Основы Slackware Linux.
Официальный учебник.

Дэвид Кэнтрелл,*Логэн Джонсон,†Крис Люменс,‡

перевод В. Толпекин

24 апреля 2001 г.

*David Cantrell
†Logan Johnson
‡Chris Lumens

Оглавление

Глава 1

Предисловие

1.1 Предисловие авторов

Операционная система Slackware Linux — это мощная платформа для компьютеров, основанных на Intel процессорах. Она разработана, как стабильная, безопасная и многофункциональная система. Как для применения в качестве high-end сервера, так и для использования в качестве мощной рабочей станции.

Эта книга предназначена для начинающих пользователей операционной системы Slackware Linux. Это не значит, что будут рассмотрены все детали использования дистрибутива. Но будут освещены основные возможности его использования и предоставлены основные рабочие знания для работы в системе.

По мере приобретения вами опыта работы в Slackware Linux, мы надеемся, вы найдёте эту книгу удобной для использования в качестве настольного справочника. Мы также надеемся, что вы предоставите её для прочтения вашим друзьям, когда они обнаружат для себя "насколько крута эта Slackware Linux, которой вы пользуетесь".

Скорее всего, эта книга не покажется вам занятным романом, но мы искренне постарались сделать её настолько занимательной, насколько возможно. Разумеется, мы так же надеемся, что вы сможете научиться чему-то и найдёте её полезной.

Итак, шоу начинается.

1.1.1 Соглашения, использованные в этой книге

Эта книга написана в SGML с использованием DocBook 4.0 DTD. Поэтому нами были использованы встроенные в DocBook элементы для ссылок на имена файлов, на команды и на содержание фалов.[1] Вам стоит ознакомиться с некоторыми из соглашений, принятых в этой книге, до того, как вы продолжите чтение.

Где бы мы не упоминали команду, которую вам необходимо выполнить, она будет выглядеть таким образом:

command

Вполне может случится так, что команда будет длиннее, чем ширина страницы. В этом случае команда будет разорвана с помощью бэкслэша, и продолжена с новой строки. Пример:

**ifconfig eth0 192.168.1.10 broadcast 192.168.1.255 **
netmask 255.255.255.0

Имена файлов и каталогов будут выглядеть следующим образом:

`filename`
`directory`

Экраны с выводом команд и содержанием конфигурационных файлов, так же используются в этой книге. Они будут выглядеть следующим образом:

```
command output
```

Когда мы будем приводить команды, которые вам следует выполнить, мы будем показывать их как запущенные из простой командной строки. Когда подразумевается, что команда должна быть выполнена простым поль-

[1] Перевод написан в TEX. Поэтому стили несколько отличаются от оригинальных. Прим. переводчика.

зователем, она будет показана приглашением командной строки со знаком доллара ($). Когда подразумевается, что команда должна быть выполнена root пользователем, приглашение командной строки будет показано со знаком хэш (#).

1.2 Предисловие переводчика

Однажды я познакомился с Линукс. Первый дистрибутив, попавший мне в руки был RedHat 6.02 где-то в 1999 году. В Одессе есть такое замечательное место — книжный рынок. Так вот именно там я приобрёл двойной диск с гордым названием Linux Office 2000. Что сейчас для меня самого звучит очень загадочно. Фактически, первый диск был копией упомянутой версии RedHat, а на втором диске было море документации на русском языке, StarOffice 5.1, несколько не легковесных игр и толпа всяких утилит и программ.

При попытке установить это счастье я мгновенно столкнулся с проблемой, которую (по привычке полученной от работы с MS Windows) пытался решить экспериментально. Суть на самом деле была в том, что не запускался графический установщик. Моя тогдашняя видео-карта не поддерживалась ещё XFree86. Это была ATI Rage Mobility. Да и монитор я был не в состоянии правильно настроить, так как тогда ещё не знал всю специфику TFT мониторов.

Последующий год я потратил на то, что искал "свой" дистрибутив. И очень рад, что очень скоро появился на свет Madrake 7.0 RE. Я своими глазами и на своём компьютере увидел, что Линукс может быть удобной и мощной русскоязычной рабочей станцией. За это хочется сказать огромное спасибо разработчикам сего прекрасного дистрибутива. Который я бы рекомендовал начинающим пользователям. А потом, кто знает, может он вам настолько понравиться, что вы продолжите работать с ним и далее.

Ещё через некоторое время я прочитал где-то в новостях, кажется на unixware.ru, что "вышла новая версия мощной 32 разрядной операционной системы Slackware Linux 7.1...". Так как у меня была уйма свободного времени, я взялся попробовать это чудо. После красивого Mandrake-овского графического установщика я попал в программу в стиле старой DOS-овской программы установки, как показалось мне на первый взгляд. Практически никакой автоматики. В общем, поиграл я немного с этим счастьем, и в очень расстроенных чувствах понял, что это не для меня. Всё показалось мне очень сложным. И основной недостаток оказался в том, что я не мог ничего распечатать. А знакомая мне по RedHat и Mandrake утилита printool просто напросто отсутствовала. В общем, пришлось мне расстаться с этим счастьем.

Потом было ещё много всего — Peanut, Turbo... Затем по определённым причинам у меня не стало моего Compaq-a Pressario. Ещё через время дома появился 386-DX40/8Mb/512kb/4-x speed CD-ROM и аж 540Мб HD. По работе мне надо было срочно писать доклад о проделанной работе. Так что пыхтел, как паровоз. Два дня думал, что на такой быстрый и современный компьютер можно поставить. Ответ пришёл из Zip-howto. Так как CD я тогда писать не мог по техническим причинам, пришлось воспользоваться внешним Zip (ppa). В упомянутом документе речь шла именно о Slackware. Работа была сделана во время, и компьютер отправился на свалку. Но я уже загорелся и был очарован производительностью, получаемой при работе в консоли. Следующим моим шагом в изучении Линукс оказалось прочтение книги "Slackware Linux Essentials", которую я наглым образом (т.е. бесплатно) скачал с интернета. Потратил один вечер, чтобы собрать всё в кучу. Немного форматирования. Прочитал интересный документ Cyrillic-HowTO

в разделе документация на сайте lug.irk.ru. Где и ознакомился с программой a2ps для преобразования текста в .ps. Немного колдовства, прикрутил к ней русские шрифты и вуаля. Книга попала ко мне в руки.

1.2.1 Координаты перевода

Домашняя страничка перевода:

http://sle.how-to.ru

Хостинг любезно предоставлен Алексеем Чегляковым.

Если у вас есть какие-то пожелания или советы по поводу перевода, пишите по адресу:

dolphin77@mail.od.ua

Глава 2

Введение

2.1 Введение в Slackware Linux

2.1.1 Что такое Linux?

Linux начался с Линуса Торвальдса[1] в 1991 году, как персональный проект. Линус пытался найти способ запуска Unix-подобной операционной системы без существенных материальных затрат. В дополнение к этому он хотел изучить подробности ввода и вывода 386-го процессора. То что получилось он выложил для бесплатного в терминах GNU General Public License (см. раздел 2.1.3 на стр.12) для использования с возможностью модификации всем желающим.

Сегодня Linux вырос в одного из основных игроков на рынке операционных систем. Он портирован на большое число различных процессорных архитектур, включая Compaq-овский Alpha, Sun-овские SPARC и UltraSPARC а так же на Motorola PowerPC чипы (например, на компьютеры Aple Macintosh и IBM RS/6000). Linux сейчас разрабатывается сотнями, если не тысячами программистов со всего мира. В нём работают такие программы, как Sendmail, Apache и BIND, которые являются наиболее распространёнными сер верными программами в Интернет.

На самом деле, термин "linux" относится только к ядру — центру операционной системы. Ядро ответственно за управление процессором, памятью, жёсткими дисками и периферийными устройствами. Это на самом деле всё, делает что Linux. Он контролирует работу компьютера, и следит за тем, чтобы все программы работали. Все те программы, которые делают Linux пригодным к использованию, разработаны независимыми группами. Ядро и программы связываются вместе различными компаниями и людьми, чтобы организовать операционную систему. Мы называем это дистрибутивом Linux.

