Основы Slackware Linux

Основы Slackware Linux

вторая редакция

slackware®
linux

Основы Slackware Linux, вторая редакция

Издательство: Slackware Linux, Inc., 1164 Claremont Drive, Brentwood, CA 94513

Ведущий автор второй редакции: Алан Хикс.

Редакторы второй редакции: Мюррей Стоукли и Фу-Канг Чен.

Русский перевод второй редакции: Павел Марьянов.

Авторы первой редакции: Крис Люменс, Девид Кантрелл, и Логан Джонсон.

История изданий:

июнь 2000	первая редакция
май 2005	вторая редакция

ISBN: 1-57176-338-4

Содержание

Список таблиц

Список иллюстраций

Список примеров

Предисловие

Целевая аудитория

Операционная система Slackware Linux - это мощная платформа для компьютеров на базе процессоров Intel. Она разработана для использования в качестве как стабильного, безопасного и многофункционального сервера класса high-end, так и мощной рабочей станции.

Цель этой книги - познакомить вас с операционной системой Slackware Linux. Это не означает, что будут рассмотрены все аспекты использования дистрибутива. Скорее будут показаны его основные возможности, а также будут предоставлены базовые знания для работы в системе.

По мере приобретения вами опыта в использовании Slackware Linux мы надеемся, что вы найдёте эту книгу удобной для использования в качестве настольного справочника. Также мы надеемся, что вы поделитесь ею со всеми своими друзьями, когда они придут к вам расспросить о крутой операционной системе Slackware Linux, которую вы используете.

Хотя эта книга может показаться вам не слишком занятным романом, мы постарались сделать её маскимально интересной. С определённой долей удачи мы попытаемся представить её в виде некоего подобия фильма. Разумеется, мы также надеемся, что, благодаря ей, вы сможете чему-то научиться и найдёте её полезной.

Итак, шоу начинается.

Изменения, сделанные после

первого издания

Второе издание представляет собой венец упорной работы преданных участников Проекта документации Slackware в течение нескольких лет. Ниже представлены основные изменения, сделанные в новой редакции:

- Гл. 3 (Установка): добавлены новые скриншоты инсталлятора с учётом изменившихся наборов дисков и установки с CD.
- Гл. 4 (Настройка системы): добавлена новая информация о ядрах Linux 2.6.x.
- Гл. 5 (Настройка сети): была добавлена дополнительная информация о Samba, NFS и DHCP. Таже был добавлен раздел о работе с беспроводной сетью. Теперь в этой главе отображены главные изменения в настройкой сети в Slackware.
- Гл. 6 (Система X Window): полностью переписана для систем на базе Xorg. В этой главе также описан графический менеджер входа в систему - xdm.
- Гл. 13 (Основные сетевые команды): добавлена информация о дополнительных сетевых утилитах.
- Гл. 14 (Безопасность): новая глава в этой редакции. Рассказывает, как обеспечить безопасность системы Slackware Linux.
- Гл. 17 (Emacs): новая глава в этой редакции. В ней описано использование Emacs - мощного редактора для Unix.
- Гл. 18 (Управление пакетами): обновлена информация о скриптах SlackBuild.
- Сделано ещё много других изменений (больших и маленьких), учитывающих изменения в Slackware по мере его развития.

Организация книги

Гл. 1, Введение

Содержит вводный материал о **Linux**, **Slackware**, а также о движениях **Open Source** и **Free Software**.

Гл. 2, Помощь

Описывает ресурсы для получения помощи, доступные в системе **Slackware Linux** и в Интернете.

Гл. 3, Установка

Описывает процесс установки шаг за шагом со снимками экрана, дающих наглядное представление.

Гл. 4, Настройка системы

Описывает важные конфигурационные файлы и компиляцию ядра.

Гл. 5, Настройка сети

Рассказывает, как подключить машину со **Slackware Linux** к сети. Также описывает **TCP/IP**, **PPP/dial-up**, работу с беспроводными сетями и многое другое.

Гл. 6, Система **X Window**

Описывает, как настроить и использовать в **Slackware** графическую систему **X Window**.

Гл. 7, Загрузка

Описывает процесс, в результате которого компьютер загружает **Slackware Linux**. Также рассмотрена двойная загрузка с операционными системами **Microsoft Windows**.

Гл. 8, Командный процессор (шелл)

Описывает мощный интерфейс командной строки для Linux.

Гл. 9, Структура файловой системы

Описывает структуру файловой системы, включая понятие владельца, права доступа и создание ссылок.

Гл. 10, Работа с файлами и каталогами

Описывает команды, используемые для работы с файлами и каталогами из командной строки.

Гл. 11, Управление процессами

Описывает команды Linux для управления многочисленными работающими приложениями.

Гл. 12, Основы администрирования системы

Описывает основные задачи администрирования системы: добавление и удаление пользователей, корректное выключение системы и много другое.

Гл. 13, Основные сетевые команды

Описывает коллекцию сетевых клиентов, входящих в состав Slackware.

Гл. 14, Безопасность

Описывает множество различных утилит, доступных в Slackware, для обеспечения безопасности вашей системы, включая `iptables` и `tcpwrappers`.

Гл. 15, Архиваторы

Описывает различные доступные в Linux утилиты для сжатия и архивирования данных.

Гл. 16, vi

Описывает мощный текстовый редактор `vi`.

Гл. 17, Emacs

Описывает мощный текстовый редактор `Emacs`.

Гл. 18, Управление пакетами Slackware

Описывает утилиты для работы с пакетами Slackware и процесс, используемый для создания своих собственных пакетов и tag-файлов.

Гл. 19, ZipSlack

Описывает ZipSlack-версию Linux, которую можно использовать в Windows без необходимости установки.

Прил. A, GNU General Public License

Содержит лицензионное соглашение, на условиях которого можно копировать и распространять Slackware Linux и эту книгу.

Соглашения, используемые в этой книге

Чтобы обеспечить ясность и простоту чтения текста, ниже представлены некоторые соглашения, используемые в книге.

Типографские соглашения

Курсив

Курсивный шрифт используется для команд, текста с особым значением и первого использования технических терминов.

`fi>=>H8@8==K9H@8DB`

`fi>=>H8@8==K9` шрифт используется для сообщений об ошибках, команд, переменных окружения, названий портов, имён хостов, имён пользователей, названий групп, названий устройств, переменных и фрагментов кода.

Жирный шрифт

Жирный шрифт используется для данных, вводимых пользователем в примерах.

Данные, вводимые пользователем

Клавиши показываются **жирным шрифтом** для отделения от основного текста. Комбинации клавиш, которые должны вводиться одновременно, показаны со знаком ‘+’ между клавишами, например:

Ctrl+Alt+F1

Здесь подразумевается, что пользователь должен одновременно нажать клавиши **Ctrl**, **Alt** и **F1**.

Клавиши, которые должны быть нажаты последовательно, будут разделены запятыми, например:

Ctrl+X, **Ctrl+S**

Здесь подразумевается, что пользователь должен одновременно нажать клавиши **Ctrl** и **X**, а затем одновременно нажать **Ctrl** и **S**.

Примеры

Примеры, начинающиеся с `E:\>`, обозначают команду MS-DOS®. Если не сделано специальное замечание, эти команды могут быть выполнены из окна “Командная строка” в современной среде Microsoft® Windows®.

```
D:\> rawrite a: bare.i
```

Примеры, начинающиеся с `#`, обозначают команду, которая должна быть выполнена в Slackware под суперпользователем. Вы можете войти в систему под `root`'ом для ввода команды или войти под своей обычной учётной записью и воспользоваться командой su(1) для получения прав суперпользователя.

```
# dd if=bare.i of=/dev/fd0
```

Примеры, начинающиеся с `%`, обозначают команду, которая должна быть выполнена под учётной записью обычного пользователя. Если не сделано специальное замечание, для настройки переменных окружения и других команд командного процессора используется синтаксис C-shell'а.

```
% top
```

Благодарности

Этот проект является результатом многомесячной работы многих людей. Для меня было невозможным оставить эту работу в подвешенном состоянии. Многим людям мы выражаем свою благодарность за их самоотверженные действия: Кейту Келлеру (Keith Keller) за его работу над беспроводными сетями; Джусту Кремерсу (Joost Kremers) за его огромную работу, проделанную одной рукой над написанием главы по emacs'у; Саймону Вильямсу (Simon Williams) за главу по безопасности; Юргену Филиппиртсу (Jurgen Phillippaerts) за основные сетевые команды; Cibao Cu Ali G Colibri за вдохновение и хорошие пинки под зад. Бесчисленному множеству других людей, которые присылали своих пожелания и замечания. Далеко не полный список включает: Jacob Anhoej, John Yast, Sally Welch, Morgan Landry и Charlie Law. Также я хотел бы поблагодарить Кейта Келлера для предоставление хостинга для почтовой рассылки этого проекта, а также Карлу Инглису (Carl Inglis) за первоначальный веб-хостинг. Последними, но не в последнюю очередь, я хотел бы поблагодарить Патрика Фолькердинга (Patrick J. Volkerding) за Slackware

Linux, а также Девида Кантрелла (David Cantrell), Логана Джонсона (Logan Johnson) и Криса Люменса (Chris Lumens) за первую редакцию Slackware Linux Essentials. Без их первоначальной работы ничего бы этого не было. Спасибо многим другим людям, кто сделал свой вклад (большой и не очень) в этот проект, но не был перечислен здесь. Я надеюсь, они простят мне мою слабую память.

Алан Хикс (Alan Hicks), май 2005

Замечания переводчика

Этот перевод я посвящаю Шурику.

Если бы не ты, я бы не взялся за эту работу. Надеюсь, эта книга поможет тебе в дальнейшей работе с DeepStyle Linux.

Хочу выразить особую благодарность Alice Lafox и Олегу Цимаенко (Lafox.Net[1]) за то, что позволили мне работать над этим проектом и не дали умереть с голоду :) (по сути они финансировали этот перевод).

Огромное спасибо Хоттабу (http://deepstyle.org.ua) за прекрасный дистрибутив DeepStyle Linux, а также за небольшую помощь в виде консультаций в спорных моментах перевода. Slackware был моим первым дистрибутивом Linux и следет отдать ему должное - это был действительно уникальный и полезный опыт. Ну а благодаря DeepStyle, Slackware Linux получил ещё большее признание в сегменте русско- и украиноязычных пользователей Linux, захватив в свои сети и Шурика ;)

Спасибо Lao с форума Lafox.Net, который давал весьма дельные советы по общему переводу текстов.

Отдельного замечания заслуживает перевод В. Толпекина, выполненный в далёком 2001-м году. Поначалу я хотел использовать его за основу, просто обновив перевод до актуальной редакции книги. Однако быстро отказался от такого вида работы. Связано это в первую очередь с тем, что стилистика наших переводов всё-таки немного отличается, хотя суть и смысл текста

1 http://lafox.net

остаётся прежним. Честно говоря, перевод Толпекина мне понравился, снимаю шляпу... Однако мне было довольно трудно сверять оригинальный английский текст с довольно таки устаревшим переводом, заниматься вычиткой переведённого материала и переводить from scratch новые части. Для меня это было лишней тратой моего времени. Я гораздо быстрее и качественнее выполняю перевод "с нуля", имея перед глазами только оригинал на английском языке. Поэтому перевод этой редакции "Slackware Linux Essentials" был выполнен мною по сути с нуля (за исключением нескольких отдельных фраз), и именно поэтому на титульной странице не фигурирует имя В.Толпекина. Однако я уважаю копирайты и упоминаю хотя бы здесь о первом переводчике книги.

Перевод выполнен в домашних условиях и на базе Центра распространения свободного программного обеспечения Lafox.Net. Программное обеспечение: ОС Mandriva Linux 2006 (Cooker), KDE 3.5.x, kate+ispell, qemu (FreeBSD 6.0, docproj).

Переводчик: Павел Марьянов (acid_jack@ukr.net), март - май 2006.

Глава 1.

Введение в Slackware Linux

1.1. Что такое Linux?

Линус Торвальдс (Linus Torvalds) начала работать над Linux - ядром операционной системы - в 1991 в качестве личного проекта. Линус начал этот проект потому, что он хотел работать в операционной системе на базе Unix без существенных материальных затрат. В дополнение к этому он хотел изучить подробности ввода и вывода 386-го процессора. Linux был абсолютно бесплатно предоставлен для всеобщего доступа, чтобы любой мог изучить его и внести свои улучшения согласно условиям General Public License (описание лицензии см. в Разд. 1.3 и Прил. A). Сегодня Linux вырос в одного из ведущих игроков на рынке операционных систем. Он был портирован на большое число различных системных архитектур, включая HP/Compaq'овские Alpha, Sun'овские SPARC и UltraSPARC, а также на чипы PowerPC от Motorola (на компьютерах Apple Macintosh и IBM RS/6000). Сотни, если не тысячи, программистов по всему миру разрабатывают сейчас Linux. В нём работают такие программы как Sendmail, Apache и BIND, которые являются очень популярным программным обеспечением, используемым на Интернет-серверах. Важно знать, что термин "Linux" на самом деле означает ядро - сердце операционной системы. Это ядро отвечает за управление процессором, памятью, жёсткими дисками и

периферийным оборудованием вашего компьютера. Это как раз то, что Linux делает на самом деле: он контролирует внутренние действия вашего компьютера и обеспечивает работу всех программ. Различные компании и отдельные люди собирают вместе ядро и различные программы для получения законченной операционной системы. Такие сборки мы называем дистрибутивом Linux.

Слово о GNU

Проект ядра Linux был начат в 1991 году усилиями одного единственного человека - Линуса Торвальдса. Однако, как сказал однажды Исаак Ньютон: "Если я видел дальше других, то только потому, что стоял на плечах гигантов." Когда Линус Торвальдс начал создавать ядро, Фонд свободного программного обеспечения (Free Software Foundation, FSF) уже сформировал идею общего программного обеспечения. Результат своих усилий они назвали GNU - рекурсивный акроним, означающий просто "GNU's Not Unix" (GNU - это не Unix). Программное обеспечение GNU работало на ядре Linux с первого же дня. А для компиляции ядра использовался их компилятор `gcc`. Сегодня многие утилиты GNU от `gcc` до `gnutar` всё ещё являются базой для всех ведущих дистрибутивов Linux. По этой причине многие сторонники Фонда свободного программного обеспечения горячо настаивают на том, что их работу следует оценивать так же, как и ядро Linux. Они упорно настаивают, что все дистрибутивы Linux должны ссылаться на самих себя как на дистрибутивы GNU/Linux.

Этот вопрос является темой многих стычек и перебранок, превзойти которые может только "священная война" между vi и emacs. Целью этой книги является не накал страстей и без того жарких дискуссий, а скорее ознакомление новичков с терминологией. Когда мы говорим о GNU/Linux, это означает дистрибутив. Когда же мы говорим о Linux, это может быть и ядро, и дистрибутив. Это может сбить с толку. Обычно термин GNU/Linux не используется, потому что его сложнее произносить.

1.2. Что такое Slackware?

Slackware был создан Патриком Фолькердингом (Patrick Volkerding) в конце 1992 года и впервые был представлен широкой общественности 17 июля 1993 года. Это был первый дистрибутив Linux, получивший широкое распространение. Фолькердинг был впервые начал изучать Linux, когда для проекта ему понадобился недорогой интерпретатор LISP. Одним из немногих дистрибутивов, доступных на тот момент, был SLS Linux от Soft Landing Systems. Фолькердинг использовал SLS Linux, исправляя в нём ошибки по мере их нахождения. В конечном итоге он решил объединить все исправления ошибок в свой собственный дистрибутив, чтобы его могли использовать он сам и его друзья. Этот персональный дистрибутив быстро получил большую популярность, поэтому Фолькердинг решил дать ему имя Slackware и сделал его доступным для широкой общественности. Со временем Патрик добавил в Slackware новые вещи: программу установки с дружественным интерфейсом, основанным на системе меню, а также понятие управления пакетами, которое позволяет пользователям легко выполнять в своей системе добавление, удаление или обновление пакетов с программным обеспечением.

Существует много причин, по которым Slackware является самым старым из существующих по сей день дистрибутивов Linux. Он не пытается эмулировать Windows, он старается быть похожим на Unix настолько, насколько это возможно. Он не пытается обвешать процессы рюшечками, графическими интерфейсам в стиле point-and-click. Вместо этого он предоставляет пользователям полный контроль над системой, позволяя им непосредственно видеть, что происходит. Его разработка ведётся без установки граничной даты выпуска: каждая версия выходит тогда, когда она готова.

Slackware предназначен для людей, которым доставляет удовольствие изучать и тонко настраивать свои системы, чтобы эти системы делали только то, что нужно их пользователям. Стабильность и простота Slackware - это главные качества системы, благодаря которым

пользователи используют её уже многие годы и продолжают использовать по сей день. На сегодняшний день Slackware гордится своей репутацией как устойчивого сервера, так и рабочей станции без неожиданных сюрпризов. Вы можете встретить настольные системы Slackware, работающие с любым оконным менеджером или средой рабочего стола, или вообще без них. Под управлением Slackware работают мощные бизнес-решения, используя все возможности сервера, которые он может предоставить. Пользователи Slackware относятся к наиболее опытным пользователям Linux.

1.3. Open Source и Free Software

В сообществе Linux существуют два главных идеологических направления работы. Целью Движения за свободное программное обеспечение (Free Software Movement или FSF, к которому мы сейчас вернёмся) является освобождение всего программного обеспечения от ограничений, присущих интеллектуальной собственности. Сторонники этого движения верят, что эти ограничения препятствуют техническому прогрессу и направлены против общего блага для сообщества. Цели Движения за открытые исходные тексты (Open Source Movement) являются почти такими же самыми, однако имеют более прагматический подход. Сторонники этого движения предпочитают основывать свои аргументы на экономических и технических преимуществах предоставления исходного кода, как полностью свободного, а не на моральных и этических принципах, которые лежат в основе Движения за свободное программное обеспечение.

По другую сторону баррикады находится ряд групп, которые хотят иметь больший контроль над своим программным обеспечением.

Во главе Движения за свободное программное обеспечение находится Фонд свободного программного обеспечения (Free Software Foundation) - организация по сбору средств для проекта GNU. Свободное ПО - это скорее идеология. В этом аспекте наиболее часто используемое выражение звучит так: “Free as in speech, not free as in beer” (“свобода слова, а не бесплатное

пиво"). (От англ. free - свободный, бесплатный (прим. переводчика)). По своей сути свободное ПО - это попытка гарантировать определённые права и пользователям, и разработчикам. Эти свободы включают: свободу использовать программу для любых целей, изучать и модифицировать исходный код, распространять далее исходные тексты и предоставлять его с любыми внесёнными вами изменениями. Для того, чтобы гарантировать эти эти свободы, была создана GNU General Public License (GPL). Если быть кратким, GPL утверждает, что любой, кто распространяет скомпилированную программу, на которую распространяется действие GPL, должен также предоставить её исходный код, а также волен вносить изменения в программу до тех пор, пока эти изменения также будут доступны в виде исходных текстов. Это гарантирует, что если программа однажды была "открыта" сообществу, она не может быть "закрыта" за исключением тех случаев, когда все авторы каждой из частей кода (даже модификаций) дадут на это согласие. На большинство программ Linux распространяется действие GPL.

Следует отметить, что в GPL ничего не говорится о стоимости. Как бы странно это ни звучало, вы можете взымать плату за свободное программное обеспечение. В данном случае под "free" подразумевается свобода исходного кода, а не цена, которую вы платите за программное обеспечение. Однако, если кто-то однажды продаст или даже просто даст вам скомпилированную программу, выпущенную под GPL, он также должен предоставить её исходный код.

Другой популярной лицензией является лицензия BSD. BSD-лицензия в отличие от GPL не требует предоставлять исходный код программы. Программное обеспечение, выпущенное под лицензией BSD, разрешается распространять дальше в виде бинарных файлов или исходных текстов только при условии соблюдения нескольких условий. Данные об авторе не могут использоваться с целью рекламирования программы. Это также освобождает автора от ответственности за ущерб, который может быть получен в результате использования программного обеспечения. Большая часть программного обеспечения в Slackware Linux находится под

действием лицензии BSD.

На передовых позициях более молодого движения Open Source находится Open Source Initiative - организация, которая существует только для того, чтобы предоставлять техническую поддержку программного обеспечения с открытыми исходными текстами, т.е. ПО, для которого доступен исходный код и готовая к работе программа. OSI не предлагает особую лицензию, но заинтересована в поддержке различного вида лицензий на ПО с открытыми исходными текстами.

Идея, лежащая в основе OSI, заключается в привлечении большего числа компаний, работающих с открытыми исходными текстами, позволяя им писать свои собственные opensource-лицензии и сертифицировать их в OSI. Многие компании хотят предоставлять исходный код, но не хотят использовать GPL. Так как они не могут полностью изменить GPL, вместо этого им предлагается разработать свою собственную лицензию и сертифицировать её в этой организации.

Хотя Free Software Foundation и Open Source Initiative работают вместе, помогая друг другу, это разные организации. FSF использует особую лицензию и предоставляет программное обеспечение под этой лицензией. OSI стремится поддерживать все лицензии на ПО с открытым исходным кодом, включая лицензии FSF. Доводы, которые все приводят в пользу предоставления свободного доступа к исходному коду, иногда разделают эти два движения, но сам факт, что две идеологически разные группы работают в одном направлении для достижения общей цели, заставляет верить в усилия каждого из них.

Глава 2.

Получение поддержки и помощи

Довольно часто случается так, что вам необходима помощь по работе определённой команды, настройке программы или необходимости заставить работать то или иное оборудование. А, возможно, вы просто хотите лучше разобраться с данной командой или узнать, какие для неё доступны опции. К счастью существует множество способов для получения такой помощи. Когда вы устанавливаете Slackware, у вас есть возможность установить пакеты из категории "F", в состав которой входят документы FAQ и HOWTO. Сами программы также могут поставляться со своей документацией, содержащей описание их опций, конфигурационных файлов и их использования.

2.1. Методы получения справки из системы

man

Команда `man` (сокращение от "manual" - руководство, справочник) это традиционная форма интерактивной документации в операционных системах Unix и Linux. Будучи представленными в виде файлов в специальном формате, "man pages" (комп. сленг. "маны") написаны для огромного числа команд и распространяются вместе с программным обеспечением. При вызове из командной строки `man какая-то_команда` (обычно) будет показана страница руководства для указанной команды; в нашем примере это была бы воображаемая `какая-то_команда`.

Как понимаете, количество страниц руководства может увеличиваться очень быстро, что приводит к запутыванию и затруднению работы с ними, даже для опытных пользователей. Поэтому, чтобы избежать этих проблем, страницы сгруппированы по разделам. Эта система существует уже очень давно, настолько давно, что вы часто будете встречать команды, программы и даже библиотечные функции, которые ссылаются на свой раздел страницы руководства.

Например:

Вы можете увидеть ссылку на `man`(1). Эта цифра означает, что документация по команде "`man`" находится в 1-м разделе 1 (команды пользователя); вы можете указать, что вам нужна страница по "man" именно из первого раздела с помощью команды `man 1 man`. Указание раздела полезно в том случае, если существует несколько пунктов с одинаковым именем.

Таблица 2-1. Разделы страниц руководства

Раздел	Содержание
Раздел 1	команды пользователя (только для ознакомления)
Раздел 2	системные вызовы
Раздел 3	вызовы библиотеки C
Раздел 4	устройства (напр., `hd`, `sd`)
Раздел 5	форматы файлов и протоколы (напр., wtmp, `/etc/passwd`, nfs)
Раздел 6	игры (только для ознакомления)
Раздел 7	соглашения, макропакеты и т.п. (напр., nroff, ascii)
Раздел 8	администрирование системы (только для ознакомления)

В дополнение к `man`(1) также ещё существуют команды `whatis`(1) и `apropos`(1), целью которых является упрощение поиска информации в системе man.

Команда `whatis` даёт очень кратное описание системных команд; что-то в духе карманного справочника по командам.

Пример:

```
% whatis whatis
whatis (1)  - search the whatis database for complete words
```

Команда `apropos` используется для поиска страницы, содержащей указанное ключевое слово.

Пример:

```
% apropos wav
cdda2wav    (1)  - a sampling utility that dumps CD audio data into wav sound files
netwave_cs  (4)  - Xircom Creditcard Netwave device driver
oggdec      (1)  - simple decoder, Ogg Vorbis file to PCM audio file (WAV or RAW)
wavelan     (4)  - AT&T GIS WaveLAN ISA device driver
wavelan_cs  (4)  - AT&T GIS WaveLAN PCMCIA device driver
wvlan_cs    (4)  - Lucent WaveLAN/IEEE 802.11 device driver
```

Если вы хотите получить дополнительную информацию по этим командам, почтите их собственные страницы руководства ;)

Каталог `/usr/doc`

Исходные тексты для большинства собранных нами пакетов поставляются с различного рода документацией: файлы **README**, инструкции по использованию, файлы лицензий и т.п. Все эти документы устанавливаются в каталог `/usr/doc`. Каждая из программ (обычно) устанавливает свою собственную документацию в таком виде:

```
/usr/doc/$?@>3@0<<0-$25@A8O
```

Где *`$программа`* - это название программы, по которой вы хотите получить информацию, а *`$версия`* - (очевидно) соответствующая версия программного пакета, установленного в вашей системе.

Например, чтобы прочитать документацию по команде `man`(1), вам следует перейти (`cd`) в каталог:

```
% cd /usr/doc/man-$версия
```

Если прочтение соответствующей страницы руководства не дало вам достаточно информации, тогда следущим вашим шагом должно быть посещение каталога `/usr/doc`.

Документы HOWTO и mini-HOWTO

Истинный дух сообщества Open Source заключается в предоставлении коллекции документов HOWTO/mini-HOWTO. Название этих файлов говорит само за себя (примерный перевод звучит “как сделать...”) - документы и руководства, описывающие как сделать то или иное. Если вы установили коллекцию HOWTO, эти документы будут находиться в каталоге `/usr/doc/Linux-HOWTOs`, а mini-HOWTO - в `/usr/doc/Linux-mini-HOWTOs`.

В этот же набор пакетов входит коллекция FAQ, что является акронимом

*F*requently
*A*sked
*Q*uestions

Эти документы написаны в виде “Вопросов с ответами” (что, собственно, и обозначает аббревиатура - Часто задаваемые вопросы). FAQ’и часто могут оказаться весьма полезным источником для поиска информации, если вы ищете что-то в духе “Как быстро исправить” что-нибудь. Если вы установили FAQ’и во время установки системы, вы найдёте их в каталоге `/usr/doc/Linux-FAQs`.

Эти файлы стоит прочитать в том случае, если вы не уверены в том, что делать дальше в определённых ситуациях. Они охватывают удивительно широкий диапазон рассматриваемых тем и (довольно часто) очень

подробно. Приятного вам чтения!

2.2. Интерактивная справка

В дополнение к документации, поставляемой с ОС Slackware Linux, существует ещё множество онлайновых ресурсов для изучения.

Официальный веб-сайт и форумы помощи

Официальный веб-сайт Slackware[1]

Официальный веб-сайт Slackware Linux иногда оказывается устаревшим, однако всё равно содержит информацию о последних версиях Slackware. Также когда-то ещё существовал форум поддержки, пока на него не посыпались орды "троллей", нарушителей общественного порядка и нытиков. Сопровождение форума добавило слишком много лишних хлопот Пату и поэтому он решил закрыть его. Однако на http://www.userlocal.com/phorum/ было обнаружено, что старый форум снова работает, а также доступен архив со старыми данными с возможностью поиска по ним.

После того, как на http://slackware.com были закрыты форумы, появилось несколько других сайтов, предлагающих разместить у себя форумы поддержки Slackware. После долгих раздумий Патрик дал своё согласие на то, чтобы www.linuxquestions.org считался официальным форумом Slackware Linux.

Поддержка по электронной почте

Все, кто приобрёл официальный набор компакт-дисков, получают право на получение от разработчиков бесплатной технической поддержки

1 http://www.slackware.com

по установке посредством электронной почты. Следует отметить, что мы - разработчики и преобладающее большинство пользователей Slackware - относимся к поколению "старой школы". Это означает, что мы предпочитаем помогать тем, кто желает помочь сам себе и действительно заинтересован в этом. Мы всегда стараемся максимально помочь всем, кто присылает нам письма с интересующим их вопросами. Однако ознакомьтесь сначала с документацией и загляните на веб-сайт (в особенности в FAQ и, возможно, на некоторые форумы, перечисленные ниже) перед тем, как обращаться к нам по электронной почте. Таким образом вы сможете получить получить ответ гораздо быстрее, и чем меньше почты нам придётся обработать, тем скорее мы сможем оказать помощь тем, кто действительно в ней нуждается.

Адрес электронной почты для технической поддержки: `support@slackware.com`. Другие адреса e-mail и контактная информация доступны на веб-сайте.

Почтовые рассылки проекта Slackware Linux

У нас есть несколько постовых рассылок, доступных в обычном виде и в виде дайджеста. Ознакомьтесь сначала с инструкциями о том, как подписаться.

Чобы подписаться на рассылку, отправьте письмо на адрес:

`majordomo@slackware.com`

с фразой "`subscribe` *`[название рассылки]`*" в теле письма. Перечень рассылок представлен ниже (используйте только одно название из списка).

Архивы почтовых рассылок можно найти на сайте Slackware по адресу:

`http://slackware.com/lists/archive/`

`slackware-announce`

Рассылка `slackware-announce` предназначена для объявлений о новых

версиях, важных обновлениях и другой общей информации.

`slackware-security`

Рассылка `slackware-security` предназначена для объявлений, касающихся вопросов безопасности. В эту рассылку присылаются новости обо всех эксплойтах и других уязвимостях, непосредственно угрожающих системе Slackware.

Эти рассылки также доступны в виде дайджеста. Это означает, что вы каждый день получаете одно большое сообщение, вместо нескольких сообщений в течение дня. Т.к. рассылки Slackware не позволяют пользователям рассылать свои сообщения, а трафик настолько небольшой, что многие пользователи найдут в использовании дайджест-формата не так уж и много преимуществ. Тем не менее вы можете подписаться на них: `slackware-announce-digest` или `slackware-security-digest`.

Неофициальные веб-сайты и форумы поддержки

Веб-сайты

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

Просто мастер кунг-фу среди поисковых систем. Если вам нужно гарантированно получить абсолютно всю самую свежую информацию, тогда никаках вариантов - это именно то, что вам нужно.

Google:Linux (http://www.google.com/linux)

Поисковые запросы, имеющие отношение сугубо к Linux

Google:BSD (http://www.google.com/bsd)

Поисковые запросы, касающиеся BSD. Slackware является настолько

общей операционной системой, работающей как Unix, что здесь очень часто можно встретить нужную информацию, практически на 100% имеющую отношение к Slackware. Неоднократно поиски по BSD давали гораздо более подробную техническую информацию, чем поисковые запросы по Linux.

Google:Groups (http://groups.google.com)

Поиск информации в сообщениях Usenet.

http://userlocal.com

Просто виртуальный кладезь знаний, хороших советов, результатов личного опыта и интересных статей. Зачастую это первый ресурс, о котором вы услышите, желая у знать о новых и текущих разработках в мире Slackware.

Прочие веб-ресурсы

linuxquestions.org[6]

Официальный форум для пользователей Slackware.

Форум Slackware на LinuxISO.org[7]

“Сайт для получения помощи и загрузки Linux.”

alt.os.linux.slackware FAQ[8]

Ещё один FAQ

6 http://www.linuxquestions.org/questions/forumdisplay.php?forumid=14
7 http://forums.linuxiso.org/viewforum.php?f=25
8 http://wombat.san-francisco.ca.us/perl/fom

Группы Usenet (NNTP)

Usenet долгое время была местом сбора компьютерных фанатов для обмена информацией и получения помощи. Существует всего лишь несколько конференций, посвящённых Slackware Linux, однако в них собираются очень опытные пользователи.

`alt.os.linux.slackware`

Конференция `alt.os.linux.slackware` - больше известная как aols (не путать с AOL®!) - это один из наиболее активных ресурсов, на котором можно получить техническую помощь по проблемам со Slackware. Как и в любой конференции Usenet некоторые бесполезные участники (“тролли”) могут доставить много пустых хлопот, постоянно споря со всеми. Научитесь игнорировать таких троллей и выделять действительно полезных людей, чтобы извлечь максимум пользы из этого ресурса.

Глава 3.

Установка

Перед тем, как вы сможете приступить к использованию Slackware Linux, вам понадобиться получить его и выполнить установку. Получить Slackware очень легко: вы можете приобрести его или бесплатно загрузить из Интернета. Установить его также легко, если вы обладаете некоторыми базовыми знаниями о компьютерах и хотите узнать ещё кое-что. Программа установки представляет сама по себе пошаговый процесс. Благодаря этому, вы можете довольно быстро получить работающую систему. К тому же Slackware может похвастаться почти самым минимальным временем установки среди других полноценных дистрибутивов Linux.

3.1. Получение Slackware

Официальные наборы дисков и коробки

Официальный набор компакт-дисков со Slackware Linux можно получить в компании Slackware Linux, Inc. Комплект дисков состоит из 4 CD. Первый диск содержит всё программное обеспечение, необходимое для установки базового сервера с системой X Window. Второй диск представляет собой "живой" CD (LiveCD), т.е. загрузочный диск, который загружается в оперативную память и предоставляет в ваше распоряжение временную систему, чтобы вы могли побаловаться с ней или восстановить свои данные или существующую систему. Этот диск также содержит несколько

пакетов, таких как настольные среды KDE и GNOME. Кроме того на нём есть несколько полезных пакетов, без которых вы можете вполне обойтись. Находятся они в каталоге “extra”. Третий и четвёртый компакт-диски содержат исходные тексты всего Slackware вместе с оригинальным изданием этой книги.

Также можно приобрести коробочную версию дистрибутива, которая включает 4 диска и копию этой книги, плюс атрибутику, которой позавидуют все ваши знакомые слаководы. Кроме того можно заказать CD по сниженной цене.

Предпочтительным способом приобретения товаров Slackware является посещение онлайнового магазина Slackware.

http://store.slackware.com

Вы также можете сделать свой заказ по телефону или через электронную почту.

Таблица 3-1. Информация для связи со Slackware Linux, Inc.

Способ	Контакты
Телефон	1-(925) 674-0783
Веб-сайт	http://store.slackware.com
Email	orders@slackware.com
Почта	1164 Claremont Drive, Brentwood, CA 94513

Через Интернет

Slackware Linux также свободно доступен в Интернете. Вы можете использовать электронную почту для обращения за помощью в службу технической поддержки, однако более высокий приоритет имеют те, кто приобрёл официальный набор дисков. Другими словами мы получаем очень много электронной корреспонденции, а наше время при этом

довольно ограничено. Перед тем, как обращаться в службу технической поддержки, прочтите сначала Гл. 2.

Официальный веб-сайт проекта Slackware Linux:

http://www.slackware.com/

Основной FTP-сервер Slackware Linux:

ftp://ftp.slackware.com/pub/slackware/

Учтите, что наш сервер FTP хоть и открыт для общего доступа, но не обладает неограниченной пропускной способностью каналов. Чтобы загрузить Slackware, пожалуйста, попробуйте сначала воспользоваться ближайшим к вам зеркалом. Неполный список зеркал можно найти на сайте http://www.slackware.com/getslack.

3.2. Системные требования

Для простой установки Slackware требуется как минимум следующее:

Таблица 3-2. Системные требования

Оборудование	Требование
Процессор	586
Оперативная память	32 МБ
Дисковое пространство	1 ГБ
Оптический привод	4x CD-ROM

Если в вашем распоряжении есть загрузочный CD, вам скорее всего не понадобится дисковод. Понятное дело, что в противном случае (если у вас нет привода CD-ROM) вам понадобится дисковод для установки по сети. Для установки через NFS нужна сетевая карта. Ддополнительную информацию смотрите в разделе под названием NFS.

Требования к дисковому пространству имеют более хитрый характер. Рекомендованного 1ГБ обычно хватает для минимальной установки, однако если вы делаете полную установку, вам понадобится порядка двух гигабайт свободного дискового пространства плюс дополнительное место для личных файлов... Большинство пользователей не делает полную установку. Зачастую многие разворачивают систему Slackware на дисковом пространстве в 100МБ.

Slackware можно установить на системы с меньшим объёмом памяти, менее ёмкими жёсткими дисками и более слабыми процессорами, однако для этого понадобится сделать несколько “финтов ушами”. Если вы готовы немного поработать в этом направлении, ознакомьтесь с файлом `LOWMEM.TXT` из дерева дистрибутива на предмет полезных хитростей и уловок.

Категории программного обеспечения

Исторически с целью упрощения программное обеспечение Slackware было разбито на категории (series). Они получили название “наборы дисков”, потому что разрабатывались для установки с дискет. Сейчас категории программ используются в основном для классификации пакетов, входящих в состав дистрибутива Slackware. Сегодня установка с дискет больше невозможна.

Ниже представлено короткое описание каждой из категорий программного обеспечения.

Таблица 3-3. Категории программ

Категория	Содержание
A	Базовая система. Содержит программное обеспечение, достаточное для получения рабочей системы, включая текстовый редактор и основные коммуникационные программы.

Категория	Содержание
AP	Различные приложения, для работы которых не требуется система X Window.
D	Инструменты для разработки программ. Компиляторы, отладчики, интерпретаторы и страницы руководства.
E	Текстовый редактор Emacs от GNU.
F	Документы FAQ, HOWTO и другая разнообразная документация.
GNOME	Среда рабочего стола GNOME.
K	Исходный код ядра Linux.
KDE	Среда рабочего стола KDE (K Desktop Environment). Графическая среда с оформлением и внешнем виде в стиле MacOS и Windows. В этой категорию также входит библиотека Qt, необходимая для работы KDE.
KDEI	Пакеты с локализацией настольной среды KDE.
L	Библиотеки. Динамически подключаемые библиотеки, необходимые для работы многих программ.
N	Сетевые программы. Демоны, почтовые клиенты, telnet, клиенты чтения новостей и т.п.
T	Система форматирования документов teTeX.
TCL	Язык TCL (Tool Command Language). Tk, TclX и TkDesk.
X	Базовый комплект для системы X Window.
XAP	Графические приложения, не являющиеся частью больших настольных сред (например, Ghostscript и Netscape).
Y	Консольные игрушки BSD

Способы установки

С дисковода

Хотя когда-то Slackware Linux можно было установить с дискет, растущий

объём программных пакетов (а точнее самих программ) привёл к необходимости отказаться от установки с дискет. До Slackware версии 7.1 можно было выполнить частичную установку с дискет. Можно было почти полностью установить категории A и N, получив при этом базовую систему, из которой можно было доустановить остальную часть дистрибутива. Если вы рассматриваете вариант установки с дискет (обычно на старом оборудовании), обычно всё же рекомендуется найти другой способ или взять более старый релиз. По этой причине всё ещё довольно популярен Slackware 4.0, равно как и 7.0.

Пожалуйста, учтите, что необходимость в дискетах всё ещё присутствует в случае установки с CD-ROM, когда у вас нет загрузочного компакт-диска, а также для установки через NFS.

С CD-ROM'а

Если в вашем распоряжении имеется загрузочный CD, который доступен в официальном наборе дисков, распространяемом Slackware Linux, Inc. (см. раздел Получение Slackware), для вас будет гораздо проще выполнить установку с CD. В противном случае вам понадобится загрузиться с дискет. Также, если вы являетесь обладателем нестандартного оборудования, которое вызывает у ядра проблемы с загрузкой с CD, вам возможно понадобится воспользоваться специальными дискетами.

Начиная с версии 8.1, в Slackware используется новый способ создания загрузочных компакт-дисков, который не работает на некоторых чипах BIOS (следует отметить, что в те времена от этого страдало большинство CD с Linux). В этом случае рекомендуется выполнить загрузку с дискеты.

В Разд. 3.2.3 и Разд. 3.2.5 представлена всё необходимая информация по выбору и созданию загрузочных дискет.

С NFS

NFS (Network File System, сетевая файловая система) - это способ предоставления удалённым машинам доступа к локальным файловым системам. Использование NFS позволяет вам установить Slackware по сети с другого компьютера. На машине, с которой вы выполняете установку, должно быть настроено экспортирование дерева с дистрибутивом для машины, на которую вы выполняете установку. При этом, конечно же, подразумевается, что вы обладаете некоторыми знаниями об NFS, которая описана в Разд. 5.6.

Также возможна установка с NFS с использованием таких методов как PLIP (через параллельный порт), SLIP и PPP (однако не через модемное соединение). Тем не менее мы рекомендуем использовать сетевую карту, если таковая имеется. Ведь в конце концов установка операционной системы через порт принтера - это ОЧЕНЬ медленный процесс.

Загрузочный диск

Загрузочный диск - это дискета, которую вы загружаете в начале установки. Она содержит сжатый образ ядра, которое используется для управления оборудованием во время установки. Поэтому этот диск очень нужен (только если вы не загружаетесь с компакт-диска, как это было описано в разделе установки с CD-ROM'а). Загрузочные диски находятся в дереве дистрибутива в каталоге `bootdisks/`.

Есть ещё несколько загрузочных дисков Slackware, которые вы можете использовать (порядка 16 штук). Полный список с описанием каждого их них доступен в дереве дистрибутива в файле `bootdisks/README.TXT`. Однако большинство людей могут использовать образы `bare.i` (для IDE-устройств) или `scsi.s` (для SCSI-устройств).

Подробные инструкции о том, как создать диск из файла, смотрите в Разд. 3.2.6.

После загрузки вам будет предложено вставить root-диск. Мы рекомендуем вам не обращать на это внимание и продолжать работать с загрузочным диском.

Root-диск

Root-диски содержат программу установки и файловую систему, которая используется во время установки. В них также есть необходимость. Образы root-дисков находятся в дереве дистрибутива в каталоге rootdisks. Вам понадобится создать два root-диска из образов `install.1` и `install.2`. Там же вы найдёте диски `network.dsk`, `pcmcia.dsk`, `rescue.dsk` и `sbootmgr.dsk`.

Дополнительный диск

Дополнительный диск нужен в том случае, если вы выполняете установку с NFS или с PCMCIA-устройств. Дополнительные диски находятся в каталоге rootdsks дерева дистрибутива и имеют названия `network.dsk` и `pcmcia.dsk`. Недавно были добавлены ещё два диска - `rescue.dsk` и `sbootmgr.dsk`. Аварийный диск (rescue) - это небольшой образ root-дискеты, который разворачивается в памяти на 4МБ. Он содержит различные базовые сетевые утилиты и редактор vi для быстрого восстановления повреждённых систем. Диск `sbootmgr.dsk` используется для загрузки с других устройств. Выполните загрузку с этой дискеты, если ваш привод CD-ROM не хочет загружаться с компакт-дисков Slackware. Вам будет предложено несколько вариантов загрузки, а также удобный способ для обхода проблем с некоторыми BIOS'ами.

После своей загрузки root-диск проинструктирует вас о том, как его использовать.

Создание дисков

После того как вы выбрали образ загрузочного диска, вам необходимо перенести его на дискету. Процесс мало чем отличается в зависимости от того, какую операционную системы вы используете для создания дисков. Если вы работаете в Linux (или любой другой UNIX-подобной ОС), вам необходимо воспользоваться командой `dd`(1). При условии, что `bare.i` - это файл образа вашего диска, а ваш дисковод - `/dev/fd0`, команда для создания дискеты `bare.i` будет выглядеть так:

```
% dd if=bare.i of=/dev/fd0
```

Если вы работаете в ОС компании Microsoft, вы должны будете воспользоваться программой `RAWRITE.EXE`, которая есть в дистрибутиве в том же каталоге, что и образы дискет. При условии, что `bare.i` - это файл образа вашего диска, а дисковод - `A:`, откройте окно с командной строкой DOS и наберите следующее:

```
C:\ rawrite a: bare.i
```

3.3. Разметка диска

После загрузки с предпочитаемого носителя вам понадобится разметить свой жёсткий диск. Раздел диска - это область диска, на которой будет создана файловая система для установки на неё Slackware. Как минимум мы рекомендуем создать два раздела: один для корневой файловой системы (`/`), а второй - под пространство для свопинга.

После загрузки root-диска вы увидите на экране приглашение для входа в систему. Войдите под root'ом (без пароля). В командной строке командного процессора запустите `cfdisk`(8) или `fdisk`(8). Программа `cfdisk` предоставляет пользователю интерфейс более дружественный, чем

у `fdisk`, однако у него отсутствуют некоторые функции. Ниже мы кратко опишем программу `fdisk`.

Начнём с запуска `fdisk` для выбранного вами жёсткого диска. В Linux жёстким дискам не присваиваются буквенные обозначения, они представляются в виде файлов. Первый жёсткий IDE-диск (первичный старший) - это `/dev/hda`, первичный младший - это `/dev/hdb` и т.д. Для SCSI-дисков используется подобная система, однако в виде `/dev/sdX`. Вам понадобится запустить `fdisk` и передать ему в качестве параметра свой жёсткий диск:

```
# fdisk /dev/hda
```

Как и все хорошие Unix-программы `fdisk` предоставляет вам строку приглашения (а вы подумали, что получите меню, не так ли?). Первым делом вам необходимо проверить наличие существующих разделов. Делается это путём набора **`p`** в приглашении `fdisk`:

```
Command (m for help): p
```

При этом на экран будет выведена вся информация о существующих разделах. Большинство людей используют для установки свободный диск, а затем удаляют на нём всю разметку, чтобы освободить место для разделов Linux.

> **Внимание**ОЧЕНЬ ВАЖНО СДЕЛАТЬ РЕЗЕРВНУЮ КОПИЮ ЛЮБОЙ ВАЖНОЙ ИНФОРМАЦИИ, КОТОРУЮ ВАМ НУЖНО СОХРАНИТЬ ПЕРЕД ТЕМ, КАК УДАЛЯТЬ РАЗДЕЛЫ, НА КОТОРЫХ ОНА НАХОДИТСЯ.

Не существует простого способа восстановления данных после удаления разделов жёсткого диска, поэтому делайте резервную копию перед манипуляциями с разделами.

Взглянув на таблицу разделов, вы должны увидеть в ней номера разделов, их размеры и типы. Там присутствует ещё и другая информация, однако сейчас вам не следует обращать на неё внимание. Сейчас нам нужно удалить на этом диске все разделы и создать разделы для Linux. Для удаления воспользуйтесь командой **d**:

```
Command (m for help): d
Partition number (1-4): 1
```

Процесс необходимо повторить для каждого из разделов. После их удаления можно создавать разделы для Linux. Мы решили создать один раздел для корневой файловой системы и один для свопинга. Следует отметить, что схемы разметки Unix являются предметом для множества горячих споров и перебранок, а также то, что большинство пользователей будут рассказывать вам о самом лучшем способе разметки. Как минимум вы должны создать один раздел для `/` и один для свопинга. Со временем вы разработаете свой собственный способ, наиболее подходящий для вас.

Я использую в основном две схемы разметки. Первая - для настольной системы. 4 раздела: `/`, `/home`, `/usr/local` и своп. Такая схема позволяет мне переустановить или обновить полностью всю систему, находящуюся в `/`, не затрагивая личные файлы в `/home` и скомпилированные вручную приложения из каталога `/usr/local`. Для серверов я часто заменяю раздел `/usr/local` на `/var`. Многие серверные приложения хранят информацию в этом разделе и отделение его от раздела `/` даёт заметный выигрыш в производительности. Сейчас же мы ограничимся двумя разделами: `/` и своп.

