Gnumeric имеются фильтры для импорта файлов в форматах:

- MS Excel (вся суммарная информация, весь текст таблиц + формулы, стили, выделения, встроенные изображения, настройки принтера);
- Формат Lotus 1-2-3 .wk1;
- Applix;
- Psion;
- Sylk;
- XBase;
- Oleo;
- XML (собственный формат);
- HTML (различный);
- CSV (значения, разделенные запятыми).

а также фильтр для экспорта таблиц в формат Excel (итоговая информация, весь текст таблиц + формулы, названия). Реализована 361 функция электронных таблиц и 17 аналитических инструментов. Поддержка MS Excel выполнена на высоком уровне, большая часть функций работает (не реализованы пока некоторые финансовые и математические функции). Gnumeric локализован для различных языков и обрабатывает числовые форматы в соответствии с правилами различных стран и языков.

Что касается СУБД, то скажу только, что для Linux разработано или адаптировано более десятка разных СУБД, от Gadfly до Oracle. Наиболее часто в литературе упоминаются PostgreSQL и MySQL. По этим двум СУБД уже выпущено несколько книг (на русском языке), так что если вас эта тема интересует, вы без труда найдете необходимую информацию.

15.5. Графический редактор GIMP

Если СУБД встречается далеко не на каждом персональном компьютере, то программы работы с графикой нужны каждому пользователю, если не для создания или редактирования картинок, так хотя бы для их просмотра. Хотя для Linux и созданы разнообразные специализированные средства просмотра изображений и управления коллекциями картинок, но для простого пользователя вроде меня может оказаться вполне достаточным применение стандартного для KDE браузера Konqueror. Как я уже говорил, этот браузер позволяет просматривать файлы самых разных форматов, в том числе и графических. Впрочем, в составе KDE имеется и отдельная программа для просмотра изображений KView, а в состав большинства дистрибутивов входит программа Image Magic, которая позволяет не только просматривать графические файлы, но и производить множество операций по их трансформации (изменения яркости, размера, вращения, растяжения и т. п.).

Но основной программой для работы с графикой под Linux является GIMP.

Само название GIMP есть акроним от GNU Image Manipulation Program, что переводится как программа манипуляции образами проекта GNU. Она предназначена для решения таких задач, как ретуширование фотографий, наложение графических образов и их создание. По своим возможностям она не уступает таким программам, как Adobe Photoshop или Corel PhotoPaint. Однако она имеет перед последними то огромное преимущество, что свободно доступна через Интернет и в составе различных дистрибутивов Linux. GIMP славен также тем, что для него легко писать дополнения (плагины) на разных языках программирования, а поэтому существует уже обширный набор таких дополнений.

GIMP в основном работает с изображениями в растровом формате (bitmap). То есть с изображениями, составленными из отдельных пикселов (pixels) — крошечных точек, каждая из которых окрашена каким-то своим цветом. Множество таких точек образуют целостную картину.

Существует и другой формат сохранения изображений в файле — векторный. При этом способе рисунок создается путем задания кривых, координат и заполнения некоторых областей определенным цветом. GIMP обеспечивает некоторую поддержку векторной графики с помощью плагина Gfig, но не позволяет производить полноценное редактирование изображений в векторном формате и не может быть использован для создания сложных векторных диаграмм. Кроме того, GIMP обладает возможностями создания анимации в форматах AVI и GIF, а также может показывать видео в формате MPEG.

Ниже приводится краткий список возможностей и особенностей GIMP:

- полный набор инструментов для рисования, включая кисти, карандаш, пульверизатор и т. п.;
- сохранение изображения в памяти в виде отдельных частей, так что размер изображения ограничен только доступным дисковым пространством;
- поддержка альфа-каналов (прозрачности);
- работа со слоями;
- развитые возможности поддержки скриптов, причем существует даже возможность вызова встроенных функций GIMP из внешних скриптов, таких как Script-Fu, Perl-Fu (скрипты на Perl) и Python-Fu (скрипты на Python);
- многоуровневые операции отмены действия (undo) и повторного выполнения (redo), ограниченные только дисковым пространством;
- наличие инструментов трансформации рисунка, включая вращение, масштабирование, искривление и зеркальное отображение;
- список поддерживаемых форматов включает PostScript, JPEG, GIF, PNG, XPM, TIFF, TGA, MPEG, PCX, BMP и многие другие;

- наличие инструментов выделения прямоугольных, эллиптических областей, областей произвольной формы, связанных областей и выделение форм в изображении (intelligent scissors);
- наличие возможности вызова плагинов, что позволяет легко наращивать функциональность программы, добавлять возможность работы с новыми форматами файлов и новыми фильтрами.

Когда вы впервые запускаете GIMP и впервые сталкиваетесь с его интерфейсом, он может показаться вам весьма необычным, поскольку состоит из нескольких отдельных окон. В управлении программой широко используются выпадающие меню, появляющиеся после щелчка правой кнопкой мыши по отдельным элементам окон. Основных окон два — окно инструментов или Toolbox (рис. 15.19) и окно изображения — Image Window (рис. 15.20).

Рис. 15.19. Главное окно программы GIMP

Кроме этих окон можно открыть (из меню, вызываемого через команду **Файл | Диалоги)** различные вспомогательные окна, служащие для выбора инструментов, шаблонов, задания цвета (палитра), слоев (layers) и т. д.

Окно инструментов (см. рис. 15.19) содержит:

- главное меню, состоящее всего из трех элементов — **Файл,** Расш(ирения) и **Справка;**
- кнопки инструментов (два ряда квадратиков с изображениями);
- панель выбора цветов фона и "краски" (внизу слева);
- индикатор статуса инструментов (внизу справа).

Одиночный щелчок левой кнопкой мыши по кнопке с указанием инструмента означает выбор соответствующего инструмента, а двойной щелчок вызывает появление окна настройки. Задержите указатель мыши над любым элементом, и вы увидите краткое пояснение назначения этого элемента. Одиночный щелчок по квадратикам на панели выбора цвета и по любому из трех элементов на панели статуса вызовет появление соответствующего окна, в котором можно произвести нужный выбор. Такие же окна можно вызвать и через меню **Файл | Диалоги.**

Второе окно, которое вы всегда будете использовать при работе с рисунком, — это окно изображения (Image Window).

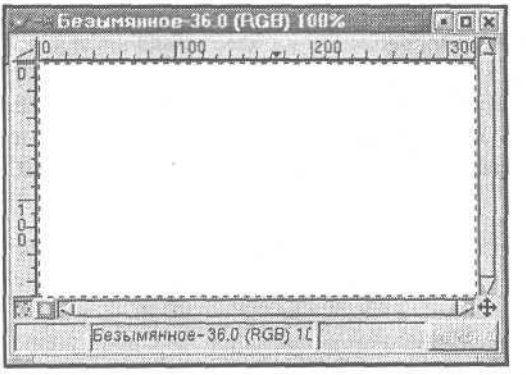


Рис. 15.20. Окно изображения (Image Window) в GIMP

На рис. 15.20 показано, как выглядит это окно при его открытии (через команду меню **Файл | Новый),** когда в нем еще ничего не рисовали и не открывали файл с готовым рисунком. Белое поле в этом окне — это то место, где вы можете проявить свои творческие способности. Если отображаемый в окне рисунок не помещается в окне, можно использовать линейки прокрутки.

Основной способ взаимодействия с окном изображения заключается в вызове меню. Это достигается щелчком левой кнопки мыши по треугольнику в левом верхнем углу окна или щелчком правой кнопки мыши в любом месте рабочей зоны (нажатие левой кнопки, естественно, служит для применения выбранного инструмента). Меню это многоуровневое. Если вас утомляет необходимость постоянно щелкать мышкой для вызова меню, вы можете щелкнуть по пунктирной линии, идущей вдоль верхней границы выпадающего меню, и тогда это меню будет постоянно (пока вы его сами не закроете) находиться на экране в виде отдельного окна.

Я надеюсь, что первоначальное представление о том, как запустить GIMP и управлять им вы получили. Полное описание приемов работы с этой программой не входит в мою задачу. Если вам это интересно, загляните на сайт А. Селезнева "GIMP по-русски" по адресу **http://gimp.linux.ru.net/**.

Кроме того, рекомендую вам просмотреть обзор другого программного обеспечения для работы с графикой под Linux, который вы найдете в статье В. Галактионова "Существует ли графика для Linux", опубликованной в № 4 журнала "Мир ПК" за 2001 год **(http://osp.ru/pcworld/2001/04/078.htm).**

15.6. Персональный органайзер

Если вы используете свой компьютер не только для игр, но и в качестве рабочего инструмента, вам будет очень полезен какой-то продукт класса персональных организаторов (органайзеров). Такая программа вовремя напомнит вам *о* всех заранее запланированных событиях, встречах, совещаниях и т. п. В составе графической среды KDE имеется программа этого класса – Календарь KDE. Я не буду сравнивать ее возможности с возможностями других подобных программ, я просто приведу описание ее основных возможностей.

После запуска программы на панели рядом с часами появится небольшой значок программы (он изображен на рис. 15.21).

Рис. 15.21. Значок календаря на панели

Один щелчок мышкой по этому значку — и разворачивается окно программы. Вид его зависит от того, в каком состоянии программа завершила работу в предыдущий раз. Основных видов четыре, один из них изображен на рис. 15.22 (календарь на день). Можно отобразить календарь на день, на неделю (причем показывать можно только рабочую неделю, а можно — вместе с выходными), на месяц. Имеется также вариант, при котором перечень событий отображается в виде списка. Можно переключить календарь на отображение списка задач (в других вариантах этот список отображается в левой нижней панели Календаря).

Задать новое событие можно через команду **Действие** главного меню Календаря. Открывается окно с тремя вкладками, изображенное на рис. 15.23.

На первой вкладке вы вводите название события, время его начала и (при желании) время завершения. Здесь же можно ввести какой-то комментарий к событию и указать, за какое время до его наступления программа должна напомнить вам о нем, чтобы, например, не опоздать на назначенную встречу. По вашему желанию можно отнести событие к одной из определенных категорий. Если вы поставите отметку напротив надписи **Повторяющееся событие,** то стоит перейти на вкладку **Повторение,** чтобы задать параметры повторения (рис. 15.24). На вкладке **Приглашенные** можно задать список участников события.

Когда наступит установленное для напоминания о событии время, на экране появится небольшое окно с соответствующей информацией (рис. 15.25).

Чтобы ввести новую задачу, нужно тоже воспользоваться главным меню программы или дважды щелкнуть по панели **Задачи** (см. рис. 15.22).

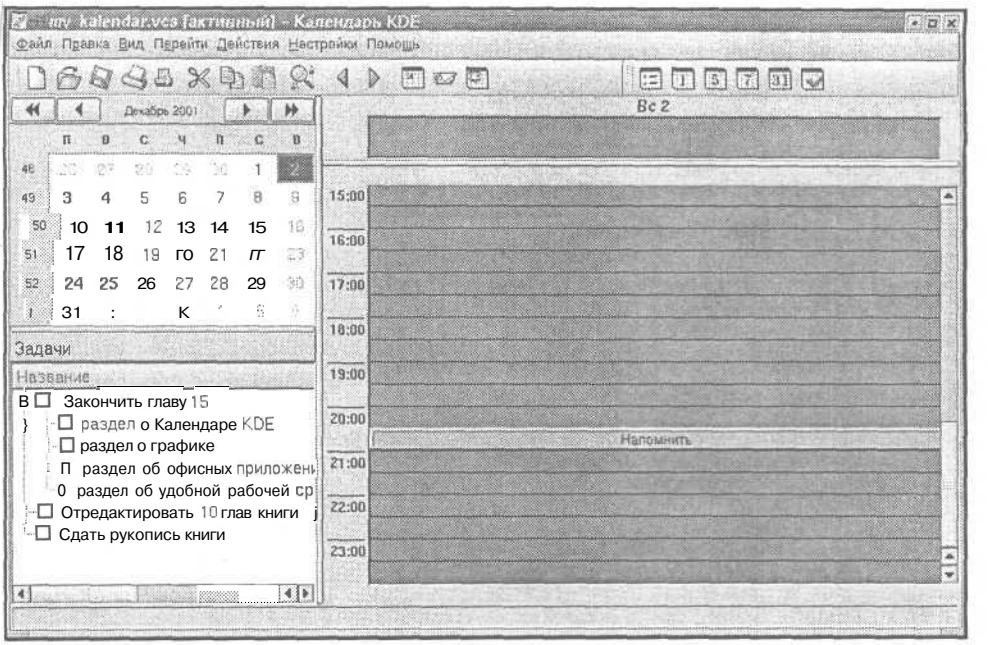


Рис. 15.22. Календарь на день

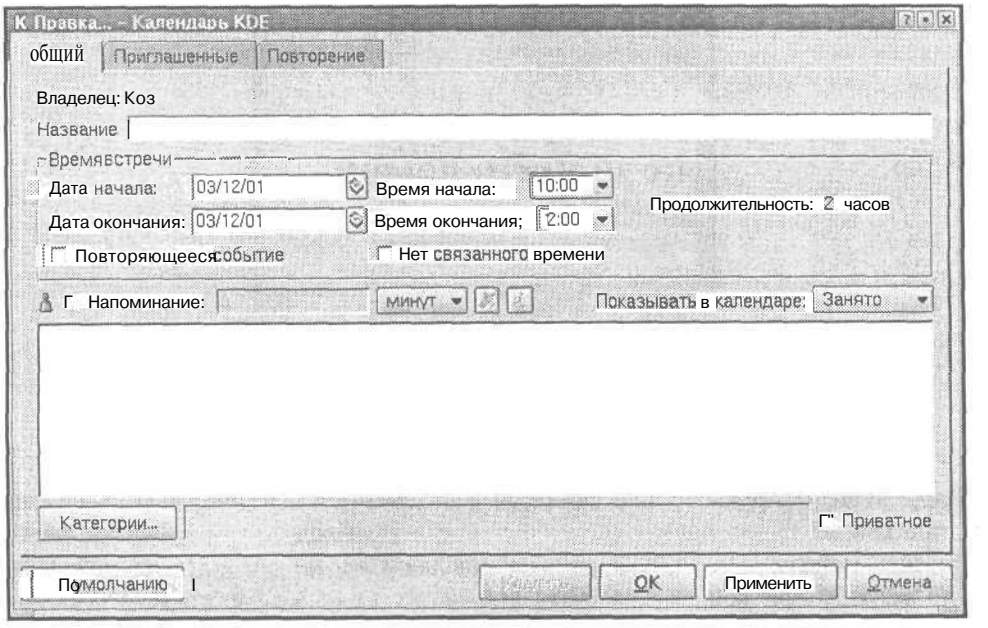


Рис. 15.23. Календарь. Задание нового события

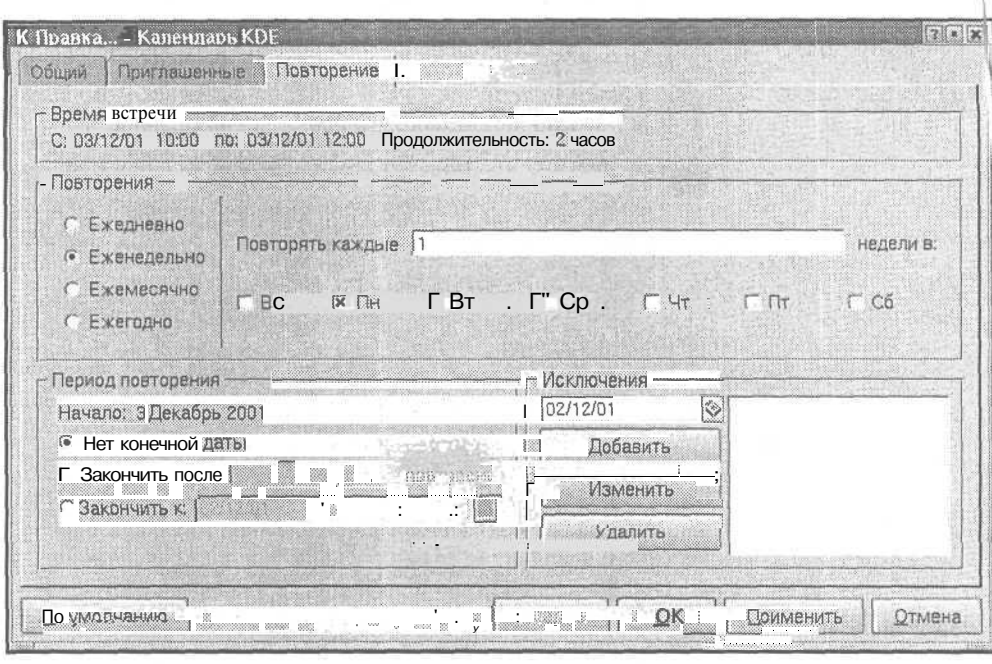


Рис. 15.24. Календарь. Определяем периодичность события

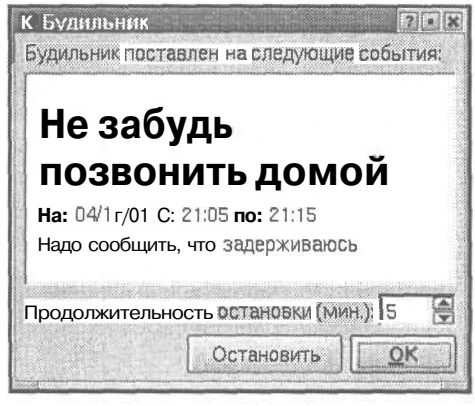


Рис. 15.25. Сообщение программы Календарь

После этого остается только заняться выполнением запланированных задач, а программа будет напоминать вам, когда необходимо прерваться, чтобы принять участие в запланированных совещаниях, не опоздать на назначенную встречу или вовремя принять лекарство. Если же вы желаете просмотреть полный перечень запланированных событий и/или задач за определенный период, это можно сделать, экспортировав содержимое базы данных в

HTML-файл (команда меню **Файл | Экспорт).** Как это будет выглядеть, вы можете увидеть на рис. 15.26.

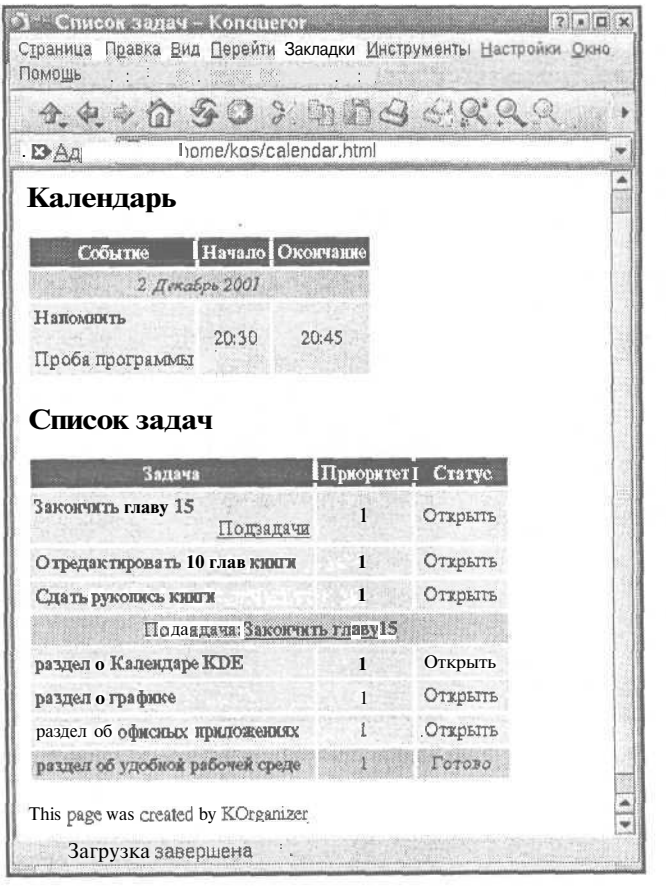


Рис. 15.26. Вывод календаря в виде HTML-файла

15.7. Общение с остальным миром

Итак, вы организовали собственную работу. Теперь надо организовать взаимодействие с коллегами. Самое распространенное средство для этого – программа электронной почты. В *гл. 14* была рассмотрена программа KMail - вполне пригодный для использования продукт. Альтернативой KMail может служить программа Netscape Messenger (рис. 15.27). Рассказывать о ней более подробно я здесь не могу, да в этом и нет необходимости,

поскольку интерфейс программы русифицирован и вам не составит труда разобраться с ним самостоятельно.

С помощью той же программы можно организовать просмотр новостей.

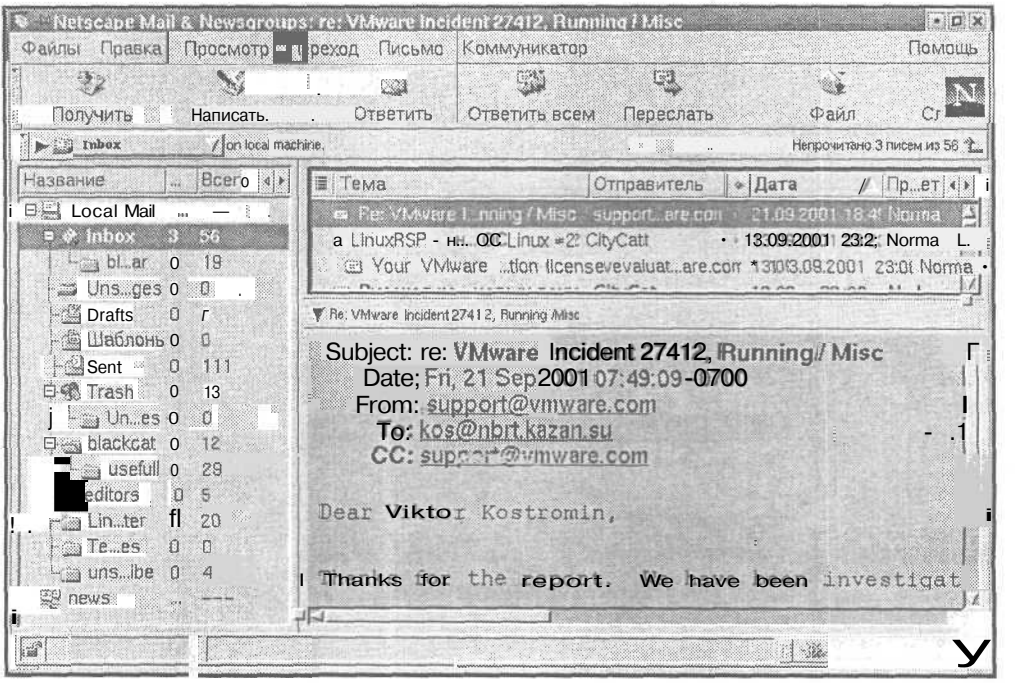


Рис. 15.27. Интерфейс программы Netscape Messenger

В качестве интернет-браузера я использую либо Netscape Navigator, либо Konqueror, а в качестве FTP-клиента — встроенные средства Midnight Commander. Для соединения с каким-либо сервером по FTP с помощью Midnight Commander надо вызвать меню любой из панелей и выбрать команду **FTP-соединение.** Появится строка ввода, изображенная на рис. 15.28. Если вы не помните, в каком формате вводится адрес удаленной машины, то нажмите клавишу <F1> и вы получите исчерпывающую подсказку. Можно сделать еще проще — набрать в командной строке

```
[user]# cd ftp://somehost.org/pub/dir/
```

или даже сразу указать имя и пароль пользователя:

```
[user]# cd ftp://USER:PASSWORD@someserver.ru/mydir
```

Можно занести часто используемые адреса FTP в списке любимых каталогов (нажмите <Ctrl>+<\>) и потом вызывать их при необходимости с помощью той же комбинации клавиш <Ctrl>+<\>. Работать по протоколу FTP

из программы Midnight Commander очень удобно — вы работаете с каталогами удаленной машины так же, как с локальными (если не считать задержек по времени, конечно).

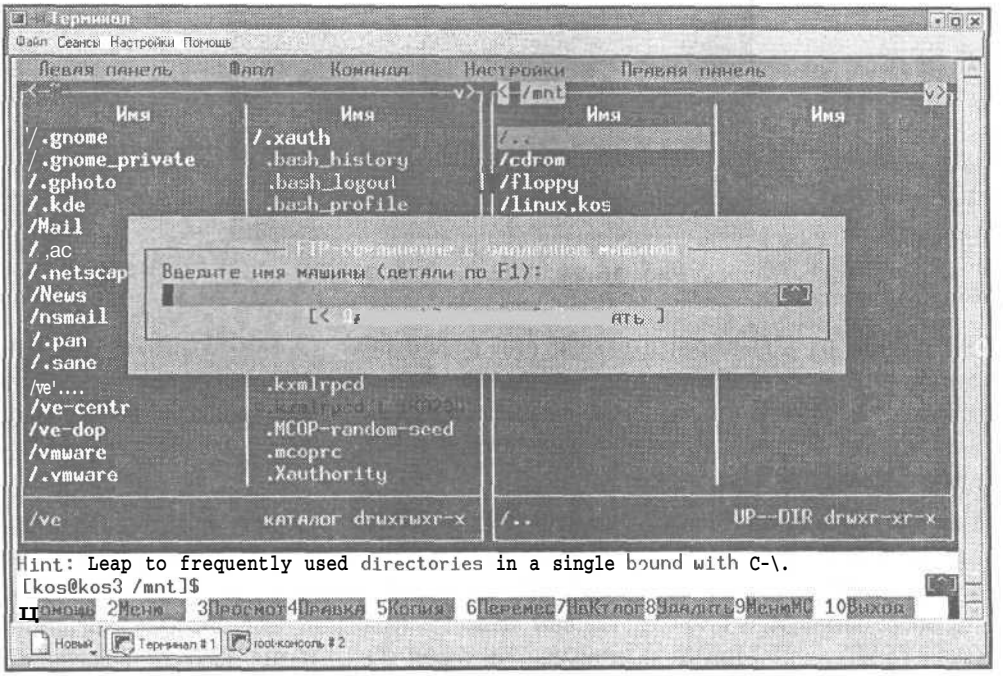


Рис. 15.28. Соединение с FTP-сервером

15.8. Средства мультимедиа и игры

Ну и, в заключение этой главы, осталось рассмотреть средства мультимедиа. Правда, сам я, за неимением времени, эти средства использую крайне редко. Поэтому рекомендую желающим просмотреть обзор средств мультимедиа, который имеется в статьях А. Федорчука (их легко найти в Интернете), а также в его книге *(см. [П1.6] приложения)*. А для тех, кто не желает тратить время на поиски, приведу краткий обзор. Начну с программ для воспроизведения аудио-записей.

15.8.1. Звук

Естественно, предполагается, что звуковая карта у вас установлена и настроена *(см. гл. 9)*. Я уже упоминал, что в составе KDE имеется стандартный

проигрыватель аудиодисков. Однако после установки Red Hat CE 7.1 я обнаружил в главном меню KDE другую программу — XPlayCD (рис. 15.29).

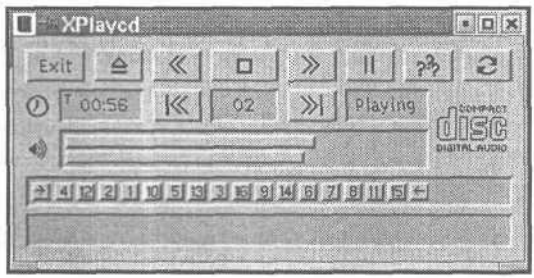


Рис. 15.29. Главное окно XPlayCD

В отличие от CD-проигрывателя она рассчитана на стереозаписи: две полоски рядом с изображением динамика позволяют отдельно регулировать громкость звука в правой и левой колонках (щелкайте мышкой по полоске или справа от нее). Других преимуществ у XPlayCD по сравнению с CD-проигрывателем я не увидел.

Как CD-проигрыватель, так и XPlayCD позволяют только прослушивать аудиозаписи в том формате, который используется на CD-ROM. Если же вы хотите воспроизвести какой-либо файл формата MP3, получившего большое распространение в Интернете, вам надо воспользоваться программой xmms (рис. 15.30) — X MultiMedia System (которая до 10 июня 1999 г. называлась X11Amp). Последняя устанавливается вместе с KDE, а по виду и возможностям очень напоминает известный Windows Amplifier, или WinAmp. Версия 1.2.4, которая установилась у меня из дистрибутива Red Hat CE 7.1, была довольно капризна, но после того как я скачал (с сайта http://www.xmms.org) и установил версию 1.2.5, программа начала работать вполне устойчиво.

Рис. 15.30. Главное окно программы xmms

Рассмотрим некоторые элементы главного окна программы. В правом верхнем углу вы видите три обычные кнопки управления окном: свернуть в значок на панели, минимизировать (оставить только заголовок окна), закрыть программу.

В том варианте, который загружается по умолчанию, хоть и с трудом, но можно разглядеть на черном фоне пять вертикально расположенных букв: O, A, I, D, V. Щелчок левой кнопкой мыши по одной из этих букв вызывает следующую реакцию.

- O — появляется меню настроек программы;
- A — по идее должно заставлять программу выводить свое окно поверх всех остальных окон, но пока, видимо, не работает;
- I — выводит информацию о воспроизводимой мелодии (естественно в том объеме, который сохранен в MP3-файле);
- D — увеличивает вдвое размер окна;
- V — появляется меню управления режимами визуализации.

Рядом с буквами находится панель визуализации, в которой отображается время воспроизведения (либо прошедшее, либо оставшееся, в зависимости от настройки) и спектрограмма мелодии (вид которой тоже можно настроить).

Под названием программы находится панель, в которой во время воспроизведения бегущей строкой отображается название воспроизводимой мелодии и ее номер в текущем списке (playlist). Щелчком правой кнопки мыши по этой панели вызывается еще одно окно настроек (которое можно вызвать также щелчком левой кнопки по непонятному значку в верхнем левом углу окна программы). Поскольку все команды упоминавшихся меню русифицированы (по крайней мере у меня), то описывать их назначение не имеет смысла. Ниже бегущей строки с названием мелодии выводится скорость битового потока (MP3 bitrate in KBps), обычно 128 или 112 Кбит/с, ширина полосы в килогерцах (обычно 44) и индикатор стерео- или монорежима.

Ниже той панели, о которой мы только что говорили, находятся два ползунка — регулятор громкости и баланса, и две кнопки. Первая из этих кнопок (EQ), вызывает графический эквалайзер, а вторая (PL) — окно управления списком воспроизводимых мелодий (playlist). Ниже располагается индикатор (в виде ползунка), показывающий, какая часть мелодии воспроизводится.

Еще ниже располагаются обычные для CD-проигрывателя кнопки управления воспроизведением (переключение на предыдущую мелодию списка, воспроизведение, пауза, стоп, переключение на следующую мелодию в списке, извлечение CD-диска, что в данном случае означает загрузку нового списка мелодий). Далее следуют кнопки воспроизведения мелодий из спи-

ска в случайном порядке и повторного воспроизведения списка после его завершения.

Большинству настраиваемых опций программы соответствуют "горячие" клавиши, однако приводить здесь их перечень я не вижу смысла: все они указаны в соответствующих командах меню, и те, кто привык пользоваться больше клавиатурой, чем мышью, легко смогут их найти и заучить. Я также не буду объяснять назначение и приемы работы с графическим эквалайзером: вы либо знаете это лучше меня, либо, как и я, вообще не пользуетесь этим устройством.

Таким образом, осталось рассказать только о том, как использовать списки мелодий ("плей-листы"). Итак, чтоб вызвать окно редактирования списка мелодий, щелкните по маленькой кнопке, рядом с которой стоят буквы PL. Появится окно, изображенное на рис. 15.31, в котором имеется 5 кнопок.

Рис. 15.31. Редактор списков мелодий

Эти кнопки ведут себя несколько необычно. Каждая из них предоставляет несколько вариантов выбора, но чтобы произвести выбор, надо не просто щелкнуть по кнопке, а, удерживая кнопку нажатой, перевести указатель мыши на нужный вариант и только после этого отпустить клавишу мыши. Давайте рассмотрим назначение этих кнопок (в порядке их расположения – слева направо). В табл. 15.1 приведены изображения того, что вы увидите, когда будете удерживать каждую кнопку, и что получите, отпустив кнопку мыши на каждом из появляющихся вариантов.

Таблица 15.1. *Кнопки управления playlist*

Кнопка	Описание
	• + **URL** — появится строка ввода, в которой надо будет ввести URL файла в Интернете (интранете); только имейте в виду, что в этой строке можно ввести имя конкретного файла с мелодией и нельзя ввести адрес списка мелодий; • + **DIR** — появляется окно с деревом каталогов, позволяющее вам выбрать каталог с файлами мелодий; все файлы из этого каталога будут включены в список мелодий; • + **FILE** — появится диалоговое окно выбора файлов, а после выбора файла (только одного) соответствующая мелодия будет включена в список (в виде имени файла или названия, если программа сумеет извлечь его из файла)
	• **- MISC** — появляется предложение удалить "мертвые" (dead) файлы, т. е. файлы, ссылка на которые присутствует в списке, но является по каким-либо причинам ошибочной (например, файл уже удален с диска); • - ALL — удаление всех файлов из текущего списка; • **— CROP** — удалить из списка все файлы кроме тех, которые помечены; • **- FILE** — удалить помеченные файлы
	• **INV** SEL — инвертировать текущие пометки; • **SEL ZERO** — снять все отметки с файлов текущего списка; • SEL ALL — отметить все файлы текущего списка; При выборе файлов для пометки можно пользоваться клавишами <Shift> и <Ctrl>, чтобы выбрать несколько файлов подряд или добавить к числу отмеченных отдельные файлы
	• **SORT LIST** — появляется дополнительное меню, в котором можно выбрать один из возможных вариантов сортировки списка; • **FILE INF**— появляется окно с информацией о текущей мелодии (если, конечно, программа в состоянии извлечь такую информацию из файла); • **MISC OPT** — эта кнопка служит только для доступа к двум другим вариантам, доступным по этой кнопке
	• **NEW LIST** — удаляет текущий список мелодий; • **SAVE LIST** — позволяет сохранить текущий список мелодий в отдельном файле; • **LOAD LIST** — позволяет загрузить список мелодий из файла

Загрузить список файлов в xmms можно не только с помощью редактора списка мелодий, но и непосредственно из командной строки, с помощью одной из следующих команд:

```
xmms file1.mp3 file2.mp3 file3.mp3
xmms *.mp3
xmms playlist.m3u
```

Если в это время программа xmms уже запущена, то текущий список мелодий будет удален, а список, заданный командной строкой, будет загружен.

Что еще надо сказать, так это то, что xmms может воспроизводить не только файлы формата MP3, но и обычные аудио-CD. Только для того, чтобы создать список мелодий в этом случае, надо воспользоваться кнопкой **+DIR** и выбрать каталог /mnt/cdrom.

15.8.2. Видео

Программа aKtion

Если среди программ для прослушивания аудио под Linux имеется явный лидер (я имею в виду xmms), то среди программ для воспроизведения видео такого лидера пока нет. А. Федорчук для просмотра видео рекомендует программу aKtion. Устанавливается она, правда, не из всех дистрибутивов. На моем домашнем компьютере, где стоит ALT Linux Junior 1.0, эта программа нашлась, а вот в составе дистрибутива Red Hat 7.1 Cyrillic Edition ее не обнаружилось. Зато там есть программа Noatun, которая тоже может воспроизводить видео. Однако aKtion показалась мне более совершенной. Давайте вначале ее и рассмотрим. Ведь любую программу можно установить, даже если она не входит в дистрибутив. Программу aKtion, а также все необходимые для нее библиотеки можно найти либо на сайте KDE **(http://www.kde.org)** или на сайте программы xanim (**http://xanim.va.pubnix.com/home.html**), т. к. aKtion представляет собой графическую оболочку для Xanim. Она поддерживает многие видео-форматы, включая MPEG1, QuickTime и анимированный GIF.

После запуска программы вы увидите окно, изображенное на рис. 15.32.

Управление программой осуществляется очень просто. Вдоль нижней границы окна располагаются семь кнопок. Над кнопками расположен ползунок регулятора громкости. Первая из экранных кнопок открывает стандартное диалоговое окно KDE **Открыть файл** с возможностью задания фильтров отображения файлов тех форматов, которые поддерживаются программой aKtion. Назначение следующих четырех очевидно из тех изображений, которые на них нанесены: воспроизведение, остановка, перемотка к началу, переход в конец (правда, у меня две последних кнопки действуют как-то не так, как ожидается). Следующая кнопка, с изображением гаечного ключа,

служит для вызова окна меню настроек (рис. 15.33), а последняя кнопка вызывает подсказку по программе.

Рис. 15.32. Окно программы aKtion после запуска

Окно настроек программы можно также вызвать, если во время демонстрации видео удерживать нажатой правую кнопку мыши в любой точке выводимого изображения. Появится выпадающее меню, содержащее несколько команд для управления изображением и среди них команду **Настройка программы.**

Окно настроек содержит 5 вкладок. Начнем с вкладки **Разное** (см. рис. 15.33). Первая группа параметров определяет способ загрузки видеофайла. При первом варианте **(Загружать в память)** файл просто полностью загружается в память до начала воспроизведения. Во втором варианте **(Загружать в память и декодировать)** файл не только загружается в память, но еще и производится его декомпрессия, и только после этого начинается его воспроизведение. В третьем варианте **(Загружать только необходимые участки)** предварительная загрузка не производится, части файла загружаются в память по мере необходимости. Последний вариант используется по умолчанию.

Следующие три параметра касаются использования разных типов памяти. Я тут использую значения, установленные по умолчанию.