2.1.2 Что такое Slackware?

Slackware был первым дистрибутивом, получившим широкое применение. Он был начат Патриком Волькэрдиногм[2] в конце 1992 года. Патрик ознакомился с Linux, когда искал недорогой интерпретатор языка LISP для своего проекта. В то время существовало всего несколько дистрибутивов, и Патрик выбрал дистрибутив от Soft Landing Systems (SLS Linux).

Тем не менее, у SLS были свои проблемы и Патрик начал исправлять мелкие ошибки, по мере их обнаружения. В конечном итоге, он решил объединить все исправления в свой собственный дистрибутив, для себя и своих друзей. Этот частный дистрибутив очень быстро обрёл популярность и Патрик сделал его доступным для общественности под именем Slackware.

Патрик так же добавил новые черты дистрибутиву. Такие как программу установки с дружеским интерфейсом, основанную на системах меню и ввёл концепцию менеджмента пакетов программ. Что позволило пользователям легко добавлять, удалять или обновлять пакеты программ в их системе.

[1] Linus Torvalds
[2] Patrick Volkerding

2.1.3 О программном обеспечении с открытым исходным текстом и о свободном программном обеспечении

Внутри Linux сообщества существует два идеологических направления разработки программ. Направление Свободных Программ (Free Software), к которому мы скоро вернёмся, движется в направлении создания программ, свободных от ограничений интеллектуальной собственности, которые по их мнению являются препятствием технического развития и работают против общества. Движение Программ с Открытым Исходным Текстом (Open Source) работает в направлении достижения большинства тех же целей, но принимает более "прагматическое" приближение к их выполнению и предпочитает положить за основу экономические и технические достоинства предоставления публичной общественности исходных текстов программ в бесплатной форме, в отличие от моральных и этических принципов, которые управляют первым из приведённых движений.

Лидеры Free Software, это Free Software Foundation(FSF) — организация, существующая за счёт фондов для GNU проекта. Free Software это больше идеология. Часто используемое выражение "свобода слова, а не бесплатное пиво". В своей основе, свободное программное обеспечение — это попытка гарантировать определённые права, как авторам, так и пользователям. Эта свобода включает в себя свободу в использовании программы для любых целей, свободу в изучении и изменении исходного текста, свободу предоставления всем желающим изменённого вами текста. Для того, чтобы обеспечить эти степени свободы, была создана GNU General Public License (GPL). GPL, утверждает, что все, кто предоставляет скомпилированную программу, лицензированную в соответствии с GPL, обязаны предоставить её исходный текст и любой желающий может произвести изменения в программе в том случае, если эти изменения также доступны в форме исходного текста. Это гарантирует, что если программа однажды была "открыта" обществу, она больше не может быть "закрыта", за исключением случая, когда авторы каждой порции кода (даже авторы изменений) дадут своё согласие. Большинство программ в Linux лицензированы GPL.

Важно отметить, что GPL не говорит ничего о цене программ. Как это ни странно звучит, вы можете получать деньги за свободное программное обеспечение. Здесь "свободное" относится к приобретаемым вами свободам действий с исходным текстом программ, а не к цене, которую вы платите за программы. (Тем не менее, если кто-то продал, или даже просто дал вам программу, скомпилированную под лицензией GPL, он обязан предоставить её исходный текст.)

В самом начале существования Open Source движения, Open Source Initiative (OSI) являлась организацией исключительно существующей для поддержки программ с открытым исходным кодом. Т.е. программы, которые имеют как доступный исходный текст, так и выполняемую версию. Они не предлагают специальной лицензии, но вместо этого они поддерживают различные типы лицензий для открытого исходного кода.

Идея, стоящая за OSI — привлечь как можно больше компаний в свои ряды, позволяя им писать свои собственные лицензии для программ с открытым исходным текстом и сертифицировать эти лицензии в OSI. Многие компании хотят предоставить исходные коды, но не хотят при этом пользо-

ваться GPL. Ввиду того, что не могут изменить GPL кардинальным образом, им предоставляется возможность получить их собственную лицензию и сертифицировать её в вышеупомянутой организации.

Несмотря на то, что Free Software Foundation и Open Source Initiative движутся в одном и том же направлении, это не одно и то же. FSF использует определённую лицензию и предоставляет программы под этой лицензией. OSI ищет поддержки для всех лицензий с открытым кодом, включая лицензию от FSF. Основные аргументы на пути свободного предоставления исходных текстов иногда разделяют оба движения, но сам факт, что две идеологически различных группы работают на пути достижения одних и тех же целей, вызывает доверие к попыткам каждой из них.

2.2 Получение поддержки и помощи

Довольно часто случается так, что вам необходима инструкция по использованию определённой команды, установке определённой программы, или по настройке оборудования. К счастью для вас, существует множество способов получения такой помощи. Если вы установили пакеты программ из F — раздела, то обширный набор документации уже имеется на вашем компьютере. Программы сами по себе могут поставляться с файлами документации содержащими описание их различных опций, файлов настройки и использования. В конце концов, вы всегда можете обратиться на официальный вэб-сайт Slackware для получения помощи.

2.2.1 Методы получения справки из системы

man

man (сокращение от "manual" – руководство) это традиционный способ получения справки в ОС Unix и Linux. Файлы, отформатированные особым образом — "man pages", содержащие описание большинства команд и поставляемые вместе с программами. Выполнение команды **man какаятокоманда** выведет на экран страницу man для команды или программы **какаятокоманда**.

Ввиду того, что существует огромное множество man страниц, они разбиты на группы. Это разбиение было произведено очень давно. Поэтому практически везде, где встречается ссылка на эти страницы, будет указан номер группы, к которой относится эта конкретная man страница. Например, вы можете увидеть **man**(1). Это показывает вам, что команда **man** описана в разделе 1 — команды пользователя (user commands); вы можете указать, что вы хотите обратиться к странице man из раздела 1 для команды "man", воспользовавшись командой **man 1 man**. Указание номера раздела, в котором необходимо искать документацию весьма полезно в том случае, когда производится поиск по команде с различными функциями, но идентичным именем.

Раздел	Содержание
Раздел 1	команды пользователя
Раздел 2	системные вызовы
Раздел 3	вызовы библиотеки C
Раздел 4	устройства
Раздел 5	форматы файлов и протоколов
Раздел 6	игры
Раздел 7	соглашения, макропакеты и т.д.
Раздел 8	администрирование системы

В дополнение к **man**(1), есть команды **whatis**(1) и **apropos**(1), которые изначально были разработаны для облегчения поиска информации в системе **man**. **whatis** даёт очень краткое описание системных команд, что-то вроде короткого карманного справочника по командам. **apropos** применяется для поиска **man** страницы, содержащей указанное ключевое слово (keyword).

Смотрите их странички man для получения подробностей:)

Каталог `/usr/doc`

Исходные тексты большинства программных пакетов, которые мы включили в дистрибутив поставлялись с какого-то рода документацией. `README` файлы, инструкции по использованию, файлы лицензий... документация любого рода, поставляемая с исходниками программ входящих в вашу систему и установленных на неё в последствии, находится в каталоге `/usr/doc`.

Если информации, предоставленной в man вам не достаточно, то ваша следующая остановка — каталог `/usr/doc`.

Документы HOWTO и mini-HOWTO

HOWTO - аббревиатура от англ. how to — как сделать. Эти документы описывают, как делать что-либо. Если вы установили пакет с коллекцией HOWTO, то вы можете найти их в каталоге `/usr/doc/Linux-HOWTOs`, а mini-HOWTO — в `/usr/doc/Linux-mini-HOWTOs`.

В этом же пакете вы так же можете найти коллекцию документов FAQ (Frequently Asked Questions - Часто Задаваемые Вопросы. Иногда можно найти русский вариант аббревиатуры ЧаВО) — это коллекция вопросов и ответов на них.