Теперь настало время создать разделы с помощью команды **n**:

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4):1
First cylinder (0-1060, default 0):0
```

```
Last cylinder or +size or +sizeM or +sizeK (0-1060, default 1060):+64M
```

Убедитесь, что вы создали основные разделы. Первый раздел предназначается для свопинга. Мы сообщаем fdisk'у, что нам нужно сделать первый раздел основным. Начинаем мы его с нулевого цилиндра, а для конечного цилиндра указываем +64M. Таким образом мы создали раздел для свопинга размером 64 мегабайта. Размер swap-раздела по сути зависит от объёма оперативной памяти. Обычной практикой было создание раздела размером в два объёма ОЗУ (однако при наличии 512МБ и более памяти это правило теряет смысл; тогда от использования свопа можно вообще отказаться - прим.переводчика). Затем мы создаём второй основной раздел, начиная с первого доступного цилиндра и до конца диска.

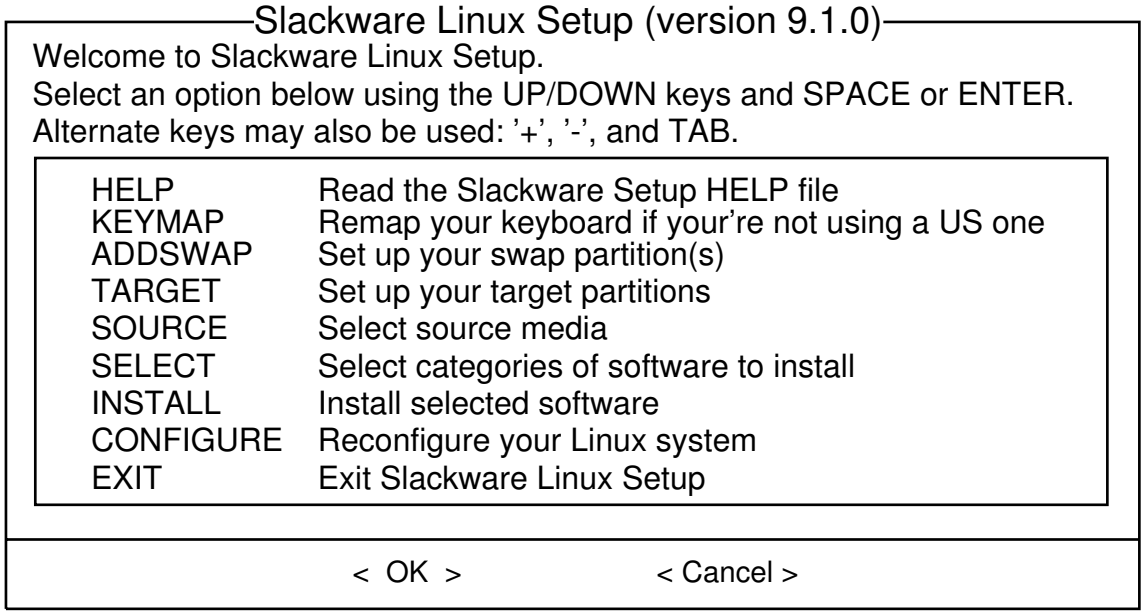
```
Command (m for help):n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4):2
First cylinder (124-1060, default 124):124
Last cylinder or +size or +sizeM or +sizeK (124-1060, default 1060):1060
```

Всё почти готово. Теперь нам необходимо изменить тип первого раздела на 82 (Linux swap). Введите **t**, чтобы изменить тип, выберите первый раздел и наберите 82. Перед записью изменений на диск взгляните напоследок на новую таблицу разделов. Для этого воспользуйтесь в fdisk командой **p**. Если всё выглядит правильно, наберите **w**, чтобы записать изменения на диск и выйти из fdisk.

3.4. Программа установки - *setup*

После создания разделов вы можете приступать к установке Slackware. Следующим этапом процесса установки является запуск программы setup(8). Для этого просто наберите setup в приглашении командного

процессора. Программа `setup` представляет собой систему с меню, которая по сути устанавливает пакеты Slackware и настраивает вашу систему.



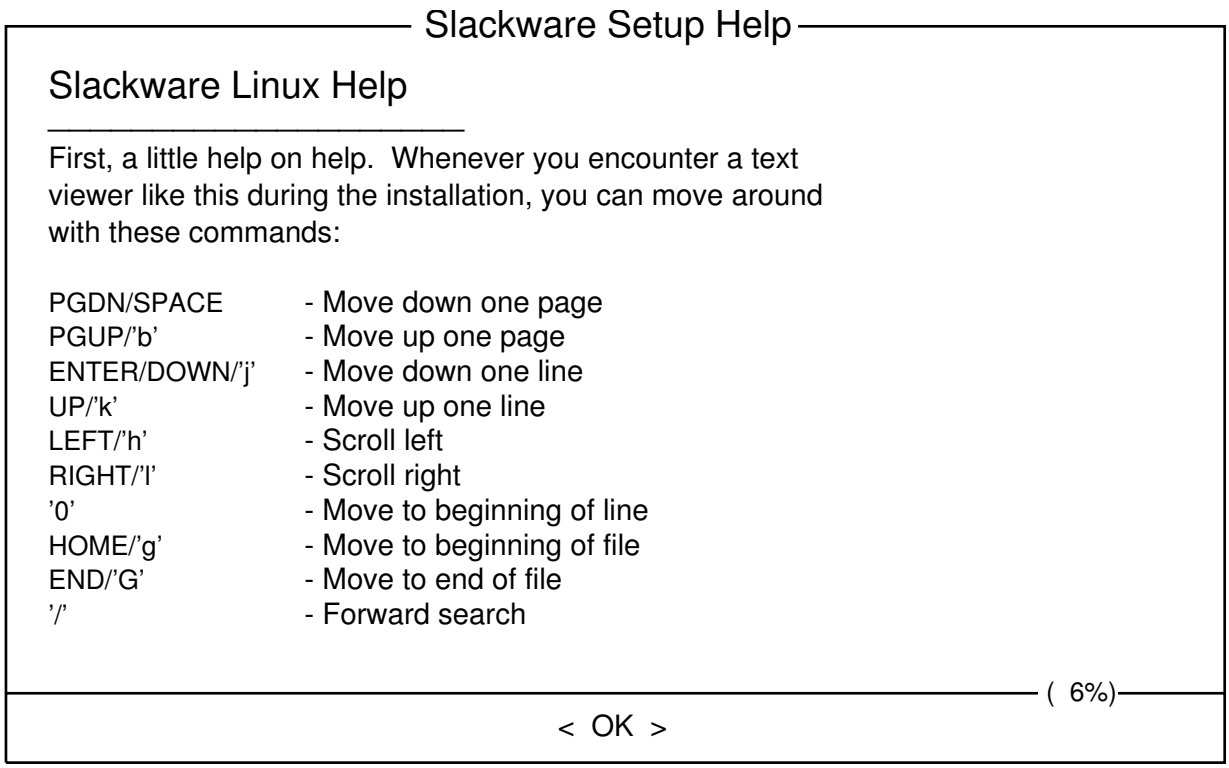
Процесс установки выглядит примерно так: вы проходите все этапы программы `setup` в порядке их перечисления. (Конечно же вы можете выполнять их почти в произвольном порядке, однако есть шанс, что не всё будет потом нормально работать.) Пункты меню выбираются с помощью клавиш со стрелками вверх и вниз, а кнопки "Okay" и "Cancel" затем выбираются при помощи клавиш влево и вправо. С другой стороны любой пункт можно выбрать с помощью соответствующей клавиши, подсвеченной в названии пункта. Опции, имеющие маркеры выбора (т.е. со знаком `[X]`), переключаются с помощью пробела.

Естественно всё это описано в разделе "help" программы `setup`, однако мы считаем нужным дать нашим читателям то, за что они заплатили.

HELP (СПРАВКА)

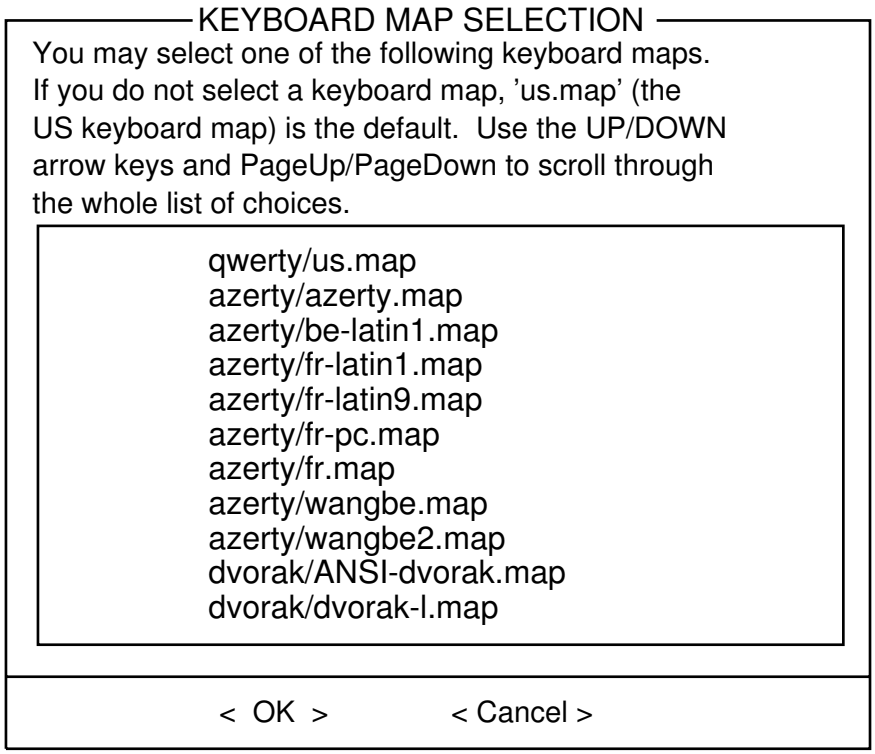
Если это ваша первая установка Slackware, вам может понадобиться

обратиться к окну со справкой. Она даст вам описание каждой части `setup` (довольно похожее на то, что мы сейчас представим, однако менее подробное) и инструкции для навигации при дальнейшей установке.

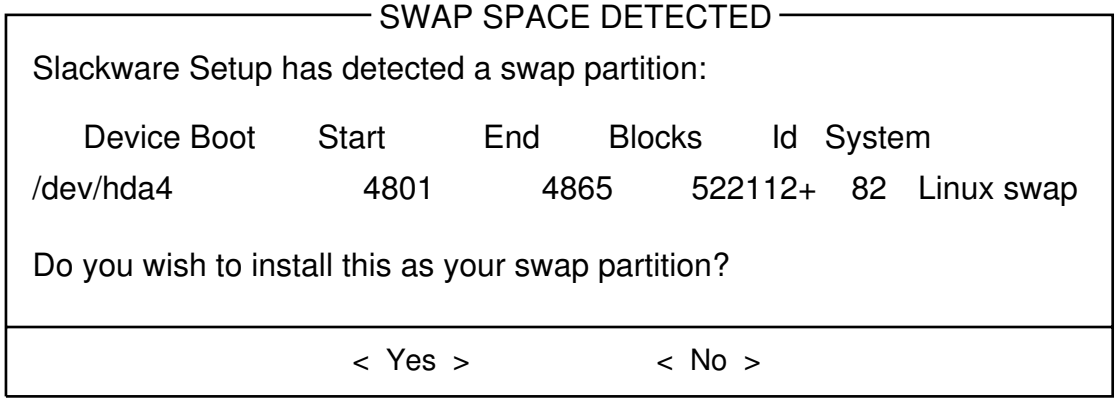


KEYMAP (РАСКЛАДКА КЛАВИАТУРЫ)

Если вам нужна раскладка клавиатуры, отличная от американской "qwerty", вам следует заглянуть в этот раздел. В нём предлагается набор альтернативных раскладок для приятной работы за вашей клавиатурой.



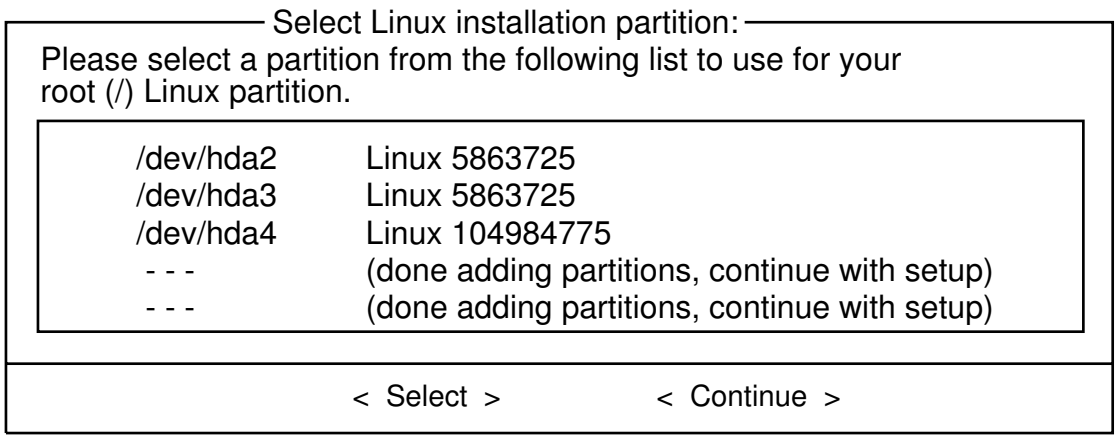
ADDSWAP (ДОБАВИТЬ СВОП)



Если вы создали раздел для свопинга (см. Разд. 3.3), этот раздел позволит вам задействовать его. В нём будут автоматически определены и показаны разделы для свопинга вашего жёсткого диска, позволив вам выбрать один

для того, чтобы отформатировать и включить его.

TARGET (ЦЕЛЬ)

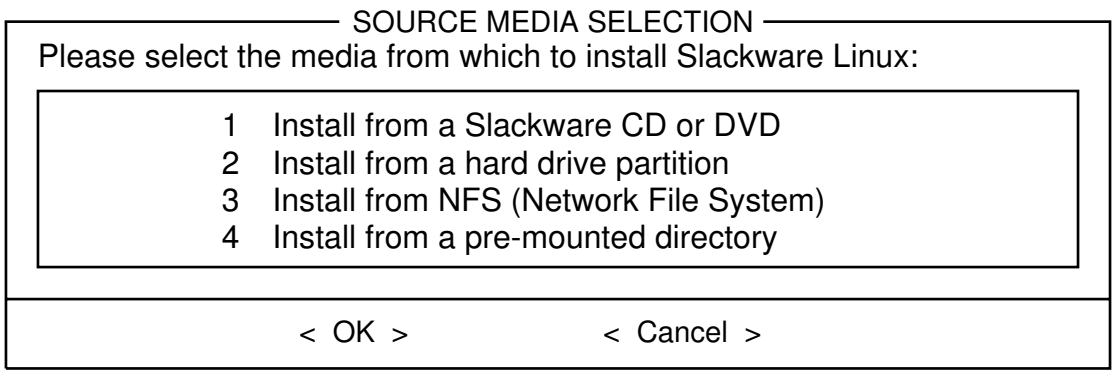


Этот раздел предназначен для форматирования других (не swap) разделов и привязывания их к точкам монтирования файловой системы. На экран будет выведен список разделов вашего жёсткого диска. Для каждого из разделов вам будет дана возможность отформатировать его. В зависимости от используемого ядра файловыми системами могут быть: reiserfs (по умолчанию), ext3, ext2, jfs и xfs. большинство людей используют reiserfs или ext3. В ближайшем будущем мы можем увидеть поддержку и reiserfs4.

Первым пунктом в этом разделе является выбор раздела, в который будет установлена корневая файловая система (/). После этого вы сможете создать файловые системы на других разделах на своё усмотрение. (Например, вы можете создать третий раздел, скажем, `/dev/hda3`, для размещения на нём файловой системы для домашних каталогов. Это просто пример. Создавайте разделы так, как вам нравится.)

SOURCE (ИСТОЧНИК)

В этом разделе вы выбираете источники, с которых будет выполняться установка Slackware. В настоящее время таких источников четыре: CD-ROM, жёсткий диск, NFS или предварительно примонтированный каталог.



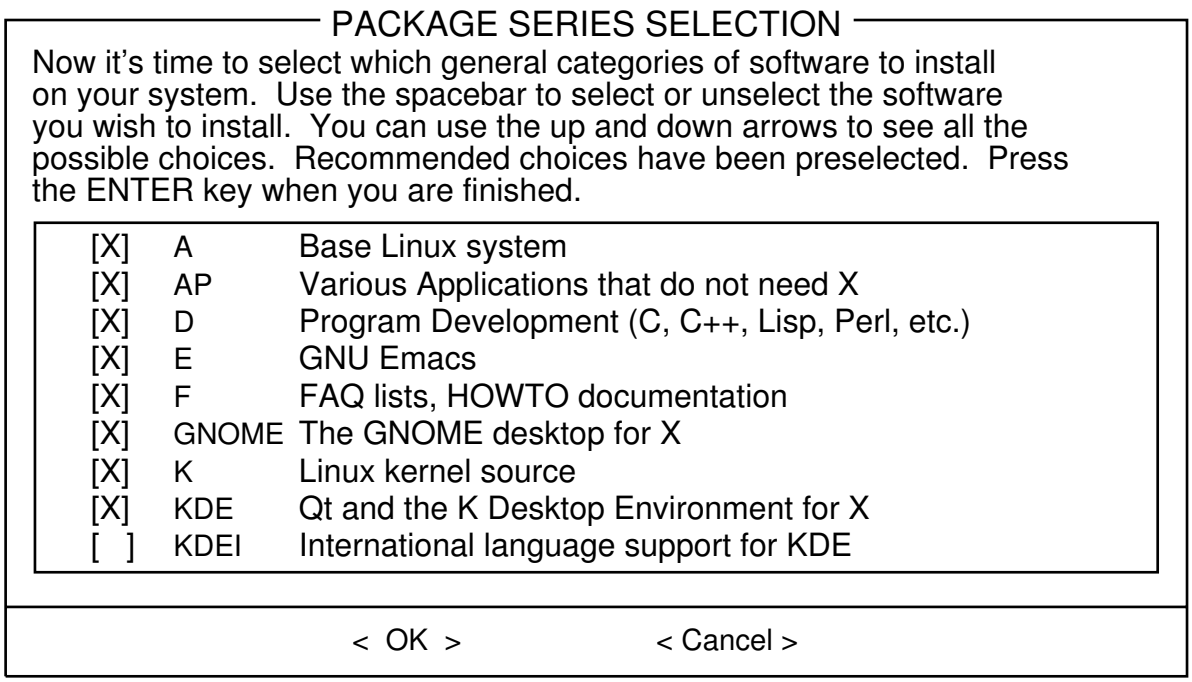
Пункт CD-ROM позволяет выполнить установку с CD-ROM. В нём будет предложено просканировать привод CD-ROM или будет показан список, из которого вы сможете выбрать тип своего привода. Убедитесь, что компакт-диск со Slackware находится в приводе перед тем, как разрешить просканировать его.

В пункте NFS будет запрошена информация о вашей сети и сервере NFS. Соответственно, последний должен быть настроен и доступен в сети. Также учтите, что вы не можете использовать имена хостов: и для своей машины, и для сервера NFS вы должны будете использовать IP-адреса (на диске с setup отсутствует распознаватель имён). Как правило вам потребуется воспользоваться дискетой `network.dsk` для добавления поддержки своего сетевого контроллера.

Предварительно смонтированный каталог предлагает больше возможностей. Вы можете использовать этот способ для установки с таких источников, как Jaz-диски, NFS-ресурсы, примонтированные через PLIP, и файловые систем FAT. Примонтируйте файловую систему в нужное вам место перед запуском setup, а затем укажите здесь это местоположение.

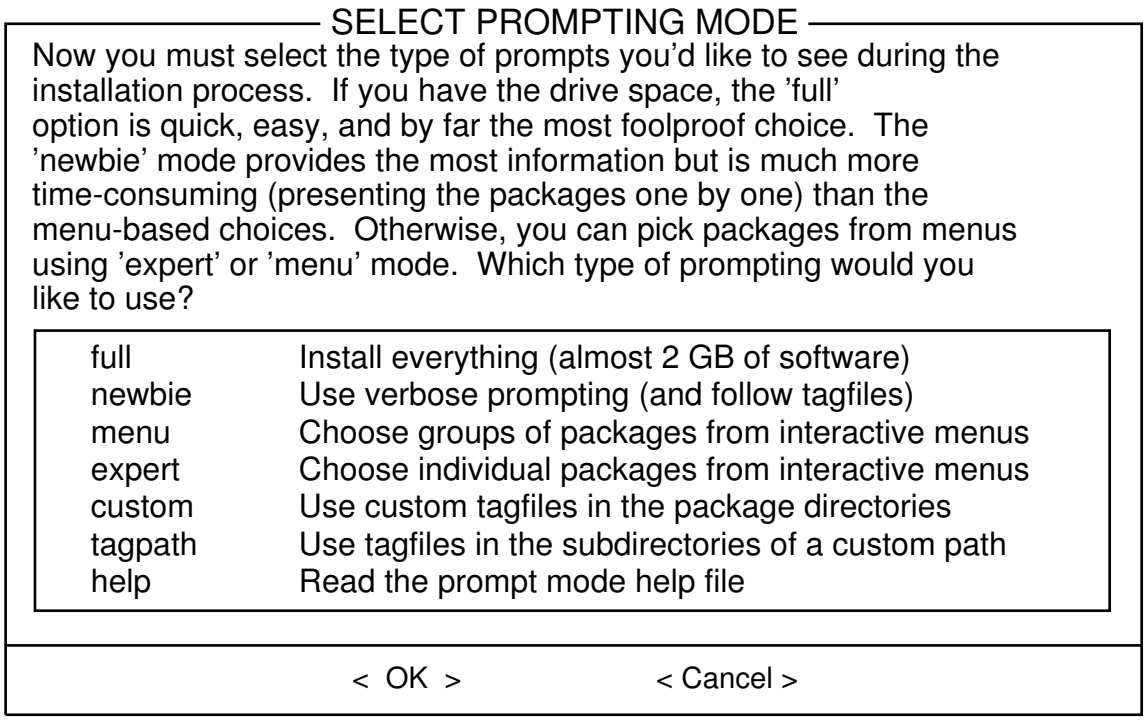
SELECT (ВЫБОР)

Этот раздел позволяет вам выбрать категории устанавливаемого программного обеспечения. Эти категории описаны в Разд. 3.2.1. Пожалуйста, обратите внимание, что вы должны установить категорию A, чтобы получить минимальную рабочую систему. Все остальные категории являются необязательными.



INSTALL (УСТАНОВКА)

При условии, что вы уже прошли этапы "target", "source" и "select", раздел `install` позволит вам выбрать пакеты из выбранных категорий программ. В противном случае вам будет предложено вернуться назад и закончить работу с другими разделами программы setup. Этот раздел позволяет вам выбрать один из шести способов установки: `full`, `newbie`, `menu`, `expert`, `custom` и `tag path`.



Пункт `full` установит все пакеты изо всех категорий программного обеспечения, которые вы выбрали в разделе "select". Дальнейший процесс будет полностью автоматическим. Это самый простой способ установки, поскольку вам не нужно принимать никаких решений об установке того или иного пакета. Конечно же для этого варианта требуется больше всего дискового пространства.

Следующий вариант - `newbie`. Этот пункт устанавливает все необходимые пакеты в выбранных категориях. Для всех остальных пакетов он выводит диалоговое окно, в котором вы можете выбрать "Yes", "No" или "Skip" (Да, Нет, Пропустить). Yes и No говорят сами за себя, а Skip осуществляет переход к следующей категории программ. Вдобавок вы увидите описание и информацию о размере для каждого из пакетов, чтобы вам легче было принять решение об его установке. Мы рекомендуем этот вариант для новых пользователей, т.к. он гарантирует, что вы получите всё необходимое программное обеспечение. Однако он будет более медленным из-за диалогов.

`Menu` - это более быстрый и усовершенствованный вариант пункта для новичков. Для каждой из категорий отображается меню, в котором вы можете выбрать все пакеты, которые вы хотите установить. Требуемые пакеты в этом меню не отображаются.

Для более опытных пользователей этом в разделе предлагается пункт `expert`. Он даёт вам полный контроль над установкой пакетов. Вы можете отменить выбор пакетов, которые являются абсолютно необходимыми, что может привести к получению неработающей системы. С другой стороны вы можете чётко контролировать всё, что происходит в вашей системе. Просто выберите в категориях пакеты, которые вы хотите установить. Этот способ не рекомендуется для новичков, поскольку с его помощью очень легко "выстрелить себе в ногу."

Пункты `custom` и `tag path` также предназначены для опытных пользователей. Эти способы позволяют вам выполнить установку, основанную на собственных tag-файлах, самостоятельно созданных в дереве дистрибутива. Это полезно при установке на большое число машин, т.к. позволяет значительно ускорить процесс. За дополнительной информацией обращайтесь к Разд. 18.4.

После выбора метода установки будет наблюдаться разная реакция. Если вы выбрали full или menu, на экране появится меню, позволяющее вам выбрать пакеты для установки. Если вы выбрали full, начнётся немедленная установка пакетов. Если вы выбрали newbie, пакеты будут устанавливаться до завершения установки необязательных пакетов.

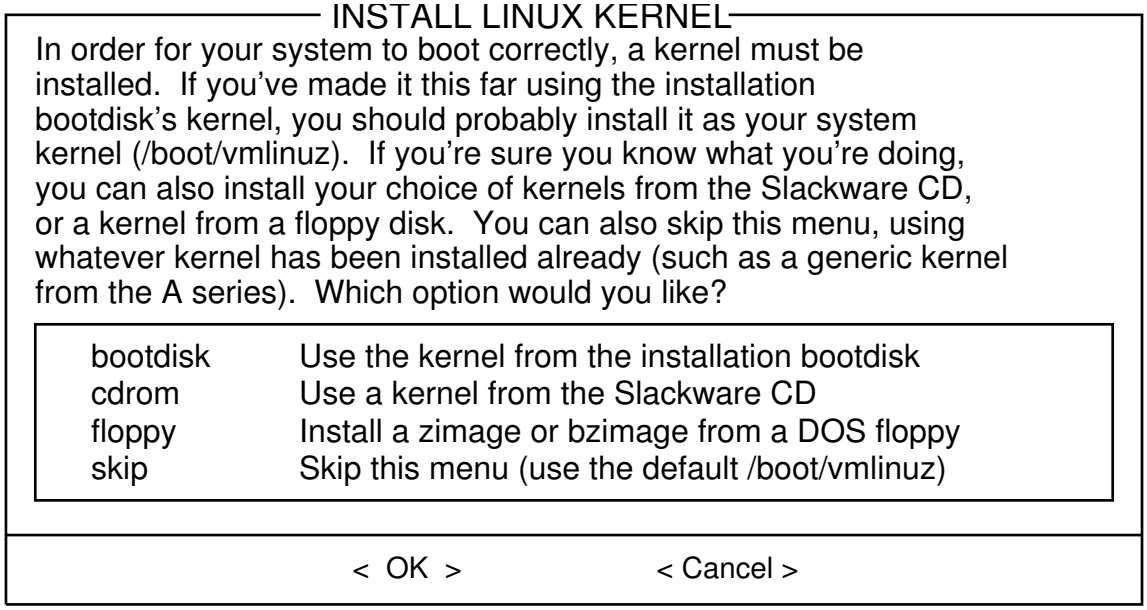
Обратите внимание, что во время установки у вас может закончиться свободное дисковое пространство. Если вы выбрали слишком много пакетов по отношению к свободному месту на диске, у вас возникнут проблемы. Наиболее безопасным способом будет установка некоторого программного обеспечения, плюс последующая доустановка нужных пакетов. Это легко осуществить с помощью утилит управления пакетами Slackware. Об этом вы можете узнать в Гл. 18.

CONFIGURE (НАСТРОЙКА)

Этот раздел позволяет вам выполнить начальную настройку системы после установки пакетов. То, что вы здесь увидите, зависит от от установленного программного обеспечения. Однако в любом случае вы увидите следующее:

Выбор ядра

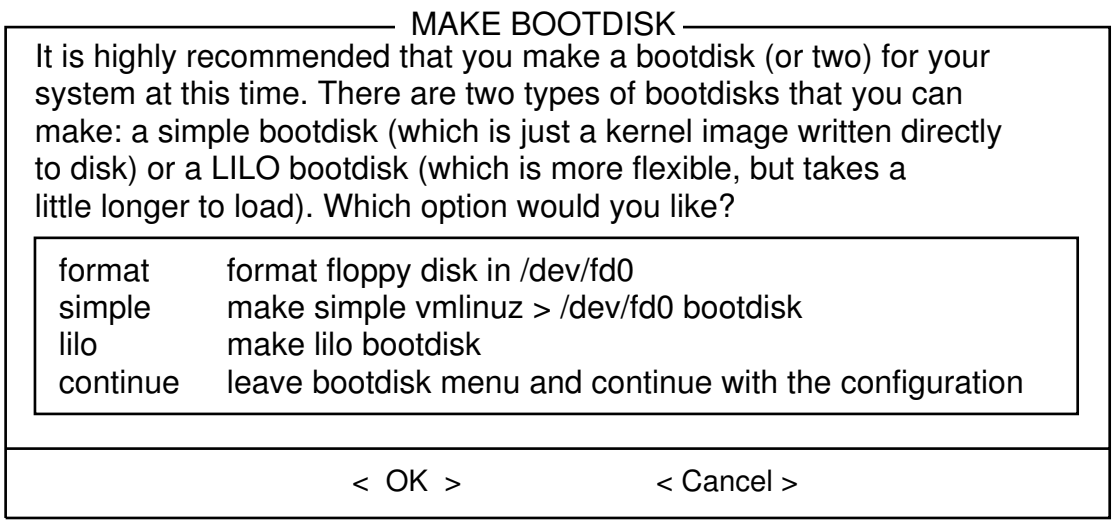
Здесь вам будет предложено выбрать ядро для установки. Вы можете взять ядро с загрузочного диска, который вы использовали при установке, с компакт-диска со Slackware или с другой подготовленной дискеты (если вы, конечно, позаботились об этом заранее). Или же вы можете пропустить этот этап. В этом случае будет установлено ядро по умолчанию.



Создание загрузочного диска

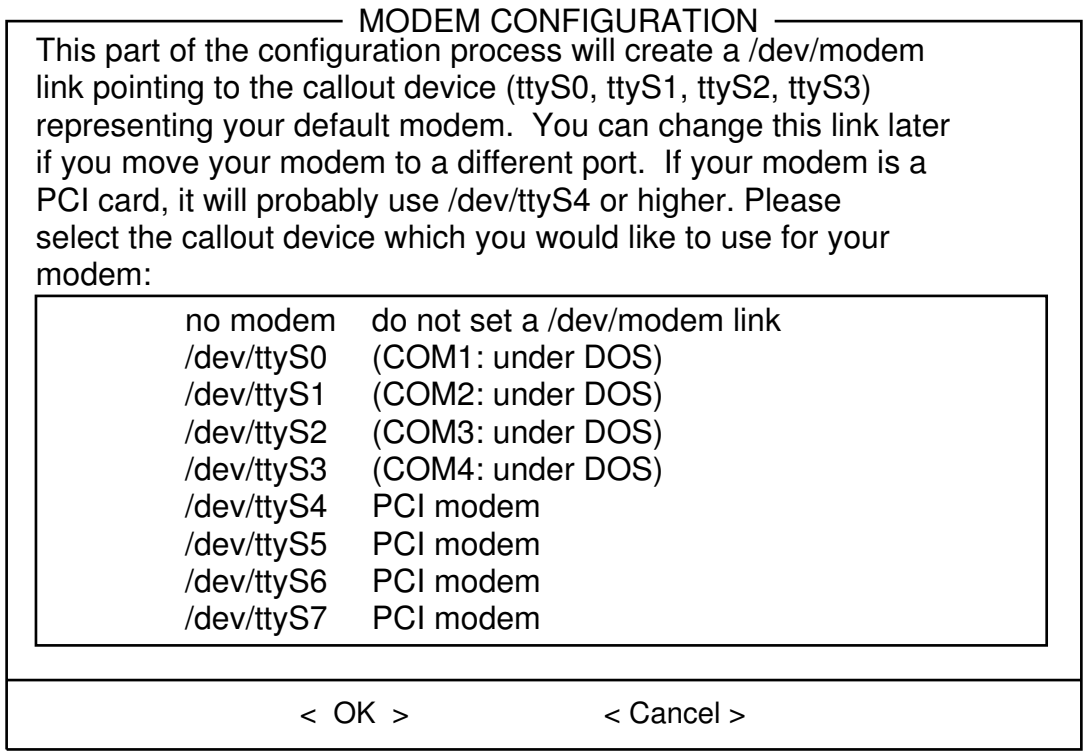
Создание загрузочного диска для использования в будущем - это неплохая идея. У вас есть возможность отформатировать дискету, а затем создать

загрузочный диск одного из двух типов. Первый тип - `простой` - просто записывает (см. рисунок) ядро на дискету. Более гибким (и настоятельно рекомендуемым) вариантом является `lilo`, в котором, конечно же, также будет создан загрузочный диск с lilo. Дополнительную информацию о LILO смотрите в Разд. 7.1. Также вы можете выбрать `continue` (продолжить), при этом загрузочный диск создан не будет.



Модем

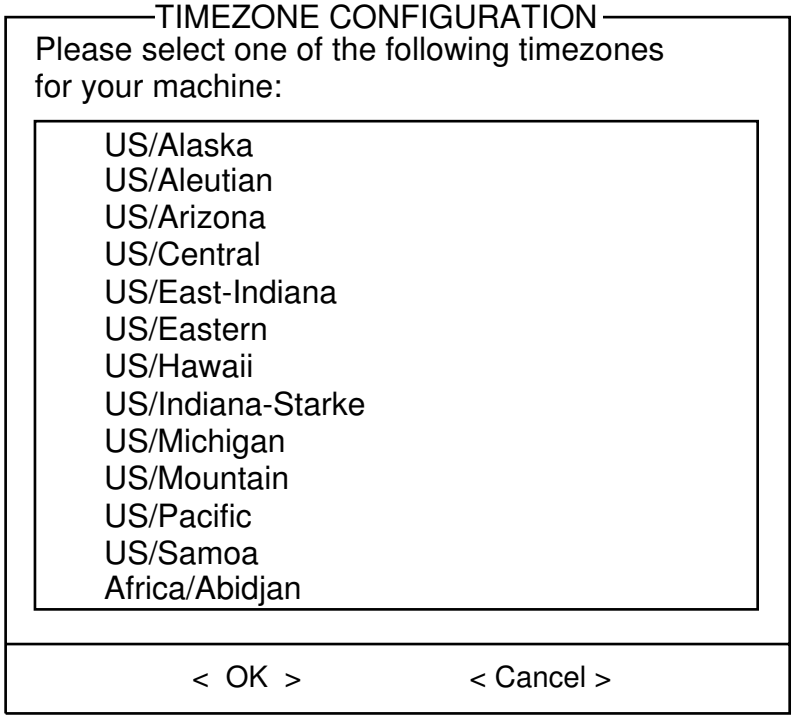
У вас будет запрошена информация о вашем модеме. Если подробнее, то вам будет задан вопрос, если ли у вас вообще модем, и если таковой имеется, то к какому последовательному порту он подключён.



Следующие подразделы конфигурации системы могут и не появиться в зависимости от того, установлены или нет соответствующие пакеты.

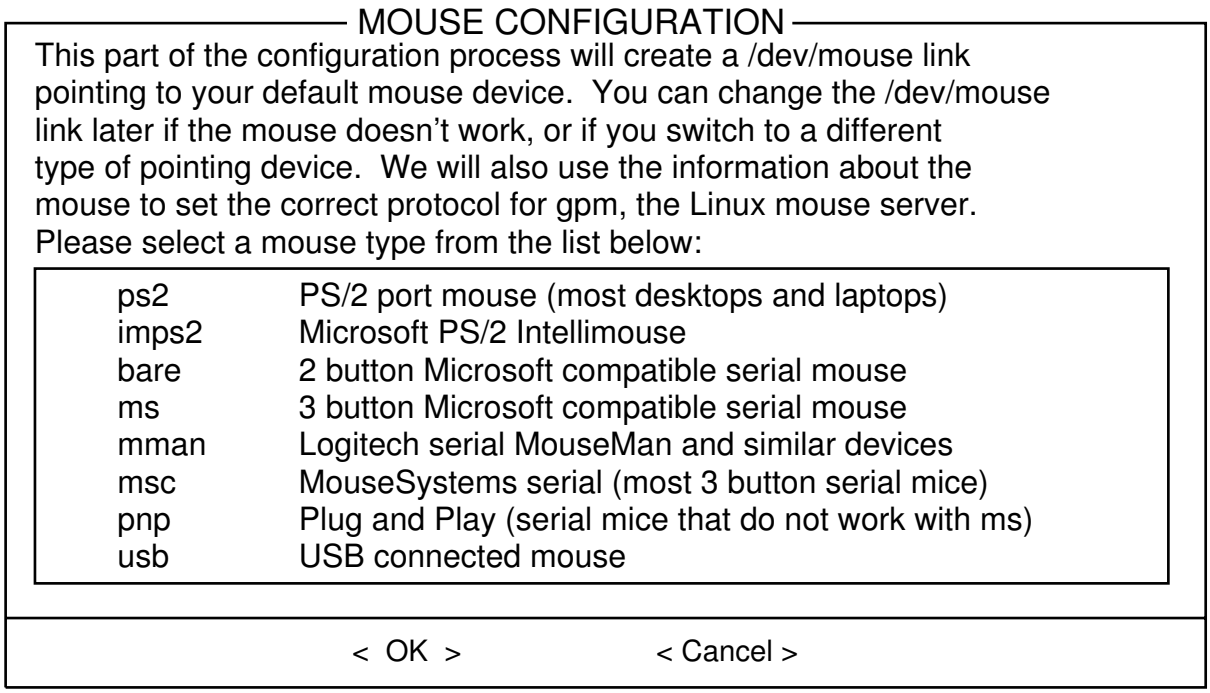
Часовой пояс

Говорит само за себя: в каком часовом поясе вы находитесь. Если вы работаете в часовом поясе Зулу, мы приносим свои извинения, потому что (довольно длинный) список сделан в алфавитном порядке и ваш вариант находится в самом его конце.



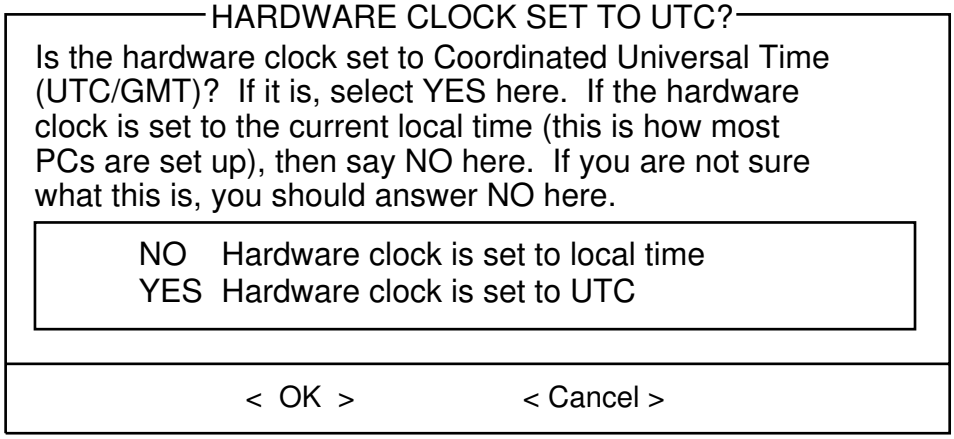
Мышь

В этом подразделе вам надо всего лишь указать тип своей мыши и решить, хотите ли вы задействовать при загрузке включение поддержки мыши в консоли - `gpm`(8).



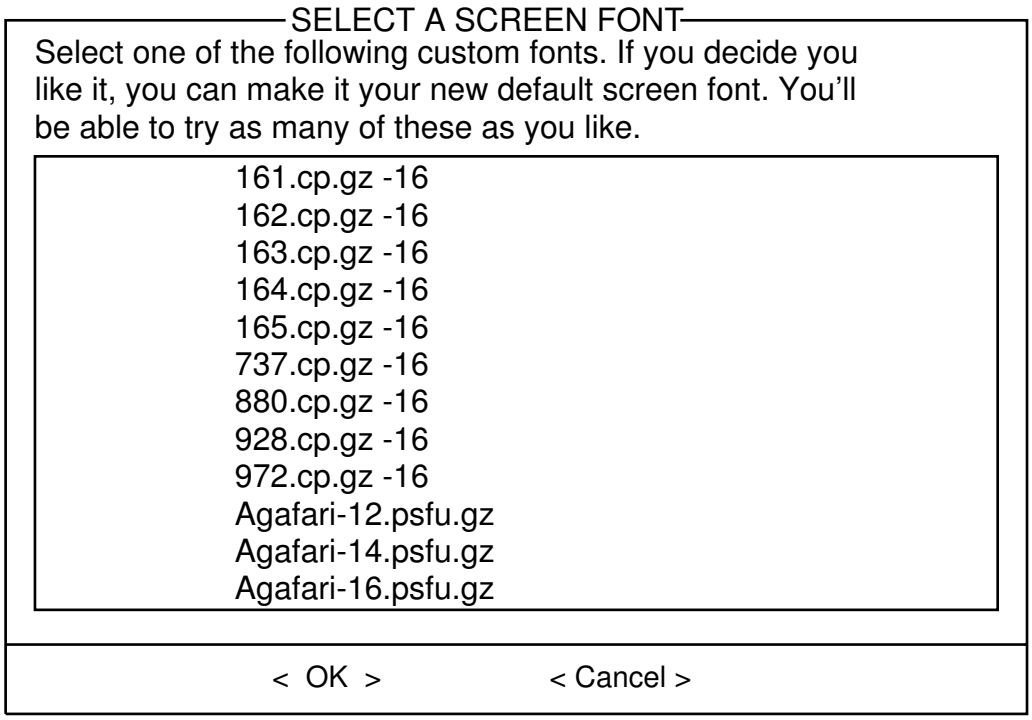
Аппаратные часы

В этом подразделе вам будет задан вопрос, выставлены ли ваши аппаратные часы по всеобщему скоординированному времени (UTC или GMT). В большинстве ПК это не так, поэтому скорее всего ваш ответ будет отрицательным.



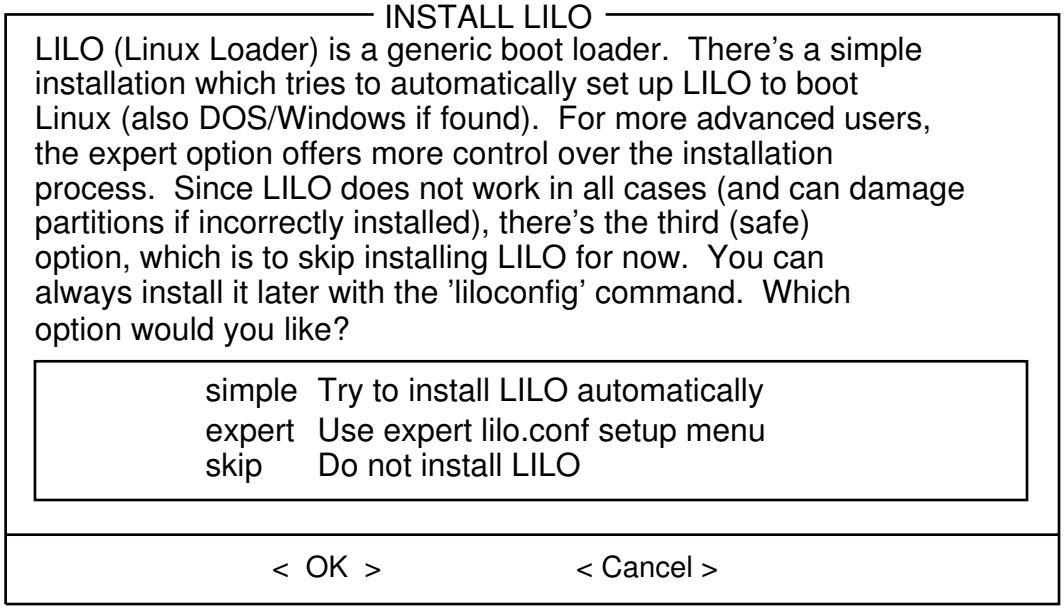
Шрифт

Этот подраздел позволяет вам выбрать из списка шрифт для консоли.



LILO

Здесь вам будет предложено установить LILO (LInux LOader, загрузчик Linux; см. Разд. 7.1).



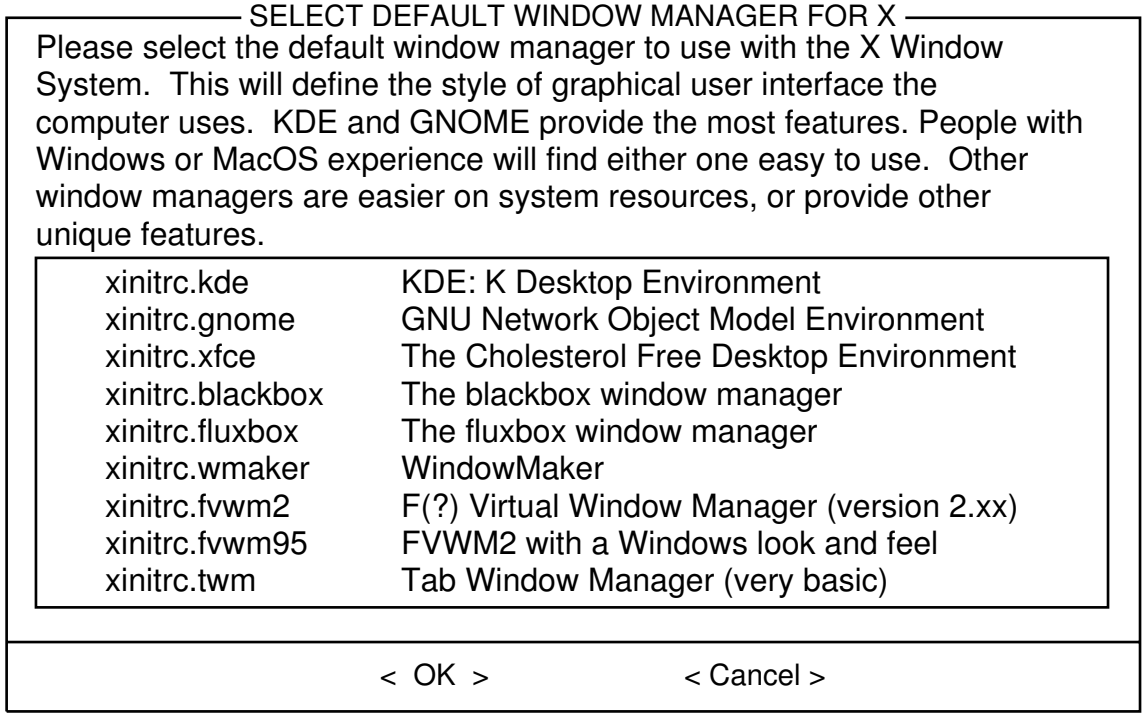
Если Slackware является единственной операционной системой на вашем компьютере, для вас вполне должно хватить пункта `simple`. Если вы используете двойную загрузку, тогда лучше выбрать `expert`. Дополнительную информацию о двойной загрузке смотрите в Разд. 7.3. Третий вариант - `do not install` - рекомендуется только в том случае, если вы понимаете, что делаете, и у вас есть веская причина для отказа от установки LILO. Если вы выполняете установку в режиме эксперта, вам будут предложены варианты местоположений для установки LILO. Вы можете установить его в MBR (Master Boot Record, главная загрузочная запись) своего жёсткого диска, в суперблок корневого раздела Linux или на дискету.

Сеть

Подраздел настройки сети по сути представляет собой работу с программой `netconfig`. Подробную информацию смотрите в Разд. 5.1.

Менеджер X Window

Этот подраздел позволит вам выбрать оконный менеджер по умолчанию для X. Более подробную информацию об X и оконных менеджерах смотрите в Гл. 6.



Не имеет значения, какие пакеты вы устанавливали, последним этапом настройки будет вопрос, хотите ли вы продолжить и установить пароль `root`'а. По соображениям безопасности это, скорее всего, будет хорошей идеей, однако (как практически и всё в Slackware) это зависит только от вас.

Глава 4.

Настройка системы

Перед тем, как вы сможете настроить более сложные части своей системы, было бы неплохо изучить её организацию и команды, которые могут быть использованы для поиска файлов и программ. Также хорошо бы было знать, нужно ли вам компилировать своё ядро и какие шаги нужно выполнить для этого. Эта глава познакомит вас с организацией системы и её конфигурационными файлами. Затем вы сможете перейти к настройке более сложных частей своей системы.

4.1. Обзор системы

Важно понять, как система Linux собирается воедино, перед тем как погружаться в различные аспекты её настройки. Linux значительно отличается от систем DOS, Windows или Macintosh (за исключением Mac OS на базе Unix), но эти разделы помогут вам разобраться в этой ситуации, чтобы вы легко смогли настроить свою систему до получения нужного результата.

Структура файловой системы

Первой существенной разницей между Slackware Linux и системами DOS или Windows является файловая система. Специально для новичков: мы не используем буквы дисков для идентификации различных разделов. В Linux существует один главный каталог. Вы можете сравнить его с диском `C:` в DOS. Все разделы в вашей системе монтируются в подкаталоги главного

каталога. Это можно считать неким подобием постоянно расширяющегося жёсткого диска.

Мы называем главный каталог корневым и обозначается он одной косой чертой (/). Такой подход может показаться странным, однако он значительно упрощает жизнь, когда вам нужно увеличить дисковое пространство. Допустим, например, что у вас не хватает места на диске, на котором находится каталог `/home`. Большинство людей устанавливают Slackware и делают один большой корневой раздел. Поскольку раздел может быть примонтирован в любой каталог, вы можете просто пойти в магазин, приобрести новый жёсткий диск и примонтировать его в `/home`. Теперь в вашей системе появилось новое дисковое пространство. И всё это без особых затруднений.

Ниже представлены описания главных каталогов верхнего уровня, имеющихся в Slackware.

`bin`

В нём находятся важнейшие пользовательские программы. Они представляют собой минимальный набор программ, необходимый пользователю для работы в системе. Это программы наподобие командных процессоров и команд для работы с файловой системой (`ls`, `cp` и т.п.). Каталог `/bin` обычно не подвергается изменениям после установки системы. Если это происходит, то, как правило, в виде предоставляемых нами обновлений пакетов.

`boot`

Файлы, используемые загрузчиком Linux (LILO). В этом каталоге также происходят небольшие изменения после установки системы. В нём находится ядро, начиная со Slackware версии 8.1. В предыдущих релизах ядро хранилось просто в `/` , однако распространённой практикой является помещение ядра и связанных файлов именно в этот каталог для упрощения двойной загрузки.

`dev`

В Linux всё является файлом, даже аппаратное обеспечение, наподобие последовательных портов, жёстких дисков и сканеров. Для получения доступа к этому оборудованию должен присутствовать файл, называемый узлом устройства. Все узлы устройств находятся в каталоге `/dev`. Вы найдёте это справедливым для многих Unix-подобных операционных систем.

`etc`

В этом каталоге находятся конфигурационные файлы системы: от конфигурационных файлов X Window, базы данных пользователей, до загрузочных скриптов системы. Со временем системный администратор очень хорошо познакомится с этим каталогом.

`home`

Linux является многопользовательской операционной системой. Каждый пользователь в системе имеет уникальную учётную запись и отдельный каталог для личных файлов. Этот каталог также называется домашним каталогом пользователя. Каталог `/home` предназначен как раз для размещения в нём домашних каталогов пользователей.

`lib`

Здесь находятся системные библиотеки, необходимые для базовых операций и работы системы. Среди всего прочего в нём находятся библиотека C, динамический компоновщик, библиотека ncurses и модули ядра.

`mnt`

В этом каталоге находятся временные точки монтирования для работы с жёскими дисками и съёмными накопителями. В нём вы найдёте точки монтирования своих приводов CD-ROM и дисководов.

`opt`

Пакеты с опциональным программным обеспечением. Идея, лежащая в основе `/opt`, заключается в том, чтобы каждый программный пакет устанавливался в `/opt/software-package`, что упрощает его удаление в будущем. Slackware помещает в `/opt` некоторые программы (как, например, KDE в `/opt/kde`), однако вы свободно можете помещать в него всё, что угодно.

`proc`

Это уникальный каталог. Он является не реальной, а виртуальной частью файловой системы, которая предоставляет доступ к информации о ядре. Различные куски информации, о которых (по мнению ядра) вы хотите знать, представлены в виде файлов в каталоге `/proc`. С помощью некоторых из этих файлов вы можете также отправлять информацию в ядро. Попробуйте выполнить `cat /proc/cpuinfo`.

`root`

Системный администратор известен в системе как `root`. Домашним каталогом `root`'а является `/root`, а не `/home/root`. Причина довольно проста. Что будет, если `/home` находится на отдельном разделе (не на `/`) и не может быть примонтирован? `root`'у скорее всего потребуется войти в систему и восстановить её работоспособность. Если его домашний каталог находится на повреждённой файловой системе, у него возникнут трудности со входом в систему.

`sbin`

Здесь находятся важнейшие программы, запускаемые `root`'ом и во время процесса загрузки системы. Обычные пользователи не запускают программы из этого каталога.

`tmp`

Место для хранения временных данных. Все пользователи имеют право на чтение и запись в этот каталог.

`usr`

Это большой каталог в системе Linux. Практически всё остальное находится именно в нём: программы, документация, исходный код ядра и система X Window. Это каталог, в который вы скорее всего будете устанавливать программы.

`var`

Здесь находятся файлы журналов системы, кэшируемые данные и блокировочные файлы программ. Этот каталог предназначен для часто изменяющихся данных.

Теперь вы хорошо осведомлены о каталогах файловой системы и о том, что они содержат. Более подробная информация о структуре файловой системы доступна на странице руководства hier(7). Следующий раздел поможет вам легко находить нужные файлы без необходимости делать это вручную.

Поиск файлов

Теперь вы знаете, что находится в главных каталогах системы, но это всё-таки не слишком-то и поможет вам найти среди них то, что вам нужно. Имеется в виду, что вы можете вручную рыться в каталогах, однако для этого есть более быстрые методы. Для поиска файлов в Slackware есть четыре основные команды.

which

Первая команда - это `which`(1). `which` обычно используется для быстрого

поиска программ. Она просто выполняет поиск по переменной окружения `PATH` и возвращает первый найденный результат, а также путь к нему. Взгляните на этот пример:

```
%  which bash
/bin/bash
```

В нём видно, что `bash` находится в каталоге `/bin`. Это очень ограниченная команда, т.к. она выполняет поиск только в вашей переменной `PATH`.

whereis

Команда `whereis`(1) работает примерно как и `which`, однако она также ищет страницы руководства и файлы с исходными текстами. Поиск `bash` с помощью `whereis` должен дать следущее:

```
%  whereis bash
bash: /bin/bash /usr/bin/bash /usr/man/man1/bash.1.gz
```

Эта команда сообщила нам не только, где находится программа, а также и местонахождение документации к ней. Однако эта команда всё ещё ограниченна. Что, если вам нужно найти определённый конфигурационный файл? Для этого вы не можете использовать ни `which`, ни `whereis`.

find

Команда `find`(1) позволяет пользователю выполнять поиск по файловой системе с помощью большого набора поисковых аргументов. Пользователи могут искать файлы по именам с использованием шаблонов подстановки, диапазонов времени их изменения или создания и других расширенных свойств. Например, чтобы найти в системе файл `xinitrc`, можно воспользоваться следующей командой.