Далее идет строка ввода, в которой можно указать, с какого каталога будет начинаться поиск нужного видеофайла (если строка пуста, то начальной точкой поиска будет ваш домашний каталог). Вы можете ввести полное имя нужного каталога или выбрать его, нажав кнопку справа с тремя точками.

Ниже строки, задающей начальный каталог, идут два параметра, назначение которых очевидно из названий: **Воспроизводить в цикле и Автоматически начинать воспроизведение видео после открытия файла,** а затем две строки ввода для задания местоположения файла программы xanim и ее параметров.

На вкладке **Аудио** особых пояснений требует только одна опция: **Пропускать кадры для синхронизации видео и звука.** Эта опция необходима при работе

на медленных машинах, на которых видео и аудио могут воспроизводиться 4 разной скоростью (изображение отстает от звука). В таком случае надо включить данную опцию. Это приведет к снижению качества воспроизведения изображения, поскольку часть кадров видео будет пропускаться, но результат может быть вполне приемлемым (все зависит от скорости процессора и объема памяти в компьютере).

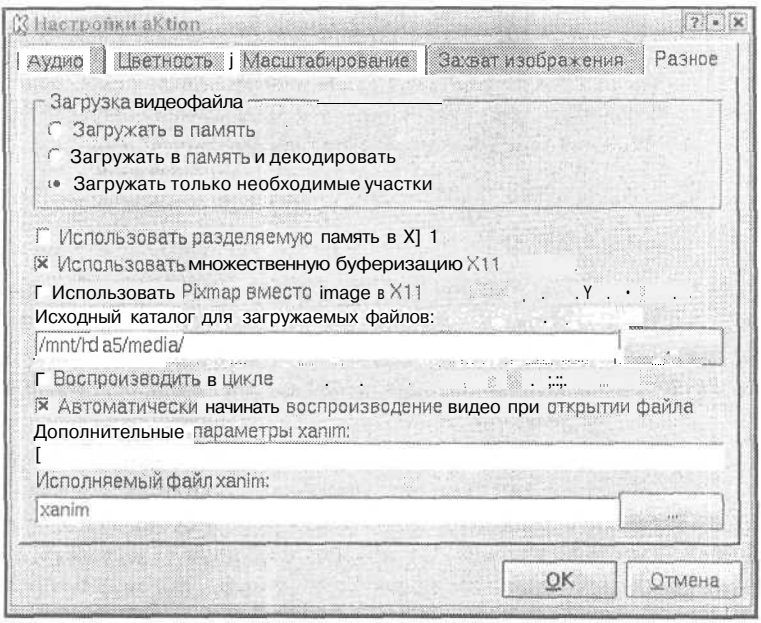


Рис. 15.33. Окно настроек программы aKtion

Выключение параметра **Показывать уровень громкости** приводит к исчезновению ползунка регулятора громкости в основном окне программы, а переключатель **Начальная громкость** задает уровень громкости, устанавливаемый при запуске программы.

Вкладка **Цветность** содержит некоторые параметры, менять которые может оказаться необходимым в тех случаях, когда ваш дисплей не позволяет воспроизводить True Color (16-битные цвета). По умолчанию здесь задано отсутствие преобразований цвета (Нет). Установка значений, больших 1,0, в последней строке вкладки **(Гамма дисплея)** приводит к тому, что изображение становится ярче.

Вкладка **Масштабирование** позволяет выбрать режим вывода изображения: от половинного до полноэкранного.

Программа aKtion предоставляет возможность сделать снимок экрана в процессе вывода видео, для чего надо в нужный момент нажать клавишу <C>.

Вкладка **Захват изображения** позволяет задать формат файла с захваченным изображением и указать каталог, в котором этот файл будет сохранен. Возможны следующие форматы выходных файлов: BMP, JPEG, PBM, PGM, PNG (используется по умолчанию), PPM, XBM, XPM.

Поскольку aKtion является просто оболочкой к программе xanim, то список поддерживаемых форматов видеофайлов вы можете просмотреть по команде man xanim. К сожалению, с помощью aKtion удается просмотреть далеко не любой видеофайл. По крайней мере мне не удалось с ее помощью просмотреть два недавно приобретенных CD-диска с фильмами в формате MPEG4. А вот программа xine из дистрибутива ASP Linux 7.2 с ними справилась.

Программа Xine

Аудио/видеопроигрыватель xine (произносится "кси:н" с долгим "и") предназначен как для воспроизведения отдельных аудио- или видеопотоков (то есть файлов .mp3 или .mpv), так и AVI-файлов (используя win32-кодеки), а также фильмов в формате видео-CD (VCD), SVCD и DVD (незащищенных для воспроизведения защищенных DVD-форматов требуется установить дополнительный плагин). Домашняя страница программы находится по адресу **http://xine.sourceforge.net**, а RPM-пакет можно найти на **http://rufus.w3.org.** Программа рассчитана на воспроизведение следующих мультимедиа-форматов:

- аудиоформаты:
  - MPEG audio уровней 1, 2 и 3 (известный также как MP3);
  - a/52 (известный также как AC3 и Dolby Digital);
  - DTS (через встроенный декодер);
  - OggVorbis (открытая альтернатива MP3);
  - используя Video For Windows Dynamic-Link-Libraries (DLL), можно воспроизводить файлы форматов DivX audio (WMA), ADPCM, GSM и многих других.
- видеоформаты:
  - MPEG1 и MPEG2;
  - MPEG4, известный также как OpenDivX;
  - Microsoft MPEG4, известный также как DivX;
  - Motion JPEG;
  - используя Video For Windows Dynamic-Link-Libraries (DLL) можно воспроизводить файлы форматов Indeo, Cinepak, Windows Media 7/8 и многих других;

- комбинированные форматы (видео+аудио в одном файле):
  - незащищенные варианты DVD;
  - Video CD (исключая гибридные форматы Video-on-CD);
  - программные потоки MPEG (то есть файлы .mpg и .mpeg);
  - MPEG transport streams (.ts);
  - Ogg Vorbis container streams (.ogg);
  - Microsoft Audio Video Interleave (.avi);
  - Microsoft Advanced Streaming Format (.asf);
  - Quicktime - - Note: Xine does not support the Sorenson codec used in many Quicktime files;
  - Raw MPEG audio and video streams (.mpv и .mp3).

Для достижения нормального качества воспроизведения файлов формата MPEG2 xine требует системы на процессоре Pentium II с частотой не ниже 400 МГц.

Запускать xine можно из меню KDE или из командной строки. Когда xine запускается в первый раз, на экране монитора появляется главное окно программы (окно просмотра) — черный прямоугольник с логотипом программы (рис. 15.34).

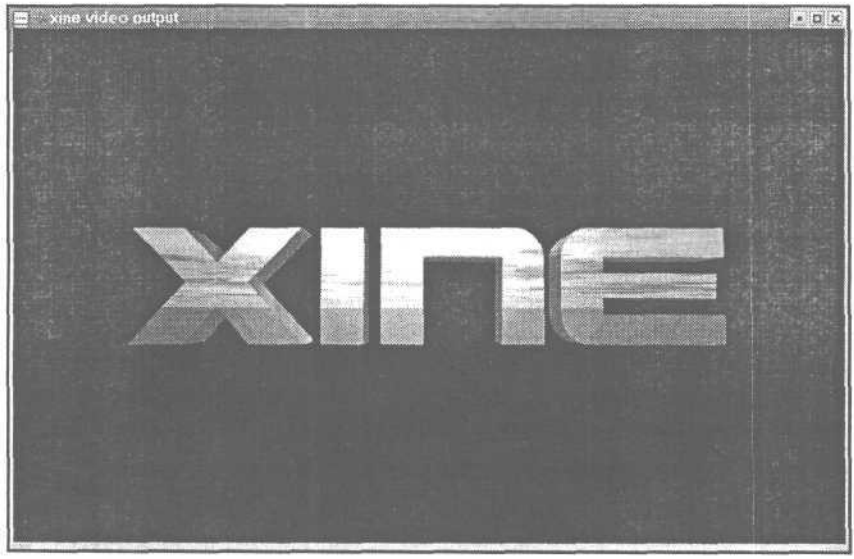


Рис. 15.34. Главное окно программы xine (окно просмотра)

Одновременно обычно выводится и второе окно с панелью управления (рис. 15.35). Если оно не появилось, вы можете вызвать его щелчком правой кнопки мыши в окне просмотра или нажатием клавиши <G> на клавиатуре. Таким же образом можно отключить вывод панели управления ("спрятать -Панель"). На панели управления имеется набор кнопок, аналогичный набору управляющих кнопок на панели обычного видеомагнитофона: кнопки запуска воспроизведения, временной остановки (pause), перемотки в начало и конец и т. д.

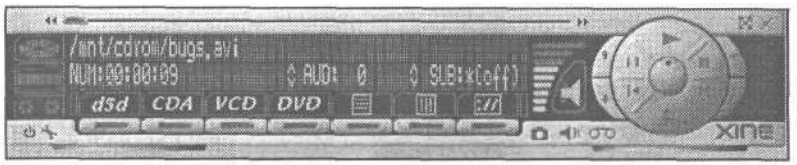


Рис. 15.35. Панель управления программы xine

Если вы, например, хотите воспроизвести VCD или DVD, то после появления на экране панели управления щелкните кнопкой мыши по кнопке с надписью VCD (или DVD). Это заставляет xine искать диск в соответствующем устройстве. Если диск найден, вы можете запустить воспроизведение щелчком по кнопке с изображением треугольника в верхней части круглой "ручки" на правой стороне панели.

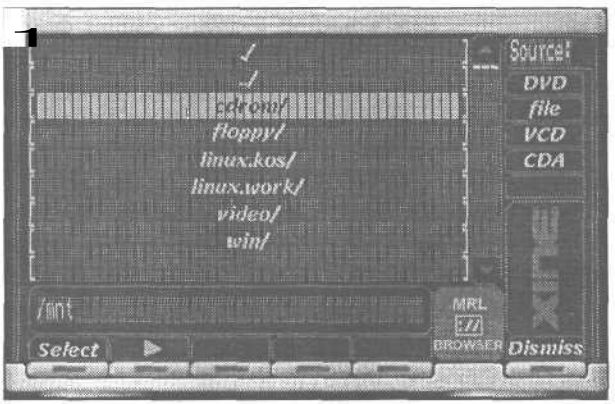


Рис. 15.36. Выбор файла для воспроизведения

Если же вы хотите воспроизвести фильм, который находится в файле на жестком диске или CD, надо сначала добраться до нужного файла. Для этого нужно открыть окно MRL-браузера (рис. 15.36), что достигается щелчком по кнопке на панели управления, над которой изображен значок ://. Аббре-

виатура **MRL** означает Media Resource Locator, т. е. указатель медиа-ресурса(ов). В качестве такого указателя могут выступать:

- полный путь к файлу на диске, перед которым стоит указание на тип MRL в виде "file://", например, file://some/file.vob;
- указатель на программный канал: fifo://[[mpeg1:mpeg2]:/]some/fifo. В этом случае xine получает видео из специального программного канала /some/fifo. По умолчанию xine предполагает, что входной поток имеет формат MPEG2, если же нужно воспроизвести формат MPEG1 или MPEG2, нужно явно указать это в MRL;
- указатель на стандартный ввод stdin: stdin://[mpeg1:mpeg2]. Как и в случае с программным каналом (fifo:// MRL), xine предполагает, что вход идет в формате MPEG2, если ему явно не указано противное;
- указание на DVD-диск: dvd://VTS_xx_y.VOB. Отличие этого варианта от задания источника в виде file:// MRL состоит в том, что xine читает данные непосредственно с DVD, не используя файловую систему;
- vcd://track — воспроизводится указанный трек с VCD;
- html://server.somewhere.tld/foo/bar.mpg -- xine обращается к указанному URL и пытается воспроизвести файл.

В окно MRL-браузера вы вначале должны выбрать тип источника, щелкнув левой кнопкой мыши по одной из кнопок в поле окна, обозначенном словом Source. Не могу вам сказать, что произойдет после выбора VCD, DVD или CDA — не было у меня дисков в этих форматах. Если же вы выберете вариант file, то в основном поле окна отобразится содержимое текущего каталога. Дальше можно перейти в любой каталог вашей файловой структуры (в том числе в каталог, куда смонтирован CD-ROM) и выбрать файл для воспроизведения. Проще всего переходы по структуре каталогов осуществлять, щелкая мышью по нужному каталогу (чтобы переместить на него подсветку), а затем щелкая по кнопке Select. Когда вы уже добрались до нужного файла и переместили на него подсветку, нужно вместо Select щелкнуть по кнопке с изображением треугольника. Начнется показ выбранного фильма в окне просмотра. Если же после выбора файла щелкнуть по кнопке Select, то для начала воспроизведения придется воспользоваться упоминавшейся выше кнопкой панели управления с изображением треугольника.

Окно MRL-браузера можно закрыть щелчком по кнопке Dismiss. Можно также одним щелчком правой кнопки мыши по свободному полю в окне просмотра разом закрыть все вспомогательные окна (включая окно панели управления). Тот же эффект достигается нажатием клавиши <G> (обратно на экран они вызываются тем же способом).

Все фильмы (или файлы), которые вы просматривали в текущем сеансе работы с программой, запоминаются в списке просмотренных фильмов (или плей-листе — playlist). Этот список можно просмотреть в отдельном окне (рис. 15.37), которое называется Playlist Editor. Вызвать его на экран можно

щелчком по четвертой слева кнопке на панели управления xine. В список можно добавлять фильмы (кнопка **Add** справа) или удалять из него как отдельные позиции, так и весь список разом с помощью кнопок в нижней части окна (всмотритесь в изображения над кнопками и вы поймете их назначение). Можно также сохранить список фильмов (кнопка Save) или загрузить ранее сохраненный список (кнопка **Load** в правой стороне окна).

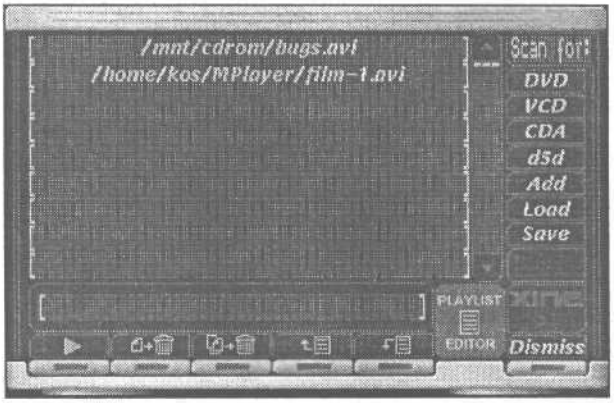


Рис. 15.37. Список фильмов/файлов (playlist)

Программа имеет еще одно вспомогательное окно, о котором стоит упомянуть (рис. 15.38). Оно появляется после нажатия на пятую слева кнопку на панели управления. Назначение этого окна состоит в изменении некоторых параметров вывода изображения на экран и изменения "шкурок" (skins), т. е. изменения внешнего вида окон. Впрочем, имеющиеся в этом окне ползунки изменения яркости, контраста и других параметров изображения в текущей версии (0.9.8) не работают. А вот "шкурки" можно менять и даже добавлять новые, скачивая их из Интернета (так же как в программах XMMS или Winamp). Для смены "шкурки" достаточно выбрать ее имя щелчком кнопки мыши по ее имени в правой части диалогового окна изменения параметров (рис. 15.38).

А теперь вернемся к описанию панели управления программы xine, изображенной на рис. 15.35. После того как вы выбрали файл для воспроизведения, имя этого файла высвечивается на панели. Если воспроизведение не началось автоматически, можно запустить его, щелкнув по треугольнику, расположенному в верхней части круга (или круглой "ручки") панели справа. Остальные значки на этом круге выполняют следующие функции (перечисление идет по часовой стрелке):

- остановить воспроизведение;
- начать воспроизведение следующего фильма из списка (плей-листа);

- ❒ извлечь носитель из дисковода (естественно, не работает, если загружен файл с жесткого диска);
- О начать воспроизведение предыдущего фильма из списка (плей-листа);
- ❒ пауза/продолжить воспроизведение.

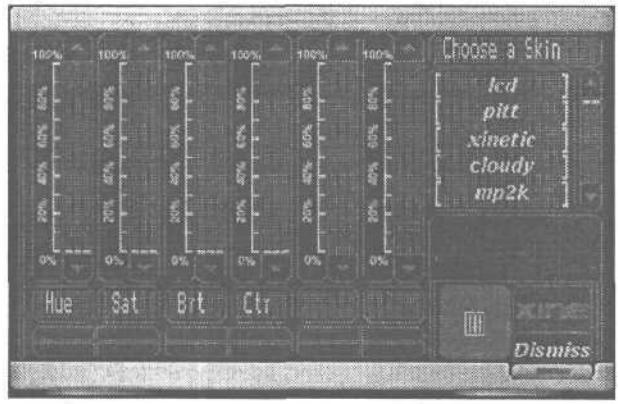


Рис. 15.38. Окно изменения параметров

Слева из-под круга "выступают" две кнопки, которые позволяют ускорить или замедлить воспроизведение (две аналогичных кнопки справа от круга в текущей версии не работают). Левее круга расположен регулятор громкости звука в виде изображения динамика (или громкоговорителя). Щелчком мыши по этому изображению можно включить или полностью выключить звук (тот же эффект достигается щелчком по уменьшенному изображению динамика, расположенному ниже основного). Плавное изменение громкости производится с помощью щелчков мышкой по горизонтальным полоскам, находящимся левее основного выключателя звука. Ниже этого выключателя находится изображение фотоаппарата, с помощью которого можно получить моментальный снимок экрана, наподобие того, что изображен на рис. 15.39.

В правом верхнем углу панели управления находятся два значка в виде крестиков. Правый из них служит для выхода из программы, а левый (крестик со стрелками) — для переключения между полноэкранным и оконным режимами работы программы. Замечу, что при переключении в полноэкранный режим панель управления и вспомогательные окна остаются на экране. Если они мешают, их можно убрать с экрана (и вернуть на экран при необходимости) с помощью клавиши **<G>.**

В левом нижнем углу панели вы тоже видите два значка. Один из них — выключатель проигрывателя, а второй (сильно уменьшенное изображение гаечного ключа) вызывает окно настроек. У программы масса возможностей по

настройке и конфигурации, однако рассматривать их мы не будем. Если вас этот вопрос заинтересует, прочитайте комментарии в файле ~/.xine/config.

И, наконец, ползунок-индикатор в верхней части панели управления позволяет быстро перейти к любой стадии воспроизведения (просто щелкните по горизонтальной черте в нужном месте).

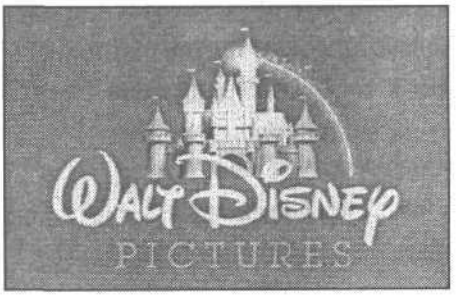


Рис. 15.39. Начинаем просмотр фильма

Мы рассматривали только графическую оболочку программы xine. Между тем, можно запустить xine просто из командной строки, причем в командной строке можно указать дополнительные аргументы. Например, по команде

```
$ xine --help
```

вы получите краткую сводку возможных значений аргументов командной строки и список управляющих клавиш. Можно также непосредственно запустить воспроизведение из командной строки, указав источник видео:

```
$ xine <mrl>
```

где `<mrl>` — указатель медиаресурса, т. е. Media Resource Locator. Например, для воспроизведения файла /some/where/foo.vob команда должна иметь вид:

```
$ xine file://some/where/foo.vob
```

или просто

```
$ xine /some/where/foo.vob
```

Для воспроизведения третьего трека с видео-CD (VCD) надо дать команду:

```
$ xine vcd://3
```

а для прямого доступа к DVD — команду типа:

```
$ xine dvd://VTS_01_1.VOB
```

И в заключение приведу список комбинаций клавиш, с помощью которых можно управлять программой в процессе воспроизведения (вне зависимости

от того, каким образом она была запущена), не вызывая на экран панель управления (но при условии, что окно программы является активным):

- ❒ <Enter> — начать воспроизведение;
- ❒ <пробел> — пауза;
- О <↑> — ускорить воспроизведение;
- *П* <↓> — замедлить воспроизведение;
- П <0> — перейти в начало текущего потока (фильма или мелодии);
- *П* <1>—<9> — перейти в точку, отстоящую на 10—90% от начала текущего потока;
- *П* <←> — сместиться на 15 секунд назад;
- П <→> — сместиться на 15 секунд вперед;
- *П* <Ctrl>+<←> — сместиться на 60 секунд назад;
- П <Ctrl>+<→> — сместиться на 60 секунд вперед;
- П <PgUp> — перейти к предыдущему пункту в списке фильмов (playlist);
- П <PgDown> — перейти к следующему пункту в списке фильмов (playlist);
- ❒ <F> — переключение в полноэкранный режим (и обратно);
- П <A> — переключение отношения размеров сторон окна просмотра (AUTO/16:9/4:3/DVB);
- *П* <I> — переключение в режим черезстрочного вывода (interlaced mode);
- П <, > — увеличить/уменьшить размер окна просмотра;
- О <+> или <–> — переход на следующий/предыдущий аудиоканал, при этом на панели управления изменяется значение указателя **AUD:** (эта опция используется в том случае, когда в источнике записано несколько аудиопотоков, как это иногда бывает на DVD);
- *П* <,> <.> — сменить канал субтитров (change subtitle channel), при этом на панели управления изменяется значение указателя **SUB:**;

Примечание

Нужно, наверное, сказать, что субтитры бывают записаны в файле видеофильма в виде отдельного потока (подобно аудио). Поток 0 соответствует субтитрам на языке, используемом по умолчанию. Если имеются субтитры на других языках, эта опция позволяет выбрать нужный поток.

- ❒ <n>, <m> — синхронизировать аудио- и видеопотоки;
- *П* <Ноте> — переустановить синхронизацию аудио- и видеопотоков;
- П <G> — скрыть управляющие окна (панель, плей-лист и т. п.);

- <H> — скрыть окно просмотра (эта клавиша работает только в том случае, если панель управления отображается на экране); такая возможность бывает полезна при воспроизведении аудиопотоков, например, формата MP3 (а также при неожиданном появлении начальника);
- <Q> — выход из программы.

В заключение нужно сказать, что программа xine (по крайней мере, в том виде, как она поставляется разработчиком), не может воспроизводить зашифрованные DVD-диски. Дело в том, что некоторые производители DVD-дисков с целью увеличения собственной прибыли защищают свои диски с помощью собственных алгоритмов шифрования, так что воспроизводить эти диски можно только на оборудовании (или с использованием программного обеспечения) этих же производителей. Как сообщается в документации к программе xine, включить в программу код, обеспечивающий возможность воспроизведения таких записей не было бы проблемой. Но это, возможно, стало бы нарушением законодательства некоторых стран (в первую очередь — американского). Поэтому разработчики не обеспечивают такой возможности, хотя и считают использование алгоритмов шифрования нарушением прав потребителей.

Программа MPlayer

В заключение раздела, посвященного воспроизведению видео, я хочу упомянуть еще об одной программе, о которой я пока знаю только понаслышке. Просто в списке рассылки mandrake-russian я как-то увидел письмо, автор которого выражал недовольство программой xine и спрашивал, нет ли чего лучше. Ему ответили, что существует гораздо более удобная программа, которая называется MPlayer, и сообщали, где ее можно найти (**http://mplayerhq.hu**). Я, естественно, заглянул по указанному адресу. Как сообщается на сайте разработчиков, MPlayer может воспроизводить видео следующих форматов:

- VCD (Video CD), непосредственно с CD-ROM или из соответствующего файла бинарного образа диска;
- DVD, непосредственно с DVD-диска, используя для дешифрования библиотеку libdvdread;
- MPEG 1/2 System Stream (PS/PES/VOB) и Elementary Stream (ES);
- RIFF AVI;
- ASF/WMV 1.0;
- QT/MOV со сжатыми и несжатыми заголовками;
- VIVO (viv-файлы)
- **FLI;**

- RealMedia (rm-файлы);
- NuppelVideo (nuv-файлы);
- yuv4mpeg.

Программа поддерживает чтение из входного потока (stdin) или из сети по протоколу HTTP.

MPlayer в основном подпадает под условия лицензии GPL, но содержит и код, который распространяется на других условиях, в частности, библиотеку OpenDivX, которая имеет специальную лицензию.

Разработчики поставляют эту программу только в исходных кодах, заявляя, что ее распространение в бинарных кодах пока невозможно как по техническим, так и по юридическим причинам.

Более подробное описание программы MPlayer я не могу привести, поскольку узнал о ней на самом последнем этапе работы над данной книгой и не успел ее установить и освоить. Однако я посчитал необходимым упомянуть о ней, учитывая положительные отзывы в листах рассылки.

В заключение раздела остается только сказать, что, как показывают приведенные описания, уже сейчас (несмотря на то, что разработка упомянутых в этом разделе программ еще не завершена) имеются вполне работоспособные программные средства для воспроизведения видео под Linux.

15.8.3. Игры

Конечно, далеко не каждый пользователь компьютера тратит много времени на компьютерные игры, но вряд ли найдется такой из нас, кто никогда не запускает их в свободную минуту. Почему-то широко распространено следующее мнение: "Linux — это операционная система для программистов. Если вы хотите играть, запускайте Windows!". Однако это далеко не так и я попытаюсь обосновать это утверждение в настоящем разделе.

Минимальный набор игр и развлечений включен в состав графической среды KDE. Я лично, вполне довольствуюсь тем, что там есть, и даже всего тремя играми (рис. 15.40): Тетрис, Маджонг и Пасьянс.

Но для настоящего игрока этот набор, конечно, покажется очень бедным. Впрочем, стандартный набор игр, входящий в состав Windows, еще беднее. "Настоящие" игры — это, по мнению большинства, игры трехмерные, похожие на видеофильмы, главным действующим лицом в которых является играющий. Но такие игры требуют хорошей видеокарты и программной поддержки трехмерной графики. Третья версия пакета XFree86 содержала лишь код, необходимый для 2D-приложений. Чтобы играть в 3D-игры, нужно было добавлять специальное ПО. В четвертую версию пакета XFree86 включен быстродействующий механизм трехмерной графики. Кроме того, была разработана открытая спецификация OpenAL, которая обеспечивает в играх

поддержку 3D-аудио. Точно так же, как переход Microsoft с DirectX 5 на DirectX 6 изменил ситуацию для разработчиков игр под Windows, переход от версии 3 к версии 4 XFree86 с добавлением OpenAL и других разработок позволил перенести под Linux самые популярные игры, ранее существовавшие только в Windows-версиях, а также и разработать новые.

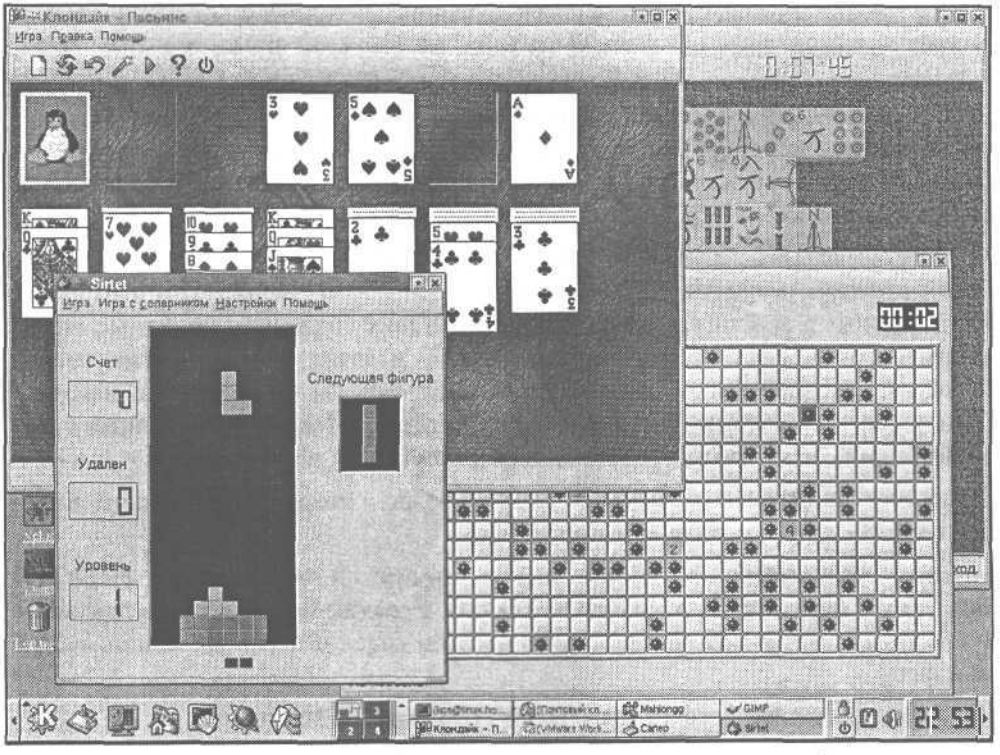


Рис. 15.40. Игры из стандартного набора KDE

Одна из самых известных игр, — Doom, созданная компанией id Software, впервые была опубликована в конце 1993 г. В этой игре вы должны пройти сложный лабиринт, сражаясь с фантастическими монстрами, которые стремятся вас "убить". Существуют сетевые версии этой игры. Первоначально выпущенная для MS-DOS, Doom была затем перенесена на NeXT, SGI, Macintosh-и, Atari Jaguar, MS Windows. Первая версия для Linux была разработана Дэвидом Тейлором (David Taylor). В декабре 1997 г. id Software выпустила Doom под лицензией GNU General Public License. Вы можете найти версию этой игры под названием LxDoom по адресу **http://lxdoom.linuxgames.com/current.html**.

Еще в 1995 г. была начата разработка проекта FreeCiv — клона популярной игры Civilization. FreeCiv обеспечивала режим коллективной игры через Интернет еще за несколько месяцев до того, как компания Hasbro Interactive включила подобные возможности в свой продукт.

Но самый большой вклад в дело убеждения разработчиков и пользователей в том, что к Linux как к платформе для игр можно относиться серьезно, внесла другая игра — Quake_ XE "Игры:Quake" _, которая вышла в 1996 г. К 1997 г. один из основателей фирмы Id Software Джон Кармак (John Carmack) предложил сообществу разработчиков Quake на условиях лицензии GNU General Public License, а программист Дейв Кирш (Dave Kirsh) из той же компании перенес эту игру на Linux. Сегодня компании используют Quake III для демонстрации новейшей аппаратуры и дистрибутивов Linux. Кармак (и его компания) активно поддерживает идею игр на Linux; он перенес на открытую платформу и Quake III: Arena.

Таким образом, к настоящему времени все типы популярных компьютерных игр доступны в версиях для ОС Linux. Для Linux созданы и аркадные игры, и симуляторы, и стратегии, и "стрелялки", и игры любых других типов. В составе дистрибутивов полного комплекта игр вы, конечно, не найдете. Если у вас есть желание поиграть в такие игры, их надо устанавливать дополнительно, для чего либо купить их, либо скачать из Интернета.

Я закончу свой рассказ об играх ссылками, по которым вы сможете разыскать такую игру, которая вам понравится.

Начните знакомство с миром игр с чтения "Linux Gamers HOWTO" (**http://www.dirac.org/linux/LG-HOWTO.txt**). Русский перевод этого документа сделал Дмитрий Самойлов. Самую свежую версию перевода можно найти по адресу

http://linuxgames.hut.ru/data/docs/HOWTO/LG-HOWTO-ru.html

После этого можно отправиться на сайт **http://freshmeat.com,** где вы найдете около 300 различных игр. На русскоязычном сайте **http://linuxgames.hut.ru** имеется база данных по играм, в которой 184 игры, в том числе 143 — свободно распространяемые.

На сайте **http://linuxgames.org.ru/games.php3** вы найдете перечень игр под Linux, классифицированных по жанрам, с кратким описанием и ссылками на статьи с более подробными сведениями.

Если вы не нашли нужной игры на этих сайтах, можете заглянуть еще по следующим адресам:

- Linux Games **(http://www.linuxgames.com);**
- Linux Game Tome **(http://happypenguin.org);**
- Loki Entertainment Software **(http://www.lokigames.com);**

- Hyperion Software **(http://www.hyperion-software.com);**
- Tribsoft (**http://www.tribsoft.com/**);
- Unreal Tournament (**http://www.unrealtournament.com/**).

На этом я закончу рассмотрение вопроса о создании под Linux удобной рабочей среды пользователя. Изложение здесь поневоле получилось достаточно поверхностным. Но если сделать описание более подробным, то каждый раздел этой главы превратился бы в целую книгу. Я все же надеюсь, что приведенный мной обзор может служить обоснованием вывода о возможности решения той задачи, которая была сформулирована в начале главы, а именно, о возможности использования Linux для создания "удобной пользовательской среды".

Глава 16

Обратная сторона файловой системы

16.1. Типы файловых систем, поддерживаемых в Linux

Как уже было сказано в *гл. 4,* файловая система — одна из основных составляющих любой операционной системы, т. к. она обеспечивает хранение информации на физических носителях и доступ приложений к этой информации. В *разд. 4.4* была достаточно подробно рассмотрена та сторона файловой системы, которая обращена к пользователю — логическая структура каталогов и файлов. В этом разделе мы рассмотрим внутренние механизмы работы файловых систем, т. е. обратную (невидимую для пользователя) сторону файловой системы. Эта сторона обращена к физическим устройствам и определяет способ хранения информации на носителях, а также механизмы записи и извлечения этой информации по запросам приложений. Здесь в основе всего лежит способ адресации отдельных участков носителя и механизмы размещения отрезков файла по этим участкам.

Но, прежде чем перейти к описанию конкретных механизмов, стоит отметить, что Linux умеет работать с несколькими типами файловых систем. Основной файловой системой для Linux является "вторая расширенная файловая система" (second extended filesystem), которую кратко обозначают как ext2fs. Именно ее механизмы будут подробно рассматриваться в настоящем разделе. Но прежде, чем перейти к ее рассмотрению, ненадолго отвлечемся для того, чтобы кратко перечислить некоторые типы файловых систем, которые поддерживаются в Linux. Их список приведен в табл. 16.1.

Таблица 16.1. *Типы файловых систем, поддерживаемые Linux*

Тип файловой системы	Назначение
minix	Файловая система minix — это первая файловая система, которая использовалась в Linux. Она имела массу недостатков: ограничения размера раздела жесткого диска 64 мегабайтами; длина имени файла была ограничена 30 символами и т. д. Она продолжает использоваться для дискет и RAM-дисков

***Таблица 16.1** (продолжение)*

Тип файловой системы	Назначение
extfs	Еще одна из ранних версий файловой системы для Linux, расширение файловой системы minix. В настоящее время заменена файловой системой ext2 и уже не используются
ext2fs	Вторая расширенная файловая система (second extended filesystem) была создана как расширение файловой системы extfs. ext2fs обеспечивает более высокую производительность (в части скорости и использования центрального процессора), поддерживаются длинные имена и большие размеры файлов
xiaf	Файловая система xiaf была создана на основе minix с целью обеспечения большей устойчивости и безопасности. Она обеспечивает выполнение основных функций файловой системы без излишней сложности
msdos	Файловая система, используемая для разделов, сформатированных в MS-DOS и Windows. Имена файлов в msdos должны удовлетворять стандарту 8.3
umsdos	Файловая система UMS-DOS является расширением файловой системы DOS, используемым под Linux. В ней добавлено использование длинных имен файлов, идентификаторы пользователя и группы (UID/GID), разрешения в стиле POSIX и специальные файлы (устройства, именованные каналы и т. д.) при этом совместимость с DOS не потеряна
hpfs	Файловая система для разделов OS/2. В Linux обеспечивается только чтение из разделов hpfs
proc	Это файловая система, которая используется для обращения к структурам данных ядра. Файлы этой системы не занимают дискового пространства. Подробнее см. страницу man proc(5)
nfs	Сетевая файловая система, используемая для доступа к дискам, расположенным на удаленных компьютерах
swap	Раздел или файл свопинга ОС Linux
sysv	Файловая система Unix Systen V. Она поддерживает файловые системы Xenix FS, SystemV/386 FS и Coherent FS
iso9660	Файловая система для монтирования CD-ROM, соответствующая стандарту ISO 9660
vfat	Файловая система FAT-32. Поддерживаются длинные имена файлов
smb	Это сетевая файловая система, которая поддерживает протокол SMB, используемый Windows, Windows NT и Lan Manager. Для того чтобы использовать эту файловую систему, надо иметь специальную программу монтирования smbmount

Таблица 16.1 (окончание)

Тип файловой системы	Назначение
ncpfs	Это сетевая файловая система, обеспечивающая поддержку протокола NCP, применяемого в Novell NetWare. Для того чтобы использовать эту файловую систему, надо иметь специальную программу, которую можно найти на сайте **ftp://linux01.gwdg.de/pub/ncpfs**

Эту таблицу нельзя считать полной по той простой причине, что работа по созданию новых типов файловых систем для Linux продолжается постоянно. Примерами вновь разрабатываемых файловых систем являются журналируемая файловая система JFS фирмы IBM и файловая система ReiserFS. Эти системы и их отличия от основной на настоящий момент файловой системы Linux, ext2fs, я постараюсь охарактеризовать в конце данной главы.

А теперь подробнее рассмотрим основной на сегодняшний день тип файловой системы для Linux — ext2fs.