Эти файлы очень полезны в случае, если вы не вполне уверенны, как сделать какое-то определённое действие. Потрясающий спектр тем освещён в зачастую поражающих подробностях.

2.2.2 Интерактивная помощь

В дополнение к документации, поставляемой в ОС Slackware Linux, есть несколько способов получения помощи в режиме он-лайн.

Вэб сайт и форум

www.slackware.com

Официальный сайт Slackware Linux содержит достаточно много информации о системе. Вы можете найти там вводную информацию по системе, руководство по установке, список Часто Задаваемых Вопросов (FAQ) и много другой полезной информации, как для новичков, так зачастую и для опытных пользователей.

Так же вы можете найти там форум. Раздел, где пользователи могут обмениваться опытом работы в Slackware и помогать друг другу с решением вопросов и проблем. Этот способ получения помощи, как показывает опыт, является популярным и в то же время очень эффективным и, вероятно, это будет ваша первая остановка на пути получения информации. (Ваши сообщения открываются широкой области пользователей а это означает, что шансы на получение быстрого решения проблемы достаточно высоки. Зачастую вопрос возникший у вас уже встречался кем-то, и он сможет оперативно поделиться своим опытом решения проблемы). Пожалуйста, перед тем, как задавать вопрос, поищите его в форуме, возможно ответ уже есть там и ждёт вас.

Поддержка через электронную почту (e-mail)

Все, кто приобрели официальный пакет CD, имеют право на получение бесплатной поддержки по установке, через e-mail. Мы из старой школы. Мы стараемся, насколько это в наших силах, помочь всем, кто посылает нам e-mail с вопросами о помощи. Пожалуйста, перед тем как отправлять нам письмо, проверьте вашу документацию и вэб сайт (в особенности раздел FAQ и форум). Зачастую таким способом вы получите более быстрое решение, чем спрашивая по e-mail. И чем меньше писем нам приходит, тем быстрее мы сможем помочь всем.

Адрес электронной почты для получения технической поддержки следующий: <support@slackware.com> Другие адреса электронной почты и контактная информация указанны на нашем вэб сайте.

Глава 3

Установка

3.1 Установка

Перед тем, как приступить к использованию Slackware Linux, вам необходимо получить и установить её. Получение системы — процесс довольно простой. Вы можете купить её, а можете загрузить бесплатную версию из интернет. В том случае, если вы обладаете элементарными знаниями о вашем компьютере и желаете изучить что-то новое, установить систему так же не составит большого труда. Программа установки выполняется шаг-за-шагом. Поэтому вы очень быстро оживите и запустите свою систему.

3.1.1 Получение Slackware

Официальный диск и набор в коробке

Официальный набор Slackware Linux CD поставляется от Slackware, Inc. Покупая официальный набор, вы получаете удобство установки с CD, поддержку по установке через электронную почту, 30-страничный буклет по установке и даже больше. Набор Slackware Linux включает в себя набор CD и официальное руководство пользователя. Покупка официального набора — лучший способ помочь проекту Slackware Linux.

Метод	**Информация**
телефон	1-800-786-9907
вэб-сайт	http://www.slackware.com
электронная почта	<orders@slackware.com>
обычная почта	4041 Pike Lane, Suite F Concord, CA 4520-1207

Через интернет

Slackware Linux так же доступен бесплатно в сети интернет. Вы можете спросить по email о поддержке, но более высокий приоритет всегда отдаётся тем, кто приобрёл официальный набор.

Официальный сайт проекта Slackware Linux находится по адресу:
http://www.slackware.com/
Основной FTP расположен здесь:
ftp://ftp.slackware.com/pub/slackware/

3.1.2 Требования к системе

Для обычной установки Slackware необходимо, как минимум:

Оборудование	**Требования**
Процессор	386
ОЗУ	16Мб
Место на диске	500Мб
Флоппи диск	1.44Мб

Если у вас есть загрузочный CD, то вам скорее всего не понадобится флоппи диск. Разумеется, если вы собираетесь устанавливать систему с CD, вам понадобится привод CD. Сетевая плата необходима при установке с использованием NFS. Смотрите раздел 4.2.6 на стр. 54 для дополнительной информации.

Требования к свободному месту на диске, указанные в таблице весьма приблизительны. Рекомендованные 500Мб обычно достаточны, но для полной установки вам понадобится порядка одного гигабайта дискового пространства. Большинству пользователей нет необходимости выполнять полную установку. Между прочим, многие работают с Slackware всего на 100Мб.

Slackware может быть установлен на систему с меньшим количеством ОЗУ и меньшим жёстким диском, но это потребует немного колдовства. Если вы заинтересованны в такой установке, загляните в `LOWMEM.TXT` файл дистрибутива, для получения основных инструкций.

Разделы программ

С целью упрощения, Slackware исторически разделён на разделы программ. Когда-то названных "дисковыми разделами", потому что они были ориентированы на установку с флоппи дисков. Сейчас дистрибутив разбит на разделы в основном с целью структурирования программ, поставляемых с дистрибутивом. Сегодня установка с флоппи дисков всё ещё возможна для программ из A и для большинства программ из N разделов (смотри таблицу).

Раздел	**Содержание**
A	Основная система. Содержит необходимый минимум системных программ, текстовый редактор и основные коммуникационные программы.
AP	Различные приложения, которые работают без X Window системы.
D	Инструменты для разработки программ. Компиляторы, дэбаггеры, интерпретаторы и man странички для них.
DES	Содержит GNU libc crypt() функцию.
E	GNU emacs.
F	FAQи, HOWTO, и другая дополнительная документация.
GTK	Рабочая среда GNOME , GTK widget библиотека, и GIMP.
K	Исходный код ядра Linux.
KDE	Рабочая среда KDE. Qt библиотека, необходимая для KDE, так же находится здесь.
N	Сетевые программы. Демоны, почтовые программы, telnet, программы чтения новостей, и так далее.
T	Система форматирования документов teTeX.
TCL	Tool Command Language. Tk, TclX, и TkDesk.
X	Основа X Window Системы.
XAP	Приложения X которые не являются частью основной окружающей среды рабочего стола. (Например, Ghostscript и Netscape).
XD	Разработка программ для X11. Библиотеки, и т.д.
XV	Библиотеки XView, OpenLook Virtual и Non-Virtual Оконные Менеджеры и различные другие приложения XView.
Y	Игры

Методы установки

Флоппи Ранее представлялось возможным установить весь Slackware Linux с флоппи дисков. Возросший с тех пор объём программных пакетов (точнее, некоторых из них) изменил эту идеологию. Сегодня лишь два программных раздела все ещё можно установить с флоппи. А серия может быть полностью установлена с флоппи и так же, большинство программ из серии N. Это позволит вам получить основную систему, которая впоследствии может быть использована для установки остальных программ из дистрибутива через сеть.

Следует отметить, что флоппи диски всё ещё могут потребоваться вам при установке с CD-ROMa, если у вас нет загрузочного CD, а так же для установки через NFS.

CDROM Если у вас есть загрузочный CD, из официального набора, (см. раздел 3.1.1 на стр. 17), то процесс установки будет несколько проще для вас. Если нет, то вам понадобится загрузиться с флоппи диска. Также, если у вас установлено специфическое оборудование, которое приводит к проблемам при использовании ядра с загрузочного CD, вам может понадобится использовать специализированные флоппи диски.

Смотрите подразделы "Загрузочный диск" и "Дополнительный диск", для получения информации по выбору и созданию флоппи дисков для загрузки, если возникла такая необходимость.

NFS NFS (Network File System — сетевая файловая система) — способ предоставления файловых систем удалённым машинам. Установка через NFS позволяет вам установить Slackware с другого компьютера в вашей сети. Машина, с которой вы хотите устанавливать систему, должна быть настроена для экспортирование дерева каталогов дистрибутива Slackware, машине на которую вы собираетесь устанавливать систему. Разумеется, для этого вам необходимы некоторые знания NFS. Вы можете обратиться к разделу 4.2.6, на стр. 54 этого руководства.