```
%  find / -name xinitrc
/var/X11R6/lib/xinit/xinitrc
```

Выполнение `find` займёт некоторое время, поскольку она должна выполнить сквозной поиск по всему дереву файлов, начиная с корня. Также, если эта команда будет выполнена обычным пользователем, на экран будут выводиться сообщения об ошибке при попытке зайти в каталоги, доступ к которым есть только у `root`'а. Однако `find` нашла наш файл, так что всё в порядке. Вот только если бы она была чуточку быстрее...

slocate

Команда `slocate`(1) выполняет поиск по всей файловой системе примерно как и find, однако поиск ведётся по базе данных, а не по самой файловой системе. Обновление базы данных выполняется автоматически каждое утро, чтобы у вас был более свежий список файлов своей системы. Вы можете вручную запустить `updatedb`(1) для обновления базы данных slocate (перед этим вы должны сначала получить права `root`'а с помощью `su`). Вот пример работы `slocate`:

```
%  slocate xinitrc   # нам не нужно получать права root'а
/var/X11R6/lib/xinit/xinitrc
/var/X11R6/lib/xinit/xinitrc.fvwm2
/var/X11R6/lib/xinit/xinitrc.openwin
/var/X11R6/lib/xinit/xinitrc.twm
```

Мы получили даже больше, чем искали, и довольно быстро. С помощью этих команд вы сможете найти в своей системе Linux всё, что угодно.

Каталог `/etc/rc.d`

В каталоге `/etc/rc.d` находятся файлы инициализации системы. В Slackware

для файлов инициализации используется схема в стиле BSD, а не System V, которая стремится усложнить изменение конфигурации без использования программ, специально разработанных для этих целей. В init-скриптах BSD каждый уровень запуска определён в одном единственном rc-файле. В System V для каждого уровня запуска выделен отдельный каталог, содержащий большое число скриптов инициализации. Таким образом создаётся организованная и простая в обслуживании структура.

Существует несколько разных категорий файлов инициализации: запуск системы, уровни запуска, инициализация сети и совместимость с System V. По традиции всё остальное мы свалим в другую категорию.

Запуск системы

Первая программа, запускающаяся в Slackware помимо ядра Linux - это `init`(8). Эта программа читает файл `/etc/inittab`(5), чтобы узнать, как запускать систему. Она запускает скрипт `/etc/rc.d/rc.S`, чтобы подготовить систему перед тем, как перейти на нужный вам уровень запуска. Файл `rc.S` активирует виртуальную память, монтирует файловые системы, очищает определённые каталоги с журналами, инициализирует устройства Plug-and-Play, загружает модули ядра, настраивает устройства PCMCIA, поднимает последовательные порты и запускает init-скрипты System V (если таковые найдены). Очевидно, что `rc.S` содержит много информации, однако в `/etc/rc.d` есть несколько скриптов, которые `rc.S` вызовет для завершения своей работы:

`rc.S`

: Это скрипт инициализации самой системы.

`rc.modules`

: Загружает модули ядра. С его помощью поднимаются сетевые карты, поддержка PPP и другие службы. Если этот скрипт находит `rc.netdevice`, он также запустит и его.

`rc.pcmcia`

Сканирует и настраивает любые устройства PCMCIA, которые могут присутствовать в вашей системе. Это наиболее полезно для пользователей ноутбуков, у которых наверняка имеется модем или сетевая карта PCMCIA.

`rc.serial`

Настраивает последовательные порты, запуская соответствующие команды `setserial`.

`rc.sysvinit`

Ищет скрипты инициализации System V для нужного уровня запуска и запускает их. Более подробно это описано ниже.

Скрипты инициализации уровня запуска

После того, как завершена инициализация системы, `init` приступает к инициализации уровня запуска. Уровень запуска описывает состояние, в котором будет работать ваша машина. Звучит слишком сложно? Другими словами, уровень запуска сообщает `init`'у, будете ли вы допускать одновременную работу в системе нескольких пользователей или только одного, будут ли запущены сетевые службы и что вы будете использовать для входа в систему: систему X Window System или `agetty`(8). Представленные ниже файлы определяют различные уровни запуска в Slackware Linux.

`rc.0`

Останавливает систему (уровень запуска 0). По умолчанию он является символической ссылкой на `rc.6`.

`rc.4`

Многопользовательская работа (уровень запуска 4), но в X11 с KDM, GDM или XDM в качестве менеджера входа в систему.

`rc.6`

Перезагрузка системы (уровень запуска 6).

`rc.K`

Работа в однопользовательском режиме (уровень запуска 1).

`rc.M`

Многопользовательский режим (уровни запуска 2 и 3), но со стандартным текстовым входом в систему. Это в Slackware уровень загрузки по умолчанию .

Инициализация сети

Уровни запуска 2, 3 и 4 запустят сетевые службы. Следующие файлы отвечают за инициализацию сети:

`rc.inet1`

Этот файл, созданный `netconfig`'ом, отвечает за настройку сетевых интерфейсов.

`rc.inet2`

Запускается после `rc.inet1` и запускает основные сетевые службы.

`rc.atalk`

Запускает службы AppleTalk.

`rc.httpd`

Запускает веб-сервер Apache. Как и некоторые другие rc-скрипты он может быть использован только для остановки и перезапуска службы. Скрипт `rc.httpd` принимает аргументы stop, start или restart.

`rc.news`

Запускает сервер новостей.

Совместимость с System V

Совместимость с инициализацией System V была реализована в Slackware, начиная с версии 7.0. Многие другие дистрибутивы Linux используют этот метод вместо BSD-стиля. В общих чертах каждому уровню запуска соответствует отдельный подкаталог со скриптами инициализации, в то время как BSD-стиль предоставляет один скрипт для каждого уровня запуска.

Скрипт `rc.sysvinit` будет искать любые init-скрипты System V, находящиеся в `/etc/rc.d`, и запустит их, если выбран соответствующий уровень запуска. Это полезно для определённых пакетов с коммерческим программным обеспечением, которые устанавливают скрипты System V.

Другие файлы

Описанные ниже скрипты относятся к другим скриптам инициализации системы. Обычно они запускаются из одного из главных скриптов, описанных выше, поэтому всё, что вам нужно сделать, это отредактировать их содержимое.

`rc.gpm`

Запускает службы общего назначения для работы с мышью, позволяющие вам копировать и вставлять текст в консоли Linux.

Изредка gpm может вызвать проблемы с мышью при работе в X Windows. Если у вас возникают проблемы с мышью в X'ах, попробуйте убрать разрешение на выполнение с этого файла и остановить сервер gpm.

`rc.font`

Загружает определённый экранный шрифт для консоли.

`rc.local`

Содержит определённые команды запуска для вашей системы. После чистой установки этот файл пуст, т.к. он зарезервирован для локальных администраторов. Этот скрипт запускается после всех инициализаций.

Чтобы задействовать этот скрипт, вам нужно дать ему разрешение на выполнение с помощью команды `chmod`. Чтобы отключить скрипт, снимите с него это разрешение. Дополнительная информация о `chmod` доступна в Разд. 9.2.

4.2. Выбор ядра

Ядро является частью операционной системы и оно обеспечивает доступ к оборудованию, управление процессами и всей системой в целом. Ядро обеспечивает поддержку вашего аппаратного обеспечения, поэтому выбор ядра является важным этапом настройки.

Slackware предоставляет вам на выбор более десятка предварительно откомпилированных ядер, каждое со стандартным набором драйверов и дополнительными специальными драйверами. Вы можете использовать одно из готовых ядер или собрать из исходных текстов своё собственное. В любом случае вам необходимо убедиться, что ваше ядро имеет поддержку

необходимого оборудования в вашей системе.

Каталог `/kernels` на CD-ROM со Slackware

Предварительно откомпилированные ядра Slackware доступны в каталоге `/kernels` на компакт-диске со Slackware или на FTP-сайте в главном каталоге Slackware. Доступные ядра изменяются по мере выхода новых релизов, поэтому документация в этом каталоге всегда является самым достоверным источником информации. В каталоге `/kernels` присутствуют отдельные подкаталоги для каждого из ядер. Названия подкаталогов соответствуют названиям соответствующих загрузочных дисков. Во всех подкаталогах вы найдёте следующие файлы:

Файл	Назначение
`System.map`	Файл карты системы для этого ядра
`bzImage`	Собственно, образ ядра
`config`	Конфигурационный файл для исходных текстов этого ядра

Чтобы использовать ядро, скопируйте файлы `System.map` и `config` в каталог `/boot`, а ядро скопируйте в `/boot/vmlinuz`. Запустите `/sbin/lilo`(8), чтобы установить LILO для нового ядра, а затем перезагрузите свою систему. Вот и всё. Новое ядро установлено.

Ядра, имена которых заканчиваются на .i, обладают поддержкой IDE. Т.е. они не поддерживают SCSI, как это реализовано в базовом ядре. Ядра, имена которых заканчиваются на .s, обладают поддержкой SCSI. Они также поддерживают IDE, как и в .i-ядрах.

Компиляция ядра из исходных текстов

Новички часто задают вопрос: “Следует ли мне собрать новое ядро для своей системы?” Ответ звучит довольно чётко. Существует несколько

случаев, когда вам нужно собрать специальное ядро для своей системы. Большинство пользователей могут использовать предварительно откомпилированные ядра в сочетании с загружаемыми модулями для получения полностью работающей системы. Вам понадобится скомпилировать ядро для своей системы, если вы обновляетесь до ядра, версия которого на данный момент отсутствует в Slackware, или если вы пропатчили исходный код ядра, чтобы добиться поддержки особого оборудования, которая отсутствует в оригинальных исходных текстах ядра. Любой пользователь двухпроцессорной системы наверняка захочет собрать ядро с поддержкой SMP. Также многие пользователи найдут работу самостоятельно собранного ядра на своей машине более быстрой. Кроме того вы можете найти полезным собрать ядро с оптимизацией по свою модель процессора.

Самостоятельная сборка ядра не настолько уж и сложна. Первым этапом является проверка того, установлены ли в вашей системе исходные тексты ядра. Убедитесь, что во время установки системы вы установили пакеты из категории K. Также необходимо убедиться, что вы установили пакеты из категории D, в особенности компилятор C, GNU make и GNU binutils. Вообще неплохо было бы полностью установить всю категорию D, если вы планируете заниматься какого-либо вида разработкой. Вы также можете загрузить последнюю версию исходного кода ядра с сервера `http://www.kernel.org/mirrors`.

Компиляция ядра Linux версии 2.4.x

```
% su -
Password:
# cd /usr/src/linux
```

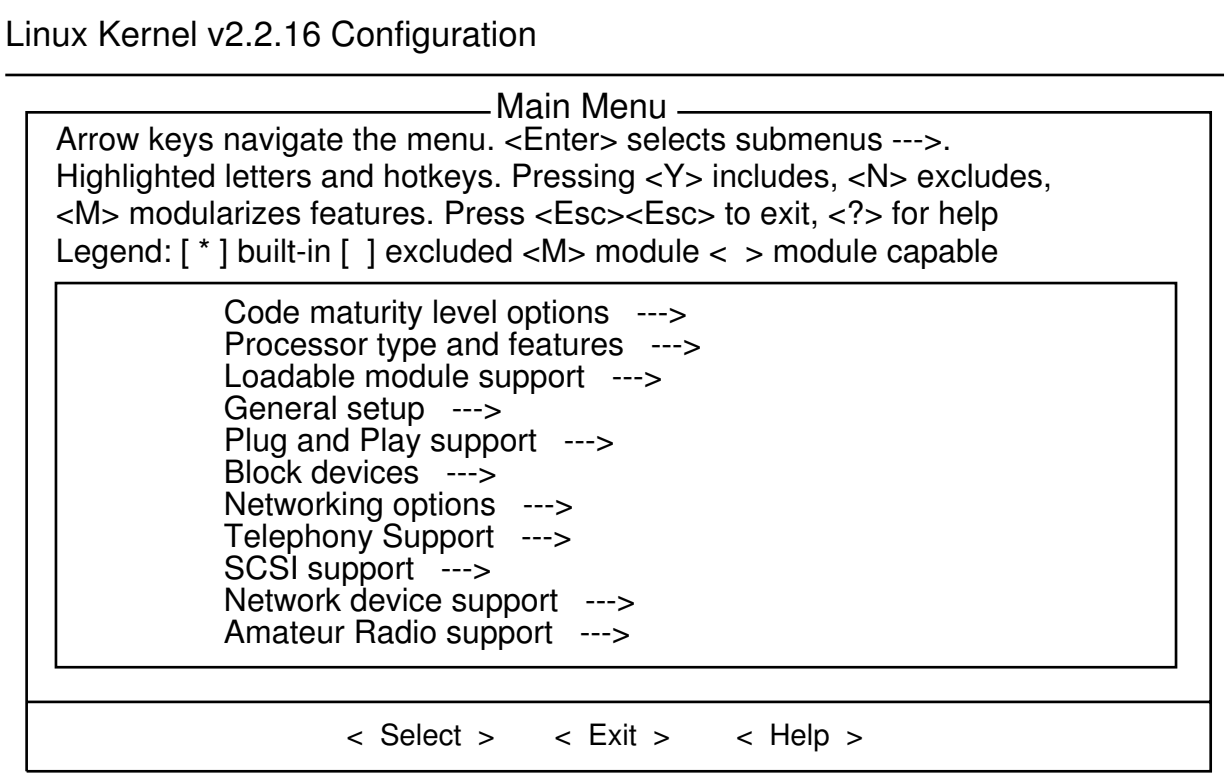
Первым делом нужно привести исходный код в его базовое состояние. Для этого воспользуемся следующей командой (обратите внимание, что вам может понадобится сделать резервную копию файла `.config`, поскольку эта команда без предупреждения удалит его):

```
# make mrproper
```

Теперь вы можете сконфигурировать ядро для своей системы. Для текущей версии ядра для этого предлагаются три варианта. Первым является простая система вопросов и ответов в текстовом режиме. В ней вам задаётся целая серия вопросов, а затем создаётся конфигурационный файл. Проблема этого способа заключается в том, что, если вы где-то ошиблись, вам придётся начать всё заново. Наиболее предпочитаемым способом для большинства пользователей является система с использованием меню. И в завершение, есть ещё графическая утилита для конфигурации ядра. Выберите любой понравившийся вам способ и выполните соответствующую команду:

```
# make config              (текстовая версия "вопрос-ответ")
# make menuconfig          (с использованием меню; текстовая версия)
# make xconfig             (графическая версия; убедитесь сначала, что вы работаете в X'ах)
```

Рисунок 4-1. Меню конфигурирования ядра



Новички наверняка найдут использование `menuconfig` самым простым и лёгким. Доступны экраны с подсказкой, поясняющие различные части ядра. После завершения настройки ядра выйдите из программы-конфигуратора. Она сама запишет все необходимые конфигурационные файлы. Теперь можно подготовить для сборки дерево с исходными текстами:

```
# make dep
# make clean
```

Следующим этапом является компиляция ядра. Сначала попробуйте выполнить команду `bzImage`, представленную ниже.

```
# make bzImage
```

В зависимости от скорости вашего процессора эта операция может занять

продолжительное время. Во время процесса сборки вы будете видеть сообщения компилятора. После создания образа ядра вам потребуется скомпилировать все части ядра, которые вы отметили как модульные.

```
# make modules
```

Теперь мы можем установить скомпилированные ядро и модули. Чтобы установить ядро в системе Slackware, необходимо выполнить следующие команды:

```
# mv /boot/vmlinuz /boot/vmlinuz.old
# cat arch/i386/boot/bzImage > /vmlinuz
# mv /boot/System.map /boot/System.map.old
# cp System.map /boot/System.map
# make modules_install
```

Вам потребуется отредактировать `/etc/lilo.conf` и добавить раздел для загрузки своего старого ядра на тот случай, если новое не заработает. После этого выполните `/sbin/lilo`, чтобы установить новую загрузочную запись. Теперь вы можете выполнить перезагрузку и загрузить своё новое ядро.

Ядро Linux версии 2.6.x

Компиляция ядра версии 2.6 лишь немногим отличается от сборки ядер версии 2.4 или 2.2, однако важно, чтобы вы поняли разницу, перед тем погружаться в этот процесс. Больше нет необходимости выполнять `make dep` и `make clean`. Также процесс компиляции ядра серии 2.6 не настолько подробен. Следствием этого является то, что процесс сборки стал более понятным, однако имеются некоторые небольшие добавки. Если у вас возникают проблемы со сборкой ядра, настоятельно рекомендуется снова включить подробный режим. Для этого просто добавьте `V=1` в команду компиляции. Это позволит вам получить более подробный отчёт, который может помочь разработчику ядра или другому опытному пользователю при решении разного рода проблем.

```
# make bzImage V=1
```

Использование модулей ядра

Модули ядра - это ещё одно название для драйверов устройств, которые могут быть внедрены в работающее ядро. Они позволяют вам получить расширенную поддержку оборудования, поддерживаемого вашим ядром, без необходимости использования другого ядра или сборки своего собственного.

Также модули могут быть загружены или выгружены в любое время, даже в работающей системе. Это упрощает для системных администраторов обновление отдельных драйверов. Можно скомпилировать новый модуль, выгрузить старый и загрузить новый, и всё это без перезагрузки системы!

Модули в системе находятся в каталоге `/lib/modules/25@A8O_O4@O`. Они могут быть загружены во время загрузки системы посредством файла `rc.modules`. Этот файл содержит довольно хорошие комментарии и примеры для основных компонентов аппаратного обеспечения. Чтобы увидеть список активных на данный момент модулей, используйте команду `lsmod`(1):

```
# lsmod
Module                  Size  Used by
parport_pc              7220   0
parport                 7844   0  [parport_pc]
```

Здесь видно, что загружен только модуль параллельного порта. Чтобы выгрузить модуль, используйте команду `rmmod`(1). Модули могут быть загружены с помощью команды `modprobe`(1) или `insmod`(1). Как правило команда `modprobe` более безопасна, потому что она загружает все модули, от которых зависит модуль, который вы пытаетесь загрузить.

Многим пользователям вовсе не обязательно загружать или выгружать модули вручную. Они могут использовать автоматический загрузчик

модулей ядра для управления ими. По умолчанию в Slackware во все ядра встроен `kmod`. `kmod` - это функция ядра, которая позволяет ему загружать модули по мере их запроса. Дополнительную информацию о самом `kmod` и о том, как он настраивается, смотрите в `/usr/src/linux/Documentation/kmod.txt`. Для этого вам нужно установить пакет с исходными кодами или загрузить сами исходные тексты с http://kernel.org.

Дополнительную информацию можно найти на страницах руководства по любой из этих команд, плюс в файле `rc.modules`.

Глава 5.

Настройка сети

5.1. Введение: netconfig - ваш друг.

При первоначальной установке Slackware программа setup запускает программу `netconfig`. `netconfig` пытается выполнить для вас следующие действия:

- Спрашивает у вас имя и домен вашего компьютера.
- Даёт краткое поясние различных схем IP-адресации, сообщая при этом, когда они должны использоваться; и спрашивает у вас, какую схему адресации вы хотите использовать для настройки своей сетевой карты:
  - Static-IP
  - DHCP
  - Loopback
- Часто предлагается проверить настройку сетевой карты.

`netconfig` обычно возмёт на себя примерно 80% работы по настройке вашего подключения к локальной сети, если вы позволите ему сделать это. Обратите внимание, что вам настоятельно рекомендуется проверить свой конфигурационный файл по ряду причин:

1. Вам следует никогда не доверять программе setup в плане настройки своего компьютера. Если вы используете её, вам следует самому проверить полученную конфигурацию.
2. Если вы всё ещё изучаете Slackware и управление системой Linux, полезным может оказаться просмотр рабочей конфигурации. По крайне мере вы будете знать, как должна выглядеть ваша конфигурация. Это позволит вам в будущем устранить проблемы, возникшие вследствие неверной настройки системы.

5.2. Настройка сетевого оборудования

Приняв решение о том, что вы хотите поднять на своей машине со Slackware какой-либо тип сети, первым делом вам понадобится Linux-совместимая сетевая карта. Вы должны будете немного уделить ей внимания, дабы убедиться в том, что эта карта действительно совместима с Linux (за информацией о текущем состоянии своей сетевой карты обращайтесь, пожалуйста, к Linux Documentation Project и/или документации к ядру). Как правило скорее всего вы будете приятно удивлены количеством сетевых карт, поддерживаемых более современными ядрами. Говоря об этом, я бы всё равно посоветовал вам перед тем, как приобретать себе карту, обратиться к любому из списков совместимого оборудования (например, The GNU/Linux Beginners Group Hardware Compatibility Links[1] и The Linux Documentation Project Hardware HOWTO[2]), доступных в Интернете. Немного лишнего времени, потраченного на это исследование, может сохранить вам дни или даже недели, проведённые в попытках заставить заработать карту, которая вообще не совместима с Linux.

Когда вы ознакомитесь со списками оборудования, совместимого с Linux, доступными в Интернете, или прочтёте документацию к ядру, установленному в вашей системе, для вас будет весьма полезным обратить

1 http://www.eskimo.com/%7Elo/linux/hardwarelinks.html
2 http://www.linux.org/docs/ldp/howto/Hardware-HOWTO/

внимание на то, какой вам понадобится использовать модуль ядра для поддержки своей сетевой карты.

Загрузка сетевых модулей

Модули ядра, которые должны будут загружаться во время запуска системы, берутся из файла `rc.modules` каталога `/etc/rc.d` или посредством автоматической загрузки модулей ядра, инициируемой скриптом `/etc/rc.d/rc.hotplug`. Стандартный файл `rc.modules` содержит раздел поддержки сетевых устройств. Если вы откроете этот файл и найдёте нужный раздел, вы обратите внимание, что он сначала проверяет права на запуск файла `rc.netdevice` в каталоге `/etc/rc.d/`. Этот скрипт создаётся в том случае, если `setup` успешно автоматически определил ваше сетевое во время установки системы.

Под блоком "if" находится список сетевых устройств и закомментированные строки с modprobe. Найдите свою карту, раскомментируйте соответствующую строку modprobe и сохраните файл. Теперь запуск `rc.modules` под `root`'ом должен загрузить драйвер вашего сетевого устройства (а также все остальные раскомментированные модули, что перечислены в списке). Обратите внимание, что для некоторых модулей (например, для драйвера ne2000) требуются параметры; убедитесь, что вы выбрали правильную строку.

Сетевые карты (10/100/1000Base-T и Base-2)

Этот заголовок охватывает всё разнообразие внутренних сетевых PCI- и ISA-карт. Драйверы для этих карт предоставляются в виде загружаемых модулей ядра, как было описано в предыдущем параграфе. `/sbin/netconfig` должен был опросить вашу карту и успешно настроить файл `rc.netdevice`. В противном случае скорее всего проблема заключается в том, что вы пытаетесь загрузить неверный для данной карты модуль (это не такая уж и новость, что для различных поколений одной и той же модели

карты одного и того же производителя требуются разные модули). Если вы настаиваете на том, что модуль, который вы пытаетесь загрузить, действительно является правильным, тогда вам лучше всего обратиться к документации по этому модулю, дабы попытаться выяснить, нужны ли ему специальные параметры во время инициализации.

Модемы

Как и сетевые карты, модемы могут иметь поддержку шин различного типа. До последнего времени большинство модемов представляли собой 8- или 16-битные ISA-карты. Благодаря усилиям Intel и всех производителей материнских плат, направленным на полное вытеснение шины ISA, сейчас мы имеем ситуацию, при которой большинство модемов - это внешние устройства, подключаемые к последовательному или USB-порту, или внутренние PCI-модемы. Если вы хотите, чтобы ваш модем работал в Linux, тогда *ЧРЕЗВЫЧАЙНО* важно провести предварительный анализ предстоящей покупки, в особенности, если вы рассматриваете вариант приобретения PCI-модема. Многие, если не большинство, PCI-модемы, доступные сегодня в магазинах, являются т.н. win-модемами. В этих модемах на самой карте отсутствует базовое аппаратное обеспечение: функции, выполняемые этим оборудованием обычно загружаются в процессор драйвером модема и операционной системой Windows. Это означает, что у них отсутствует стандартный последовательный интерфейс, который ожидает увидеть демон PPPD, когда он пытается дозвониться к вашему Интернет-провайдеру.

Если вы хотите быть абсолютно уверены в том, что приобретаемый вами модем будет работать в Linux, выбирайте внешний аппаратный модем, подключаемый к последовательному порту вашего ПК. Это гарантирует более надёжную работу и меньшее количество проблем при его установке и обслуживании, однако для таких устройств требуется внешнее питание и цена на них выше.

Существуют различные веб-сайты, предоставляющие драйверы и помощь по настройке устройств на базе win-модемов. Некоторые пользователи сообщали об успешной настройке и установке драйверов для различных win-модемов на базе чипсетов Lucent, Conexant и Rockwell. Поскольку требуемое программное обеспечение для этих устройств не входит в состав Slackware и оно серьёзно варьируется от драйвера к драйверу, мы не будет подробно рассматривать его.

PCMCIA

Как часть установки Slackware вам предлагается возможность установить пакет pcmcia (из категории "A"). Этот пакет содержит приложения и файлы настройки, необходимые для работы PCMCIA-карт в Slackware. Важное замечание: пакет pcmcia устанавливает только стандартное программное обеспечение, необходимое для работы с PCMCIA-картами в Slackware. С ним НЕ устанавливаются никакие драйверы или модули. Доступные модули и драйверы будут находиться в каталоге `/lib/modules/‘uname-r‘/pcmcia`. Вам придётся немного поэкспериментировать самостоятельно, чтобы найти модуль, который будет работать с вашей сетевой картой.

Вам понадобится отредактировать файл `/etc/pcmcia/network.opts` (для Ethetnet-карты) или `/etc/pcmcia/wireless.opts` (если вы являетесь обладателем карты для беспроводной связи). Как и большинство конфигурационных файлов Slackware, эти два файла содержат очень подробные комментарии и вам довольно просто будет определить, какие изменения необходимо внести.

5.3. Настройка TCP/IP

На данный момент ваша сетевая карта уже должна быть физически

вставлена в ваш компьютер, и должны быть загружены соответствующие драйверы ядра. Вы пока что не сможете установить связь посредством своей сетевой карты, однако о ней можно получить информацию с помощью команды `ifconfig -a`.

```
# ifconfig -a
eth0 Link encap:Ethernet HWaddr 00:A0:CC:3C:60:A4
UP BROADCAST NOTRAILERS RUNNING MULTICAST MTU:1500 Metric:1
RX packets:110081 errors:1 dropped:0 overruns:0 frame:0
TX packets:84931 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:114824506 (109.5 Mb) TX bytes:9337924 (8.9 Mb)
Interrupt:5 Base address:0x8400

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:2234 errors:0 dropped:0 overruns:0 frame:0
TX packets:2234 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:168758 (164.8 Kb) TX bytes:168758 (164.8 Kb)
```

Если вы набрали просто `/sbin/ifconfig` без параметра `-a`, вы не увидите интерфейс `eth0`, потому что вашей сетевой карте ещё не присвоен действительный IP-адрес или маршрут.

Хотя существует много различных способов для настройки и организации сети, всех их можно свести к двум типам: статический и динамический. В статических сетях каждый узел (в просторечии, нечто, имеющее IP-адрес) всегда имеет один и тот же IP-адрес. В динамических сетях IP-адреса для узлов выдаются централизованно одной машиной, называемой DHCP-сервером.

DHCP

DHCP (Dynamic Host Configuration Protocol, протокол динамического конфигурирования хоста) определяет, какой IP-адрес может быть

присвоен компьютеру во время загрузки. Во время загрузки DHCP-*клиент* отправляет в локальную сеть запрос на поиск DHCP-*сервера*, чтобы последний назначил ему IP-адрес. DHCP-сервер имеет пул (или *диапазон*) доступных IP-адресов и ответит на такой запрос выделением клиенту IP-адреса из пула вместе со *временем аренды*. По истечении этого времени данный IP-адрес считается устаревшим, и клиент должен связаться с сервером и повторить операцию получения или продления срока действия IP-адреса.

Затем клиент примет выданный сервером IP-адрес и настроит запрошенный интерфейс на использование этого адреса. Однако существует ещё одна довольно удобная возможность, которую DHCP-клиенты используют для получения IP-адреса. Клиент запоминает последний выданный ему адрес, а затем обращается к серверу с запросом повторно выдать ему тот же самый IP-адрес. Если это возможно, сервер так и сделает, в противном случае будет назначен новый адрес. Итак, согласование происходит следующим образом:

Клиент: Если в этой ЛВС сервер DHCP?
Сервер: Да, есть. Это я.
Клиент: Мне нужен IP-адрес.
Сервер: Можешь взять 192.168.10.10 сроком на 19200 секунд.
Клиент: Спасибо.

Клиент: Если в этой ЛВС сервер DHCP?
Сервер: Да, есть. Это я.
Клиент: Мне нужен IP-адрес. В последний раз у меня был 192.168.10.10. Могу я снова
Сервер: Да, можешь (или Нет, не можешь: вместо этого получай 192.168.10.12).
Клиент: Спасибо.

DHCP-клиентом в Linux является `/sbin/dhcpcd`. Если вы загрузите `/etc/rc.d/rc.inet1` в свой любимый текстовый редактор, вы заметите, что `/sbin/dhcpcd` вызывается примерно в середине скрипта. Результатом этого вызова будет беседа, представленная выше. Также `dhcpcd` будет следить

за временем, оставшимся до конца аренды текущего IP-адреса, и при необходимости автоматически обратится к серверу DHCP с запросом на продление аренды. DHCP также может предоставлять сопутствующую сетевую информацию: используемый сервер ntp, маршрут по умолчанию и т.п.

Настроить DHCP в Slackware довольно просто. Просто запустите `netconfig` и выберите DHCP, когда будет предложено. Если у вас несколько сетевых интерфейсов и вы не хотите, чтобы `eth0` был настроен посредством DHCP, просто отредактируйте файл `/etc/rc.d/rc.inet1.conf` и измените соответствующую переменную для своей карты на "`YES`".

Статический IP

Статические IP-адреса представляют собой фиксированные адреса, которые изменяются только вручную. Они используются в тех случаях, когда администратор не хочет, чтобы информация об IP изменялась, например, для внутренних серверов в локальной сети, серверов, подключенных к Интернету и маршрутизаторов. Используя статическую IP-адресацию, вы присваиваете адрес, и он остаётся неизменным. Другие машины знают, что вы всегда доступны по определённому IP-адресу и могут в любой момент связаться с вами, используя этот адрес.

`/etc/rc.d/rc.inet1.conf`

Если вы планируете назначить IP-адрес своей новой машине со Slackware, вы можете сделать это через скрипт `netconfig` или можете отредактировать файл `/etc/rc.d/rc.inet1.conf`. В `/etc/rc.d/rc.inet1.conf` вы найдёте следующее:

```
# Primary network interface card (eth0)
IPADDR[0]=""
NETMASK[0]=""
USE_DHCP[0]=""
```

```
DHCP_HOSTNAME[0]=""
```

А ниже:

```
GATEWAY=""
```

В этом случае наша задача заключается просто в указании корректной информации, заключённой в кавычки. Эти переменные используются скриптом `/etc/rc.d/rc.inet1` во время загрузки системы для настройки сетевых интерфейсов. Для каждого из интерфейсов просто введите верную информацию об IP или укажите "`YES`" для переменной `USE_DHCP`. Slackware поднимет интерфейсы с указанными здесь данными в порядке нахождения сетевых устройств.

Переменная `DEFAULT_GW` определяет маршрут по умолчанию для Slackware. Все соединения между вашим компьютером и другими машинами в Интернете должны осуществляться через этот шлюз, если для них не указан другой маршрут. Если вы используете DHCP, вам обычно нет необходимости указывать здесь что-либо, посколько сервер DHCP сам укажет вам, какой шлюз использовать.

`/etc/resolv.conf`

Итак, у вас есть IP-адрес, у вас есть шлюз по умолчанию, у вас даже может быть десять миллионов долларов (дайте нам немножко), однако что в этом хорошего, если вы не можете преобразовать имена в IP-адреса? Никто не захочет набирать `72.9.234.112` в адресной строке своего браузера для того, чтобы зайти на `www.slackbook.org`. В конце концов кто кроме авторов сможет запомнить этот IP-адрес? Нам нужно настроить DNS, но как? Вот для чего нужен `/etc/resolv.conf`.

Есть шанс, что нужные параметры уже присутствуют в `/etc/resolv.conf`. Если вы настраиваете своё сетевое подключение, используя DHCP, сервер DHCP должен был обновить этот файл за вас. (Если быть точным,

DHCP-сервер просто сообщил службе `dhcpcd`, что нужно именно поместить в этот файл, и она подчинилась команде.) Если вам нужно вручную обновить свой список серверов DNS, отредактируйте файл `/etc/resolv.conf`. Ниже представлен пример:

```
# cat /etc/resolv.conf
nameserver 192.168.1.254
search lizella.net
```

Первая строка довольно проста. Директива nameserver сообщает нам, какому DNS-серверу отправлять запросы. В силу сложившихся обстоятельств это всегда IP-адрес. У вас их может быть сколько угодно. Slackware охотно проверит их один за другим до тех пор, пока не получит нужный ответ.

Вторая строка немного интереснее. Директива search даёт нам список доменных имён, подразумеваемых в каждом DNS-запросе. Это позволяет вам связываться с машиной, используя только первую часть её FQDN (Fully Qualified Domain Name, полное доменное имя машины). Например, если в вашем пути search указан “slackware.com”, вы сможете обратиться к `http://store.slackware.com`, указав в своём веб-браузере только `http://store`.

```
# ping -c 1 store
PING store.slackware.com (69.50.233.153): 56 data bytes
64 bytes from 69.50.233.153 : icmp_seq=0 ttl=64 time=0.251 ms
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.251/0.251/0.251 ms
```

/etc/hosts

Теперь, когда наш DNS нормально работает, что если нам вдруг понадобится обойти наш сервер DNS или добавить DNS-запись для машины, которая отсутствует в DNS? В Slackware есть всеми любимый

файл `/etc/hosts`, содержащий локальный список доменных имён и соответствующих им IP-адресов.

```
# cat /etc/hosts
127.0.0.1           localhost  locahost.localdomain
192.168.1.101       redtail
172.14.66.32        foobar.slackware.com
```

Здесь вы можете видеть, что localhost имеет IP-адрес `127.0.0.1` (всегда зарезервирован для localhost), к redtail'у можно обратиться как к `192.168.1.101`, а `foobar.slackware.com` имеет адрес `172.14.66.32`.

5.4. PPP

Многие люди всё ещё выходят в Интернет через коммутируемые соединения. Наиболее общим методом является PPP, хотя иногда ещё используется и SLIP. Научить свою систему общаться с удалённым сервером по PPP довольно легко. Мы включили в дистрибутив несколько утилит, чтобы помочь вам в этом.

pppsetup

Slackware содержит программу под названием `pppsetup` для настройки системы на использование вашей учётной записи дозвона. Внешне она похожа на нашу программу `netconfig`. Перед запуском убедитесь, что у вас есть права root'а. Затем наберите `pppsetup`, чтобы запустить её. Вы должны будете увидеть экран наподобие этого:

Программа задаст вам ряд вопросов, на которые вы должны будете дать соответствующие ответы. Это будет информация об устройстве вашего модема, строке инициализации модема и номере телефона провайдера. Также некоторые пункты будут иметь значения по умолчанию, которые в большинстве случаев вы можете оставить без изменений.

После запуска программы она создаст скрипты `ppp-go` и `ppp-off`. Они используются, соответственно, для установки и разрыва PPP-соединения. Оба скрипта находятся в `/usr/sbin` и для их запуска необходимы права root'а.

`/etc/ppp`

Для большинства пользователей достаточно будет запустить `pppsetup`. Однако возможна ситуация, когда вы захотите изменить некоторые параметры, используемые демоном PPP. Вся конфигурационная информация хранится в каталоге `/etc/ppp`. Вот перечень файлов и их назначение:

`ip-down`	Этот скрипт запускается демоном `pppd` после разрыва PPP-соединения.
`ip-up`	Этот скрипт запускается демоном `pppd` после успешной установки PPP-соединения. Поместите в этот файл любые команды, которые вы хотите запустить после успешного подключения.
`options`	Общие конфигурационные параметры `pppd`.
`options.demand`	Общие конфигурационные параметры `pppd` при запуске в режиме дозвона по запросу.
`pppscript`	Эти команды отправляются в модем.
`pppsetup.txt`	Журнал того, что вы вводили при запуске `pppsetup`.

Замечание: Большинство этих файлов будет отсутствовать до тех пор, пока вы не запустите `pppsetup`.

5.5. Беспроводная связь

Сети беспроводной связи всё ещё остаются относительно новыми решениями в мире компьютеров хотя и быстро завоёвывают рынок, поскольку всё больше людей начинают приобретать ноутбуки и хотят сразу получать доступ к сети безо всякой возни с какими-то старыми кабелями с витой парой. И не похоже, чтобы эта тенденция замедляла свой ход. К сожалению беспроводная связь ещё не так хорошо поддерживается в Linux, как традиционная проводная связь.

Есть три основных этапа настройки Ethernet-карты 802.11 для беспроводной связи:

1. Аппаратная поддержка карты беспроводной связи.
2. Настройка карты на подключение к точке беспроводного доступа.
3. Настройка сети.

Аппаратная поддержка

Аппаратная поддержка карты беспроводной связи обеспечивается ядром: либо через модули, либо непосредственно самим ядром. В общем случае работа самых новых Ethernet-карт обеспечивается через модули ядра, поэтому вам потребуется подобрать соответствующий модуль и загрузить его с помощью `/etc/rc.d/rc.modules`. `netconfig` может и не определить вашу карту, поэтому вам, возможно, придётся самостоятельно определить тип и модель своей карты. Для получения дополнительной информации о драйверах ядра для различных карт беспроводной связи загляните на http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/.

Настройка параметров беспроводной связи

Основная часть этой работы выполняется утилитой `iwconfig`, поэтому как

обычно читайте её страницу руководства, если вам нужна дополнительная информация.

Сначала вам потребуется настроить свою точку доступа (access point). Точки доступа немного различаются в плане принятой терминологии и способе их настройки, поэтому вам может придётся слегка повозиться, чтобы разобраться со своим оборудованием. В общем случае вам понадобится как минимум следующая информация:

- Идентификатор домена или название сети (называемый `iwconfig`'ом как ESSID).
- Канал, используемый WAP'ом.
- Параметры шифрования, включая все используемые ключи (желательно в шестнадцатиричном виде).

> **Внимание**ЗАМЕЧАНИЕ О WAP. WAP полон недостатков, однако это лучше, чем ничего. Если вы хотите повысить уровень безопасности своей беспроводной сети, вам следует рассмотреть использование технологий VPN или IPSec, рассмотрение которых выходит за рамки этой книги. Вы также можете настроить WAP таким образом, чтобы не публиковался его ID домена/ESSID. Подробное обсуждение политики реализации беспроводной связи выходит за рамки этого раздела, однако быстрый поиск в Google даст вам гораздо больше, чем вы хотели бы узнать.

После того, как вы собрали эту информацию и использовали `modprobe` для загрузки соответствующего драйвера ядра, вы можете открыть на редактирование `rc.wireless.conf` и добавить в него свои параметры. Файл `rc.wireless.conf` выглядит немного непонятным. Как минимум вам необходимо внести изменения в раздел generic: свои ESSID и ключ KEY, и CHANNEL, если это требуется вашей картой. (Попробуйте не устанавливать CHANNEL, если это работает - замечательно; в противном случае установите для него соответствующее значение.) Если вы наберётесь смелости, вы можете отредактировать файл так,

чтобы установить только необходимые переменные. Имена переменных в `rc.wireless.conf` соответствуют параметрам `iwconfig` и считываются скриптом `rc.wireless` с последующим использованием соответствующих команд `iwconfig`.

Если у вас есть ключ в шестнадцатиричном виде, это просто идеально, поскольку вы можете быть достаточно уверенными в том, что ваш WAP и `iwconfig` найдут общий язык посредством этого ключа. Если у вас есть только текстовая строка, вы не можете быть уверены в том, что ваш WAP преобразует её в шестнадцатиричный ключ, поэтому вам может быть придётся немного поломать голову (или получить ключ для WAP в hex-формате).

После того, как вы отредактировали `rc.wireless.conf`, запустите `rc.wireless` под `root`'ом, а затем - `rc.inet1` (опять же под `root`'ом). Вы можете протестировать своё беспроводное подключение стандартными утилитами, такими как `ping`, в сочетании с `iwconfig`. Если вы используете проводное подключение, вы можете воспользоваться `ifconfig`'ом, чтобы выключить эти интерфейсы, пока вы будете тестировать беспроводную сеть, дабы убедиться в том, что между ними не происходит взаимное влияние. Вам также может потребуется проверить свои изменения путём перезагрузки.

Теперь, когда вы увидели, как редактировать `/etc/rc.d/rc.wireless` для своей сети по умолчанию, давайте поближе рассмотрим iwconfig, чтобы увидеть как он со всем этим работает. При этом вы узнаете быстрый и грязный путь настройки wifi на случай, если вы окажетесь в Интернет-кафе, кофейном магазине или в зоне действия любого другого хот-спота и захотите выйти в онлайн.

Первым делом надо сообщить своей карте беспроводной связи, к какой сети необходимо подключиться. Убедитесь, что вы заменили "`eth0`" на сетевой интерфейс, который использует ваша карта, и изменили "*`mynetwork`*" на ESSID, который вы хотите использовать. Затем вы должны будете указать ключ шифрования (если есть), используемый в вашей беспроводной сети. И, наконец, укажите используемый канал (если нужно).

```
# iwconfig eth0 essid "mynetwork"
# iwconfig eth0 key xxxxxxxxxxxxxxxxxxxxxxxxxx
# iwconfig eth0 channel n
```

Вот, что должно быть на вашей стороне беспроводного соединения.

Настройка сети

Это делается точно так же, как и для проводных сетей. Просто обратитесь к предыдущим разделам этой главы.

5.6. Сетевые файловые системы

На данный момент в вашем распоряжении должно быть работающее TCP/IP-подключение к вашей сети. Вы должны быть в состоянии пинговать другие компьютеры сети и, если вы соответствующим образом настроили шлюз, вы также должны быть в состоянии пинговать компьютеры в Интернете. Как известно, главной целью подключения компьютера к сети, является получение доступа к информации. Хотя некоторые люди могут подключать компьютер к сети просто так, большинство людей хотели бы предоставлять и получать доступ к файлам и принтерам. Они хотели бы получать доступ к документам в Интернете или играть в онлайновые игры. Установив в свою новую систему Slackware поддержку TCP/IP и необходимое программное обеспечение, вы всё это получите; однако, установив только поддержку TCP/IP, функциональность будет очень ограниченной. Чтобы предоставлять и получать общий доступ к файлам, нам потребуется переносить их туда и обратно, используя FTP или SCP. Мы не можем посматривать на нашем новой компьютере со Slackware дерево файлов через значки “Сетевое окружение” или “Вся сеть” с Windows-компьютеров. Мы хотели бы иметь возможность иметь постоянный доступ к файлам на других Unix-машинах.

В идеале мы хотели бы использовать *сетевую файловую систему*, позволяющую нам иметь прозрачный доступ к файлам на компьютерах. Программам, которые мы используем для работы с информацией, хранимой на компьютерах, на самом деле даже не надо знать на каком компьютере хранится нужный файл. Им нужно только знать, что этот файл существует, и способ для его получения. Дальнейшее уже является задачей операционной системы, обеспечивающей доступ к этому файлу с помощью доступных локальных и сетевых файловых систем. Две наиболее часто используемые сетевые файловые системы - это SMB (реализованная через Samba) и NFS.

SMB/Samba/CIFS

SMB (Server Message Block, блок серверных сообщений) - это потомок более старого протокола NetBIOS, изначально разработанного в IBM для их продукта LAN Manager. Компанию Microsoft в свою очередь всегда интересовал NetBIOS и его наследники (NetBEUI, SMB и CIFS). Проект Samba начал своё существование в 1991 году, когда он был написан для обеспечения связи между IBM PC и сервером Unix. Сегодня предоставление общего доступа к файлам и службам печати через сеть SMB является предпочитаемым методом практически для всего цивилизованного мира, поскольку его поддерживает и Windows.

Конфигурационный файл Samba `/etc/samba/smb.conf` является одним из самых хорошо документированных конфигурационных файлов, которые вы сможете найти. К вашим услугам уже готовые примеры с настройками общих ресурсов, так что вы можете просмотреть и изменить их согласно своим потребностям. Если же вам нужен ещё больший контроль, к вашим услугам страница руководства smb.conf. Поскольку Samba имеет такую хорошую документацию, мы не будем её здесь переписывать. Однако быстро остановимся на основных моментах.

`smb.conf` разбит на несколько разделов: по одному разделу на общий ресурс плюс один глобальный раздел для настройки параметров, которые

используются везде. Некоторые параметры являются действительными только в глобальном разделе, а некоторые верны только за его пределами. Помните, что глобальный раздел может быть переопределён любым другим разделом. За дополнительной информацией обращайтесь к страницам руководства.

Вы скорее всего захотите отредактировать свой файл `smb.conf`, чтобы отразить в нём параметры своей локальной сети. Советуем вам изменить перечисленные ниже пункты:

```
[global]
# workgroup = Домен NT или Рабочая группа, напр: LINUX2
workgroup = MYGROUP
```

Измените значение **workgroup** на домен/рабочую группу, которые используются в вашей ЛВС.

```
# server string является эквивалентом поля Description в NT
server string = Samba Server
```

Это будет описание вашего компьютера **Slackware**, показываемое в папке Сетевое окружение (или Вся сеть).

```
# Режим безопасности. Большинство людей захотят установить режим
# безопасности user. Подробности смотрите в файле security_level.txt.
# ЗАМЕЧАНИЕ: Чтобы получить поведение Samba-1.9.18, необходимо использовать
# "security = share".
security = user
```

Вы почти наверняка захотите использовать в своей системе **Slackware** уровень безопасности **user**.

```
# Вы можете использовать шифрование паролей. Пожалуйста,
# прочтите файлы ENCRYPTION.txt, Win95.txt и WinNT.txt
# в документации по Samba.
# Не включайте этот параметр, если вы не прочли эти документы.
encrypt passwords = yes
```

Если шифрование паролей не включено, вы не сможете использовать Samba с системами NT4.0, Win2k, WinXP и Win2003. Для предыдущих версий операционных систем Windows для предоставления доступа к общим ресурсам шифрование не требовалось.

SMB является протоколом с аутентификацией, т.е. вы можете указать имя пользователя и пароль, чтобы воспользоваться возможностями этой службы. Мы сообщаем серверу samba о том, что имена пользователей и пароли верны, посредством команды `smbpasswd`. `smbpasswd` допускает использование общих ключей для добавления как обычных пользователей, так и машин-пользователей (для SMB необходимо, чтобы вы добавили NETBIOS-имена компьютеров как машин-пользователей, ограничивая тем самым круг компьютеров, с которых может осуществляться аутентификация).

```
Добавление пользователя в файл /etc/samba/private/smbpasswd.
# smbpasswd -a user
Добавление имени компьютера в файл /etc/samba/private/smbpasswd.
# smbpasswd -a -m machine
```

Важно учесть, что данное имя пользователя или имя машины должно уже существовать в файле `/etc/passwd`. Вы можете добиться этого с помощью команды `adduser`. Обратите внимание, что при использовании команды `adduser` для добавления имени компьютера к нему необходимо добавить знак доллара (“`$`”). Однако этого *не* нужно делать при работе с `smbpasswd`. Утилита `smbpasswd` самостоятельно добавляет знак доллара. Нарушение этого правила посредством `adduser` приведёт к ошибке при добавлении имени машины в samba.

```
# adduser machine$
```

Сетевая файловая система (NFS)

NFS (Network File System) изначально была написана компанией Sun

для Solaris - их реализации системы Unix. И хотя её значительно легче поднять и настроить по сравнению с SMB, NFS гораздо менее безопасна. Главным слабым местом в безопасности является несложность подмены идентификаторов пользователя и группы одной машины на идентификаторы с другой машины. В протоколе NFS не реализована аутентификация. Было заявлено, что в будущих версиях протокола NFS безопасность будет повышена, однако на время написания этой книги это ещё не было сделано.

Настройка NFS осуществляется через файл `/etc/exports`. Когда вы загрузите стандартный файл `/etc/exports` в редактор, вы увидите пустой файл с комментарием вверху на две строки. Нам надо будет добавить строку в файл exports для каждого из каталогов, которые мы хотим экспортировать, с перечнем клиентских рабочих станций, которым будет разрешён доступ к этому каталогу. Например, если нам нужно экспортировать каталог `/home/foo` для рабочей станции Bar, нам надо будет добавить в наш файл `/etc/exports` такую строку:

```
/home/foo Bar(rw)
```

Ниже представлен пример файла `exports` из страницы руководства:

```
# образец файла /etc/exports
/               master(rw) trusty(rw,no_root_squash)
/projects       proj*.local.domain(rw)
/usr            *.local.domain(ro) @trusted(rw)
/home/joe       pc001(rw,all_squash,anonuid=150,anongid=100)
/pub            (ro,insecure,all_squash)
```

Как видите, существует несколько различных опций, однако большинство из них должны быть понятными из этого примера.

NFS полагает, что заданный пользователь с одной из машин в сети имеет один и тот же идентификатор на всех остальных машинах. Когда NFS-клиент делает попытку чтения или записи на NFS-сервер, UID передаётся как часть запроса на чтение/запись. Этот UID считается

таким же, как если бы запрос был выполнен с локальной машины. Как видите, если кто-то сможет произвольным образом указать заданный UID при обращении к ресурсам на удалённой машине, неприятности могут случиться и случаются. Средство, отчасти позволяющее избежать этого, заключается в монтировании всех каталогов с параметром `root_squash`. Это переопределяет UID любого пользователя, объявившего себя root'ом, на другой UID, предотвращая таким образом root'овый доступ к файлам и каталогам в экспортируемом каталоге. Похоже, что `root_squash` включается по умолчанию по соображениям безопасности, однако авторы всё равно рекомендуют явно указывать его в своём файле `/etc/exports`.

Вы также можете экспортировать каталог на сервере непосредственно из командной строки, воспользовавшись командой `exportfs`, как показано ниже:

```
# exportfs -o rw,no_root_squash Bar:/home/foo
```

Эта команда экспортирует каталог `/home/foo` для компьютера "`Bar`" и предоставляет ему доступ на чтение/запись. Кроме того на сервере NFS не включен параметр `root_squash`, означающий, что любой пользователь на Bar с UID "0" (UID root'а) будет иметь на сервере те же привилегии, что и root. Синтаксис выглядит довольно странно (обычно, когда вы указываете каталог в виде `computer:/directory/file`, вы ссылаетесь на файл в каталоге на заданном компьютере).

Дополнительную информацию о файле exports вы найдёте в странице руководства.

Глава 6.

Настройка X

Начиная со Slackware-10.0, работа системы X Window в Slackware обеспечивается Xorg'ом. X (комп. сленг "иксы") отвечают за предоставление пользователю графического интерфейса. Они не зависят от операционной системы, как это реализовано в Windows или MacOS.

Система X Window реализована в виде множества программ, выполняемых в рабочей среде пользователя. Двумя главными компонентами являются сервер и оконный менеджер. Сервер предоставляет набор функций для низкоуровневого взаимодействия с вашим видеооборудованием. Оконный менеджер находится над сервером и предоставляет интерфейс пользователя. Преимуществом такого подхода является возможность иметь целый набор различных графических интерфейсов путём простой смены оконного менеджера.

Настройка X'ов может оказаться сложной задачей. Это связано с тем, что для PC-архитектуры доступно огромное количество видеокарт, большая часть из которых использует различные программные интерфейсы. К счастью на сегодня большинство карт поддерживают базовые видеостандарты наподобие VESA, и если ваша карта из их числа, вы сможете запустить X с помощью команды `startx` сразу после установки системы.

Если эта команда не работает с вашей картой, или если вы хотели бы воспользоваться дополнительными возможностями своей видеокарты, таким как аппаратное ускорение или аппаратный 3D-рендеринг, тогда вам придётся перенастроить X.

Чтобы настроить X вам нужно создать файл `/etc/X11/xorg.conf`. Этот файл содержит много подробной информации о вашей видеокарте, мыши и мониторе. Этот конфигурационный файл очень сложен, но к счастью существует несколько программ, которые помогут вам создать его. Здесь мы упомянем только некоторые из них.

6.1. *xorgconfig*

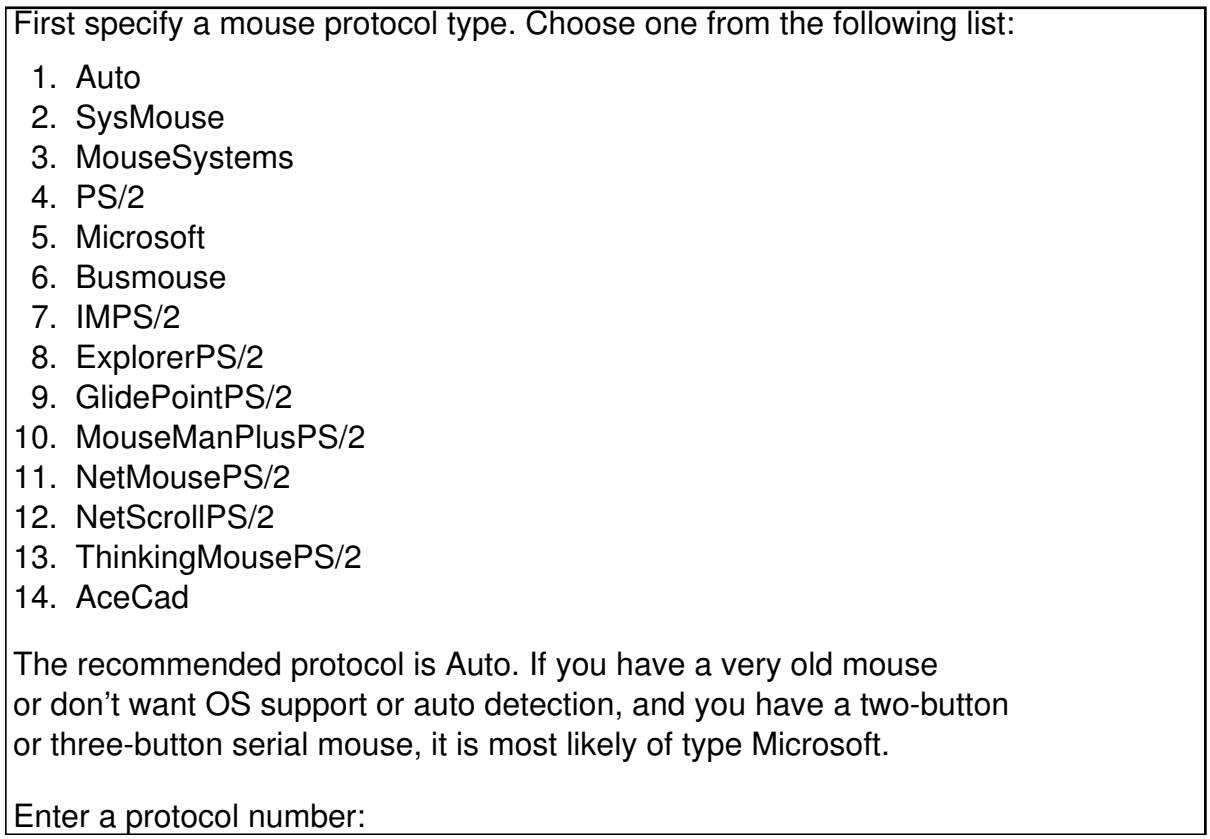
Это простой интерфейс с меню, похожий на тот, что используется в инсталляторе Slackware. Он просто говорит X-серверу, чтобы тот взглянул на видеокарту, а затем, основываясь на полученной информации, создал файл с наилучшей исходной конфигурацией, которую он сможет получить. Созданного файла `/etc/X11/xorg.conf` для начала должно быть вполне достаточно для большинства систем (и он должен работать без внесения изменений).

`xorgconfig` - это программа настройки X с консольным интерфейсом, которая была разработана для опытных системных администраторов. Ниже представлен пример работы с `xorgconfig`. Сначала запустите программу:

```
# xorgconfig
```

При этом вы увидите в полноэкранном режиме информацию об `xorgconfig`. Для продолжения нажмите **ENTER**. `xorgconfig` попросит вас проверить, правильно ли у вас установлена переменная `PATH`. С ней всё должно быть в порядке, поэтому продолжайте, нажав на **ENTER**.

Рисунок 6-1. *xorgconfig*: настройка мыши



Выберите в представленном меню свою мышь. Если ваша последовательная мышь в нём отсутствует, выберите протокол Microsoft protocol - он самый распространённый и наверняка будет работать. Затем `xorgconfig` спросит у вас, будете ли вы использовать `ChordMiddle` и `Emulate3Buttons`. Вы увидите подробное описание этих параметров на экране. Используйте их в том случае, если в X не работает средняя кнопка мыши, или если у вашей мыши только две кнопки (`Emulate3Buttons` позволяет вам путём одновременного нажатия двух кнопок эмулировать третью кнопку мыши). Затем введите название устройства своей мыши. Вариант по умолчанию (`/dev/mouse`) должен подойти, поскольку во время настройки Slackware для него была создана ссылка. Если у вас запущен GPM (сервер Linux для мыши) в режиме повторителя, вы можете указать

тип мыши как `/dev/gpmdata`, чтобы X получал информацию о мыши через `gpm`. В некоторых случаях (особенно в случае с bus-мышами) это может работать лучше, однако большинству пользователей этого делать не следует.

`xorgconfig` спросит вас о включении привязок для специальных клавиш. Если вам это нужно, скажите "**y**". Большинство пользователей могут ответить "**n**" - выберите этот вариант, если вы не уверены.

Рисунок 6-2. *xorgconfig*: горизонтальная синхронизация

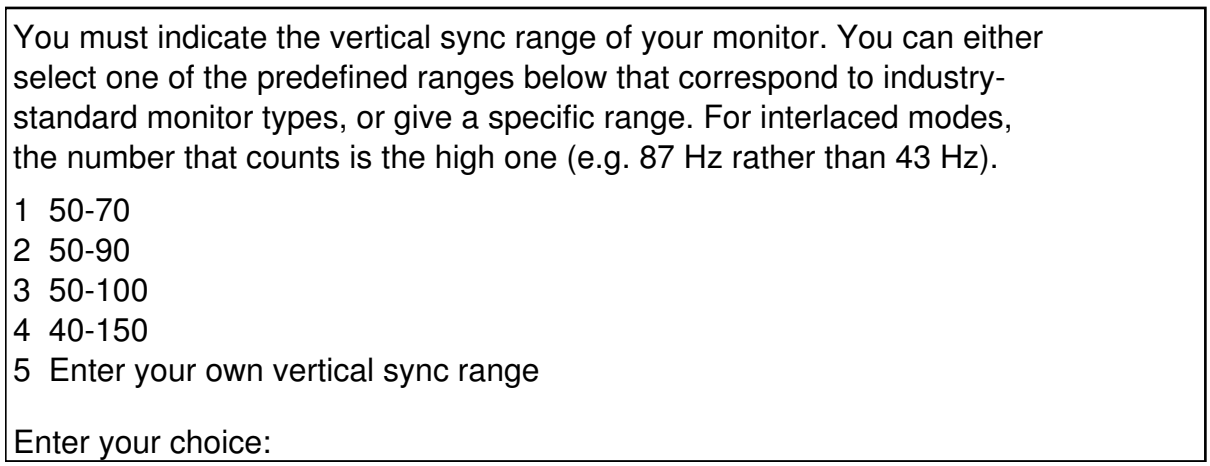
```
You must indicate the horizontal sync range of your monitor. You can either
select one of the predefined ranges below that correspond to industry-
standard monitor types, or give a specific range.
It is VERY IMPORTANT that you do not specify a monitor type with a horizontal
sync range that is beyond the capabilities of your monitor. If in doubt,
choose a conservative setting.
    hsync in kHz; monitor type with characteristic modes
 1  31.5; Standard VGA, 640x480 @ 60 Hz
 2  31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
 3  31.5, 35.5; 8514 Compatible, 1024x768 @ 87 Hz interlaced (no 800x600)
 4  31.5, 35.15, 35.5; Super VGA, 1024x768 @ 87 Hz interlaced, 800x600 @ 56 Hz
 5  31.5 - 37.9; Extended Super VGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
 6  31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
 7  31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
 8  31.5 - 64.3; Monitor that can do 1280x1024 @ 60 Hz
 9  31.5 - 79.0; Monitor that can do 1280x1024 @ 74 Hz
10  31.5 - 82.0; Monitor that can do 1280x1024 @ 76 Hz
11  Enter your own horizontal sync range