16.2. Структура дискового раздела в ext2fs

Производители жестких дисков обычно поставляют свои изделия отформатированными на низком уровне. Насколько я знаю, это означает, что все дисковое пространство с помощью специальных меток разбито на "сектора" размером 512 байт. Такой диск (или дисковый раздел) должен быть подготовлен для использования в определенной операционной системе. В MS-DOS или Windows процедура подготовки называется форматированием, а в Linux — созданием файловой системы. Создание файловой системы ext2fs заключается в создании в разделе диска определенной логической структуры. Эта структура строится следующим образом.

Во-первых, на диске выделяется загрузочная область. Загрузочная область создается в любой файловой системе. На первичном разделе она содержит загрузочную запись — фрагмент кода, который инициирует процесс загрузки операционной системы при запуске. На других разделах эта область не используется. Все остальное пространство на диске делится на блоки. Блок может иметь размер 1, 2 или 4 килобайта. Блок является адресуемой единицей дискового пространства. Выделение места файлам осуществляется целыми блоками, поэтому при выборе размера блока приходится идти на компромисс. Большой размер блока, как правило, сокращает число обращений к диску при чтении или записи файла, но зато увеличивает долю нерацио-

нально используемого пространства, особенно при наличии большого числа файлов маленького размера.

Блоки, в свою очередь, объединяются в группы блоков. Группы блоков в файловой системе и блоки внутри группы нумеруются последовательно, начиная с 1. Первый блок на диске имеет номер 1 и принадлежит группе с номером 1. Общее число блоков на диске (в разделе диска) является делителем объема диска, выраженного в секторах. А число групп блоков не обязано делить число блоков, потому что последняя группа блоков может быть неполной. Начало каждой группы блоков имеет адрес, который может быть получен как

$$((номер\_группы - 1)\times(число\_блоков\_в\_группе)).$$

Каждая группа блоков имеет следующее строение:

1. Супер-блок.
2. Описание группы блоков (Group Descriptors).
3. Битовая карта блоков (Block Bitmap).
4. Битовая карта индексных дескрипторов (Inode Bitmap).
5. Таблица индексных дескрипторов (Inode Table).
6. Область блоков данных.

Такая структура служит повышению производительности файловой системы за счет того, что сокращается расстояние между таблицей индексных дескрипторов и блоками данных, а, следовательно, сокращается время поиска нужного места головками в процессе операций записи/считывания файла.

Первый элемент каждой группы блоков (суперблок) одинаков для всех групп, а все остальные — индивидуальны для каждой группы. Суперблок хранится в первом блоке каждой группы блоков. Суперблок является начальной точкой файловой системы. Он имеет размер 1024 байта и всегда располагается по смещению 1024 байта от начала файловой системы. Наличие нескольких копий суперблока объясняется чрезвычайной важностью этого элемента файловой системы. Дубликаты суперблока используются при восстановлении файловой системы после сбоев.

Информация, хранимая в суперблоке, используется для организации доступа к остальным данным на диске. В суперблоке определяется размер файловой системы, максимальное число файлов в разделе, объем свободного пространства и содержится информация о том, где искать незанятые участки. При запуске ОС суперблок считывается в память, и все изменения файловой системы вначале находят отображение в копии суперблока, находящейся в ОП и записываются на диск только периодически. Это позволяет повысить производительность системы, т. к. многие пользователи и процессы постоянно обновляют файлы. С другой стороны, при выключении системы суперблок обязательно должен быть записан на диск, что не позволяет вы-

ключать компьютер простым отключением питания. В противном случае, при следующей загрузке информация, записанная в суперблоке, окажется не соответствующей реальному состоянию файловой системы.

Структура суперблока приведена в табл. 16.2.

Таблица 16.2. *Структура суперблока*

Название поля	Тип	Комментарий
s_inodes_count	ULONG	Число индексных дескрипторов в файловой системе
s_blocks_count	ULONG	Число блоков в файловой системе
s_r_blocks_count	ULONG	Число блоков, зарезервированных для суперпользователя
s_free_blocks_count	ULONG	Счетчик числа свободных блоков
s_free_inodes_count	ULONG	Счетчик числа свободных индексных дескрипторов
s_first_data_block	ULONG	Первый блок, который содержит данные. В зависимости от размера блока это поле может быть равно 0 или 1
s_log_block_size	ULONG	Индикатор размера логического блока: 0 = 1 Кбайт; 1 = 2 Кбайт; 2 = 4 Кбайт
s_log_frag_size	LONG	Индикатор размера фрагментов (кажется, понятие фрагмента в настоящее время не используется)
s_blocks_per_group	ULONG	Число блоков в каждой группе блоков
s_frags_per_group	ULONG	Число фрагментов в каждой группе блоков
s_inodes_per_group	ULONG	Число индексных дескрипторов (inodes) в каждой группе блоков
s_mtime	ULONG	Время, когда в последний раз была смонтирована файловая система
s_wtime	ULONG	Время, когда в последний раз производилась запись в файловую систему
s_mnt_count	USHORT	Счетчик числа монтирований файловой системы. Если этот счетчик достигает значения, указанного в следующем поле (s_max_mnt_count), файловая система должна быть проверена (это делается при перезапуске), а счетчик обнуляется
s_max_mnt_count	SHORT	Число, определяющее, сколько раз может быть смонтирована файловая система

Таблица 16.2 (окончание)

Название поля	Тип	Комментарий
s_magic	USHORT	"Магическое число" (OxEF53), указывающее, что файловая система принадлежит к типу ex2fs
s_state	USHORT	Флаги, указывающее текущее состояние файловой системы (является ли она чистой (clean) и т. п.)
s_errors	USHORT	Флаги, задающие процедуры обработки сообщений об ошибках (что делать, если найдены ошибки)
s_pad	USHORT	Заполнение
s_lastcheck	ULONG	Время последней проверки файловой системы
s_checkinterval	ULONG	Максимальный период времени между проверками файловой системы
s_creator_os	ULONG	Указание на тип ОС, в которой создана файловая система
s_rev_level	ULONG	Версия (revision level) файловой системы
s_reserved	ULONG[235]	Заполнение до 1024 байт

Вслед за суперблоком расположено описание группы блоков (Group Descriptors). Это описание представляет собой массив, имеющий структуру, приведенную в табл. 16.3.

Таблица 16.3. *Структура описания группы блоков*

Название поля	Тип	Назначение
bg_block_bitmap	ULONG	Адрес блока, содержащего битовую карту блоков (block bitmap) данной группы
bg_inode_bitmap	ULONG	Адрес блока, содержащего битовую карту индексных дескрипторов (inode bitmap) данной группы
bg_inode_table	ULONG	Адрес блока, содержащего таблицу индексных дескрипторов (inode table) данной группы
bg_free_blocks_count	USHORT	Счетчик числа свободных блоков в данной группе
bg_free_inodes_count	USHORT	Число свободных индексных дескрипторов в данной группе

Таблица 16.3 (окончание)

Название поля	Тип	Назначение
bg_used_dirs_count	USHORT	Число индексных дескрипторов в данной группе, которые являются каталогами
bg_pad	USHORT	Заполнение
bg_reserved	ULONG[3]	Заполнение

Размер описания группы блоков можно вычислить как

(*размер_группы_блоков_в_ext2x число_групп*) / *размер_блока*,

при необходимости округляем.

Информация, которая хранится в описании группы, используется для того, чтобы найти битовые карты блоков и индексных дескрипторов, а также таблицу индексных дескрипторов. Не забывайте, что блоки и группы блоков нумеруются, начиная с 1.

Битовая карта блоков (block bitmap) — это структура, каждый бит которой показывает, отведен ли соответствующий ему блок какому-либо файлу. Если бит равен 1, то блок занят. Эта карта служит для поиска свободных блоков в тех случаях, когда надо выделить место под файл. Битовая карта блоков занимает число блоков, равное

(*число_блоков_в_группе*/ 8) / *размер_блока*

(при необходимости округляем).

Битовая карта индексных дескрипторов выполняет аналогичную функцию по отношению к таблице индексных дескрипторов: показывает, какие именно дескрипторы заняты.

Следующая область в структуре группы блоков служит для хранения таблицы индексных дескрипторов файлов. Структура самого индексного дескриптора подробнее рассматривается ниже в *разд. 16.3*.

И, наконец, все оставшееся место в группе блоков отводится для хранения собственно файлов.

16.3. Индексные дескрипторы файлов

Каждому файлу на диске соответствует один и только один индексный дескриптор файла, который идентифицируется своим порядковым номером - индексом файла. Это означает, что число файлов, которые могут быть созданы в файловой системе, ограничено числом индексных дескрипторов, которое либо явно задается при создании файловой системы, либо вычисляется исходя из физического объема дискового раздела.

Строение индексного дескриптора файла приведено в табл. 16.4.

Таблица 16.4. Структура индексного дескриптора

Название поля	Тип	Описание
i_mode	USHORT	Тип и права доступа к данному файлу
i_uid	USHORT	Идентификатор владельца файла (Owner Uid)
i_size	ULONG	Размер файла в байтах
i_atime	ULONG	Время последнего обращения к файлу (Access time)
i_ctime	ULONG	Время создания файла
i_mtime	ULONG	Время последней модификации файла
i_dtime	ULONG	Время удаления файла
i_gid	USHORT	Идентификатор группы (GID)
i_links_ count	USHORT	Счетчик числа связей (Links count)
i_blocks	ULONG	Число блоков, занимаемых файлом
i_flags	ULONG	Флаги файла (File flags)
i_reservedl	ULONG	Зарезервировано для ОС
i_block	ULONG[15]	Указатели на блоки, в которых записаны данные файла (это поле подробно описано в *разд. 16.4)*
i_version	ULONG	Версия файла (для NFS)
i_file_acl	ULONG	ACL файла
idiracl	ULONG	ACL каталога
i_faddr	ULONG	Адрес фрагмента (Fragment address)
i_frag	UCHAR	Номер фрагмента (Fragment number)
i_fsize	UCHAR	Размер фрагмента (Fragment size)
i_pad1	USHORT	Заполнение
i_reserved2	ULONG[2]	Зарезервировано

Поле типа и прав доступа к файлу представляет собой двухбайтовое слово, каждый бит которого служит флагом, индицирующим отношение файла к определенному типу или установку одного конкретного права на файл (представлено в табл. 16.5).

Таблица 16.5. *Структура поля, задающего тип и права доступа*

Идентификатор	Значение	Назначение флага (поля)
S_IFMT	FOOO	Маска для типа файла
S_IFSOCK	AOOO	Доменное гнездо (socket)

Таблица 16.5 (окончание)

Идентификатор	Значение	Назначение флага (поля)
S_IFLNK	C000	Символическая ссылка
S_IFREG	8000	Обычный (regular) файл
S_IFBLK	6000	Блок-ориентированное устройство
S_IFDIR	4000	Каталог
S_IFCHR	2000	Байт-ориентированное (символьное) устройство
S_IFIFO	1000	Именованный канал (fifo)
S_ISUID	0800	SUID — бит смены владельца
S_ISGID	0400	SGID — бит смены группы
S_ISVTX	0200	Бит сохранения задачи (sticky bit)
S_IRWXU	01C0	Маска прав владельца файла
S_IRUSR	0100	Право на чтение
S_IWUSR	0080	Право на запись
S_IXUSR	0040	Право на выполнение
S_IRWXG	0038	Маска прав группы
S_IRGRP	0020	Право на чтение
S_IWGRP	0010	Право на запись
S_IXGRP	0008	Право на выполнение
S_IRWXO	0007	Маска прав остальных пользователей
S_IROTH	0004	Право на чтение
S_IWOTH	0002	Право на запись
S_IXOTH	0001	Право на выполнение

Среди индексных дескрипторов имеется несколько дескрипторов, которые зарезервированы для специальных целей и играют особую роль в файловой системе (табл. 16.6).

***Таблица 16.6.** Особые индексные дескрипторы*

Идентификатор	Значение	Описание
EXT2_BAD_INO	1	Индексный дескриптор, в котором перечислены адреса дефектных блоков на диске (Bad blocks inode)

Таблица 16.6 (окончание)

Идентификатор	Значение	Описание
EXT2_ROOT_INO	2	Индексный дескриптор корневого каталога файловой системы (Root inode)
EXT2_ACL_IDX_INO	3	ACL inode
EXT2_ACL_DATA_INO	4	ACL inode
EXT2_BOOT_LOADER_INO	5	Индексный дескриптор загрузчика (Boot loader inode)
EXT2_UNDEL_DIR_INO	6	Индексный дескриптор каталога для удаленных файлов (Undelete directory inode)
EXT2_FIRST_INO	11	Первый незарезервированный индексный дескриптор

Самый важный дескриптор в этом списке — дескриптор корневого каталога. Этот дескриптор указывает на корневой каталог, который, подобно всем каталогам, представляет собой связанный список, состоящий из записей переменной длины. Каждая запись имеет структуру, изображенную в табл. 16.7.

Таблица 16.7. *Структура дескриптора, описывающего корневой каталог*

Название поля	Тип	Описание
Inode	ULONG	Номер индексного дескриптора (индекс) файла
Rec_len	USHORT	Длина этой записи
Name_ len	USHORT	Длина имени файла
Name	CHAR[0]	Имя файла

Использование записей переменной длины позволяет применять длинные имена файлов без пустой траты дискового пространства. Отдельная запись в каталоге не может пересекать границу блока (то есть должна быть расположена целиком внутри одного блока). Поэтому, если очередная запись не помещается целиком в данном блоке, она переносится в следующий блок, а предыдущая запись продолжается таким образом, чтобы она заполнила блок до конца.

16.4. Система адресации данных

Система адресации данных — это одна из самых существенных составных частей файловой системы. Именно система адресации позволяет находить нужный файл среди множества как пустых, так и занятых блоков на диске.

В ext2fs система адресации реализуется полем i_block индексного дескриптора файла.

Поле `i_block` в индексном дескрипторе файла представляет собой массив *из* 15 адресов блоков. Первые 12 адресов в этом массиве (`EXT2_NDIR_BLOCKS [12]`) представляют собой прямые ссылки (адреса) на номера блоков, в которых хранятся данные из файла. Следующий адрес в этом массиве (`EXT2_IND_BLOCK`) является косвенной ссылкой, т. е. адресом блока, в котором хранится список адресов следующих блоков с данными из этого файла. В этом блоке могут быть записаны адреса

(размер _блока / размер_ULONG)

блоков с данными файла.

Следующий адрес в поле i_block индексного дескриптора (`EXT2_DIND_BLOCK`) указывает на блок двойной косвенной адресации (double indirect block). Этот блок содержит список адресов блоков, которые, в свою очередь, содержат списки адресов следующих блоков данных того файла, который задается данным индексным дескриптором.

И, наконец, последний адрес (`EXT2_TIND_BLOCK`) в поле i_block индексного дескриптора задает адрес блока тройной косвенной адресации, т. е. блока со списком адресов блоков, которые являются блоками двойной косвенной адресации.

Теперь вы знаете, как устроены индексные дескрипторы файлов, т. е. можете представить, как в файловой системе ext2fs осуществляется запись в файл и чтение из файла.

Может быть здесь еще надо бы рассказать о команде mkfs, которая служит для создания файловой системы в разделе диска. Но вкратце мы ее рассмотрели в *разд. 4.12,* а за более подробными пояснениями читатель может обратиться к интерактивным руководствам.

16.5. Виртуальная файловая система VFS

До сих пор наш рассказ о файловой системе касался только "статических", если можно так выразиться, составных частей файловой системы. Но, я думаю, вы понимаете, что все это хозяйство обслуживается какими-то программными модулями. Эти программные части можно разделить на две составных части. Одна часть входит в состав ядра и образует так называемую виртуальную файловую систему (VFS). VFS обеспечивает унифицированный программный интерфейс к услугам файловой системы, причем безотносительно к тому, какой тип файловой системы (vfat, ext2fs, nfs и т. д.) имеется на конкретном физическом носителе. Поэтому каждая файловая система должна предоставлять еще какие-то конкретные процедуры доступа к своим

файлам, для того, чтобы использоваться под Linux. Виртуальная файловая система VFS, расположенная как бы между приложениями и конкретными файловыми системами, позволяет пользовательским приложениям получать доступ к множеству файловых систем разных типов.

16.6. Новые файловые системы

Файловая система ext2fs была создана по образу и подобию файловой системы UNIX (UNIX File System — UFS). Обе они (особенно UFS) создавались еще в те времена, когда диски и другие физические носители данных имели довольно маленький (по современным меркам) объем. Увеличение объема дисков вело к возрастанию объема разделов диска, увеличению размеров отдельных файлов и каталогов. Это породило ряд проблем, связанных с ограниченностью внутренних структур данных файловой системы.

Существуют две основных проблемы этого рода.

- Во-первых, эти структуры не способны работать с носителями информации увеличенного объема. В них отведено строго фиксированное число битов для хранения данных о размере дисковых разделов и размерах файлов, фиксированное число битов для хранения логических номеров блоков и т. д. Как следствие, число файлов и каталогов и их размер ограничены.
- Вторая проблема связана с производительностью. В силу заложенных в старые файловые системы алгоритмов решение некоторых задач стало требовать слишком большого времени на носителях увеличенного объема. Одним из самых характерных примеров такого рода проблем является трудоемкость восстановления файловой системы после сбоев (например, после неожиданного отключения питания). Это восстановление выполняется с помощью программы fsck и для очень больших дисков стало требовать нескольких часов.

Естественно, что появление этих проблем породило и попытки их решения. Были разработаны новые типы файловых систем, при создании которых учитывались требования масштабируемости. Наиболее известными разработками файловых систем новых типов являются:

- файловая система ext3fs;
- XFS;
- журналируемая файловая система JFS фирмы IBM;
- ReiserFS.

В табл. 16.8 приведены данные по увеличению основных параметров, обеспечиваемых новыми файловыми системами. Данные заимствованы из статьи Juan *I.* Santos Florido "Journal File Systems", опубликованной в 55-ом выпуске Linux Gazette (July 2000).

Таблица 16.8. *Некоторые параметры файловых систем новых типов*

Файловая система	Размер блока	Максим. размер файловой системы	Максим. размер файла
Ext3FS	1-4 Кбайт	4 Тбайт	2 Гбайт
XFS	от 512 байт до 64 Кбайт	18 тысяч Пбайт	9 тысяч Пбайт
JFS	512, 1024, 2048, 4096 байт	от 4 Пбайт (при 512-байтных блоках) до 32 Пбайт (при 4-килобайтовых блоках)	От 512 Тбайт (при 512-байтовых блоках) До 4 Пбайт (при 4-килобайтовых блоках)
ReiserFS	До 64 Кбайт (пока что фиксирован, 4 Кбайт)	4 Гбайт блоков, 16 Тбайт	4 Гбайт, 2^{10} Пбайт в ReiserFS (3.6.*xx*)

16.7. Журналируемые файловые системы

Основная цель, которая преследуется при создании журналируемых файловых систем, состоит в том, чтобы обеспечить быстрое восстановление системы после сбоев (например, после потери питания). Дело в том, что если произойдет такой сбой, то часть информации о расположении файлов теряется, поскольку не все изменения сразу записываются на диск. После этого программа fsck вынуждена просматривать весь диск блок за блоком (пользуясь битовыми матрицами занятых блоков и индексных дескрипторов) с целью восстановления потерянных связей. При увеличении размера дисков вдвое вдвое увеличивается и время, которое требуется для просмотра всего диска. А при тех объемах, которых достигают современные диски, особенно на серверах, время, необходимое для того, чтобы просмотреть весь диск, стало недопустимо велико: ведь сервер в это время не отзывается! Кроме того, нет гарантии, что все связи удастся восстановить.

В журналируемых файловых системах для решения этой проблемы применяют технику транзакций, развитую в теории баз данных. Суть этой техники в том, что действие не считается завершенным, пока все изменения не сохранены на диске. А чтобы сбои, происходящие в течение времени, необходимого для завершения всех операций, не приводили к необратимым последствиям, все действия и все изменяемые данные протоколируются. Если сбой все-таки произойдет, то по этому протоколу можно вернуть систему в безошибочное состояние.

Главное отличие в технике транзакций, применяемой в базах данных, от аналогичной техники, применяемой в журналируемых файловых системах,

состоит в том, что в базах данных сохраняются в протоколе как сами изменяемые данные, так и вся управляющая информация, в то время как понятие транзакции в файловых системах подразумевает сохранение только метаданных: индексных дескрипторов изменяемого файла, битовых карт распределения свободных блоков и свободных индексных дескрипторов. Дело в том, что если сохранять все изменяемые данные, то теряется смысл кэширования записи на диск и уменьшается скорость дисковых операций. Метаданные же, во-первых, меньше по размеру, а, во-вторых, сохраняются в специально выделенной области диска, что позволяет избежать чрезмерных затрат времени на ведение протокола.

Файловые системы ext3fs и JFS являются журналируемыми. Надо отметить, что ext3fs нельзя назвать совершенно новой разработкой, это просто надстройка над ext2fs, обеспечивающая ведение журнала и организацию транзакций. Файловые системы XFS и JFS являются открытыми версиями коммерческих файловых систем.

16.8. Файловая система ReiserFS

Кроме проблемы быстрого восстановления после сбоев, в файловой системе ext2fs имеется еще несколько нерешенных проблем.

Одна из самых насущных — это проблема нерационального использования дискового пространства. Конечно, ext2fs использует диск гораздо более рационально, чем FAT, но, как вам хорошо известно, "памяти много не бывает"!

Собственно проблема возникает из-за следующего противоречия:

- если размер блока выбрать большим (кластер по 32 Кбайт в FAT), то при сохранении большого числа мелких файлов на диске неразумно используется дисковое пространство, т. к. маленькие файлы (и концы больших файлов) занимают целые блоки (Juan I. Santos Florido в своей статье называет это "внутренней фрагментацией");
- если размер блока выбрать маленьким (512 байт), то снижается производительность ввода/вывода, т. к. надо прочитать много блоков, которые могут быть разбросаны по диску (это "внешняя фрагментация").

Еще две проблемы, с которыми мы сталкиваемся в файловой системе ext2fs, связаны с поиском. Первая проблема возникает при записи на диск нового файла. Поскольку распределение свободных блоков хранится в виде битовой карты свободных блоков и свободных индексных дескрипторов, то файловая система вынуждена производить последовательный просмотр этих массивов для нахождения свободного места. В худшем случае это может потребовать времени, пропорционального объему диска.

Вторая проблема поиска связана с поиском файлов в больших каталогах. Поскольку файлы мы ищем по именам, приходится последовательно про-

сматривать все записи в каталоге. Время такого поиска тоже пропорционально размеру каталога и вырастает в проблему при больших размерах каталогов.

Между тем методы снижения трудоемкости поиска давно разработаны, только надо для хранения информации о свободных объектах использовать не простые списки, а несколько более сложные структуры данных. В системе ReiserFS для этого применяются так называемые "сбалансированные деревья" или "B+Trees", время поиска в которых пропорционально не количеству объектов (файлов в каталоге или числа блоков на диске), а логарифму этого числа. В сбалансированном дереве все ветви (пути от корня до "листа") имеют одинаковую (или примерно одинаковую) длину. ReiserFS использует сбалансированные деревья для хранения всех объектов файловой системы: файлов в каталогах, данных о свободных блоках и т. д. Это позволяет существенно повысить производительность обращения к дискам.

Кроме того, ReiserFS является журналируемой, т. е. в ней решена и проблема быстрого восстановления после сбоев.

Я привел в этой главе только самые поверхностные данные о новых типах файловых систем, поскольку статей о них пока опубликовано довольно мало. Тем не менее, за время подготовки книги некоторая дополнительная информация появилась, так что если вас этот вопрос интересует, поищите сведения в Интернете.

Глава 17

Обновление ядра

17.1. Что такое ядро и когда его надо менять

Каждый, кто хоть немного интересовался тем, что такое Linux, обязательно встречал в различных руководствах термин "ядро", по-английски — *kernel.* Ядро — это важнейшая часть Linux, как и любой другой операционной системы, поскольку именно ядро обеспечивает взаимодействие с аппаратной частью компьютера, распределение ресурсов, управление процессами и многое другое. Когда вы загружаете какое-то приложение с жесткого диска в оперативную память, или переключаетесь между уже работающими приложениями, или когда какое-то приложение записывает информацию в файл на диске, операционная система или активное приложение должно запросить доступ к той части аппаратуры, которая ему необходима. Ядро обеспечивает исполнение таких запросов других частей операционной системы и приложений, а также распределяет память между запускаемыми приложениями. Ядро, таким образом, является посредником между аппаратным и программным обеспечением компьютера, обеспечивающим их взаимодействие.

Работа по совершенствованию ядра Linux ведется международным сообществом разработчиков постоянно, и регулярно появляются новые версии ядра. Естественно, что пользователи хотят иметь последнюю (или, по крайней мере, одну из последних) версий ядра ОС и рано или поздно вы приходите к выводу о том, что пора обновить ядро.

Можно задать вопрос: "В каких случаях это необходимо?". Действительно, если система неплохо работает со старым ядром, то стоит ли заниматься его обновлением? Основными причинами, приводящими к выводу о необходимости обновления ядра, являются:

- обновление аппаратуры компьютера, подключение новых устройств, которые не поддерживаются старым ядром;
- необходимость работы с новыми программами, которые рассчитаны на новую версию ядра и отказываются работать с версией, установленной у вас;
- обнаружение каких-то ошибок в старой версии ядра, в частности таких, которые представляют угрозы с точки зрения безопасности;

- желание повысить производительность системы, используя более совершенную версию ядра, либо оптимизировать ядро для работы с конкретным набором аппаратных средств, имеющихся на вашем компьютере;
- простое любопытство и желание работать с последней версией системы.

Обновить ядро можно двумя способами: установкой готового бинарного образа нового ядра из RPM-пакета и компиляцией ядра из исходных текстов. Первый способ проще, но надо иметь в виду, что скомпилированное где-то и кем-то ядро скорее всего не является оптимальным вариантом для вашей системы. Поэтому приходится применять второй способ — компиляцию ядра из исходных кодов. Для начинающих пользователей Linux компиляция ядра из исходных кодов кажется чем-то суперсложным и недоступным. Однако я думаю, что, прочитав настоящую главу, вы убедитесь, что это не намного сложнее, чем установка ПО из RPM-пакета.

17.2. Нумерация версий ядра

Прежде чем браться за обновление ядра, вы должны четко представлять себе, что за версию вы собираетесь установить. В первую очередь необходимо иметь в виду, что разработчики ядра поддерживают две ветки ядра: стабильную и экспериментальную. Все новшества, вносимые в ядро, вначале появляются в экспериментальных версиях. И только после того, как сообщество разработчиков и добровольных тестировщиков опробует эти новшества, они переносятся в так называемую стабильную версию.

Версии ядра принято нумеровать тремя цифрами, разделенными точками, например, 2.4.8, при этом четная вторая цифра в номере ядра обозначает стабильные версии ядра, а нечетная — экспериментальные версии. Так что, принимая решение об установке новой версии ядра, вы должны продумать ответ на вопрос, хотите ли вы участвовать в выявлении возможных ошибок в нестабильной версии или предпочитаете работать с уже оттестированным ядром.

Как заявил Линус Торвальдс в одном из своих интервью, он предпочитает как раз заниматься экспериментальной веткой, разрабатывать код, работающий с новыми устройствами. Основным координатором разработки стабильной ветки является в настоящее время Алан Кокс, регулярно выпускающий обновленные версии или "заплатки" к стабильным версиям ядра.[1]

[1] Во время подготовки книги к изданию появилось сообщение, что Алан Кокс передал функции координации работ по стабильной ветке ядра Марчело Тосатти (Marcello Tosatti).

17.3. Установка нового ядра из RPM-пакета

Честно сказать, я довольно долгое время не решался браться за обновление ядра, поскольку первая из предпринятых мною попыток оказалась неудачной, причем до того неудачной, что мне пришлось полностью переустановить систему. Я тогда пытался установить ядро из исходных текстов. Но однажды я наткнулся в новостях на сообщение о том, что выпушен RPM-пакет с ядром 2.2.16-1. Поскольку мой опыт работы с RPM-пакетами был вполне положительным, я решился попытаться еще раз, и попытка эта оказалась вполне успешной!

Итак, вначале рассмотрим установку нового ядра, откомпилированного кем-то и представленного в виде RPM-пакета. Естественно, что первым делом надо скачать RPM-пакет с новым ядром. Если вы не ставите своей целью тестирование новшеств в ядре, то ищите RPM-пакет со стабильной версией, т. е. с четной второй цифрой в номере версии ядра (номер версии указывается в названии пакета). Я скачал ядро версии 2.2.16-1 с сервера **http://rufus.w3.org/linux/RPM/**.

Скачав ядро, запустите команду

```
[root]# rpm -i kernel-2.2.16-1.i386.rpm
```

По этой команде программа rpm установит в каталог /boot четыре файла: System.map-x.y.z-a, vmlinux-x.y.z-a, vmlinuz-x.y.z-a и module-info-x.y.z-a (где x.y.z-a — это номер версии нового ядра), создаст каталог /lib/modules/x.y.z-a, в котором разместит модули нового ядра, а также установит скрипт /sbin/installkernel.

Если у вас есть такое желание, вы можете предварительно (или потом) выполнить команду

```
[root]# rpm -qpl kernel-2.2.16-1.i386.rpm,
```

которая выдаст полный список устанавливаемых из пакета файлов. Выполнение этих команд еще ничего не меняет в системе, только подготавливает файлы для запуска нового ядра.

Я после установки пакета с ядром решился сразу запустить скрипт /sbin/installkernel. Скрипт завис, его пришлось снимать комбинацией клавиш <Ctrl>+<C>. Тогда я стал разбираться с тем, что делает этот скрипт. Вот его полный текст:

```
#-------------------начало скрипта-------------------
#! /bin/sh
# /sbin/installkernel — written by tyson@rwii.com
#
INSTALL_PATH=/boot
```

```
KERNEL_VERSION=$1
BOOTIMAGE=$2
MAPFILE=$3
if [ -f $INSTALL_PATH/vmlinuz-$KERNEL_VERSION ]; then
mv $INSTALL_PATH/vmlinuz-$KERNEL_VERSION \
$INSTALL_PATH/vmlinuz.old;
fi
if [ -f $INSTALL_PATH/System.map-$KERNEL_VERSION ]; then
mv $INSTALL_PATH/System.map-$KERNEL_VERSION \
$INSTALL_PATH/System.map.old;
fi
cat $BOOTIMAGE > $INSTALL_PATH/vmlinuz-$KERNEL_VERSION
cp $MAPFILE $INSTALL_PATH/System.map-$KERNEL_VERSION
ln -fs vmlinuz-$KERNEL_VERSION $INSTALL_PATH/vmlinuz
ln -fs System.map-$KERNEL_VERSION $INSTALL_PATH/System.map
if [ -x /sbin/lilo ]; then /sbin/lilo; else /etc/lilo/install; fi
#-------------------конец скрипта-------------------
```

Как видите, скрипту требуются некоторые параметры, а я запускал его без аргументов. Однако, поскольку программа rpm уже разместила в /boot файлы vmlinux-2.2.16-1, vmlinuz-2.2.16-1 и System.map-2.2.16-1, а также создала ссылки с именами vmlinuz и System.map, я решил, что осталось только перезапустить lilo. Но прежде чем запускать lilo, я подредактировал файл /etc/lilo.conf, добавив туда секцию с указанием на ядро 2.2.16 (просто скопировал секцию с меткой linux, после чего подправил строки image и label). Причем новую секцию поставил первой. Вот что у меня получилось (добавленные мной строки выделены жирным шрифтом):

```
#-------------------начало файла /etc/lilo.conf ----------
boot=/dev/hdb1
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz-2.2.16-1
  label=linux-2.2.16
  root=/dev/hdb1
  read-only
image=/boot/vmlinuz-2.2.11-4bc
  label=linux
  root=/dev/hdb1
```

```
    read-only
other=/dev/hda1
    label=dos
    table=/dev/hda
#-------------------конец файла /etc/lilo.conf -------------------
```

Обратите внимание на то, что в данном случае Linux грузится со второго диска, поскольку на первом диске стоит Windows NT, и в качестве загрузчика используется NT Loader.

После этого я запустил команду `/sbin/lilo`, причем вначале с параметрами -t -v, чтобы посмотреть, что она будет делать, а только потом уже — без параметров, для реального исполнения. Когда команда отработала, я перезагрузился, однако на этапе загрузки система зависла, выведя строку `LIL-`. Пришлось воспользоваться загрузочной дискетой (заготовленной еще при установке системы) и выполнить команду

```
[root]# dd if=/dev/hdb1 of=/mnt/hda1/bootsect.lnx bs=512 count=1
```

Все правильно, команда `lilo` меняет загрузочный сектор, а я забыл "подсунуть" загрузчику его обновленный вариант.

Еще раз перезагружаюсь и с радостью вижу сообщение о том, что загрузилось ядро 2.2.16-1.

Если у вас LILO является основным загрузчиком (установлен в MBR первого диска), то последнюю операцию (копирование загрузочного сектора в файл) выполнять, конечно, не нужно.

17.4. О компиляции нового ядра

17.4.1 Зачем вообще нужно компилировать ядро?

Как было сказано в начале данной главы, основная функция ядра состоит в том, чтобы обеспечить взаимодействие с аппаратными средствами компьютера. Обслуживание некоторых составляющих аппаратного обеспечения (таких как память, например) напрямую встроено в ядро. Для тех частей аппаратуры, которые могут быть нестандартными, имеются драйверы устройств, которые обеспечивают взаимодействие ОС с аппаратурой. Большинство пользователей компьютеров с ОС Windows знакомы с понятием драйвера хотя бы потому, что после установки нового оборудования они вынуждены устанавливать и программные драйверы для этого оборудования. Только после этого становится возможным использовать вновь установленную аппаратную составляющую. В терминологии, принятой в Linux, драйверы называются "модулями". Таким образом, поддержка аппаратных устройств может быть обеспечена двумя способами: либо путем встраивания такой поддержки в ядро, либо путем использования соответствующего модуля (драйвера).

Компании, которые выпускают дистрибутивы Linux (такие как Red Hat, Caldera, Debian и т. д.), вынуждены встраивать в ядро поддержку как можно более широкого спектра устройств, потому что они не могут заранее знать, какие устройства (модели устройств) будут установлены на компьютере конкретного пользователя. Поддержка в ядре широкого спектра устройств облегчает установку и поддержку системы для покупателей, избавляя их от ненужных сложностей.

Как результат, ядро, поставляемое в составе дистрибутива, скорее всего содержит код для поддержки устройств, которых никогда не будет на вашем конкретном компьютере. С другой стороны, если у вас есть устройство, которое не поддерживается стандартным ядром, у вас может появиться обоснованное желание встроить поддержку этого устройства в ядро. Оптимизация ядра под конкретный набор аппаратных устройств ускоряет загрузку системы и экономит память.

Пользователи Linux имеют возможность или скомпилировать ядро с поддержкой всех устройств, имеющихся на конкретном компьютере, или скомпилировать ядро, поддерживающее минимальный набор оборудования, и загружать модули для поддержки остальных устройств. Если поддержка всего оборудования осуществляется в ядре, то такое ядро называется "монолитным". Ядро, скомпилированное таким образом, что поддержка части оборудования осуществляется с использованием модулей (драйверов), называется "модульным".

Какой тип ядра вам выбрать при компиляции? Однозначного ответа на этот вопрос дать нельзя. Если вы не имеете привычки менять аппаратную конфигурацию компьютера, тогда вам лучше встроить поддержку всех имеющихся компонентов в ядро. Необходимо только иметь в виду, что чем больше устройств поддерживаются непосредственно ядром, тем больше его объем. А поскольку ядро полностью загружается в оперативную память, повышаются требования к объему памяти. На медленных компьютерах из-за большого размера ядра может снизиться общая производительность. Если же вы часто меняете конфигурацию компьютера (например, у вас имеются съемные жесткие диски или другие временно подключаемые устройства), то, вероятно, имеет смысл использовать для управления ими подключаемые модули, которые загружаются в память только при необходимости (экономя тем самым системные ресурсы). Таким образом, в самом общем случае поддержка некоторой части устройств должна быть встроена в ядро, а остальные устройства должны поддерживаться за счет использования загружаемых модулей.

Кроме желания иметь ядро, оптимизированное для вашей системы, необходимость перекомпилировать ядро может быть вызвана обнаружением каких-то ошибок в старой версии ядра, в частности таких, которые представляют угрозы с точки зрения безопасности (когда еще появится RPM-пакет с исправленной версией ядра?).

Я был вынужден заниматься установкой ядра из исходных кодов потому, что система виртуальных машин VMware отказалась работать с установленным у меня ядром 2.2.16, сообщив, что эта версия ядра не поддерживает работу с CD-ROM из VMware, и предложив мне либо установить более позднюю версию ядра, либо вернуться к версии 2.2.15. Попытки установить новую версию ядра из RPM-пакетов тоже не решили проблему, потому что конфигурационный скрипт VMware сообщал, что ему не хватает header-файлов. Установка пакетов kernel-headers (полностью соответствующих ядру) тоже не привела к успеху, вот и пришлось сделать попытку установить ядро из исходных текстов.

Надо сказать, что к тому времени мой опыт установки программного обеспечения для Linux из исходников был очень ограничен. Поэтому приступал я к этой процедуре только под давлением обстоятельств (очень хотелось запускать MS Office под Linux, не прибегая к перезагрузке компьютера). Приводимый ниже текст является как раз описанием того, что я тогда делал. Поскольку мой эксперимент оказался удачным, я могу со спокойной совестью утверждать: ничего такого, что оказалось бы не под силу начинающему пользователю, в компиляции ядра из исходных кодов нет.