Теоретически, возможно использование NFS при установке такими методами, как PLIP (через параллельный порт), SLIP и PPP (надеемся, не через модем). Тем не менее, мы рекомендуем использовать сетевые карты, если есть такая возможность. В конце концов, установка системы через порт принтера это очень и очень медленный процесс.

Загрузочный диск Загрузочный диск, это диск с которого вы загружаетесь, для начала установки. Он содержит образ сжатого ядра, которое используется для управления оборудованием в процессе установки. По сему он смертельно необходим для установки (кроме случая загрузки с CD, как это обсуждалось в подразделе CD-ROM). Образы загрузочных дисков расположены в каталоге `bootdsks.144/` дерева каталогов дистрибутива.

Существует более 60 вариантов загрузочных дисков. Полный их список с описанием каждого, вы можете найти здесь: `bootdsds.144/WHICH.ONE`. Тем не менее, большинство пользователей могут воспользоваться `bare.i` (для IDE устройств), или `scsi.s` (для SCSI устройств) образами загрузочных дисков.

Смотри подраздел "Создание дисков" для получения инструкций по созданию дисков из образов.

После загрузки ядра, вас попросят вставить root диск.

Root диск root диск содержит программу установки и файловую систему, которая используется в процессе установки. Он так же необходим. Образы root дисков расположены в директории `rootdsks` дерева каталогов дистрибутива.

К счастью, их гораздо меньше, чем загрузочных дисков. На самом деле, их всего три.

`color.gz` наиболее используемый. Он цветной и это красиво.

`text.gz` то же, что и `color.gz`, только не цветной. Можете проверить.[1]

`umsdos.gz` используется для установки системы на FAT (Windows) раздел. Что рекомендуется делать лишь с экспериментальными целями. Для тех, кому хочется попробовать Slackware на Windows разделе, мы настоятельно рекомендуем воспользоваться ZipSlack или BigSlack, вариантами дистрибутива.

Дополнительный диск Дополнительный диск необходим в том случае, если вы производите установку через NFS или установку с использованием PCMCIA устройств. Дополнительные диски находятся в каталоге `rootdsks`, и называются `network.dsk` и `pcmcia.dsk`.

root диск даст вам инструкции по поводу использования дополнительным диском, после загрузки.

Создание дисков После того, как вы выбрали образ загрузочного диска, вам необходимо разместить его на флоппи. В зависимости от операционной системы, используемой вами, для создания загрузочного диска, процесс немного различен. Если вы используете Linux (или другую Unix-подобную систему), вам необходимо воспользоваться командой **dd**(1). Предположим, имя выбранного образа `hejaz.dsk` и ваш дисковод для флоппи дисков `/dev/fd0`, в этом случае вам необходимо воспользоваться командой:

 # dd if=hejaz.dsk of=/dev/fd0

Если вы пользуетесь Microsoft OS, то вам необходимо воспользоваться программой `RAWRITE.EXE`, которая есть в дистрибутиве в том же каталоге, где и образы дисков. Опять, предположим, что `hejaz.dsk` это выбранный вами образ диска, а ваш дисковод называется `A:`, откройте командную строку DOS и выполните следующую команду:

c:\ rawrite a: hejaz.dsk

Разбиение диска После загрузки выбранным вами способом, будет необходимо пере разметить жёсткий диск. Надо создать раздел, куда вы установите Slackware. Как минимум, мы рекомендуем создать два раздела; один для корневой файловой системы (\) и один для подкачки (swap space).

[1] Это не совсем так, однажды я испытал его. Программа установки не имеет интерфейса меню. Прим. переводчика.

После того, как root диск загрузится, он предоставит вам приглашение для входа в систему (login). Войдите как root (вам не нужен пароль для этого). В приглашении командной строки наберите, либо **cfdisk**(8), либо **fdisk**(8). Программа **cfdisk**(8) обладает более дружественным интерфейсом, чем программа **fdisk**, но не содержит некоторых команд последней. Ниже мы вкратце рассмотрим программу **fdisk**.

Начнём с запуска **fdisk** для выбранного вами жёсткого диска. В Linux жёстким дискам не присваиваются буквенные обозначения, но каждому диску соответствует определённый файл. Первый IDE диск (primary master) называется `/dev/hda`; primary slave — `/dev/hdb`, и так далее. SCSI диски определяются по такой же системе, но в виде `/dev/sdX`. Вам необходимо запустить **fdisk** в применении к выбранному жёсткому диску:

fdisk /dev/hda

Как и все хорошие Unix программы, **fdisk** выдаст вам приглашение командной строки (а вы думали, что получите меню, не так ли?). Первое, что вам необходимо сделать, это проверить уже существующие разделы. Мы сделаем это, напечатав **p** в командной строке программы:

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

Программа выведет на экран всю информацию о существующих метках и разделах. Большинство людей выбирают свободный диск для установки системы, и удаляют все существующие на нём метки и разделы.

ВНИМАНИЕ: ОЧЕНЬ ВАЖНО СОЗДАТЬ РЕЗЕРВНЫЕ КОПИИ ВСЕЙ ИНФОРМАЦИИ, КОТОРУЮ ВЫ ЖЕЛАЕТЕ СОХРАНИТЬ ДО ТОГО, КАК УДАЛИТЬ РАЗДЕЛ НА КОТОРОМ ОНА НАХОДИТСЯ.

Не существует простого способа восстановления данных после удаления раздела жёсткого диска, так что создайте резервную копию до того, как играть с разделами.

В таблице разделов вы увидите номер раздела, его размер и тип. Вы найдёте больше информации на экране, но пока она вам не понадобится. Мы собираемся удалить все разделы на этом диске, и создать там разделы Linux. Для удаления раздела используется команда **d**:

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

Необходимо повторить процедуру для всех разделов, которые вы желаете удалить. После удаления старых разделов, мы готовы создать новые для Linux. Мы решили создать один раздел для корневой файловой системы и один для swap раздела. Следует отметить, что схемы разбиения диска для Unix систем являются предметом многочисленных споров, и большинство пользователей расскажет вам лучший способ сделать это. Наш совет, для начала — два раздела. Один для корневой файловой системы и второй — раздел подкачки (swap). Со временем вы узнаете больше о схеме разбиения жёсткого диска, подходящей для вашей системы.

Теперь мы создадим разделы, используя команду **n**:

```
Command (m for help):n
Command action
   e extended
   p primary partition (1-4)
p
Partition number (1-4):
First cylinder (0-1060, default 0):0
Last cylinder or +size or +sizeM or +sizeK (0-1060, default 1060):+64M
```