Enter your choice (1-11):
```

В следующем разделе введите диапазон частот своего монитора. Чтобы запустить его настройку, нажмите **ENTER**. Вы увидите список с типами мониторов - выберите один из них. Будьте осторожны с тем, чтобы не превысить возможности своего монитора. Это может повредить ваше

оборудование.

Рисунок 6-3. *xorgconfig*: вертикальная синхронизация



Укажите диапазон частот вертикальной развёртки своего монитора (вы можете найти его в руководстве по монитору). Затем xorgconfig попросит вас ввести строку, идентифицирующую тип вашего монитора в файле `xorg.conf`. Введите в эти 3 строки всё, что угодно (или вообще ничего не указывайте).

Рисунок 6-4. *xorgconfig*: видеокарта

```
 0  * Generic VESA compatible                    -
 1  * Generic VGA compatible                     -
 2  * Unsupported VGA compatible                 -
 3  ** 3DLabs, TI (generic)            [glint]   -
 4  ** 3Dfx (generic)                  [tdfx]    -
 5  ** ATI (generic)                   [ati]     -
 6  ** ATI Radeon (generic)            [radeon]  -
 7  ** ATI Rage 128 based (generic)    [r128]    -
 8  ** Alliance Pro Motion (generic)   [apm]     -
 9  ** Ark Logic (generic)             [ark]     -
10  ** Chips and Technologies (generic) [chips]  -
11  ** Cirrus Logic (generic)          [cirrus]  -
12  ** Cyrix MediaGX (generic)         [cyrix]   -
13  ** DEC TGA (generic)               [tgz]     -
14  ** Intel i740 (generic)            [i740]    -
15  ** Intel i810 (generic)            [i810]    -
16  ** Linux framebuffer (generic)     [fbdev]   -
17  ** Matrox Graphics (generic)       [mga]     -

Enter a number to choose the corresponding card definition.
Press enter for the next page, q to continue configuration.
```

Теперь у вас есть возможность взглянуть на базу данных моделей видеокарт. Вам понадобится сделать это, поэтому скажите "**y**" и выберите свою карту из показанного списка. Если вы не наблюдаете именно свою модель карты, попробуйте выбрать ту, в которой используется такой же набор микросхем, и скорее всего одна будет нормально работать.

Далее сообщите `xorgconfig`'у, какой объём памяти установлен на вашей видеокарте. `xorgconfig`'у потребуется, чтобы вы ввели какое-нибудь описание для своей карты. Если хотите, вы можете ввести описание в этих трёх строках.

Затем вам будет предложено выбрать используемое разрешение экрана. И опять же использование предложенных по умолчанию значений

должно для начала устроить вас. Позже вы можете отредактировать `/etc/X11/xorg.conf` и переопределить режимы, чтобы умолчанию использовалось разрешение 1024x768 (или любое другое, которое вам понравится).

На данном этапе программа `xorgconfig` спросит вас, хотите ли вы сохранить текущую конфигурацию в файл. Ответьте "yes" и конфигурационный файл для X будет сохранён, завершая таким образом процесс настройки. Теперь вы можете запустить X с помощью команды `startx`.

6.2. *xorgsetup*

Вторым способом настройки X является использование `xorgsetup` - программы автоматической настройки, поставляемой вместе со Slackware.

Чтобы запустить `xorgsetup`, войдите в систему под root'ом и наберите:

```
# xorgsetup
```

Если у вас уже есть файл `/etc/X11/xorg.conf` (т.е. вы уже настроили X), вам будет предложено сделать резервную копию существующего конфигурационного файла перед тем, как продолжить. Исходный файл будет переименован в `/etc/X11/xorg.conf.backup`.

6.3. xinitrc

`xinit`(1) - это программа, запускающая X'ы. К ней есть некое подобие интерфейса под названием `startx`(1), который инициализирует один сеанс системы X Window. Вы могли и не слышать об `xinit` (возможно, вам это и не понадобится). Однако её конфигурационный файл определяет, какие программы (в том числе и какой оконный менеджер) будут запускаться при запуске X'ов. Сначала `xinit` проверяет ваш домашний каталог на предмет наличия в нём файла `.xinitrc`. Если файл

найден, он будет запущен, в противном случае будет использован файл `/var/X11R6/lib/xinit/xinitrc` (используемый по умолчанию для всей системы). Вот пример файла `xinitrc`:

```
#!/bin/sh
# $XConsortium: xinitrc.cpp,v 1.4 91/08/22 11:41:34 rws Exp $

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

# объединение значений по умолчанию и раскладок клавиатуры

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi

if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi

# запуск некоторых полезных программ

twm &
xclock -geometry 50x50-1+1 &
xterm -geometry 80x50+494+51 &
xterm -geometry 80x20+494-0 &
exec xterm -geometry 80x66+0+0 -name login
```

Все блоки “if” предназначены для объединения различных конфигурационных параметров из других файлов. Наиболее интересная

часть находится в конце файла, в которой описан запуск различных программ. Описанный сеанс начнётся с запуска оконного менеджера `twm`(1), часов и трёх терминалов. Обратите внимание на команду `exec` перед последним `xterm`. Она заменяет запущенный в данный момент командный процессор (тот, что выполняет этот скрипт `xinitrc`) на шелл `xterm`(1). Когда пользователь закроет этот `xterm`, сеанс X будет завершён.

Чтобы настроить свой сценарий запуска X'ов, скопируйте стандартный файл `/var/X11R6/lib/xinit/xinitrc` в `~/.xinitrc` и отредактируйте его, заменив эти строки с программами на то, что пожелаете. У меня окончание этого файла довольно простое:

```
# Запуск оконного менеджера:
exec startkde
```

Обратите внимание, что в `/var/X11R6/lib/xinit` есть несколько различных файлов `xinitrc.*`, которые соответствуют различным оконным менеджерам и графическим интерфейсам пользователя. Вы можете использовать любой из них, который вам понравится.

6.4. *xwmconfig*

Многие годы Unix был практически единственной операционной системой для серверов за исключением высокопроизводительных профессиональных рабочих станций. Только людям с техническим складом ума нравилось использовать Unix-подобные операционные системы, и этот факт был хорошо отражён на пользовательском интерфейсе. Графические интерфейсы пользователя (GUI) были довольно бедными по внешнему оформлению, поскольку разрабатывались они для запуска нескольких графических приложений наподобие CAD-программ и программ для рендеринга графики. Основная часть работы с файлами и управление системой выполнялось из командной строки. Различные поставщики вычислительной техники (Sun Microsystems, Silicon Graphics

и др.) продавали рабочие станции, которые пытались предоставить целостную систему "внешнего представления и оформления", однако широкое разнообразие инструментария для GUI, используемого разработчиками, приводило к невозможности достичь единообразия рабочего стола. Полоса прокрутки могла выглядеть по-разному в двух различных приложениях. Меню могли появляться в различных местах. Программы могли иметь различные кнопки и управляющие элементы. Цвета тоже широко варьировались и обычно были жёстко запрограммированы в каждом инструментарии для разработки. До тех пора, пока пользователями оставались в основном технические профессионалы, всё это не имело особого значения.

С появлением свободных Unix-подобных операционных систем и постоянно растущим числом разнообразных графических приложений X'ы в конце концов получили широкое распространение на рабочих столах пользователей. Однако большинство пользователей уже привыкли к внешнему оформлению и виду Microsoft Windows и Apple MacOS; отсутствие такого соответствия в X-приложениях стало препятствием на пути их дальнейшего распространения. Ответным действием стало появление двух проектов с открытым исходным кодом: K Desktop Environment или KDE и GNU Network Object Model Environment, известного как GNOME. Каждый из них обладал широким набором приложений: от панелей задач и файловых менеджеров до игр и офисных пакетов, написанных с помощью одного и того же инструментария и имеющих хорошую интеграцию, предоставляя таким образом стандартизированный и единообразный рабочий стол.

Отличия между KDE и GNOME в общем случае довольно незначительные. Внешне они отличаются друг от друга, поскольку в них используются различные инструментарии разработки GUI. KDE основан на библиотеке Qt от Troll Tech AS, а GNOME использует GTK - инструментарий, разработанный для GNU Image Manipulation Program (или просто GIMP). KDE и GNOME, будучи отдельными проектами, имеют свои собственные команды дизайнеров и программистов со своими стилями и философией

разработки. Однако в каждом из случаев результат получается в принципе один и тот же: согласованная, хорошо интегрированная среда рабочего стола и коллекция приложений. Функциональные возможности, удобство в использовании и приятный внешний вид и KDE, и GNOME делают их конкурентами для всего, что доступно в других операционных система.

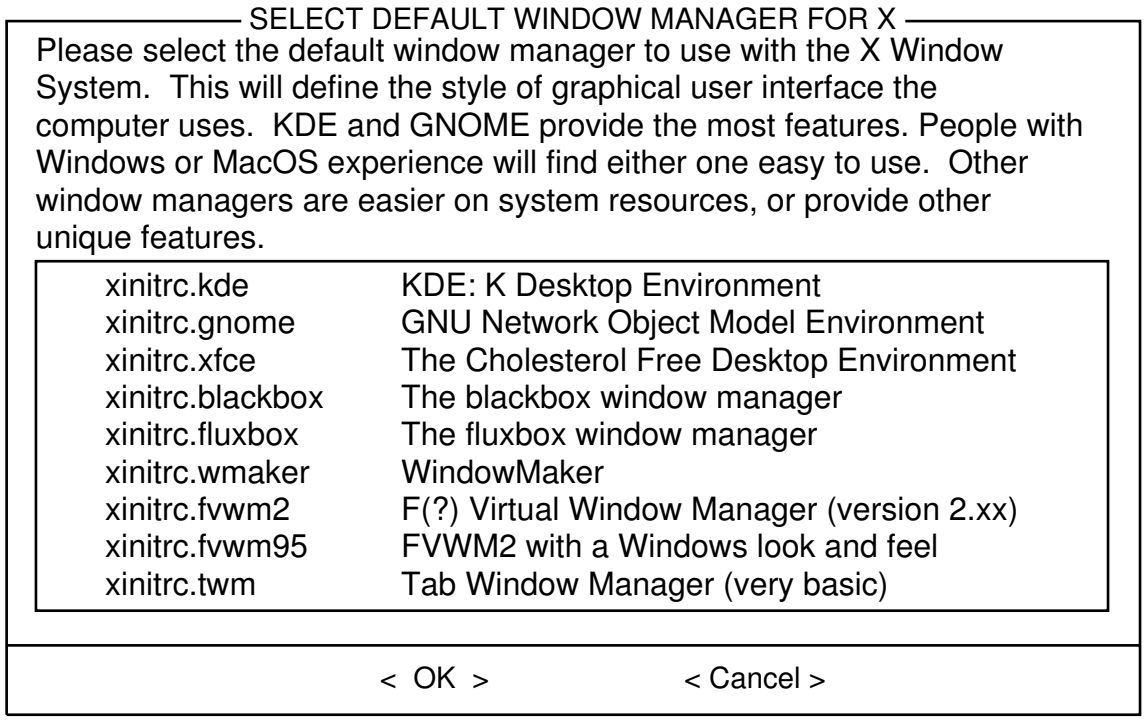
Однако самым лучшим достоинством этих настольных сред является их свободность. Это означает, что вы можете использовать как одну из них, так и другую или обе среды сразу (да, вы не ошиблись, обе одновременно). Всё зависит от вашего выбора.

Кроме GNOME и KDE в Slackware есть большая коллекция оконных менеджеров. Некоторые из них были разработаны для эмуляции других операционных систем, некоторые для настройки "под себя", а некоторые для максимального быстродействия. Они слишком разнообразны. Естественно, вы можете установить их сколько пожелаете, побаловаться во всеми и решить, что из них вам наиболее подходит.

Чтобы упростить выбор оконного менеджера, в Slackware есть программа под названием `xwmconfig`, которую можно использовать для выбора рабочего стола или оконного менеджера. Запускается она так:

```
% xwmconfig
```

Рисунок 6-5. Настройка рабочего стола с помощью *xorgconfig*



Вам будет представлен список всех установленных рабочих столов и оконных менеджеров. Просто выберите из этого списка то, что вам нужно. Каждому из пользователей вашей системы понадобится запустить эту программу, т.к. разные пользователи могу использовать разные рабочие столы, и не все захотят использовать тот, который вы выбрали как стандартный во время установки системы.

Затем просто запустите X'ы, и вы готовы к дальнейшей работе.

6.5. *xdm*

Поскольку Linux становится всё более и более полезным в качестве настольной операционной системы, многие пользователи хотели бы, чтобы машина загружалась сразу с графической оболочкой. Для этого вам понадобится сообщить Slackware загружаться непосредственно в

X'ы и определить графический менеджер входа в систему. Slackware предоставляет на выбор три такие графические утилиты: `xdm`(1), `kdm` и `gdm`(1).

`xdm` - это графический менеджер входа в систему, идущий в комплекте с системой X.org. Он используется повсеместно, однако не настолько удобен по функциональности, как его альтернативы. `kdm` - это графический менеджер входа в систему, поставляемый вместе с KDE (K Desktop Environment). И, наконец, `gdm` - это менеджер, который поставляется вместе с GNOME. Любой из этих вариантов позволит вам войти в систему под любым пользователем, выбрав при этом нужный рабочий стол.

К сожалению в состав Slackware не входит удобная программа наподобие `xwmconfig` для выбора менеджера входа в систему, поэтому, если вы установите все три, вам может понадобиться отредактировать некоторые файлы, чтобы оставить работать только один менеджер. Но сначала мы рассмотрим загрузку с графическим режимом.

Для того, чтобы X'ы запускались во время загрузки, вам необходимо загрузиться в 4-й уровень запуска. Уровни запуска (runlevel) - это просто способ сообщения `init`(8)'у выполнить что-то другое, когда он запускает операционную систему. Делается это путём редактирования конфигурационного файла `init`'а - `/etc/inittab`.

```
# Стандартные уровни запуска Slackware:
#   0 = останов
#   1 = однопользовательский режим
#   2 = не используется (однако настроен так же, как и 3-й уровень)
#   3 = многопользовательский режим (уровень запуска по умолчанию в Slackware)
#   4 = X11 с KDM/GDM/XDM (менеджеры сеансов)
#   5 = не используется (однако настроен так же, как и 3-й уровень)
#   6 = перезагрузка

# Уровень загрузки по умолчанию (не устанавливайте в 0 или 6)
id:3:initdefault:
```

Для того, чтобы Slackware загружался в графический режим, нужно просто 3 заменить на 4.

```
# Уровень загрузки по умолчанию (не устанавливайте в 0 или 6)
id:4:initdefault:
```

Теперь Slackware загрузится в уровень запуска 4 и выполнит `/etc/rc.d/rc.4`. Этот файл запускает X'ы и вызывает выбранный вами менеджер входа в систему. Итак, как же нам теперь выбрать этот менеджер? Для этого есть несколько способов, и мы расскажем о них после того, как взглянем на файл `rc.4`.

```
# Пытаемся использовать gdm - менеджер сеансов GNOME:
if [ -x /usr/bin/gdm ]; then
  exec /usr/bin/gdm -nodaemon
fi

# Нет такого? Хорошо, попытаемся использовать kdm - менеджер сеансов KDE:
if [ -x /opt/kde/bin/kdm ]; then
  exec /opt/kde/bin/kdm -nodaemon
fi

# Если XDM - это всё, что у вас есть, я допускаю, что нужно выполнить следующее:
if [ -x /usr/X11R6/bin/xdm ]; then
  exec /usr/X11R6/bin/xdm -nodaemon
fi
```

Как видите, `rc.4` сначала проверяет, является ли `gdm` исполняемым файлом, и если это так - запускает его. Вторым в списке стоит `kdm`, а последним - `xdm`. Одним из способов выбора менеджера сеансов является простое удаление из системы того, который вы не хотите использовать, с помощью команды `removepkg`. Узнать подробнее о `removepkg` вы можете в Гл. 18.

Как вариант вы можете снять разрешение на выполнение с тех файлов, которые вы не хотите использовать. Работа с `chmod` описана в Гл. 9.

```
# chmod -x /usr/bin/gdm
```

Наконец, вы можете просто закомментировать строки с менеджером сеансов, который вы не хотите использовать.

```
# Пытаемся использовать gdm - менеджер сеансов GNOME:
# if [ -x /usr/bin/gdm ]; then
#   exec /usr/bin/gdm -nodaemon
# fi

# Нет такого? Хорошо, попытаемся использовать kdm - менеджер сеансов KDE:
if [ -x /opt/kde/bin/kdm ]; then
  exec /opt/kde/bin/kdm -nodaemon
fi

# Если XDM - это всё, что у вас есть, я допускаю, что нужно выполнить следующее:
if [ -x /usr/X11R6/bin/xdm ]; then
  exec /usr/X11R6/bin/xdm -nodaemon
fi
```

Любые строки, которые начинаются со знака решётки (#), считаются комментариями, и командный процессор полностью игнорирует их. Таким образом, даже если gdm установлен и является исполняемым, командный процессор (в нашем случае bash) не будет беспокоиться о его проверке.

Глава 7.

Загрузка

Процесс загрузки вашей системы Linux иногда может быть простым, а иногда - сложным. Многие пользователи устанавливают Slackware на свой компьютер и всё. Затем они просто включают его, и он готов к использованию. А иногда даже простая загрузка машины может оказаться сложной задачей. Для большинства пользователей больше всего подходит LILO. В Slackware для загрузки операционной системы имеются LILO и Loadlin. LILO может работать с жёсткого диска, раздела, главной загрузочной записи жёсткого диска или дискеты, представляя собой очень гибкий инструмент. Loadlin работает из командной строки DOS, убивая DOS и запуская вместо него Linux.

Другим популярным инструментом для загрузки Linux является GRUB. GRUB официально не поддерживается и не входит в состав Slackware. Slackware придерживается стандарта “проверено - работает” для всего, что входит в состав дистрибутива. И хотя GRUB работает хорошо и имеет некоторые функции, которых нет у LILO, LILO справляется со всеми все основными задачами начального загрузчика как надёжная проверенная временем начальная запись. GRUB, будучи более молодым, пока ещё не добился такого же признания. Поскольку он не входит в состав Slackware, мы не будем его здесь рассматривать. Если вы хотите использовать GRUB (возможно, он остался от другой ОС Linux, и вы хотите использовать его для двойной загрузки), обратитесь к его документации.

В этом разделе описывается использование LILO и Loadlin - двух загрузчиков, включенных в состав Slackware. В нём также рассмотрены некоторые типичные сценарии двойной загрузки и их настройка.

7.1. LILO

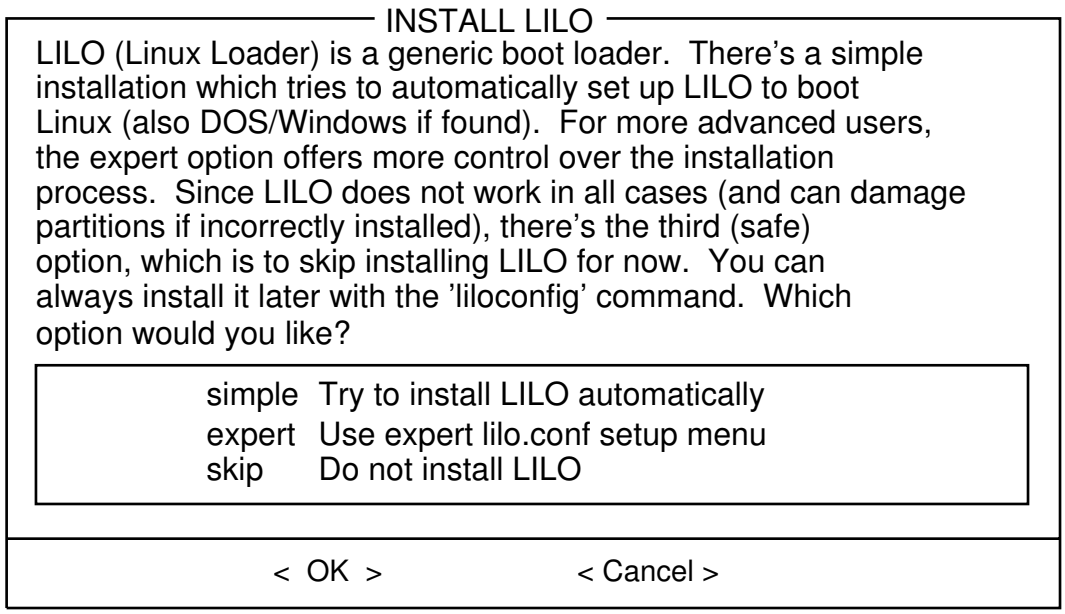
Загрузчик Linux (Linux Loader или LILO) является наиболее популярным загрузчиком из тех, что используются в Linux. Он является полностью настраиваемым и с лёгкостью может использоваться для загрузки других операционных систем.

В Slackware Linux есть утилита с меню для настройки LILO под названием `liloconfig`. Эта программа впервые запускается во время установки и настройки системы, однако вы можете запустить её позже, набрав в консоли `liloconfig`.

LILO считывает свои настройки из файла `/etc/lilo.conf`(5). Он не считывает их при каждой загрузке, а только при своей установке. Каждый раз, когда вы изменяете конфигурацию LILO, он должен быть переустановлен в загрузочный сектор. Многие ошибки LILO возникают из-за изменений в файле `lilo.conf`, однако повторный запуск lilo для установки этих изменений вызовет сбой. `liloconfig` поможет вам создать конфигурационный файл, чтобы вы смогли установить LILO для своей системы. Если вы предпочитаете отредактировать `/etc/lilo.conf` вручную, тогда, чтобы переустановить LILO, выполните в консоли команду `/sbin/lilo` (под `root`'ом).

Когда вы запускаете `liloconfig` в первый раз, он будет выглядеть примерно так:

Рисунок 7-1. ***liloconfig***



Если вы впервые настраиваете LILO, вам следует выбрать simple. В противном случае вы можете выбрать expert, если вы хорошо знаете LILO и Linux. Выбор пункта simple запустит настройку LILO.

Если ваше ядро собрано с поддержкой видеобуфера, `liloconfig` спросит, какое разрешение видео вы хотите использовать. Это разрешение, которое также используется сервером видеобуфера XFree86. Если вы не хотите, чтобы консоль работала в специальном видеорежиме, выберите normal и вы получите стандартный текстовый режим 80x25.

Следующий этап настройки LILO заключается в выборе места его установки. Это, как правило, самый важный этап. Список ниже поясняет места для установки:

`Root`

При этом LILO устанавливается в начало корневого раздела с Linux. Это самый безопасный вариант, если на вашем компьютере присутствует ещё одна операционная система. Он обеспечивает

сохранность любых других загрузчиков. Недостатком является то, что LILO будет загружаться оттуда только в том случае, если жёсткий диск с Linux является первым диском в вашей системе. Вот почему многие люди создают небольшой раздел `/boot` как первый диск в своей системе. Это позволяет установить и ядро, и LILO в начало диска, где LILO сможет найти их. Предыдущие версии LILO имели досадное ограничение, известное как "1024 cylinder limit" (ограничение 1024-го цилиндра). LILO не мог загружать ядра с разделов, находящихся за 1024-м цилиндром. В последних версиях LILO эта проблема устранена.

`Floppy`

Этот вариант ещё более безопасен, чем предыдущий. В нём создаётся дискета, которую вы можете использовать для загрузки своей системы Linux. При этом загрузчик находится вообще за пределами жёсткого диска, поэтому, чтобы загрузить Slackware, вам нужна только эта дискета. Недостатки этого метода очевидны. Во-первых, дискеты печально известны своей ненадёжностью. Во-вторых, начальный загрузчик больше не является частью вашей системы. Если вы потеряете дискету, вам придётся создать новую, чтобы загрузить свою систему.

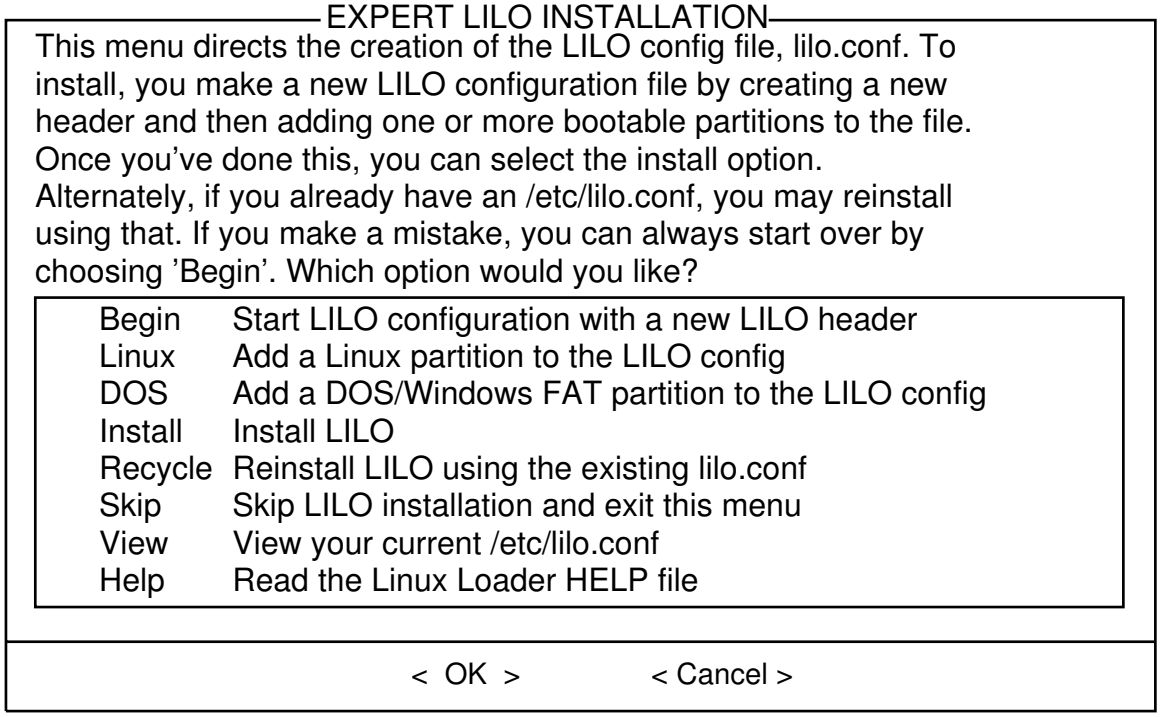
`MBR`

Вы наверняка захотите выбрать этот вариант, если Slackware является единственной операционной системой на вашем компьютере, или если вы будете использовать LILO для выбора между несколькими операционными системами на своём компьютере. Это наиболее предпочитаемый метод для установки LILO и он подойдёт практически для любого компьютера.

ВниманиеВ этом варианте будет перезаписан любой другой загрузчик, находящийся в MBR.

После выбора места установки `liloconfig` создаст конфигурационный файл и установит **LILO**. Вот и всё. Если вы выбрали режим **expert**, вы увидите специальное меню. Это меню позволяет вам дополнительно настроить файл `/etc/lilo.conf`, добавить в меню загрузки другие операционные системы и настроить **LILO** на передачу ядру специальных параметров во время загрузки. Меню **expert** выглядит примерно так:

Рисунок 7-2. *liloconfig*: меню expert



Какой бы ни была конфигурация вашей системы, настроить начальный загрузчик довольно легко. И `liloconfig` значительно помогает вам в этом.

7.2. LOADLIN

Другим вариантом загрузки Slackware Linux является LOADLIN. LOADLIN - это исполняемый DOS-файл, который может быть использован для запуска Linux из работающей системы DOS. Для него требуется, чтобы ядро Linux находилось на разделе DOS, тогда LOADLIN сможет загрузить его и корректно запустить всю систему.

Во время процесса установки LOADLIN будет скопирован в домашний каталог root'а в виде ZIP-файла. Автоматического процесса для настройки LOADLIN не существует. Вам понадобится скопировать ядро Linux (обычно это `/boot/vmlinuz`) и файл LOADLIN из домашнего каталога root'а в раздел DOS.

LOADLIN полезен в том случае, если вы хотите создать меню загрузки на своём разделе DOS. Меню можно добавить в файл `AUTOEXEC.BAT`, что позволит вам выбирать загрузку Linux или DOS. За загрузку Linux будет отвечать LOADLIN, запуская таким образом вашу систему Slackware. Следующий файл `AUTOEXEC.BAT` вполне можно использовать в Windows 95 для создания загрузочного меню:

```
@ECHO OFF
SET PROMPT=$P$G
SET PATH=C:\WINDOWS;C:\WINDOWS\COMMAND;C:\
CLS
ECHO Please Select Your Operating System:
ECHO.
ECHO [1] Slackware Linux
ECHO [2] Windows 95
ECHO.
CHOICE /C:12 "Selection? -> "
IF ERRORLEVEL 2 GOTO WIN
IF ERRORLEVEL 1 GOTO LINUX
:WIN
CLS
ECHO Starting Windows 95...
WIN
GOTO END
```

```
:LINUX
ECHO Starting Slackware Linux...
CD \LINUX
LOADLIN C:\LINUX\VMLINUZ ROOT=<корневой раздел> RO
GOTO END
:END
```

В качестве корневого раздела вам потребуется указать название устройства Linux, например, `/dev/hda2` или что-то в таком духе. Вы всегда можете использовать LOADLIN из командной строки. Просто используйте синтаксис из примера выше. В документации, поставляемой с LOADLIN, представлено много примеров его использования.

7.3. Двойная загрузка

Многие пользователи настраивают свои компьютеры на загрузку Slackware Linux и другой операционной системы. Если у вас возникли трудности с настройкой своей системы, ниже мы представили несколько типичных сценариев двойной загрузки.

Windows

Настройка компьютера на использование MS Windows и Linux является, наверное, самым распространённым вариантом двойной загрузки. Для настройки такой загрузки существует много способов, однако в этом разделе мы опишем только два.

Часто при настройке систем с двойной загрузкой пользователи разрабатывают идеальный план, согласно которому всё должно работать, но забывают учесть порядок установки. Очень важно понимать, что операционные системы необходимо устанавливать в определённом порядке, чтобы можно было использовать двойную загрузку. Linux всегда предлагает взять под свой контроль над тем, что будет записано в главную

загрузочную запись. Следовательно всегда рекомендуется устанавливать Linux в последнюю очередь. Первой необходимо установить Windows, поскольку она всегда принудительно помещает свой загрузчик в MBR, перезаписывая любую запись, которую Linux мог поместить туда.

С помощью LILO

Большинство людей предпочтут использовать LILO в качестве менеджера загрузки Linux и Windows. Как было сказано выше, вам следует сначала установить Windows, а потом - Linux.

Допустим, что в вашей системе есть только один жёсткий IDE-диск ёмкостью 40ГБ. Также допустим, что вам нужно выделить половину места на нём для Windows, а вторую половину - для Linux. Это вызовет ряд проблем с при попытке загрузить Linux.

```
20GB   Windows boot (C:)
1GB   Linux root (/)
19GB  Linux /usr (/usr)
```

Для Linux вам кроме того нужно ещё выделить раздел под пространство для свопинга соответствующего размера. Как правило его размер составляет два объёма оперативной памяти. Таким образом в системе с 64МБ ОЗУ размер свопа будет составлять 128МБ и т.д. Размер пространства для свопинга является темой многих обсуждений в IRC и Usenet. Не существует по-настоящему “правильного” способа для его определения, однако вполне достачно будет придерживаться приведенного выше правила (в моей системе 512МБ ОЗУ и я вообще не использую своп - прим.переводчика).

После разметки диска вам следует приступить к установке Windows. После этого вы можете установить Linux. Особого внимания заслуживает установка LILO. Для установки LILO вам потребуется выбрать режим expert.

Запустите настройку нового LILO. Вам потребуется установить главную загрузочную запись (MBR) таким образом, чтобы её можно было использовать для выборочной загрузки одной из операционных систем. Используя меню, добавьте разделы с Linux и Windows (или DOS). После этого вы можете установить LILO.

Перезагрузите компьютер. Должен загрузиться LILO, представив вам меню, которое позволяет вам выбрать загружаемую операционную систему. Просто выберите название нужной вам ОС (эти названия были выбраны вами во время настройки LILO) и нажмите Enter.

LILO обладает хорошими возможностями в плане своей настройки. Он не ограничен загрузкой только Linux и DOS. Он в состоянии загрузить практически всё, что угодно. Страницы руководства `lilo`(8) и `lilo.conf`(5) содержат более подробную информацию.

Что делать, если LILO не работает? Есть ситуации, в которых на отдельных машинах LILO просто отказывается работать. К счастью есть ещё один способ для осуществления двойной загрузки Linux и Windows.

С помощью LOADLIN

Этот способ можно использовать в том случае, если LILO не работает в вашей системе, или если вы просто не хотите настраивать LILO. Этот способ является идеальным способом для пользователей, которые часто переустанавливают Windows. Потому что при каждой переустановке Windows последняя перезаписывает MBR, уничтожая при этом LILO. С использованием LOADLIN эта проблема вам не грозит. Самым большим недостатком этого способа является то, что с помощью LOADLIN вы можете загрузить только Linux.

Используя LOADLIN, вы можете устанавливать операционные системы в любом порядке. Будьте осторожны с установкой чего-либо в MBR, вам не нужно этого делать. LOADLIN работает только с загрузочным разделом Windows. Поэтому во время установки Slackware пропустите этап настройки

LILO.

После установки операционной системы скопируйте файл `loadlinX.zip` (где *X* - это номер версии наподобие `16a`) из домашнего каталога root'а на раздел с Windows. Также скопируйте туда образ ядра. Это вам понадобится выполнить в Linux. В примере ниже показано, как сделать это:

```
# mkdir /win
# mount -t vfat /dev/hda1 /win
# mkdir /win/linux
# cd /root
# cp loadlin* /win/linux
# cp /boot/vmlinuz /win/linux
# cd /win/linuz
# unzip loadlin16a.zip
```

При этом на разделе Windows будет создан каталог `C:\LINUX` (подразумевается, что это `/dev/hda1`). Также в него будут скопированы рабочие файлы LOADLIN. После этого вам нужно будет перезагрузиться в Windows, чтобы настроить меню загрузки.

В Windows откройте окно с командной строкой. Сначала нам нужно убедиться в том, что система не настроена на загрузку графического интерфейса.

```
C:\> cd \
C:\> attrib -r -a -s -h MSDOS.SYS
C:\> edit MSDOS.SYS
```

Добавьте в файл следующую строку:

```
BootGUI=0
```

После этого сохраните файл и закройте редактор. Теперь отредактируйте файл `C:\AUTOEXEC.BAT`, добавив в него меню загрузки. Ниже представлен пример с куском файла `AUTOEXEC.BAT`, реализующий загрузочное меню:

```
cls
```

```
echo System Boot Menu
echo.
echo 1 - Linux
echo 2 - Windows
echo.
choice /c:12 "Selection? -> "
if errorlevel 2 goto WIN
if errorlevel 1 goto LINUX
:LINUX
cls
echo "Starting Linux..."
cd \linux
loadlin c:\linux\vmlinuz root=/dev/hda2 ro
goto END
:WIN
cls
echo "Starting Windows..."
win
goto END
:END
```

Самой важной строкой является строка, запускающая LOADLIN. В ней мы указываем загружаемое ядро, корневой раздел Linux и то, что сначала его нужно примонтировать в режиме только для чтения.

Утилиты для обоих этих методов есть в Slackware Linux. Также существует множество других загрузчиков, однако эти два способа должны работать в большинстве случаев реализации двойной загрузки систем.

Устаревший хак для Windows NT

Это последний общий вариант двойной загрузки. В давние времена LILO не мог загрузить Windows NT. Для этого пользователям Linux нужно было хакать NTLDR, что вызывало ещё больше проблем, чем при настройке двойной загрузки Windows 9x и Linux. Следует понимать, что следующие инструкции являются устаревшими. LILO научился загружать Windows

NT/2000/XP/2003 много лет назад. Однако, если вы используете старую машину, вам может понадобиться воспользоваться этим хаком.

1. Установите Windows NT.
2. Установите Linux, убедившись при этом, что LILO установлен в суперблок раздела Linux.
3. Извлеките первые 512 байт корневого раздела Linux и сохраните их в виде файла на разделе Windows NT.
4. Отредактируйте файл `C:\BOOT.INI` в Windows NT, добавив в него пункт Linux.

Установка Windows NT обязательно должна быть выполнена перед установкой Linux. После этого потребуется немного повозиться. Извлечение первых 512 байт раздела Linux на самом деле проще, чем это может показаться на первый взгляд. Для этого вы должны находится в Linux. При условии, что разделом Linux является `/dev/hda2`, выполните следующую команду:

```
# dd if=/dev/hda2 of=/tmp/bootsect.lnx bs=1 count=512
```

Вот и всё. Теперь вам нужно скопировать `bootsect.lnx` на раздел Windows NT. Однако здесь мы можем столкнуться с другой проблемой. В Linux не поддерживается запись в файловую систему NTFS. Если вы установили Windows NT и отформатировали диск в NTFS, вам придётся скопировать этот файл на дискету с FAT, а затем прочитать её из Windows NT. Если вы отформатировали диск с Windows NT в FAT, мы можете просто примонтировать этот раздел в Linux и скопировать туда файл. В любом случае вам нужно будет переложить файл `/tmp/bootsect.lnx` с раздела Linux в `C:\BOOTSECT.LNX` на разделе Windows NT.

Последний этап заключается в добавлении нового пункта в меню загрузки Windows NT. В Windows NT откройте окно с командной строкой.

```
C:\WINNT> cd \
```

```
C:\> attrib -r -a -s -h boot.ini
C:\> edit boot.ini
```

Добавьте в конец файла следующую сроку:

```
C:\bootsect.lnx="Slackware Linux"
```

Сохраните изменения и закройте редактор. После перезагрузки Windows NT вы увидите в меню загрузки системы новый пункт с Linux. Выберите его, чтобы загрузить Linux.

Linux

Да, люди на самом деле так поступают. Это определённо самый простой способ реализации двойной загрузки. Вы можете просто использовать LILO и добавить новые пункты в файл `/etc/lilo.conf`. Это всё, что нужно сделать.

Глава 8.

Командный процессор (shell)

В графической среде интерфейс предоставляется программой, создающей различные графические объекты: окна, полосы прокрутки, меню и т.п. В консольной среде интерфейс пользователя предоставляется командным процессором (shell, шелл), интерпретирующим команды. Сразу после входа в систему (о чём будет рассказано в этой главе) пользователи попадают в командный процессор и могут заниматься своими делами. Цель этой главы - познакомить вас с командным процессором и, в частности, с самым распространённым шеллом среди пользователей Linux - Bourne Again Shell (bash). Более детальную информацию обо всём, о чём идёт речь в этой главе, можно получить на странице руководства `bash`(1).

8.1. Пользователи

Вход в систему

После загрузки системы вы увидите на экране что-то вроде этого:

```
Welcome to Linux 2.4.18
Last login: Wed Jan   1 15:59:14 -0500 2005 on tty6.
darkstar login:
```

Хммм... а ведь никто ничего не говорил насчёт логина. И что такое darkstar? Не беспокойтесь. Вы скорее всего не установили никакого гиперпространственного соединения с искусственной луной Империи (я

боюсь, что протокол гиперпространственной связи в настоящий момент не поддерживается ядром Linux. Может быть в версии 2.8 его поддержка будет реализована.) Darkstar (Тёмная звезда) - это просто название одного из наших компьютеров, и по умолчанию используется его название. Если вы указали название своего компьютера во время загрузки, вы должны увидеть вместо darkstar своё название.

Что же касается логина... Если впервые впервые сталкиваетесь с этим, вам надо будет войти в систему под `root`'ом. Вам будет предложено ввести пароль. Если вы установили его во время загрузки, то сейчас самое время воспользоваться им. В противном случае просто нажмите Enter. Вот и всё - вы в системе!

Root: суперпользователь

Итак, кто такой или *что* же такое `root`? И что его учётная запись делает в *вашей* системе?

В мире Unix и подобных ему операционных систем (например, Linux), существует такое понятие как пользователи. Позже мы разберёмся с этим более подробно, но сейчас более важным является понимание того, что `root` является самым главным пользователем. `root` всесилен и всемогущ, *никто* не может не подчиниться ему. Это просто недопустимо. `root` является так называемым "суперпользователем", и это действительно так. А лучше всего то, что `root` - это *вы*.

Круто, не правда ли?

Если вы всё ещё не уверены: да, это очень круто. Однако ловушка заключается в том, что root по сути может нарушить работу чего угодно. Если вам надоело читать эти дифирамбы в честь root'а (прим. переводчика), вы можете перейти сразу к Разд. 12.1.1, чтобы узнать, как добавить пользователя, затем войти под ним в систему и начать работу. Обычно нормальным тоном считается, что получать права root'а лучше только в том, случае, если это действительно необходимо, чтобы таким

образом свести к минимуму возможность случайного нарушения работы системы.

Кстати, если вы решите, что вам нужно получить права root'а, работая в системе под другим пользователем, - нет никаких проблем. Просто воспользуйтесь командой `su`(1). Вы должны будете ввести пароль `root`'а, а затем вы получите в своё распоряжение командный процессор с правами `root`'а до тех пор, пока не наберёте `exit` или `logout`. Используя `su`, вы также можете стать любым другим пользователем, указав пароль этого пользователя: `su jack`, например, сделает вас мной (переводчиком этого руководства).

> **Замечание:** `root`'у разрешается выполнять SU для любого пользователя без необходимости вводить его пароль.

8.2. Командная строка

Запуск программ

Трудно достичь нужного результата, не запуская программы. Вы, конечно, можете сделать из своего компьютера подставку для чего-нибудь или подпереть им дверь, чтобы она не закрывалась, а кому-то может быть нравится шум вентиляторов в корпусе. Никто не запрещает вам делать этого. И я думаю, что все согласятся с тем, что использование компьютера в качестве жужжащей подставки для маминых цветов - это не то, за что ПК получили популярность, которой они могут сейчас похвастаться.

Итак, помните, что почти всё в Linux является файлом? Это также относится и к программам. Каждая команда, которую вы запускаете (если только она не встроена в командный процессор), представляет собой файл,

находящийся где-то в системе. Вы запускаете программу, просто указывая полный путь к ней.

Например, помните команду `su` из последнего раздела? На самом деле она находится в каталоге `/bin`: `/bin/su` точно так же запустит её.

Однако почему тогда работает просто команда `su`? Ведь вы не указывали, что она находится в `/bin`. Она с таким же успехом может находиться и в `/usr/local/share`, не так ли? Как шелл об этом *узнал*? Ответ на этот вопрос заключается в переменной окружения `PATH` (путь). В большинстве командных процессоров используется `PATH` или что-то очень похожее на эту переменную. Как правило в ней содержится список каталогов, в которых выполняется поиск запускаемых вами программ. Поэтому, когда вы запускали `su`, ваш шелл прошёлся по списку этих каталогов, проверив в них наличие исполняемого файла `su`, который бы он смог запустить. Первый найденный файл и был запущен. Это происходит и в том случае, если вы запускаете программу без указания полного пути к ней. Если вы получите ошибку типа "`Command not found`" ("`Команда не найдена`"), это означает только то, что программы, которую вы пытаетесь запустить, нет в вашей переменной `PATH`. (Естественно, это произойдёт также в том случае, если программы вообще не существует...) Более детально мы рассмотрим переменные окружения в Разд. 8.3.1.

Помните также, что "." (точка) является условным обозначением текущего каталога, поэтому, если вы окажетесь в каталоге `/bin`, `./su` будет работать так, как если бы вы указали полный путь.

Шаблоны подстановки

Практически все командные процессоры воспринимают некоторые символы как замену или аббревиатуру, вместо которых может подставляться всё что угодно. Такие символы называются шаблонами подстановки или масками (wildcard); наиболее общими являются `*` (звёздочка) и `?`. По общепринятому соглашению `?` обычно означает один

произвольный символ. Например, допустим, что вы находитесь в каталоге с тремя файлами: `ex1.txt`, `ex2.txt` и `ex3.txt`. Вам нужно скопировать все эти файлы (используя команду `cp`, описанную в Разд. 10.5.1) в другой каталог, например, в `/tmp`. Тогда набирание `cp ex1.txt ex2.txt ex3.txt /tmp` представляет собой довольно много лишней работы. Гораздо проще набрать `cp ex?.txt /tmp`; здесь `?` будет соответствовать каждому из символов "1", "2" и "3", и все они будут по очереди подставлены в командную строку.

Что вы там говорите? Это *всё ещё* слишком много работы? Вы правы. Это ужасно. У нас есть трудовое законодательство, защищающее нас в такого рода ситуациях. К счастью в нашем распоряжении есть ещё звёздочка (`*`). Как мы уже упоминали ранее, `*` соответствует "любому количеству символов", включая 0. Поэтому, если кроме этих трёх файлов в каталоге больше ничего нет, мы можем просто набрать `cp * /tmp`, одним махом переместив всех их. Допустим, что есть ещё пара файлов с именами `ex.txt` и `hejaz.txt`. Нам нужно скопировать только `ex.txt`; команда `cp ex* /tmp` выполнит нужное действие.

Команда `cp ex?.txt /tmp` затронет, естественно, только три первоначальные файла; в имени `ex.txt` нет символа, соответствующего маске `?`, поэтому этот файл будет пропущен.

Другим общим шаблоном подстановки является пара квадратных скобок `[ ]`. Любые символы, заключенные в эти скобки, будут подставлены в выражение поиска соответствий. Звучит слишком непонятно? Всё не так плохо. Допустим, например, что у нас есть каталог с такими восемью файлами: `a1`, `a2`, `a3`, `a4`, `aA`, `aB`, `aC` и `aD`. Нам нужно найти только те файлы, чьи имена оканчиваются цифрами; в этом нам поможет шаблон `[ ]`.

```
% ls a[1-4]
a1 a2 a3 a4
```

Однако, что если нам нужны только файлы `a1`, `a2` и `a4`? В предыдущем примере мы использовали выражение, соответствующее всем значениям от

1 до 4. Однако мы также можем перечислять через запятую отдельные значения.

```
% ls a[1,2,4]
a1 a2 a4
```

Я знаю, о чём вы сейчас подумали: “А как насчёт букв?” В Linux регистр символов имеет значение, т.е. `a` и `A` - это разные символы, и связь между ними присутствует только в вашем уме. Прописные символы всегда следуют перед строчными, поэтому `A` и `B` идут перед `a` и `b`. Вернёмся к нашему примеру. Если нам нужны файлы `a1` и `A1`, мы легко можем найти их с помощью шаблона `[ ]`.

```
% ls [A,a]1
A1 a1
```

Обратите внимание, что если бы мы использовали дефис вместо запятой, мы бы получили неверный результат.

```
% ls [A-a]1
A1 B1 C1 D1 a1
```

Вы также можете комбинировать строки с дефисом и запятыми.

```
% ls [A,a-d]
A1 a1 b1 c1 d1
```

Перенаправление ввода/вывода и использование конвейеров

(Здесь должен идти какой-то вступительный текст.) Думаю, в третьей редакции он таки появится :) (прим. переводчика).

```
% ps > blargh
```

Вы знаете, что это такое? Здесь я запустил `ps`, чтобы увидеть список выполняющихся процессов; команда `ps` описана в Разд. 11.3. Эта часть не слишком интересна. Более интересной частью является `> blargh`, которая выражается, грубо говоря, в следующем: взять вывод команды `ps` и записать его в файл `blargh`. Однако подождите, можно сделать ещё круче.

```
% ps | less
```

В этой команде вывод `ps` перенаправляется по конвейеру (pipe) в программу `less`, чтобы вывод можно было просмотреть в более удобном виде.

```
% ps >> blargh
```

Это третье из наиболее часто используемых перенаправлений; оно выполняет то же самое, что и “>”, но за тем исключением, что “>>” добавит вывод команды `ps` к файлу `blargh`, если этот файл существует. В противном случае будет создан файл как и в случае с перенаправлением “>” (“>” уничтожает текущее содержимое файла `blargh`.)

Также ещё существует оператор “<”, который означает “взять входные данные из следующего файла”, однако он используется не так уж и часто.

```
% fromdos < dosfile.txt > unixfile.txt
```

Перенаправления проявляются во всей своей красе, когда вы начинаете комбинировать их:

```
% ps | tac >> blargh
```

При этом будет выполнена команда `ps`, изменён порядок строк её вывода, а результат добавлен в конец файла `blargh`. Вы можете комбинировать сколько угодно таких перенаправлений, однако помните, что интерпретируются они слева направо.

Дополнительную информацию о перенаправлении смотрите на странице руководства `bash`(1) .

8.3. Bourne Again Shell (bash)

Переменные окружения

Система Linux - сложный зверь, в котором нужно многое отслеживать, в котором множество мелких деталей играют роль при повседневной работе со различными программами (о некоторых из которых вам может даже не понадобится ничего знать). Никому не хочется передавать кучу параметров каждой из запускаемых программ, сообщая ей какой используется тип терминала, имя хоста, как должна выглядеть строка приглашения и т.п.

Поэтому в качестве механизма копирования у пользователей есть так называемое окружение. Окружение определяет условия, в которых выполняются программы, и некоторые из этих из этих определений являются переменными; пользователь может изменять их . Практически любой командный процессор имеет свои переменные окружения (в противном случае это скорее всего не слишком удобный шелл). Здесь мы дадим обзор команд bash'а, предназначенных для работы с его переменными окружения.

Запустив `set` без опций, вы увидите все определённые на данный момент переменные окружения вместе с их значениями. Как и большинство встроенных в `bash` команд `set` может выполнять много других вещей (с помощью параметров); однако вместо их рассмотрения мы сошлёмся на страницу руководства `bash`(1). В Прим. 8-1 показана часть вывода команды `set`, выполненной на компьютере автора. Обратите внимание в этом примере на переменную `PATH`, которую мы обсуждали ранее. Программы изо всех этих каталогов можно запускать, набирая просто их названия, не указывая полный путь к ним.

```
% unset ПЕРЕМЕННАЯ
```

Команда `unset` удаляет любые указанные вами переменные, уничтожая

Пример 8-1. Вывод списка переменных окружения с помощью *set*

```
% set
PATH=/usr/local/lib/qt/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:
/usr/openwin/bin:/usr/games:.:/usr/local/ssh2/bin:/usr/local/ssh1/bin:
/usr/share/texmf/bin:/usr/local/sbin:/usr/sbin:/home/logan/bin
PIPESTATUS=([0]="0")
PPID=4978
PS1='\h:\w\$ '
PS2='> '
PS4='+ '
PWD=/home/logan
QTDIR=/usr/local/lib/qt
REMOTEHOST=ninja.tdn
SHELL=/bin/bash
```

и саму переменную, и её значение. `bash` вообще забудет о том, что такая переменная существовала. (Не беспокойтесь. Если вы вы что-то и определили в этом сеансе шелла, это скорее всего будет переопределено в любом другом сеансе.)