Я экспериментировал на версии 2.2.16-22 из свежеустановленного дистрибутива ASPLinux Release Candidate 3 и устанавливал ядро версии 2.4.2. Поэтому все примеры в данной главе приводятся для случая, когда система уже работает на ядре версии 2.2.*x* и вы пришли к решению установить ядро версии *2A.x*.

17.4.2. Что надо знать до начала компиляции

Пожалуй, самое первое, к чему нужно быть готовым, приступая к компиляции ядра, — это то, что процедура эта длительная. Так что не рассчитывайте скомпилировать ядро "между делом", в свободную минутку. Заранее планируйте, что потратите на это несколько часов, иначе вы будете вынуждены прервать процедуру посередине.

Во-вторых, до начала компиляции ядра вам необходимо хотя бы в общих чертах представлять, какую именно конфигурацию аппаратного обеспечения будет обслуживать новое ядро, и решить, какие устройства будут обслуживаться самим ядром, а какие — модулями.

В-третьих, ядро версии 2.4.2 даже в архивированном (программой bzip2) виде занимает более 20 Мбайт, а после разархивации объем исходных текстов превышает 108 Мбайт. Еще столько же может потребоваться для промежуточного tar-архива, так что надо иметь не менее 250 Мбайт свободного места. Поэтому прежде чем приступить к компиляции, позаботьтесь о том, чтобы места на диске было достаточно.

В-четвертых, приготовьтесь отвечать на те вопросы, которые программа задаст вам в процессе конфигурирования ядра (а их около 500). Перевод всех этих вопросов, относящийся к версии *2.0.x* ядра, можно найти в файле Configure.help, который находится по адресу **http://nevod.perra.su/service/linux/doc/kernel/Configure.help**.

Этот файл можно использовать и для более поздних версий, поскольку в основном вопросы, задаваемые при конфигурации, одинаковы, только возникают некоторые дополнительные в связи с появлением в ядре новых возможностей. Лучше всего скачать этот файл, распечатать его и использовать этот перевод в процессе конфигурирования ядра *(см. разд 17.5.3)*. Если вы не смогли скачать этот файл, то после развертывания исходных текстов ядра (см. ниже) вы найдете файл Configure.help в подкаталоге linux/Documentation. Имеет смысл просмотреть его (а лучше — распечатать) до начала конфигурирования, даже если вы не очень хорошо владеете английским.

На этом предварительные предупреждения сделаны, и я перехожу к изложению пошаговых инструкций по компиляции и установке нового ядра. В дальнейшем предполагается, что все описываемые действия выполняются с правами суперпользователя root, и что домашним каталогом является каталог /root.

17.5. Семь шагов к новому ядру

17.5.1. Получение и разархивация ядра

Исходные тексты новой версии ядра можно скачать с сайта **ftp.kernel.org**. Как уже было сказано, bzip2-архив исходных кодов ядра версии 2.4.2 имеет объем более 20 Мбайт, так что скачать его — тоже еще проблема. Но перекачивать исходные тексты необходимо только в том случае, если вы желаете установить новую версию ядра. Если же вы просто хотите перекомпилировать существующее ядро (скажем, из-за того, что необходимо обеспечить поддержку какого-то протокола или нового оборудования), то можно обойтись и без перекачки пакета из Интернета, поскольку с дистрибутивами обычно поставляются и исходные коды (на втором диске или просто в подкаталоге SRPM). Кроме того, даже если вы захотели скомпилировать ядро новой версии, существует возможность сократить объем информации, которую необходимо скачивать из Интернета, но об этом мы поговорим в последнем разделе данной главы. А пока будем предполагать, что вы тем или иным способом получили полный пакет исходных кодов ядра.

Поместите архив в каталог, в котором вы имеете достаточные права, например, в ваш домашний каталог. Не используйте каталог /usr/src/linux для разархивации исходников! Этот каталог содержит (обычно неполный) набор

заголовочных файлов (kernel headers), которые используются заголовочными файлами библиотек (the library header files). Они должны соответствовать установленным в системе библиотекам, поэтому не стоит заранее вносить путаницу в эти файлы.

Распакуйте архив командой:

```
[root]# bzip2 -d linux-2.4.XX.tar.gz | tar xvf -
```

(где xx надо заменить на номер версии ядра, у меня была просто 2). Если вы скачали архив, сжатый программой gzip, то, естественно, команда будет иметь вид:

```
[root]# gunzip linux-2.4.XX.tar.gz | tar xvf -
```

Можно также воспользоваться следующим вариантом команды:

```
[root]# tar xvzf linux-2.4.2.tar.gz
```

В результате в текущем каталоге появится новый каталог linux. Сделайте его текущим с помощью команды cd.

17.5.2. Обновление программного обеспечения

Если вы владеете английским, то неплохо просмотреть файл README в каталоге linux и файлы Changes и Configure.help в подкаталоге linux/Documentation. Впрочем, тем кто не владеет английским, тоже необходимо заглянуть по крайней мере в файл linux/Documentation/Changes. Дело в том, что в этом файле приведен состав программного обеспечения, необходимого для компиляции нового ядра.

В приводимой ниже табл. 17.1 показан его состав для случая ядра 2.4.2.

Таблица 17.1. *ПО, необходимое для компиляции нового ядра*

Программа	Версия	Как определить версию
Gnu C	2.91.66	`gcc --version`
Gnu make	3.77	`make --version`
Binutils	2.9.1.0.25	`ld -v`
util-linux	2.10o	`fdformat --version`
Modutils	2.4.2	`/sbin/insmod -V`
e2fsprogs	1.19	`/sbin/tune2fs`
Reiserfsprogs	3.x.0b	`reiserfsck 2>&1|grep reiserfsprogs`
pcmcia-cs	3.1.21	`cardmgr -V`

Таблица 17.1 *(окончание)*

Программа	Версия	Как определить версию
PPP	2.4.0	`pppd --version`
isdn4k-utils	3.1pre1	`isdnctrl 2>&1|grep version`

Приведенные в правой колонке команды позволяют произвести проверку того, что необходимый пакет имеется и имеет соответствующую версию (более новые версии не возбраняются). Не все перечисленные в этой таблице пакеты безусловно необходимы для компиляции ядра: если в вашей системе нет PCMCIA-карт (PC Card), например, то вам не нужен и пакет pcmcia-cs. Я посчитал ненужными последние 4 пакета (reiserfs и pcmcia у меня нет, а удаленный дозвон и соединение по isdn я не использую), а для остальных пакетов нашел на сайте **rpmfind.net** последние версии и установил их. Все указанные пакеты установились из RPM-файлов без проблем.

Совет для "чайников" вроде меня: используйте команду `rpm -Uhv paket_name`, а не `rpm -i paket_name`.

17.5.3. Конфигурирование будущего ядра

Следующий этап заключается в конфигурировании будущего ядра. Если вы при установке обновленных версий ПО покинули каталог linux, то вернитесь в него (например, с помощью команды `cd ~/linux`).

Примечание

В некоторых случаях перед началом конфигурирования ядра следует выполнить команду make `mrproper`. Необходимо это бывает тогда, когда этот каталог уже использовался для компиляции ядра и надо удалить следы былых, может быть ошибочных, действий, в частности, ранее созданные файлы с расширением .o.

Собственно конфигурирование выполняется командой make config. Конфигурация в этом случае будет производиться в текстовом режиме. Процедура конфигурации будет заключаться в том, что вы должны будете последовательно ответить на серию вопросов о том, какое значение присвоить определенному параметру. На каждый вопрос предлагается обычно несколько вариантов ответа. Допустимые варианты ответа предлагаются в виде символов, заключенных в квадратные скобки. Я думаю, что смысл символов "y" и "n" пояснять не требуется. Пояснения требуют символы "?" и "t".

Символ "?" присутствует среди возможных вариантов ответа на любой вопрос и позволяет получить подсказку (конечно, по-английски). Эти подсказки и составляют содержание файла Configure.help, который упоминался

выше, и перевод которого (даже не в последней версии) я очень рекомендую вам иметь под рукой.

Если вы выбираете вариант "т", это означает, что драйвер соответствующего устройства будет сконфигурирован (и впоследствии скомпилирован) в виде отдельного подключаемого модуля.

Один из вариантов ответа на каждый вопрос представлен большой буквой, что означает, что он выбирается по умолчанию (когда вы не задаете явно свой вариант выбора, а просто нажимаете клавишу <Enter>).

В процессе конфигурации ядра следует иметь в виду следующее:

- включение в ядро драйвера любого устройства делает его больше и, более того, добавление ненужных драйверов в ядро может привести к проблемам, когда ядро будет пытаться обратиться к несуществующему устройству;
- если задать тип процессора ("Processor type") выше, чем 386, то ядро не будет работать на 386-х процессорах. Ядро обнаружит это при загрузке и откажется работать;
- ядро, скомпилированное с опцией эмуляции математического сопроцессора, все равно будет использовать сопроцессор, если он имеется. Функция эмуляции никогда не будет использоваться в этом случае. Правда, ядро в этом случае будет чуть больше, но зато будет работать на любых компьютерах, независимо от того, имеется ли на них сопроцессор;
- некоторые вопросы помечены указаниями "development", "experimental" или "debugging", которые указывают на то, что соответствующая функция или драйвер включены в ядро в виде эксперимента. Включение этих функций в ядро может сделать ядро не только больше, но и нанести ущерб стабильности его работы. Так что на такие вопросы лучше ответить "n", если, конечно, вы не ставите своей целью провести тестирование этих новых доработок в ядре;
- если вы компилируете ядро для использования на своем персональном компьютере, то поддержка мультипроцессорной обработки (Symmetric multi-processing support) вам, скорее всего, не нужна.

Впрочем, я начал уже давать пояснения по тому, как задавать значения отдельных параметров. Но их, во-первых, очень много, так что эта глава грозит распухнуть до неприличия, а, во-вторых, самое приемлемое решение определяется конкретной конфигурацией вашего компьютера и вашими личными потребностями и пожеланиями, так что обратитесь лучше к упоминавшемуся выше файлу Configure.help.

В заключение данного раздела замечу, что вместо make `config` можно использовать два альтернативных варианта команды конфигурации: make `menuconfig` И make `xconfig`. **Эти Команды отличаются ОТ** make `config` тем, что предоставляют пользователю возможность вместо ответа на вопросы

производить выбор вариантов из предлагаемых меню. Команда `make menuconfig` работает в текстовом режиме, а `make xconfig` — в графическом. На рис. 17.1 представлен вид окна, которое появляется при запуске команды `make xconfig`.

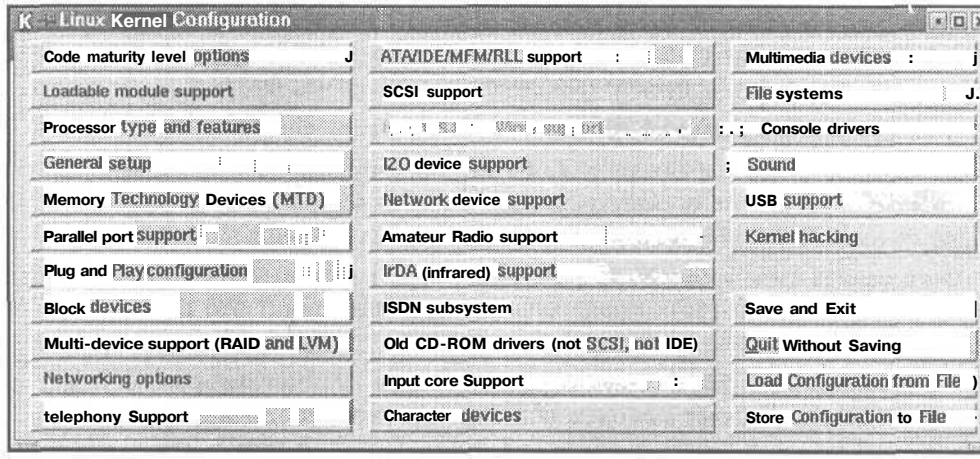


Рис. 17.1. Основное меню программы make xconfig

Кнопки, представленные в этом окне, отображают основные группы параметров конфигурации (кроме 4 кнопок в правом нижнем углу окна, которые служат для сохранения конфигурации или загрузки ранее сохраненных вариантов конфигурации). Щелчок по любой кнопке, соответствующей группе параметров, приводит к появлению нового окна, в котором уже можно задавать значения для конкретных параметров. Например, щелчок по кнопке **General setup** приводит к появлению окна, изображенного на рис. 17.2.

Как видите, здесь можно просто выбрать один из трех вариантов "y", "n" и "m" (если какой-то из вариантов недоступен, то и возможности его выбрать нет) или получить подсказку по отдельному параметру (кнопка **Help).** Поскольку число параметров в этой группе велико, справа имеется полоса прокрутки.

Впрочем, мне не кажется, что конфигурировать ядро, используя графический интерфейс, проще, чем с помощью команды `make config`. Так или иначе, надо продумать и дать ответ на вопрос о каждом параметре конфигурации. Так что, может быть, лучше уж не торопиться и последовательно отвечать на вопросы, задаваемые `make config`? А иначе (в графическом режиме, я имею в виду) возникает шанс сбиться с пути и пропустить какую-то из кнопок. С другой стороны, преимуществом графического режима является возможность вернуться к какому-то из уже пройденных этапов и изменить

ранее заданные значения тех или иных параметров. Кроме того, в каждом из окон имеется кнопка Next, пользуясь которой, можно последовательно пройти все этапы конфигурации.

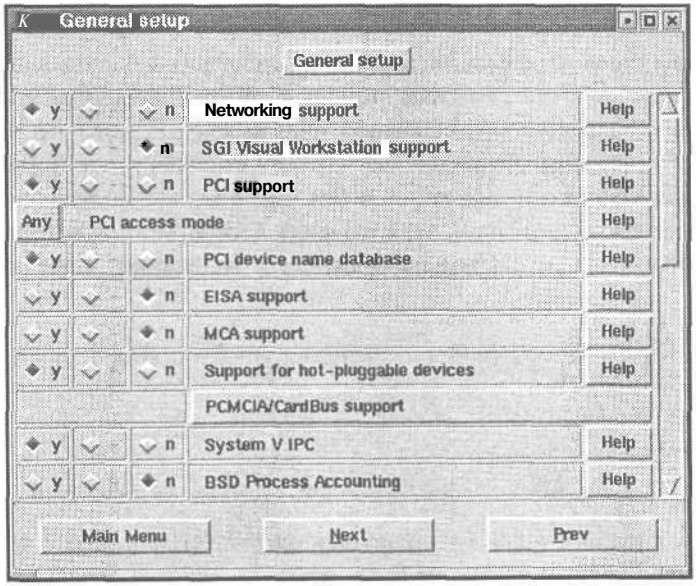


Рис. 17.2. Меню General setup

В общем, представление о вариантах осуществления шага конфигурации вы получили, а решать вам. Отмечу только, что воспользоваться командой make menuconfig вы сможете только при условии, что у вас в системе установлена библиотека ncurses и пакет ncurses-devel.

Я думаю, что сказанного достаточно для того, чтобы вы смогли сконфигурировать ядро, поэтому переходим к дальнейшим шагам.

17.5.4. Проверки

Документация, поставляемая вместе с ядром, советует после завершения конфигурации выполнить два действия:

1. Заглянуть в файл Makefile, чтобы вручную поправить некоторые значения.
2. Дать команду make dep для установки зависимостей.

Конечно, для ручной правки файла Makefile надо понимать, для чего и как этот файл используется. Если вы этого не знаете, то можно этого и не делать, положившись на опыт и интуицию разработчиков. Ну, а вторую реко-

мендацию я советую выполнить. По экрану побегут непонятные сообщения, причем будет их очень много. Но если в конце не появляется сообщения об ошибке, значит, все завершилось успешно.

17.5.5. Компиляция ядра

Вот мы и добрались до основного этапа — собственно компиляции ядра. На этом **Этапе ваше участие СВОДИТСЯ ТОЛЬКО К запуску** команды `make bzImage`, которая служит для создания сжатого образа ядра. Снова по экрану бегут непонятные сообщения (которые просто не успеваешь воспринять). О том, что процесс компиляции завершился без ошибок, вы можете судить по появлению сообщений примерно такого вида (естественно, я привожу те сообщения, которые были выданы в процессе компиляции ядра на моем компьютере):

```
Root device is (3,5)
Boot sector 512 bytes.
Setup is 2360 bytes.
System is 887 kB
make[1]: Выход из каталог '/opt/kernel/2.4.2/linux/arch/i386/boot'
```

И в текущем каталоге должны появиться файлы System.map и vmlinux. Кроме того, в подкаталогах каталога linux тоже появляется масса новых файлов (в том числе и o-файлы, которые упоминались выше, как мешающие повторному проведению компиляции ядра в том же каталоге).

На этом собственно компиляция ядра и заканчивается. В дополнение приведу еще три замечания из документации к ядру.

- Если в качестве загрузчика вы используете LILO, то после компиляции можно выполнить команду `make install`, однако рекомендуется вначале проверить (и подкорректировать) конфигурационный файл LILO.
- Хотя такое событие и очень маловероятно, но если ваша система не может загружать ядро в формате bzImage, вы можете скомпилировать его как zImage. Однако надо иметь в виду, что поддержка zImage будет в ближайшем будущем отменена, поэтому разработчики предлагают тем, у кого возникнут проблемы с загрузкой ядра в формате bzImage, сообщить об этом (приложив детальное описание конфигурации) в список рассылки linux-kernel и Питеру Анвину (H. Peter Anvin, **hpa+linux@zytor.com**).
- Если вы вдруг захотите создать загрузочный диск (без корневой файловой системы или LILO), вставьте дискету в дисковод A: и дайте команду `make bzdisk`.

17.5.6. Компиляция модулей

Если вы сконфигурировали какие-то драйверы как отдельные модули (выбирали при конфигурации вариант "т" при ответе на некоторые вопросы), то вы теперь должны еще выполнить команду make modules, а затем еще команду make `modules_install`. В файле Documentation/modules.txt можно найти дополнительную информацию по этому поводу, а также объяснение того, как использовать модули.

В завершение этапа компиляции стоит выполнить еще команду make `clean`, чтобы удалить образующиеся в процессе компиляции промежуточные объектные файлы (файлы с расширением .o).

17.5.7. Установка ядра

После этого остается сделать последний шаг — установить ядро и перезагрузиться. Для установки ядра вы должны иметь права суперпользователя. (Хотя в начале главы и было сказано, что для компиляции ядра надо иметь права суперпользователя, однако все предыдущие шаги можно было выполнить и от имени обычного пользователя. Не стоит работать от имени суперпользователя без необходимости!)

Разработчики рекомендуют вначале сохранить где-нибудь копию старого ядра на случай, если что-то пойдет не так, как задумано. Эта рекомендация особенно актуальна для случая, если вы ставите ядро из нестабильной ветки, поскольку такие версии могут содержать неотлаженный код. Кроме самого ядра надо сделать backup-копии модулей, соответствующих ядру. Если вы устанавливаете новое ядро с тем же самым номером версии, что и у работающего в вашей системе ядра, сделайте резервную копию всего каталога С МОДУЛЯМИ **перед Выполнением Команды** make `modules_install`.

Для того чтобы иметь возможность загружать новое ядро, копию образа ядра (которая после компиляции создана в виде файла .../linux/arch/i386/boot/bzImage) необходимо поместить туда, где у вас расположены загружаемые ядра (обычно это каталог /boot).

Скопируйте в каталог /boot три файла: файлы System.map и vmlinux, появившиеся в каталоге linux, и файл .../linux/arch/i386/boot/bzImage. Я при копировании добавил к их именам номер версии ядра, превратив их, соответственно, в System.map-2.4.2, vmlinux-2.4.2 и vmlinuz-2.4.2, чтобы не путать с теми ядрами, которые уже были в системе ранее. Переименовывать bzImage в vmlinuz в принципе необязательно, потому что образ ядра может иметь как то, так и другое имя, и обычно располагается либо в корневом каталоге (/), либо в каталоге /boot.

Далее нам осталось только обеспечить загрузку нового ядра с помощью то. Для этого надо вначале подкорректировать файл /etc/lilo.conf. Ваш файл /etc/lilo.conf может иметь примерно такой вид:

```
boot = /dev/hda2
compact
delay = 50
root = current
image = /boot/vmlinuz-2.2.11-4bc
 label = linux
 read-only
other = /dev/hda1
 table = /dev/hda
 label = dos
```

Начиная со строки image, идут секции конфигурационного файла, соответствующие разным операционным системам, которые должны загружаться по выбору пользователя. В каждой такой секции имеется строка label. В этой строке записывается имя, которое вводится в ответ на приглашение LILO и служит для выбора пользователем загружаемой ОС.

Скопируйте секцию image и замените в новой секции название образа ядра и метку. Файл /etc/lilo.conf примет следующий вид:

```
boot = /dev/hda2
compact
delay = 50
root = current
image = /boot/vmlinuz-2.2.11-4bc
 label = linux
 read-only
image = /boot/vmlinuz-2.4.2
 label = linux-2.4.2
 read-only
other = /dev/hda1
 table = /dev/hda
 label = dos
```

После того как вы откорректировали файл /etc/lilo.conf, необходимо выполнить команду /sbin/lilo, чтобы изменения вступили в силу. Эта команда (которая в руководстве называется map-installer) обновляет карту загрузки системы. Прежде чем запускать /sbin/lilo для модификации загрузочных процедур, выполните эту команду с параметром -t. При этом будет выполнена вся процедура инсталляции загрузчика, кроме изменения map-файла и записи модифицированного загрузочного сектора, т. е. выполнен тест но-

вого варианта. Если добавить еще опцию -v, это позволит убедиться в том, что сделанные вами изменения разумны.

Не стоит полностью убирать возможность загрузки предыдущей, работающей версии ядра, тем более, что это никак не мешает загрузке нового ядра, ведь LILO обеспечивает многовариантную загрузку.

Все! После переустановки LILO командами

```
[root]# /sbin/lilo -t -v
[root]# /sbin/lilo
```

вы можете перезагрузить компьютер и выбрать при загрузке новое ядро.

17.6. Заключение

В заключение этой главы необходимо привести несколько коротких замечаний.

Если вы уже компилировали ядро из исходных кодов, то перекачивать полный их текст нет необходимости, можно провести обновление ядра через так называемые патчи (patch). Но, поскольку сам я таких операций не производил, рассказать об этом способе не берусь.

В документации к ядру говорится, что команду make config пропускать ни в коем случае нельзя, даже если вы провели самые незначительные обновления в исходных кодах.

Если вы хотите провести перекомпиляцию установленного ядра с минимальными трудозатратами, то вместо make config можно воспользоваться командой make oldconfig, которая задаст вам только вопросы, касающиеся тех параметров, которые вы хотите изменить. Но, насколько я понимаю, ее можно использовать только в том случае, если вы уже проводили установку ядра из исходников в своей системе, поскольку эта команда использует существующий файл ./.config. В этом файле сохраняются все ответы на вопросы, которые были вам заданы при выполнении команды make config. При проведении повторной процедуры конфигурирования ядра командой make oldconfig используется тот ответ на каждый вопрос, который сохранен в файле ./.config. Если же в этом файле нет готового ответа (не задано значение соответствующего параметра), программа конфигурирования будет ждать ответа пользователя. А готового ответа не оказывается в этом файле в двух случаях:

- если устанавливается обновленная версия ядра, в которой появился новый параметр;
- если вы закомментировали или удалили строку, соответствующую данному параметру, из файла ./.config.

Поэтому, если вы хотите просто перекомпилировать ядро, чтобы, например, вынести какие-то драйверы в отдельные модули, удалите соответствующие строки из файла ./.config и воспользуйтесь командой make oldconfig вместо make config.

Глава 18

Виртуальный компьютер (система VMware)

Операционная система Linux приобретает все большее число сторонников и уже начинает наступление на рабочие станции и ПК *(см. [П20.1] приложения)*. Существует, однако, одно большое препятствие на пути дальнейшего распространения этой ОС — привычка большинства пользователей ПК работать с Microsoft Office. Пакет этот, несмотря на то, что его часто ругают, пользуется популярностью и, надо признать, заслуженно. Поэтому, даже работая постоянно с ОС Linux, поневоле приходится сталкиваться с файлами, созданными с помощью программ из MS Office и сохраненными в собственных форматах: DOC, XLS, MDB. Как же быть приверженцам ОС Linux? Как организовать общение с миром приверженцев ОС Windows?

Отвечая на этот вопрос можно сказать, что некоторые офисные пакеты для Linux, например, KOffice, или текстовый процессор AbiWord, умеют работать с файлами MS. Однако пока что добиться 100%-ной совместимости в этом вопросе не удалось. Можно установить две операционные системы в разные разделы диска и перезагружаться в Windows, когда возникнет необходимость поработать с файлами от Microsoft Office *(см. [П20.2] приложения)*. Но это весьма неудобно. Так же неудобны и различные программы-перекодировщики, преобразующие doc-файлы в HTML или другой открытый формат, доступный для программ под Linux. (С появлением пакета OpenOffice.org ситуация начинает изменяться, см. *гл. 12)*.

Короче говоря, сообщество приверженцев Linux поставлено перед проблемой. И оно, естественно, эту проблему решает и даже несколькими способами. Их основная идея проста — создать эмулятор Windows, работающий под Linux и позволяющий запускать программы этой ОС. Но пути решения этой задачи каждый выбирает разные. Насколько мне известно, наибольших успехов в реализации идеи создания эмулятора на сегодня добились две фирмы: VMware, создавшая систему виртуальных машин VMware, и Netraverse с продуктом Win4Lin *(см. [П20.3] приложения)*. Оба эти продукта фактически позволяют запускать на компьютере вторую операционную систему, *одновременно* с уже работающей базовой ОС Linux, под управлением которой продолжает работать физический компьютер. В этой главе мы рассмотрим систему виртуальных машин от VMware.

18.1. Что такое "виртуальный компьютер"

Виртуальный компьютер — специальная программа, запускаемая в ОС Linux и моделирующая физический компьютер на основе процессора Intel x86. Монитором виртуального компьютера является окно графической оболочки X Window, в которое производится вывод информации. На рис. 18.1 видно, как выглядит Windows NT 4.0, работающая на виртуальном компьютере, запущенном из ОС Linux.

Примечание

Надо сразу сказать, что отдельная версия системы виртуальных машин разработана фирмой VMware и для того случая, когда базовой ОС является Windows NT/2000, так что можно и Linux запускать в окне графической оболочки Windows. Однако этот вариант здесь не рассматривается.

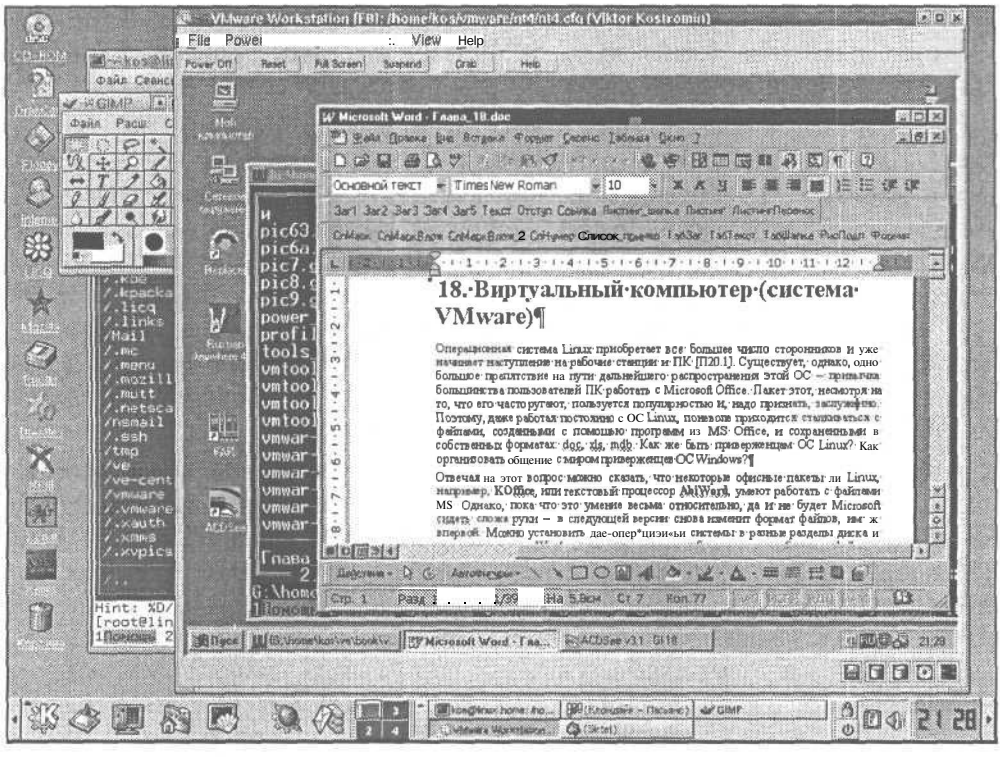


Рис. 18.1. Windows NT 4.0, запущенная на виртуальном компьютере с базовой ОС Linux

Виртуальный компьютер "строится" из следующего набора виртуальных устройств:

- виртуальные IDE и SCSI жесткие диски;
- виртуальный CD-ROM;
- стандартный дисковод гибких дисков;
- контроллер жестких IDE-дисков Intel 82371 PCI Bus Master, поддерживающий два первичных (primary) и два вторичных (secondary) IDE-диска;
- адаптер SCSI-дисков, совместимый с BusLogic BT-958;
- стандартный PCI графический адаптер;
- стандартная 101/102-клавишная клавиатура, PS/2-совместимая мышь;
- сетевая карта AMD PCNET Family Ethernet adapter (PCI-ISA);
- последовательные порты COM1—COM4,
- параллельные порты LPT1—LPT2;
- звуковая карта, совместимая с Sound Blaster 16.

Этот набор виртуальных устройств отличается от набора устройств реального компьютера, на котором запускается виртуальная машина (за исключением некоторых устройств, например, процессора и клавиатуры), и не зависит от последнего. Если операционная система устанавливается непосредственно внутри виртуального компьютера, то в процессе установки все эти устройства определяются корректно. При "включении питания" виртуального компьютера (что делается с помощью специальной кнопки **Power On/Off** в меню программы-эмулятора), видно, как BIOS осуществляет тестирование "аппаратной части" и даже, как и на физическом компьютере, можно войти в программу Setup, чтобы задать или изменить настройки BIOS. На виртуальный компьютер можно установить любую операционную систему и работать с ней обычным образом.

Естественно, что две ОС, одновременно работающие на одном физическом компьютере, так или иначе борются за реальные ресурсы базового компьютера, а поэтому требования к нему достаточно высоки. Фирма-разработчик формулирует эти требования следующим образом:

- Pentium II 266MHz или выше, с ОЗУ как минимум 64 Мбайт;
- видеоадаптер, поддерживаемый сервером XFree86 (для получения всех преимуществ полноэкранного режима);
- в качестве базовой операционной системы может использоваться ОС Linux с ядром 2.0.32 или выше, библиотекой glibc версии от glibc2 до glibc6 (с glibc1 не работает), для SMP-систем ядро должно быть версии 2.2.0 или выше;
- для VMware необходим X-сервер, причем рекомендуется XFree86-3.3.4 или выше.

18.2. Инсталляция системы виртуальных машин

Для установки необходимо сначала скачать ПО с сайта компании VMware, а также получить лицензию на его использование. Можно, конечно, купить лицензию (стоимость ее около 300 долларов), однако можно пользоваться и временной (30-дневной) лицензией, тем более, что компания пока что позволяет без ограничений обновлять ее. Неудобство, конечно, но терпимое. После регистрации вы получаете сообщение, что лицензия отправлена по почте, и в ожидании запускаете перекачку файла vmware-x.y.z-nnn.i386.rpm, (где x.y.z — номер версии, а nnn — номер релиза). После этого надо выполнить следующие действия:

1. Для инсталляции системы виртуальных машин надо иметь права пользователя root, поэтому запускаем терминальное окно и выполняем команду su.
2. Производим установку RPM-пакета

   rpm -Uhv vmware-x.y.z-nnn.i386.rpm

   (где vmware-x.y.z-nnn.i386.rpm — имя файла, который вы скачали).
3. Запускаем конфигурационный скрипт /usr/bin/vmware-config.pl.

   Замечние

   Этот скрипт можно будет использовать для того, чтобы заново сконфигурировать VMware каждый раз, когда происходит замена или обновление ядра. Переустанавливать VMware при этом нет необходимости. При выполнении скрипта на экране появляется ряд вопросов, некоторые ответы на которые можно найти в [П20.4, П20.5] *(см. приложение).* Отмечу только два момента. Первый момент касается используемого ядра. Я натолкнулся на это затруднение, когда ставил VMware на систему с ядром 2.2.16. К сожалению, в этой версии ядра имеется какая-то особенность, препятствующая нормальной работе системы виртуальных машин, поэтому пришлось поменять ядро на более позднюю версию. Однако если вы установите новое ядро не из исходных текстов, а из RPM-пакета (при инсталляции Linux из дистрибутива Red Hat и его клонов тоже происходит установка ядра из такого пакета), то в системе может не оказаться файлов заголовков ядра. А один из вопросов, задаваемых скриптом /usr/bin/vmware-config.pl, касается местонахождения файлов заголовков, соответствующих запущенной версии ядра. Необходимые файлы можно установить (переключившись во второй терминал) из пакета kernel-headers-x.y.z.i386.rpm (соответствующего установленному у вас ядру) и указать правильный путь к ним (этот путь можно узнать, просмотрев вывод команды rpm -qpl kernel-headers-x.y.z.i386.rpm).
4. После завершения работы скрипта можно покинуть shell, запущенный от имени root.

Инсталляция собственно системы виртуальных машин завершена, однако надо еще установить лицензию, создать собственно виртуальный компьютер, установить на нем ОС и пакет VMware Tools.

18.3. Установка лицензии на использование VMware

Лицензия высылается в виде текстового файла, присоединенного к сообщению. Для ее установки необходимо перейти в домашний каталог, создать в нем подкаталог с именем .vmware (`mkdir .vmware`), и скопировать полученный файл лицензии в этот подкаталог. Убедитесь, что имя файла начинается с подстроки license. Теперь можно уже запустить систему виртуальных машин командой vmware (исполняемый файл vmware находится в каталоге /usr/bin, так что должен запускаться без указания полного пути) и создать в ней собственно виртуальный компьютер (или несколько таких компьютеров).

18.4. Создание виртуальной машины

Для создания виртуальной машины (для краткости будем иногда писать ВМ) проще воспользоваться мастером конфигурации, но гораздо нагляднее этот процесс происходит при использовании "редактора конфигурации" (рис. 18.2), который запускается посредством выбора команды **Configuration Editor** в меню **Settings** системы VMware. Первым делом подключим жесткий IDE-диск, щелкнув по значку + рядом с надписью **IDE Devices.** После того как появятся 4 дополнительных строки, соответствующих четырем каналам контроллера жестких дисков, щелкните по строке с надписью **P-M. Not installed.** Эта строка обозначает жесткий диск на первом канале (Primery Master) и утверждает, что таковой не установлен.

Надо иметь в виду, что нельзя устанавливать второй диск (Slave), если не установлен первый диск (Master) на соответствующем канале контроллера (P-S не устанавливают до P-M, а S-S, соответственно, до S-M). Если не соблюсти это правило, то виртуальная машина не сможет загружаться с заданного таким образом диска. Обычно используют первую позицию, P-M (primary master), для подключения жесткого диска и третью позицию, S-M (secondary master), для диска CD-ROM. Возле надписи **Device Type** находится список выбора типа диска: виртуальный диск, плоский диск (plain disk), реальный диск (raw disk) и CD-ROM. Виртуальный диск (virtual disk) — это файл в файловой системе базового компьютера, который для ОС виртуальной машины выглядит как реальный физический диск. Этот файл может располагаться как на диске базового компьютера, так и на удаленной файловой системе. Если создается виртуальная машина с виртуальным жестким

диском, то можно установить на нее новую ОС без переразбиения физического диска и даже без перезагрузки базового компьютера. Плоский диск (plain disk) подобен виртуальному, но может иметь размеры более 2 Гбайт. Он компонуется из нескольких файлов-экстентов (extents), размером не более 2 Гбайт. В отличие от виртуальных дисков при создании плоского диска все отводимое под такой диск пространство сразу занимается и заполняется нулями. Реальный диск (Raw disk) — это жесткий диск или раздел жесткого диска базового компьютера, к которому получает прямой доступ виртуальная машина. При подключении такого диска становится возможным загрузить в виртуальной машине операционную систему, ранее установленную в один из разделов базового компьютера (при условии, что этот раздел находится на локальном IDE- или SCSI-диске).