Вам следует убедиться, что вы создали раздел. Первый раздел будет разделом подкачки. Мы указали **fdisk** сделать первый раздел основным (primary). Он начинается с нулевого цилиндра, а для последнего цилиндра мы написали **+64M**. Таким образом мы получили раздел размером в 64 мегабайта для подкачки. (Размер необходимый для раздела подкачки, зависит от того, сколько ОЗУ есть у вас в системе. Существует соглашение, что размер раздела подкачки должен быть в два раза больше, чем объём ОЗУ.) Затем мы определим раздел номер 2, начинающийся первым доступным цилиндром и заканчивающийся в самом конце жёсткого 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
```

Ну вот, разбиение почти завершено. Теперь необходимо изменить тип первого раздела на тип 82 (Linux swap). Наберите **t**, чтобы изменить тип, выберите первый раздел, и наберите **82**. До того как записывать изменения на диск, вам следует ещё раз посмотреть на таблицу разделов. Воспользуйтесь для этого командой **p**. Если всё хорошо, то наберите **w**, чтобы сохранить изменения на диск и выйти из **fdisk**[2].

Программа установки

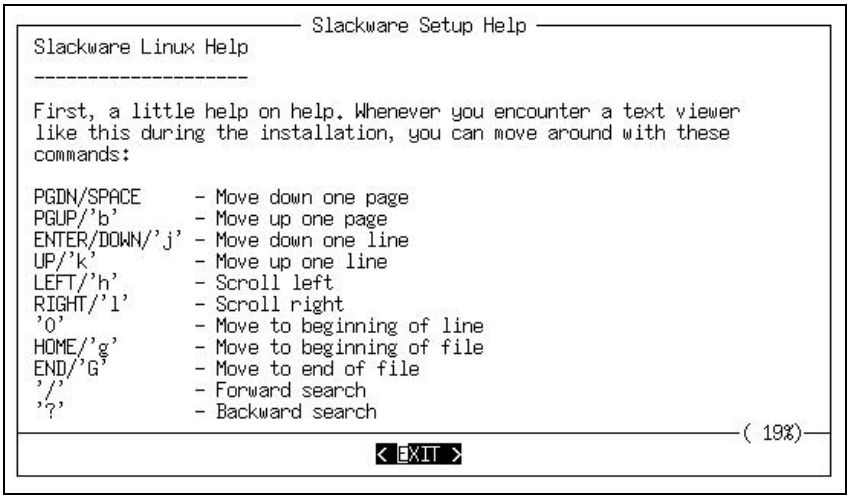
После того, как вы создали разделы, вы готовы к установке Slackware. Следующий шаг в процессе установки — это запуск программы **setup**(8). Чтобы запустить её, просто наберите **setup** в приглашении командной строки оболочки. **setup** — меню управляемая система для фактической установки Slackware пакетов и настройки вашей системы.

[2] или **q** для выхода без сохранения изменений. Прим. переводчика.

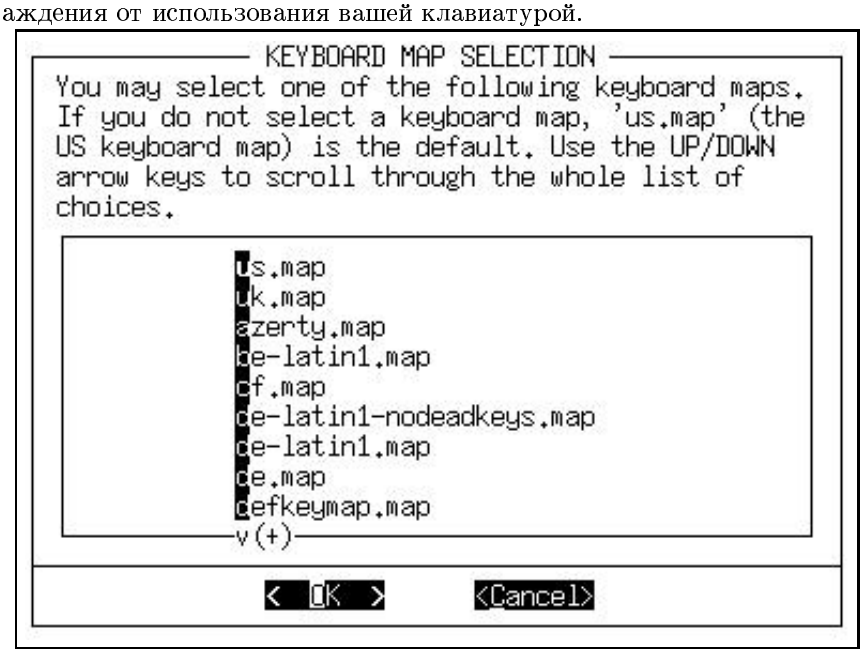
Процесс установки происходит по следующему сценарию: Вы проходите через каждую опцию программы установки в том порядке, в котором они перечислены. (Конечно, вы можете проделать всё это в практически любом желаемом порядке, но шансы на то, что это не сработает достаточно высоки.) Выбор пункта меню производится при помощи кнопок-стрелок вверх и вниз, а выбор кнопок "Okay" или "Cancel" производится при помощи стрелок вправо и влево. В добавок к этому, каждому пункту меню соответствует определённая кнопка, которая подсвечена на экране в имени опции. Опции-флаги или, иначе говоря, переключатели (те которые отмечены [X]) помечаются при помощи клавиши пробел.

Разумеется, всё это вы можете найти в разделе "help" программы установки, но мы пользуемся принципом — пользователю всё самое лучшее за его деньги.

HELP Если вы устанавливаете Slackware впервые, вы скорее всего захотите заглянуть в этот раздел. Там дано описание каждого раздела **setup** (очень похожее на то, что мы пишем сейчас, но менее предвзятое) и инструкции по навигации через процесс установки.



KEYMAP Если вам необходима раскладка клавиатуры, отличная от United States "qwerty", вы возможно захотите заглянуть в этот раздел[3]. Там вы найдёте большой список альтернативных раскладок, для получения наслаждения от использования вашей клавиатурой.



ADDSWAP Если вы создали раздел подкачки swap (см. раздел "Разбиение жёсткого диска" на стр. 20), то этот пункт поможет вам активизировать его. Он автоматически обнаружит разделы подкачки, выведет на экран информацию о существующих разделах подкачки, позволяя вам выбрать один, который будет отформатирован и включён.

TARGET Пункт target (цель) определяет, какие из других (не swap) разделов должны быть отформатированы и подключены к точкам монтирования вашей файловой системы. На экран выводится список разделов вашего жёсткого диска. Для каждого раздела вам будет предоставлена возможность отформатировать его (а так же, проверить на наличие bad-блоков) и список для выбора размера инод. Для обычного использования, значение размера инод можно выбрать предлагаемое по умолчанию.

Первая опция в пункте target — выбор раздела, на который установить корневую (\) файловую систему. После этого вы сможете подключить другие разделы к файловой системе, на ваше усмотрение. (Например, вы можете захотеть, чтобы ваш третий раздел, скажем `/dev/hda3`, был каталогом домашних файловых систем пользователей. Это лишь пример; подключайте ваши разделы, как считаете нужным.)

SOURCE Пункт source (источник) позволяет вам выбрать, носитель информации, с которого вы будете устанавливать Slackware. На сегодняшний

[3]этот шаг можно пропустить. В Дополнении 1 будет описан процесс русификации дистрибутива. В том числе и настройка клавиатуры. Прим. переводчика.