```
% export ПЕРЕМЕННАЯ=значение
```

Команда `export` действительно очень удобна. С её помощью вы присваиваете переменной окружения `ПЕРЕМЕННАЯ` определённое "`значение`". Если `ПЕРЕМЕННАЯ` не существует, она будет немедленно создана. Если `ПЕРЕМЕННАЯ` уже имеет какое-то значение, оно будет потеряно. Это нехорошо, если вы, например, хотите просто добавить каталог в свою переменную `PATH`. В этом случае вам надо выполнить что-то вроде этого:

```
% export PATH=$PATH:/типа/новый/каталог
```

Обратите внимание на использование `$PATH`: если вам нужно, чтобы `bash` интерпретировал переменную (т.е. использовал её значение), поставьте перед именем переменной знак `$`. Например, команда `echo $PATH` выведет на экран значение переменной `PATH`; в моём случае это выглядело так:

```
% echo $PATH
/usr/local/lib/qt/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:
/usr/openwin/bin:/usr/games:.:/usr/local/ssh2/bin:/usr/local/ssh1/bin:
/usr/share/texmf/bin:/usr/local/sbin:/usr/sbin:/home/logan/bin
```

Завершение ввода табуляцией

(Здесь снова идёт какой-то вступительный текст.)

1. В интерфейсе командной строки выполняется много ввода с клавиатуры.
2. Ввод с клавиатуры - это работа.
3. Никто не любит работать.

Из пунктов 3 и 2 следует, что (4) никто не любит осуществлять ввод с клавиатуры. К счастью `bash` избавляет нас нас от (5) пункта (никому не нравится интерфейс командной строки).

Вы спросите: “Каким образом `bash` предоставляет эту прекрасную возможность?” В дополнение к расширениям шаблонов подстановки, рассмотренным нами выше, `bash` обладает возможностью завершения ввода табуляцией (tab completion).

Завершение работает примерно так: вы набираете имя файла. Он может находиться в вашей переменной `PATH`, или вы можете набирать полный путь к нему. Всё, что вам нужно сделать - набрать достаточно полное имя файла, чтобы уникально идентифицировать его. Затем нажмите клавишу Tab. `bash` выяснит, что вам нужно и завершит ввод имени за вас!

Рассмотрим пример. К каталоге `/usr/src` находятся два подкаталога: `/usr/src/linux` и `/usr/src/sendmail`. Нам нужно увидеть, что находится в `/usr/src/linux`. Мы набираем только `ls /usr/src/l`, жмём **`TAB`**, и `bash` даёт нам полный путь `ls /usr/src/linux`.

Теперь представим, что существуют два каталога: `/usr/src/linux` и `/usr/src/linux-old`. Если мы наберём `/usr/src/l` и нажмём **TAB**, `bash` заполнит за нас столько, сколько сможет, и мы получим `/usr/src/linux`. На этом можно остановиться или нажать ещё раз **TAB**, при этом `bash` выведет список каталогов, которые соответствуют тому, что мы набрали ранее.

Отсюда следует, что можно обойтись меньшим вводом с клавиатуры (и, следовательно, людям может понравится интерфейс командной строки).

8.4. Виртуальные терминалы

Представьте, что вы над чем-то работаете, и вам понадобилось параллельно сделать что-то ещё. Вы можете просто прервать свою основную работу и переключиться на другую задачу, но ведь это многопользовательская система, не правда ли? Вы ведь можете войти в систему несколько раз, верно? Так почему вы должны выполнять только одну задачу?

А вы и не должны. Мы не можем иметь на своей машине несколько клавиатур, мышек и мониторов. Большинству из нас они не понадобятся. Очевидно, что использование дополнительного аппаратного обеспечения не решит проблему. Решение предоставляется программному обеспечению, и Linux смело берёт это на себя, предоставляя "виртуальные терминалы" или "ВТ".

Нажимая **Alt** одновременно с функциональными клавишами, вы можете переключаться между виртуальными терминалами; каждая функциональная клавиша соответствует одному терминалу. По умолчанию Slackware имеет приглашения для входа в систему (login'ы) на 6-ти ВТ. Комбинация **Alt+F2** переключит вас во второй терминал, **Alt+F3** - в третий и т.д.

Остальные функциональные клавиши зарезервированы для X-сеансов.

Каждый X-сеанс использует свой собственный ВТ, начиная с седьмого (**Alt+F7**) и до двенадцатого. При работе в X'ах комбинация **Alt+Функц. клавиша** заменяется на **Ctrl+Alt+Функц. клавиша**. Поэтому если вы находитесь в X'ах и хотите перейти в консольный интерфейс (не завершая X-сеанса), комбинация **Ctrl+Alt+F3** переключит вас в третий терминал. (**Alt+F7** вернёт вас назад в X'ы при условии, что вы работали в первом X-сеансе.)

Утилита **screen**

Однако как насчёт ситуаций, когда виртуальных терминалов просто нет? Что тогда? К счастью в **Slackware** есть прекрасная утилита для управления "экранами" с одноимённым названием - `screen`. `screen` - это эмулятор терминала, который имеет виртуальные терминалы с похожими возможностями. При запуске `screen` появится небольшая вводная информация о программе, а затем вы будете возвращены в терминал. В отличие от стандартных виртуальных терминалов `screen` имеет свои собственные команды. Все команды `screen` начинаются с комбинации **Crtl+A**. Например, **Ctrl+A+C** создаст новый терминальный сеанс. **Ctrl+A+N** переключит вас в следующий терминал. **Ctrl+A+P** переключит вас в предыдущий терминал.

`screen` также поддерживает отключение и повторное подключение к сеансам `screen`, что довольно удобно при удалённой работе по `ssh` и `telnet` (подробнее мы рассмотрим их позже). **Ctrl+A+D** отключит вас от текущего сеанса. Запуск `screen -r` выведет на экран список всех запущенных на данный момент сеансов **screen**, к которым вы можете подключиться.

```
% screen -r
There are several suitable screens on:
     1212.pts-1.redtail      (Detached)
     1195.pts-1.redtail      (Detached)
     1225.pts-1.redtail      (Detached)
     17146.pts-1.sanctuary   (Dead ???)
Remove dead screens with 'screen -wipe'.
```

```
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
```

Запуск `screen -r 1212` подключит вас к первому в списке сеансу. Я упоминал ранее о том, насколько это удобно при удалённой работе. Если я авторизовался на удалённом сервере Slackware по `ssh`, и моё подключение было разорвано по каким-либо причинам (например, перебои с питанием на локальной машине), тогда вся моя работа катилась коту под хвост, что могло иметь ужасные последствия для настраиваемого мною сервера. С использованием `screen` такого не происходит. При разрыве соединения вы просто отключаетесь от сеанса. После восстановления соединения вы можете вновь подключиться к своему сеансу screen и спокойно продолжить работать.

Глава 9.

Структура файловой системы

Мы уже обсуждали структуру каталогов в Slackware Linux. По состоянию на текущий момент вы должны быть в состоянии самостоятельно найти нужные вам файлы и каталоги. Однако файловая система представляет собой нечно большее, чем просто структуру каталогов.

Linux является многопользовательской операционной системой. Каждый объект системы является многопользовательским, даже файловая система. Система хранит информацию наподобие той, кто является владельцем файла и кто может прочитать его. В файловой системе есть ещё и другие уникальные вещи, например, ссылки и точки монтирования NFS. В этом разделе обо всё этом будет подробно рассказано, а также о работе файловой системы в многопользовательском режиме.

9.1. Понятие владельца

В файловой системе хранится информация о владельце для каждого файла и каталога в системе. Под владельцем подразумеваются пользователь и группа, владеющие отдельно взятым файлом. Самым простым способом получения этой информации является использование команды `ls`:

```
% ls -l /usr/bin/wc
-rwxr-xr-x   1 root     bin    7368 Jul 30  1999 /usr/bin/wc
```

Здесь нас интересуют третья и четвёртая колонки. В них содержатся имена пользователя и группы, владеющие этим файлом. Мы видим, что в данном

случае владельцем файла является пользователь “root” и группа“bin”.

Владельцев файла можно легко изменить с помощью команд chown(1) (от англ. “change owner”) и chgrp(1) (от англ. “change group”). Чтобы изменить владельца файла на daemon, нужно воспользоваться командой chown:

```
# chown daemon /usr/bin/wc
```

Чтобы изменить группу, владеющую файлом, на “root”, нужно воспользоваться командой chgrp:

```
# chgrp root /usr/bin/wc
```

Также chown можно использовать, указав сразу и владельца, и группу:

```
# chown daemon:root /usr/bin/wc
```

В приведенном выше примере можно использовать точку вместо двоеточия. Результат будет таким же, однако рекомендуется использовать двоеточие. Использование точки для этих целей считается устаревшим и может быть удалено в следующих версиях chown. Такие имена пользователей становятся очень популярными на серверах Windows Exchange, а наиболее часто встречаются в адресах электронной почты, например: mr.jones@example.com. Администраторам Slackware Linux настоятельно рекомендуется избегать использования таких имён пользователей, потому что в некоторых скриптах точка всё ещё используется для обозначения пользователя и группы, владеющих файлом или каталогом. В нашем примере команда chmod интерпретировала бы mr.jones как пользователя “mr” и группу “jones”.

Наличие у файлов владельца является важным аспектом системы Linux, даже если вы используете её как простой пользователь. Иногда вам может понадобится изменить владельцев файлов и устройств.

9.2. Права доступа

Права доступа являются другим не менее важным аспектом использования файловой системы многими пользователями. Благодаря этому, вы можете определять, кто может читать, записывать и выполнять файлы.

Информация о правах доступа хранится в виде четырёх восьмеричных цифр, каждая из которых определяет различный набор прав доступа. Существуют такие права доступа: для пользователя, для группы и для остальных. Четвёртая восьмеричная цифра используется для хранения специальной информации: идентификатора пользователя (UID), идентификатора группы (GID) и sticky-бита. Ниже представлен перечень восьмеричных значених, присваиваемых различным режимам доступа (им также соответствуют буквенные обозначения, которые могут быть просмотрены программами типа `ls` и могут быть изменены командой `chmod`):

Таблица 9-1. Права доступа в виде восьмеричных цифр

Тип доступа	Восьмеричное значение	Буквенное обозначение
“sticky”-бит	1	t
установка ID пользователя	4	s
установка ID группы	2	s
чтение	4	r
запись	2	w
выполнение	1	x

Восьмеричные значения суммируются для каждой группы доступа. Например, если вам нужно чтобы группа имела право на “чтение” и “запись”, вам нужно использовать цифру “6” в той части информации о правах доступа, которая соответствует группе.

Права доступа к bash'у по умолчанию:

```
% ls -l /bin/bash
-rwxr-xr-x   1 root     bin  477692 Mar 21 19:57 /bin/bash
```

Для каталога первый дефис заменяется на букву "d". Затем следуют три группы прав доступа (владелец, группа и остальные). В примере видно, что владелец имеет права на чтение, запись и выполнение (rwx). Группа имеет только права на чтение и выполнение (r-x). И все остальные имеют права только на чтение и выполнение (r-x).

Каким же образом можно установить права доступа к другому файлу такие же, как у bash'а? Сначала давайте создадим файл для примера:

```
% touch /tmp/example
% ls -l /tmp/example
-rw-rw-r---  1 david    users    0 Apr 19 11:21 /tmp/example
```

Для изменения прав доступа к файлу example нужно воспользоваться командой chmod(1) (он англ. "change mode"). Просто добавьте необходимые восьмеричные цифры, чтобы получить требуемые права доступа. Чтобы пользователь имел права на чтение, запись и выполнение, мы получим цифру 7. Для чтения и записи - 5. Сложите их вместе (не суммируйте!) и передайте в качестве аргумента в chmod:

```
% chmod 755 /tmp/example
% ls -l /tmp/example
-rwxr-xr-x   1 david    users    0 Apr 19 11:21 /tmp/example
```

Вы могли бы задать вопрос: "Почему бы просто не создать файл уже с такими правами доступа?" Ответ очень прост. bash содержит небольшую встроенную команду под названием umask. Она включена в большинство командных процессоров Unix и управляет правами доступа, которые назначаются новым создаваемым файлам. Мы рассмотрели немного встроенных команд bash'а в Разд. 8.3.1. Работа umask очень похожа на работу chmod, только в обратном порядке. Вы указываете восьмеричные

значения, которые не должны присутствовать в правах для создаваемых файлов. По умолчанию значение **umask** составляет `0022`.

```
% umask
0022
% umask 0077
% touch tempfile
% ls -l tempfile
-rw--------  1 david    users    0 Apr 19 11:21 tempfile
```

Дополнительную информацию смотрите на странице руководства `bash`.

Чтобы установить специальные права доступа с помощью `chmod`, просуммируйте цифры и укажите полученное число в первой колонке. Например, чтобы установить **ID** пользователя и **ID** группы, нужно использовать в первой колонке цифру **6**:

```
% chmod 6755 /tmp/example
% ls -l /tmp/example
-rwsr-sr-x   1 david    users    0 Apr 19 11:21 /tmp/example
```

Если вас смущают восьмеричные числа, при работе с `chmod` вы можете использовать буквы. Наборы прав доступа:

Владелец	**u**
Группа	**g**
Остальные	**o**
Всё выше перечисленное	**a**

Для получения результатов из приведенных выше примеров нам потребуется воспользоваться несколькими командами:

```
% chmod a+rx /tmp/example
% chmod u+w /tmp/example
% chmod ug+s /tmp/example
```

Некотрые люди предпочитают использовать буквы, а не числа. В любом

случае права доступа будут получены точно такие же.

Работа с восьмеричным форматом зачастую оказывается быстрее, и вы часто будете встречать его в shell-скриптах. Однако иногда использование букв оказывается более продуктивным. Например, при использовании восьмеричного формата нет простого способа изменения прав доступа к файлам и каталогам для группы, на затрагивающего при этом других групп. А при использовании букв - это простая задача.

```
% ls -l /tmp/
-rwxr-xr-x   1 alan    users    0 Apr 19 11:21 /tmp/example0
-rwxr-x---   1 alan    users    0 Apr 19 11:21 /tmp/example1
----r-xr-x   1 alan    users    0 Apr 19 11:21 /tmp/example2
% chmod g-rwx /tmp/example?
-rwx---r-x   1 alan    users    0 Apr 19 11:21 /tmp/example0
-rwx------   1 alan    users    0 Apr 19 11:21 /tmp/example1
-------r-x   1 alan    users    0 Apr 19 11:21 /tmp/example2
```

Выше мы несколько раз упоминали о правах доступа, связанных установкой ID пользователя (SIUD, set user ID) и ID группы (SGID, set group ID). Вас могло заинтересовать, что же это такое. Обычно, когда вы запускаете программу, она работает под вашей учётной записью. Другими словами она имеет права, которые есть у вас как у пользователя. То же самое касается и группы. Когда вы запускаете программу, она выполняется с правами вашей текущей группы. С правами доступа, устанавливающими идентификатор пользователя, вы можете заставить программу всегда выполняться с правами её владельца (например "root"'а). Установка идентификатора группы работает точно так же, но только по отношению к группе.

Будьте осторожны с этими битами. SUID- и SGID-программы могут нанести серьёзный ущерб безопасности вашей системы. Если вы часто устанавливаете SUID-биты на программы, владельцем которых является `root`, тем самым вы позволяете любым пользователям запускать эти программы с правами `root`'а. Поскольку он не имеет никаких ограничений в системе, вы можете представить, насколько это может быть опасным

для безопасности вашей системы. Короче говоря, устанавливать ID пользователя и группы не так уж и плохо, однако используйте их только в общих случаях.

9.3. Ссылки

Ссылки представляют собой указатели на файлы. С помощью ссылок вы можете сделать так, что файл будет присутствовать одновременно в нескольких местах и может иметь различные имена. Существует два типа ссылок: жёсткие и мягкие.

Жёсткие ссылки - это по сути различные имена одного отдельно взятого файла. Они могут существовать в пределах только одной файловой системы и удаляются только в том случае, когда из системы удаляется настоящее имя. Они полезны в некоторых случаях, однако большинство пользователей находят использование мягких ссылок более гибким и удобным.

Мягкие ссылки (часто называемые символическими ссылками или “симлинками”) могут указывать на файлы, находящиеся за пределами файловой системы. На самом деле это небольшие файлы, содержащие необходимую информацию. Вы можете добавлять и удалять символические ссылки, не затрагивая при этом файлы, на которые они указывают. И поскольку симлинки представляют собой файлы со своей собственной информацией, они могут указывать даже на каталоги. Например, довольно часто `/var/tmp` является символической ссылкой на каталог `/tmp`.

Ссылки не имеют собственного набора прав доступа или владельцев, зато эти свойства есть у файлов, на которые они указывают. Вот общий пример:

```
% ls -l /bin/sh
lrwxrwxrwx   1 root     root     4 Apr  6 12:34 /bin/sh -> bash
```

Командный процессор `sh` в Slackware - это на самом деле `bash`. Удаление ссылок выполняется с помощью `rm`. Для создания ссылок используется команда `ln`. Эти команды будут рассмотрены более подробно в Гл. 10.

Однако с символическими ссылками надо соблюдать большую осторожность. В частности на машинах, в которых каждую ночь выполняется резервное копирование данных на магнитную ленту. Был случай, когда были созданы два симлинка на каталоги, ссылающиеся друг на друга. Программа резервировного копирования продолжала добавлять эти два каталога на магнитную ленту до тех пор, пока на ней не закончилось свободное пространство. Обычно набор проверок предотвращает создание символической ссылки в такой ситуации, однако наш случай был особым.

9.4. Монтирование устройств

Как было уже рассмотрено ранее в Разд. 4.1.1, все диски и устройства в вашем компьютере являются частью одной большой файловой системы. Разделы жёсткого диска, CD-ROM'ы и дискеты находятся в одном дереве файлов. Для того, чтобы подключить эти носители к файловой системе, и вы могли получить доступ к ним, вам необходимо использовать команды `mount`(1) и `umount`(1).

Некоторые устройства монтируются автоматически во время загрузки системы. Они перечислены в файле `/etc/fstab`. Для любого устройства, которое нужно монтировать автоматически, должна присутствовать своя запись в этом файле. Для использования остальных накопителей вам каждый раз понадобится выполнять команду монтирования.

`fstab`

Давайте рассмотрим пример файла `/etc/fstab`:

```
% cat /etc/fstab
/dev/sda1       /               ext2        defaults        1   1
/dev/sda2       /usr/local      ext2        defaults        1   1
/dev/sda4       /home           ext2        defaults        1   1
/dev/sdb1       swap            swap        defaults        0   0
/dev/sdb3       /export         ext2        defaults        1   1
none            /dev/pts        devpts      gid=5,mode=620  0   0
none            /proc           proc        defaults        0   0
/dev/fd0        /mnt            ext2        defaults        0   0
/dev/cdrom      /mnt/cdrom      iso9660     ro              0   0
```

В первой колонке указывается название устройства. В нашем случае устройства - это пять разделов, распределённые по двух жёстким **SCSI**-дискам, две специальные файловые системы, для которых не нужны устройства, дискета и **CD-ROM**. Во второй колонке указывается точка монтирования устройства. Это должно быть название каталога за исключением случая с разделом для свопинга. В третьей колонке находится тип файловой системы устройства. Для **Linux** зачастую это будут файловые системы `ext2` (устарела), `ext3`, `reiserfs`, `xfs` и `jfs`. Приводы **CD-ROM** и **DVD** обозначаются как `iso9660` и `auto`, а устройствами системы **Windows** будут `msdos`, `vfat` или `ntfs`.

В четвёртой колонке перечисляются параметры, применяемые к примонтированным файловым системам. Значение **defaults** прекрасно подойдёт практически во всех случаях. Однако для устройств, используемых только для чтения, следует добавить флаг `ro`. Существует ещё много других параметров. Чтобы узнать о них больше, обратитесь к странице руководства `fstab`(5). Последние две колонки используются утилитой `fsck` и другими программами, которым нужно работать с устройствами. Для получения более подробной информации также обращайтесь к странице руководства.

При установке **Slackware Linux** программа **setup** создаст за вас основную часть файла `fstab`.

mount и *umount*

В подключении устройства к вашей файловой системе нет ничего сложного. Всё, что вам нужно сделать - выполнить команду `mount` с несколькими опциями. Использование `mount` можно упростить, если добавить в файл `/etc/fstab` пункт с монтируемым устройством. Например, допустим, что нам нужно примонтировать CD-ROM, а наш файл `fstab` похож на тот, что был представлен в предыдущем разделе. Тогда нам нужно вызвать `mount` таким образом:

```
% mount /cdrom
```

Поскольку в `fstab` есть запись с указанной точкой монтирования, для `mount` будет известно, какие опции нужно использовать. Если записи для этого устройства нет, тогда `mount` нужно использовать с набором параметров:

```
% mount -t iso9660 -o ro /dev/cdrom /cdrom
```

В эту команду включена так же самая информация, что и в примере файла `fstab`, однако нам всё равно нужно указать все части. Здесь `-t iso9660` - это тип файловой системы монтируемого устройства. В нашем случае это файловая система iso9660, которая наиболее часто используется на накопителях CD-ROM. Опция `-o ro` сообщает о том, что устройство используется только для чтения. `/dev/cdrom` - это название монтируемого устройства, а `/cdrom` - точка монтирования в файловой системе.

Перед тем, как извлечь дискету, CD-ROM или любой другой съёмный накопитель, который в данный момент примонтирован, вы должны сначала отмонтировать его. Это выполняется командой `umount`. Не спрашивайте, куда подевалась буква “n”, потому что мы не сможем ответить вам (правильно должно звучать “unmount” - прим. переводчика). В качестве аргумента к `umount` вы можете использовать название устройства или точку его монтирования. Например, если вам нужно отмонтировать CD-ROM из предыдущего примера, подойдёт любая из этих команд:

```
# umount /dev/cdrom
# umount /cdrom
```

9.5. Монтирование NFS

NFS означает Network Filesystem (сетевая файловая система). Она не является полноценной частью реальной файловой системы, однако может использоваться для добавления частей к примонтированной файловой системе.

В больших Unix-средах часто применяется совместное использование одних и тех же программ, наборов домашних каталогов и почтовых спулов. Проблема предоставления всем машинам одинаковых копий решается с помощью NFS. Мы можем использовать NFS для предоставления всем рабочим станциям общего доступа к набору домашних каталогов. Рабочие станции могут монтировать ресурсы NFS так, как если бы они находились на их собственных машинах.

Дополнительную информацию смотрите в Разд. 5.6.2 и страницах руководства `exports`(5), `nfsd`(8) и `mountd`(8).

Глава 10.

Работа с файлами и каталогами

Linux старается быть как можно более похожим на Unix. Исторически так сложилось, что операционные системы Unix были ориентированы на использование командной строки. И хоть в Slackware есть графический интерфейс, командная строка всё ещё остаётся главным средством управления всей системой. Поэтому важно знать и понимать действие некоторых основных команд для работы с файлами.

В следующих разделах описаны общие команды управления файлами и представлены примеры их использования. Существует ещё большое число других команд, однако ряд представленных команд поможет вам начать изучать их. Также здесь даны только краткие описания. Более подробную информацию вы найдёте в соответствующих страницах руководства по каждой из команд.

10.1. Навигация: *ls*, *cd* и *pwd*

ls

Эта команда выводит перечень файлов каталога. Пользователи Windows и DOS обратят внимание на схожесть её с командой `dir`. Запуск `ls`(1) без опций выведет перечень файлов только текущего каталога. Чтобы увидеть, что находится в корневом каталоге, вы можете выполнить следующие команды:

```
% cd /
% ls
bin   cdr    dev  home  lost+found  proc  sbin   tmp  var
boot  cdrom  etc  lib   mnt         root  suncd  usr  vmlinuz
```

Для большинства людей будет проблемой определить, что в данном выводе является каталогом, а что - файлом. Некоторые пользователи предпочитают, чтобы `ls` добавлял идентификатор типа к каждому пункту перечня, например так:

```
% ls -FC
bin/   cdr/    dev/  home/  lost+found/  proc/  sbin/   tmp/  var/
boot/  cdrom/  etc/  lib/   mnt/         root/  suncd/  usr/  vmlinuz
```

У каталогов в конце имени добавляется косая черта, у исполняемых файлов - звёздочка и т.д.

Также `ls` можно использовать для получения другой информации о файлах. Например, чтобы увидеть дату создания, владельцев и права доступа, нужно использовать длинный формат:

```
% ls -l
drwxr-xr-x   2 root     bin          4096 May  7 09:11 bin/
drwxr-xr-x   2 root     root         4096 Feb 24 03:55 boot/
drwxr-xr-x   2 root     root         4096 Feb 18 01:10 cdr/
drwxr-xr-x  14 root     root         6144 Oct 23 18:37 cdrom/
drwxr-xr-x   4 root     root        28672 Mar  5 18:01 dev/
drwxr-xr-x  10 root     root         4096 Mar  8 03:32 etc/
drwxr-xr-x   8 root     root         4096 Mar  8 03:31 home/
drwxr-xr-x   3 root     root         4096 Jan 23 21:29 lib/
drwxr-xr-x   2 root     root        16384 Nov  1 08:53 lost+found/
drwxr-xr-x   2 root     root         4096 Oct  6 12:47 mnt/
dr-xr-xr-x  62 root     root            0 Mar  4 15:32 proc/
drwxr-x--x  12 root     root         4096 Feb 26 02:06 root/
drwxr-xr-x   2 root     bin          4096 Feb 17 02:02 sbin/
drwxr-xr-x   5 root     root         2048 Oct 25 10:51 suncd/
drwxrwxrwt   4 root     root       487424 Mar  7 20:42 tmp/
drwxr-xr-x  21 root     root         4096 Aug 24 03:04 usr/
drwxr-xr-x  18 root     root         4096 Mar  8 03:32 var/
```

Допустим, что вам нужно получить перечень скрытых файлов в текущем каталоге. Следующая команда поможет вам сделать это:

```
% ls -a
.              bin   cdrom  home        mnt   sbin   usr
..             boot  dev    lib         proc  suncd  var
.pwrchute_tmp  cdr   etc    lost+found  root  tmp    vmlinuz
```

Файлы, чьи имена начинаются с точки, (т.н. dot-файлы) являются скрытыми, когда вы выполняете `ls`. Такие файлы вы увидите только, если будете использовать опцию `-a`.

Есть ещё много других опций, которые можно найти на странице руководства. Не забывайте, что можно комбинировать параметры, передаваемые в `ls`.

cd

Команда `cd` используется для смены рабочего каталога. Вам просто надо набрать `cd` с путём к каталогу, в который нужно перейти. Вот несколько примеров:

```
darkstar:~$ cd /bin
darkstar:/bin$ cd usr
bash: cd: usr: No such file or directory
darkstar:/bin$ cd /usr
darkstar:/usr$ ls
bin
darkstar:/usr$ cd bin
darkstar:/usr/bin$
```

Обратите внимание, что без указания слэша в начале пути, команда пытается перейти в каталог, находящийся в текущем каталоге. Вызов `cd` без параметров переместит вас в ваш домашний каталог.

Команда `cd` не похожа на другие утилиты. Это встроенная команда шелла. Эти команды рассматриваются в Разд. 8.3.1. Сейчас это для вас может

не иметь значения. В основном это означает, что для таких команд нет страниц руководства. Вместо этого вам нужно воспользоваться встроенной справкой командного процессора. Например:

```
% help cd
```

При этом будут показаны опции команды `cd` и их использование.

pwd

Команда `pwd` используется для информирования вас о вашем текущем местонахождении. Чтобы воспользоваться командой `pwd`, просто наберите её. Например:

```
% cd /bin
% pwd
/bin
% cd /usr
% cd bin
% pwd
/usr/bin
```

10.2. Пейджеры: *more*, *less* и *most*

more

`more`(1) - это то, что мы называем “пейджером” (от англ. *page* - страница). Довольно часто данных, выводимых определённой командой, оказывается слишком много, чтобы уместиться на одном экране. Отдельные команды не знают, как уместить свои выходные данные на нескольких страницах. Эту работу они оставляют утилитам, называемых пейджерами.

Команда `more` разбивает выходные данные на отдельные экраны и ожидает от вас нажатия на пробел, перед тем как вывести следующую порцию. Нажатие на клавишу Enter сдвинет вывод только на одну следующую строку, а не на целую страницу. Вот хороший пример:

```
% cd /usr/bin
% ls -l
```

При этом на экран будет выведен большой перечень файлов. Чтобы разбить эти данные на несколько страниц, просто отправьте их по конвейеру в команду more:

```
% ls -l | more
```

“По конвейеру” означает через специальный символ (shift + обратная косая черта). Конвейер - это сокращённое обозначение перенаправления выходного потока команды `ls` во входной поток команды `more`. По конвейеру вы можете отправить в команду `more` практически всё, что угодно, не только вывод команды `ls`. Также работа с конвейерами рассмотрена в Разд. 8.2.3.

less

Команда `more` довольно удобна, однако часто оказывается, что вы уже пролистали нужный вам экран. more не предоставляет возможности прокрутки назад. А команда `less`(1) имеет такую функцию. Она используется таким же образом, как и `more`, поэтому предыдущие примеры также будут работать с ней. Таким образом `less` больше, чем `more` (англ., игра слов: *less* - меньше, *more* - больше). Джуст Кремерс (Joost Kremers) так говорит от этом:

> `less` is more, but `more` then `more` is, so `more` is less `less`, so use more `less`, if you want less `more`.

(Сможете перевести? Я буду рад услышать ваш вариант - прим. переводчика.)

most

Там, где за бортом остаются `more` и `less`, появляется `most`(1). Если `less` больше, чем `more`, то `most` больше, чем `less` (от англ. ***most*** - наибольший). В то время как остальные пейджеры могут показывать единовременно только один файл, `most` в состоянии работать с любым количеством файлов, лишь бы каждое окно с файлом было размером по крайне мере в две строки. У `most` есть много опций, подробнее о них смотрите в странице руководства.

10.3. Простой вывод: *cat* и *echo*

cat

`cat`(1) - это сокращение от слова "concatenate" (связывать, соединять). Изначально эта утилита была разработана для объединения текстовых файлов в один большой файл, но её можно использовать для многих других целей.

Чтобы объединить два или более файлов в один, нужно просто перечислить их имена после команды `cat`, а затем перенаправить вывод в новый выходной файл. `cat` работает со стандартными потоками ввода и вывода, поэтому вам необходимо использовать символы перенаправления командного процессора. Например:

```
% cat file1 file2 file3 > figfile
```

Эта команда возьмёт содержимое `file1`, `file2` и `file3` и объединит их вместе. Новые выходные данные отправляются на стандартный вывод.

`cat` также можно использовать для отображения файлов. Многие люди выводят `cat`'ом текстовые файлы через команды `more` или `less`, например так:

```
% cat file1 | more
```

При этом содержимое `file1` будет выведено на экран и передано по конвейеру в команду `more`, чтобы файл можно было смотреть постранично.

Другим общим способом использования команды `cat` является копирование файлов. С помощью `cat` Вы можете скопировать любой файл, например так:

```
% cat /bin/bash > ~/mybash
```

Здесь программа `/bin/bash` была скопирована в ваш домашний каталог в файл с именем `mybash`.

Существует много вариантов использования `cat`, и здесь описаны только некоторые из них. Поскольку в `cat` широко используются стандартные потоки ввода и вывода, эта команда идеально подходит для использования в шелл-скриптах или как часть сложных команд.

echo

Команда `echo`(1) выводит на экран указанный текст. Выводимую на экран строку необходимо указать после самой команды `echo`. По умолчанию `echo` выведет саму сроку, а после неё - знак новой строки. Вы можете использовать опцию `-n`, чтобы не добавлять после вывода новую строку. Опция `-e` заставит `echo` искать в строке escape-символы и выполнять их.

10.4. Создание: *touch* и *mkdir*

touch

Команда `touch`(1) используется для изменения временной метки файла. С помощью этой команды вы можете изменять время доступа к файлу и время его изменения. Если указанный файл не существует, `touch` создаст файл нулевого размера с указанным именем. Чтобы пометить файл текущим временем системы, выполните следующую команду:

```
% ls -al file1
-rw-r--r--    1 root     root         9779 Feb  7 21:41 file1
% touch file1
% ls -al file1
-rw-r--r--    1 root     root         9779 Feb  8 09:17 file1
```

Для `touch` есть много различных опций, включая указание типа изменяемой временной метки, явное указание времени и многие другие. На странице руководства есть исчерпывающая информация об этих опциях.

mkdir

Команда `mkdir`(1) создаёт новые каталоги. При запуске это команды просто укажите имя каталога, который вы хотите создать. В следующем примере создаётся каталог `hejaz`, находящийся в текущем каталоге:

```
% mkdir hejaz
```

Вы также можете указать путь, наподобие этого:

```
% mkdir /usr/local/hejaz
```

С опцией `-p` `mkdir` создаст все родительские каталоги. В приведенном выше примере произойдёт ошибка, если не существует каталог `/usr/local`. С указанием опции `-p` будут созданы каталоги `/usr/local` и `/usr/local/hejaz`:

```
% mkdir -p /usr/local/hejaz
```

10.5. Копирование и перемещение

cp

Команда cp(1) копирует файлы. Пользователи DOS могут заметить, что она похожа на команду copy. Для cp есть много опций, поэтому перед тем, как использовать её, прочтите страницу руководства.

Общим вариантом использования cp является копирование файла из одного местоположения в другое. Например:

```
% cp hejaz /tmp
```

При этом файл hejaz из текущего каталога будет скопирован в каталог /tmp.

Многие пользователи предпочитают оставлять без изменений временные метки файлов, как в этом примере:

```
% cp -a hejaz /tmp
```

При этом в копии временные метки не изменяются.

Чтобы рекурсивно скопировать содержимое каталога в другой каталог, воспользуйтесь следующей командой:

```
% cp -R mydir /tmp
```

При этом каталог *mydir* будет полностью скопирован в каталог /tmp.

Также, если вы хотите скопировать каталог или файл о оставить без изменений все права доступа и временные метки, используйте cp -p.

```
% ls -l file
```

```
-rw-r--r--    1 root     vlad            4 Jan  1 15:27 file
% cp -p file /tmp
% ls -l /tmp/file
-rw-r--r--    1 root     vlad            4 Jan  1 15:27 file
```

У команды `cp` есть много опций, которые подробно описаны на её странице руководства.

mv

Команда `mv`(1) перемещает файлы из одного местоположения в другое. Звучит довольно просто, не так ли?

```
% mv oldfile /tmp/newfile
```

У команды `mv` есть несколько полезных опций командной строки, которые подробно описаны на странице руководства. На практике `mv` с опциями практически не используется.

10.6. Удаление: *rm* и *rmdir*

rm

Команда `rm`(1) удаляет файлы и целые деревья каталогов. Пользователи DOS найдут её похожей на команды `del` и `deltree`. `rm` может быть очень опасной, если вы не следите за своими действиями. Хотя в некоторых случаях можно восстановить только что удалённый файл, это может оказаться слишком сложным процессом (и, возможно, дорогостоящим), и его рассмотрение выходит за рамки этой книги.

Чтобы удалить один файл, укажите его имя при запуске rm:

```
% rm file1
```

Если для файла отсутствуют права на запись, вы можете получить сообщение об ошибке с отказом в доступе. Чтобы принудительно удалить неважно какой файл, используйте опцию `-f`, например так:

```
% rm -f file1
```

Чтобы удалить целый каталог, используйте одновременно две опции `-r` и `-f`. Ниже представлен хороший пример, удаляющий всё содержимое на вашем жёстком диске. И вам наверняка не захочется его выполнить. А вот и сама команда:

```
# rm -rf /
```

Будьте осторожны при работе с `rm`. Вы можете "выстрелить себе в ногу". У этой команды также есть различные опции, которые хорошо описаны на странице руководства.

rmdir

Команда `rmdir`(1) удаляет каталоги из файловой системы. Каталог должен быть пуст перед удалением. Синтаксис довольно прост:

```
% rmdir <каталог>
```

В этом примере в текущем каталоге удаляется подкаталог `hejaz`:

```
% rmdir hejaz
```

Если удаляемый каталог не существует, `rmdir` сообщит вам об этом. Вы также можете указать полный путь к каталогу, например, так:

```
% rmdir /tmp/hejaz
```

В этом примере будет сделана попытка удаления каталога `hejaz`, находящегося в каталоге `/tmp`.

Вы также можете удалить каталог и все его родительские каталоги с помощью опции `-p`.

```
% rmdir -p /tmp/hejaz
```

В этом примере будет сначала сделана попытка удаления каталога `hejaz`, находящегося в каталоге `/tmp`. Если попытка удастся, утилита попытается удалить `/tmp`. Работа `rmdir` будет продолжаться до тех пор, пока не возникнет ошибка или не будет удалено всё указанное дерево.

10.7. Связывание файлов с помощью *ln*

Команда `ln`(1) используется для установки связей между файлами. Эти связи могут быть жёсткими или мягкими (символическими) ссылками. Разница между этими двумя типами ссылок обсуждалась в Разд. 9.3. Если вам нужно создать символическую ссылку на каталог `/var/media/mp3` и поместить её (ссылку) в свой домашний каталог, выполните следующую команду:

```
% ln -s /var/media/mp3 ~/mp3
```

Опция `-s` сообщает `ln`'у о том, что нужно создать именно символическую ссылку, а не жёсткую. Следующей опцией является цель ссылки и последняя опция - имя ссылки. В данном случае в вашем домашнем каталоге будет создан файл с именем `mp3`, указывающий на `/var/media/mp3`. Вы можете дать ссылке любое название, просто изменив в команде последний параметр.

Создать жёсткую ссылку так же легко. Всё, что вам нужно сделать - это опустить опцию `-s`. Однако жёсткие ссылки обычно не могут ссылаться на каталоги или файлы на других файловых системах. Чтобы создать

жёсткую ссылку `/usr/bin/email`, указывающую на `/usr/bin/mutt`, выполните следующую команду:

```
# ln /usr/bin/mutt /usr/bin/email
```

Глава 11.

Управление процессами

Каждая выполняемая программа называется процессом. Эти процессы варьируются от вещей типа системы X Window до системных программ (демонов), которые запускаются во время загрузки системы. Каждый процесс выполняется от имени какого-либо пользователя. Процессы, запускаемые во время загрузки, обычно выполняются от имени `root`'а или `nobody`. Процессы, запускаемые вами, выполняются с вашим правами. Процессы, запускаемые другим пользователями, работают соответственно с правами этих пользователей.

Вы имеете полный контроль над процессами, которые вы запустили. Кроме того `root` может управлять всеми процессами в системе, включая те, что были запущены другими пользователями. Процессами можно управлять и наблюдать за ними с помощью различных программ, а также с помощью некоторых команд в шелле.

11.1. Перевод в фоновый режим

Программы, запускаемые из командной строки, начинают работать в приоритетном режиме (foreground). Это позволяет вам видеть данные, выводимые программой, и взаимодействовать с ней. Однако в некоторых случаях вам хотелось бы, чтобы программа работала, не захватывая полностью ваш терминал. Это называется работой программы в фоновом режиме (background), и для осуществления этой задачи существует несколько способов.

Первым способом перевода процесса в фоновый режим является добавление в командную строку амперсанда при запуске программы. Например, допустим, что вы хотите воспользоваться консольным mp3-плеером `amp` для воспроизведения mp3-файлов, однако в то же время вам нужно работать в том же самом терминале. Следующая команда запустит amp в фоновом режиме:

```
% amp *.mp3 &
```

Программа запустится и будет работать как обычно, а вы снова вернётесь в командную строку.

Другим способом перевода процесса в фоновый режим является осуществление этого во время работы процесса. Сначала запустите программу. Во время её работы нажмите **Control+z**. При этом процесс будет приостановлен. Иными словами приостановленный процесс "ставится на паузу". Он мгновенно прекращает свою работу, но может быть вновь запущен в любое время. После того, как вы приостановили процесс, вы возвращаетесь в командную строку. Затем вы можете перевести процесс в фоновый режим, набрав:

```
% bg
```

При этом приостановленный процесс вновь продолжит свою работу, но уже в фоновом режиме.

11.2. Перевод в приоритетный режим

Если вам нужно взаимодействовать с фоновым процессом, вы можете снова перевести его в приоритетный режим. Если у вас выполняется только один фоновый процесс, вы можете переключиться на него, набрав:

```
% fg
```

Если программа не завершила своё выполнение, она возьмёт под свой контроль ваш терминал, и вы не сможете получить доступ к приглашению командной строки. Иногда программа завершает своё выполнение, работая в фоновом режиме. В этом случае вы получите сообщение типа:

```
[1]+  Done                    /bin/ls $LS_OPTIONS
```

Это означает, что фоновый процесс (в данном случае `ls` - не слишком интересно) завершил свою работу.

Одновременно могут выполняться несколько фоновых процессов. В этом случае вам нужно узнать, какой именно процесс вам нужно перевести назад в приоритетный режим. Просто набрав `fg`, вы вернёте назад процесс, который был последним переведён в фоновый режим. А что, если у вас целый список этих фоновых процессов? К счастью в bash'е есть команда для вывода перечня всех заданий. Называется она `jobs` и её вывод представляет собой нечто, похожее на следующее:

```
% jobs
[1]   Stopped                 vim
[2]-  Stopped                 amp
[3]+  Stopped                 man ps
```

По сути это список всех процессов, переведённых в фоновый режим. Как видите, все они остановлены (stopped). Это означает, что работа этих процессов приостановлена. Числа слева - это идентификаторы, согласно которым сортируются все фоновые процессы. Идентификатором с плюсом (`man ps`) отмечен процесс, который будет переведён в приоритетный режим, если вы наберёте просто `fg`.

Если вам нужно переключиться в `vim`, наберите следующее:

```
% fg 1
```

и `vim` возьмёт консоль под свой контроль. Перевод процессов в фоновый режиме весьма полезен, если у вас есть только один терминал через коммутируемое подключение. У вас может быть запущено несколько

программ, работающих в одном терминале, между которыми вы можете периодически переключаться.

11.3. *ps*

Итак, теперь вы знаете как переключаться между несколькими процессами, запущенными из командной строки. А также вы знаете, что в системе постоянно выполняется много других процессов. Так каким же образом можно получить перечень всех этих программ? Для этого вам нужно воспользоваться командой `ps`(1). У этой команды есть много опций, поэтому мы рассмотрим только самые важные из них. Чтобы получить полный список опций, обратитесь к странице руководства ps. Использование страниц руководства подробно рассмотрено в Разд. 2.1.1.

Просто наберите `ps`, чтобы получить список программ, выполняемых в вашем терминале. В этот список входят процессы, работающие в приоритетном режиме (любой командный процессор, в котором вы работаете, и сама команда `ps`). Также перечисляются фоновые процессы, которые могут выполняться в данный момент. В большинстве случаев это будет очень короткий список:

Рисунок 11-1. Наиболее общий вывод команды *ps*

```
% ps
   PID TTY          TIME CMD
  7923 ttyp0    00:00:00 bash
  8059 ttyp0    00:00:00 ps
```

И хотя процессов здесь не так уж и много, это весьма типичная информация. Вы увидите те же самые колонки при использовании ps, независимо от того, сколько процессов выполняется в системе. Что же это тогда значит?

PID - это *идентификатор процесса* (process ID). Все работающие процессы имеют уникальные идентификаторы в диапазоне от 1 до 32767. Каждому новому процессу присваивается следующий свободный PID. Когда процесс завершает свою работу (или убивается, как вы увидите в следующем разделе), он освобождает свой PID. При достижении в системе максимального PID, следущим берётся первый свободный PID с наименьшим номером и так по кругу.

Колонка TTY обозначает терминал, в котором выполняется процесс. При простом вызове ps будет выведен список только тех программ, которые выполняются в текущем терминале. Поэтому все процессы будут иметь одну и ту же информацию в колонке TTY. Как видно в примере, оба процесса в списке выполняются в терминале ttyp0. Это означает, что они выполняются или удалённо, или в каком-нибудь X-терминале.

Колонка TIME содержит данные о времени, в течение которого процесс использует ресурсы центрального процессора. Однако это не то время, в течение которого выполняется процесс. Помните, что Linux является многозадачной операционной системой. В ней постоянно выполняется множество процессов и каждый из них получает небольшую порцию процессорного времени. Поэтому колонка TIME должна содержать для каждого процесса как можно меньшее значение времени по сравнению с тем временем, в течение которого выполняется этот процесс. Если вы видите в этой колонке значение более нескольких минут, это может означать, что что-то идёт не так.

И наконец в колонке CMD представлена сама программа. В ней отображается только имя программы, безо всяких опций командной строки или подобной информации. Для получения этой информации вам нужно использовать одну из множества опций команды ps. Мы вкратце рассмотрим их здесь.

Вы можете получить полный список процессов, выполняемых в вашей системе, используя правильный набор опций. Это скорее всего приведёт к тому, что вы получите большой список процессов (на моём ноутбуке на время написания этого предложения их было 55 (а на моём ПК - 110 :) -

прим. переводчика), поэтому я несколько сокращу вывод:

```
% ps -ax
 PID TTY      STAT   TIME COMMAND
   1 ?        S      0:03 init [3]
   2 ?        SW     0:13 [kflushd]
   3 ?        SW     0:14 [kupdate]
   4 ?        SW     0:00 [kpiod]
   5 ?        SW     0:17 [kswapd]
  11 ?        S      0:00 /sbin/kerneld
  30 ?        SW     0:01 [cardmgr]
  50 ?        S      0:00 /sbin/rpc.portmap
  54 ?        S      0:00 /usr/sbin/syslogd
  57 ?        S      0:00 /usr/sbin/klogd -c 3
  59 ?        S      0:00 /usr/sbin/inetd
  61 ?        S      0:04 /usr/local/sbin/sshd
  63 ?        S      0:00 /usr/sbin/rpc.mountd
  65 ?        S      0:00 /usr/sbin/rpc.nfsd
  67 ?        S      0:00 /usr/sbin/crond -l10
  69 ?        S      0:00 /usr/sbin/atd -b 15 -l 1
  77 ?        S      0:00 /usr/sbin/apmd
  79 ?        S      0:01 gpm -m /dev/mouse -t ps2
  94 ?        S      0:00 /usr/sbin/automount /auto file /etc/auto.misc
 106 tty1     S      0:08 -bash
 108 tty3     SW     0:00 [agetty]
 109 tty4     SW     0:00 [agetty]
 110 tty5     SW     0:00 [agetty]
 111 tty6     SW     0:00 [agetty]
 [вывод сокращён]
```

Большинство этих процессов запускаются во время загрузки на большинстве систем. Я внёс некоторые изменения в свою систему, поэтому ваш список наверняка будет совершенно иным. Однако вы увидите большинство этих процессов и в своей системе. Как видите, эти опции позволили получить опции командной строки выполняемых процессов. Уязвимость в ядре в функции `ptrace` стала причиной исправления кода ядра, в результате которого для многих выполняемых процессов больше не показываются опции командной строки. В данном примере названия этих процессов заключены в квадратные скобки, например у процессов с PID'ами от 108 до 111. В этом листинге также появилось несколько новых колонок и некоторая другая интересная информация.

Во-первых, вы заметите, что большинство процессов в списке работают в tty"?". Они не привязаны ни к одному из терминалов. Это является самым распространённым случаем для демонов, т.е. процессов, которые выполняются без привязки к какому-либо терминалу. Примеры демонов: sendmail, BIND, apache и NFS. Обычно они ожидают запросов от клиентов и возвращают информацию на полученные запросы.

Во-вторых, здесь появился новый столбец: `STAT`. В нём отображается состояние процесса. `S` означает, что процесс спит (sleeping), т.е. ожидает какого-либо события. `Z` означает процесс-зомби. Такой процесс появляется в том случае, когда умирает его родительский процесс, оставив после себя дочерний процесс. Это нехорошая ситуация. `D` означает процесс, который перешёл в "непробудный" сон. Часто такие процессы невозможно убить даже путём отправки им сигнала SIGKILL. Вы узнаете больше о SIGKILL в следующем разделе, посвящёном команде `kill`. `W` означает paging (страничную подкачку файлов). Мёртвые процессы обозначаются как `X`. Процессы, отмеченные как `T`, трассируются или остановлены. `R` означает, что процесс можно запустить.

Если вы хотите получить ещё более подробную информацию о выполняемых процессах, попробуйте выполнить следующую команду:

```
% ps -aux
 USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
 root         1  0.0  0.0   344   80 ?        S    Mar02   0:03 init [3]
 root         2  0.0  0.0     0    0 ?        SW   Mar02   0:13 [kflushd]
 root         3  0.0  0.0     0    0 ?        SW   Mar02   0:14 [kupdate]
 root         4  0.0  0.0     0    0 ?        SW   Mar02   0:00 [kpiod]
 root         5  0.0  0.0     0    0 ?        SW   Mar02   0:17 [kswapd]
 root        11  0.0  0.0  1044   44 ?        S    Mar02   0:00 /sbin/kerneld
 root        30  0.0  0.0  1160    0 ?        SW   Mar02   0:01 [cardmgr]
 bin         50  0.0  0.0  1076  120 ?        S    Mar02   0:00 /sbin/rpc.port
 root        54  0.0  0.1  1360  192 ?        S    Mar02   0:00 /usr/sbin/sysl
 root        57  0.0  0.1  1276  152 ?        S    Mar02   0:00 /usr/sbin/klog
 root        59  0.0  0.0  1332   60 ?        S    Mar02   0:00 /usr/sbin/inet
 root        61  0.0  0.2  1540  312 ?        S    Mar02   0:04 /usr/local/sbi
 root        63  0.0  0.0  1796   72 ?        S    Mar02   0:00 /usr/sbin/rpc.
 root        65  0.0  0.0  1812   68 ?        S    Mar02   0:00 /usr/sbin/rpc.
 root        67  0.0  0.2  1172  260 ?        S    Mar02   0:00 /usr/sbin/cron
```

```
root         77  0.0  0.2  1048  316 ?        S    Mar02   0:00 /usr/sbin/apmd
root         79  0.0  0.1  1100  152 ?        S    Mar02   0:01 gpm
root         94  0.0  0.2  1396  280 ?        S    Mar02   0:00 /usr/sbin/auto
chris       106  0.0  0.5  1820  680 tty1     S    Mar02   0:08 -bash
root        108  0.0  0.0  1048    0 tty3     SW   Mar02   0:00 [agetty]
root        109  0.0  0.0  1048    0 tty4     SW   Mar02   0:00 [agetty]
root        110  0.0  0.0  1048    0 tty5     SW   Mar02   0:00 [agetty]
root        111  0.0  0.0  1048    0 tty6     SW   Mar02   0:00 [agetty]
[вывод сокращён]
```

Это практически вся информация о системе. В ней присутствует дополнительная информация о процессе: какой пользователь его запустил, сколько он использует системных ресурсов (колонки **%CPU**, **%MEM**, **VSZ** и **RSS**) и когда был запущен. Очевидно, что такая подробная информация может пригодиться системному администратору. Следует отметить ещё один момент: данные теперь вылезают за пределы экрана, и вы не можете увидеть их полностью. Опция `-w` заставит `ps` переносить длинные строки.

Это не слишком удобно, но это работает. Теперь в вашем распоряжении полный отчёт по всем процессам. Существует даже ещё более подробная информация, которую вы можете получить о процессе. Обратитесь к хорошей странице руководства по команде `ps`. Тем не менее представленные выше опции являются самыми популярными, и скорее всего именно их вы будете использовать наиболее часто.

11.4. *kill*

Иногда некоторые программы выходят из под контроля и тогда вам нужно “поставить их на место”. Программа для такого вида администрирования называется `kill`(1). Также её можно использовать для управления процессами разными способами. Самым очевидным способом использования `kill` является убивание процесса (от англ. ***kill*** - убивать). Вам придётся воспользоваться этой программой в том случае, если программа вышла из под контроля и начинает использовать много системных ресурсов или если вас просто тошнит от её работы.

Чтобы убить процесс, вам нужно знать его PID или имя. Чтобы узнать идентификатор, воспользуйтесь командой `ps`, описанной в предыдущем разделе. Например, чтобы убить процесс 4747, вам нужно выполнить следующее:

```
% kill 4747
```

Учтите, чтобы убить процесс, вы должны быть его владельцем. Это особенность системы безопасности. Если бы вам разрешалось убивать процессы других пользователей, вы могли бы осуществлять всевозможные злоумышленные действия. Ну и, естественно, `root` может убить любой процесс в системе.

Существует ещё одна разновидность утилиты `kill` под название `killall`(1). Эта программа полностью соответствует своему названию (*kill all* - убить всех): она убивает все работающие процессы с заданным именем. Если вам нужно убить все процессы `vim`, вы можете набрать следующую команду:

```
% killall vim
```

Все запущенные вам процессы с именем `vim` будут убиты. Выполнив эту команду под `root`'ом, вы убьёте также все процессы `vim`, запущенные пользователями системы. Это также представляет собой интересный способ для выбрасывания из системы всех пользователей (включая самого вас):

```
# killall bash
```

Иногда обычный kill не справляется с поставленной задачей. Определённые процессы не будут умирать. Тогда вам нужно использовать более сильное средство. Если этот упрямый PID 4747 не отвечает на ваш запрос kill, вы можете выполнить следующее:

```
% kill -9 4747
```

Это наверняка заставит процесс 4747 умереть. То же самое вы можете

использовать и с `killall`. В данном случае вы просто отправляете процессу другой сигнал. Обычный вызов `kill` отправляет процессу сигнал `SIGTERM` (terminate), который сообщает ему, что нужно остановить свою работу, сбросить буферы и выгрузить себя из памяти. `kill -9` отправляет процессу сигнал `SIGKILL` (kill), который по сути просто убивает его. Процессу не разрешается "чисто" завершить свою работу, и иногда это приводит к нежелательным последствиям, таким как повреждение данных. Ниже представлен полный список сигналов. Вы можете получить этот список с помощью следующей команды:

```
% kill -l
  1) SIGHUP     2) SIGINT    3) SIGQUIT   4) SIGILL
  5) SIGTRAP    6) SIGABRT   7) SIGBUS    8) SIGFPE
  9) SIGKILL   10) SIGUSR1  11) SIGSEGV  12) SIGUSR2
 13) SIGPIPE   14) SIGALRM  15) SIGTERM  17) SIGCHLD
 18) SIGCONT   19) SIGSTOP  20) SIGTSTP  21) SIGTTIN
 22) SIGTTOU   23) SIGURG   24) SIGXCPU  25) SIGXFSZ
 26) SIGVTALRM 27) SIGPROF  28) SIGWINCH 29) SIGIO
 30) SIGPWR
```

В `kill` нужно использовать только числа, а в `killall` можно использовать названия сигналов без приставки "SIG". Вот ещё один пример:

```
% killall -KILL vim
```

И напоследок, одним из полезных вариантов использования `kill` является перезапуск процесса. Отправка большинству процессов сигнала `SIGHUP` заставит их повторно прочитать свои конфигурационные файлы. Это особенно полезно для сообщения системным процессам о том, что им нужно перечитать свои конфигурационные файлы после того, как вы их отредактировали.

11.5. *top*

И в завершение. Существует программа, которая позволяет вам

просматривать постоянно обновляющуюся информацию о процессах, выполняющихся в системе. Эта команда называется `top`(1) и запускается она так:

```
% top
```

При этом на экран будет выведена информация о выполняющихся в системе процессах плюс некоторая дополнительная информация о системе: средняя загрузка, количество процессов, состояние процессора, информация о свободной памяти. Также предоставляется подробная информация о процессах, включая PID, пользователя, приоритет, использование процессора и памяти, время работы и название программы.