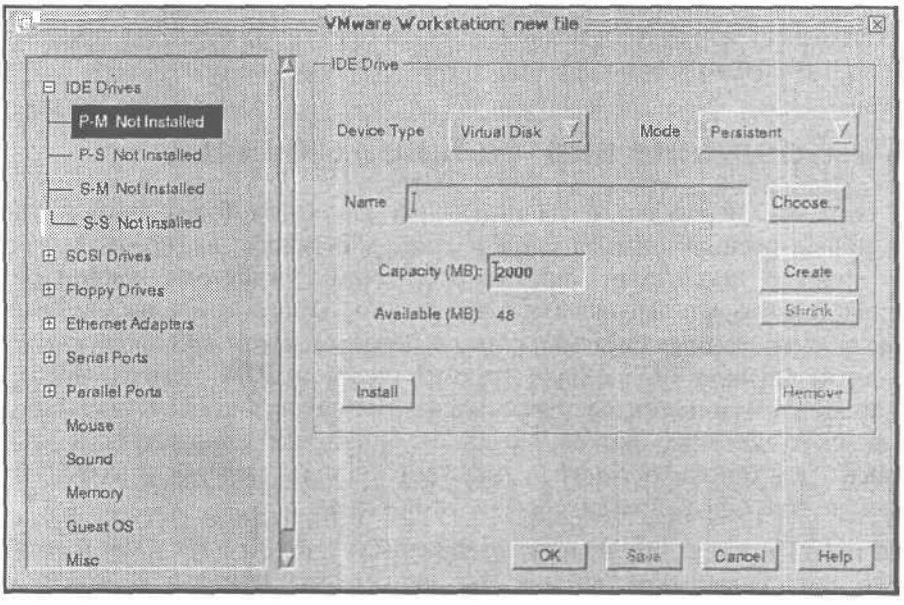


Рис. 18.2. Окно редактора конфигурации

После выбора типа диска надо выбрать один из трех возможных режимов его работы: persistent, nonpersisten, undouble.

В режиме "с записью" ("persistent") все операции записи немедленно производятся на реальный диск (или в файл, моделирующий реальный диск).

В режиме "без записи" ("nonpersistent") записи на диск, осуществляемые виртуальным компьютером, выглядят как операции записи на реальный диск, но фактически запись данных на физический диск не производится, и данные теряются по завершении сессии работы на виртуальном компьютере

(когда "выключается питание" виртуального компьютера или производится перезагрузка ОС). В этом режиме VMware только читает с реального диска, а операции записи в течение сессии производятся во временный файл (redo log file), который уничтожается при завершении сессии. Все блоки данных, которые были модифицированы и записаны в файл .redo, при повторном обращении к ним считываются уже из этого файла, а не с реального диска. По завершении сессии файл уничтожается. Файл .redo располагается в том же каталоге, где по умолчанию располагаются файлы виртуальных дисков, однако его местоположение можно изменить, воспользовавшись командой **Misc** в меню редактора конфигурации. Режим "без записи" удобен, когда требуется запускать виртуальный компьютер из одного и того же состояния, например при тестировании разрабатываемого или для демонстраций нового ПО. В этом режиме могут работать все типы дисков.

Режим "с отложенной записью" ("undoable") очень похож на режим "без записи" в том смысле, что все операции записи на диск, осуществляемые виртуальным компьютером, фактически производятся во временный файл (.redo) на реальном диске. Но при отключении питания виртуального компьютера пользователю предлагаются на выбор три возможности: записать все изменения на реальный диск; отказаться от изменений, возвращая диск к его исходному состоянию; запомнить изменения, чтобы в следующей сессии начать работу с того состояния, в котором закончена работа в предыдущем сеансе (сохранить redo-файл). Режим полезен, когда надо поэкспериментировать с установкой нового ПО или выполнением некоторых административных функций, что может вызвать проблемы в работе компьютера. Если сохранить файл redo, то при следующем запуске ВМ будет предложено либо восстановить все изменения, сделанные в ходе предыдущей сессии, либо отказаться от них, либо отключить ВМ.

Если вы только начинаете осваивать систему VMware, то оптимальным вариантом из числа рассмотренных является, на мой взгляд, подключение виртуального диска в режиме "persistent".

После установки типа диска и режима записи введите в поле **Name** имя файла, которое будет использоваться для данного виртуального диска и укажите размер виртуального диска (если это просто виртуальный диск, то его объем не может быть больше 2 Гбайт). Первоначально файл виртуального диска имеет объем не более 1 Мбайт, и его размер возрастает только по мере установки программного обеспечения на виртуальную машину. После задания всех параметров виртуального диска щелкните по кнопке **Install**.

SCSI-диски устанавливаются вполне аналогично IDE-дискам.

Установка CD-ROM еще проще, чем жесткого диска. Тут надо задать только один параметр — имя устройства. Можно еще определить, подключать ли CD-ROM автоматически при запуске виртуальной машины. Для подключе-

ния дисковода гибких дисков надо выбрать тип: устройство **(Device)** или файл **(File),** и задать или выбрать имя устройства (например, /dev/fd0 или /dev/fd1), определив, будет ли дисковод подключаться автоматически. Необходимо иметь в виду, что физический Floppy-дисковод не может использоваться одновременно операционными системами двух (и более) виртуальных машин или виртуальным и базовым компьютером. В процессе работы с виртуальной машиной можно в любой момент отключить дисковод, воспользовавшись командой **Devices** главного меню. И, наоборот, можно подключить дисковод через ту же команду меню, освободив его предварительно в других ВМ и в базовом компьютере. На базовом компьютере для этого надо размонтировать диск (в Linux) или переключиться на какие-то каталоги других дисков во всех запущенных программах (если на базовом компьютере запущена Windows).

Следующим шагом по идее должна быть установка сетевой карты, но пока этот этап пропустим, поскольку, как и в реальном компьютере, подключиться к сети можно и позже. То же самое можно сказать про подключение последовательных и параллельных портов и звуковой карты. А вот подключить мышь и выделить виртуальной машине оперативную память просто необходимо. Система VMware позволяет пользователям задавать как объем оперативной памяти, выделяемой каждому виртуальному компьютеру, так и общее количество ОП, зарезервированное для использования виртуальными машинами. Правильная настройка этих параметров очень важна, поскольку может существенно повлиять на производительность как виртуального компьютера, так и системы в целом. Количество памяти, выделяемое всем виртуальным машинам, задается через меню **Settings** системы VMware. При работе с редактором конфигурации можно определить только количество ОП, выделенное данной виртуальной машине. Для начала, пока у вас всего одна виртуальная машина, выделите ей половину оперативной памяти базового компьютера.

Остается только выбрать операционную систему и задать значения некоторых дополнительных параметров (команда **Misc).** Для первого раза проще всего согласиться с тем вариантом, который задается по умолчанию. Единственный из этих дополнительных параметров, значение которого нужно ввести, это путь к файлу .redo, и то только в том случае, когда необходим режим работы "с отложенной записью".

После создания виртуальной машины необходимо сохранить ее конфигурацию в файле, для чего щелкнуть по кнопке Save и ввести имя конфигурационного файла.

Поскольку на созданный таким образом виртуальный компьютер еще не установлена ОС, переходим к ее установке. Процедура установки ОС – обычная (разве что потребуется войти в BIOS виртуального компьютера и установить возможность загрузки с CD-ROM): вставляется загрузочный CD-ROM в дисковод и запускается виртуальный компьютер.

Сразу же после первого запуска ОС на виртуальной машине просто необходимо установить дополнительные компоненты для ОС, которые называются VMware Tools и включают в себя некоторые дополнительные драйверы, в частности драйвер SVGA. Хотя система VMware способна работать и без него, однако из графических режимов на виртуальной машине будет доступен только режим VGA (640x480, 16 цветов). Если же установить драйвер SVGA из VMware Tools, то будут поддерживаться высокие разрешения дисплея и глубина цвета до 32 бит, причем повысится и быстродействие графической подсистемы.

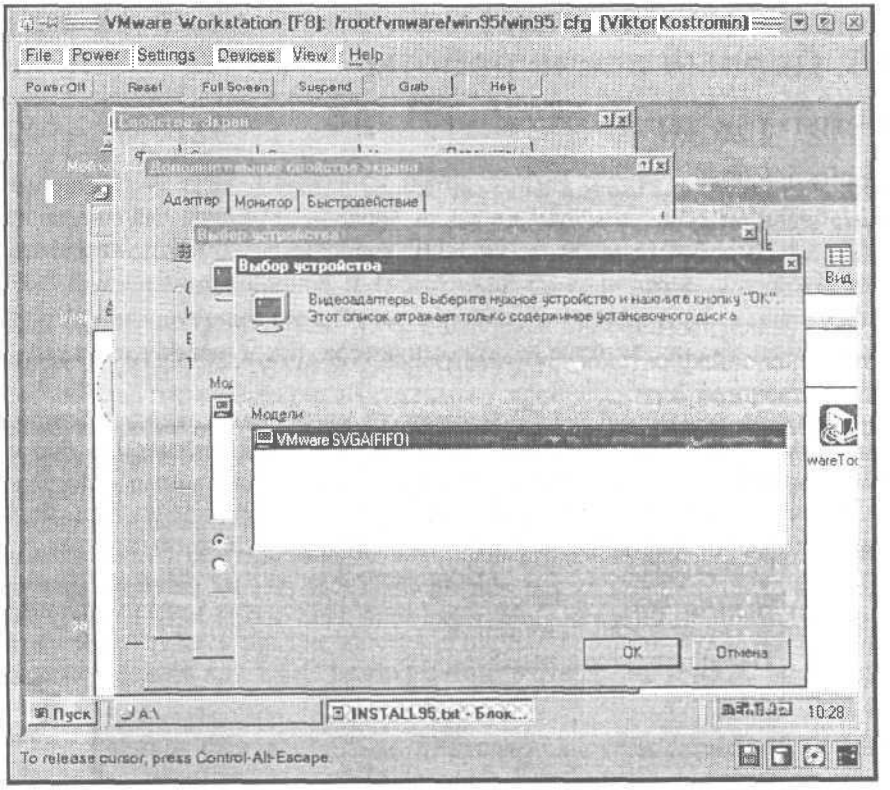


Рис. 18.3. Установка драйвера SVGA

Для установки VMware Tools после запуска ОС на виртуальном компьютере передайте управление базовой ОС (комбинацией клавиш <Ctrl>+<Alt>+ +<Esc>) и выберите команду меню **Settings | VMware Tools Install** программы VMware. Вы увидите сообщение о том, что конфигурация виртуальной машины временно изменена для установки VMware Tools. Изменение конфигурации виртуальной машины заключается в том, что вместо диска A:

подключается образ дискеты, содержащий необходимые для установки VMware Tools файлы. Запустите с этой виртуальной дискеты программу VMwareTools.exe (из подкаталога, соответствующего установленной вами ОС). В процессе инсталляции программа установки предупреждает, что будет запущена утилита изменения установок видеоадаптера, с чем нужно согласиться, после чего остается только нажать кнопку **Finish.** Инструкции по конфигурации видеодрайвера даются в открывающемся автоматически окне программы Notepad. После того как откроется окно свойств экрана, доберитесь до вкладки **Адаптер,** щелкните по кнопке **Изменить** и выберите вариант **Установить с диска** (остальное понятно из рис. 18.3).

18.5. Первый сеанс работы на виртуальном компьютере

Запускать систему VMware можно двумя способами. Первый заключается в том, что открывается окно терминала, и вводится команда vmware &. Второй способ (в KDE) — в главном меню KDE выбирается команда **Запустить программу** ("горячие" клавиши — <Alt>+<F2>) и в появившемся окне вводится команда vmware. Независимо от способа запуска вы увидите диалоговое окно выбора конфигурации виртуального компьютера, представленное на рис. 18.4.

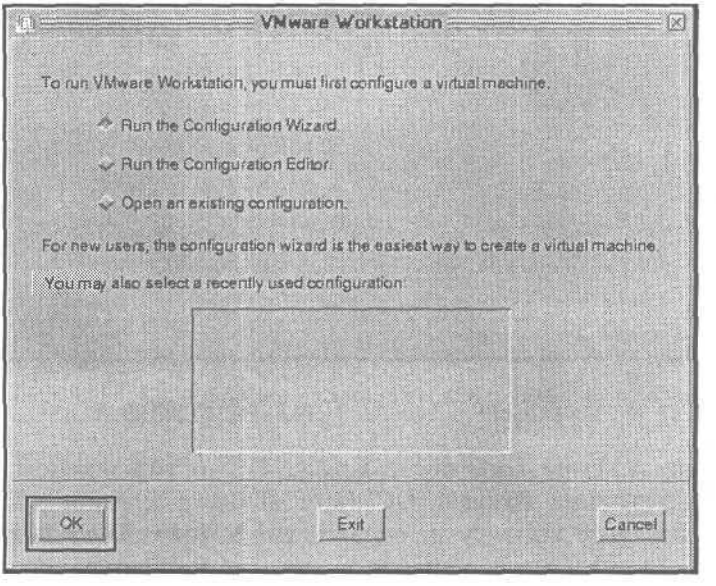


Рис. 18.4. Окно выбора конфигурации

Поскольку у вас пока создан только один виртуальный компьютер, то достаточно подсветить строку с именем единственного конфигурационного файла, а затем надо "включить питание" кнопкой **Power On** главного окна программы VMware.

Дальше все будет происходить так, как и при загрузке выбранной ОС на обычном компьютере. В частности, если есть желание посмотреть или изменить установки BIOS виртуального компьютера, то надо успеть вовремя нажать клавишу <F2>, чтобы попасть в меню BIOS.

Первое, о чем надо помнить при работе с виртуальной машиной — это способ выхода из окна виртуальной машины комбинацией клавиш <Ctrl>+<Alt>+<Esc>, что особенно полезно при работе в полноэкранном режиме ВМ. В этот режим можно переключиться с помощью кнопки **Full Screen** на панели VMware. В этом случае будет полная иллюзия того, что физический компьютер работает под управлением Windows и только "волшебная" комбинация клавиш <Ctrl>+<Alt>+<Esc> может вернуть вас к действительности.

Следующим этапом работы на "новом компьютере" будет установка необходимого ПО внутри виртуальной машины, которая осуществляется точно так же, как и на обычном компьютере.

Запустите виртуальную машину. На всякий случай проверьте, что виртуальная машина имеет доступ к дисководу CD-ROM или дисководу гибких дисков (в зависимости от того, какой из них потребуется в процессе инсталляции). Для этого можно воспользоваться командой **Devices** главного меню системы VMware.

Вставьте установочный диск в соответствующий дисковод и запустите программу установки. Как это сделать, я здесь не буду объяснять, надеюсь, что вы имеете некоторые навыки работы с той операционной системой, которую собираетесь запускать на ВМ.

После инсталляции ОС я установил на своем виртуальном компьютере MS Office и еще ряд программ, с которыми привык работать в старой системе. Сделайте то же самое и начинайте работать!

18.6. О некоторых особенностях работы с виртуальным компьютером

18.6.1. Копирование и вставка

Если на виртуальной машине установлен пакет VMware Tools, имеется возможность осуществлять операции копирования и вставки между приложениями, запущенными в виртуальной машине и на базовом компьютере, а также между двумя виртуальными машинами.

18.6.2. Приостановка и мгновенное восстановление состояния ВМ

В любой момент работы с виртуальной машиной можно приостановить ее работу и сохранить текущее состояние, а впоследствии продолжить работу с точки останова, причем все открытые приложения и документы будут снова открыты и готовы к дальнейшей работе. Сохранять состояние виртуальной машины можно либо на диске, либо в оперативной памяти. По умолчанию сохранение осуществляется на диск (правда, для выполнения этой операции на диске должно быть достаточно свободного места). Если вы хотите, чтобы состояние ВМ сохранялось в ОП, воспользуйтесь редактором конфигурации, чтобы изменить установку по умолчанию.

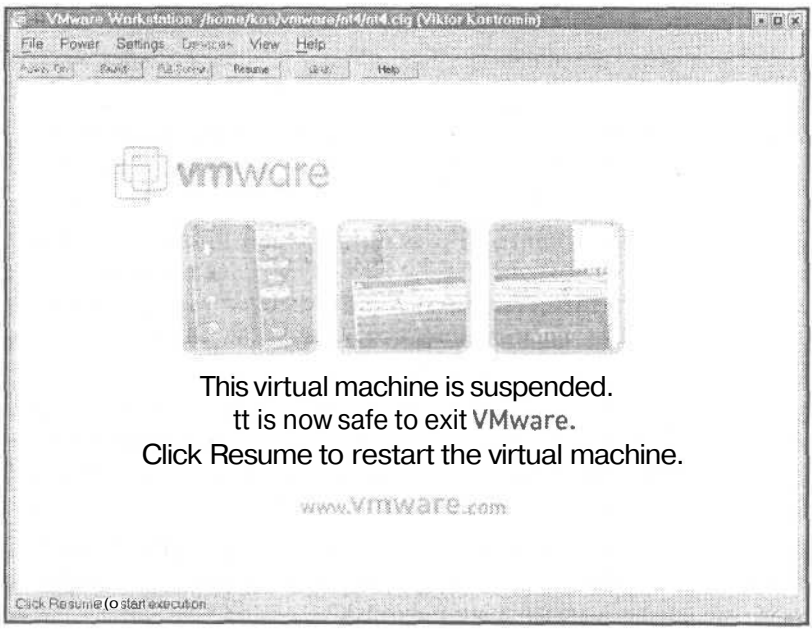


Рис. 18.5. Виртуальная машина приостановлена

Скорость сохранения и восстановления состояния ВМ зависит от того, как много изменений было сделано в последнем сеансе работы. В общем случае первое сохранение длится дольше, чем последующие.

Для сохранения состояния ВМ требуется:

1. Вернутся в режим работы в окне (комбинация клавиш <Ctrl>+ +<Alt>+<Esc>).
2. Щелкнуть по клавише **Suspend** на панели VMware.

После этого можно безопасно выйти из системы VMware через команду **File | Exit.**

Для возврата в состояние до приостановки:

1. Запустить VMware и выбрать ту виртуальную машину, работу которой приостановили.
2. Щелкнуть по кнопке **Resume** на панели VMware.

Все приложения, с которыми вы работали в момент приостановки ВМ, снова запустятся, причем окажутся в том самом состоянии, какое было в момент остановки.

18.6.3. Выключение ВМ

Как и на реальных компьютерах, перед выключением питания виртуальной машины необходимо выполнить процедуру остановки (Shutdown) запущенной на ней ОС. После того как ОС будет остановлена (появится соответствующее сообщение, либо окно VMware станет черным), щелкните по клавише **Power Off** на панели VMware. После этого можно закрыть VMware любым способом из тех, которые служат для закрытия окна (например, воспользовавшись командой меню **File | Exit).**

18.6.4. Использование прямого доступа к памяти

Windows 95 OSR2 и более поздние версии обладают возможностью использования прямого доступа к памяти (Direct Memory Access, DMA) при обращении к IDE-дискам. Однако эта опция не часто используется по умолчанию, хотя ее применение в виртуальном компьютере может дать существенный прирост производительности. Для того чтобы задействовать режим DMA в Windows 95/98 на виртуальном компьютере, запустите **Панель управления** и выберите меню **Система;** в появившемся окне **Свойства: Система** выберите вкладку **Устройства,** найдите пункт **Дисковые накопители** и раскройте список, щелкнув по значку +. Затем последовательно для каждого подключенного IDE-диска поставьте отметку (галочку) рядом с надписью **DMA,** после чего перезапустите ОС. Не забудьте задействовать режим DMA в базовой ОС Linux. О том, как это делается, было подробно рассказано в *разд. 9.5.*

В результате этих усилий VMware и все запускаемые в виртуальном компьютере приложения будут работать быстрее.

18.6.5. Выделение оперативной памяти для VMware

Система VMware позволяет пользователям задавать как объем оперативной памяти, выделяемой каждому виртуальному компьютеру, так и общее количество ОП, зарезервированное для использования виртуальными машинами. Правильная настройка этих параметров очень важна, поскольку может существенно повлиять на производительность как виртуального компьютера, так и системы в целом.

Первый конфигурационный параметр, значение которого может устанавливаться пользователем, — это общее количество памяти, которое резервируется для всех запущенных виртуальных машин. Этот параметр может быть задан перемещением движка в окне, вызываемом через команду Host Reserved Memory в меню Settings.

В общем случае память, используемая каждой виртуальной машиной, берется из того же самого пула памяти, который используется ОС на базовом компьютере и всеми запущенными на нем приложениями. Однако, для повышения общей производительности, система VMware устанавливает задаваемый пользователем лимит памяти для всех виртуальных машин. Когда VMware использует эту память, она недоступна для других приложений, запущенных на базовом компьютере. Но когда VMware не использует эту память, она становится доступной для других приложений. Резервируя память, VMware позволяет виртуальным машинам работать более эффективно.

Память, используемая системой VMware, включает память, отдаваемую операционной системе виртуального компьютера, а также некоторое количество избыточной памяти, необходимой для функционирования самого виртуального компьютера. Объем этой избыточной памяти зависит от нескольких факторов, но обычно не превышает 10 мегабайт. Кроме того, для нормального функционирования ОС виртуальной машины надо зарезервировать достаточное для этой ОС количество памяти.

Количество реально резервируемой системой VMware оперативной памяти динамически меняется в процессе работы системы. VMware использует зарезервированную память только тогда, когда определяет, что это необходимо для достижения приемлемой производительности виртуального компьютера. Даже если запущены несколько ВМ, реально может использоваться только часть зарезервированной памяти, а неиспользуемая зарезервированная память отдается ОС базового компьютера и запущенным в ней приложениям.

Рекомендуется резервировать для системы VMware 50% физической памяти базового компьютера. Отходить от этого правила могут только опытные пользователи, поскольку изменение этого параметра может существенно повлиять на производительность как базового, так и виртуального компью-

тера. Если выбрать слишком большое значение этого параметра, это может привести к сильному замедлению работы базового компьютера или даже к его зависанию. Слишком малое значение этого параметра приводит к падению производительности виртуального компьютера и ограничивает число ВМ, которые могут быть одновременно запущены.

Linux плохо ведет себя при нехватке оперативной памяти. По этой причине не стоит запускать одну или несколько виртуальных машин, если им требуется больше ОП, чем остается на базовом компьютере после запуска ОС и других приложений. Точнее, надо придерживаться следующего правила: "*Общее количество оперативной памяти, выделяемой для всех одновременно запущенных виртуальных машин, не может превышать количества физической ОП минус объем памяти, которая необходима для работы ОС базового компьютера и запущенных в ней приложений*".

Впрочем, система VMware сама ограничивает количество ВМ, которые могут быть одновременно запущены, исходя из количества зарезервированной для нее ОП. Если вы пытаетесь включить питание виртуальной машины, а количества зарезервированной ОП для ее работы недостаточно, включения ВМ не произойдет.

Второй конфигурационный параметр, который могут изменять пользователи системы VMware, — это объем физической оперативной памяти, выделяемый данной виртуальной машине. Значение этого параметра задается в Редакторе конфигурации **(Settings | Configuration Editor | Memory).** Минимальное значение этого параметра определяется требованиями ОС. Мастер конфигурации вообще не запрашивает у пользователя значение этого параметра, выбирая его исходя из того, какую ОС выбрал пользователь.

Оптимальное значение размера памяти, отводимой виртуальному компьютеру, зависит от нескольких факторов.

- Какие приложения будут запускаться на виртуальной машине.
- Будут ли другие виртуальные машины, запущенные наряду с данной ВМ, конкурировать с ней за разделение оперативной памяти.
- Какие приложения будут запускаться на базовом компьютере одновременно с данной виртуальной машиной.

18.7. Подключение физических дисков к виртуальному компьютеру

Итак, мы создали и запустили виртуальный компьютер, работающий с виртуальным жестким диском. Но изолированный компьютер в наши дни уже смотрится как-то архаично, и естественно возникает желание обмениваться файлами как с базовым компьютером, так и с другими компьютерами (как

реальными, так, может быть, и виртуальными). Давайте рассмотрим, как это желание удовлетворить. В настоящем разделе опишем работу с физическими дисками, а в следующем поговорим о выходе в локальную сеть.

18.7.1. Необходимые меры предосторожности

Прежде чем описывать процедуры подключения физического диска к виртуальному компьютеру, надо рассказать о некоторых опасностях, которые тут нас подстерегают. В документации, размещенной на сайте фирмы VMware, имеется следующее предостережение:

"Поддержка работы с физическими дисками является продвинутой особенностью (an advanced feature) системы VMware и может использоваться только пользователями, которые уже знакомы с продуктом. А чтобы познакомиться с продуктом, вы должны, как минимум, создать и сконфигурировать виртуальную машину с виртуальным диском и установить на нее операционную систему. Что касается загрузки ранее установленной на физический диск операционной системы в виртуальный компьютер, то она может не работать для некоторых конфигураций аппаратного обеспечения и операционной системы".

Это не означает, что подключение к виртуальной машине реальных дисков в принципе невозможно. Просто надо делать такое подключение корректно, с соблюдением некоторых мер предосторожности.

Основная опасность, связанная с использованием реальных дисков, состоит в одновременном доступе к одному разделу жесткого диска из нескольких операционных систем. Все ОС создавались в расчете на полный контроль над компьютером. Поскольку каждая ОС представления не имеет о другой, то когда две ОС пытаются производить операции записи или чтения в одном и том же разделе реального диска, может произойти потеря или даже разрушение данных. Дело в том, что система VMware (пока еще) не регулирует дисковые операции базовой операционной системы. Поэтому раздел реального диска не должен одновременно использоваться (быть смонтирован) в ОС на базовом компьютере и в виртуальной машине.

Следовательно, вы должны удостовериться, что базовая ОС "не видит" раздел, с которым работает ОС виртуального компьютера. Безопасность работы с реальными дисками определяется выполнением этого требования. Поэтому, прежде чем подключить раздел реального диска к виртуальной машине, размонтируйте его в базовой ОС.

Если вам необходимо осуществить обмен данными между базовым и виртуальным компьютерами, можно подключать один и тот же диск к этим компьютерам поочередно. Для этого придется вначале смонтировать раздел в базовой ОС Linux, перенести на него необходимые данные, размонтировать диск, запустить VMware и виртуальный компьютер, скопировать данные на

виртуальный диск, выключить VMware и снова отдать диск базовой ОС. Альтернативой такому очевидно неудобному способу переноса данных является использование сетевых возможностей ОС, таких как протоколы Samba или NFS, для переноса данных из одного компьютера в другой. Эти возможности будут рассмотрены в следующем разделе, а пока давайте научимся подключать физический диск (или раздел на таком диске) к уже созданному виртуальному компьютеру в качестве второго жесткого диска.

18.7.2. Подключение физического диска к виртуальному компьютеру

Итак, мы имеем виртуальный компьютер, на котором работает ОС Windows (в одном из ее вариантов), запускаемая с виртуального диска C:. И, предположим, у нас имеется раздел жесткого диска (пусть, для определенности, это будет раздел /dev/hda2), который был отформатирован в той же ОС от Microsoft (в FAT, FAT32 или NTFS, в зависимости от варианта ОС). Естественно, возникает желание получить доступ к этому разделу из виртуального компьютера. Попытаемся подключить этот раздел в качестве диска D: виртуального компьютера. Но прежде, чем описывать конкретные процедуры подключения, дадим некоторые предварительные сведения.

Права доступа к дискам

Жесткие диски, к которым вы хотите получить доступ из виртуального компьютера (и, в частности, диски, с которых происходит запуск операционных систем, как это будет описано в *разд. 18.7.3),* должны быть доступны как по чтению, так и по записи для пользователей, запускающих систему VMware. В большинстве дистрибутивов Linux физические диски (такие как /dev/hda, /dev/hdb) принадлежат группе disk. Если это так, то можно просто добавить пользователей системы VMware в эту группу. Можно также просто поменять владельца устройства. Пожалуйста, тщательно продумайте вопросы безопасности при выборе способа предоставления доступа к дискам. Самый простой и вполне приемлемый на персональном компьютере способ заключается в том, чтобы дать пользователям системы VMware доступ ко всем физическим устройствам /dev/hd[abcd], к которым надо обращаться из виртуальных машин, а в вопросах разграничения доступа положиться на конфигурационные файлы VMware.

Файл описания физического диска

Чтобы система VMware могла получить доступ к физическим дискам, для каждого из таких дисков должен быть создан небольшой файл, содержащий некоторые данные, необходимые виртуальной машине для получения доступа к разделам данного диска. В документации на VMware такой файл называют Safe Raw Disk, мы будем называть его файлом описания физиче-

ского диска. Вот типичный пример такого файла для компьютера, на котором установлены ОС Windows NT и Linux:

```
DEVICE /dev/hda
# Partition type: MBR
RDONLY 0 62
# Partition type: HPFS/NTFS
ACCESS 63 8193149
# Partition type: Linux swap
NO_ACCESS 8193150 8466254
```

Как видите, этот файл содержит информацию о разделах диска, типе файловой системы в каждом разделе (правда, только в строке комментария) и о правах доступа к разделу. Эту информацию можно представить в виде табл. 18.1.

Таблица 18.1. *Информация из файла описания диска*

Тип раздела	Размещение (секторы)	Access Rights
Загрузочная запись	С 0 по 62 включительно	Read-Only
NTFS или FAT	С 63 по 8 193 149 включительно	Read-Write
Linux swap	С 8 193 150 по 8 466 254 включительно	Нет доступа

Если операционная система, запущенная на виртуальном компьютере, попытается произвести операции чтения или записи в секторы, доступ к которым запрещен в таком файле описания физического диска, система VMware выдаст пользователю диалоговое окно, в котором потребует подтвердить правомочность данной операции или отказаться от ее выполнения.

Процедура подключения физического диска

Имея в виду только что сказанное, можно приступить к подключению физического диска к виртуальному компьютеру, для чего надо выполнить следующие действия.

1. Добавьте пользователя, от имени которого будете запускать систему VMware, в группу disk (это делается путем редактирования файла /etc/group суперпользователем).
2. Убедитесь в том, что подключаемый физический диск не смонтирован в файловой системе базового компьютера.
3. Чтобы создать файл описания физического диска, запустите систему VMware, выберите нужную конфигурацию (но не включайте питание виртуального компьютера) и откройте меню **Settings | Configuration Editor,** после чего щелкните по значку + слева от указания на IDE- или SCSI-диски.

4. Найдите строку, в которой указано, что соответствующий диск не установлен (Not **installed),** и установите на нее подсветку (курсор). Предположим, например, что вы выбрали строку **P-S Not Installed** среди IDE-дисков. Значит виртуальная машина будет считать, что данный физический диск подключен как второй диск (slave) к первому контроллеру (primary IDE controller). Соответственно, если в группе SCSI-дисков найдется строка **SCSI 0:1 Not Installed,** то для виртуального компьютера такой диск будет иметь номер 1 на SCSI-контроллере. Если строки **Not Installed** не найдется, то к вашему виртуальному компьютеру подключено уже 4 IDE-диска (или, соответственно, 7 SCSI-дисков), т. е. достигнут предел. В таком случае удаляйте какой-нибудь диск, пользуясь кнопкой **Remove.**
5. В поле **Device Type** установите (выберите) значение **Raw Disk.**
6. В поле Name введите имя для файла описания физического устройства (например, raw_hda.dsk).
7. Щелкните по кнопке **Create Raw Disk.**
8. В появившейся строке ввода укажите имя физического диска (не раздела, а именно диска, например, /dev/hda для IDE-диска или /dev/sda для SCSI).
9. Появится новое окно, в котором выведен список разделов, имеющихся на данном физическом диске. Для каждого раздела укажите права доступа, которые будет иметь в данном разделе виртуальная машина. Для каждого раздела нужно выбрать один из следующих вариантов задания прав:
   - No Access — виртуальный компьютер не будет иметь возможности ни читать, ни производить запись в данный раздел. Этот вариант выбирают только в том случае, если необходимо проконтролировать попытки (несанкционированного) обращения к данному разделу;
   - **Read/Write** — виртуальный компьютер будет иметь возможность и читать, и производить запись в данный раздел. Эту опцию выбирают только для тех разделов, которые содержат файловые системы, "родные" для операционной системы виртуального компьютера;
   - **Read-Only** — виртуальный компьютер будет иметь возможность только читать из данного раздела. Выбирайте этот вариант для всех остальных разделов на диске.
10. Щелкните по кнопке Save. В некоторых случаях после этого может появиться окно, сообщающее, что два раздела на диске пересекаются (имеют общие секторы) и, следовательно, для них должны быть заданы одинаковые права доступа. Такого вообще-то быть не должно (и эту ситуацию необходимо как-то исправлять), но если все же такое окно появится, вы можете задать одинаковые права для обоих разделов и снова щелкнуть по кнопке Save. Файл описания физического диска будет за-

писан в каталог, где хранятся остальные файлы вашей виртуальной машины (что-то вроде /home/user1/vmware/nt4/).

11. Щелкните по кнопке Install для того, чтобы присоединить выбранный физический диск к виртуальному компьютеру. Как и в случае виртуального диска, вы можете задать для физического диска один из трех возможных режимов работы: "с записью" ("persistent"), "без записи ("nonpersistent") или "с отложенной записью" ("undoable").

После завершения всех этих действий можете загрузить ОС в виртуальном компьютере, и вы должны увидеть в своей системе новый диск.

Если в последующем вам почему-либо потребуется отключить физический диск от виртуального компьютера (например, для того, чтобы смонтировать его в файловой системе базового компьютера), откройте Редактор конфигурации (Settings | Configuration Editor) и щелкните по экранной кнопке Remove на вкладке, соответствующей данному диску. На этой же вкладке имеется кнопка Edit Raw Disk, с помощью которой можно откорректировать права доступа к разделам диска, определяемые файлом физического диска. Обратиться к этой опции вам придется в тех случаях, когда вы, скажем, заменили физический диск в компьютере или модифицировали разбиение его на разделы.

18.7.3. Загрузка ОС с физического диска

Раз имеется возможность подключать физические диски к виртуальному компьютеру, то, естественно, возникает вопрос: "А нельзя ли загружать операционную систему виртуального компьютера с физического диска?" Такой вопрос особенно актуален в том случае, когда до установки системы VMware на вашем компьютере уже были установлены в разные разделы как одна из операционных систем Windows, так и ОС Linux (в которой вы запускаете виртуальный компьютер). И ответ на этот вопрос положителен. Система VMware может даже использовать загрузчики, установленные ранее на компьютере. Загрузчик будет работать внутри VMware и даст возможность пользователю выбрать операционную систему, запускаемую на виртуальном компьютере. Можно и заново установить, например, Windows 98 на физический диск, а потом запускать ее в виртуальной машине.

VMware пока что (в версии 2) поддерживает загрузку с реальных дисков только для IDE устройств (в то время как файл, моделирующий виртуальный диск, может быть расположен как на IDE, так и на SCSI диске). Однако использование ОС, установленной на физическом диске, сопряжено с некоторыми особенностями, которые надо учитывать при настройке обеих ОС (даже кроме тех опасностей, о которых мы уже говорили в *разд. 18.7.1)*. Первой из таких особенностей является необходимость создания отдельного профиля оборудования для Windows.

Операционные системы фирмы Microsoft (включая Windows 95, Windows 98, Windows NT 4.0) используют понятие "профиля оборудования". Каждый профиль определяет некоторый набор известных системе устройств. Если заданы два или более профиля, пользователю в процессе загрузки предлагается выбрать один из них.

ОС Windows 95, Windows 98 и Windows 2000 благодаря механизму Plug and Play в процессе загрузки проверяют соответствие реальных устройств указанному профилю оборудования. Несоответствие приводит к тому, что снова запускается механизм определения устройств и установки драйверов. Хотя в большинстве случаев этот процесс завершается успешно, это существенно замедляет загрузку.

Windows NT не поддерживает Plug and Play и использует профиль оборудования для инициализации устройств. Несоответствие реального набора тому, что указано в профиле, вызывает выдачу сообщения об ошибке и отключение (точнее неподключение) устройства.

А поскольку конфигурация виртуального компьютера отличается от конфигурации физического компьютера, то для запуска одной из операционных систем семейства Windows внутри виртуальной машины надо создать отдельный профиль оборудования, чтобы упростить процесс загрузки. Поэтому процесс создания и конфигурирования виртуальной машины, которая использует операционную систему, установленную в один из разделов физического диска, имеет некоторые отличия от процесса создания виртуальной машины, работающей с виртуальными дисками.

1. Вначале проинсталлируйте операционную систему, которую вы хотите запускать на виртуальном компьютере, на физический IDE-диск реального компьютера (естественно, это делать не нужно, если ОС уже была установлена ранее).
2. До запуска системы VMware загрузите эту ОС (имеется в виду одна из ОС семейства Windows) на реальном компьютере и создайте два профиля оборудования. Для этого откройте **Панель управления,** войдите в меню **Система** и переключитесь на вкладку **Профиль оборудования.** Там уже имеется как минимум один профиль, который называется **Текущий** (Original configuration). Щелкните по кнопке **Копировать** и назовите новый профиль, например, **Виртуальная машина.**
3. **Только для Windows NT/2000:** Отключите некоторые устройства во вновь созданном профиле. Для этого откройте окно **Устройства** из панели управления, выберите отключаемое устройство и нажмите кнопку **Остановить.** Отключить необходимо аудиоплату, MIDI, джойстик, плату Ethernet и другие сетевые, а также USB-устройства (отключать их надо только во вновь созданном профиле, не промахнитесь). Если вы установили и предполагаете запускать в виртуальном компьютере Windows 95

или Windows 98, то отключать устройства не требуется. Они будут отключены автоматически на стадии загрузки ОС.