день есть четыре варианта решения этого вопроса. Флоппи, CD-ROM, NFS или заранее под-монтированный каталог.

Вариант установки с флоппи требует большого количества дискет. Этот вариант требует много времени и терпения, но он возможен[4]. Помните, что вам необходимо создать флоппи до того, как вы запустите программу установки.

Выбор пункта CD-ROM активизирует установку с CD. Этот пункт предложит вам на выбор, либо автоматический поиск вашего CD-ROM привода, либо выбор устройства, соответствующего приводу из списка. Убедитесь в том, что Slackware CD вставлен в привод компакт дисков, до того, как начнёте сканирование. После того, как программа найдёт CD-ROM вы должны будете выбрать, какой из вариантов установки вы желаете произвести: "slakware" или "slaktest". Обычный выбор — это slakware, который является стандартной установкой. slaktest опция устанавливает минимальный набор программ на жёсткий диск, и оставляет большинство программ на CD. Вам понадобится "live" CD из официального набора для того, чтобы воспользоваться этим вариантом установки.

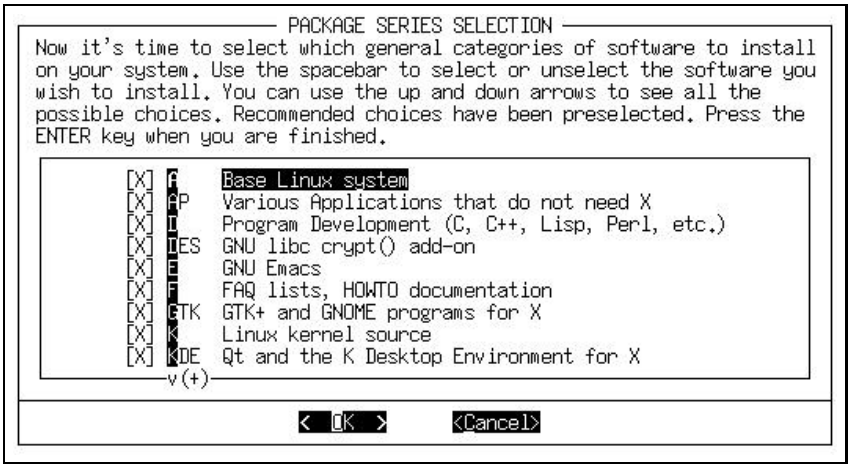
Вариант установки через NFS попросит вас ответить на вопросы о вашей сети и сетевой информации вашего NFS сервера. NFS сервер должен быть настроен заранее. Так же следует отметить, что вы не можете пользоваться сетевыми именами, вы должны указывать IP адрес и для вашего компьютера, и для NFS сервера (на установочном диске нет преобразователя имён).

Установка из ранее под-монтированного каталога является наиболее гибким пунктом. Вы можете воспользоваться этим методом для установки с таких носителей, как Jaz диск, NFS подключённый через PLIP и с файловых систем FAT[5]. Под-монтируйте файловую систему к выбранной вами точке монтирования до запуска программы установки, затем укажите эту точку здесь.

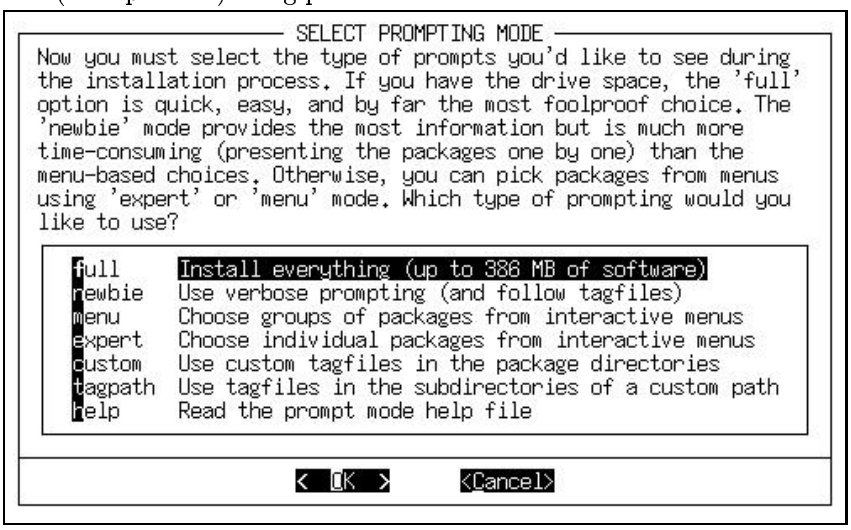
SELECT Этот пункт позволяет вам выбрать, какие разделы программ вы желаете установить. Эти разделы были описаны в разделе 3.1.2 на стр. 18. Пожалуйста, обратите внимание на то, что вы должны установить раздел A, чтобы получить минимальную работающую систему. Все другие разделы не являются обязательными.

[4]Согласно анонсу, начиная с Slackware 7.2 вариант установки с дискет будет недоступен. Прим. переводчика

[5]а так же с ISO образа компакт диска. См. Дополнение 1. Прим. переводчика



INSTALL В случае, если вы уже прошли через пункты "target", "source" и "select", этот пункт позволит вам выбрать, какие пакеты программ из выбранных вами разделов вы желаете установить. Иначе вам будет предложено вернуться назад и завершить всё в других пунктах программы установки. Этот пункт позволяет вам выбрать один из шести различных методов установки: full (полный), newbie (новичок), menu (меню), expert (эксперт), custom (выборочный) и tag path.



Подпункт full установит все пакеты из выбранных вами в пункте "select" разделов программ. Никаких больше вопросов. Это самый простой метод установки, так как вам не надо принимать никаких решений по поводу того, какие пакеты устанавливать, а какие — нет. Конечно, этот подпункт наиболее требователен к дисковому пространству.

Следующий доступный подпункт — newbie. Этот подпункт устанавливает все действительно необходимые пакеты в выбранных вами разделах. Для каждого из остальных пакетов вам будет предложено выбрать "Yes", "No" или "Skip". Yes или No очевидны, а Skip пропустит все остальные необязательные пакеты из данного раздела программ, и перейдёт к следующему. Дополнительно вам будет выведено описание и требования к дисковому

пространству для каждого из пакетов с целью помочь выбрать то, что вам действительно необходимо. Этот вариант рекомендуется для новых пользователей, так как он гарантирует, что все необходимые пакеты будут установлены. Тем не менее, этот метод немного медленен, из за постоянных опросов.

Гораздо более быстрый и расширенный метод — menu. Для каждого раздела программ вы увидите меню, в котором вы можете выбрать, какие из пакетов (не обязательных для этого раздела), должны быть установлены. Пакеты, установка которых необходима не показываются в этом меню.