```
 6:47pm  up 1 day,  18:01,  1 user,  load average: 0.02, 0.07, 0.02
61 processes: 59 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  2.8% user,  3.1% system,  0.0% nice, 93.9% idle
Mem:   257992K av,  249672K used,    8320K free,   51628K shrd,  78248K buff
Swap:   32764K av,     136K used,   32628K free,                 82600K cached

  PID USER     PRI  NI  SIZE  RSS  SHARE  STAT  LIB %CPU %MEM  TIME COMMAND
  112 root      12   0 19376  18M   2468  R       0  3.7  7.5 55:53 X
 4947 david     15   0  2136 2136   1748  S       0  2.3  0.8  0:00 screenshot
 3398 david      7   0 20544  20M   3000  S       0  1.5  7.9  0:14 gimp
 4946 root      12   0  1040 1040    836  R       0  1.5  0.4  0:00 top
  121 david      4   0   796  796    644  S       0  1.1  0.3 25:37 wmSMPmon
  115 david      3   0  2180 2180   1452  S       0  0.3  0.8  1:35 wmaker
 4948 david     16   0   776  776    648  S       0  0.3  0.3  0:00 xwd
    1 root       1   0   176  176    148  S       0  0.1  0.0  0:13 init
  189 david      1   0  6256 6156   4352  S       0  0.1  2.4  3:16 licq
 4734 david      0   0  1164 1164    916  S       0  0.1  0.4  0:00 rxvt
    2 root       0   0     0    0      0  SW      0  0.0  0.0  0:08 kflushd
    3 root       0   0     0    0      0  SW      0  0.0  0.0  0:06 kupdate
    4 root       0   0     0    0      0  SW      0  0.0  0.0  0:00 kpiod
    5 root       0   0     0    0      0  SW      0  0.0  0.0  0:04 kswapd
   31 root       0   0   340  340    248  S       0  0.0  0.1  0:00 kerneld
   51 root       0   0    48   48     32  S       0  0.0  0.0  0:00 dhcpcd
   53 bin        0   0   316  316    236  S       0  0.0  0.1  0:00 rpc.portmap
   57 root       0   0   588  588    488  S       0  0.0  0.2  0:01 syslogd
```

Она называется `top` потому, что программы, наиболее активно использующие процессор, будут находиться вверху (*top* - верх). Интересное

замечание: сам **top** будет первым в списке в большинстве систем с низкой активностью вследствие своего использования процессора. Однако `top` весьма удобен для выявления сбойных программ и их убивания.

Однако, допустим, что вам нужно выводить список только своих процессов или процессов какого-то другого пользователя. Процессы, которые вы хотите увидеть, могут отсутствовать среди тех, что активно используют процессор. Опция `-u` позволит вам указать имя пользователя или его **UID** и наблюдать только процессами, владельцем которых является этот **UID**.

```
% top -u alan
  PID USER      PR  NI  VIRT RES   SHR S %CPU %MEM   TIME+   COMMAND
 3622 alan      13   0 11012  10m 6956 S  1.0  2.1   0:03.66 gnome-terminal
 3739 alan      13   0  1012 1012  804 R  0.3  0.2   0:00.06 top
 3518 alan       9   0  1312 1312 1032 S  0.0  0.3   0:00.09 bash
 3529 alan       9   0   984  984  848 S  0.0  0.2   0:00.00 startx
 3544 alan       9   0   640  640  568 S  0.0  0.1   0:00.00 xinit
 3548 alan       9   0  8324 8320 6044 S  0.0  1.6   0:00.30 gnome-session
 3551 alan       9   0  7084 7084 1968 S  0.0  1.4   0:00.50 gconfd-2
 3553 alan       9   0  2232 2232  380 S  0.0  0.4   0:00.05 esd
 3555 alan       9   0  2552 2552 1948 S  0.0  0.5   0:00.10 bonobo-activati
 3557 alan       9   0  2740 2740 2224 S  0.0  0.5   0:00.05 gnome-smproxy
 3559 alan       9   0  6496 6492 5004 S  0.0  1.3   0:00.31 gnome-settings-
 3565 alan       9   0  1740 1740 1440 S  0.0  0.3   0:00.28 xscreensaver
 3568 alan       9   0  7052 7052 4960 S  0.0  1.4   0:02.28 metacity
 3572 alan       9   0 11412  11m 7992 S  0.0  2.2   0:01.58 gnome-panel
 3574 alan       9   0 12148  11m 8780 S  0.0  2.4   0:00.64 nautilus
 3575 alan       9   0 12148  11m 8780 S  0.0  2.4   0:00.00 nautilus
 3576 alan       9   0 12148  11m 8780 S  0.0  2.4   0:00.00 nautilus
```

Как видите, в настоящее время у меня запущены `X`, `top`, `gnome-terminal` (в котором я пишу эти строки) и много других процессов, имеющих отношение к X'ам, и которые используют основную часть ресурсов процессора. Это хороший способ для наблюдения за тем, насколько активно пользователи работают в вашей системе.

`top` также может наблюдать за процессами по их **PID**, игнорируя при этом простаивающие процессы и процессы-зомби. Лучшим источником информации о возможностях этой утилиты является страница руководства по `top`.

Глава 12.

Основы системного администрирования

Whoa whoa whoa whoa whoa... Я знаю о чём вы думаете. “Я не системный администратор! Я даже не хочу быть системным администратором!”

Факт заключается в том, что вы являетесь администратором любого компьютера, для которого у вас есть пароль `root`’а. Это может быть ваша настольная рабочая станция с одним или двумя пользователями или же это может быть большой сервер с несколькими сотнями пользователей. В любом случае вам необходимо знать, как управлять пользователями и как безопасно завершить работу системы системы. Эти задачи выглядят простыми, однако касательно их есть несколько моментов, о которых нужно помнить постоянно.

12.1. Пользователи и группы

Как упоминалось в Гл. 8, вам обычно не следует входить в систему под `root`’ом. Вместо этого вам следует создать учётную запись обычного пользователя для повседневного использования и использовать учётную запись root’а только для задач по обслуживанию системы. Для создания пользователя вы можете использовать либо утилиты, поставляемые со Slackware, либо отредактировать файл паролей вручную.

Вспомогательные скрипты

Самым простым способом для управления пользователями и группами является использование вспомогательных скриптов и программ. В состав Slackware входят следующие программы для работы с пользователями: `adduser`, `userdel`(8), `chfn`(1), `chsh`(1) и `passwd`(1). Команды `groupadd`(8), `groupdel`(8) и `groupmod`(8) предназначены для работы с группами. Эти программы, за исключением `chfn`, `chsh` и `passwd`, запускаются в основном только `root`'ом, и поэтому находятся они в `/usr/sbin`. `chfn`, `chsh` и `passwd` могут выполняться кем-угодно и находятся они в `/usr/bin`.

Пользователей можно добавлять при помощи утилиты `adduser`. Мы начнём рассмотрение этого вопроса с полной процедуры, показывая все задаваемые вопросы и давая краткие пояснения к ним. Ответ по умолчанию заключён в квадратные скобки, и его можно использовать почти во всех диалогах, если вы только не захотите изменить что-либо. (Для удобства под диалогами будет представлен их перевод. На самом деле их там нет, однако они вполне могут появиться в DeepStyle Linux, если Хоттаб поднапряжётся конечно :) - прим. переводчика.)

```
# adduser
Login name for new user []: jellyd
Логин для нового пользователя []: jellyd
```

Это означает имя, которое пользователь будет использовать для входа в систему. Традиционно сложилось так, что логины состоят из восьми или менее символов в нижнем регистре. (Вы можете использовать использовать и большее количество символов, включая цифры, однако старайтесь не делать этого, если у вас нет на то серьёзной причины.)

Вы также можете указать логин в качестве аргумента сразу в командной строке:

```
# adduser jellyd
```

В любом случае после ввода логина adduser попросит вас ввести

идентификатор пользователя:

```
User ID ('UID') [ defaults to next available ]:
ID пользователя ('UID') [ доступно значение по умолчанию ]:
```

Идентификатор пользователя (user ID, UID) - это то, по чём на самом деле определяются права владения в Linux. Каждый пользователь имеет уникальный номер, начиная с 1000 (в Slackware). Вы можете сами указать UID для нового пользователя или же позволить программе самой назначить его из списка доступных.

```
Initial group [users]:
Исходная группа [users]:
```

По умолчанию все пользователи заносятся в группу `users`. Вы можете занести нового пользователя в другую группу, однако это не рекомендуется, если только вы не отдаёте себе отчёт в том, что делаете.

```
Additional groups (comma separated) []:
Дополнительные группы (через запятую) []:
```

Этот вопрос позволяет занести нового пользователя в дополнительные группы. Пользователь может быть членом нескольких групп одновременно. Это полезно в том случае, если вы определили отдельные группы для таких целей, как изменение веб-файлов, запуск игр, запись на оптические носители, прослушивание музыки и т.п. Например, в некоторых системах команду `su` могут использовать только члены группы `wheel`. В Slackware по умолчанию есть группа `sys`, членам которой разрешается воспроизводить звук через внутреннюю звуковую карту.

```
Home directory [/home/jellyd]
Домашний каталог [/home/jellyd]
```

По умолчанию домашние каталоги находятся в `/home`. Если у вас слишком большая система, вы можете перенести домашние каталоги в другое место

(или даже в несколько других мест). Этот этап позволяет вам определить местонахождение домашнего каталога пользователя.

```
Shell [ /bin/bash ]
Командный процессор [ /bin/bash ]
```

В Slackware Linux командным процессором по умолчанию является bash, и его будет достаточно для большинства пользователей. Если ваш новый пользователь пришёл из мира Unix, он может быть уже хорошо знаком с различными шеллами. На этом этапе вы можете изменить шелл пользователя или же он позже может сам изменить его с помощью команды chsh.

```
Expiry date (YYYY-MM-DD) []:
Дата окончания действия (YYYY-MM-DD) []:
```

Для учётных записей можно установить дату истечения их действия. По умолчанию эта дата отсутствует. По желанию вы можете изменить это. Этот параметр может оказаться полезным для администраторов провайдеров услуг Интернета, которые могут пожелать, чтобы учётная запись была заблокирована по истечении некоторого срока, если не была получена оплата за следующий период.

```
New account will be created as follows:
---------------------------------------
Login name:          jellyd
UID:                 [ Next available ]
Initial group:       users
Additional groups:   [ None ]
Home directory:      /home/jellyd
Shell:               /bin/bash
Expiry date:         [ Never ]

Будет создана учётная запись со следующими данными:
---------------------------------------
Логин:                     jellyd
UID:                        [ следующий доступный ]
Исходная группа:          users
```

```
Дополнительные группы:    [ нет ]
Домашний каталог:          /home/jellyd
Командный процессор:      /bin/bash
Дата окончания действия: [ никогда ]
```

Перед вами представлена вся информация, которую вы ввели для новой учётной записи, и у вас есть возможность прервать создание этой учётной записи. Если вы где-то ошиблись, нажмите **Control+C** и начните всё сначала. В противном случае, если всё в порядке, нажмите **Enter**, и учётная запись будет создана.

```
Creating new account...

Changing the user information for jellyd
Enter the new value, or press return for the default
        Full Name []: Jeremy
        Room Number []: Smith 130
        Work Phone []:
        Home Phone []:
        Other []:

Создание новой учётной записи...

Изменение информации о пользователе jellyd
Введите новое значение или нажмите Enter
        Полное имя []: Jeremy Smith
        Офис []: 130
        Рабочий телефон []:
        Домашний телефон []:
        Другая информация []:
```

Вся эта информация является необязательной. Если не хотите, можете не указывать её. Пользователь сам может изменить её в любое время с помощью команды `chfn`. Однако может оказаться полезным указать по крайней мере полное имя и номер телефона, чтобы вы позже могли связаться с этим человеком.

```
Changing password for jellyd
```

```
Enter the new password (minimum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
Re-enter new password:
Password changed.

Account setup complete.
```

```
Изменение пароля пользователя jellyd
Введите новый пароль (минимум - 5, максимум - 127 символов)
Пожалуйста, используйте комбинацию букв верхнего и нижнего регистров и цифры.
Новый пароль:
Новый пароль (повтор):
Пароль изменён.

Настройка учётной записи завершена.
```

Вы должны будете ввести пароль для нового пользователя. Обычно, если этого пользователя в данный момент нет рядом с вами, вам просто нужно установить какой-то простой пароль, а затем сказать пользователю, чтобы он изменил его на что-то более безопасное.

> **Замечание:** *Выбор пароля*: выбор безопасного пароля является первой линией обороны против атак на взлом системы. Вам не следует использовать простые для угадывания пароли, потому что это облегчает задачу несанкционированного проникновения в вашу систему. В идеале пароль должен представлять собой строку из случайного набора символов, включая буквы верхнего и нижнего регистров, цифры и специальные символы. (Символ табуляции не следует использовать, поскольку у вас могу возникнуть проблемы со входом в систему с разного типа компьютеров.) Также не следует пытаться использовать кириллицу, поскольку, даже если вам удастся добиться этого (что, в принципе, возможно), результат может оказаться просто непредсказуемым (прим. переводчика). Есть много программ, генерирующих за вас случайные пароли; поищите их в Интернете.
>
> В общем случае просто придерживайтесь общепринятых рекомендаций: не выбирайте в качестве пароля дату чьего-либо дня рождения, общие фразы, что-то, что можно найти на вашем столе, или что-то такое, что можно легко ассоциировать

с вами. Пароль наподобие “secure1” или любой другой пароль, который вы увидели в печатном издании или на веб-странице, также будет плохим выбором.

Удалять пользователей довольно легко. Просто запустите `userdel` с именем удаляемой учётной записи в качестве аргумента. Вам следует сначала убедиться, что пользователь вышел из системы, и отсутствуют процессы, выполняемые от его имени. Также помните, что после того, как вы удалите пользователя, вся информация о его пароле также исчезнет.

```
# userdel jellyd
```

Эта команда удалит из вашей системы упомянутого пользователя `jellyd`. Неплохая чистка :) Пользователь был удалён из файлов `/etc/passwd`, `/etc/shadow` и `/etc/group`, однако не был удалён его домашний каталог.

Если вы хотите удалить и его каталог, вам нужно было использовать команду со следующей опцией:

```
# userdel -r jellyd
```

Временное отключение учётных записей будет рассмотрено в следующем разделе, посвящённом паролям, поскольку временные изменения влекут за собой изменение пароля пользователя. Изменение другой информации, касающейся учётной записи, рассматривается в Разд. 12.1.3.

Программы для добавления и удаления групп очень просты. `groupadd` добавляет в файл `/etc/group` ещё одну запись с уникальным идентификатором группы, а `groupdel` удаляет указанную группу. Вам остаётся только отредактировать `/etc/group`, чтобы добавить пользователей в нужную группу. Например, чтобы добавить группу с именем `cvs`:

```
# groupadd cvs
```

А чтобы удалить её:

```
# groupdel cvs
```

Изменение паролей

Программа `passwd` изменяет пароли путям внесения изменений в файл `/etc/shadow`. В этом файле хранятся в зашифрованном виде все пароли системы. Чтобы изменить свой собственный пароль, наберите следующее:

```
% passwd
Changing password for chris
Old password:
Enter the new password (minumum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and numbers.
New password:

Изменение пароля пользователя chris
Старый пароль:
Введите новый пароль (минимум - 5, максимум - 127 символов)
Пожалуйста, используйте комбинацию букв верхнего и нижнего регистров и цифры.
Новый пароль:
```

Как видите, вам нужно ввести свой старый пароль. Он не будет показан на экране, когда вы будете набирать его, точно так же, как и при входе в систему. Затем вам надо будет ввести новый пароль. `passwd` выполнит ряд проверок вашего нового пароля, и если он не будет одобрен, вы увидите предупреждение. Если хотите, вы можете проигнорировать эти предупреждения. Вам надо будет ещё раз ввести новый пароль для подтверждения.

В случае, если у вас есть права `root`'а, вы можете изменить пароль другого пользователя:

```
# passwd ted
```

Затем вам нужно будет пройти процедуры, описанные выше, за тем исключением, что вам не надо будет вводить старый пароль пользователя.

(Одно из многих преимуществ `root`'а...)

При необходимости вы можете также временно отключить (заблокировать) учётную запись и позже повторно включить (разблокировать) её. И блокирование, и разблокирование учётной записи можно выполнить при помощи утилиты `passwd`. Чтобы отключить учётную запись, выполните под `root`'ом следующее:

```
# passwd -l david
```

При этом пароль пользователя david будет изменён на нечто, что никогда не сможет совпасть с какой-либо зашифрованной комбинацией символов. Разблокировать учётную запись можно с помощью следующей команды:

```
# passwd -u david
```

Теперь учётная запись david'а снова возвращена в нормальное состояние. Отключение учётной записи может быть полезным в том случае, если пользователь не соблюдает правила, которые вы установили для своей системы, или если вы обнаружили слишком много экспортированных копий `xeyes`(1) на своём рабочем столе.

Изменение информации о пользователях

Существует два типа личной информации, которую пользователи могут изменить в любое время: это командный процессор и информация для finger. В Slackware Linux для изменения этих данных используются утилиты `chsh` (change shell) и `chfn` (change finger).

Пользователь может выбрать себе любой шелл из тех, что перечислены в файле `/etc/shells`. Большинство пользователей вполне устроит `/bin/bash`. Другие могут быть хорошо знакомы с другим шеллом, который используется в их системе на работе или в школе, и поэтому наверняка захотят пользоваться тем, что уже хорошо знают. Чтобы изменить свой командный процессор, воспользуйтесь командой `chsh`:

```
% chsh
Password:
Changing the login shell for chris
Enter the new value, or press return for the default
        Login Shell [/bin/bash]:

Пароль:
Изменение командного процессора пользователя chris
Введите новое значение или нажмите Enter
        Командный процессор [/bin/bash]:
```

После ввода своего пароля введите полный путь к новому командному процессору. Сначала убедитесь в том, что он перечислен в файле `/etc/shells`(5). Также `root` может изменить шелл любого пользователя, запустив `chsh` и именем этого пользователя в качестве аргумента.

Информация finger содержит опциональные данные о вас: полное имя, номера телефонов и номер офиса/кабинета. Эти данные могут быть изменены с помощью команды `chfn`, а процесс представляет собой ту же самую процедуру, что и при создании учётной записи. Как обычно `root` может изменить finger-информацию любого пользователя.

12.2. Пользователи и группы: сложный путь

Конечно же пользователей и группы можно добавлять, изменять и удалять без использования скриптов и программ, поставляемых со Slackware. Это не так уж и сложно, хотя после прочтения о том, как это делается, вы скорее всего решите, что использовать скрипты всё же проще. Однако важно знать о том, как на самом деле хранится информация с вашим паролем на тот случай, если вам надо будет восстановить эту информацию, а утилит Slackware под рукой не окажется.

Сначала нужно добавить нового пользователя в файлы `/etc/passwd`(5), `/etc/shadow`(5) и `/etc/group`(5). В файле `passwd` хранится информация о

пользователях вашей системы, но (как ни странно) без их паролей. Когда-то пароли в нём хранились, однако довольно давно это было отменено по соображениям безопасности. Файл **passwd** должен быть доступен на чтение всем пользователям системы. Однако зашифрованные пароли при этом не должны быть доступны на чтение всем, поскольку, имея их, злоумышленники путём перебора могут взломать пароли пользователей. Поэтому зашифрованные пароли хранятся в файле **shadow**, который доступен на чтение только **root**'у, а все пароли пользователей, указываемые в файле `passwd` - это просто "`x`". В файле `group` перечислены все группы и их члены.

Вы можете использовать утилиту `vipw` для безопасного редактирования файла `/etc/passwd` и утилиту `vigr` для безопасного редактирования `/etc/group`. Для безопасного редактирования файла `/etc/shadow` используйте `vipw -s`. (Здесь под "безопасностью" подразумевается, что никто другой не сможет изменить файл, пока вы редактируете его. Если вы являетесь единственным администратором своей системы, вам не стоит об этом беспокоиться, однако лучше с самого начала научиться придерживаться такого поведения.)

Давайте изучим содержимое файла `/etc/passwd` и узнаем, как добавить нового пользователя. Типичная запись в `passwd` выглядит примерно так:

```
chris:x:1000:100:Chris Lumens,Room 2„:/home/chris:/bin/bash
```

На каждого пользователя приходится одна строка, а поля разделяются запятыми. Поля: логин, зашифрованный пароль (в **Slackware** это "`x`" для любого пользователя, поскольку в системе используются теневые пароли), идентификатор пользователя, идентификатор группы, опциональные данные **finger** (через запятую), домашний каталог и командный процессор. Чтобы добавить нового пользователя вручную, добавьте в конце файла новую строку, заполнив её соответствующей информацией.

Добавляемая вами информация должна удовлетворять определённым требованиям, в противном случае у вашего нового пользователя будут

проблемы со входом в систему. Сначала убедитесь, что в поле с паролем стоит `x`, и что и имя пользователя, и его ID являются уникальными. Назначьте пользователю группу: 100 (группа "users" в Slackware) или свою группу по умолчанию (используйте её номер, а не название). Назначьте пользователю домашний каталог (который вы создадите чуть позже) и командный процессор (помните, что действительные шеллы перечислены в файле `/etc/shells`).

Затем необходимо добавить запись в файл /etc/shadow, в котором хранятся зашифрованные пароли. Обычно запись в этом файле выглядит так:

```
chris:$1$w9bsw/N9$uwLr2bRER6YyBS.CAEp7R.:11055:0:99999:7:::
```

И опять же каждому пользователю соответствует одна строка, поля в которой разделены запятыми. Поля (по порядку) - это логин; зашифрованный пароль; количество дней от начала эры UNIX (1 января 1970 года), когда был изменён пароль; количество дней, по истечении которых пароль должен быть изменён; количество дней до истечения срока действия пароля, когда пользователь начнёт получать уведомления; количество дней после истечения срока действия, когда учётная запись будет заблокирована; и зарезервированное поле.

Как видите, основная часть информации посвящена истечению срока действия учётной записи. Если вы не будете использовать эти данные, вам нужно только заполнить несколько полей специальными значениями. В противном случае вам придётся произвести ряд вычислений и принять определённое решение перед тем, как заполнить эти поля. Для нового пользователя поле пароля заполните каким-нибудь мусором. На данный момент вам не нужно беспокоиться о пароле, потому что вы измените его через минуту. Единственным символом, который нельзя использовать в поле пароля, является двоеточие. Также оставьте пустым поле " количество дней от начала эры UNIX, когда был изменён пароль". Введите `0`, `99999` и `7` точно так же, как в представленном выше примере, а все остальные поля оставьте пустыми.

(Для тех, кто думает, что выше показан мой зашифрованный пароль, и верит, что он может вломиться в мою систему, вперёд и с песней. Если вы сможете взломать этот пароль, вы получите пароль к тестовой системе, находящейся за файерволом. Это настолько полезно... :))

В стандартной системе Slackware все обычные пользователи являются членами группы "`users`". Однако при желании вы можете создать новую группу или добавить нового пользователя в дополнительные группы. Для этого вам понадобится отредактировать файл `/etc/group`. Вот типичная запись из него:

```
cvs::102:chris,logan,david,root
```

Поля: название группы, пароль группы, идентификатор группы и члены группы, разделённые запятыми. Создание группы - это просто добавление новой строки с уникальным идентификатором группы плюс перечисление всех пользователей, которых вы хотите сделать членами этой группы. Все пользователи, входящие в эту новую группу и уже работающие в системе, должны будут выйти и снова войти в систему, чтобы эти изменения вступили в силу.

На данном этапе хорошей идеей было бы использование команд `pwck` и `grpck` для проверки корректности сделанных вами изменений. Сначала используйте `pwck -r` и `grpck -r`: опция `-r` не вносит изменения, а перечисляет изменения, которые вам было бы предложено внести, если бы команда была запущена без этой опции. Вы можете использовать этот вывод для принятия решения о том, нужно ли вам ещё изменять какие-либо файлы перед тем, как запустить `pwck` или `grpck` без опции `-r`, или просто оставить свои изменения нетронутыми.

Теперь вам следует воспользоваться утилитой `passwd` для создания рабочего пароля для пользователя. Затем используйте команду `mkdir` для создания домашнего каталога пользователя в месте, указанном вами в файле `/etc/passwd`. И, наконец, используйте команду `chown` для изменения владельца только что созданного каталога для нового пользователя.

Под удалением пользователя подразумевается удаление всех существующих записей для этого пользователя. Удалите запись пользователя из файлов `/etc/passwd` и `/etc/shadow`, а также удалите его логин изо всех групп в файле `/etc/group`. При желании вы можете удалить домашний каталог пользователя, файл почтового спула и его запись в crontab (если таковые существуют).

Удаление групп выполняется аналогично: удалите запись группы в файле `/etc/group`.

12.3. Корректное завершение работы

Очень важно корректно завершать работу своей системы. Простое отключение энергии с помощью кнопки питания может привести к серьёзному повреждению файловой системы. Пока система работает, файлы используются даже в том случае, если вы ничего не делаете. Помните, что в фоновом режиме всё время работает много процессов. Эти процессы обслуживают систему и держат открытими много файлов. Когда на системном блоке отключается питание, эти файлы не закрываются корректным образом и могут быть повреждены. В зависимости от того, какие файлы были повреждены, система может оказаться полностью в нерабочем состоянии! В любом случае при последующей загрузке системы вам придётся пройти через длительную процедуру проверки файловой системы.

> **Замечание:** Если вы используете в своей системе журналируемые файловые системы наподобие ext3 или reiserfs, вы частично защищены от повреждения файловой системы, и проверка вашей файловой системы при перезагрузке займёт меньше времени, чем при проверке файловой системы без журналирования как в случае с ext2. Однако этот вариант защиты не является оправданием для некорректного выключения своей системы! Журналируемые ФС подразумевают защиту ваших файлов от событий, на которые вы не силах повлиять, а не от вашей личной лени.

В любом случае, когда вы перезагружаете или выключаете свой компьютер, важно сделать это правильно. Для этого есть несколько способов и вы можете выбрать любой из них на свой вкус. Поскольку выключение и перезагрузка - это похожие процедуры, большинство способов выключения системы также применимы и к её перезагрузке.

Первый способ - с помощью программы `shutdown`(8), и наверное это самый популярный способ. Утилиту `shutdown` можно использовать для перезагрузки или выключения системы в заданное время, и она может выводить всем работающим в системе пользователям сообщение о том, что система сейчас будет выключена.

Наиболее общее использование shutdown для выключения компьютера:

```
# shutdown -h now
```

В данном случае особое сообщение пользователям не отправляется. Они увидят стандартное сообщение `shutdown`. "`now`" - это время, когда мы хотим выполнить завершение работы, а "`-h`" означает выключение системы. Это выглядит не слишком удобным для многопользовательской системы, однако он прекрасно подойдёт для вашего домашнего компьютера. Более правильным способом в многопользовательской системе будет предоставление всем пользователям некоторого запаса времени на выход из системы:

```
# shutdown -h +60
```

При этом система будет выключена через один час (60 минут), что прекрасно подойдёт для многопользовательской системы. В важных системах выключение должно быть запланировано заблаговременно, и вам следует сообщать о времени её выключения во всех соответствующих источниках информации, используемых для системных уведомлений (электронная почта, доски объявлений, `/etc/motd` и др.).

Для перезагрузки системы используется та же самая команда, только опция "`-h`" заменяется на "`-r`":

```
# shutdown -r now
```

Вы можете использовать такую же запись времени для `shutdown -r`, что и для `shutdown -h`. Есть ещё много других вещей, которые вы можете выполнить с помощью `shutdown` для управления выключением или перезагрузкой машины; подробности смотрите в странице руководства.

Второй способ выключения или завершения работы компьютера - это использование команд `halt`(8) и `reboot`(8). Как видно по названиям, `halt` немедленно останавливает операционную систему, а `reboot` перезагружает её (по сути `reboot` - это просто символическая ссылка на `halt`.) Вызываются они так:

```
# halt
# reboot
```

Низкоуровневый способ выключения или перезагрузки системы заключается в отправке сообщения непосредственно процессу `init`. Все другие методы представляют собой просто удобные способы общения с `init`, однако вы можете непосредственно сообщить, что нужно выполнить, с помощью `telinit`(8) (обратите внимание, что в названии используется только одна буква "l" (англ. *tell* - говорить)). С помощью `telinit` вы можете сообщить `init`'у, на какой уровень запуска нужно выполнить переход. При этом будет запущен специальный скрипт. Этот скрипт убьёт или породит процессы, нужные на этом уровне запуска. Это подойдёт для перезагрузки и выключения, потому что для обоих этих процессов есть свои специальные уровни запуска.

```
# telinit 0
```

Уровень запуска 0 - это останов системы. Сообщение `init`'у о переходе на уровень запуска 0, приведёт к тому, что будут убиты все процессы,

файловые системы будут размонтированы и машина будет выключена. Это подходящий способ для выключения системы. На многих ноутбуках и современных настольных компьютерах это также приведёт к завершению работы системы.

```
# telinit 6
```

Уровень запуска 6 - это перезагрузка системы. Все процессы будут убиты, файловые системы будут размонтированы, а машина отправится на перезагрузку. Это также подходящий способ для перезагрузки системы.

Для любознательных, при переключении на уровень запуска 0 или 6, либо с помощью `shutdown`, `halt` или `reboot`, запускается скрипт `/etc/rc.d/rc.6`. (Скрипт `/etc/rc.d/rc.0` - это символическая ссылка на `/etc/rc.d/rc.6`.) Вы можете изменить этот файл на своё усмотрение, однако убедитесь в том, что хорошенько протестировали свои изменения!

Существует ещё один последний способ перезагрузки системы. Для всех остальных способов необходимо, чтобы вы имели права `root`'а. Однако перезагрузить систему можно даже если без его привелегий, но при условии, что у вас есть физический доступ к клавиатуре. Нажатие **Control+Alt+Delete** (заветные "три педали") вызовет немедленную перезагрузку машины. (На самом деле, когда вы нажимаете **Control+Alt+Delete**, вызывается команда `shutdown`.) "Педали" не всегда срабатывают при работе в X Windows. Вам может понадобиться нажать **Control+Alt+F1** (или с другой функциональной клавишей), чтобы перейти в чистую консоль перед тем, как воспользоваться заветной комбинацией.

И в завершение, файл, который полностью контролирует все аспекты загрузки и выключения, - это `/etc/inittab`(5). В общем случае вам не нужно изменять этот файл, однако он может дать вам представление о том, почему некоторые вещи работают так, как они работают. Как всегда подробности смотрите на странице руководства.

Глава 13.

Основные сетевые команды

Сеть представляет собой несколько компьютеров, соединённых между собой. Сеть может быть настолько простой, как несколько соединённых компьютеров у вас дома или в офисе, или настолько сложной, как большая университетская сеть или даже весь Интернет. Когда ваш компьютер является частью сети, вы можете получить доступ к другим системам напрямую или через сетевые службы наподобие почтового или веб-сервера.

Существует большое разнообразие сетевых программ. Некоторые удобны для диагностики работы сети. Другие (наподобие почтовых клиентов и веб-браузеров) полезны для повседневной работы и общения с другими людьми.

13.1. *ping*

Утилита `ping`(8) отправляет указанному хосту ICMP-пакеты `ECHO_REQUEST`. Если хост ответит, вы получите ICMP-пакет. Звучит странно? Хорошо. Вы можете "пропинговать" IP-адрес, чтобы определить, доступна машина или нет. Если ответа нет, значит что-то не в порядке. Вот пример диалога двух пользователей Linux:

damned: kastor, попингуй!
kastor: Слыш, сам попингуй!
damned: От попингуя слышу!

Это пример того, насколько `ping` полезен для повседневного использования. Он предоставляет очень быстрый способ проверки, включена ли и подключена ли машина к сети. Синтаксис довольно прост:

```
% ping www.slackware.com
```

Естественно, для этой команды есть несколько опций. Дополнительную информацию смотрите на странице руководства `ping`(1).

13.2. *traceroute*

Команда `traceroute`(8) очень полезна для диагностики сети. `traceroute` показывает все хосты, через которые проходит пакет по пути к конечному назначению. С помощью этой команды вы можете увидеть количество “хопов” (hops) между вами и сайтом Slackware:

```
% traceroute www.slackware.com
```

Будут показаны все хосты с временем отклика каждого из них. Вот пример вывода команды:

```
% traceroute www.slackware.com
traceroute to www.slackware.com (204.216.27.13), 30 hops max, 40 byte packets
1  zuul.tdn (192.168.1.1)  0.409 ms  1.032 ms  0.303 ms
2  207.171.227.254 (207.171.227.254)  18.218 ms  32.873 ms  32.433 ms
3  border-sf-2-0-4.sirius.com (205.134.230.254) 15.662 ms 15.731 ms 16.142 ms
4  pb-nap.crl.net (198.32.128.20)  20.741 ms  23.672 ms  21.378 ms
5  E0-CRL-SFO-03-E0X0.US.CRL.NET (165.113.55.3) 22.293 ms 21.532 ms 21.29 ms
6  T1-CDROM-00-EX.US.CRL.NET (165.113.118.2)  24.544 ms  42.955 ms 58.443 ms
7  www.slackware.com (204.216.27.13)  38.115 ms  53.033 ms  48.328 ms
```

Утилита `traceroute` похожа на `ping` в том, что использует ICMP-пакеты. Для `traceroute` можно указать несколько опций. Они как всегда подробно описаны на странице руководства.

13.3. Утилиты для работы с DNS

Служба доменных имён (Domain Name Service, DNS) - это магический протокол, который позволяет вашему компьютеру преобразовывать имена доменов наподобие www.slackware.com в IP-адреса наподобие 64.57.102.34. Компьютеры не могут определить маршрут пакетов к www.slackware.com, однако они могут определить маршрут к IP-адресу этого домена. Так обеспечивается удобный способ запоминания машин. Без DNS пришлось бы содержать нереальную базу данных о том, какой IP-адрес принадлежит какому компьютеру, и это при условии, что IP-адрес не будет изменяться! Очевидно, что лучше использовать для компьютеров имена, однако как связать эти имена с IP-адресами?

host

Это может выполнить утилита `host`(1). `host` используется для сопоставления имён с IP-адресами. Это очень быстрая и простая утилита с небольшим количеством функций.

```
% host www.slackware.com
www.slackware.com is an alias for slackware.com.
slackware.com has address 64.57.102.34
```

Однако давайте представим, что нам нужно определить домен по IP-адресу. Что тогда?

nslookup

`nslookup` была и остаётся программой, выжившей с незапамятных времён. `nslookup` уже давно устарела и может быть убрана в следующих релизах. Для неё даже нет страницы руководства.

```
% nslookup 64.57.102.34
Note:  nslookup is deprecated and may be removed from future releases.
```

```
Consider using the 'dig' or 'host' programs instead.  Run nslookup with
the '-sil[ent]' option to prevent this message from appearing.
Server:         192.168.1.254
Address:        192.168.1.254#53

Non-authoritative answer:
www.slackware.com       canonical name = slackware.com.
Name:   slackware.com
Address: 64.57.102.34
```

dig

`dig`(1) (domain information groper) - это программа для поиска информации в DNS. `dig` может получить с DNS-сервера практически всё, что угодно, включая реверсивные запросы, записи A, CNAME, MX, SP и TXT. У `dig` есть много опций, и, если вы не слишком знакомы с этой утилитой, вам следует сначала ознакомиться с её страницей руководства.

```
% dig @192.168.1.254 www.slackware.com mx

; <<>> DiG 9.2.2 <<>> @192.168.1.254 www.slackware.com mx
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26362
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.slackware.com.             IN      MX

;; ANSWER SECTION:
www.slackware.com.      76634   IN      CNAME   slackware.com.
slackware.com.          86400   IN      MX      1 mail.slackware.com.

;; AUTHORITY SECTION:
slackware.com.          86400   IN      NS      ns1.cwo.com.
slackware.com.          86400   IN      NS      ns2.cwo.com.

;; ADDITIONAL SECTION:
ns1.cwo.com.            163033  IN      A       64.57.100.2
ns2.cwo.com.            163033  IN      A       64.57.100.3
```

```
;; Query time: 149 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: Sat Nov  6 16:59:31 2004
;; MSG SIZE  rcvd: 159
```

Это должно дать вам представление о том, как работает `dig`. "@192.168.1.254" обозначает используемый dns-сервер. "www.slackware.com" - это домен, по которому выполняется поиск, а "mx" - это тип выполняемого поиска. Запрос выше сообщил нам, что электронная почта для домена `www.slackware.com` будет отправлена для обработки на хост `mail.slackware.com`.

13.4. *finger*

Команда `finger`(1) извлекает информацию об указанном пользователе. Вы даёте finger'у имя пользователя или адрес e-mail, и он попытается связаться с соответствующим сервером, чтобы получить от него имя пользователя, номер офиса, телефон и другую информацию. Пример:

```
% finger johnc@idsoftware.com
```

`finger` может возвратить имя пользователя, состояние почты, телефонные номера и файлы типа "dot plan" и "dot project". Естественно, возвращаемая информация для каждого `finger`-сервера будет своей. В Slackware по умолчанию предоставляются следующие данные:

- Имя пользователя
- Номер офиса
- Номер домашнего телефона
- Номер рабочего телефона
- Состояние логина

- Состояние e-mail
- Содержимое файла `.plan` из домашнего каталога пользователя
- Содержимое файла `.project` из домашнего каталога пользователя

Первые четыре записи можно установить с помощью команды `chfn`. Она сохраняет их значения в файле `/etc/passwd`. Чтобы изменить информацию в своих файлах `.plan` или `.project`, просто отредактируйте их в своём любимом текстовом редакторе. Они должны находиться в вашем домашнем каталоге и называться `.plan` и `.project`.

Многие пользователи "тыкают пальцем" (`finger`) в свои собственные учётные записи с удалённых машин, чтобы узнать, не сменился ли у них электронный почтовый ящик. С другой стороны вы можете увидеть план пользователя или его текущий проект.

Как и у многих других команд у `finger` есть свои опции. Подробную информацию о них вы можете узнать на странице руководства.

13.5. *telnet*

Кто-то однажды сказал, что `telnet`(1) была самой крутой вещью в мире компьютеров. Возможность удалённого входа в систему и работа на нём с другого компьютера - вот, что отличало Unix и Unix-подобные операционные системы от других ОС.

`telnet` позволяет вам входить в систему на удалённом компьютере так, словно вы сидите за его терминалом. После того, как будут проверены ваши имя пользователя и пароль, вы получите приглашение командного процессора. С этого момента вы можете делать всё, что угодно, для чего требуется текстовая консоль: писать электронные письма, читать новости, перемещать файлы и т.п. Если вы работаете в X'ах вы подключаетесь к другой машине по `telnet`'у, вы можете запускать X-программы на удалённом компьютере с отображением их на вашем мониторе.

Чтобы войти в систему на удалённой машине, используйте следующий синтаксис:

```
% telnet <имя_хоста>
```

Если хост ответит, вы получите приглашение для входа в систему. Введите свои имя пользователя и пароль. Вот и всё. Теперь в вашем распоряжении командный процессор удалённой системы. Чтобы завершить telnet-сеанс, используйте команду `exit` или `logout`.

> **Внимание**`telnet` не шифрует передаваемую информацию. Всё передаётся чистым текстом, даже пароли. Поэтому не рекомендуется использовать `telnet` при работе через Интернет. Вместо этого используйте `Secure Shell` (безопасный шелл). Он шифрует весь передаваемый трафик и является свободно доступным.

Другое использование **telnet**'а

После того, как мы убедили вас больше не использовать telnet-протокол для входа в систему на удалённой машине, мы покажем вам несколько полезных полезных вариантов использования `telnet`.

Вы можете использовать `telnet` для подключения к определённому порту хоста.

```
% telnet <имя_хоста> [порт]
```

Это может быть довольно удобным в случае, когда вам нужно быстро протестировать определённую службу, имея при этом полный контроль над командами, и получить полный отчёт о том, что происходит. Таким способом вы можете в интерактивном режиме протестировать или использовать сервер SMTP, POP3, HTTP и т.п.

На следующем рисунке вы увидите, как можно использовать `telnet` для подключения к HTTP-серверу на 80-й порт и получения от него некоторой

базовой информации.

Рисунок 13-1. Подключение к веб-серверу по telnet'у

```
% telnet store.slackware.com 80
Trying 69.50.233.153...
Connected to store.slackware.com.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 25 Apr 2005 20:47:01 GMT
Server: Apache/1.3.33 (Unix) mod_ssl/2.8.22 OpenSSL/0.9.7d
Last-Modified: Fri, 18 Apr 2003 10:58:54 GMT
ETag: "193424-c0-3e9fda6e"
Accept-Ranges: bytes
Content-Length: 192
Connection: close
Content-Type: text/html

Connection closed by foreign host.
%
```

То же самое вы можете выполнять и для других протоколов, работающих с чистым текстом, при условии, что вы знаете, к какому порту подключаться и какие команды использовать.

13.6. Безопасный шелл (secure shell)

Безопасный шелл сегодня получает все лавры, которые раньше доставались `telnet`'у. `ssh`(1) позволяет устанавливать соединение с удалённой машиной и выполнять на ней программы так, как если бы вы находились перед её монитором. Однако `ssh` шифрует все данные, передаваемые между двумя компьютерами, так что даже в случае их

перехвата понять их будет невозможно. Ниже представлено типичное подключение по ssh.

```
% ssh carrier.lizella.net -l alan
The authenticity of host 'carrier.lizella.net (192.168.1.253)' can't be
established.
RSA key fingerprint is 0b:e2:5d:43:4c:39:4f:8c:b9:85:db:b2:fa:25:e9:9d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'carrier.lizella.net' (RSA) to the list of
known hosts.
Password: password
Last login: Sat Nov  6 16:32:19 2004 from 192.168.1.102
Linux 2.4.26-smp.
alan@carrier:~$ ls -l MANIFEST
-rw-r--r--  1 alan users 23545276 2004-10-28 20:04 MANIFEST
alan@carrier:~$ exit
logout
Connection to carrier.lizella.net closed.
```

В данном примере устанавливается ssh-соединение с `carrier.lizella.net` и проверяются права доступа к файлу `MANIFEST`.

13.7. Электронная почта (e-mail)

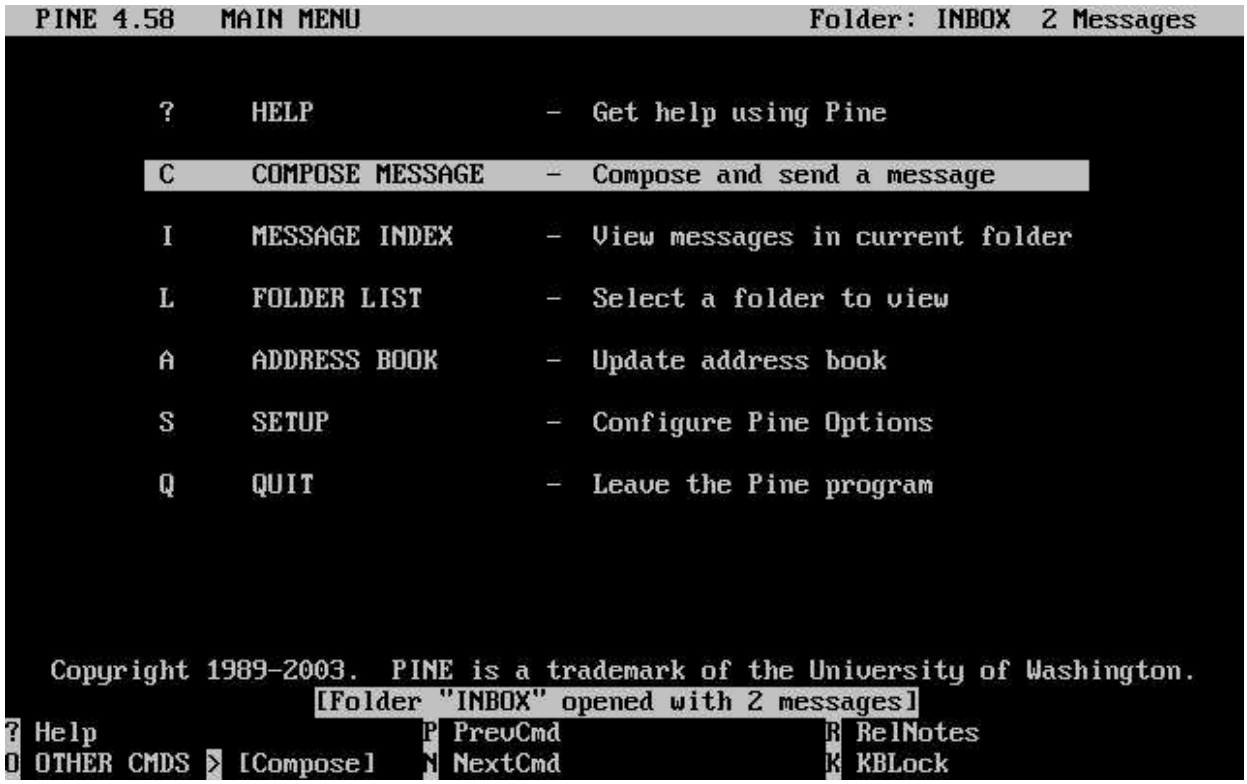
Электронная почта является одной из самых популярных служб в Интернете. В 1998 году было зафиксировано, что электронных писем было отправлено больше, чем обычных. Такая почта, несомненно, является более удобной и практичной.

В состав Slackware входит стандартный почтовый сервер и несколько почтовых клиентов. Все описанные ниже почтовые клиенты являются являются текстовыми. Многие пользователи Windows могут быть против этого, однако вы обнаружите, что консольные клиенты очень удобны, особенно при удалённой проверке почты. Не волнуйтесь, существует много графических почтовых клиентов, например, KDE'шный Kmail.

pine

`pine`(1) is not `elm`. Вашингтонский университет создал для своих студентов собственные программы для чтения электронной почты и новостей из Интернета. `pine` - это один из наиболее популярных на сегодня почтовых клиентов и он доступен практически для любого Unix и даже для Windows.

Рисунок 13-2. Главное меню pine



Вы увидите меню с командами и строку с клавишами команд в нижней части экрана. `pine` является довольно сложной программой, поэтому здесь мы не будем подробно рассматривать все её функции.

Чтобы увидеть содержимое ящика с входящими сообщениями, наберите **`i`**. Будет выведен список ваших писем вместе с датой, автором и темой. Выберите нужное вам сообщение и нажмите **`Enter`**, чтобы просмотреть

его. С помощью клавиши **r** можно начать писать ответ на сообщение. После завершения написания ответного сообщения нажмите **Ctrl+X**, чтобы отправить его. Вы можете нажать **i**, чтобы вернуться назад к списку сообщений.

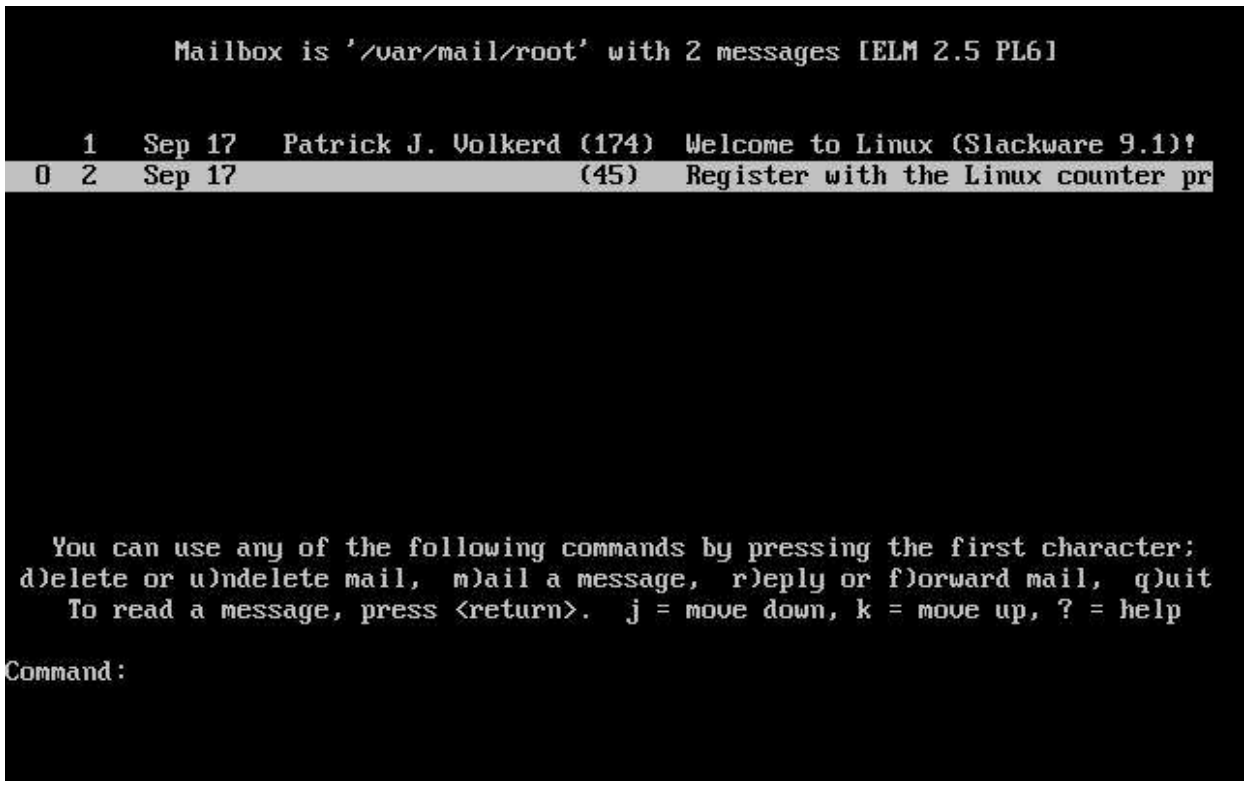
Если вам нужно удалить сообщение, нажмите **d**. Подсвеченное сообщение будет отмечено как подготовленное к удалению. `pine` удаляет письма при выходе из программы. `pine` также позволяет вам хранить сообщения в папках. Вы можете получить список папок с помощью клавиши **l**. Выберите в списке письмо и нажмите **s**, чтобы сохранить его в другой папке. Будет выведен запрос о том, в папку с каким именем нужно выполнить сохранение.

`pine` содержит очень много функций. Вам определённо стоит изучить его страницу руководства на предмет получения более подробной информации. Там будет представлена самая последняя информация о программе.

elm

`elm`(1) - это ещё один популярный текстовый почтовый клиент. Хотя его интерфейс не настолько дружественный, как у `pine`, появился он гораздо раньше.

Рисунок 13-3. Главный экран elm



По умолчанию вы окажетесь в ящике со входящей почтой. Список сообщений содержит номер сообщения, дату, отправителя и тему. Используйте клавиши со стрелками вверх и вниз для перехода к нужному вам сообщению. Нажмите **Enter**, чтобы прочитать письмо.

Чтобы написать новое сообщение, наберите в главном окне **m**. Клавиша **d** пометит письмо флагом для удаления. И клавиша **r** запустит написание ответа на сообщение, которое вы сейчас читаете. Все эти клавиши показаны в нижней части экрана рядом с приглашением.

Страница руководства содержит более подробное описание `elm`, поэтому вам рекомендуется ознакомиться с ней перед тем, как начать использовать `elm`.

mutt

“Все почтовые клиенты - отстой. Однако этот клиент менее отстойный.” Оригинальный интерфейс `mutt` был основан на интерфейсе `elm`’а с добавлением новых возможностей и функций, позаимствованных у других популярных почтовых клиентов. В результате получился mutt.

Некоторые из возможностей `mutt`’а:

- поддержка цветов
- тематическая сортировка писем
- поддержка MIME и PGP/MIME
- поддержка pop3 и imap
- поддержка нескольких форматов почтовых ящиков (mbox, MMDF, MH, maildir)
- *широкая* настраиваемость

Рисунок 13-4. Главный экран mutt

Если вам нужен почтовый клиент, который позволил бы вам контролировать абсолютно всё, тогда скорее всего это `mutt`. Все настройки по умолчанию могут быть настроены под ваши нужды, привязки клавиш также могут быть изменены. При желании вы можете добавить свои макросы.

Вам наверняка понадобится прочитать страницу руководства `muttrc`, которая поможет вам настроить всё, что угодно. Или же вы можете изучить пример файла `muttrc`, входящий в состав пакета.

nail

`nail`(1) - это почтовый клиент, работающий в режиме командной строки. Он очень примитивен и не предлагает ничего похожего на

пользовательский интерфейс. Однако **mailx** довольно удобен в случаях, когда вам нужно быстро отправить что-либо по почте, написать скрипт массовой рассылки, протестировать настройки своего **MTA** или что-то подобное. Обратите внимание, что в **Slackware** на `nail` создаётся символическая ссылка в виде `/usr/bin/mail` и `/usr/bin/mailx`. Любая из этих трёх команд запускает одну и ту же программу. По сути чаще всего вы будете видеть `nail` в виде ссылки `mail`.

Базовая командная строка представляет собой следующее:

```
% mailx <тема> <адрес-получателя>
```

`mailx` читает тело сообщения со стандартного входа. Поэтому вы можете "cat'нуть" файл в эту команду, чтобы отправить его по почте, или можете просто набрать текст и нажать **Ctrl+D** по завершении написания сообщения.

Вот пример отправки по почте исходного текста программы другому человеку.