4. Перезагрузите компьютер и запустите Linux.
5. Убедитесь, что раздел физического диска, который отведен для использования операционной системой виртуального компьютера, не смонтирован в Linux. Удалите или закомментируйте соответствующую строку в файле /etc/fstab, а в данном сеансе размонтируйте этот раздел из командной строки.
6. Установите права доступа к разделам жесткого диска. О том, как это сделать, было сказано *в разд. 18.7.2.* Самый простой и вполне приемлемый способ заключается в том, чтобы включить пользователей системы VMware в группу disk, дав тем самым доступ ко всем физическим устройствам /dev/hd[abcd], которые содержат операционные системы или загрузчик, а в вопросах разграничения доступа положиться на конфигурационные файлы VMware. Таким образом обеспечивается доступ для загрузчика к файлам, необходимым для запуска операционных систем (например, LILO требуется доступ по чтению к каталогу /boot в разделе Linux для запуска операционных систем, отличных от Linux, которые могут быть расположены в других разделах или на других дисках).
7. Сконфигурируйте виртуальную машину под вновь установленную операционную систему (используя **Мастер конфигурации** или **Редактор конфигурации).** При выполнении процедуры конфигурации для реальных дисков учтите следующие моменты.
   - При выборе типа виртуального диска выберите вариант **Existing Partition.**
   - Для раздела диска, в котором находится соответствующая операционная система, установите опцию **read/write** (для этого надо щелкнуть мышкой по экранной кнопке **Partitions** в окне Редактора конфигурации, соответствующем нужному жесткому диску). Для основной загрузочной записи (Master boot record, MBR) и для других разделов диска(ов) рекомендуется дать право только на чтение (read only), поскольку, например, загрузчик LILO для загрузки операционной системы должен иметь возможность прочитать файл из каталога /boot в Linux-разделе.

Примечание

Еще раз напомним, что если позволить виртуальной машине производить запись в раздел, который одновременно смонтирован в файловой системе Linux, то возможны непредвиденные последствия *(см. разд. 18.7.1).* Поэтому, прежде чем позволять виртуальной машине производить запись в раздел, убедитесь, что этот раздел не смонтирован в Linux на базовом компьютере.

8. Запустите VMware и проверьте созданную конфигурацию. Для этого можно дать команду `vmware <config-file>`, где `<config-file>` — это полный путь к конфигурационному файлу, созданному Мастером конфигурации (имена таких файлов оканчиваются на .cfg). Можно также дать просто команду vmware и открыть файл конфигурации через меню **File** | Open. После этого откройте меню Settings | Configuration **Editor** и убедитесь в том, что в конфигурации IDE-дисков указан хотя бы один физический диск и для него введено имя файла описания диска (raw disk description file). Имена этих файлов обычно имеют вид <configuration-name>.hda.dsk, <configuration-name>.hdb.dsk, и т. д. Можно проверить и другие опции конфигурации, особенно такие, для которых вы приняли значения по умолчанию, например, вы можете изменить значение объема памяти, выделяемой виртуальной машине.

9. Включите питание виртуальной машины (кнопка Power On). Система VMware запускает Phoenix BIOS, после чего считывается главная загрузочная запись загрузочного диска (master boot record, MBR). Если вы сконфигурировали систему с использованием нескольких IDE-дисков, VMware BIOS будет пытаться произвести загрузку ОС с этих дисков в следующей последовательности:

   - Primary Master
   - Secondary Master
   - Primary Slave
   - Secondary Slave

   Если у вас несколько SCSI-дисков, VMware BIOS производит загрузку в порядке номеров SCSI-устройств.

   Если в вашей системе сконфигурированы как SCSI-, так и IDE-диски, VMware BIOS сначала пытается загрузить ОС со SCSI-устройств, затем — с IDE-дисков. Опрос устройств производится в той же последовательности, как было сказано выше.

   Порядок обращения к дискам в процессе загрузки можно изменить через меню Boot в Phoenix BIOS виртуальной машины. Для этого после включения питания VMware нажмите клавишу <F2>, чтобы попасть в меню BIOS.

10. Если у вас установлено несколько операционных систем (многовариантная загрузка), то выберите нужную ОС тем же способом, как вы делали это до установки системы VMware (из меню, предлагаемого при загрузке).

И. В процессе загрузки ОС должно появиться меню выбора конфигурации (если, конечно, вы создали отдельный профиль оборудования для виртуального компьютера).

   Введите номер, соответствующий конфигурации виртуального компьютера (в ситуации, изображенной на рис. 18.6, это будет 2) и нажмите клавишу <Enter>. В процессе дальнейшей загрузки ОС вы получите не-

которые сообщения об ошибках и дополнительные задержки в процессе загрузки, но это нормально.

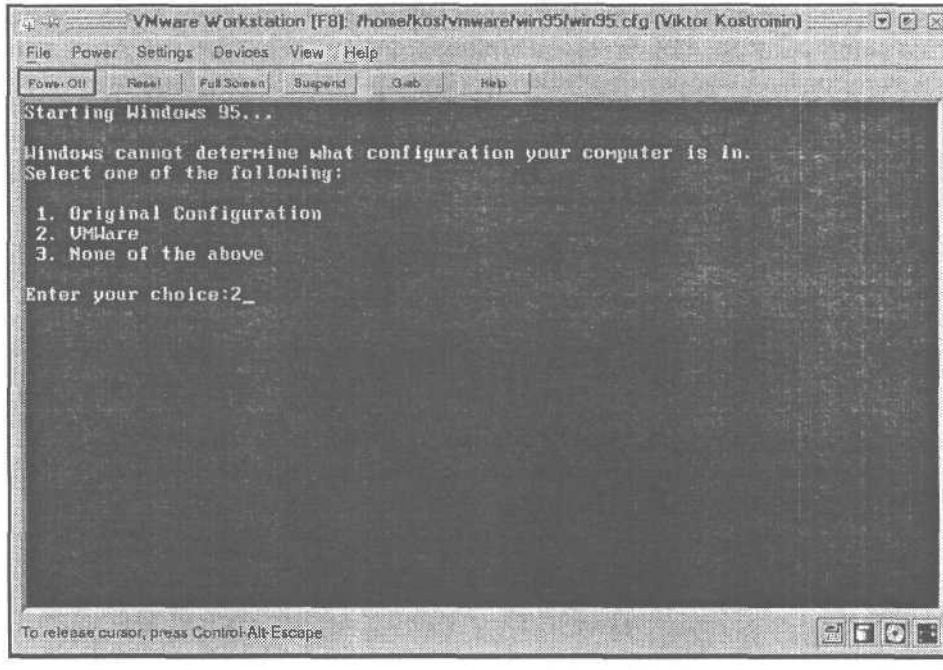


Рис. 18.6. Выбор профиля оборудования для виртуального компьютера

12. **Только для Windows** 2000: После того как вы запустите Windows 2000 (в качестве ОС на виртуальном компьютере) вы увидите диалоговое окно **Найдено новое оборудование** (Found New Hardware), в котором предлагается установить новый драйвер для видеоконтроллера. Этого делать не нужно. Щелкните по кнопке **Отмена** (Cancel) для того, чтобы закрыть диалоговое окно и откажитесь от предлагаемой перезагрузки компьютера. Windows 2000 автоматически обнаружит и установит драйвер для сетевой карты AMD PCnet PCI Ethernet. После этого вы должны установить пакет VMware Tools для Windows (на виртуальном компьютере). После того как будет установлен SVGA-драйвер от фирмы VMware, Inc. (входящий в состав пакета VMware Tools для Windows), перезагрузите ОС Windows 2000 на виртуальной машине. После перезагрузки вы можете поменять разрешение экрана у виртуальной машины **(Свойства экрана | Параметры).**

    Если вы хотите использовать звуковую карту, работая с ОС Windows 2000 на виртуальном компьютере, прочитайте руководство по ее подключению на сайте фирмы VMware.

Только для Windows 95/98: вы увидите диалоговое окно **Обнаружено новое оборудование.** Windows предложит вам произвести поиск драйверов для него. Для большинства устройств драйверы уже установлены при инсталляции системы, однако в некоторых случаях может понадобиться установочный CD-ROM-диск. Windows попросит вас несколько раз перезагрузиться при установке новых драйверов.

В некоторых случаях Windows может не распознать CD-ROM-диск, когда выдается запрос на поиск драйверов. В таком случае рекомендуется попытаться указать в качестве пути к драйверу каталог C:\Windows\System\ или отказаться от установки драйвера данного конкретного устройства. Подключение таких устройств может быть выполнено позже.

Когда Windows установит виртуальные устройства и драйверы для них, надо удалить из системы неработающие устройства, соответствующие реальному оборудованию. Для этого нажмите на значок **Система** в панели управления и выберите используйте вкладку **Устройства.** Выберите неработающее устройство и щелкните по кнопке **Удалить.** Только учтите, что нужно предварительно выбрать профиль оборудования, соответствующий виртуальному компьютеру, чтобы не удалить устройства, работающие при запуске ОС с физического диска.

Только для Windows NT: После завершения загрузки ОС просмотрите протокол загрузки, чтобы определить те устройства, которые не подключились. Вы можете отключить их в профиле **Виртуальный компьютер,** используя менеджер устройств **(Панель управления | Устройства).**

13. Убедитесь, что все виртуальные устройства работают корректно, особенно сетевые адаптеры. Помните, что состав оборудования виртуального компьютера существенно отличается от набора устройств, реально имеющихся на вашем физическом компьютере.

    Только для Windows 95/98: Если какое-то виртуальное устройство отсутствует, воспользуйтесь опцией **Панель управления | Добавить новое оборудование.**

14. Установите VMware Tools (если вы еще не сделали этого). Пакет VMware tools будет запускаться в обеих конфигурациях оборудования, но окажет какое-то влияние на работу только в конфигурации "Виртуальный компьютер".

Примечание

Когда вы в следующий раз загрузите Windows в реальном компьютере, используя профиль оборудования, соответствующий реальной конфигурации аппаратуры, в списке устройств могут появиться некоторые виртуальные устройства. Вы можете удалить их или отключить тем же самым способом, который был описан выше для отключения реальных устройств из профиля оборудования, соответствующего виртуальному компьютеру.

Примечание

Если вы при задании конфигурации виртуального компьютера установили для реального диска режим "с отложенной записью" (undoable), то при перезагрузке ОС вы должны будете либо согласиться с тем, чтобы все операции с диском, проделанные внутри виртуальной машины были сохранены на диске, либо отказаться от сохранения изменений. Подробнее о режимах работы дисков см. е разд. *18.4.*

18.8. Выход в локальную сеть

Подключить физический диск к виртуальному компьютеру удается не всегда. Самая очевидная причина затруднений состоит в том, что на физическом диске создана файловая система, с которой не умеет работать ОС виртуального компьютера. И хотя можно пытаться установить специальные драйверы, но делать это (на мой взгляд) не стоит, поскольку можно организовать обмен данными с базовым компьютером, а также со всем остальным миром, с помощью сетевых средств. Они изначально создавались в расчете на взаимодействие различных ОС, так что предоставляют необходимые средства обмена данными. Судя по моему опыту, это гораздо более реалистичный и безопасный способ организации такого обмена. Но сначала несколько пояснений общего плана.

18.8.1. Четыре варианта организации сетевых служб в системе VMware

Каждая виртуальная машина, которую вы создаете, может иметь свою независимую конфигурацию сетевых служб. Существует 4 возможных варианта конфигурации:

- Без подключения к сети (No networking).
- Host-only networking.
- Bridged networking.
- Custom networking.

Конфигурация "No networking" просто означает, что виртуальная машина работает сама по себе, не имея возможности взаимодействовать с операционной системой базового компьютера или другими компьютерами (включая виртуальные компьютеры, работающие на том же базовом). Этот вариант стоит рассматривать только в том случае, когда виртуальная машина будет использоваться, например, в целях тестирования ПО или для обеспечения безопасности хранимой на ней информации. Такая конфигурация задается очень просто — достаточно при конфигурировании виртуальной машины не подключать сетевой адаптер (либо отключить его впоследствии).

Вариант "Host-only networking" означает, что виртуальный компьютер сможет взаимодействовать с операционной системой базового компьютера и любым виртуальным компьютером, запущенным на базовом компьютере и тоже имеющим сетевые возможности. Но виртуальный компьютер в такой конфигурации не сможет взаимодействовать с системами, находящимися *вне* базового компьютера (если только не используется прокси-сервер, запущенный на базовом компьютере). Создается как бы частная виртуальная сеть, которая состоит из базового компьютера и всех запущенных на нем виртуальных (в частности, одного виртуального). Обычно все хосты такой сети используют стек протоколов TCP/IP, хотя жесткого требования использовать именно его нет. Но какие бы протоколы не использовались, каждый компьютер в такой сети должен иметь свой адрес. Адреса могут назначаться "статически" или "динамически". В последнем случае используются такие протоколы, как DHCP (Dynamic Host Configuration Protocol).

Если вариант "Host-only networking" задействуется при инсталляции системы VMware, на базовом компьютере по умолчанию запускается DHCP-сервер. Этот сервер используется для поддержки протокола DHCP *только* для виртуальных компьютеров, подключаемых к виртуальной сети через интерфейс vmnet1. Операционные системы виртуальных компьютеров, в которых задана опция использования DHCP, при загрузке получают IP-адрес автоматически, без какой-либо дополнительной настройки. Операционные системы Windows по умолчанию обычно подключают использование динамически назначаемых адресов.

Если в настройках операционной системы не задано использование протокола DHCP, то в системе необходимо явно прописать "статический IP-адрес". Этот вариант адресации имеет смысл применять в том случае, когда вы хотите, чтобы виртуальные машины могли обращаться друг к другу по именам, а не по IP-адресам. Но в таком случае вы должны поддерживать базу данных соответствий имен и IP-адресов на каждом компьютере или запустить на базовом компьютере сервер имен (DNS). В документации фирмы VMware рекомендуется применять статическую IP-адресацию (или сконфигурировать DHCP-сервер так, чтобы виртуальной машине всегда назначался один и тот же IP-адрес) в том случае, если вы намереваетесь использовать виртуальный компьютер в течение длительного периода времени. Если же вы устанавливаете его ненадолго, используйте DHCP и предоставьте ему возможность распределять IP-адреса.

Отметим, что обычно (в соответствии с принятыми в Интернете соглашениями) IP-адреса для виртуальных сетей распределяются в соответствии с табл. 18.2.

Таблица 18.2. *Распределение IP-адресов*

Диапазон	Используется для	Пример
<net>.1*	Базовый компьютер	192.168.0.1

Вообще говоря, можно запустить на базовом компьютере одновременно как стандартный сервер Samba, так и тот вариант этого сервера, который поставляется вместе с системой VMware Workstation. Однако при этом надо учитывать, что версия стандартного Samba-сервера должна быть не ниже 2.0.6 и он должен быть корректно сконфигурирован. Определить версию стандартного Samba-сервера можно командой `smbd -v`, а для корректной настройки его фирма VMware предлагает воспользоваться примером конфигурационного файла smb.conf, размещенным на сайте фирмы.

Поддержка сетевых возможностей *в операционной системе виртуального компьютера* осуществляется с помощью виртуального Ethernet-адаптера(ов). К одному виртуальному компьютеру можно подключить до 3 таких адаптеров и они "представляются" операционной системе как адаптеры типа AMD PCNET PCI. Большинство операционных систем умеют распознавать такие адаптеры и автоматически подключают соответствующий драйвер. Поэтому для завершения конфигурирования сети в ОС виртуального компьютера остается только корректно задать необходимые параметры сетевого подключения (сетевой адрес компьютера, маску подсети, IP-адрес сервера имен и т. д.).

18.8.3. Назначение MAC-адресов для виртуальных компьютеров

Когда "включается питание" виртуального компьютера, система VMware автоматически назначает ему MAC-адрес (уникальный адрес физического устройства, используемый на канальном уровне для управления доступом к устройствам). Система гарантирует, что виртуальным машинам будут присвоены уникальные MAC-адреса в рамках одного базового компьютера. Но не гарантируется, что при каждом запуске виртуального компьютера ему будет назначаться один и тот же MAC-адрес. Точно так же система не может обеспечить (хотя и пытается это сделать) назначение уникальных адресов для виртуальных компьютеров, запускаемых на нескольких базовых компьютерах в одной физической сети.

Если вы хотите, чтобы виртуальный компьютер всегда получал один и тот же MAC-адрес или хотите добиться полной уникальности назначаемых адресов, вы можете назначать их "вручную", а не автоматически. Для этого надо добавить в конфигурационный файл виртуального компьютера строку следующего вида:

```
ethernet0.address = 00:50:56:XX:YY:ZZ
```

Заметим, что если вы, придерживаясь указанного формата, присвоите фиксированный адрес только некоторым виртуальным компьютерам в сети (а другие будут получать адреса динамически), конфликтов между адресами, назначенными "вручную", и адресами, назначенными автоматически, возникнуть не должно.

18.8.4. Установка средств сетевой поддержки

А теперь, получив необходимые предварительные знания, займемся подключением виртуального компьютера к сети.

Но прежде чем заниматься таким подключением, вы должны определить для себя, какой из вариантов подключения вы будете использовать: "Host-only" или "Bridged networking" (как было сказано выше, вариант "Custom" мы не рассматриваем). Эти два варианта ниже будут рассмотрены отдельно.

Хотя в разделе об инсталляции системы VMware было сказано, при задании конфигурации виртуального компьютера можно отказаться от конфигурирования сетевой поддержки, однако для того, чтобы такую поддержку задействовать, вам придется переустановить систему VMware. К счастью, сделать это очень просто, причем при такой переинсталляции не нарушается конфигурация созданных в системе виртуальных компьютеров (в частности, сохраняется вся информация, записанная на виртуальных дисках). Дело в том, что и конфигурация виртуальных машин и все относящиеся к ним файлы хранятся в двух подкаталогах домашнего каталога пользователя, создавшего виртуальный компьютер: ~/vmware и ~/.vmware. Эти каталоги не изменяются при переустановке ПО VMware, и после такой переустановки ранее созданные виртуальные машины будут снова запускаться без проблем (по крайней мере, в том случае, если вы не меняете версии ПО).

Для того чтобы переустановить систему Vmware, надо сначала удалить ее, а потом установить заново. Если она устанавливалась из RPM-пакета, то переустановка выполняется командами

```
[root]# rpm -qa | grep VMware
```

(позволяет узнать точное имя установленного пакета, которое нужно в следующей команде)

```
[root]# rpm -e VMware-2.0.3-799
[root]# rpm -Uhv VMware-2.0.3-799.i386.rpm
```

причем перед запуском третьей команды надо перейти в каталог, где располагается указанный пакет.

Если вы устанавливали систему из tar-архива, то для ее удаления надо запустить скрипт vmware-uninstall.pl.

После того как вы переустановили ПО, необходимо (как и при первой инсталляции) запустить скрипт vmware-config.pl, и теперь уже не пропускать

этап задания конфигурации сети. Ниже приводится образец диалога, который происходит на этом этапе:

```
Do you want this script to automatically configure your system to allow
your Virtual Machines to access the host filesystem? (yes/no/help)

The version of Samba used in this version of VMware is licensed as de-
scribed in the "/usr/share/doc/vmware/SAMBA-LICENSE" file.
Hit enter to continue.

Enabling networking (this is required to share the host filesystem).
Trying to find a suitable vmnet module for your running kernel.

None of VMware's pre-built vmnet modules is suitable for your running
kernel. Do you want this script to try to build the vmnet module for your
system (you need to have a C compiler installed on your system)? [yes]

Extracting the sources of the vmnet module. Building the vmnet module.
The module loads perfectly in the running kernel.

Enabling host-only networking (this is required to share the host file-
system).
Do you want this script to probe for an unused private subnet?
(yes/no/help) [yes]

What will be the IP address of your host on the private network?
192.168.36.20
What will be the netmask of your private network? 255.255.255.0
```

Ответ на первый из приведенных в этом примере вопросов определяет, будет ли на базовом компьютере установлен Samba-сервер фирмы VMware. Если вы отвечаете утвердительно (yes), то на базовом компьютере устанавливается Samba-сервер vmware-smbd (это специально доработанная фирмой версия сервера Samba, см. выше). После этого ваше участие требуется еще только для того, чтобы решить, задать ли IP-адреса самому или предоставить их выбор скрипту. Это решение вы принимаете, когда отвечаете на вопрос

```
Do you want this script to probe for an unused private subnet?
(yes/no/help).
```

Если вы решили использовать только вариант "Bridged networking", то лучше ответить "n" и указать в ответе на следующий вопрос реальный адрес, полученный от администратора сети. Если же вы решите создать виртуальную

сеть ("Host-only networking"), то лучше предоставить выбор адресов скрипту. Впрочем, и в последнем случае можно задавать адреса самому, придерживаясь соглашений, о которых было кратко рассказано выше.

Если же на первый вопрос в приведенном примере вы отвечаете отрицательно (то есть отказываетесь от установки Samba-сервера от фирмы VMware), то следующим вопросом будет:

```
Do you want to be able to use the network in your Virtual Machines?
[yes] .
```

Как видите, этот вопрос предполагает только один вариант ответа (если вы отвечаете "нет", то отказываетесь от поддержки сети вообще). В случае же утвердительного ответа скрипт еще раз интересуется, не желаете ли вы сконфигурировать и host-only networking:

```
Do you want to be able to use host-only networking in your Virtual
Machines? [yes]
```

Если вы ответите "нет", то сможете использовать только вариант "Bridged networking", о чем свидетельствует появление следующего сообщения:

```
Starting VMware services:
     Virtual machine monitor                  [ OK ]
     Virtual ethernet                         [ OK ]
     Bridged networking on /dev/vmnet0        [ OK ]
```

Если же вы задействовали и вариант "Host-only networking", то это сообщение примет вид:

```
Starting VMware services:
     Virtual machine monitor                                  [ OK ]
     Virtual ethernet                                         [ OK ]
     Bridged networking on /dev/vmnet0                        [ OK ]
     Host-only networking on /dev/vmnet1 (background)         [ OK ]
```

В заключение скрипт еще попросит вас ввести имя и пароль пользователя, которому будет дан доступ к серверу Samba, и завершит работу.

Как следует из изложенного, интерфейс vmnet0 (используемый для "Bridged networking") задействуется в любом случае, даже если вы пытаетесь настроить сетевые службы системы VMware только на использование варианта "Host-only".

После завершения работы конфигурационного скрипта запустите систему VMware, выберите (если их несколько) нужный вам конфигурационный файл виртуального компьютера (через меню **File | Open)** и, не запуская виртуальный компьютер, проделайте следующее:

1. Запустите Редактор конфигурации (меню **Settings | Configuration Editor).**

2. Щелкните по значку + слева от надписи **Ethernet Adapters.** Появятся три дополнительные строки, соответствующие трем возможным виртуальным сетевым адаптерам. Переместите подсветку (курсор) на первую из этих строк. В правой части окна (которое до этого было пустым) появится картинка, подобная той, что изображена на рис. 18.7.

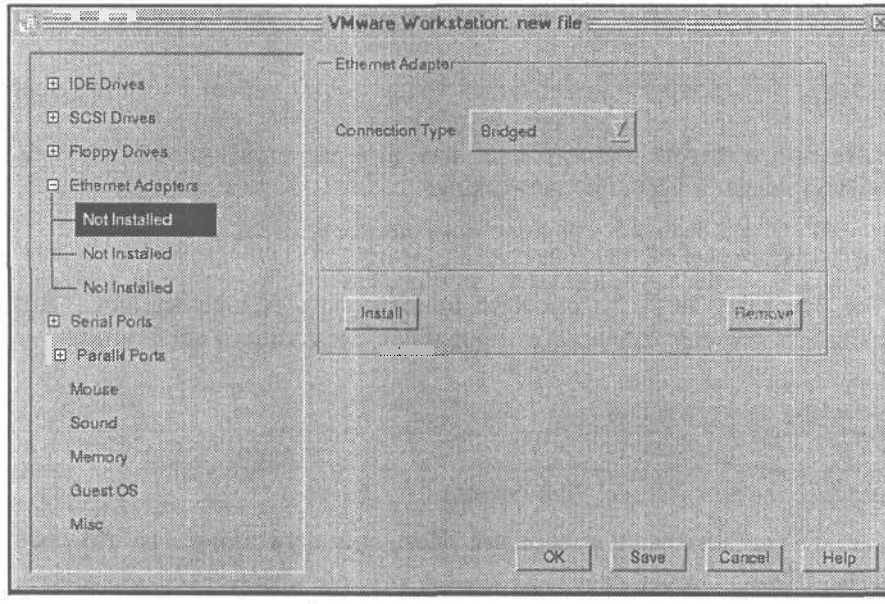


Рис. 18.7. Окно настройки сетевого адаптера

3. Щелкните по треугольнику возле выпадающего меню выбора типа подключения **(Connection Type)** и выберите один из трех возможных вариантов **(Bridged, HostOnly** или **Custom),** в соответствии с принятым вами ранее решением. Как было сказано выше, вариант **Custom** выбирать не стоит, пока вы не освоите систему VMware значительно лучше автора этой книги.
4. После этого надо щелкнуть по кнопке **Install** и сохранить конфигурацию щелчком по кнопке Safe.

На этом установка необходимых сетевых средств системы VMware завершена. Однако требуется еще сконфигурировать сетевые службы операционной системы, запускаемой на виртуальном компьютере (в качестве пояснения приведем рис. 18.8).

При этом в качестве сетевой платы надо выбрать плату AMD PCNET Family Ethernet adapter (PCI-ISA), а затем либо задать фиксированный сетевой адрес, либо задействовать динамическое получение адреса по протоколу

DHCP. Более подробно о том, как это можно сделать, смотрите в руководствах по ОС, запускаемой на виртуальном компьютере. Стоит только отметить, что если вы установите на виртуальном компьютере несколько ОС и будете выбирать одну из них при зафузке, то надо будет настроить сетевые службы в каждой из этих ОС. Поскольку чаще всего в каждый момент времени может быть запущена только одна из этих ОС, можно использовать один и тот же IP-адрес во всех этих системах. Скорее всего вам придется перезапустить ОС виртуального компьютера, чтобы сделанные изменения вступили в силу.

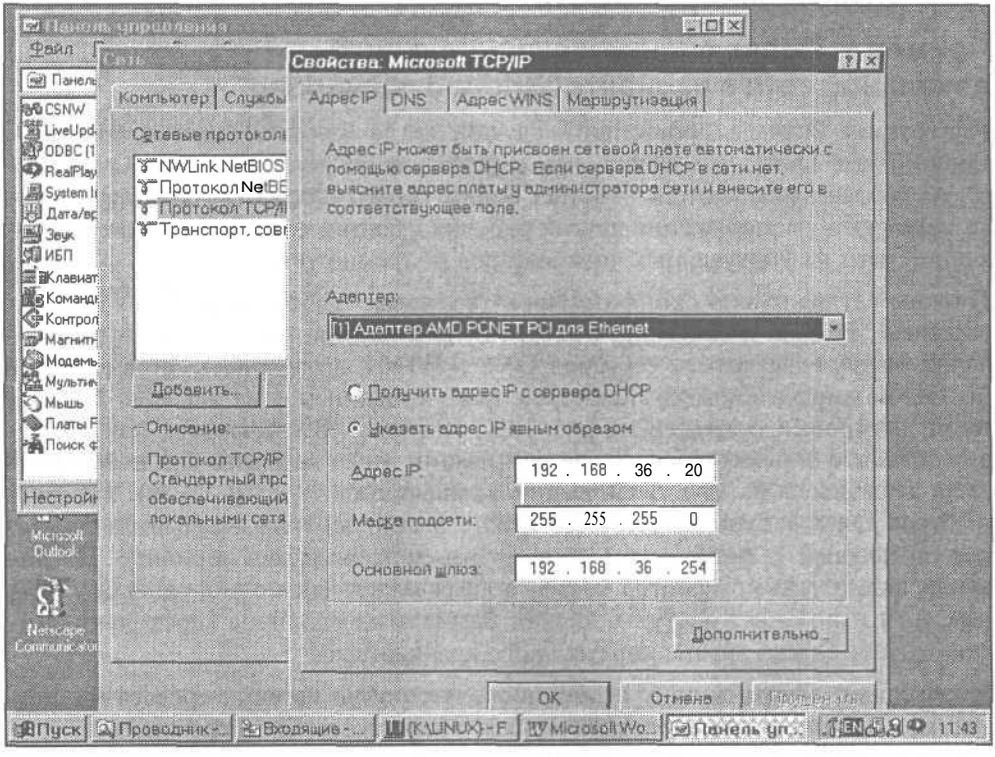


Рис. 18.8. Настройка сетевых средств в ОС виртуального компьютера

Если вы подключаетесь к реальной физической сети, то после этого вы сможете "увидеть" другие компьютеры локальной сети, раскрыв окно **Сетевое окружение** в Windows. А следовательно вы получите доступ и к тем ресурсам (дискам, каталогам, принтерам), которые на этих компьютерах отданы "в общее пользование". Однако ресурсы вашего базового компьютера вы, скорее всего, еще не увидите. Для того чтобы дать доступ из Windows к дискам Linux-компьютера, необходимо запустить на последнем сервер Samba и

правильно его настроить. Полностью описывать здесь настройки Samba-сервера нет никакой возможности: об этом написаны отдельные книги.

А теперь рассмотрим несколько примеров настройки для конкретных ситуаций, связанных с системой VMware. Надеюсь, что их рассмотрение позволит понять принципы такой настройки, а в совокупности с ранее приведенными сведениями, — и настроить нужную вам конфигурацию выхода виртуального компьютера в сеть.

18.8.5. Несколько примеров настройки выхода в сеть

Пример 1. Подключение к существующей локальной сети в варианте "Bridged networking"

Рассмотрим сначала простейший случай, когда базовый компьютер, работающий под Linux, уже подключен к реальной физической сети. Кроме того, на базовом компьютере работает Samba-сервер, предоставляющий какие-то каталоги в распоряжение других рабочих станций сети, и создание виртуальной сети из виртуальных компьютеров не планируется.

В таком случае нужно сконфигурировать сетевые службы системы VMware в варианте "Bridged networking", получить у администратора сети реальный IP-адрес, маску сети, адреса серверов DNS и WINS, и настроить сетевые службы ОС на виртуальном компьютере с использованием этих адресов. Пример такой настройки приведен на рис. 18.8, а на рис. 18.9 показано, как выглядит "Сетевое окружение" в случае реализации этого варианта для небольшой сети, состоящей из двух физических компьютеров (`Kos3` и Linux). На компьютере Linux запущена система VMware и виртуальный компьютер vmware, подключенный к физической сети по рассматриваемому варианту. Из рисунка видно, что компьютер vmware показывается как полноправный участник сети и получает доступ к дискам базового компьютера (поскольку снимок сделан в окне экрана виртуального компьютера).

Необходимо отметить одну особенность настройки сетевых средств на виртуальном компьютере, проявляющуюся тогда, когда ОС виртуального компьютера загружается с физического диска. Особенность состоит в том, что в этом случае обязательно надо создать отдельный профиль оборудования для загрузки ОС Windows в виртуальном компьютере, в которой отключить реальную сетевую карту Ethernet, как это было сказано в *разд. 18.7*. В противном случае у вас могут возникнуть трудности с подключением адаптера AMD PCNET PCI для Ethernet, который должен работать в виртуальном компьютере. Впрочем, то же самое верно и для других вариантов, так что не забывайте создавать отдельный профиль оборудования для запуска на виртуальном компьютере.

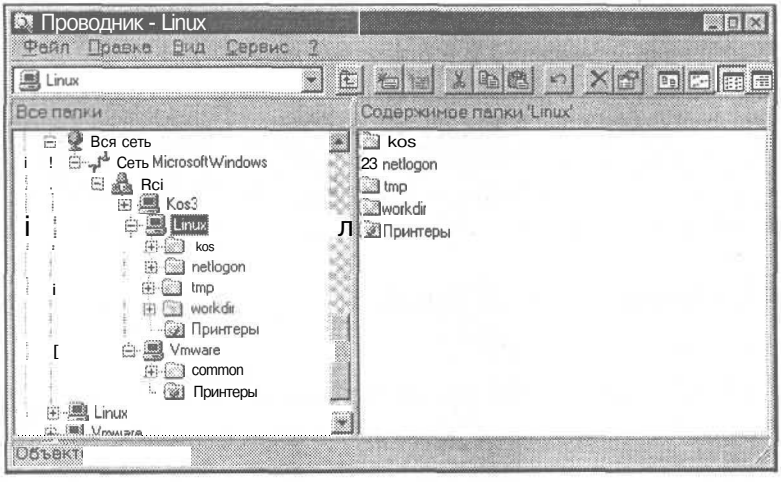


Рис. 18.9. Доступ к диску базового компьютера через "Сетевое окружение"

Пример *2.* Создание сети на изолированном компьютере

В качестве следующего примера рассмотрим создание виртуальной сети на изолированном (скажем, домашнем) компьютере. При конфигурировании системы VMware в этом случае надо выбрать вариант "Host-only networking", для чего на вопрос конфигурационного скрипта vmware-config.pl "Do you want this script to automatically configure your system to allow your virtual machines to access the host file system?" надо ответить "Yes". При этом будет установлен Samba-сервер vmware-smbd (а также необходимый для его работы демон vmware-nmbd) и будет организован их запуск при загрузке ОС Linux. Конфигурационный файл для такого сервера располагается не в каталоге /etc/samba, как для стандартного сервера Samba, а в каталоге /etc/vmware/vmnet1/smb, хотя и называется по-прежнему smb.conf. О том, как его настроить, вы можете узнать в документации по Samba или из man-страницы (man 5 smb.conf), поскольку он строится точно так же, как файл smb.conf для стандартного сервера Samba.

Естественно, что необходимо настроить сетевые службы в ОС Linux базового компьютера и на виртуальном компьютере. IP-адреса можно задать произвольным образом. Поскольку в такой сети будет работать только несколько компьютеров (в простейшем случае — всего два), то сервер DHCP запускать не имеет смысла, проще прописать все компьютеры и их адреса в файле /etc/hosts.

Рис. 18.10 иллюстрирует именно такой вариант работы виртуального компьютера. На рисунке показано, что весь диск базового компьютера подклю-

чен как сетевой диск G: к виртуальному компьютеру, причем в окне проводника Windows отображается вся структура каталогов Linux.

Рис. 18.10. Диск базового компьютера смонтирован как диск G: в ОС виртуального компьютера

Конечно, давать полный доступ к Linux-разделу диска в реальной жизни не стоит, если вы задумываетесь и об обеспечении безопасности в вашей сети, т. к. все системные файлы Linux становятся доступны для редактирования из виртуального компьютера. Надо корректно настроить сервер Samba, предоставив доступ только к специально выделенному каталогу.

Пример 3. Соединение виртуальной и физической сети

Теперь предположим, что вы решили создать несколько виртуальных компьютеров на одном базовом, объединить их в виртуальную сеть и соединить ее с реальной сетью. При этом сетевая часть IP-адреса виртуальной сети отличается от сетевой части адреса реальной сети. В этом случае система VMware снова конфигурируется по варианту "Host-only networking". Только теперь необходимо указать Samba-серверу, что он должен обслуживать как интерфейс с реальной сетью (или даже несколько таких интерфейсов), так и виртуальный интерфейс vmnet1, на который работает виртуальная сеть. Делается такое указание путем корректировки строки interfaces в файле /etc/smb.conf. Она должна принять следующий вид:

```
interfaces = <физические сети> <виртуальная сеть>.1/24
```

где <физические сети> — это список обслуживаемых физических сетей, а <виртуальная сеть> — это сетевая часть адреса, назначенного для виртуальной сети. Предположим для примера, что базовый компьютер имеет в реальной сети адрес 209.220.166.34, а в виртуальной сети в варианте "Host-only" ему присвоен адрес 192.168.0.1. Тогда указанная строка принимает вид:

```
interfaces = 209.220.166.34/24 192.168.0.1/24
```

или, задавая маску сети явным образом, так:

```
interfaces = 209.220.166.34/255.255.255.0 92.168.0.1/255.255.255.0
```

Подробнее об этом см. man-страницу 5 smb.conf.

Если вы не помните, какой IP-адрес присвоен виртуальному интерфейсу, дайте **Команду** /sbin/ifconfig vmnet1.

18.8.6. Доступ к дискам виртуального компьютера из ОС базового

Во всех трех рассмотренных примерах мы говорили только о том, как получить доступ к дискам базового компьютера из виртуального. Однако можно поставить вопрос и наоборот: как получить доступ к дискам виртуального компьютера из ОС базового? Очевидно, что за счет использования сетевых средств такая задача тоже легко решается.

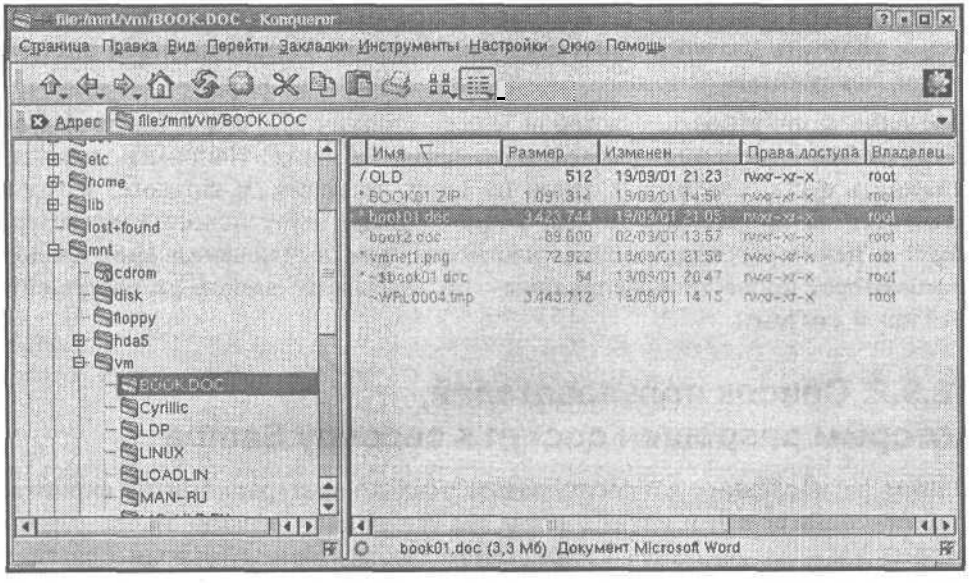


Рис. 18.11. Ресурс виртуального компьютера смонтирован в каталог /mnt/vm в ОС базового компьютера (просмотр через Konqueror)

Если на базовом компьютере установлен пакет Samba, то отдельные каталоги на дисках виртуального компьютера vmware, работающего под Windows, можно монтировать в файловой системе Linux на базовом компьютере примерно такой командой

```
[user]$ /usr/sbin/smbmount //vmware/public /mnt/vm1 -U user1
```

(Подробнее см. документацию по Samba или страницу man smbmount).