Для более опытных пользователей, программа установки предлагает подпункт expert. Этот метод позволяет вам получить абсолютный контроль над тем, какие из пакетов должны быть установлены. Вы можете установить то, что вы желаете. Это может привести к неработающей системе в том случае, если вы не установите некоторые из абсолютно необходимых пакетов. С другой стороны, вы полностью контролируете, что должно быть установлено в вашей системе. Мы настоятельно не рекомендуем пользоваться этим способом установки новым пользователям. Так как вы довольно легко можете "прострелить себе ногу".

custom и path tag варианты установки так же рекомендуются для использования только опытным пользователям. Эти методы позволяют вам произвести установку на основе пользовательских tag файлов, созданных вами в дереве каталогов дистрибутива. Это очень удобно, если вам необходимо установить систему на большое количество компьютеров, сравнительно быстро. Для получения дополнительной информации по использованию tag файлов смотрите раздел 5.9.4 на стр. 142.

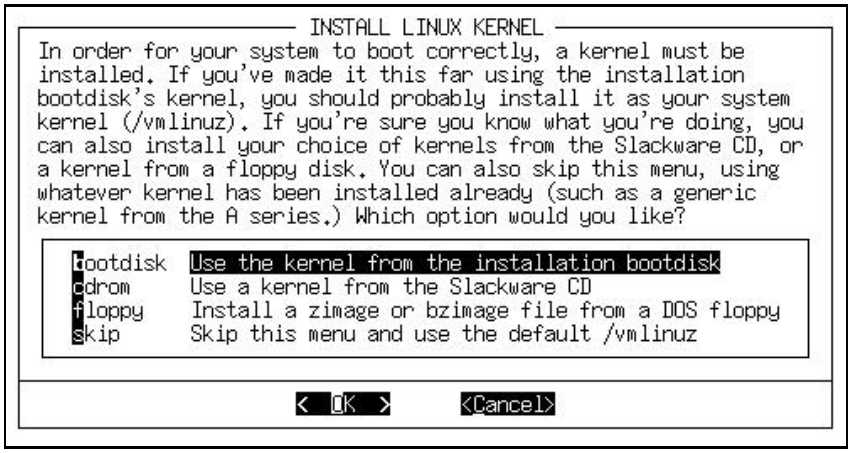
После того, как вы выбрали, каким из предложенных способов воспользоваться, возможны различные варианты продолжения. Если вы выбрали full или menu, то появится экран с меню, в котором вы можете выбрать пакеты для установки. Если вы выбрали full, то программа установки немедленно перейдёт к процессу копирования пакетов программ на выбранный вами ранее раздел жёсткого диска. Если вы выбрали newbie, то пакеты начнут копироваться до тех пор, пока не дойдёт очередь одного из дополнительных пакетов.

Пожалуйста, помните, что если вы выбрали слишком много пакетов программ для установки, по сравнению с тем, сколько свободного пространства имеется на жёстком диске, выбранном в пункте target, то место на диске может закончиться. Наиболее безопасным решением будет, не спешить с установкой некоторых из программ, а установить их позднее. Это можно проделать весьма легко, при помощи инструментов Slackware для работы с пакетами программ. Для информации смотрите раздел 5.9 на стр. 138.

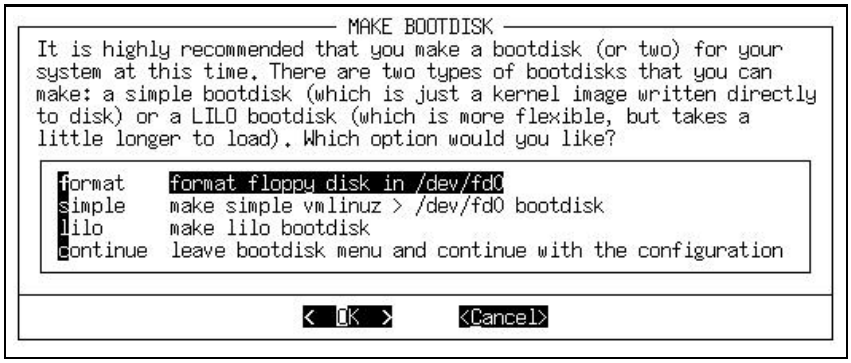
CONFIGURE Пункт configure (настройка) позволяет вам выполнить основные настройки системы. То что вы увидите здесь, во многом зависит от того, какие пакеты программ вы установили. Но всегда вы увидите следующее:

Kernel selection — выбор ядра. Здесь вы должны выбрать, какое ядро будет использоваться. Вы можете установить ядро с загрузочного диска, использованного вами в процессе установки, с CD диска Slackware или с другой дискеты, которую вы (всегда думая наперёд) приготови-

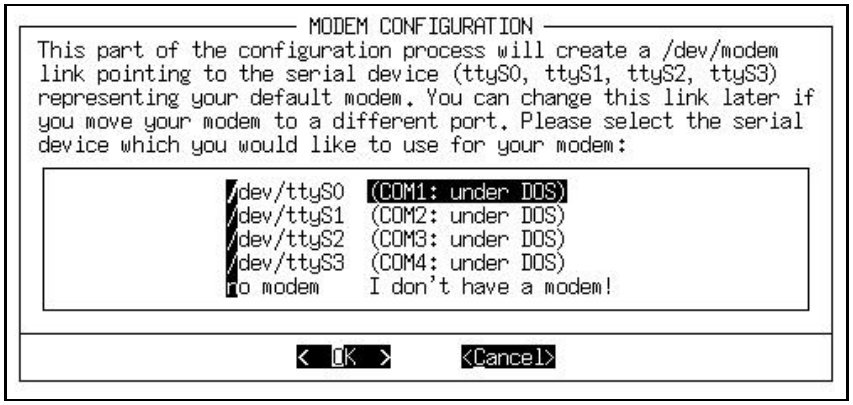
ли заранее. Так же вы можете пропустить выбор ядра. В этом случае будет использовано ядро по умолчанию.



Make a boot disk — создание загрузочного диска. Создание загрузочного диска для использования в будущем, вероятно является хорошей идеей. Вы сможете отформатировать флоппи диск и затем создать один из двух видов загрузочного диска. Первый тип, simple — просто записывает ядро на флоппи. Более гибкий (и настоятельно нами рекомендуемый) вариант — создать загрузочный диск lilo. Для подробностей по lilo, смотрите раздел 4.4.1 на стр. 75. Так же вы можете продолжить без создания загрузочного флоппи диска.



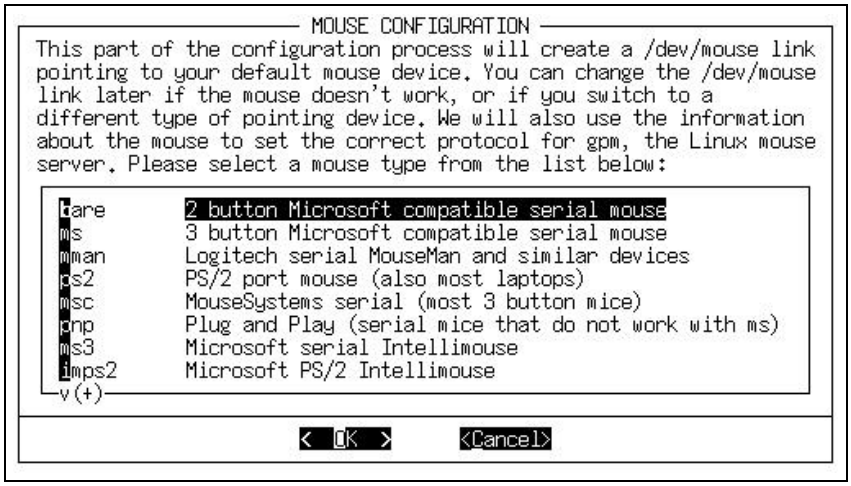
Modem Вам будут заданы вопросы о настройках вашего модема. Точнее, вы должны будете выбрать, есть ли у вас модем и если он у вас есть, то к какому последовательному порту он подключён.



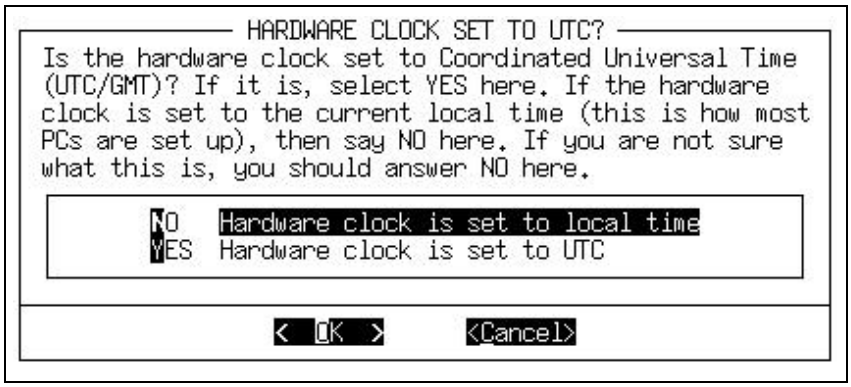
Следующие пункты могут появится, а могут и нет, в зависимости от того, были или не были установлены соответствующие им пакеты программ.

Timezone — часовой пояс. Всё довольно понятно: вас спросят, в каком часовом поясе вы находитесь. Если вы работаете по времени Зулу (Zulu) то мы приносим вам свои извинения, так как ваш часовой пояс находится в самом конце списка.

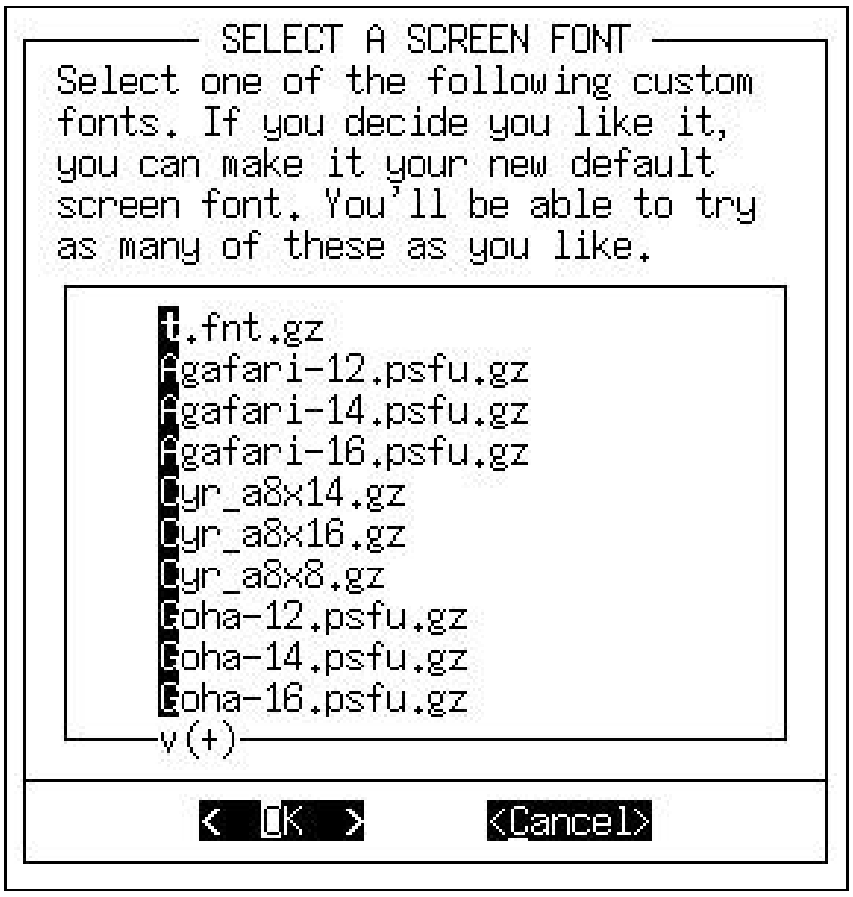
Mouse — мышь. Этот пункт спросит, какого типа мышь установлена в вашей системе, а так же, хотите ли вы, чтобы **gpm**(8) (поддержка мыши в режиме командной строки) была запущенна при загрузке.



Hardware clock — аппаратные часы. Этот раздел спрашивает, идут ли аппаратные часы вашего компьютера в соответствии с Координированным Универсальным Временем (UTC или GMT). Для большинства компьютеров ответ будет нет.



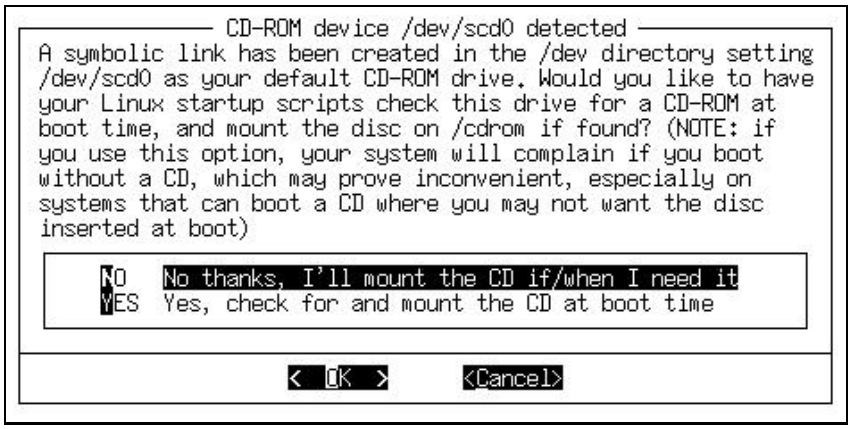
Font — шрифт. Подраздел font позволяет вам выбрать из списка подходящий шрифт для режима командной строки.