```
% cat randomfunc.c | mail -s "Here's that function" asdf@example.net
```

Страница руководства содержит более подробную информацию о возможностях `nail`, поэтому вам рекомендуется прочитать её перед тем, как использовать программу.

13.8. Браузеры

Первое, о чём думали люди, когда слышали слово "Интернет" - это "сёрфинг по сети". Или просмотр веб-сайтов с помощью веб-браузеров. Это, наверное, самое распространённое использование Интернета для среднестатистического пользователя.

В **Slackware** популярные графические веб-браузеры представлены в

категории “XAP”, а также консольные браузеры из категории “N”. Ниже мы вкратце рассмотрим некоторые наиболее общие варианты их использования.

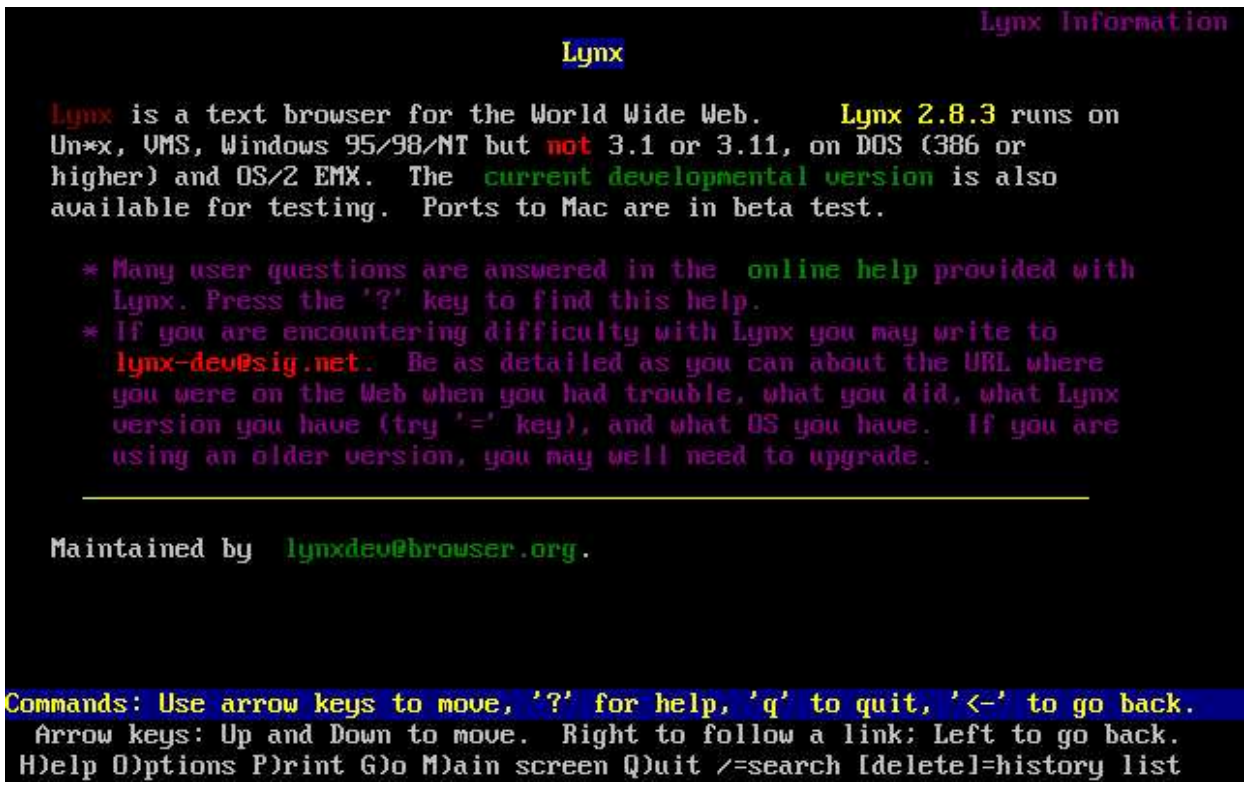
lynx

`lynx`(1) - это консольный веб-браузер. Он представляет собой очень быстрое средство для поиска чего-либо в Интернете. Иногда графика приводит к нужному результату только в том случае, если вы точно знаете, что за ней скрывается.

Чтобы запустить `lynx`, просто наберите в командной строке `lynx`:

```
% lynx
```

Рисунок 13-5. Стартовая страница lynx по умолчанию



Вы можете запустить `lynx` с адресом сайта, который нужно открыть:

```
% lynx http://www.slackware.com
```

`lynx` показывает в нижней части своего экрана командные клавиши и их действие. Клавиши со стрелками вверх и вниз осуществляют перемещение по документу, **Enter** выполняет переход по выбранной ссылке, а клавиша со стрелкой влево возвращает вас на предыдущую загруженную страницу. Нажатие клавиши **d** загрузит выбранный в данный момент файл. Команда **g** вызовет приглашение Go, в котором вы можете ввести URL, который нужно открыть.

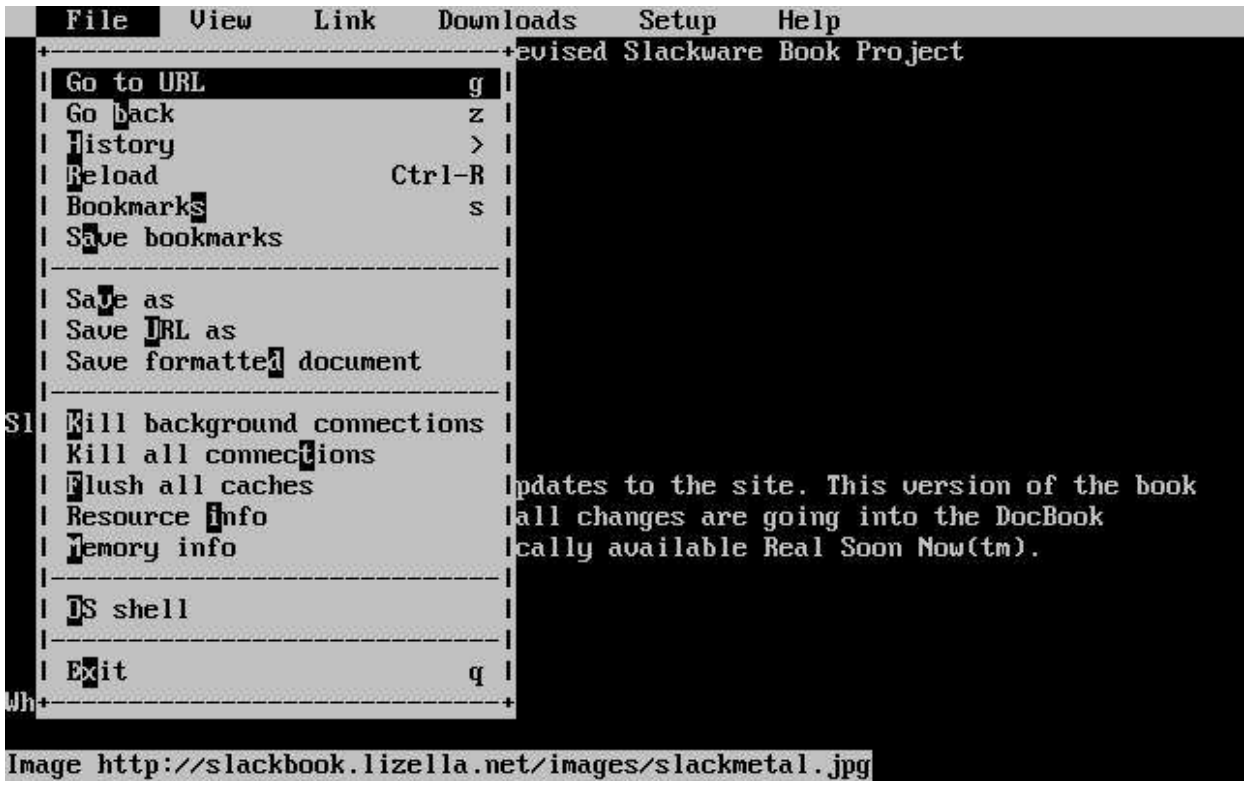
В `lynx` существует ещё много других команд. Для получения дополнительной информации вы можете или обратиться к странице руководства, или набрать **h**, чтобы вызвать экран со справкой.

links

Как и `lynx`, `links` - это консольный браузер, всю навигацию в котором вы можете осуществлять с помощью клавиатуры. Однако, если вы нажмёте клавишу **Esc**, в верхней части его экрана откроется очень удобное выпадающее меню. Это намного упрощает его использование, отбрасывая необходимость изучения всех клавишных команд. Люди, не использующие текстовые браузеры каждый день, по достоинству оценят эту возможность.

`links` обладает улучшенной поддержкой фреймов и таблиц по сравнению с `lynx`.

Рисунок 13-6. Links с открытым меню File



wget

`wget`(1) - эту консольная утилита, предназначенная для загрузки файлов по указанному URL. Хотя `wget` по сути не является веб-браузером, изначально он использовался для полной или частичной загрузки веб-сайтов для их просмотра в автономном режиме или для быстрой загрузки отдельных файлов с HTTP- или FTP-серверов. Основной синтаксис:

```
% wget <url>
```

Также вы можете использовать опции. Например, следующая команда загрузит веб-сайт Slackware:

```
% wget --recursive http://www.slackware.com
```

`wget` создаст каталог `www.slackware.com` и сохранит в нём файлы в таком же порядке, в каком они находятся на сайте.

Также `wget` может загружать файлы с FTP-серверов; просто укажите в URL FTP вместо HTTP.

```
% wget ftp://ftp.gnu.org/gnu/wget/wget-1.8.2.tar.gz
--12:18:16--  ftp://ftp.gnu.org/gnu/wget/wget-1.8.2.tar.gz
           => `wget-1.8.2.tar.gz'
Resolving ftp.gnu.org... done.
Connecting to ftp.gnu.org[199.232.41.7]:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.   ==> PWD ... done.
==> TYPE I ... done. ==> CWD /gnu/wget ... done.
==> PORT ... done.   ==> RETR wget-1.8.2.tar.gz ... done.
Length: 1,154,648 (unauthoritative)

100%[====================================>] 1,154,648     209.55K/s    ETA 00:00

12:18:23 (209.55KB/s) - `wget-1.8.2.tar.gz' saved [1154648]
```

У `wget`'а есть много других опций, который делают его удобным для использования в скриптах для работы с сайтами (зеркалирование веб-сайтов и т.п.). Дополнительную информацию смотрите на странице руководства.

13.9. FTP-клиенты

FTP означает File Transfer Protocol (протокол передачи файлов). Он позволяет вам обмениваться файлами между двумя компьютерами. Для этого существуют FTP-серверы и FTP-клиенты. В этом разделе мы поговорим о клиентах.

Если вы ещё не поняли, "клиент" - это вы. "Сервер" - это компьютер, который отвечает на ваши FTP-запросы и позволяет вам войти в систему.

Вы будете загружать и выгружать файлы на сервер. Клиент не может принимать FTP-подключения, он может только подключаться к серверам.

ftp

Чтобы подключиться к серверу FTP, просто выполните команду `ftp`(1), указав хост:

```
% ftp <имя хоста> [порт]
```

Если на хосте работает FTP-сервер, он запросит у вас имя пользователя и пароль. Вы можете войти в систему под собой или как "anonymous" (анонимно). Анонимные FTP-сайты являются очень популярными в качестве архивов с программным обеспечением. Например, чтобы загрузитьSlackware Linux через FTP, вы должны использовать анонимный FTP.

После подключения вы увидите приглашение `ftp>`. Для FTP существуют специальные команды, однако они очень похожи на другие стандартные команды системы. Ниже представлены некоторые основные команды и их действие:

Таблица 13-1. Команды *ftp*

Команда	Действие
`ls`	вывод списка файлов
`cd <имя_каталога>`	смена каталога
`bin`	установка двоичного режима передачи
`ascii`	установка текстового режима передачи
`get <имя_файла>`	загрузка файла
`put <имя_файла>`	выгрузка файла
`hash`	переключение индикатора статистики в виде знака решётки
`tick`	переключение индикатора счётчика байтов

Команда	Действие
`prom`	переключение интерактивного режима для загрузок
`mget <маска>`	загрузка файла или группы файлов; допускается использование шаблонов
`mput <маска>`	выгрузка файла или группы файлов; допускается использование шаблонов
`quit`	выход с FTP-сервера

Вы также можете использовать некоторые из следующих команд, названия которых говорят сами за себя: `chmod`, `delete`, `rename`, `rmdir`. Чтобы получить полный список всех команд и их значения, просто наберите **`help`** или **`?`** и вы получите на экране полный перечень.

FTP - это довольно простая в использовании программа, однако ей не хватает пользовательского интерфейса, использование которых стало уже практически стандартом на сегодняшний день. Страница руководства содержит описание опций командной строки `ftp`(1).

```
ftp> ls *.TXT
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
-rw-r--r--   1 root     100         18606 Apr  6  2002 BOOTING.TXT
-rw-r--r--   1 root     100         10518 Jun 13  2002 COPYRIGHT.TXT
-rw-r--r--   1 root     100           602 Apr  6  2002 CRYPTO_NOTICE.TXT
-rw-r--r--   1 root     100         32431 Sep 29 02:56 FAQ.TXT
-rw-r--r--   1 root     100        499784 Mar  3 19:29 FILELIST.TXT
-rw-r--r--   1 root     100        241099 Mar  3 19:12 PACKAGES.TXT
-rw-r--r--   1 root     100         12339 Jun 19  2002 README81.TXT
-rw-r--r--   1 root     100         14826 Jun 17  2002 SPEAKUP_DOCS.TXT
-rw-r--r--   1 root     100         15434 Jun 17  2002 SPEAK_INSTALL.TXT
-rw-r--r--   1 root     100          2876 Jun 17  2002 UPGRADE.TXT
226 Transfer complete.
ftp> tick
Tick counter printing on (10240 bytes/tick increment).
ftp> get README81.TXT
local: README81.TXT remote: README81.TXT
200 PORT command successful.
150 Opening BINARY mode data connection for README81.TXT (12339 bytes).
Bytes transferred: 12339
226 Transfer complete.
```

```
12339 bytes received in 0.208 secs (58 Kbytes/sec)
```

ncftp

ncftp(1) (произносится как "Nik-F-T-P") - это альтернатива традиционному ftp-клиенту, входящая в состав Slackware. Это всё ещё консольная программа, однако она имеет много преимуществ по сравнению с ftp:

- завершение по Tab;
- файл закладок;
- более свободные шаблоны подстановки;
- история команд.

По умолчанию ncftp пытается анонимно войти на указанный вами сервер. Вы можете заставить ncftp вывести приглашение для входа в систему с помощью опции "-u". После входа в систему вы можете использовать те же команды, что и для ftp за тем исключением, что вы увидите более приятный интерфейс, который больше напоминает bash.

```
ncftp /pub/linux/slackware > cd slackware-current/
Please read the file README81.TXT
  it was last modified on Wed Jun 19 16:24:21 2002 - 258 days ago
CWD command successful.
ncftp ...ware/slackware-current > ls
BOOTING.TXT               FAQ.TXT                   bootdisks/
CHECKSUMS                 FILELIST.TXT              extra/
CHECKSUMS.asc             GPG-KEY                   isolinux/
CHECKSUMS.md5             PACKAGES.TXT              kernels/
CHECKSUMS.md5.asc         PRERELEASE_NOTES          pasture/
COPYING                   README81.TXT              rootdisks/
COPYRIGHT.TXT             SPEEKUP_DOCS.TXT          slackware/
CRYPTO_NOTICE.TXT         SPEEK_INSTALL.TXT         source/
CURRENT.WARNING           Slackware-HOWTO
ChangeLog.txt             UPGRADE.TXT
ncftp ...ware/slackware-current > get README81.TXT
README81.TXT:                                           12.29 kB  307.07 kB/s
```

13.10. Общение с другими людьми

wall

`wall`(1) - это быстрый способ отправки сообщения всем пользователям системы. Базовый синтаксис:

```
% wall [файл]
```

В результате содержимое указанного [файла] будет показано на всех терминалах всех пользователей, работающих сейчас в системе. Если вы не укажете файл, wall будет читать стандартный вход. Поэтому вы можете просто набрать своё сообщение, а в конце нажать **Ctrl+d**.

`wall` не обладает большим числом функций за тем исключением, что вы можете отправить всем пользователям сообщение о том, что вы собираетесь выполнить задачу по обслуживанию системы или даже перезагрузить её, предоставив им, таким образом, время на то, чтобы сохранить свою работу и выйти из системы :)

talk

`talk`(1) позволяет двум пользователям общаться в чате. При этом экран разделяется по горизонтали на две равные части. Чтобы пригласить другого пользователя в чат, используйте следующую команду:

```
% talk <пользователь> [имя_tty]
```

Рисунок 13-7. Два пользователя в сеансе *talk*

Если вы указали только имя пользователя, подразумевается локальный запрос на чат. поэтому будут опрошены только локальные пользователи. Если вы хотите пригласить пользователя из определённого терминала (если этот пользователь несколько раз вошёл в систему), вам необходимо использовать имя_tty. Необходимую информацию для `talk` можно получить с помощью команды `w`(1).

`talk` может также работать с пользователями на удалённых хостах. Тогда к имени пользователя нужно просто добавить адрес e-mail. `talk` попытается связаться с этим пользователем на удалённом хосте.

`talk` несколько ограничен в своих возможностях. Он поддерживает работу только с двумя пользователями и только в полудуплексном режиме.

ytalk

`ytalk`(1) это замена `talk` с поддержкой обратной совместимости. Она входит в состав Slackware в виде команды `ytalk`. Синтаксис похож, однако имеет несколько отличий:

```
% ytalk <имя_пользователя>[#имя_tty]
```

Рисунок 13-8. Два пользователя в сеансе *ytalk*

Имя пользователя и терминал указываются так же как и в talk, за тем исключением, что вы должны соединить их знаком решётки (#).

ytalk обладает следующими преимуществами:

- поддержка более двух пользователей;

- с помощью **Esc** в любой момент может быть вызвано меню с опциями;
- вы можете выйти в шелл, оставаясь при этом в сеансе;
- и др...

Если вы администратор сервера, вам понадобится убедиться в том, что в `/etc/inetd.conf` включен порт `ntalk`. Это необходимо для нормальной работы `ytalk`.

Глава 14.

Безопасность

Безопасность важна для любой системы. Она может защитить вашу машину от атак, а также обеспечить сохранность важных данных. Эта глава полностью посвящена обеспечению безопасности вашей системы Slackware от любителей использовать чужие скрипты, крякеров и прочих злоумышленников. Учтите, что это только первоначальное обеспечение безопасности системы. Безопасность как таковая - это процесс, а не состояние.

14.1. Отключение служб

Первым делом после установки Slackware нужно отключить все ненужные вам службы. Любая служба может быть потенциальным источником уязвимости вашей системы, поэтому важно, чтобы у вас работало как можно меньше служб (т.е. только те, что вам действительно нужны). Службы запускаются из двух основных мест - `inetd` и скриптов инициализации.

Службы, запускаемые из *inetd*

Многие демоны, входящие в состав Slackware, запускаются из `inetd`(8). `inetd` - это демон (или суперсервер), прослушивающий все порты, используемые другими службами, настроенные на запуск эти демоном, и порождающий копии соответствующих демонов при попытке подключения к ним (т.н. запуск по запросу). Демоны, запускаемые из `inetd`, могут быть

отключены путём комментирования соответствующих строк в файле `/etc/inetd.conf`. Для этого откройте этот файл в своём любимом редакторе (напр., `vi`) и вы должны будете увидеть строки наподобие этой:

```
telnet stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
```

Вы можете отключить эту службу (и любую другую не нужную вам службу), закомментировав её (т.е. добавив в начало строки знак `#` (решётка)). Тогда приведенная выше строка примет следующий вид:

```
#telnet stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
```

После перезапуска `inetd` эта служба будет отключена. Вы можете перезапустить `inetd` с помощью команды:

```
# kill -HUP $(cat /var/run/inetd.pid)
```

Службы, запускаемые из скриптов инициализации

Остальные службы, запускаемые во время загрузки системы, запускаются из скриптов инициализации из `/etc/rc.d/`. Их можно отключить двумя способами: первый заключается в снятии разрешения на выполнение для соответствующего скрипта, а второй - в комментировании в скриптах соответствующих строк.

Например, SSH запускается своим собственным скриптом `/etc/rc.d/rc.sshd`. Вы можете отключить эту службу следующим способом:

```
# chmod -x /etc/rc.d/rc.sshd
```

Для служб, нет имеющих собственных скриптов запуска, для их отключения нужно закомментировать соответствующие строки в скриптах инициализации. Например, демон portmap запускается следующими строками в файле `/etc/rc.d/rc.inet2`:

```
# This must be running in order to mount NFS volumes.
# Start the RPC portmapper:
if [ -x /sbin/rpc.portmap ]; then
  echo "Starting RPC portmapper:  /sbin/rpc.portmap"
  /sbin/rpc.portmap
fi
# Done starting the RPC portmapper.
```

Этот демон можно отключить, добавив знак # в начале соответствующих строк:

```
# This must be running in order to mount NFS volumes.
# Start the RPC portmapper:
#if [ -x /sbin/rpc.portmap ]; then
#  echo "Starting RPC portmapper:  /sbin/rpc.portmap"
#  /sbin/rpc.portmap
#fi
# Done starting the RPC portmapper.
```

Эти изменения вступят в силу только после перезагрузки системы или перехода с 3-го уровня запуска на 4-й. Вы можете выполнить это, набрав в консоли следующую команду (вам понадобится снова войти в систему после перехода на 1-й уровень запуска):

```
# telinit 1
# telinit 3
```

14.2. Управление доступом к хосту

iptables

`iptables` - это программа настройки фильтрации пакетов для Linux версий 2.4 и выше. Ядро 2.4 (если быть точнее, 2.4.5) впервые было представлено (в виде опции) в Slackware версии 8.0, а используемым по умолчанию было

сделано в Slackware 8.1. В этом разделе раскрываются только основы использования этой программы, более полную информацию вы можете получить на сайте http://www.netfilter.org/. Эти команды могут быть вставлены в скрипт `/etc/rc.d/rc.firewall`, который необходимо сделать исполняемым, чтобы эти правила вступили в силу во время загрузки. Учтите, что неверные команды `iptables` могут заблокировать для вас вашу собственную машину. Если только вы не уверены на 100% в своих знаниях, всегда проверяйте, есть ли у вас локальный доступ к машине.

Первым делом следует установить политику по умолчанию на DROP во всех цепочках для входящих подключений:

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
```

После того, как всё запрещено, вы можете начать назначать политики, разрешающие подключения. Первым делом разрешите любой трафик для уже установленных соединений:

```
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Чтобы не нарушить работу приложений, использующих адрес обратной связи, обычно рекомендуется добавить следующее правило:

```
# iptables -A INPUT -s 127.0.0.0/8 -d 127.0.0.0/8 -i lo -j ACCEPT
```

Эти правила разрешают любой входящий и исходящий трафик для сети 127.0.0.0/8 (127.0.0.0 - 127.255.255.255) на интерфейсе обратной связи (`lo`). При создании правил неплохо было бы, чтобы правила были как можно более точными, дабы не разрешить по недосмотру какой-нибудь злоумышленный трафик. Другими словами, под правилами, разрешающими слишком мало, понимается больше правил и больше набора на клавиатуре.

Следующим делом следовало бы разрешить доступ к определённым службам, работающим на вашей машине. Если, к примеру, вам нужно

использовать на своей машине веб-сервер, вам надо использовать примерно такое правило:

```
# iptables -A INPUT -p tcp --dport 80 -i ppp0 -j ACCEPT
```

Это правило разрешит доступ для любой машины на 80-й порт вашей машины через интерфейс `ppp0`. Вам может понадобиться ограничить доступ к этой службе только для определённых машин. Следующее правило разрешит доступ к вашему веб-серверу только для адреса `64.57.102.34`:

```
# iptables -A INPUT -p tcp -s 64.57.102.34 --dport 80 -i ppp0 -j ACCEPT
```

Разрешение ICMP-трафика может быть полезным для задач диагностирования. Для этого вам следует использовать правило наподобие этого:

```
# iptables -A INPUT -p icmp -j ACCEPT
```

У большинства людей также есть необходимость настроить на своей шлюзовой машине трансляцию сетевых адресов (Network Address Translation, NAT), чтобы другие машины из локальной сети могли выходить через него в Интернет. Для этого вам нужно использовать следующее правило:

```
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Также вам понадобится включить IP-форвардинг. Для временного решения этой задачи вы можете воспользоваться следующей командой:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Чтобы включить IP-форвардинг на постоянной основе (т.е. чтобы изменения оставались в силе после перезагрузки), вам понадобится открыть в своём любимом текстовом редакторе файл `/etc/rc.d/rc.inet2` и изменить в нём следующую строку:

```
IPV4_FORWARD=0
```

...на следующую:

```
IPV4_FORWARD=1
```

Дополнительную информацию о NAT смотрите в NAT HOWTO[2].

tcpwrappers

`tcpwrappers` управляют доступом к демонам на уровне приложений, а не на уровне IP. Это даёт возможность создать дополнительный уровень защиты на тот случай, если управление доступом на уровне IP (Netfilter) работает некорректно. Например, если вы пересобрали ядро, но забыли включить в нём поддержку iptables, ваша защита на уровне IP сведётся на нет. Тогда защитить свою систему вам помогут tcpwrappers.

Управление доступом к службам, защищённым tcpwrappers'ами, осуществляется с помощью файлов `/etc/hosts.allow` и `/etc/hosts.deny`.

Основная часть людей использует лишь одну строку в файле `/etc/hosts.deny`, запрещающую по умолчанию доступ ко всем демонам. Вот эта строка:

```
ALL : ALL
```

После этого вы можете заняться предоставлением доступа к службам для определённых хостов, доменов или диапазонов IP-адресов. Это осуществляется с помощью файла `/etc/hosts.allow`, имеющего такой же формат.

Многие люди начинают с разрешения всех подключений от `localhost`. Этого можно достичь с помощью следующей строки:

```
ALL : 127.0.0.1
```

2 http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.txt

Чтобы разрешить доступ к SSHd для сети 192.168.0.0/24, можете использовать следующие правила:

```
sshd : 192.168.0.0/24
sshd : 192.168.0.
```

Также существует возможность ограничить доступ для хостов из определённых доменов. Это можно выполнить с помощью следующего правила (учтите, что оно основывается на обратной записи в DNS подключающегося хоста, которая не всегда соответствует действительности или может вообще отсутствовать, поэтому я бы не рекомендовал использовать её для хостов из Интернета):

```
sshd : .slackware.com
```

14.3. Поддержание системы в актуальном состоянии

Почтовая рассылка `slackware-security`

Всякий раз, когда в Slackware обнаруживается уязвимость, связанная с безопасностью, всем подписчикам почтовой рассылки `slackware-security@slackware.com` отправляется электронное письмо. Отчёты касаются уязвимостей в любой части Slackware за исключением программного обеспечения из `/extra` или `/pasture`. Эти сообщения с анонсами по безопасности содержат подробности относительно того, как можно получить обновлённые пакеты Slackware или устранить уязвимость, если это возможно.

Подписка на почтовые рассылки Slackware подробно рассмотрена в Разд. 2.2.2.

Каталог /patches

Все обновлённые пакеты, выпущенные для определённой версии Slackware (обычно только для устранения проблем с безопасностью для уже вышедших релизов Slackware), помещаются в каталог /patches. Полный путь к этим патчам будет зависеть от используемого вами зеркала, однако в общем случае он будет иметь такой вид /?CBL/:/slackware-x.x/patches/.

Перед установкой этих пакетов неплохо было бы проверить md5sum пакета. md5sum(1) - это консольная утилита, создающая "уникальный" математический хэш файла. Если в файле будет изменён хотя бы один бит, для него будет сгенерировано совершенно другое значение md5sum.

```
% md5sum пакет-<версия>-<архитектура>-<ревизия>.tgz
6341417aa1c025448b53073a1f1d287d  пакет-<версия>-<архитектура>-<ревизия>.tgz
```

Затем вам следует сравнить это со строкой для этого пакета из файла CHECKSUMS.md5, находящегося в корне каталога slackware-$,,-!。/ (а также в каталоге /patches в патчами), или из электронного письма из рассылки slackware-security.

Если у вас есть файл с md5-суммами, в можете использовать его в качестве источника для проверки командой md5sum с опцией -c.

```
# md5sum -c CHECKSUMS.md5
./ANNOUNCE.10_0: OK
./BOOTING.TXT: OK
./COPYING: OK
./COPYRIGHT.TXT: OK
./CRYPTO_NOTICE.TXT: OK
./ChangeLog.txt: OK
./FAQ.TXT: FAILED
```

Как видите, все файлы, для которых суммы сошлись, md5sum отметил в списке как "OK", а файлы, для которых суммы не сошлись, отмечены как "FAILED".

Глава 15.

Архивирование файлов

15.1. *gzip*

`gzip`(1) - это программа сжатия от GNU. Она берет один файл и сжимает его. Использование в общем случае:

```
% gzip файл
```

Полученный файл получит название `D09;.gz` и обычно его размер будет меньше, чем у исходного файла. Обратите внимание, что `D09;.gz` заменит собой исходный `D09;`. Это означает, что `D09;` больше не будет существовать, останется только его "gzip'нутая" копия. Обычные текстовые файлы будут хорошо сжиматься, а изображения jpeg, mp3-файлы и другие мультимедийные данные сжимаются не слишком хорошо, поскольку они являются уже сжатыми. Как правило использование `gzip`'а заключается в балансировке между конечным размером файла и временем, затрачиваемым на сжатие. Максимальная степень сжатия может быть достигнута следующим образом:

```
% gzip -9 файл
```

При этом понадобится больше времени на сжатие файла, однако размер будет настолько мал, насколько это вообще возможно для `gzip`. При использовании меньших значений степени сжатия процесс пройдёт быстрее, однако файл получится большего размера.

Распаковать файлы, сжатые gzip'ом, можно с помощью двух команд, которые на самом деле являются одной и то же программой. `gzip` распакует любой файл со стандартным расширением. Стандартные расширения: `.gz`, `-gz`, `.z`, `-z`, `.Z` или `-Z`. Первый способ заключается в вызове `gunzip`(1) с именем файла в качестве аргумента:

```
% gunzip файл.gz
```

При этом в текущем каталоге останется распакованная копия входного файла, а из его имени будет убрано расширение `.gz`. `gunzip` является частью `gzip` и её действие идентично вызову `gzip -d`. Так сложилось, что `gzip` часто произносят как `gunzip`, поскольку такое название звучит более круто. :^) (Наверное имеется в виду игра слов: *gun* - пушка, ствол и *zip* - всемирно известный архиватор - прим. переводчика.)

15.2. *bzip2*

`bzip2`(1) - это альтернативная программа сжатия, поставляемая в составе Slackware Linux. В ней используется другой алгоритм, отличный от того, что используется в `gzip`, и который в результате обладает некоторыми преимуществами и недостатками. Главным преимуществом `bzip2` является размер сжатого файла. `bzip2` практически во всех случаях сжимает лучше, чем `gzip`. Иногда при этом могут быть получены файлы гораздо меньшего размера. Это может оказаться весьма полезным для людей с медленными коммутируемыми подключениями. Также помните, что при загрузке программного обеспечения с публичных серверов ftp хорошим тоном считается загрузка файлов `.bz2`, а не `.gz`, поскольку так можно снизить нагрузку на сервер.

Недостатком `bzip2` является более высокая загрузка процессора, чем у `gzip`. Это означает, что сжатие файла bzip'ом в общем случае займёт больше времени. При выборе программы сжатия вам следует сравнить соотношение скорости сжатия и полученного размера и определить, что из

них для вас важнее.

Использование `bzip2` похоже на использование `gzip`, поэтому мы не будем подробно останавливаться на нём. По аналогии с `gunzip`, `bunzip2` идентичен команде `bzip2 -d`. Основное отличие заключается в том, что `bzip2` использует расширение `.bz2`.

```
% bzip2 файл
% bunzip2 файл.bz2
% bzip2 -9 файл
```

15.3. *tar*

`tar`(1) - это архиватор на магнитную ленту от GNU. Он берёт несколько файлов и/или каталогов и объединяет их в один большой файл. Это позволяет вам сжать целое дерево каталогов, что невозможно сделать с помощью `gzip` или `bzip2`. У `tar`'а есть много опций, которые подробно описаны на его странице руководства. В этом разделе будет показано только общее использование `tar`.

Наиболее общее использование `tar` заключается в распаковке и разархивировании пакета, загруженного с веб- или ftp-сайта. Большинство файлов имеют расширение `.tar.gz`. Довольно часто их называют "tarball"'ами. Это означает несколько файлов, заархивированных `tar`'ом, а затем сжатых `gzip`'ом. Вы также можете встретить файлы `.tar.Z`. По сути это то же самое, однако встречаются они в основном на старых системах Unix.

Кроме того вы можете натолкнуться на файлы `.tar.bz2`. В таком виде распространяются исходные тексты ядра, поскольку так они занимают меньше места для загрузки. Как вы уже наверное догадались, это файлы, заархивированные `tar`'ом и сжатые bzip'ом.

Вы можете извлечь все файлы из такого архива, воспользовавшись `tar`'ом с несколькими опциями командной строки. Разархивирование тарбола

выполняется с помощью опции `-z`, которая означает, что сначала файл должен быть пропущен через `gunzip`, а уже потом распакован. Наиболее общий метод распаковки тарболов:

```
% tar -xvzf файл.tar.gz
```

Здесь довольно много опций. Что же они означают? Опция `-x` означает извлечение. Это важно, поскольку она сообщает `tar`'у, что именно нужно сделать со входным файлом. В данном случае он будет снова разбит на файлы, из которых он был "слеплен". Опция `-v` означает включение подробного режима. При этом на экран будут выведены названия всех извлекаемых из архива файлов. Неплохо было бы использовать эту опцию, чтобы распаковка не выглядела слишком скучно. Вы также можете использовать `-vv`, чтобы вывод был ещё более подробным и вы получили ещё больше информации об извлекаемых файлах. Опция `-z` сообщает `tar`'у о том, что `D09;.tar.gz` нужно сначала пропустить через `gunzip`. И, наконец, опция `-f` сообщает `tar`'у, что следующая строка в командной строке - это файл, с которым надо работать.

Существует несколько способов записи этой же команды. На старых системах, на которых отсутствует подходящая копия **GNU** `tar`, вы можете увидеть следующую запись этой же команды:

```
% gunzip файл.tar.gz | tar -xvf -
```

В этой команде файл сначала распаковывается, а результат отправляется в `tar`. Т.к. `gzip` по умолчанию записывает выходные данные на стандартный вывод, эта команда будет записывать распакованный файл на стандартный вывод. Затем конвейер перенаправляет этот поток в `tar` для распаковки. Знак "-" означает, что данные берутся со стандартного входа. Таким образом команда разархивирует поток данных, полученный из `gzip`, и запишет его на диск.

Другим способом записи первоначальной команды является убирание знака "-" перед опциями:

```
% tar xvzf файл.tar.gz
```

Также вы можете натолкнуться на bzip'нутый архив. Версия `tar`, представленная в Slackware Linux, может работать с ними точно так же, как и с gzip'нутыми архивами. Просто вместо опции `-z` вам нужно использовать `-j`:

```
% tar -xvjf файл.tar.bz2
```

Учтите, что `tar` будет сохранять извлечённые из архива файлы в текущий каталог. Поэтому, если у вас в `/tmp` есть архив, который вы хотите распаковать в свой домашний каталог, у вас есть несколько вариантов. Первый - архив можно переместить в ваш домашний каталог, а затем развернуть его с помощью `tar`. Второй - вы можете указать путь к архиву в командной строке. Третий - вы можете использовать опцию `-C`, чтобы разархивировать тарбол в указанный после этой опции каталог.

```
% cd $HOME
% cp /tmp/файл.tar.gz .
% tar -xvzf файл.tar.gz

% cd $HOME
% tar -xvzf /tmp/файл.tar.gz

% cd /
% tar -xvzf /tmp/файл.tar.gz -C $HOME
```

Все приведенные выше записи являются эквивалентными. В каждой из них архив разворачивается в ваш домашний каталог, а исходный сжатый архив остаётся на месте.

Итак, что же хорошего в том, что вы можете распаковывать архивы, если вы не можете создавать их? `tar` умеет делать и это. В большинстве случаев можно просто заменить опцию "`-x`" на "`-c`".

```
% tar -cvzf файл.tar.gz .
```

В этой команде опция `-c` сообщает `tar`'у, что нужно создать архив, а опция `-z` пропускает полученный архив через `gzip`, чтобы сжать его. `D09;.tar.gz` - это имя создаваемого файла.

Указание опции "`-f`" не всегда есть обязательным, однако в любом случае лучше её использовать. Без неё `tar` будет записывать свои данные на стандартный вывод, что обычно используется для перенаправления потока по конвейеру в другую программу, например, так:

```
% tar -cv файл.tar . | gpg --encrypt
```

Эта команда создаёт несжатый tar-архив с содержимым текущего каталога, а затем пропускает тарбол через программу `gpg`, которая шифрует и сжимает архив, делая невозможным его чтение кем-либо, у кого нет вашего секретного ключа.

15.4. *zip*

И в завершение, есть две утилиты, которые используются для работы с zip-файлами. Они очень распространены в мире Windows, поэтому в Linux тоже есть программы для работы с этим форматом. Программа для сжатия называется `zip`(1), а для распаковки - `unzip`(1).

```
% zip foo *
```

При этом будет создан файл `foo.zip`, который будет содержать все файлы из текущего каталога. `zip` автоматически добавит к имени файла расширение `.zip`, поэтому его не нужно добавлять к имени выходного файла. Вы также можете рекурсивно сжать текущий каталог, добавив в архив все находящиеся в нём подкаталоги:

```
% zip -r foo *
```

Распаковка файлов выполняется так же просто.

```
% unzip foo.zip
```

Эта команда извлечёт все файлы из архива `foo.zip`, включая каталоги.

Утилиты `zip` имеют несколько расширенных опций для создания самораспаковывающихся архивов, исключения файлов, управления размером сжатого файла, вывода информации о выполняемых действиях и многого другого. О том, как использовать эти опции, вы можете узнать из страницы руководства к `zip` и `unzip`.

Глава 16.

Редактор Vi

---

vi(1) - это стандартная программа Unix для редактирования текста, и хотя мастерство в рабое с ним сегодня не является настолько важной задачей, как это было раньше, оно всё ещё очень ценится. Существует несколько версий (клонов) vi: vi, elvis, vile и vim. Одна из них доступна практически во всех версиях Unix, а также и в Linux. Все эти версии обладают одинаковыми наборами основных функций и команд, поэтому, изучив один из клонов, вы с лёгкостью перейдёте на другой. Из-за наличия на сегодняшний день большого разнообразия текстовых редакторов, доступных в дистрибутивах Linux и версиях Unix, многие люди больше не используют vi. Тем не менее он всё ещё остаётся наиболее универсальным текстовым редактором в Unix и Unix-подобных системах. Мастерство владения vi означает, что вам не никогда не придётся сидеть за Unix-машиной и чувствовать себя неуютно, имея в своём распоряжении хотя бы один мощный текстовый редактор.

vi обладает большим числом мощный функций, включая подсветку синтаксиса, форматирование кода, мощный механизм поиска с заменой, макросы и многое другое. Эти возможности делают его особо привлекательным для программистов, веб-разработчиков и др. Системные администраторы по достоинству оценят его автоматизацию и интеграцию с командным процессором.

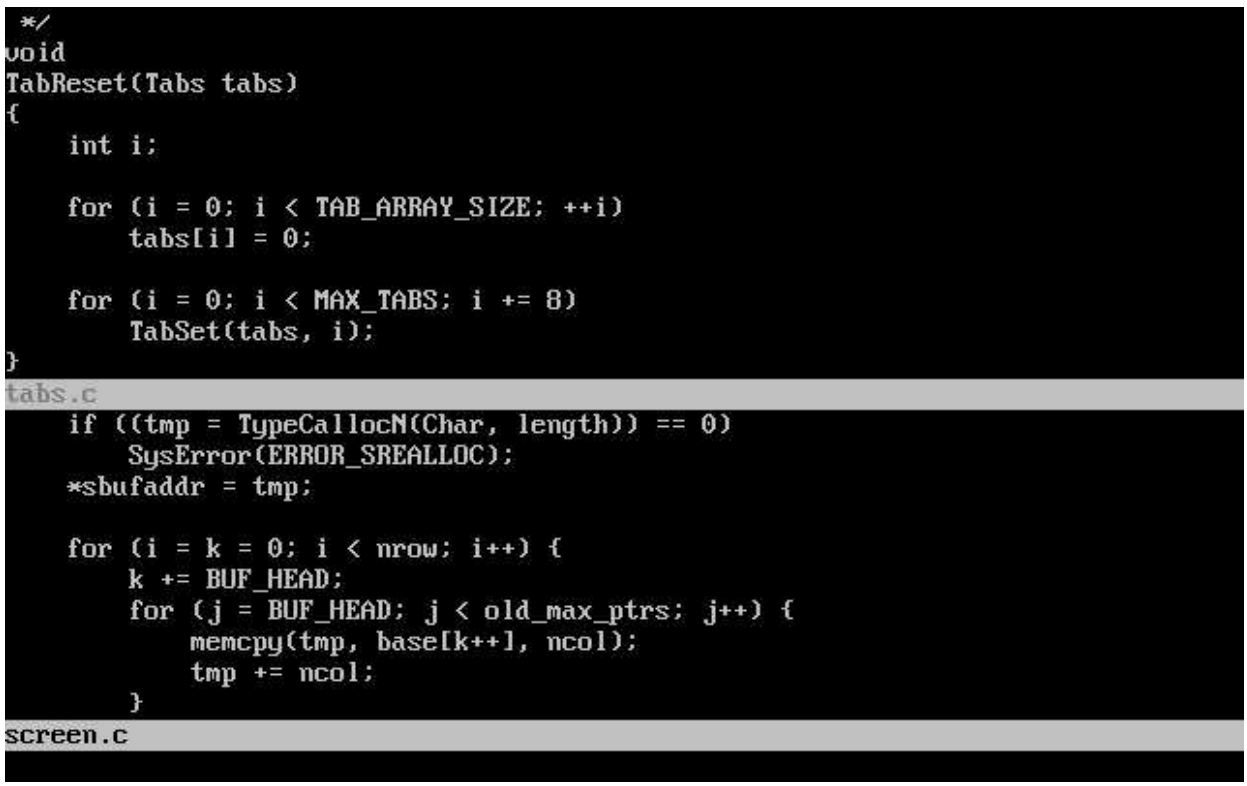
В Slackware Linux версией vi по умолчанию является elvis. Другие версии, включая vim и gvim, будут будут доступны, если вы установили соответствующие пакеты. gvim - это версия vim для X Window, имеющая панели инструментов, отделяемые меню и диалоговые окна.

16.1. Запуск vi

`vi` может быть запущен из командной строки различными способами. Самый простой из них:

```
% vi
```

Рисунок 16-1. Сеанс vi.



При этом будет запущен `vi` с пустым буфером. В большинстве случаев вы увидите просто пустой экран. Это так называемый “командный режим”, в котором от вас ожидаются какие-либо действия. Описания различных режимов работы `vi` смотрите в Разд. 16.2. Чтобы выйти из `vi`, наберите следующее:

```
:q
```

При условии, что вы не сделали в файле никаких изменений, эта команда закроет `vi`. Если были сделаны изменения, вы будете предупреждены об этом и дана информация о том, как проигнорировать их. Игнорирование изменений означает добавление восклицательного знака после "**q**":

```
:q!
```

Обычно восклицательный знак означает принудительное выполнение какого-то действия. Позже мы подробнее рассмотрим это и другие комбинации клавиш.

Вы также можете запустить `vi` вместе с уже существующим файлом. Например, файл `/etc/resolv.conf` можно открыть таким образом:

```
% vi /etc/resolv.conf
```

И, наконец, `vi` можно запустить на определённой строке файла. Это особенно полезно для программистов, когда сообщение об ошибке содержит номер строки, на которой вылетела программа. Например, вы можете запустить `vi` на 47-й строке `/usr/src/linux/init/main.c`:

```
% vi +47 /usr/src/linux/init/main.c
```

`vi` покажет на экране указанный файл и поместит курсор на заданной строке. В случае, если вы указали строку за пределами конца файла, `vi` поместит курсор на последней строке. Это особенно полезно для программистов, поскольку в случае возникновения ошибки они могут попасть сразу в нужно место файла без необходимости поиска этого места.

16.2. Режимы

`vi` работает в различных режимах, которые используются для выполнения различных задач. При первом запуске `vi` вы попадаете в командный режим. В нём вы можете выполнять различные команды для работы

с текстом, перемещения по файлу, сохранения, выхода и изменения режимов. Редактирование текста выполняется в режиме вставки. Вы можете быстро переключаться между режимами с помощью разнообразных комбинаций клавиш, описанных ниже.

Командный режим

Сначала вы попадаете в командный режим. В этом режиме вы не можете вводить текст или редактировать уже существующий. Однако вы можете манипулировать текстом, выполнять поиск, выходить, сохранять, загружать новые файлы и др. Данную информацию следует рассматривать только как знакомство с командным режимом. Описания различных команд смотрите в Разд. 16.7.

Наверное самой часто используемой командой в командном режиме является переключение в режим вставки. Оно осуществляется нажатием на клавишу **i**. Курсор изменяет свой вид, а в нижней части экрана появляется надпись -- *INSERT* -- (или -- *ВСТАВКА* --) (учтите, что это не происходит во всех клонах `vi`). Теперь все ваши нажатия на клавиатуре будут вставляться в текущий буфер и отображаться на экране. Чтобы вернуться назад в командный режим, нажмите клавишу **ESCAPE**.

Командный режим также позволяет вам перемещаться по файлу. В некоторых системах вы можете использовать для этого клавиши со стрелками. В других системах вам может потребоваться использовать более традиционные клавиши: “**hjkl**”. Вот перечень этих клавиш и их использование для перемещения:

h	перемещение влево на один символ
j	перемещение вниз на один символ
k	перемещение вверх на один символ
l	перемещение вправо на один символ

Чтобы переместиться, просто нажмите клавишу. Как вы позже увидите,

эти клавиши можно комбинировать с числами, чтобы использовать их более эффективно.

Многие из команд, которые вы увидите в командном режиме, начинаются с двоеточия. Например, выход - это **:q**, как уже упоминалось выше. Двоеточие означает, что это команда, а "**q**" говорит `vi` выйти. Другие команды представляют собой необязательные числа, следующие после буквы. Эти команды не содержат в начале двоеточия и используются в основном для работы с текстом.

Например, удаление в файле одной строки выполняется нажатием **dd**. При этом будет удалена строка, в которой находится курсор. Выполнение команды **4dd** сообщит `vi` удалить строку, в которой находится курсор и три строки после неё. В общем случае числа сообщают `vi`, сколько раз нужно выполнить команду.

Вы можете комбинировать числа с клавишами перемещения, чтобы переместиться за один раз на нужное количество символов. Например, **10k** выполнит перемещение вверх на десять строк.

Командный режим можно также использовать для вырезания и вставки текста и считывания других файлов в текущий буфер. Копирование текста выполняется с помощью клавиши **y** (**y** означает **yank** - резкий рывок). Копирование текущей строки выполняется командой **yy**, а перед ней может следовать число для копирования большего количества строк. Затем переместите курсор в нужное место и нажмите **p**. Текст будет вставлен в строку сразу после текущей.

Вырезание текста выполняется командой **dd**, а **p** можно использовать для вставки вырезанного текста назад в файл. Прочитать текст из другого файла также является простой процедурой. Просто наберите **:r**, и через пробел добавьте имя файла, содержащего текст для вставки. Содержимое файла будет вставлено в текущий буфер в строку после курсора. Более сложные клоны `vi` даже обладают возможностью автозавершения имён файлов как в командном процессоре.

Последняя рассматриваемая здесь функция - поиск. Командный режим позволяет выполнять простой поиск, а также содержит более сложные команды поиска с заменой, использующие мощную версию регулярных выражений. Подробное изучение регулярных выражений выходит за рамки этой главы, а здесь будет рассмотрен только простой поиск.

Простой поиск выполняется нажатием на клавишу **/**, после чего вводится искомый вами текст. `vi` будет искать его от текущего положения курсора до конца файла, остановившись на первом найденном совпадении. Учтите, что неполные совпадения также вызовут остановку поиска. Например, поиск слова "*the*" приведёт к остановке на словах "*then*", "*therefore*" и т.д. Это происходит вследствие того, что все эти слова содержат "*the*".

После того, как `vi` найдёт первое совпадение, вы можете продолжить поиск до следующего совпадения, нажав на клавишу **/** и **Enter**. Вы также можете выполнять поиск по файлу в обратном направлении, заменив слэш на **?**. Например, поиск по файлу в обратном направлении слова "*the*" выполняется с помощью команды **?the**.

Режим вставки

Вставка и замена текста выполняется в режиме вставки. Как уже упоминалось выше, чтобы перейти в этот режим, надо в командном режиме нажать клавишу **i**. После этого весь набираемый вами текст будет вставляться в текущий буфер. Нажатие на клавишу **ESCAPE** вернёт вас назад в командный режим.

Замена текста осуществляется несколькими способами. В командном режиме нажатие на **r** позволит вам заменить один символ под курсором. Просто введите новый символ и он заменит тот, что находится под курсором. Затем вы снова будет возвращены в назад в командный режим. Нажатие на клавишу **R** позволит вам заменить любое количество символов. Чтобы выйти из этого режима и вернуться в командный режим, просто нажмите **ESCAPE**.

Существует ещё один способ для переключения между режимами вставки и замены. Нажатие на клавишу **INSERT** в командном режиме переведёт вас в режим вставки. При работе в этом режиме клавиша **INSERT** служит переключателем между режимами вставки и замены. Нажатие на неё один раз позволит вам заменять текст. Ещё одно нажатие вернёт вас назад в режим вставки.

16.3. Открытие файлов

`vi` позволяет вам открывать файлы в командном режиме, а также сразу указывать открываемый файл при запуске из командной строки. Пример открытия файла `/etc/lilo.conf`:

```
:e /etc/lilo.conf
```

Если вы сделали изменения в текущем буфере и не сохранили их, `vi` предупредит вас об этом. Вы всё ещё можете открыть файл, не сохраняя текущий буфер, набрав команду **:e!**, а через пробел добавив имя файла. В общем случае предупреждения `vi` можно проигнорировать, добавив после команды восклицательный знак.

Если вам нужно повторно открыть текущий файл, вы можете набрать для этого команду **e!**. Это особенно полезно, если вы каким-то образом напортачили в файле и хотите повторно открыть его.

Некоторые клоны `vi` (например, `vim`) позволяют открывать несколько буферов одновременно. К примеру, чтобы открыть файл `09-vi.sgml` из своего домашнего каталога одновременно с другим уже открытым файлом, надо выполнить следующую команду:

```
:split ~/09-vi.sgml
```

Новый файл будет показан в верхней половине экрана, а старый - в нижней половине. Существует много команд для управления разделением экрана

и многие из них в чём-то похожи на команды `Emacs`. Лучшим источником поиска информации об этих командах является страница руководства для вашего клона **vi**. Обратите внимание, что многие клоны не поддерживают разделение экрана, поэтому вы можете вообще не увидеть поддержки этой функции.

16.4. Сохранение файлов

В `vi` существует несколько способов сохранения файлов. Если вам нужно сохранить содержимое текущего буфера в файл `randomness`, наберите следующую команду:

```
:w randomness
```

После того, как вы сохранили файл в первый раз, для повторного его сохранения наберите просто **:w**. Все изменения будут записаны в файл. После сохранения файла вы будете возвращены назад в командный режим. Если вам нужно сохранить файл и выйти из `vi` (очень распространённая ситуация), вам нужно набрать **:wq**. Эта команда говорит `vi` сохранить текущий файл и вернуться назад в командный процессор.

Может случиться так, что вам нужно сохранить файл, помеченный атрибутом “только для чтения”. Тогда вы можете сохранить его, добавив восклицательный знак после команды записи:

```
:w!
```

Однако вы всё-таки можете столкнуться с ситуацией, когда вы не сможете выполнить запись в файл (например, если вы пытаетесь отредактировать файл, владельцем которого является другой пользователь). В этом случае `vi` сообщит вам, что он не может сохранить файл. Если вам действительно нужно отредактировать файл, вам понадобится вернуться назад и отредактировать его под `root`’ом или (что есть более предпочтительным) под владельцем этого файла.

16.5. Выход из vi

Одним из способов выхода из `vi` является команда **:wq**, которая перед выходом сохранит текущий буфер. Вы также можете выйти из программы без сохранения посредством **:q** или (наиболее часто) **:q!**. Последняя команда используется в случае, если вы изменили файл, но не хотите сохранять эти изменения.

Иногда может случиться так, что ваша машина зависнет или `vi` "вылетит". Однако для минимизации потерь в таких случаях и `elvis`, и `vim` принимают определённые меры. Оба редактора время от времени сохраняют открытые буферы во временный файл. Этот файл обычно называется так же как и оригинальный файл, но с точкой в начале имени. Это делает такой файл скрытым.

Этот временный файл удаляется после нормального завершения работы редактора. Это означает, что временная копия будет всё ещё доступна, если что-то вылетит или зависнет. Когда вы снова вернётесь к редактированию файла, вам будут предложены на выбор варианты действий. В большинстве случаев может быть восстановлена большая часть несохранённой работы. `elvis` также отправит вам письмо (из Graceland'а как ни странно :), сообщающее вам, что есть резервная копия.

16.6. Настройка vi

Выбранный вами клон `vi` может быть настроен различными способами.

Существует много различных команд для настройки `vi` в командном режиме. В зависимости от вашего редактора вы можете включить функции, упрощающие написание кода (наподобие подсветки синтаксиса, автоматического отступа и т.п.), использовать макросы для автоматизации задач, включить замену текста и др.

Почти все эти команды могут быть указаны в конфигурационном файле

в вашем домашнем каталоге. `elvis` использует файл `.exrc`, а `vim` - `.vimrc`. Большинство команд, которые могут быть выполнены в командном режиме, могут быть указаны в конфигурационном файле. Это может быть конфигурационная информация, замены текста, макросы и др.

Рассмотрение всех этих параметров и их различий в версиях редактора представляет собой сложную и трудоёмкую задачу. Дополнительную информацию смотрите на странице руководства или на веб-сайте своего редактора `vi`. Некоторые редакторы (например, `vim`) содержат исчерпывающую встроенную справку, которая может быть вызвана командой **:help** или чем-то похожим. Вы также можете обратиться к книге "*Learning the `vi` Editor*" Ламба (Lamb) и Роббинса (Robbins) от издательства O'Reilly.

Многие общие программы в Linux по умолчанию загружают текстовый файл в `vi`. Например, редактирование crontab'ов выполняется по умолчанию в `vi`. Если вас не устраивает `vi` и вы хотели бы использовать вместо него другой редактор, вам нужно установить переменную окружения `VISUAL`, указав в ней предпочитаемый редактор. Информация об установке переменных доступна в 8-й главе в разделе под названием "Переменные окружения". Если вы хотите, чтобы этот редактор устанавливался как используемый по умолчанию при каждом вашем входе в систему, добавьте команду установки переменной VISUAL в свой файл `.bash_profile` или `.bashrc`.

16.7. Клавиши vi

Этот раздел представляет собой краткий справочник по общим командам `vi`. Некоторые из них уже были рассмотрены ранее, а некоторые окажутся для вас новыми.