Рис. 18.11 иллюстрирует эту возможность. На нем показан каталог на диске виртуального компьютера, смонтированный в файловую систему Linux. При этом в виртуальном компьютере запущен MS Word, о чем свидетельствует наличие временных файлов, создаваемых этой программой. А сам каталог в данном случае просматривается из Linux при помощи браузера Konqueror.

18.9. Несколько дополнительных замечаний

18.9.1. Снова о предосторожностях

В начале этой главы уже было сказано, что одновременное обращение к диску двух разных операционных систем может привести к неприятностям. Но об этом придется напомнить еще раз в связи с только что рассмотренной возможностью получения доступа к дискам с помощью сетевых средств. Не стоит с помощью таких средств организовывать "кольцо", смонтировав диск Windows в один из каталогов файловой структуры Linux, затем подключить Linux-диск как сетевой диск в проводнике Windows, а потом пытаться получить доступ к этому диску из Windows. У операционной системы может "закружиться голова".

Еще одна неприятность, с которой можно столкнуться, связана с тем, что в Linux не все изменения сразу запоминаются на диске. Например, попытки сохранить файл, созданный одной из Linux-программ, в каталоге, который находится на примонтированном Windows-диске виртуального компьютера, могут в некоторых ситуациях (в частности, при выключении виртуального компьютера) привести к неприятным последствиям, вплоть до разрушения файловой системы.

18.9.2. Список пользователей, которым разрешен доступ к серверу Samba

Одним из последних вопросов, задаваемых конфигурационным скриптом vmware-config.pl в том случае, когда вы установили версию Samba-сервера фирмы VMware, является предложение ввести имена и пароли пользователей, которым будет разрешен доступ в Samba-серверу. Если вы отказались от выполнения этой процедуры на этапе конфигурирования или хо-

тите добавить новых пользователей в этот список, вы должны проделать следующее.

1. Получить права суперпользователя

   ```
   [user]$ su
   ```

2. Выполнить команду

   ```
   [root]# /usr/bin/vmware-smbpasswd vmnetl -a <username>
   ```

   где <username> — это имя пользователя, которого вы добавляете в список.

3. Выполнить то, что будет сказано в инструкциях, появляющихся на экране (программа vmware-smbpasswd является вариантом стандартной программы smbpasswd. Если вы знакомы с последней, вы можете использовать любые ее опции).

4. Покинуть бюджет пользователя root:

   ```
   [root]# exit
   ```

Если вы получите сообщение Unknown virtual interface "vmnetl", значит либо у вас не используется сервер Samba от VMware, либо не задействован вариант "Host-only" (который обслуживается этим интерфейсом).

18.9.3. Как устранить "утечку" пакетов из виртуальной сети в реальную

Все системы, поддерживающие протоколы TCP/IP, обычно обладают способностью отправлять дальше те IP-пакеты, которые предназначены не им (forwarding). Поэтому, если созданная нами виртуальная сеть подключена к реальной сети, в последней может существенно повыситься трафик. Но это происходит только в том случае, когда в системах, входящих в виртуальную сеть, не отключена функция форвардинга. По умолчанию эта функция в большинстве систем отключается, но все же стоит этот момент проверить, чтобы не перегружать реальную сеть. Причем соответствующую настройку необходимо сделать как в ОС базового компьютера, так и в ОС виртуальных компьютеров.

В ОС Linux запрет форвардинга осуществляется путем записи "0" в специальный файл /proc/sys/net/ipv4/ip_forward; это можно сделать, например, командой

```
[root] # echo 0 >/proc/sys/net/ipv4/ip_forward
```

18.9.4. О применении системы VMware

В рамках данной книги не нашли отражения ответы на многие вопросы, связанные с установкой, конфигурированием и работой системы виртуальных машин, в частности процедуры подключения к виртуальному компьютеру пе-

риферийных устройств (принтеры, модемы, ZIP-дисководы и т. д.). Однако, на мой взгляд, эти вопросы не являются актуальными для тех, кто ориентируется на переход от работы под Windows к преимущественному использованию Linux. Действительно, различные периферийные устройства, наверное, легче и проще подключить к физическому компьютеру, чем к виртуальному. После этого доступ к таким устройствам, как принтер или дисковод ZIP от Iomega, можно получить через сеть, как это было показано в *разд. 18.8*. А что касается выхода в Интернет через модем, то все средства для этого (программы доступа к сети по телефонным каналам, браузеры и системы просмотра электронной почты) имеются и в Linux, так что нет никакой необходимости организовывать такой доступ через виртуальный компьютер.

Единственная серьезная причина, по которой для пользователя Linux становится необходимым применение ОС от Microsoft и, соответственно, системы виртуальных машин, — это необходимость обмениваться файлами с приверженцами Windows, которые пока что составляют большинство пользователей компьютеров вообще. Именно для обработки файлов в форматах, разработанных фирмой Microsoft, приходится запускать VMware. Но именно эта система позволяет осуществить постепенный переход на использование Linux, т. к. позволяет, с одной стороны, продолжать использовать весь набор привычного ПО, освоенного пользователем Windows, и, с другой стороны, постепенно осваивать свободные аналоги этого ПО, разработанные для Linux.

18.9.5. Немного о быстродействии

Поскольку система VMware является эмулятором, естественно возникает мысль о том, что прикладные программы будут на ней работать медленнее, чем на физическом компьютере, однако опыт показывает, что если дополнительное замедление и есть, оно так невелико, что практически незаметно. Более того, я провел серию тестов с помощью программы Sandra SiSoft, результаты которых меня просто поразили. Некоторые тесты показывают, что быстродействие виртуального компьютера с NT 4.0 по некоторым параметрам выше, чем быстродействие базового компьютера, работающего под той же ОС непосредственно. Конечно, быстродействие существенно зависит от ресурсов базового компьютера. Я начинал осваивать систему VMware на компьютере с ОЗУ 64 Мбайт и процессором 233 МГц, где все в общем-то работало, но замедление работы ОС на виртуальной машине было очень заметно. Сейчас я использую базовый компьютер с процессором Pentium III 733 МГц и 128 Мбайт ОЗУ. Память я поделил между двумя ОС поровну и замедление в быстродействии уже незаметно. Именно объем оперативной памяти является самым критичным параметром для использования системы виртуальных машин. Судя по моему опыту, 64 Мбайт физического ОЗУ все же мало, а при 128 уже можно работать достаточно комфортно.

18.9.6. О первоисточниках

Единственным источником информации при написании этой главы (а также статей [П20.7, П20.8] *приложения)* были материалы с сайта фирмы VMware **(http://www.vmware.com).** Русскоязычных публикаций о системе VMware пока практически нет кроме двух статей автора [П20.7] и [П20.8] *(см. приложение)* (содержание которых практически совпадает с содержанием настоящей главы) и статьи [П20.6] *приложения.* Дополнительную информацию на русском языке можно найти на сайтах [П20.4, П20.5] *(см. приложение).* Если вас заинтересовала система виртуальных машин, — установите ее и экспериментируйте, она стоит того, чтобы потратить на нее время.

Приложение

Источники и ссылки на дополнительные материалы

П1. Книги, посвященные Linux

В последнее время выпущено уже огромное количество книг, посвященных операционной системе Linux. Прочитать их все я, конечно, не имел возможности. Поэтому я не буду стараться их все перечислить, укажу только несколько книг, которые, на мой взгляд, стоит приобрести начинающему пользователю Linux.

1. Уэлш М. и др. Руководство по установке и использованию системы Linux: Пер. с англ. — М.: IPLabs, Институт логики, 2000.

   Это одна из самых первых книг по Linux, выдержавшая уже несколько переизданий как в оригинале, так и в русском переводе. Она открывает серию книг Linux Documentation Project (LDP). И любому начинающему пользователю стоит ее приобрести. Хотя книга широко распространена в электронных версиях (под названием "Установка Linux и первые шаги"), но удобнее пользоваться печатным изданием.

2. Кирх О. Linux для профессионалов. Руководство администратора сети. — СПб.: Питер, 2000.

   Это еще одна книга из серии LDP. В ней рассматриваются вопросы подключения к локальным сетям и использования Linux в качестве серверной ОС. Я использовал эту книгу при подготовке главы "Выход в локальные сети".

3. Red Hat 6.2 Linux. Учебный курс. Под ред. А. Пасечника. — СПб.: Питер, 2000.

   Эта книга представляет собой перевод двух руководств: "Официального руководства по установке и конфигурированию Red Hat Linux 6.2" и "Официального руководства начинающего пользователя Red Hat Linux". Если первая из этих двух частей очень подробна и детальна, то вторая по необходимости очень конспективна. Единственное (кроме процедур установки, естественно), что в этой книге рассмотрено подробнее, чем в других книгах по Linux (из тех, что я читал), — это работа с программой linuxconf. А приобрести эту книгу стоит из-за приложения D, которое содержит перечень всех входящих в дистрибутив пакетов (на 133 страницах!). Такой перечень будет очень полезен как справочное пособие.

4. Петерсен Р. LINUX: руководство по операционной системе: Пер. с англ. С. М. Тимачева / 2-е изд. — Киев: БХВ, 1999. (В комплекте с CD-ROM).

   Эта книга описывает дистрибутив Caldera, но я очень активно использовал эту книгу при изучении Red Hat Linux. Она охватывает все основные вопросы работы в системе, и хотя изложение несколько сумбурное, но пролистать ее полезно.

5. Петцке К. Linux. От понимания к применению. — М.: ДМК, 2000.

   Очень полезная книга, широко применяемая мной при освоении Linux и подготовке своих заметок. При небольшом объеме содержит очень много полезной информации.

6. Федорчук А. Офис, графика, Web в Linux. — СПб.: БХВ-Петербург, 2001.

   Эта книга написана очень живым языком и очень легко читается. И ориентирована на ту же категорию читателей, что и та книга, которую вы сейчас держите в руках. Рекомендую если не купить ее, то хотя бы взять ее в библиотеке, чтобы пролистать на досуге.

7. Немет Э. и др. UNIX. Руководство системного администратора: Пер. с англ. С. М. Тимачева / 4-е изд. — Киев: БХВ, 1999.

   Эта книга посвящена не Linux, а UNIX вообще, но купить ее стоит, поскольку это лучшая из книг, посвященных системному администрированию UNIX (это не только мое мнение, я слышал такие отзывы от нескольких системных администраторов UNIX). Все приведенные в ней сведения могут напрямую применяться и *в* Linux-системах.

8. Келли-Бутл С. Введение в UNIX: Пер. с англ. С. Орлова. — М.: Лори, 1994.

   Это была одна из первых книг, по которым я знакомился с UNIX. В ней вы найдете дополнительные материалы по таким вопросам, как оболочки, регулярные выражения, редакторы семейства vi, система X Window и т. д.

9. Такет Д. (мл.), Барнет С. Использование Linux. Специальное издание: Пер. с англ. — М.: Издательский дом Вильяме, 1999.

П2. Несколько интернет-адресов для начала

Ниже приведены только несколько адресов, с которых вы сможете начать свои исследования сайтов Интернета, посвященных Linux.

Русскоязычные сайты

1. **http://www.linux.org.ru** — последние новости и масса ссылок, а также много документации на русском языке по Linux. Ведет сайт Максим Валянский.
2. **http://linux.ru.net** — один из самых интересных на сегодняшний день сайтов из числа русскоязычных, посвященных Linux. Ведет его Антон Фарыгин. Ежедневные новости из мира Linux, ссылки на русскоязычные ресурсы, форум, документация.
3. **http://www.opennet.ru** — самый богатый по содержанию сайт из числа русскоязычных, посвященных Linux. Ведет сайт Максим Чирков. Только жаль, что навигация по сайту несколько затруднена.
4. **http://linux.webclub.ru** — очень интересный сервер с массой документации (см. раздел "Архив") и ПО для Linux. Новости, форум, ссылки, отзывы.
5. **http://www.linuxrsp.ru** — интересный новостной сайт, документация, ссылки, форум. Ведет сайт Федор Сорекс.
6. **http://linuxnews.ru** — сайт охватывает все темы, касающиеся Linux, — и документацию, и архив программ, и новости. Ведет сайт В. Калошин.
7. **http://linuxcenter.ru** — новости, список ссылок, документация.
8. **http://linux-ve.chat.ru** — здесь вы найдете самый полный список ссылок на русскоязычные ресурсы Интернета, посвященные Linux.
9. **http://www.gnu.org.ru** • страница русской команды перевода GNU. Здесь приведены переводы некоторых публикаций и документов, которые говорят о проекте GNU и о свободном программном обеспечении вообще.
10. **http://www.kde.ru** — страница русской команды перевода проекта KDE.

Основные англоязычные сайты

1. **www.linuxhq.com** LinuxHQ — сервер, посвященный ядру Linux. Информация о разных версиях ядра, новости, обзоры, неофициальные патчи и т. п.
2. **http://www.linux.com**
3. **http://www.linux.org**
4. **http://www.linux.org.uk** — европейский Web-сервер Linux. Поддерживается Аланом Коксом (Allan Cox), одним из основных разработчиков Linux.
5. **http://www.linuxdoc.org** — основной сервер с документацией по Linux.
6. **http://freshmeat.net** — ежедневные сообщения о всех новинках программного обеспечения для Linux. Огромный архив программного обеспечения для Linux.

7. **http://www.li.org** — сервер организации Linux International.
8. **http://www.linuxstart.com**
9. **http://oreilly.linux.com** — здесь вы найдете массу документации.
10. **http://www.linuxplanet.com**
11. **http://www.kde.org**
12. **http://www.gnu.org** — сервер организации, которая создала много полезных приложений для Linux, самым известным из которых является редактор Emacs. Однако основной целью организации является создание новой свободно распространяемой операционной системы типа UNIX, подобной Linux, но называющейся GNU (GNU's Not UNIX).
13. **http://slashdot.org** — самые последние новости о компьютерных технологиях, в том числе и о Linux. Публикуются авторские статьи и письма посетителей (причем без всякой правки или цензуры).
14. **http://www.Hnuxtoday.com** - публикует длинный перечень новостей, пресс-релизов, рекламных сообщений и прочих материалов. Заглянув сюда, вы узнаете о большинстве событий в Linux-сообществе.
15. **http://www.lwn.net** — еженедельные Linux-новости. Все новости, посвященные Linux, за неделю. Информация удобно разбита по категориям: новости вообще, коммерческие, новости об усовершенствованиях в ядре Linux, новые средства разработки программ для Linux, программы для Linux, вышедшие за неделю и т. д. Если вы хотите убедиться, что Linux действительно очень быстро развивается и получает поддержку крупных (и не только) коммерческих компаний, обязательно посетите этот сайт! Новости здесь снабжаются хорошими редакторскими комментариями. Выходит по четвергам.
16. **http://lwn.net/lwn/daily** — Linux Weekly News/Daily. Новости, которые в конце недели войдут в Linux Weekly News. Обновляется ежедневно.
17. **http://www.linuxnewbie.org** — хороший сайт для новичков (и не только).
18. **http://www.seul.org** — Simple End User Linux (SEUL). Проект, поставивший целью упрощение работы с Linux для конечных пользователей.
19. **http://kt.zork.net** — на этом сайте вы найдете, в частности, Kernel Traffic — еженедельные новости о ядре Linux.
20. **http://www.linuxjournal.com** — Linux-журнал. Регулярно публикуются интересные статьи на различную тематику.
21. **http://www.linuxgazette.com** — Linux-газета будет интересна как для начинающих пользователей, так и для профессионалов. В журнале можно узнать много полезной информации, от приемов работы в редакторе Emacs до описаний приемов программирования на C++ и Java (выходит и на русском языке).

22. **http://www.linuxfocus.org** - международный некоммерческий журнал LinuxFocus (выходит и на русском языке).
23. **http://www.linuxworld.com** — хороший сайт, содержащий много информативных статей. Не забудьте заглянуть в разделы "Topical Index" и "Archive".
24. **http://www.linux-mag.com** - Linux Magazin. Очень интересный журнал. Обратите в частности внимание на раздел "Кто есть кто в мире Linux".
25. **http://www.penguinmagazine.com**

Конечно, этот список далеко не полон. Я привожу его только для того, чтобы дать вам отправные точки (или начальные ориентиры) для путешествий по безбрежному морю Интернета. А куда вы потом забредете — это уже в вашей воле.

ПЗ. К главе 1 "ОС Linux: история и дистрибутивы"

1. **http://www.li.org/linuxhistory.php** — здесь вы можете получить более подробные сведения об истории возникновения Linux.
2. **http://linux.perm.ru/doc/misc/gpl.html** — текст GPL на русском языке. Другой вариант перевода лицензии GPL, сделанный юристом Е. Тяпкиной, а также переводы еще двух лицензий, имеющих отношение к свободному ПО, вы сможете найти по следующим адресам:
   - **http://www.tecon.ru/soft/gplrus.htm**
   - **http://www.tecon.ru/soft/lgplrus.htm**
   - **http://www.tecon.ru/soft/gplrus.htm**
3. **http://www.GNU.org.ru/manifesto.html** — манифест GNU.
4. Столлман Р. GNU и движение за свободно распространяемое ПО (http://www.osp.ru/os/1999/03/09.htm), март 1999.
5. Реймонд Э. С. Собор и базар. // Открытые системы. — 1999. № 9—10 (http://www.osp.ru/os/1999/09-10/071.htm). В этой широко известной статье приведен анализ одного из успешных проектов открытой разработки - fetchmail, с позиций двух совершенно различных стилей разработки ПО.
6. Иванченко В., Панишев А. Пингвин против Империи. (**http://www.submarine.ru/print.cfm?ID=410**). Усилия многочисленных разработчиков привели к тому, что среда Linux из мрачной UNIX-подобной постепенно становится все более графической и понятной пользователю. Случилось то, чего в общем-то не ждали, — Linux начал вторжение на рынок "настольных" операционных систем.
7. В разделе "Ликбез по Linux" на сервере **www.osp.ru** имеется подборка популярных статей, из которых можно составить общее представление об этой ОС (**http://www.osp.ru/linux/teach.htm**).

8. Федорчук А. Демоны, пингвины и пользователи. Начнем с пингвинов. (**http://www.softerra.ru/freeos/11118/**).
9. Сергеев А. Авторское право: аллюзии, аналогии, антиномий. (**http://www.linuxcenter.ru/lib/ideology/pravo.phtml**)
10. X-Stranger FSF и проект GNU (**http://www.nestor.minsk.by/kg/kg01/21/kg12112.html**).
11. The Great Giveaway (**http://newscientist.com/hottopics/copyleft/copyleft.html**).

П4. К главе 2 "Инсталляция ОС Linux на компьютер с Windows"

1. Almesberger W. LILO — Generic Boot Loader for Linux, v2.1, User's Guide, 4 декабря 1998 г. (После инсталляции Linux вы найдете этот документ в файле **/usr/doc/lilo-x.y/README**).[1]
2. Brouwer A. Large Disk HOWTO, v2.0, 22 January 1999 (находится в файле **/usr/doc/HOWTO/mini/Large-Disk**). Существует перевод на русский язык, который вы можете найти на сайте **http://linuxdoc.chat.ru/obsh/large/Large-Disk-HOWTO.htm** или на **http://www.mgul.ac.ru/~t-alex/Linux/Large-Disk-HOWTO/Large-Disk-HOWTO.htm**.
3. Koehntopp K. Linux Partition HOWTO, v2.4, 3 November 1997 (находится в файле **/usr/doc/HOWTO/mini/Partition**).
4. Gjoen S., HOWTO: Multi Disk System Tuning, v0.17, 3 February 1998 (находится в файле **/usr/doc/HOWTO/Multi-Disk-HOWTO**).
5. Fawcett T. The Linux Bootdisk HOWTO, v3.3, November 1998 (находится в файле **/usr/doc/HOWTO/Bootdisk-HOWTO**).
6. Spitzer C., Rubini A. Lilo mini-Howto, v2.02, 16 August 1998 (находится в файле **/usr/doc/HOWTO/mini/LILO**).
7. Reichert B. NT OS Loader + Linux mini-HOWTO, v1.11, 2 September 1997 (находится в файле **/usr/doc/HOWTO/mini/Linux+NT-Loader**).
8. Harlan M. The Linux "Linux-DOS-Win95-OS2" mini-HOWTO, v1.3.1, 11 November 1997 (находится в файле **/usr/doc/HOWTO/mini/Loadlin+Win95**).
9. Katz Y. Linux+Win95-HOWTO, 26 October 1996 (находится в файле **/usr/doc/HOWTO/mini/Linux+Win95**). Имеется перевод на русский язык, сделанный Ильгизом Кальметьевым (**http://www.linux.org.ru:8100/books/HOWTO/Linux+Win95-HOWTO.html**).

[1] Приводимые ниже ссылки на каталоги соответствуют структуре каталогов, образовавшейся после установки дистрибутива Black Cat Linux 6.02; в других дистрибутивах расположение файлов может несколько отличаться.

10. Zanelli R. Win95 + WinNT + Linux multiboot using LILO mini-HOWTO, v1.0, 26 March 1998 (находится в файле **/usr/doc/HOWTO/mini/Multiboot-with-LILO**).

11. Fischer C. The Loadlin+Win95 mini-HOWTO, v1.4.6, 13 March 1999 (находится в файле **/usr/doc/HOWTO/mini/Linux+DOS+Win95+OS2**).

12. Gortmaker P. The Linux BootPrompt-HowTo, v1.14, 1 February 1998 (находится в файле **/usr/doc/HOWTO/BootPrompt-HOWTO**).

13. Московских А. Как поставить много операционных систем на одном компьютере (**http://www.infocity.kiev.ua/os/content/os038.phtml**).

14. Сэвилл Дж. Двойная загрузка Windows 2000 и Linux с помощью NTLoader (**http://www.osp.ru/win2000/worknt/2000/02/017.htm**).

15. Проблемы с большими жесткими дисками **(http://t37.nevod.perm.su/linux/linux/bigIDE.html**).

П5. К главе 3 "Первый запуск ОС Linux"

1. На странице Алексея Махоткина **(http://alexm.here.ru/manpages-ru/index.html**) можно найти переводы man-страниц на русский язык.

2. Gonzato C. Из DOS/Windows в Linux HOWTO, перевод Alex Ott, v1.3.2, 22 февраля 1999 (**http://linux.webclub.ru/howtorus/doswinhow/dos-win-to-linux-howto.html**). Это очень полезный материал обзорного плана, позволяющий понять и усвоить основные приемы работы в командной строке Linux (особенно, если вы имеете опыт работы в командной строке DOS).

3. Greenfield L. Linux Users Guide (**http://linux.webclub.ru/books/lug/index.html),** 1994.

4. Несколько простейших команд **UNIX (http://www.mgul.ac.ru/~t-alex/Linux/kmb.htm**)

П6. К главе 4 "Знакомство с файловой системой Linux"

1. Бах М. Дж. Архитектура операционной системы Unix. Перевод А. В. Крюкова (**http://www.citforum.ru/operating_systems/bach/contents.shtml**).

2. Программа обработки архивов tar (**http://linuxdoc.chat.ru/obsh/arhiv/tar.html**).

3. Джелли Ж. (Jean-loup Gailly). Описание приемов работы с архиватором gzip (**http://linuxdoc.chat.ru/obsh/arhiv/gzip.html**).

4. Bzip2-HOWTO **(http://www.mgul.ac.ru/~t-alex/Linux/Bzip2-HOWTO/Bzip2-HOWTO.htm**).

5. Средства сравнения файлов (**http://www.sit.kiev.ua/linux/compare.shtml**).
6. МакКензи Д. (David MacKenzie), Эггерт П. (Paul Eggert), Столлман Р. (Richard Stallman). Сравнение и объединение файлов: diff, diff3, sdiff, cmp, patch, Сентябрь 1993. Редакция 1.3, diff 2.5, patch 2.1 (**http://linux.ru.net/index.php?module=library&action=show&docid=143&part=1115).** Этот документ содержит описание GNU утилит diff, diff3 sdiff и cmp, анализирующие различия между файлами, и команд; patch, использующую результаты их работы для восстановления файлов.

П7. К главе 5 "Интерфейс командной строки"

1. Хименко В. Кто командует парадом? — рассказ о командном интерпретаторе bash **(http://www.osp.ru/pcworld/2001/01/154.htm).**
2. Строкин Г. BASH конспект (**http://isabase.philol.msu.ru/bash/bash-conspect.html**).

   На персональной странице Г.Строкина (**http://isabase.philol.msu.ru/~grg/**) вы найдете дополнительно полезные скрипты для bash.
3. Соловьев А. Программирование на shell (**http://www.nevod.ru/linux/doc/shell/, http://wgc.chem.pu.ru/educate/UNIX/Shell/unix0.htm**).
4. Интерпретатор командного языка shell, текст подготовлен НПО "КЛОТО" (**http://citforum.ru/koi/operating_systems/shell/index.shtml**).

П8. К главе 6 "Программа Midnight Commander"

1. Как ПРАВИЛЬНО работать в Midnight Commander.

   Советы, с большинством которых я не согласен (но у каждого свое мнение!) (**http://linux.uatel.net/mc.phtml**).
2. Перевод файла подсказки к программе Midnight Commander (**http://linux-ve.chat.ru/kos/mc.hlp.bz2**).

П9. К главе 7 "Графический интерфейс"

1. Николаев И. FAQ по настройке монитора в XFree86 (**http://knot.pu.ru/faq/xfaq.html**).
2. Кононенко С. Графические системы Linux с точки зрения игр и мультимедиа (**http://www.citycat.ru/linux/artic/index.html**).

3. Как научить KDE говорить по-русски? **(http://www.kde.org/international/russia/localization .html)**.

   Кроме того, можете посетить сайты **www.kde.org и www.kde.ru** для получения дополнительной информации. На **www.kde.ru** имеется описание основных особенностей и преимуществ KDE в сравнении с другими графическими средами и небольшая подборка документации на русском языке.

П10. К главе 8 "Основы администрирования системы"

1. Немет Э. и др. UNIX. Руководство системного администратора: Пер. с англ. С. М. Тимачева / 4-е изд. — Киев: БХВ, 1999.
2. Виржениус Л. (Lars Wirzenius). ОС Linux. Руководство системного администратора. Версия 0.3, август 1995 **(http://www.citforum.ru/operating_systems/linux_sys/index.shtml)**.
3. Терещенко. А. Системное администрирование **(http://www.mgul.ac.ru/~t-alex/Linux/adm/adm00.htm)**.

П11. К главе 9 "Подключение и настройка аппаратных устройств"

1. Gonzato G. Configuration HOWTO v1.2.6, 19 January 1999.

   Обновленную версию этого документа вы можете найти по адресу **http://sunsite.unc.edu/mdw/HOWTO,** а русский перевод — на сервере **linux.webclub.ru.**

2. Мичурин А. Управление консолью Linux.

   "Конечно, администрировать сервер — гораздо более сложное и полезное дело, чем настройка консоли. Но про сложные и полезные вещи уже и без меня много написано, а про консоль я что-то нигде не видел, а иногда очень хочется, чтобы что-то было цветным, и динамик не пищал как резаный" **(http://webcenter.ru/~intec/alexey/hobcon.html)**.

3. Turnbull S. Alphabet Soup: The Internationalization of Linux **(http://turnbull.sk.tsukuba.ac.jp/Tools/I18N/LJ-I18N.html)**.
4. Korpela J. A tutorial on character code issues **(http://www.hut.fi/u/jkorpela/chars.html)**.
5. Czyborra R. The ISO 8859 Alphabet Soup **(http://czyborra.com/charsets/)**.

   На том же сайте можно найти описание кириллических кодировок, включая koi8-R, материалы по UNICODE, UTF-8 **(http://czyborra.com/utf/)** и другие материалы на тему кодировок. Очень интересный и полный информации сайт.

6. Паскаль И. X Keyboard Extension (**http://www.mgul.ac.ru/~t-alex/Linux/X-Keyboard/index.htm**).
7. Bagwell C. The Linux Busmouse HOWTO, v1.91, 15 June 1998.
8. Flickenger R. Speeding up Linux Using hdparm, (06/29/20000 (**http://linux.oreillynet.com/pub/a/linux/2000/06/29/hdparm.html**).
9. Taylor G. Печать в Linux HOWTO (Linux Printing HOWTO), версия 4.5 февраль 2000, перевод Alex Ott (**http://www.linux.org.ru:8101/books/HOWTO/Printing-HOWTO.html**).
10. **На http://www.picante.com/~gtaylor/pht/printer_list.cgi** поддерживается список моделей принтеров, работающих в Linux (в "Printing HOWTO" тоже имеется довольно обширный вариант такого списка).
11. Румянцев Д. Подключение принтера (parport + kernel >.1.32) (**http://dimitr.obninsk.net**).
12. Попов В. Так он еще и печатает?! **(http://www.softerra.ru/review/oses/linux/11295/)** (опубликовано 17.07.2001).
13. LinuxPrinting.org — "глобальный" ресурс по печати в Linux (**http://www.linuxprinting.org/**).
14. Internet Resources for POSTSCRIPT & GHOSTSCRIPT.

    Страница практически не содержит оригинальных данных, зато предоставляет практически исчерпывающий список ссылок на "около-PostScript" ресурсы **(http://www.geocities.com/SiliconValley/5682/postscript.html)**.
15. Tranter J. Звук в Linux HOWTO, v1.19, 23 января 1998, перевод Alex Ott (**http://linux.webclub.ru/howtorus/sound/sound-howto.html**).

П12. К главе 10 "Установка и обновление программных пакетов"

1. Barnes D. RPM HOWTO, перевод А. Отта. Общие сведения о том, как пользовать Red Hat Package Manager (RPM) (**http://linux.webclub.ru/howtorus/rpm/rpm-howto.html**).
2. Черкашин Е. Использование RPM (**http://www.citforum.ru/operating_systems/articles/linux_rpm.shtml**).
3. Громов Ю. Ю., Татаренко С. И. Программирование на языке Си (**http://www.citforum.ru/programming/c/dir.shtml**).
4. Теория и практика программирования на С в UNIX (**http://www.mirea.ac.ru/linux/tppcu/Count.htm**).

5. Марченко А. Л. C++. Бархатный путь. Избранные главы (**http://www.citforum.ru/programming/cpp_march/index.shtml**).
6. Man-страница к make, перевод Ю. Козлова **(http://spslug.sposad.ru/doc/man/make/make.1.zip)**.
7. Черняк Д. Применение GNU make **(http://www.linux.org.ru:8101/books/make.html)**.

П13. К главе 11 "Русификация ОС"

1. Балдин Е. Cyrillic-HOWTO.

   Этот материал пока находится в стадии создания, но, думаю, что в ближайшее время (когда автор закончит работу над ним) он будет исчерпывающим источником сведений по вопросу русификации Linux (**http://www.inp.nsk.su/~baldin**).
2. Font HOWTO.

   Этот HOWTO вместе с документом, указанным в следующей ссылке, послужили основой для статьи [П13.17], а, следовательно, и для главы И настоящей книги. Исходный текст этого HOWTO можно найти на Web-странице **http://pegasus.rutgers.edu/~elflord/font_howto** или на **http://www.linuxdoc.org/HOWTO/Font-HOWTO.html**. Русский перевод — на **http://linux-ve.chat.ru/kos/font_HOWTO_ru/Font-HOWTO.html**.
3. XFree86 Font Deuglification Mini HOWTO **(http://linuxdoc.org/HOWTO/mini/FDU/index.html)**.
4. TrueType HOWTO, **http://www.moisty.org/~brion/linux/TrueType-HOWTO.html**.
5. XFree86 *4.x* **(http://www.xfree86.0rg/4.0/fonts.html).** На этой страничке вы узнаете о поддержке фонтов в XFree86 4.x.
6. TrueType Fonts in Debian mini-HOWTO, **http://www.dimensional.com/~bgiles/debian-tt.html**.
7. Страничка шрифт-сервера xfsft **(http://www.dcs.ed.ac.uk/home/jec/programs/xfsft/)** все еще содержит полезную информацию, хотя она более не поддерживается. Дело в том, что Xfsft в настоящее время интегрирован в XFree86 как модуль freetype и поэтому отдельная версия шрифт-сервера уже не актуальна. Версия 4.x XFree86 включает все функции Xfsft и много дополнительных.
8. Turner D. Glyph Hell. An introduction to glyphs, as used and defined in the FreeType engine. Version 1.0 (html version) — 14 Jan 98.
9. Этот материал я обнаружил в каталоге **/usr/doc/freetype-x.x/docs** после установки Black Cat 6.02. Его полезно прочитать, если вы хотите больше узнать о контурных шрифтах, глифах, кернинге и других понятиях и терминах, связанных со шрифтами.

10. Adobe Systems Incorporated, Adobe Glyph List version 1.2, 22 Oct 1998 (**http://partners.adobe.com/asn/developer/typeforum/glyphlist.txt**).

11. Бырганов Е. Описание кодировки для работы в среде X11 на русском языке, 28.10.1999 (**http://www.inp.nsk.su/~byrganov/publish/koi8-1/koi8-1.ru.html**).

12. Flowers J. "X Logical Font Description Conventions", Version 1.5, X Consortium Standard, X Version 11, Release 6.3 (ftp://ftp.x.org/pub/R6.4/xc/doc/hardcopy/XLFD/xlfd.PS.gz).

13. Советы по решению проблем со шрифтами в Netscape вы найдете на странице Consumer articles (**http://help.netscape.com/kb/consumer/**). Проведите на этой странице поиск по слову "font".

14. Wordperfect for Linux – Fonts and Printers **(http://www.rodsbooks.com/wpfonts/).** Страничка Рода Смита (Rod Smith), автора "Using Corel Wordperfect 8 for Linux". Здесь вы найдете информацию о поддержке шрифтов TrueType в Wordperfect.

15. Some Linux for Beginners (**http://home.c2i.net/dark/linux.html#ttf**). Здесь обсуждается использование шрифтов TrueType и можно найти ссылку на скрипт для создания файла fonts.alias.

16. В архиве выпусков журнала "Publish" (**http://www.osp.ru/publish/archive/**) можно почитать статьи на близкие темы.

17. RU.LINUX.FAQ © Составление — Станислав Корсуков, © Поддержание до сентября 1999 — Михаил Браво, mbravo@kronverk.spb.su, © Поддержание — Александр Канавин, ak@sensi.org.

18. Костромин В. Шрифты и их использование в Linux // Byte/Россия. -- 2000. № 12.

П14. К главе 12 "Программы для работы с текстом".

1. Дериев И. Новая звезда электронного офиса, Обзор возможностей и недостатков пакета StarOffice версии 5.2 **(http://www.itc.kiev.ua/article.phtml?ID=3177&pid=49)**.

2. Отставнов М. Необычайные приключения StarOffice в России **(http://www.softerra.ru/freeos/10014/print.html).**

3. Маленкович С. StarOffice — электронный офис не от Microsoft (**http://www.submarine.ru/article.cfm?ID=399**).

4. Кантер Л. Рекомендации по настройке русского StarOffice 5.2 в среде Black Cat Linux 6.2 **(http://www.blackcatlinux.com/StarOffice52).**

5. Федорчук А. О квазирусификации StarOffice. Личные впечатления (**http://linusaga.virtualave.net/02office/003russtaroffice.html**).

6. Федорчук А. Linux конторский. Об офисном инструментарии и главным образом о StarOffice 5.la (**http://kulichki.rambler.ru/~anykey/05-softnotes/013softnotes.staroffice.html**). Копии той же статьи есть на **http://www.citforum.ru/operating_systems/articles/linuxkont.shtml и http://www.Iinuxrsp.ru/artic/013softnotes.staroffice.html.**

7. Федорчук А. Сага об офисе вообще и StarOffice в особенности **(http://www.linuxrsp.ru/artic/officesaga.html).**

8. Кожекин Н. Звездный путь пакета StarOffice // Мир ПК. — 1999. № 2. (**http://www.osp.ru/pcworld/1999/02/040.htm**).

9. Петрили Н. Applixware 4.2 превращает Linux в мощную среду для настольных ПК / Computerworld Россия. 1996. № 31 (http://www.osp.ru/cw/1996/31/16.htm).

10. Николаи И. Первый офисный пакет для LinuxPPC // Computerworld Россия. — 1999. № 3 (**http://www.osp.ru/cw/1999/03/24_print.htm**).

11. Милз Э., Шварц Э. Пакет офисных приложений для корпоративных систем // Computerworld Россия. — 1998. № 24.

О пакете офисных приложений WordPerfect 8 Suite для ОС Corel Linux (http://www.osp.ru/cw/1998/24/23.htm).

12. Игнатов В. Линукс в офисе: особенности национальной работы (**http://www.diskovod.ru/arts/rus/art127.htm**).

13. Сорекс Ф. Программируем документ: введение.

Первая часть цикла статей о LaTeX — издательской системе, созданной на основе системы выдающегося математика и программиста Дональда Кнута — TeX **(http://www.softerra.ru/freeos/9322/print.html).**

14. Федорчук А. Сага об NEdit'e — лучшем текстовом редакторе всех времен и народов **(http://www.linuxrsp.ru/artic/neditsaga.html).**

15. Федорчук А. О текстовых редакторах под X Window **(http://www.linuxrsp.ru/artic/002xeditors.html, http://linusaga.virtualave.net/02office/002xeditors.html**).

16. Федорчук А. Сага о текстах: текстовых редакторах, процессорах, а также русских буквах, шрифтах и правильнописании **(http://www.linuxrsp.ru/artic/textsaga.html**).

17. Федорчук А. О консольных текстовых редакторах (**http://linusaga.virtualave.net/02office/001coneditors.html**).

18. Фомичев А. Текстовые редакторы для ОС UNIX. // Открытые системы. — 1994. № 4 (**http://www.osp.ru/os/1994/04/72_print.htm**).

19. Вагнер В. Джентельменский набор Линукс-клерка (http://ppg.ice.ru/ppg/wp).
20. Федорчук А. Lyx — текстовый процессор для открытого мира (http://www.softerra.ru/freeos/10338/print.html).
21. Игнатов В. Lyx по-русски (http://ppg.ice.ru/ppg/14076).
22. Martin Ph. LinuxDoc+Emacs+Ispell HOWTO, v0.4, 27 февраля 1998, перевод Alex Ott (http://www.mgul.ac.ru/~t-alex/Linux/LinuxDoc-Emacs-Ispell-HOWTO/LinuxDoc+Emacs+Ispell-HOWTO.htm, http://www.linux.org.ru:8101/books/HOWTO/LinuxDoc+Emacs+Ispell-HOWTO.html).
23. Zawodny J. D. Emacs для начинающих HOWTO, v1.7, 14 октября 1999, перевод Alex Ott.

    Этот документ описывает редактор Emacs для пользователей Linux (http://www.mgul.ac.ru/~t-alex/Linux/Emacs-HOWTO/Emacs-Beginner-HOWTO.htm).
24. Ассоциация пользователей кириллического TEX'a (http://www.cemi.rssi.ru/cyrtug/koi/home.htm).
25. Ливотов В. Программа "Russian Anywhere" (http://www.livotov.org/software/re).
26. Англо-русский словарь Мюллера для Линукс (http://www.chat.ru/~mueller_dic).
27. Анисимов Д. Программа SLOWO (http://www.gambit.msk.su/~wolf/dic/index.html).
28. Строкин Г. Программа mu (http://isabase.philol.msu.ru/mu-online/).
29. Мурашко И. Оболочка под словари Polyglossum II для KDE (http://kdictionary.chat.ru/).
30. Прокудин А. Пингвин в клеточку. Обзор табличных процессоров для Линукс (http://www.mycomp.com.ua/article.php?id=603).
31. Вагнер В. Табличные процессоры (http://ppg.ice.ru/ppg/ppg_spreadsheet).

П15. К главе 13 "Выход в локальные сети"

1. The Linux Networking Overview HOWTO, Вер. 0.2, 10 июля 1998.

   Цель этого документа состоит в том, чтобы дать краткий обзор возможностей работы с сетями операционной системы Linux и обеспечить указания по поиску дальнейшей информации и подробностей реализации (http://www.linux.org.ru:8100/books/HOWTO/Networking-Overview-HOWTO.html).

2. Доусон Т. (Terry Dawson), Рубини А.: Перевод — Егор Дуда. Сетевая поддержка в Линуксе, Linux NET-3-HOWTO, версия 1.4, август 1998 (**http://linux.webclub.ru/howtorus/net3/net3-howto.html**).

3. Кирх О. Руководство администратора сети в ОС Linux (Network Administrator Guide), 1992—1994 (**http://www.linux.org.ru:8101/books/LDP/nag.html**).

4. Колесниченко Д. Более подробно о настройке сети.

   Статья о том, как настроить основные компоненты сети при помощи свободно распространяемых программ, входящих в Linux. Настройка сетевой платы и шлюза, кеширующего прокси squid, системы доменных имен BIND. И, конечно, Firewall, используя ipchains (**http://www.softerra.ru/freeos/11229/print.html**).

5. Терещенко А. Настройка сетевого соединения **(http://www.mgul.ac.ru/~t-alex/Linux/tcp/tcp00.htm**).

6. Пьянзин К. Сетевая файловая система UNIX.

   Основные особенности работы файловой системы NFS на платформе UNIX (**http://www.osp.ru/lan/2000/01/049.htm**).

7. Nicolai Langfeldt, NFS-HOWTO, вер. 1.0, 1 октября 1999, перевод А. Отта.

   Как установить и настроить клиент и сервер NFS (**http://www.linux.org.ru:8101/books/HOWTO/NFS-HOWTO.html, http://linux.webclub.ru/howtorus/nfs/nfs-howto.html**).

8. Файловые средства NT в Linux.

   В этой статье рассказывается о том, как организовать взаимодействие между NT и Линукс на уровне файловых систем с помощью Samba и NFS (**http://www.mgul.ac.ru/~t-alex/Linux/fs/index.htm**).

9. Wood D. SMB HOWTO, перевод А. Отта. Версия 1.1 (март 1999).

   Этот документ описывает использование в Linux протокола Server Message Block (SMB), который иногда также называется протоколом Session Message Block (SMB), протоколом NetBIOS или протоколом LanManager (**http://www.linux.org.ru:8101/books/HOWTO/SMB-HOWTO.html**).

10. Пьюри Д. Создание общих каталогов с помощью Samba **(http://www.osp.ru/win2000/workn t/2001/05/543.htm).**

11. Басин И. Samba за 5 минут.

    Установка и настройка SMB-сервисов под Unix **(http://www.fima.net/samba_ru-new.html**).

12. Лушня Ю. Подключаемся бесплатно к Windows.

    В статье описывается настройка использования разделяемых ресуров Windows NT/2000 при помощи сервера Samba, установленного на Linux (**http://www.softerra.ru/freeos/9059/print.html**).

13. Лушня Ю. Файловый сервер под Samba? Без проблем.

    Как пишет автор "в этой статье я попробую ввести начинающих пользователей Linux в процесс установки своего собственного файлового сервера под управлением Samba, а так же попытаюсь остановиться на некоторых аспектах файла smb.config" (**http://www.citycat.ru/linux/artic/sambal.html**).

14. Лушня Ю. Samba в подробностях (**http://www.linuxrsp.ru/artic/samba_podrobno.html**).

15. Минаси М. Подключение рабочих станций Linux к Windows 2000 Server.

    Рабочие станции Linux довольно часто включают в сеть Windows 2000 или Windows NT, поскольку Linux предоставляет недорогие средства реализации интернет-служб (серверов DNS, почтовых серверов и Web-серверов) на маломощном оборудовании (**http://www.infocity.kiev.ua/os/content/os046.phtml**).

16. Пьянзин К. Самба — это не только танец.

    ПО Samba на платформе UNIX способно потеснить Windows NT как сервер файлов и печати (**http://www.osp.ru/lan/2000/03/049.htm**).

17. Thorpe K. IPX в Linux HOWTO, версия 2.3, май 1998.

    Этот документ описывает как получить, установить и настроить различные утилиты для операционной системы Linux, которые используют поддержку протокола IPX в ядре Linux (**http://linux.webclub.ru/howtorus/ipx/ipx-howto.html**, **http://www.mgul.ac.ru/~t-alex/Linux/IPX-HOWTO/IPX-HOWTO.htm**, **http://www.linux.org.ru:8101/books/HOWTO/IPX-HOWTO.html**).

18. Шестак Д. Всё об FTP (**http://www.infocity.kiev.ua/inet/content/inet035.phtml**).

    Что такое FTP и как с ним работать.

П16. К главе 14 "Интернет и электронная почта"

Перед началом настройки соединения с Интернетом может оказаться полезным (или даже необходимым) почитать немного о модемах и их настройке. В таком случае можно посмотреть статью "Модемы (установка, наладка, команды, режимы)" (**http://www.citforum.ru/hardware/modem_tut/index.shtml**),

серию статей о модемах на iXBT **(http://www.ixbt.com/communication.shtml),** в особенности — статью "Как работают модемы?" **(http://www.ixbt.com/comm/mdmabout1.html).**

Далее, пожалуй, надо прочитать следующие материалы общего плана:

1. Доусон Т. Рубини А. Сетевая поддержка в Линуксе, Linux NET-3-HOWTO **(http://www.phtd.tpu.edu.ru:8101/~ott/linux/howto-rus/NET-3-HOWTO.html).**
2. PPP-HOWTO **(http://www.mgul.ac.ru/~t-alex/Linux/PPP-HOWTO/index.htm** или **http://www.linux.org.ru:8100/books/HOWTO/PPP-HOWTO.html).** Версия v3.0, 31 марта 1997.

   Этот документ рассказывает как подключить ваш Linux PC к серверу PPP, как использовать PPP для того, чтобы связать две LAN вместе и предлагает один из методов настройки вашего компьютера с Linux в качестве сервера PPP. Документ также предлагает помощь по отладке неработающих PPP-соединений.
3. Kvaleberg E. ISP-Hookup-HOWTO **(http://www.mgul.ac.ru/~t-alex/Linux/ISP-Hookup-HOWTO/ISP-Hookup-HOWTO.htm),** v1.26, 5 March 1998.

   В этом документе рассказывается как использовать Linux для подключения к интернет-провайдеру через модем. Также даны основы работы по установлению связи по телефонной линии и настройке IP, электронной почты, новостей.
4. Michael Strates "ISP-Connectivity-mini-HOWTO" **(http://www.linux.org.ru:8100/books/HOWTO/ISP-Connectivity-HOWTO.html).** Русский перевод И. Кальметьева.

   Этот документ рассказывает как настроить PPP, подсоединиться к вашему ISP, настроить почту и новости, получить постоянный IP-адрес (если это возможно), получить доменное имя и настроить работоспособную систему менее, чем за 30 минут.

После этого можно переходить к реальным действиям по настройке, руководствуясь одним из следующих материалов:

1. Настройка сетевого соединения **(http://www.mgul.ac.ru/~t-alex/Linux/tcp/tcp00.htm).**

   Очень подробный материал о том, как вручную настроить соединение с провайдером Интернет по модему.
2. Сысоев И. Настройка pppd **(http://www.nitek.ru/~igor/pppd/).**

   Это самое подробное описание настройки ppp. Хотя в статье речь идет о FreeBSD, описание полностью подходит для Linux.
3. Лецинский О. Как обустроить остров пингвина **(http://www.hardnsoft.ru/magazine.php?issue=74&article=79).**

Статья дает общий обзор того, как настроить выход в Интернет, какие программы использовать.

4. Коржов В. Как подключить Linux к Internet **(http://www.citforum.ru/operating_systems/articles/linuxinet.shtml)**.
5. Костарев А. ОС Linux как мост (gateway) между локальной сетью и Internet **(http://www.linux.org.ru/books/gateway/)**.

   В данном документе рассматривается вариант подключения локальной сети через один серийный интерфейс к одному IP-провайдеру — наиболее часто встречающийся вариант подключения к Интернету.
6. Albarral M. T. Подключение к Internet **(http://www.ods.com.ua/koi/unix/linux2inet.html)**.

   Пошаговые инструкции.
7. Мартынов А. Ваш Linux-шлюз в Internet **(http://www.bytemag.ru/Article.asp?id=172)**.

   Очень интересная статья для тех, кто настраивает выход из локальной сети в Интернет через Линукс-шлюз.
8. Казанов Е. Как я устанавливал dial-up-соединение с Windows NT **(http://linux.webclub.ru/dialup/ppp_set.html)**.

   Здесь вы найдете объяснение того, почему возникают затруднения при соединении с провайдером, использующим ПО от Microsoft.
9. Настройка kppp **(http://linux.webclub.ru/dialup/kppp.html)**. Заметка о настройке модемного соединения.
10. Как соединяться с провайдером для выхода в Internet через модем, по протоколу PPP? **(http://t37.nevod.perm.su/linux/debian/2.1/config.html#P3Q5)**.
11. Хлутчин С. Настройка PPP и почты под Linux **(http://www.linuxrsp.ru/artic/ppp-conf.html)**.
12. Хлутчин С. Linux и точки **(http://www.softerra.ru/freeos/9510/print.html)**. Как классическим способом подключиться к сети, используя протокол PPP. Подробное описание конфигурационных файлов и других настроек Linux. Опубликовано: 18.05.2001.
13. Орлов А. Электронная почта: итоги тридцатилетия **(http://www.pl-computers.ru/print.cfm?ID=613)**.

    Электронная почта, пожалуй, самый популярный сервис Интернета. Можно найти многих пользователей, имеющих доступ только к ней и никак не работающих с WWW, но отнюдь не сразу отыщется человек, имеющий дело с Сетью, но так ни разу и не воспользовавшийся средствами пересылки писем. И наверняка, у вас хоть раз да возникал вопрос: а как, собственно, вся эта система работает?

14. Кошелев А. Интернет-почта (**http://www.compress.ru/Temp/1136/index.htm**).

    Цель этой статьи — дать представление о том, как работает система электронной почты в Интернете, и описать несколько конкретных программ, которые, собственно, ее и реализуют. Вначале речь идет об электронной почте вообще и о том, как она работает. Затем рассмотрены три программы, работающие с протоколом SMTP (Send Mail Transfer Protocol): SendMail, QMail и PostFix.

15. Кошелев А. Интернет-почта сегодняшнего дня **(http://www.compress.ru/Temp/1782/index.htm)**.

    В предыдущей статье рассмотрена работа почтовых серверов. В этой статье рассказывается о почтовых клиентах для UNIX.

16. Храмцов А. Средства просмотра WWW-страниц (**http://www.citforum.ru/internet/articles/art_3.shtml**).

    Приведено краткое описание Lynx, Netscape и некоторых других браузеров.

17. Комлин А. "Безопасный" Netscape Communicator 4.7 (**http://www.infocity.kiev.ua/hack/content/hack009.phtml**).

18. Федорчук А. Лики Konqueror'a (**http://onix.nm.ru/utils_konqueror/art00.html**).

    В Konqueror разработчикам удалось многократно усовершенствовать достоинства своего предшественника. В том числе и главное — степень интеграции со средой. Изжив при этом практически все недостатки. Итогом чему явилось появление замечательного программного продукта, не имеющего аналогов в мировой практике.

П17. К главе 15 "Обитание в среде KDE"

1. На русской версии сайта KDE вы найдете документацию по KDE на русском языке. Там есть руководство по компиляции, русификации и ссылки. Там же ищите список зеркал, откуда можно скачать исходники **(http://www.kde.org/international/russia/index.html).**

2. Сайт **http://www.kde.ru/** целиком посвящен системе KDE. Новости, описание возможностей, скриншоты, форум. На сервере доступна документация по CVSUP, работа со шрифтами в Qt, статьи по русификации и установке KDE.

3. Страница русской версии KDE2 расположена по адресу **http://kde2.newmail.ru/index_rus.html**.

4. Руководство пользователя KDE **(http://www.infocity.kiev.ua/os/content/os074.phtml)**.

    Хотя эта книга пока еще не закончена и многое в ней должно быть переписано заново для соответствия с новыми версиями KDE, но и этот, первоначальный вариант, будет полезен для разрешения некоторых вопросов, возникающих у нового пользователя KDE.

5. Документация по KDE имеется на linux.org.ru **(http://www.linux.org.ru/books/kde/)**.
6. Федорчук А. KDE во втором издании **(http://onix.nm.ru/sh_kde2/artOO.html).**
7. Федорчук А. Сага о KDE Втором **(http://www.linuxrsp.ru/artic/kde2/index.html)**.
8. Федорчук А. О KDE2 beta Что новенького? **(http://www.linuxrsp.ru/artic/kde2beta.html).**
9. KDE — интегрированная рабочая среда Linux **(http://www.mgul.ac.ru/~t-alex/Linux/KDE/index.htm)**. Подборка документов на русском языке по отдельным компонентам KDE.

П18. К главе 16 "Обратная сторона файловой системы"

1. **Best S. (sbest@us.ibm.com),** IBM, JFS overview, January 2000 **(http://www-106.ibm.com/developerworks/library/jfs.html)**.
2. Petreley N. Reiserfs or ext3: Which journaling filesystem is right for you? // LinyxWorld. — 2001. Nov 20 **(http://www.idg.net/go.cgi?id=604534)**.
3. Galli R., gallir@uib.es, Dept. de Matematiques i Informatica, Universitat de les Illes Balears. Journal File Systems in Linux **(http://bulmalug.net/impresion.phtml?nIdNoticia=1154)**.

П19. К главе 17 "Обновление ядра"

1. Ward B. Linux Kernel HOWTO **(http://linux.webclub.ru/howtorus/kernelhowto/kernel-howto.html** или **http://www.mgul.ac.ru/~t-aIex/Linux/Kernel-HOWTO/Kernel-HOWTO.htm)**. Версия 1.0, 5 июня 1999. Это детальное руководство по настройке ядра, его компиляции, обновлениям и разрешению проблем на системах, построенных на базе ix86.
2. Костромин В. Семь шагов к новому ядру **(http://linux-ve.chat.ru/papers/kernel/newkernel-install.html)**.
3. Configure.help — перевод на русский язык подсказок, которые выдаются на этапе конфигурирования нового ядра (перед компиляцией). Соответствует версии 2.0.x, но этот файл можно использовать и для более поздних версий, поскольку в основном вопросы, задаваемые при конфигурации, одинаковы, только новые появляются **(http://nevod.perm.su/service/linux/doc/kernel/Configure.help)**.
4. Колесниченко Д. Конфигурирование ядра linux и повышение его производительности **(http://www.softerra.ru/freeos/13826/)**.

5. Лушня Ю. Инсталляция Linux-Kernel 2.4 на Redhat 7 (http://www.linuxrsp.ru/artic/install2.4-redhat7.html).

6. Лушня Ю. Linux-Kernel 2.4.O. Хорошо или не очень?

   Статья о достоинствах и недостатках нового, 2.4.0, ядра, о том "что хорошо" и "что плохо", а так же немного истории. (http://www.linuxrsp.ru/artic/kernel-2.4.0.html).

7. Дрейган Р. 2.4 — новое ядро Linux. // PC Magazin. — 2001. № 6.

8. О достоинствах нового ядра версии 2.4 (http://www.pcmag.ru/news.asp?ID=921).

9. Федорчук А. Путешествие к центру ядра.

   Рассуждения о том, зачем может понадобиться пересобирать ядро и способах проведения этой операции (http://www.softerra.ru/freeos/12060/print.html).

10. Федорчук А. Гигагерцевый Linux.

    Как ведут себя процессоры большой тактовой частоты под под Linux? На сколько ускоряется работа на специально оптимизированном ядре. Какая выгода при работе с большими графическими файлами в GIMP (http://www.softerra.ru/freeos/10498/print.html).

11. Федорчук А. Атлоново ядро.

    Статья о сборке ядра Linux специально оптимизированного под процессоры фирмы AMD. В частности Athlon (http://www.softerra.ru/freeos/12340/print.html).

12. Pomerantz O. Энциклопедия разработчика модулей ядра Linux (Linux Kernel Module Programming Guide). Перевод на русский: Паутов Алексей (http://linux.ru.net/index.php?module=library&action=show&docid=178&part=1510).

13. Шалунов С. ОС Hurd — разработка FSF на основе микроядра // Открытые Системы. — 1997. № 3 (http://www.osp.ru/os/1997/03/22.htm).

П20. К главе 18 "Виртуальные машины VMware"

1. Иванченко В., Панишев А. Пингвин против империи (http://www.submarine.ru/print.cfm?ID=410).

2. Костромин В. Linux вместе с Windows // Открытые системы. — 2001. № 3 (http://www.osp.ru/os/2001/03/024.htm).

3. Choong Ng. VMware Express 2.0 and Win4Lin 2.0: A Comparison Review (http://www.linuxjournal.com/articles/linux_review/0036.html).

4. Костромин В. Система виртуальных машин фирмы VMware (**http://linux-ve.chat.ru/book/vmware1.htm**).
5. Ерижоков А. Использование VMware 2.0 (**http://dhls.agava.ru/vmware.html**).
6. Отставнов М. СофтГерра или Сбылась мечта шизофреника **(http://www.computerra.ru/ofiline/2000/344/2556).**
7. Костромин В. Две системы на одном компьютере // Открытые системы. — 2001. № 7—8 **(http://www.osp.ru/os/2001/07-08/023.htm).**
8. Костромин В. Виртуальный компьютер: обмен данными с реальным миром. Открытые системы. — 2001. № 11 **(http://www.osp.ru/os/2001/ll/018.htm).**

Предметный указатель

X

Z

А

Б

В

Г

Д

З

И

К

Л

М

Н

О

П

Р

С

Т

У

Ф

Х

Ц

Ш

Э

Я

Книги издательства "БХВ-Петербург" в продаже:

Серия "В подлиннике"

Андреев А. и др. MS Windows XP: Home Edition и Professional	848 с.
Андреев А. и др. Windows 2000 Professional. Русская версия	700 с.
Андреев А. и др. Microsoft Windows 2000 Server. Русская версия	960 с.
Андреев А. и др. Новые технологии Windows 2000	576 с.
Андреев А. и др. Microsoft Windows 2000 Server и Professional. Русские версии	1056 с.
Ахаян Р. Macromedia ColdFusion	672 с.
Беленький Ю., Власенко С. Word 2000	992 с.
Браун М. HTML 3.2 (с компакт-диском)	1040 с.
Вебер Дж. Технология Java (с компакт-диском)	1104 с.
Власенко С. Компакт-диск с примерами к книгам серии "В подлиннике": "MS Office XP в целом", "MS Access 2002", "MS Word 2002", "MS Excel 2002"	32 с.
Власенко С. Microsoft Word 2002	992 с.
Гофман В. Хомоненко А. Delphi 6	1152 с.
Долженков В. MS Excel 2000	1088 с.
Долженков В. MS Excel 2002	1072 с
Закер К. Компьютерные сети. Модернизация и поиск неисправностей	1008 с.
Колесниченко О., Шишигин И. Аппаратные средства PC, 4-е издание	1024 с.
Мамаев Е. MS SQL Server 2000	1280 с.
Матросов А. и др. HTML 4.0	672 с.
Михеева В., Харитонова И. Microsoft Access 2000	1088 с.
Михеева В., Харитонова И. Microsoft Access 2002	1040 с.
Новиков Ф., Яценко А. Microsoft Office 2000 в целом	728 с.
Новиков Ф., Яценко А. Microsoft Office XP в целом	928 с.
Нортон П. Windows 98	592 с.
Ноутон П., Шилдт Г. Java 2	1072 с.
Пауэлл Т. Web-дизайн	1024 с.
Персон Р. Word 97	1120 с.
Пилгрим А. Персональный компьютер: модернизация и ремонт. Книга 2	528 с.
Питц М., Кирк Ч. XML	736 с.
Пономаренко С. Adobe Illustrator 9.0	608 с.
Пономаренко С. Adobe Photoshop 6.0	832 с.
Пономаренко С. CorelDRAW 9	576 с.
Пономаренко С. Macromedia FreeHand 9	432 с.
Русеев С. WAP: технология и приложения	432 с.
Секунов Н. Обработка звука на PC (с дискетой)	1248 с.
Сузи Р. Python (с компакт-диском)	768 с.
Тайц А., Тайц А. Adobe PageMaker 7.0	784 с.
Тайц А. М., Тайц А. А. Adobe InDesign	704 с.

Тайц А. М., Тайц А. А. CorelDRAW 10: все программы пакета	1136 с.
Тайц А. М., Тайц А. А. CorelDRAW 9: все программы пакета	1136 с.
Тихомиров Ю. Microsoft SQL Server 7.0	720 с.
Уильяме Э. и др. Active Server Pages (с компакт-диском)	672 с.
Усаров Г. Microsoft Outlook 2002	656 с.
Ханкт Ш. Эффекты CorelDRAW (с компакт-диском)	704 с.
CD-ROM с примерами к книгам серии "В подлиннике": "Access 2000", "Excel 2000", "Word 2000", "Office 2000 в целом"	

Серия "Мастер"

CD-ROM с примерами к книгам "Ресурсы MS Windows NT Server 4.0" и "Сетевые средства Windows NT Server 4"	
Microsoft Press. Электронная коммерция. B2B-программирование (с компакт-диском)	368 с.
Microsoft Press. Visual Basic 6.0	992 с.
Microsoft Press. Ресурсы MS Windows NT Server 4.0	752 с.
Айзеке С. Dynamic HTML (с компакт-диском)	496 с.
Анин Б. Защита компьютерной информации	384 с.
Асбари С. Корпоративные решения на базе Linux	496 с.
Березин С. Факс-модемы: выбор, подключение, выход в Интернет	256 с.
Березин С. Факсимильная связь в Windows	250 с.
Борн Г. Реестр Windows 98 (с дискетой)	496 с.
Бухвалов А. и др. Финансовые вычисления для профессионалов	320 с.
Валиков А. Технология XSLT	432 с.
Габбасов Ю. Internet 2000	448 с.
Гарбар П. Novell GroupWise 5.5: система электронной почты и коллективной работы	480 с.
Гарнаев А. Microsoft Excel 2000: разработка приложений	576 с.
Гарнаев А. Excel, VBA, Internet в экономике и финансах	816 с.
Гарнаев А. Гарнаев С. Web-программирование на Java и JavaScript	1040 с.
Гордеев О. Программирование звука в Windows (с дискетой)	384 с.
Гофман В., Хомоненко А. Работа с базами данных в Delphi	656 с.
Дарахвелидзе П. и др. Программирование в Delphi 5 (с дискетой)	784 с.
Дронов В. JavaScript в Web-дизайне	880 с.
Дубина А. и др. MS Excel в электронике и электротехнике	304 с.
Дубина А. Машиностроительные расчеты в среде Excel 97/2000 (с дискетой)	416 с.
Дунаев С. Технологии Интернет-программирования	480 с.
Жарков С. Shareware: профессиональная разработка и продвижение программ	320 с.
Зима В. и др. Безопасность глобальных сетевых технологий	320 с.
Киммел П. Borland C++ 5	976 с.
Кокорева О. Реестр Windows 2000	352 с.
Костарев А. PHP в Web-дизайне	592 с.
Краснов М. DirectX. Графика в проектах Delphi (с компакт-диском)	416 с.

Краснов М. Open GL в проектах Delphi (с дискетой)	352 с.
Кубенский А. Создание и обработка структур данных в примерах на Java	336 с.
Кулагин Б. 3ds max 4: от объекта до анимации	448 с.
Купенштейн В. MS Office и Project в управлении и делопроизводстве	400 с.
Куприянов М. и др. Коммуникационные контроллеры фирмы Motorola	560 с.
Лавров С. Программирование. Математические основы, средства, теория	304 с.
Лукацкий А. Обнаружение атак	624 с.
Матросов А. Maple 6. Решение задач высшей математики и механики	528 с.
Медведев Е. Трусова В. "Живая" музыка на PC (с дискетой)	720 с.
Мешков А., Тихомиров Ю. Visual C++ и MFC, 2-е издание (с дискетой)	1040 с.
Миронов Д. Создание Web-страниц в MS Office 2000	320 с.
Мещеряков Е., Хомоненко А. Публикация баз данных в Интернете	560 с.
Михеева В., Харитонова И. Microsoft Access 2000: разработка приложений	832 с.
Новиков Ф. и др. Microsoft Office 2000: разработка приложений	680 с.
Нортон П. Разработка приложений в Access 97 (с компакт-диском)	656 с.
Одинцов И. Профессиональное программирование. Системный подход	512 с.
Олифер В., Олифер Н. Новые технологии и оборудование IP-сетей	512 с.
Подольский С. и др. Разработка интернет-приложений в Delphi (с дискетой)	432 с.
Полещук Н. Visual LISP и секреты адаптации AutoCAD	576 с.
Понамарев В. COM и ActiveX в Delphi	320 с.
Пономаренко С. Adobe InDesign: дизайн и верстка	544 с.
Попов А. Командные файлы и сценарии Windows Scripting Host	320 с.
Приписнов Д. Моделирование в 3D Studio MAX 3.0 (с компакт-диском)	352 с.
Роббинс Дж. Отладка приложений	512с.
Рудометов В., Рудометов Е. PC: настройка, оптимизация и разгон, 2-е издание	336 с.
Русеев Д. Технологии беспроводного доступа. Справочник	352 с.
Соколенко П. Програмирование SVGA-графики для IBM	432 с.
Тайц А. Каталог Photoshop Plug-Ins	464 с.
Тихомиров Ю. MS SQL Server 2000: разработка приложений	368 с.
Тихомиров Ю. SQL Server 7.0: разработка приложений	370 с.
Тихомиров Ю. Программирование трехмерной графики в Visual C++ (с дискетой)	256 с.
Трельсен Э. Модель COM и библиотека ATL 3.0 (с дискетой)	928 с.
Федоров А., Елманова Н. ADO в Delphi (с компакт-диском)	816 с.
Федорчук А. Офис, графика, Web в Linux	416 с.
Чекмарев А. Windows 2000 Active Directory	400 с.
Чекмарев А. Средства проектирования на Java (с компакт-диском)	400 с.
Шапошников И. Web-сайт своими руками	224 с.
Шапошников И. Интернет-программирование	224 с.
Шапошников И. Справочник Web-мастера. XML	304 с.
Шилдт Г. Теория и практика C++	416 с.
Яцюк О., Романычева Э. Компьютерные технологии в дизайне. Логотипы, упаковка, буклеты (с компакт-диском)	464 с.

CD-ROM с примерами к книгам серии "Мастер": "Office 2000", "Excel 2000", "Access 2000". Разработка приложений

Серия "Изучаем вместе с BHV"

Березин С. Internet у вас дома, 2-е издание	752 с.
Тайц А. Adobe Photoshop 5.0 (с дискетой)	448 с.

Серия "Самоучитель"

Ананьев А., Федоров А. Самоучитель Visual Basic 6.0	624 с.
Бекаревич Ю., Пушкина Н. Самоучитель Microsoft Access 2000	480 с.
Васильев В. Основы работы на ПК	448 с.
Гарнаев А. Самоучитель VBA	512 с.
Герасевич В. Самоучитель. Компьютер для врача	640 с.
Дмитриева М. Самоучитель JavaScript	512 с.
Долженков В. Самоучитель Excel 2000 (с дискетой)	368 с.
Исагулиев К. Macromedia Dreamweaver 3	432 с.
Исагулиев К. Macromedia Dreamweaver 4	560 с.
Исагулиев К. Macromedia Flash 5	368 с.
Кетков Ю. Кетков А. Практика программирования: Бейсик, Си, Паскаль (с дискетой)	480 с.
Кирьянов Д. Самоучитель Adobe Premiere 6.0	432 с.
Кирьянов Д. Самоучитель MathCAD 2001	544 с.
Коркин И. Самоучитель Microsoft Internet Explorer 6.0	288 с.
Котеров Д. Самоучитель PHP 4	576 с.
Культин Н. Программирование на Object Pascal в Delphi 6 (с дискетой)	528 с.
Культин Н. Самоучитель. Программирование в Turbo Pascal 7.0 и Delphi, 2-е издание (с дискетой)	416 с.
Леоненков А. Самоучитель UML	304 с.
Матросов А., Чаунин М. Самоучитель Perl	432 с.
Омельченко Л., Федоров А. Самоучитель Microsoft FrontPage 2002	576 с.
Омельченко Л., Федоров А. Самоучитель Windows 2000 Professional	528 с.
Омельченко Л., Федоров А. Самоучитель Windows Millennium	464 с.
Пекарев Л. Самоучитель 3D Studio MAX 4.0	370 с.
Полещук Н. Самоучитель AutoCad 2000 и Visual LISP, 2-е издание	672 с.
Полещук Н. Самоучитель AutoCad 2002	608 с.
Понамарев В. Самоучитель Kylix	416 с.
Секунов Н. Самоучитель Visual C++ 6 (с дискетой)	960 с.
Секунов Н. Самоучитель C#	576 с.
Сироткин С. Самоучитель WML и WMLScript	240 с.
Тайц А. М., Тайц А. А. Самоучитель Adobe Photoshop 6 (с дискетой)	608 с.
Тайц А. М., Тайц А. А. Самоучитель CorelDRAW 10	640 с.
Тихомиров Ю. Самоучитель MFC (с дискетой)	640 с.
Усаров Г. Самоучитель Microsoft Outlook 2000	336 с.
Хабибуллин И. Самоучитель Java	464 с.
Хомоненко А. Самоучитель Microsoft Word 2000	688 с.

Хомоненко А. Самоучитель Microsoft Word 2002	624 с.
Шапошников И. Интернет. Быстрый старт	272 с.
Шапошников И. Самоучитель HTML 4	288 с.
Шилдт Г. Самоучитель C++, 3-е издание (с дискетой)	512 с.

Серия "Компьютер и творчество"

Деревских В. Музыка на PC своими руками	352 с.
Дунаев В. Сам себе Web-дизайнер	512 с.
Дунаев В. Сам себе Web-мастер	288 с.
Людиновсков С. Музыкальный видеоклип своими руками	320 с.
Петелин Р., Петелин Ю. Аранжировка музыки на PC	272 с.
Петелин Р., Петелин Ю. Звуковая студия в PC	256 с.
Петелин Р., Петелин Ю. Музыка на PC. Cakewalk Pro Audio 9. Секреты мастерства	420 с.
Петелин Р., Петелин Ю. Музыка на PC. Cakewalk. "Примочки" и плагины	272 с.
Петелин Р., Петелин Ю. Музыкальный компьютер. Секреты мастерства	608 с.
Петелин Р., Петелин Ю. Персональный оркестр в PC	240 с.

Серия "Техника в вашем доме"

Андрианов В. Охранные устройства для дома и офиса	304 с.
Андрианов В. Средства мобильной связи	256 с.
Андрианов В. Сотовые, пейджинговые и спутниковые средств связи	400 с.
Левченко В. Спутниковое телевидение	288 с.
Миклашевский Н. Чистая вода. Бытовые фильтры и системы очистки воды	238 с.
Пешков А. Современные фотоаппараты	224 с.
Пушков А. Домашний кинотеатр на ПК	256 с.
Скоробогатов Н. Современные стиральные машины и моющие средства, 2-е издание	240 с.

Серия "Учебное пособие"

Бекаревич Ю. Access 2000 за 20 занятий	512 с.
Бенькович Е. Практическое моделирование динамических систем (с компакт-диском)	464 с.
Гомоюнов К. Транзисторные цепи	240 с.
Васильева В. Персональный компьютер. Быстрый старт	480 с.
Дорот В. Толковый словарь современной компьютерной лексики, 2-е издание	512 с.
Культин Н. C/C++ в задачах и примерах	288 с.
Культин Н. Turbo Pascal в задачах и примерах	256 с.
Порев В. Компьютерная графика	432 с.
Робачевский Г. Операционная система Unix	528 с.
Сафронов И. Бейсик в задачах и примерах	224 с.
Солонина А. и др. Алгоритмы и процессоры цифровой обработки сигналов	464 с.
Солонина А. и др. Цифровые процессоры обработки сигналов фирмы MOTOROLA	512 с.

Угрюмов Е. Цифровая схемотехника 528 с.
Шелест В. Программирование 592 с.

Серия "Знакомьтесь"

Надеждин Н. Карманные компьютеры 304 с.
Надеждин Н. Портативные компьютеры 288 с.

Серия "Быстрый старт"

Васильева В. Персональный компьютер. Быстрый старт 480 с.
Гофман В., Хомоненко А. Delphi: быстрый старт 288 с.

Внесерийные книги

Андрианов В. Автомобильные охранные системы 272 с.
Байков В. Интернет: поиск информации и продвижение сайтов 288 с.
Гурова А. Герои меча и магии 320 с.
Живайкин П. 600 звуковых и музыкальных программ 624 с.
Закарян И., Филатов И. Интернет как инструмент для финансовых инвестиций 256 с.
Мамаев Е. MS SQL Server 7.0: проектирование и реализация баз данных 416 с.
Мамаев Е. Администрирование SQL Server 7.0 320 с.
Мещеряков М. Linux: инсталляция и основы работы (с компакт-диском) 144 с.
Попов С. Видеосистема PC 400 с.
Соломенчук В. Интернет: поиск работы, учеба, гранты 288 с.
Соломенчук В. Как сделать карьеру с помощью Интернета 416 с.
Успенский И. Интернет как инструмент маркетинга 256 с.
Шарыгин М. Сканеры и цифровые камеры 384 с.

bhv®

Виктор Костром

САМОУЧИТЕЛЬ

Linux

для пользователя

В книге подробно излагаются вопросы установ настройки, администрирования и прикладного пользования ОС Linux. Рассматривается как а операционная система, так и широкий спектр Lin программ, обеспечивающих полноценный на системных и прикладных функций. Рекоменда автора позволят читателю создать мощную и уд ную, дружественную к пользователю Linux-среду

- Интерфейс командной строки и графически интерфейс
- Поддержка мультимедийных средств
- Локальные сети и Интернет
- Взаимодействие с Windows

ISBN 5-94157-183-6

БХВ-Петербург
198005, Санкт-Петербург,
Измайловскийпр.,29
E-mail: mail@bhv.ru
Internet: www.bhv.ru
тел.: (812) 251-42-44
факс: (812) 251-12-95