Таблица 16-1. Перемещение

Действие	Клавиши
Влево, вниз, вверх, вправо	**h**, **j**, **k**, **l**
В конец строки	**$**
В начало строки	**^**
В конец файла	**G**
В начало файла	**:1**
На строку 47	**:47**

Таблица 16-2. Редактирование

Действие	Клавиша
Удаление строки	**dd**
Удаление пяти строк	**5dd**
Замена символа	**r**
Удаление символа	**x**
Удаление десяти символов	**10x**
Отмена последнего действия	**u**
Объединение текущей и следующей строк	**J**
Замена старого на новое, глобально	**%s'старое'новое'g**

Таблица 16-3. Поиск

Действие	Клавиша
Поиск "asdf"	**/asdf**
Поиск "asdf" в обратном направлении	**?asdf**
Повтор последнего поиска в прямом направлении	**/**
Повтор последнего поиска в обратном направлении	**?**

Действие	Клавиша
Повтор последнего поиска в том же направлении	**n**
Повтор последнего поиска в обратном направлении	**N**

Таблица 16-4. Сохранение и выход

Действие	Клавиша
Выйти	**:q**
Выйти без сохранения	**:q!**
Записать и выйти	**:wq**
Записать без выхода	**:w**
Перезагрузить открытый в данный момент файл	**:e!**
Записать буфер в файл `asdf`	**:w asdf**
Открыть файл `hejaz`	**:e hejaz**
Загрузить файл `asdf` в буфер	**:r asdf**
Загрузить вывод команды `ls` в буфер	**:r !ls**

Глава 17.

Редактор **Emacs**

В то время как `vi` (и его клоны) является без сомнения самым распространённым редактором в Unix-подобных системах, Emacs считается вторым хорошим редактором. Вместо различных “режимов”, применяемых в `vi`, для ввода команд в нём используются комбинации клавиш с **Control** и **Alt**. Подобным же образом вы можете использовать комбинации **Control** и **Alt** в текстовом процессоре и многих других приложениях для выполнения определённых функций. (Однако следует отметить, что эти команды редко совпадают; так во многих современных приложениях для копирования, вырезания и вставки текста используются **Ctrl-C/ X/ V**, а в Emacs для этого используются другие клавиши и несколько отличающийся принцип работы.)

Также, в отличие от `vi`, представляющего собой (отличный) редактор и больше ничего, Emacs - это программа с практически неограниченными возможностями. Emacs (по большей части) написан на Lisp - очень мощном языке программирования, характерной особенностью которого является то, что любая написанная на этом языке программа автоматически сама для себя является компилятором Lisp. Это означает, что пользователь может самостоятельно расширять возможности Emacs, а по сути писать новые программы “на Emacs’е”.

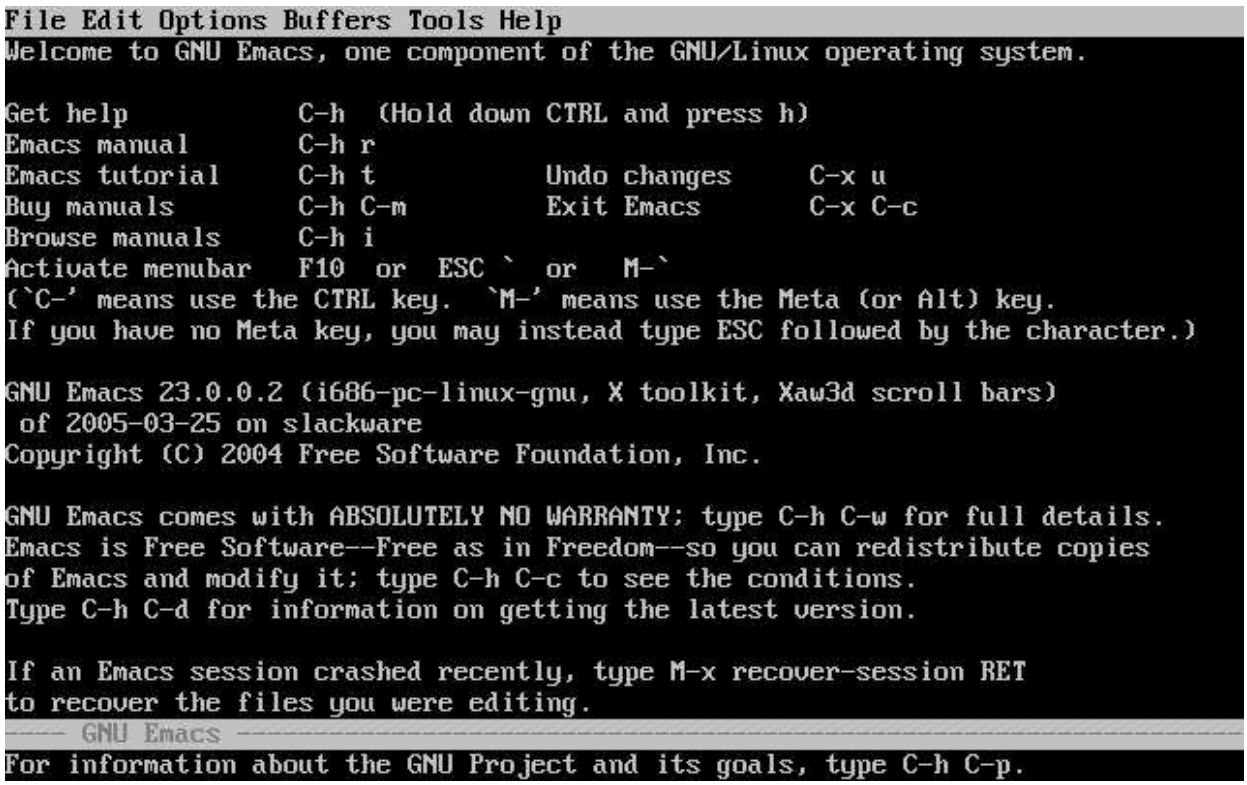
Как результат Emacs - это уже не просто редактор. Для Emacs существует много дополнительных пакетов (многие поставляются с исходными текстами), предоставляющих всевозможные функциональные возможности. Многие из них связаны с редактированием текста, что является первостепенной задачей Emacs’а, но не единственной. Например,

для Emacs существуют различные программы для работы с электронными таблицами и базами данных, игры, почтовые клиенты, клиенты чтения новостей (самый популярный - Gnus) и др.

Существует две основные версии Emacs: GNU Emacs (эта версия входит в состав Slackware) и XEmacs. Последняя *не* является версией Emacs для X'ов. И Emacs, и XEmacs работают как консоли, так и в X'ах. XEmacs был начат как проект с более чистым кодом Emacs. В настоящее время обе версии активно разрабатываются, и между двумя командами разработчиков ведётся тесное сотрудничество. В данной главе не имеет особого значения, что вы используете - Emacs или XEmacs - разница между ними для обычного пользователя не настолько важна.

17.1. Запуск emacs

Emacs можно запустить из консоли, набрав `emacs`. Если вы работаете в X'ах, Emacs (обычно) запустится в своём собственном окне с меню в верхней части, в котором вы можете найти наиболее важные функции. При запуске Emacs сначала покажет сообщение с приветствием, а затем, через несколько секунд, вы окажетесь в буфере *scratch* (см. Разд. 17.2).



Вы также можете запустить Emacs с одновременной загрузкой в него существующего файла:

```
% emacs /etc/resolv.conf
```

Эта команда запустит Emacs и загрузит в него указанный файл, пропустив сообщение с приветствием.

Командные клавиши

Как упоминалось ранее, в Emacs для команд используются комбинации клавиш с **Control** и **Alt**. Условно принято записывать эти команды в виде **C-буква** и **M-буква** соответственно. Так, **C-x** означает **Control+x**, а **M-x** означает **Alt+x**. (Буква **M** используется вместо A потому, что изначально клавиша **Alt** не существовала, зато существовала клавиша **Meta**. Эта клавиша (**Meta**) была на всех клавиатурах, однако со временем исчезла, и в

Emacs её функции заменила клавиша **Alt**.)

Многие команды Emacs состоят из последовательностей клавиш и их комбинаций. Например, **C-x C-c** (т.е. **Control-x** , после которого следует **Control-c**) завершает работу Emacs, **C-x C-s** сохраняет текущий файл. Учтите, что **C-x C-b** - это *не* то же самое, что **C-x b**. Первая комбинация означает сначала нажатие **Control-x**, а затем **Control-b**, а вторая - нажатие **Control-x**, после которого следует просто '**b**'.

17.2. Буферы

В Emacs понятие "буферы" является основополагающим. Каждый открываемый вами файл загружается в свой собственный буфер. Более того, в Emacs есть несколько специальных буферов, которые не содержат файлы, а предназначены для других целей. Названия таких буферов обычно обрамлены звёздочками. Например, буфер, показываемый Emacs'ом после запуска, - это т.н. буфер *scratch*. В нём вы можете вводить текст как обычно, однако он не будет сохранён после выхода из Emacs.

Существует ещё один специальный буфер, о котором вам необходимо знать. Это т.н. минибуфер. Этот буфер состоит из единственной строки и всегда виден на экране: это самая последняя строка в окне Emacs, находящаяся под строкой состояния текущего буфера. В минибуфере Emacs выводит сообщения для пользователя, а также он является местом для выполнения команд, требующих от пользователя ввода данных. Например, при открытии файла Emacs попросит вас ввести его имя в минибуфере.

Переключение между буферами выполняется с помощью команды **C-x b**. При этом вы увидите приглашение для ввода имени буфера (обычно в качестве имени буфера выступает название редактируемого в нём файла), а в приглашении по умолчанию будет выбран буфер, в котором вы были до

переключения в текущий буфер или до его создания. Нажатие на ***Enter*** переключит вас в буфер по умолчанию.

Если вам нужно переключиться в другой буфер, а не в тот, что Emacs предлагает по умолчанию, просто наберите его имя. Обратите внимание, что при этом вы можете использовать так называемое автозавершение **Tab**'ом: введите первые несколько букв названия буфера и нажмите **Tab**; Emacs попытается самостоятельно завершить имя буфера. Завершение **Tab**'ом работает в Emacs везде, где это имееет смысл.

Получить список открытых буферов можно с помощью комбинации **C-x C-b**. Как правило эта команда разделит экран на две части, показав в верхней половине буфер, в котором вы работали, а в нижней половине - новый буфер под названием *Buffer List*. В этом буфере будет представлен список всех буферов, их размеры и режимы, а также файлы (если таковые присутствуют), открытые в этих буферах. Избавиться от этого буфера можно, набрав **C-x 1**.

Замечание: В X'ах список буферов также доступен в меню Buffer в главном меню.

17.3. Режимы

Каждый буфер в Emacs имеет связанный с ним режим. Этот режим очень отличается от идеи режимов в `vi`: режим говорит вам о типе буфера, в котором вы находитесь. Например, для обычных текстовых файлов существует text-mode (текстовый режим), однако также существуют такие режимы как: c-mode (режим C) для редактирования программ на C, sh-mode (режим командного процессора) для редактирования шелл-скриптов, latex-mode (режим latex) для редактирования файлов LaTeX, mail-mode (почтовый режим) для редактирования электронных писем и новостей и другие режимы. Режимы предоставляют специальные

возможности и функции, полезные для различных типов редактируемых файлов. Для режимов даже можно переопределять клавиши и клавишные команды. Например, в режиме текста клавиша Tab выполняет перескакивание на окончание следующего табулятора, а в большинстве режимов для языков программирования клавиша **Tab** делает отступ текущей строки на соответствующую глубину текущего блока, в котором находится эта строка.

Упомянутые выше режимы относятся к главным режимам. У каждого буфера есть только один главный режим. Дополнительно буфер может иметь один или несколько второстепенных режимов. Второстепенный режим предоставляет дополнительные возможности, которые могут быть полезными в определённых задачах редактирования. Например, если вы нажмёте клавишу **INSERT**, вы включите режим замены, т.е. именно то, что вы и ожидали от этой клавиши. Есть ещё режим автозаполнения (auto-fill), который удобен в комбинации с режимами текста или latex: в нём каждая набранная вами строка будет автоматически разбиваться с переносом на следующую строку при достижении определённого количества символов. Без использования этого режима для расширения параграфа вам нужно набрать **M-q**. (Что вы также можете использовать для повторного форматирования параграфа после того, как вы отредактировали в нём текст и он стал выглядеть менее аккуратно.)

Открытие файлов

Чтобы открыть в Emacs файл, наберите

```
C-x C-f
```

Emacs попросит вас ввести имя файла, вставив перед ним какой-то путь по умолчанию (обычно это `~/`). После ввода имени файла (можете использовать **Tab**-завершение) и нажатия на **ENTER** Emacs откроет файл в новом буфере и покажет этот буфер на экране.

Замечание: Emacs автоматически создаст новый буфер, я не будет загружать файл в текущий буфер.

Чтобы создать в Emacs'е новый файл, вы не можете сделать это, просто начав набирать текст. Вы сначала должны создать для него буфер и назначить ему имя файла. Вы можете выполнить это с помощью команды **C-x C-f**, набрав после этого имя файла, как если бы вы открывали существующий файл. Emacs сообщит вам, что указанный вами файл не существует, создаст для него новый буфер и выведет в минибуфере сообщение "(New file)".

Если вы нажмёте **C-x C-f**, а затем вместо имени файла введёте имя каталога, Emacs создаст новый буфер, в котором вы найдёте список всех файлов из этого каталога. Вы можете переместить курсор на нужный вам файл, нажать Enter, и Emacs откроет его. (На самом деле здесь вы можете выполнить очень много действий: удалять, переименовывать, перемещать файлы и др. Emacs сейчас находится в режиме dired, представляющий собой простейший файловый менеджер.)

Если вы нажмёте **C-x C-f** и вдруг передумаете, вы можете нажать **C-g** для отмены действия. **C-g** работает практически в любых ситуациях, когда вам нужно отменить действие или команду, которое вы начали, но не хотите доводить до конца.

17.4. Основы редактирования

После того, как вы открыли файл, вы можете перемещаться по нему с помощью курсора. **Клавиши со стрелками**, **PgUp** и **PgDn** работают так, как и ожидается. **Home** и **End** перемещают вас в начало и конец строки соответственно. (Вообще-то в старых версиях они выполняли переход в начало и конец буфера.) Однако для перемещения курсора существуют

ещё комбинации с клавишами **Control** и **Meta** (**Alt**). Поскольку для их использования вам не нужно перемещать руки по всей клавиатуре, с их помощью вы сможете перемещаться гораздо быстрее. Такие наиболее важные команды перечислены в Табл. 17-1.

Обратите внимание, что многие **Meta**-команды работают "параллельно" **Control**-командам за тем исключением, что последние оперируют единицами большей величины: так **C-f** выполняет переход вперёд на один символ, а **M-f** выполняет переход вперёд на одно слово и т.д.

Также обратите внимание, что для **M-<** и **M->** требуется нажать **Shift+Alt+запятая** и **Shift+Alt+точка** соответственно, поскольку символы < и > набираются как **Shift+запятая** и **Shift+точка**. (Конечно же за тем исключением, что у вас не используется раскладка клавиатуры, отличающаяся от стандартной американской раскладки.)

Учтите, что **C-k** удаляет (или как обычно говорят - убивает) весь текст от курсора до конца строки, но не удаляет саму строку (т.е. при этом не удаляется символ конца строки). Строка удаляется только в том случае, если после курсора нет текста. Другими словами, чтобы удалить целую строку, вам нужно поместить курсор в начало строки, а затем нажать **C-k** дважды: первый раз, чтобы удалить весь текст в строке, а второй, чтобы удалить саму строку.

17.5. Сохранение файлов

Для того, чтобы сохранить файл, наберите

```
C-x C-s
```

Emacs не будет спрашивать у вас имя файла, буфер будет попросту сохранён в файл, из которого он был загружен. Если вам нужно сохранить свой текст в другой файл, наберите

Таблица 17-1. Основные команды редактирования в Emacs

Команда	Результат
C-b	перемещение на один символ назад
C-f	перемещение на один символ вперёд
C-n	перемещение на одну строку вниз
C-p	перемещение на одну строку вверх
C-a	перемещение в начало строки
C-e	перемещение в конец строки
M-b	перемещение на одно слово назад
M-f	перемещение на одно слово вперёд
M-}	перемещение на один параграф вперёд
M-{	перемещение на один параграф назад
M-a	перемещение на одно предложение назад
M-e	перемещение на одно предложение вперёд
C-d	удаление одного символа под курсором
M-d	удаление до конца текущего слова
C-v	перемещение вниз на один экран (т.е. PgDn)
M-v	перемещение вверх на один экран (т.е. PgUp)
M-<	перемещение в начало буфера
M->	перемещение в конец буфера
C-_	отмена последнего изменения (может быть повторена); обратите внимание, что на самом деле для этого вы должны нажать **Shift+Control+дефис** .
C-k	удаление до конца строки
C-s	поиск вперёд
C-r	поиск назад

```
C-x C-w
```

Если вы сохраняете файл впервые за сеанс, Emacs обычно сделает из старого файла резервную копию, имеющую такое же название, но с тильдой на конце: так, если вы редактируете файл "`cars.txt`", Emacs создаст резервный файл "`cars.txt~`".

Этот файл является копией открытого вами файла. Также во время работы Emacs будет регулярно сохранять копии вашей работы в файл с именем, обрамлённым решётками: `#cars.txt#`. Эта резервная копия удаляется, когда вы сохраняете файл командой C-x C-s.

Во время редактирования файла вы можете убить буфер, в котором он открыт, набрав команду

```
C-x k
```

При этом Emacs спросит вас, какой именно буфер вы хотите убить, выбрав по умолчанию текущий. После выбора нужного буфера, нажмите на **ENTER**. Если вы ещё не сохраняли свой файл, Emacs спросит вас, действительно ли вы хотите убить буфер.

Выход из Emacs

Когда вы закончили работу в Emacs, вы можете набрать

```
C-x C-c
```

Эта команда закроет Emacs. Если у вас есть несохранённые файлы, Emacs сообщит вам об этом и спросит, не хотите ли вы сохранить их (по очереди для каждого). Если вы дадите отрицательный ответ(ы), Emacs выведет последний диалог с подтверждением, а затем завершит свою работу.

Глава 18.

Управление пакетами

Slackware

Пакет с программным обеспечением - это набор связанных программ, уже готовых к установке. Когда вы загружаете программу в виде архива с исходными текстами, вам необходимо вручную сконфигурировать, откомпилировать и установить её. В программных пакетах это уже сделано за вас. Всё, что вам нужно сделать - это установить пакет. Другой удобной функцией пакетов с ПО является то, что их очень легко удалить и обновить, если вы пожелаете сделать это. В состав Slackware входят все программы, необходимые для управления пакетами. Вы легко можете устанавливать, удалять, обновлять, создавать и изучать пакеты.

После того, как RedHat выпустили свой менеджер пакетов RedHat (RedHat Package Manager), появился миф о том, что в Slackware нет утилиты для управления пакетами. Это просто заблуждение. В Slackware всегда был менеджер пакетов, даже ещё до того, как появился RedHat. И хотя он не настолько наворочен или распространён как rpm (или подобный ему deb-формат), `pkgtool` и связанные с этим менеджером программы настолько же удобны для установки пакетов, как и rpm. Проблема с `pkgtool` заключается не в том, что он не существует, а в том, что он не проверяет зависимости.

Очевидно многие люди в сообществе Linux думают, что менеджер пакетов по определению должен включать в себя проверку зависимостей. Ну что ж, это просто не тот случай, поскольку в Slackware как раз так и сделано.

Это не означает, что у пакетов в Slackware отсутствуют зависимости, просто менеджер пакетов не проверяет их. Отслеживание зависимостей остаётся на совести системного администратора, и нам нравится такой подход.

18.1. Обзор формата пакетов

Перед тем, как приступить к изучению утилит, вам следует разобраться с форматом пакетов Slackware. В Slackware пакет - это просто tar-архив, сжатый `gzip`'ом. Собранные пакеты предназначены для распаковки в корневой каталог.

Вот пример фиктивной программы и её пакета:

```
./
usr/
usr/bin/
usr/bin/makehejaz
usr/doc/
usr/doc/makehejaz-1.0/
usr/doc/makehejaz-1.0/COPYING
usr/doc/makehejaz-1.0/README
usr/man/
usr/man/man1
usr/man/man1/makehejaz.1.gz
install/
install/doinst.sh
```

Система работы с пакетами распакует этот файл в корневой каталог и установит его. В базе данных пакетов будет создана запись с содержимым этого пакета, чтобы позже его можно было удалить или обновить.

Обратите внимание на подкаталог `install/`. Это специальный каталог, в котором находится скрипт, выполняемый после установки, под названием `doinst.sh`. Если менеджер пакетов найдёт этот файл, он запустит его после установки пакета.

В пакет могут быть внедрены и другие скрипты, однако более подробно о них рассказано ниже в Разд. 18.3.2.

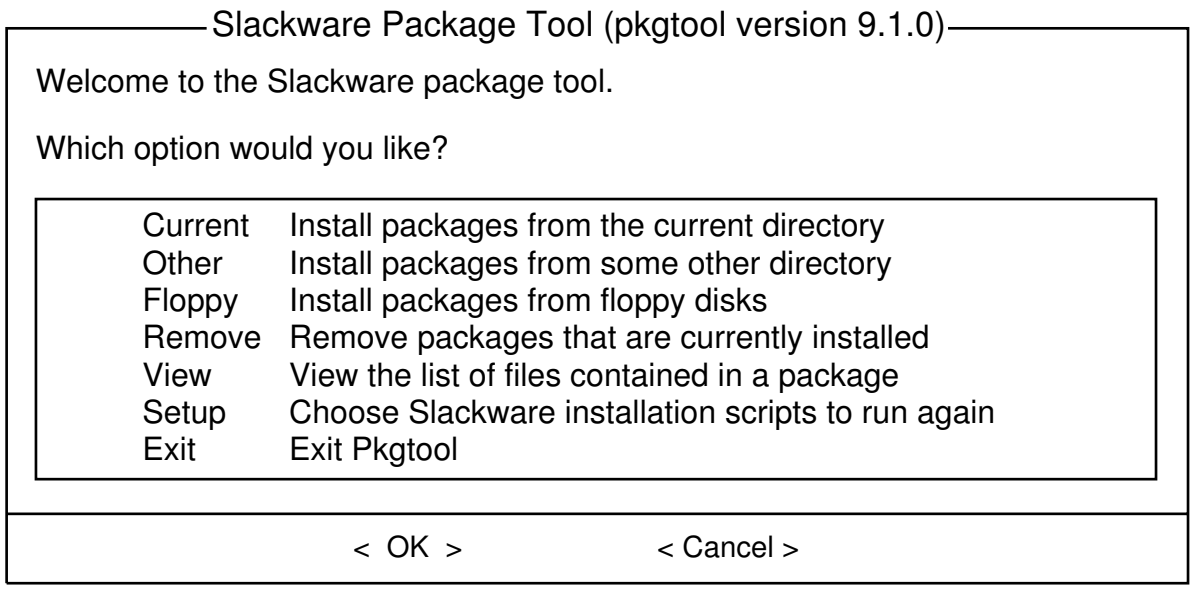
18.2. Утилиты для работы с пакетами

Для управления пакетами существуют четыре основные утилиты. Они выполняют установку, удаление и обновление пакетов.

pkgtool

`pkgtool`(8) - это программа с поддержкой меню, которая позволяет устанавливать и удалять пакеты. Главное меню показано на Рис. 18-1.

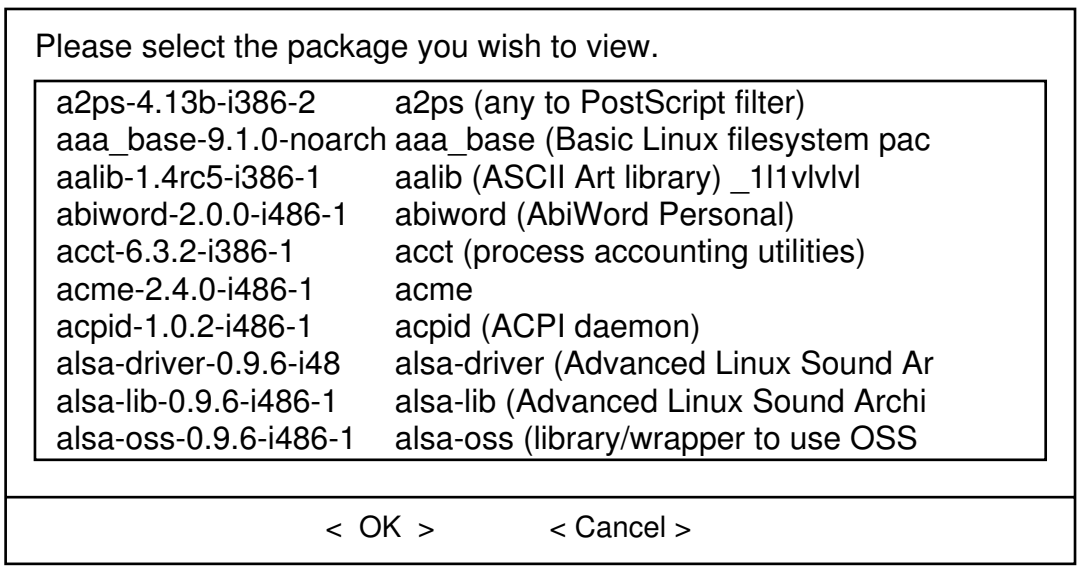
Рисунок 18-1. Главное меню pkgtool.



Установка может быть выполнена из текущего каталога, другого каталога или с дискет. Просто выберите нужный вам метод установки и pkgtool начнёт искать в указанном вами месте пригодные к установке пакеты.

Также вы можете просмотреть список установленных пакетов, как показано на Рис. 18-2.

Рисунок 18-2. Режим просмотра pkgtool



Если вам нужно удалить пакеты, выберите опцию **Remove** и вам будет представлен список всех установленных пакетов. Отметьте те, которые вы хотите удалить, и нажмите **OK**. `pkgtool` выполнит их удаление.

Некоторые пользователи отдают предпочтение этой утилите, а не утилитам командной строки. Однако, следует отметить, что эти консольные утилиты предлагают гораздо больше функций. Кроме того, обновление пакетов возможно только с помощью утилит командной строки.

installpkg

Утилита `installpkg`(8) выполняет установку в систему новых пакетов. Её синтаксис следующий:

```
# installpkg опция имя_пакета
```

У `installpkg` есть только три опции. Одновременно может быть использована только одна опция.

Таблица 18-1. Опции *installpkg*

Опция	Действие
-m	Выполняет действие makepkg в текущем каталоге.
-warn	Показывает, что произойдёт, если вы установите указанный пакет. Это полезно для ответственных систем, когда, перед тем как устанавливать что-либо, вы можете узнать что при этом произойдёт.
-r	Рекурсивно устанавливает все пакеты их текущего каталога и ниже. Имя пакета может содержать маску, которая будет использована при рекурсивном поиске пакетов для установки.

Если вы определите переменную окружения `ROOT` перед запуском `installpkg`, этот путь будет использован в качестве корневого каталога. Это полезно для настройки новых дисков для размещения на них корневых каталогов. Обычно они монтируются в `/mnt` или какой-то другой каталог, отличный от `/`.

Запись в базе данных об установленном пакете хранится в `/var/log/packages`. Запись представляет собой обычный текстовый файл, по одному на пакет. Если в пакете есть послеустановочный скрипт, он записывается в каталог `/var/log/scripts/`.

Вы можете указать несколько пакетов или использовать маски в именах файлов. Учтите, что `installpkg` не сообщит вам, если будет перезаписываться уже установленный пакет. Он просто установит его поверх старого. Если вы хотите, чтобы старые файлы из предыдущего пакета были безопасно удалены, используйте утилиту `upgradepkg`.

removepkg

Утилита `removepkg`(8) удаляет из системы установленные пакеты. Её синтаксис следущий:

```
# removepkg опция имя_пакета
```

У `removepkg` есть только четыре опции. Одновременно может быть использована только одна опция.

Таблица 18-2. Опции *removepkg*

Опция	Действие
-copy	Пакет копируется в специальный каталог для пакетов. Так можно создать дерево из оригинальных пакетов, не удаляя их.
-keep	Сохраняет временные файлы, созданные во время удаления. В действительности полезно только в целях отладки.
-preserve	Пакет удаляется, однако в тоже время он копируется в каталог для резервных копий.
-warn	Показывает, что бы произошло, если бы вы удалили пакет.

Если вы определите переменную окружения `ROOT` перед запуском `removepkg`, этот путь будет использован в качестве корневого каталога. Это полезно для настройки новых дисков для размещения на них корневых каталогов. Обычно они монтируются в `/mnt` или какой-то другой каталог, отличный от `/`.

`removepkg` выполняет поиск и в остальных установленных пакетах, но удаляет только те файлы, которые являются уникальными для указанного вами пакета. Он также найдёт послеустановочный скрипт для указанного пакета и удалит все созданные им символические ссылки.

Во время процесса удаления на экран выводится отчёт о ходе его выполнения. После удаления запись из базы данных пакетов перемещается в каталог `/var/log/removed_packages`, а скрипт, выполняемый после установки, перемещается в `/var/log/removed_scripts`.

Как и в случае с `installpkg` вы можете указать несколько пакетов или использовать маски в именах пакетов.

upgradepkg

Утилита `upgradepkg`(8) обновляет установленные пакеты Slackware. Её синтаксис следующий:

```
# upgradepkg имя_пакета
```

или

```
# upgradepkg старое_имя_пакета%старое_имя_пакета
```

`upgradepkg` сначала устанавливает новый пакет, а затем удаляет старый пакет, чтобы в системе больше не осталось старых файлов. Если у обновляемого пакета изменилось имя, используйте в команде знак процента для указания старого имени пакета (того, что установлен) и нового имени пакета (до которого вы выполняете обновление).

Если вы определите переменную окружения `ROOT` перед запуском `upgradepkg`, этот путь будет использован в качестве корневого каталога. Это полезно для настройки новых дисков для размещения на них корневых каталогов. Обычно они монтируются в `/mnt` или какой-то другой каталог, отличный от `/`.

`upgradepkg` не лишён недостатков. Вам всегда следует создавать резервные копии своих конфигурационных файлов. Если они будут удалены или перезаписаны, для нормальной работы вам потребуется восстановить их оригиналы.

Как и в случае с `installpkg` и `removepkg` вы можете указать несколько пакетов или использовать маски в именах пакетов.

rpm2tgz/rpm2targz

Менеджер пакетов Red Hat (Red Hat Package Manager, RPM) - это популярная на сегодняшний день система работы с пакетами. Многие распространители программного обеспечения предлагают свои продукты в формате RPM. Поскольку он не является нашим "родным" форматом, мы не рекомендуем вам использовать его. Однако некотрые вещи бывают доступны только в виде RPM (даже исходные тексты).

Мы предлагаем программу для преобразования RPM-пакетов в наш родной формат `.tgz`. Это позволит вам развернуть пакет (возможно, с помощью `explodepkg`) во временный каталог и изучить его содержимое.

Программа `rpm2tgz` создаст пакет Slackware с расширением `.tgz`, а `rpm2targz` создаёт архив с расширением `.tar.gz`.

18.3. Создание пакетов

Создание пакетов Slackware может быть как простым, так и сложным. Не существует чёткого метода для сборки пакетов. Единственное требование - пакет должен был tar-файлом, сжатым gzip'ом, и если в нём есть послеустановочный скрипт, это должен быть `/install/doinst.sh`.

Если вы заинтересованы в создании пакетов для своей системы или для обслуживаемой вами сети компьютеров, вам следует ознакомиться с различными сборочными скриптами, доступными в дереве исходных текстов Slackware. При создании пакетов мы используем несколько различных методов.

explodepkg

Утилита `explodepkg`(8) делает то же самое, что и `installpkg` делает при установке пакета, однако на самом деле она не устанавливает его и не делает запись в базе данных пакетов. Она просто извлекает содержимое пакета в текущий каталог.

Если вы посмотрите на дерево исходных текстов Slackware, вы увидите, как мы используем эту команду для пакетов "framework". Эти пакеты содержат каркас будущего пакета. В них находятся все необходимые имена файлов (нулевой длины), права доступа и владельцы. Скрипт сборки извлекает содержимое пакета из каталога с исходными текстами и помещает его в каталог сборки.

makepkg

Утилита `makepkg`(8) упаковывает содержимое текущего каталога в пакет Slackware. Он найдёт в дереве символические ссылки и добавит в скрипт, выполняемый после установки, блок команд, создающих эти ссылки во время установки пакета. Он также предупреждает вас о наличии в дереве пакета файлов нулевого размера.

Эта команда обычно запускается после того, как вы создали дерево своего пакета.

Скрипты SlackBuild

При необходимости пакеты Slackware могут быть собраны и другими способами. Не всё программное обеспечение в виде пакетов написано программистами для компиляции одним и тем же способом. Во многих есть опции компиляции, которые не используются в пакетах Slackware. Возможно, когда-то вам понадобятся эти функциональные возможности. Тогда вам придётся собрать свой собственный пакет. К счастью у многих

пакетов Slackware в дереве исходных текстов пакета вы можете найти скрипты SlackBuild.

Что же такое скрипт SlackBuild? Скрипты SlackBuild представляют собой исполняемые скрипты командного процессора, запускаемые вами под `root`'ом для настройки, компиляции и создания пакетов Slackware. Вы можете свободно изменять эти скрипты в дереве исходных текстов и запускать их для создания своих собственных версий стандартных пакетов Slackware.

18.4. Создание тегов и tag-файлов (для программы setup)

Программа setup управляет установкой пакетов с ПО в вашу систему Slackware. Существуют файлы, которые сообщают setup'у о том, какие пакеты должны быть установлены, какие являются необязательными, а какие должны быть по умолчанию выбраны программой setup.

Так называемый тег-файл (tagfile) находится в первом каталоге с категориями программ. В нём перечислены пакеты из этого отдельно взятого набора дисков и их статус. Статус может быть:

Таблица 18-3. Типы статусов в tag-файле

Тип	Значение
ADD	Пакет необходим для нормальной работы системы.
SKP	Пакет будет автоматически пропущен.
REC	В пакете нет необходимости, но рекомендуется его установить.
OPT	Пакет является необязательным.

Формат tag-файла прост:

```
имя_пакета: статус
```

По одному пакету на строку. Оригинальные тег-файлы для всех категорий программного обеспечения хранятся как tagfile.org. Поэтому, если с вашими файлами что-то случилось, вы всегда можете восстановить оригиналы.

Многие администраторы предпочитают писать свои тег-файлы и запускать инсталлятор в режиме “full”. Программа setup прочтёт тег-файлы и выполнит установку согласно их содержимому. Если вы используете REC или OPT, будет выведено диалоговое окно, спрашивающее пользователя, нужен ли ему или нет определённый пакет. Поэтому при написании тег-файлов для полностью автоматической установки рекомендуется использовать только статусы ADD и SKP.

Просто убедитесь, что ваши тег-файлы находятся в том же каталоге, что и оригинальные. Или же вы можете указать путь к своему тег-файлу, если у вас имеются таковые.

Глава 19.
ZipSlack

19.1. Что такое ZipSlack?

ZipSlack - это специальная версия Slackware Linux. Это предустановленная копия Slackware, готовая к запуску с раздела DOS или Windows. Это базовая установка, вы не получите в ней всего того, что есть в Slackware.

ZipSlack получил своё название за вид, в котором он распространяется - один большой файл .ZIP. Пользователи DOS и Windows скорее всего более знакомы с такими файлами. Это сжатые архивы. Архив ZipSlack содержит всё, что необходимо для получения полностью работоспособной системы.

Важно отметить, что ZipSlack значительно отличается от обычной установки. Даже несмотря на то, что они работают одинаково и содержат одни и те же программы, у них разные целевая аудитория и назначение. Ниже представлены различные преимущества и недостатки ZipSlack.

Последний момент: вам всегда следует обращать внимание на документацию, находящуюся в каталоге ZipSlack. В ней содержится последняя информация об установке, загрузке и общему использованию продукта.

Преимущества

- Не требуется повторная разметка вашего жёсткого диска.

- Удобный способ для изучения Slackware Linux без необходимости установки системы.

Недостатки

- Используется файловая система DOS - более медленная, чем родная файловая система Linux.
- Не будет работать с Windows NT.

19.2. Получение ZipSlack

Получить ZipSlack легко. Если вы приобрели официальный набор дисков Slackware Linux, тогда у вас уже есть ZipSlack. Просто найдите компакт-диск, на котором есть каталог `zipslack` и вставьте его в свой привод CD-ROM. Обычно он находится на третьем или четвёртом диске, однако всегда доверяйте названиям, которые фигурируют в этой документации, поскольку диск, на котором находится ZipSlack, может меняться.

Если вы хотите загрузить ZipSlack, вам следует сначала посетить нашу веб-страницу "Get Slack" для получения наиболее свежей информации о загрузке:

`http://www.slackware.com/getslack/`

ZipSlack является частью каждого релиза Slackware. Найдите нужный вам релиз и зайдите в соответствующий каталог на сервере FTP. Каталог с последним релизом можно найти по следующему адресу:

`ftp://ftp.slackware.com/pub/slackware/slackware/`

Вы найдёте ZipSlack в подкаталоге `/zipslack`. ZipSlack поставляется в виде одного большого файла `.ZIP` и виде набора файлов размером с дискету. Эти небольшие архивы находятся в каталоге `/zipslack/split`.

Не ограничивайтесь только файлами `.ZIP`. Вам также следует загрузить файлы с документацией и загрузочные образы, которые есть в каталоге.

Установка

После того, как вы загрузили необходимые компоненты, вам нужно распаковать файл `.ZIP`. Убедитесь в том, что вы используете 32-битный распаковщик. Размеры и имена файлов в архиве являются слишком длинными для 16-битного распаковщика. Примеры 32-битных распаковщиков: WinZip и PKZIP для Windows.

ZipSlack должен быть распакован непосредственно в корневой каталог диска (например, в `C:` или `D:`). Будет создан каталог `\LINUX`, в котором будет находиться система Slackware. Также вы найдёте в этом каталоге файлы, необходимые для загрузки системы.

После распаковки файлов на выбранном вами диске (с этого момента мы будем использовать диск `C:`) должен появиться каталог `\LINUX`.

19.3. Загрузка ZipSlack

Для загрузки ZipSlack существует несколько способов. Наиболее общим является использование файла LINUX.BAT для загрузки системы из DOS (или из DOS-режима в Windows 9x). Чтобы этот файл заработал, его нужно сначала отредактировать, чтобы он удовлетворял вашей системе.

Для начала откройте в своём любимом текстовом редакторе файл `C:\LINUX\LINUX.BAT`. В верхней его части вы найдёте большой комментарий. В нём рассказано о том, что вам нужно отредактировать в этом файле (а

также, что сделать, если вы загружаетесь с внешнего привода Zip). Не волнуйтесь, если вам непонятен параметр `root=`. Есть несколько готовых примеров, поэтому смело выберите и попробуйте один из них. Если он не работает, вы можете снова отредактировать файл: закомментируйте строку, выбранную вами ранее, и попробуйте другой пример.

После того, как вы раскомментировали нужную вам строку, удалив в её начале директиву "rem", сохраните файл и закройте редактор. Переведите свою машину в режим DOS.

Окно командной строки DOS в Windows 9x работать НЕ будет.

Наберите `C:\LINUX\LINUX.BAT`, чтобы загрузить систему. Если всё в порядке, вы должны будете увидеть приглашение для входа в систему.

Войдите в систему как `root`, без пароля. Вы наверняка захотите установить пароль для root'а, а также создать учётную запись для себя. Так что сейчас вы можете обратиться к другим разделам этой книги, в которых рассказано об общем использовании системы.

Если использование файла `LINUX.BAT` для загрузки системы не работает, прочтите файл `C:\LINUX\README.1ST`, в котором рассказано о других методах загрузки.

Глоссарий

Account (учётная запись, аккаунт)

Вся информация о пользователе, включающая имя пользователя, пароль, информацию finger, идентификатор пользователя (UID) и группы (GID) и домашний каталог. Под созданием учётной записи подразумевается добавление и определение пользователя.

Background (фоновый режим)

Любой процесс, не принимающий или не управляемый входными данными из терминала, считается выполняемым в фоновом режиме.

Boot disk (загрузочный диск)

Дискета, содержащая операционную систему (в нашем случае это ядро Linux), с которой может быть запущен компьютер.

Compilation (компиляция)

Преобразование исходного кода в бинарный код, который может "понять" машина.

Daemon (демон)

Программа, разработанная для работы в фоновом режиме (без вмешательства со стороны пользователя) и выполняющая определенную задачу (обычно какая-то служба).

Darkstar (Тёмная звезда)

Имя хоста по умолчанию в Slackware. Ваш компьютер будет называться darkstar, если вы не укажете другое имя.

Одна из рабочих машин Патрика Фолькердинга называется "Dark Star" в честь песни Grateful Dead.

Desktop Environment (настольная среда, среда рабочего стола)

Графический интерфейс пользователя (GUI), работающий поверх системы X Window и предоставляющий такие функциональные возможности как интегрированные приложения, однородное внешнее оформление программ и компонентов, управление файлами и окнами и др. Следущий этап в эволюции простого оконного менеджера.

Device driver (драйвер устройства)

Часть кода в ядре, напрямую управляющая частью оборудования системы.

Device node (узел устройства)

Специальный тип файла в файловой системе `/dev`, представляющий в операционной системе компонент аппаратного обеспечения.

DNS

Domain Name Service (служба доменных имён). Система, в которой устанавливаются соответствия между числовыми IP-адресами и текстовыми именами компьютеров, подключенных к сети.

Domain name (доменное имя, домен)

Имя компьютера в DNS без имени самого хоста.

Dot file (dot-файл)

В Linux, скрытые файлы, имена которых начинаются с точки ('.').

Dotted quad (четвёрка, разделённая точками)

Формат IP-адресов. Называется так потому, что адрес состоит из четырёх десятичных чисел (в диапазоне от 0 до 255), разделённых точками.

Dynamic loader (динамический загрузчик)

Программы, скомпилированные в Linux, обычно используют куски кода (функции) из внешних библиотек. При запуске таких программ эти библиотеки должны быть найдены, а требуемые библиотеки - загружены в память. Этим как раз и занимается динамический загрузчик.

Environment variable (переменная окружения)

Набор переменных в командном процессоре пользователя, на которые может ссылаться этот пользователь или программа в пределах данного шелла. Переменные окружения обычно используются для хранения настроек и параметров по умолчанию.

Epoch (эра)

Период в истории. "Эра" Unix отсчитывается от 00:00:00 UTC 1 января 1970 года. Эта дата считается начальной точкой отсчёта истории Unix и Unix-подобных операционных систем и всё остальное время отсчитывается относительно этой даты.

Filesystem (файловая система)

Представление хранимых данных, организованных виде "файлов" данных, распределённых по "каталогам". Файловая система является практически универсальной формой представления данных, хранимых на дисках (съёмных и несъёмных).

Foreground (приоритетный режим)

Программа, принимающая или управляемая входными данными из терминала, считается выполняемой в приоритетном режиме.

Framebuffer (видеобуфер, фреймбуфер)

Тип графического устройства. В Linux наиболее часто ссылаются на программную его реализацию, предоставляющую программам

интерфейс стандартного видеобуфера, скрывая от них драйверы оборудования. Этот уровень абстрагирования освобождает программы от необходимости работать напрямую с драйверами различных устройств.

FTP

File Transfer Protocol (протокол передачи файлов). FTP является очень популярным методом передачи данных между компьютерами.

Gateway (шлюз, гейт)

Компьютер, через который данные передаются из одной сети в другую.

GID

Group Identifier (идентификатор группы). GID - это уникальный номер, присвоенный группе пользователей.

Group (группа)

Пользователи в Unix входят в "группы", которые могут содержать много разных пользователей и используются для более общего управления доступом, чем контролирование отдельных пользователей.

GUI

Graphical User Interface (графический интерфейс пользователя).

Программный интерфейс, использующий отрисовку графических элементов: кнопки, полосы прокрутки, окна и т.п., а не просто текстовый ввод и вывод.

Home directory (домашний каталог)

"Домашний каталог" пользователя - это место, куда пользователь перемещается сразу после входа в систему. В своих домашних каталогах пользователи имеют полные права доступа и более менее свободные привилегии.

HOWTO

Документ, описывающий "how to" do something (как сделать что-либо), например, настроить файервол или управлять пользователями и группами. Существует большая коллекция этих документов, доступная в Проекте документации Linux (LDP).

HTTP

Hypertext Transfer Protocol (протокол передачи гипертекста). HTTP - это основной протокол, на котором работает "Всемирная паутина" (World Wide Web).

ICMP

Internet Control Message Protocol (протокол управляющих сообщений). Очень простой сетевой протокол, используемый в основном для "пингования".

Kernel (ядро)

Сердце операционной системы. Ядро обеспечивает основное управление процессами и интерфейсами системных устройств.

Kernel module (модуль ядра)

Кусок кода ядра (обычно это какой-либо драйвер), который может быть загружен или выгружен из памяти отдельно от основного тела ядра. Модули очень удобно использовать при обновлении драйверов или тестировании параметров ядра, потому что их можно загружать и выгружать без перезагрузки системы.

Library (библиотека)

Коллекция функций, которые могут совместно использоваться многими программами.

LILO

LInux LOader (загрузчик Linux). LILO - это наиболее распространённый менеджер загрузки Linux.

LOADLIN

LOADLIN - это программа, которая работает в MS DOS или Windows и загружает систему Linux. Наиболее часто используется в

компьютерах с несколькими операционными системами (включая Linux и DOS/Windows).

Man section (раздел man)

Страницы в стандартном онлайновом руководстве Unix ("man") для упрощения поиска разбиты на разделы. Все страницы, касающиеся программирования на C находятся в 3-м разделе, по системному администрированию - в 5-м и т.д.

MBR

Master Boot Record (главная загрузочная запись). Зарезервированное место на жёстком диске, в котром хранится информация о том, что нужно сделать при загрузке. В MBR может быть записан LILO или другой менеджер загрузки.

Motif

Популярный инструментарий, используемый во многих программах для X'ов.

MOTD

Message of the Day (фраза дня). MOTD (в Linux хранится в `/etc/motd`) - это текстовый файл, показываемый всем пользователям после входа в систему. Традиционно использовался системными администраторами в качестве "доски объявлений" для связи с другими пользователями.

Mount point (точка монтирования)

Пустой каталог в файловой системе, в который будет "примонтирована" другая файловая система.

Nameserver (сервер имён)

Информационный сервер DNS. Эти серверы преобразовывают имена DNS в числовые IP-адреса.

Network interface (сетевой интерфейс)

Виртуальное представление сетевого устройства, создаваемое ядром. Сетевые интерфейсы позволяют пользователям и программам общаться с сетевыми устройствами.

NFS

Network Filesystem (сетевая файловая система). NFS позволяет монтировать удалённые файловые систем так, как если бы они находились локально на вашем компьютере, и обеспечивает таким образом прозрачный метод для общего использования файлов.

Octal (восьмеричный)

Система исчисления по основанию 8 (с цифрами от 0 до7).

Pager (пейджер)

Программа для X'ов, позволяющая пользователю видеть и переключаться между несколькими "рабочими столами".

Partition (раздел)

Логический кусок жёсткого диска. На разделах создаются файловые системы.

PPP

Point-to-Point Protocol (протокол соединения "точка-точка"). PPP используется в основном для соединения через модем с провайдером услуг Интернета (ISP).

Process (процесс)

Выполняемая программа.

Root directory (корневой каталог)

Корневой каталог, обозначаемый как "/", является верхним элементом иерархии файловой системы. От него ответвляются все остальные каталоги, формируя "дерево файлов".

Root disk (корневой диск)

Диск (обычно несъёмный), на котором находится корневой каталог.

Routing table (таблица маршрутизации)

Набор данных ядра, используемых при “маршрутизации” данных по сети. В ней хранится такая информация: шлюз по умолчанию, какой сетевой интерфейс подключен к какой сети и т.п.

Runlevel (уровень запуска)

Состояние всей системы, определяемое init’ом.Уровень запуска 6 - это перезагрузка, уровень запуска 1 - это “однопользовательский режим”, уровень запуска 4 - это графический вход в систему и т.д. В системе Slackware существует 6 уровней запуска.

Secure shell (безопасный шелл)

Шифруемый (а значит защищённый) способ удалённого входа в систему. Существует много программ, использующих защищённое соединение; для работы необходим и клиент, и сервер.

Service (служба)

Предоставление одним “сервером” общей информации и/или данных многочисленным программам и компьютерам (“клиентам”). Примеры служб: HTTP, FTP, NFS и т.п.

Shadow password suite (система теневых паролей)

Система теневых паролей позволяет скрывать от пользователей

зашифрованные пароли, отставляя видимой для всех остальную информацию в файле `/etc/passwd`. Это помогает предотвратить попытки взлома паролей методом прямого подбора.

Shell (командный процессор, программная оболочка, шелл)

Командные процессоры предоставляют пользователю интерфейс командной строки. Если перед вами текстовая строка приглашения, значит вы находитесь в шелле.

Shell builtin (встроенные команды шелла)

Команды, встроенные в командный процессор; противопоставляются внешним программам. Например, в `bash` есть встроенная команда `cd`.

Signal (сигнал)

Программы в Unix могут общаться друг с другом посредством простых "сигналов", которым присвоены номера и которые обычно имеют особые назначения. `kill -l` выведет список всех доступных сигналов.

SLIP

Serial Line Interface Protocol (межсетевой протокол для последовательных интерфейсов). SLIP - это протокол, похожий на PPP в том, что он используется для соединения двух машин через последовательный интерфейс.

Software package (программный пакет)

Программа и связанные с нею файлы, заархивированные и сжатые в один файл вместе с необходимыми скриптами или информацией, помогающей при установке, обновлении и удалении этих файлов.

Software series (категории программного обеспечения)

Коллекция связанных программных пакетов в Slackware. Все пакеты с KDE находятся в категории “kde”, пакеты для работы с сетью - в “n” и т.п.

Source code (исходный код, исходные тексты)

Код (более или менее) удобный для восприятия человеком, на котором написано большинство программ. Исходный код компилируется в “двоичный” код.

Standard Error (stderr) (стандартный поток ошибок)

В Unix, стандартный поток вывода для ошибок. Программы выводят любые сообщения об ошибках на stderr, чтобы их можно было отделить от нормального вывода.

Standard Input (stdin) (стандартный ввод, стандартное устройство ввода)

В Unix, стандартный поток входных данных. Данные могут быть перенаправлены или отправлены по конвейеру на стандартный ввод программы из любого источника.

Standard Output (stdout) (стандартный вывод, стандартное устройство вывода)

В Unix, стандартный поток выходных данных. Обычно текст выводится программой на стандартный вывод (отделяемый от сообщений об ошибках, выводимых на stderr) и может быть перенаправлен или отправлен по конвейеру на стандартный ввод других программ или в файл.

Subnet (подсеть)

Диапазон IP-адресов, являющийся частью более широкого диапазона. Например, 192.168.1.0 - это подсеть 192.168.0.0 (здесь 0 - это маска, означающая "неопределёный адрес"); по сути это подсеть ".1".

Superblock (суперблок)

В Linux, разделы рассматриваются как наборы блоков. Размер блока составляет 512 байт. Суперблок - это первые 512 байт раздела.

Supplemental disk (дополнительный диск)

В Slackware, дискета, используемая во время установки и не содержащая ни ядра (которое находится на загрузочном диске), ни корневой файловой системы (которая находится на корневом диске). На ней находятся дополнительные необходимые файлы, такие как сетевые модули или поддержка PCMCIA.

Suspended process (приостановленный процесс)

Процесс, который был заморожен до тех пор, пока он не будет убит или возобновлён.

Swap space (пространство для свопинга)

Дисковое пространство, используемое ядром в качестве "виртуальной" оперативной памяти. Эта память медленнее, чем ОЗУ, но поскольку дисковое пространство является более дешёвым, своп обычно имеет больший объём. Пространство для свопинга полезно для ядра как хранилище для редко используемых данных и в качестве резерва, когда заканчивается физическая память.

Symbolic link (символическая ссылка, симлинк)

Специальный файл, который просто указывает на местонахождение другого файла. Символические ссылки используются для предотвращения дублирования данных, когда файл нужен в нескольких местах одновременно.

Tagfile (тег-файл)

Файл, используемый в Slackware программой `setup` во время установки. Описывает набор устанавливаемых пакетов.

Terminal (терминал)

Человеко-машинный интерфейс, состоящий по крайней мере из экрана (или виртуального экрана) и устройства ввода (в большинстве случаев

это клавиатура).

Toolkit, GUI (инструментарий графического интерфейса пользователя)

Инструментарий графического интерфейса пользователя - это набор библиотек, предоставляющих программисту код для отрисовки “виджетов” (widgets), таких как полосы прокрутки, чекбоксы и т.п., и создания графического интерфейса. Используемый программами инструментарий ГИП часто определяет их “внешнее оформление” (look and feel).

UID

User Identifier (идентификатор пользователя). Уникальный номер, идентифицирующий пользователя в системе. Большинством программ используют вместо имён пользователей их UID’ы, потому что с числами проще работать. Имена пользователей обычно используются только в тех случаях, когда пользователю необходимо увидеть результаты каких либо действий.

VESA

Video Electronics Standards Association (ассоциация по стандартизации в области видеотехники). Термин “VESA” часто используется для обозначения стандарта, определённого упомянутой выше ассоциацией. Практически все современные видеокарты являются VESA-совместимыми.

Virtual terminal (виртуальный терминал)

Использование программного обеспечения с целью эмуляции нескольких терминалов, используя при этом только один набор устройств ввода/вывода (клавиатура, монитор, мышь). Специальные комбинации клавиш переключают виртуальные терминалы на одном физическом терминале.

Window manager (оконный менеджер)

Программа для X, целью которой является предоставление графического интерфейса на основе простой прямоугольной отрисовки в системе X Window. Обычно оконные менеджеры обеспечивают в запущенных программах наличие строк заголовков, меню и т.п.

Working directory (рабочий каталог)

Каталог, в котором подразумевается нахождение программы во время своей работы.

Wrapper program (программа-упаковщик)

Программа, единственной целью которой является запуск других программ, в то же время меняя каким-либо образом их поведение путём изменения их окружения или фильтрации их входных данных.

X server (X-сервер)

Программа в системе X Window, взаимодействующая с графическим

оборудованием и управляющая запущенными программами для X.

X Window System (система X Window)

Ориентированная на работу в сети система графического интерфейса, используемая в основном на Unix-подобных системах, включая Linux.

Приложение A.
The GNU General Public License

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive

source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this

License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above

on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. How-

ever, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through

that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

12. NO WARRANTY

    BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS

NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty;

and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
   Copyright (C) <year>  <name of author>

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2 of the License, or
   (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA   02111-1307 USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year  name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

```
  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Предметный указатель

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z