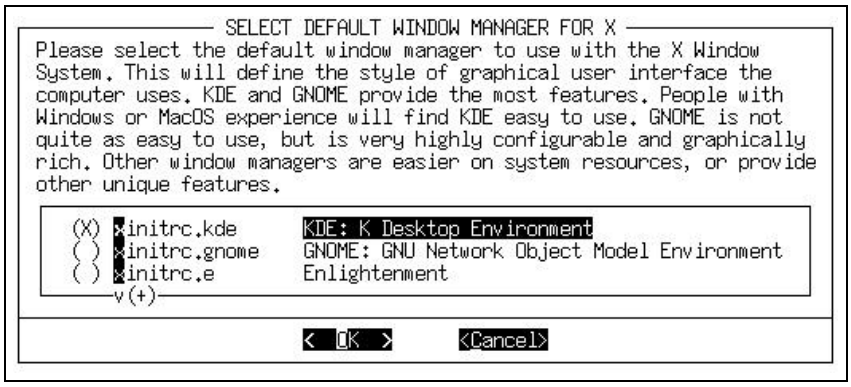
LILO Здесь вас спросят об установке LILO (LInux LOader – загрузчик Linux; см. раздел 4.4.1 на стр. 75). Если Slackware является единственной системой на вашем компьютере, то опция simple должна замечательно работать с вашей системой. Если у вас есть несколько операционных систем, то вам необходимо выбрать раздел expert. См. раздел 4.4.3 на стр. 78 для дополнительной информации. Третий пункт — не устанавливать, не рекомендован, до тех пор, пока у вас нет весьма серьёзных оснований поступить так. Если вы выполняете expert install, вам будет предоставлен выбор, куда разместить LILO. Вы можете разместить его в MBR — (Master Boot Record — главный загрузочный сектор) вашего жёсткого диска. В суперблок корневого каталога Linux, или на флоппи диск. Пожалуйста, обратите внимание на то, что если вы пользуетесь другой операционной системой, которая имеет свой загрузчик, то вам рекомендуется установить LILO в суперблок вашего Linux раздела, или на флоппи. Использование MBR в этом случае повредит загрузчик другой операционной системы и может очень сильно усложнить вашу жизнь.

Network — сеть. Раздел настройки сети, на самом деле выполнение отдельной программы **netconfig**. Смотрите соответствующий раздел 4.2.2 на стр. 46.

CD-ROM Тут вас спросят, хотите ли вы, чтобы система автоматически проверяла, есть ли диск в CD-ROM и монтировала его, если таковой имеется при загрузке.



X Window Manager — Менеджер X Window. Тут вы можете выбрать, какой менеджер окон использовать по умолчанию. См. 4.3 для подробностей.



В независимости от того, какие пакеты были установлены, программа установки спросит вас, хотите ли вы установить пароль суперпользователя. Настоятельно рекомендуется сделать это, по крайней мере из соображений безопасности; тем не менее, почти как и всё в Slackware, вы можете и не делать этого.

EXIT Само существование этого подраздела в этой книге — оскорбление вашему чувству собственного достоинства. Мы очень искренне извиняемся, и просим у вас прощения.

3.2 Итог

Slackware Linux, должно быть, уже установлен на вашем компьютере. В добавок к этому, вы приобрели немного знаний о разбиении жёсткого диска на разделы, о пакетах программ, о программ установки а так же о некоторых элементарных настройках системы. С этими знаниями вы должны быть готовы к работе по завершению настройки вашей системы.

Глава 4

Настройка

4.1 Настройка системы

Перед тем, как приступить к настройке более сложных частей системы, вам следует узнать немного больше о том, как система устроена и как можно найти в ней определённый файл или программу. Вам так же следует разобраться, необходимо ли вам компилировать ядро для вашей системы, и если да, то что для этого надо делать. Этот раздел ознакомит вас с организацией системы и её конфигурационными файлами. А затем мы перейдём к настройке более сложных частей системы.

4.1.1 Обзор системы

Очень важно понять, как организован Linux перед тем, как погружаться в различные аспекты по его настройке. Система Linux кардинальным образом отличается от систем DOS или Windows (а так же Macintosh OS). Этот раздел ознакомит вас с расположением основных элементов системы.

Организация файловой системы

Первое принципиальное отличие Slackware Linux от DOS или Windows — это организация файловой системы. Для начинающих пользователей: в Linux различным разделам жёсткого диска не сопоставляются буквы. В этой системе есть только один основной каталог. Вы можете провести аналогию с диском `C:` системы DOS. Каждый раздел жёсткого диска смонтирован (подключён)[1] к одному из каталогов основной директории. Что-то вроде всегда расширяемого диска.

Мы называем эту основную директорию корневой директорией, или корневым каталогом, а ссылаемся на неё при помощи одинарного слэша (`/`). Эта концепция может показаться вам странной, но на самом деле она значительно упрощает жизнь в том случае, если вам необходимо увеличить размер используемого дискового пространства. Например, у вас закончилось место на том диске, который содержит каталог `/home`. На самом деле, большинство пользователей при установке Slackware создают один большой корневой диск, так что это лишь пример. Так как разделы могут быть подключены к любому каталогу, вы можете просто пойти в магазин, купить ещё один жёсткий диск и подключить его к каталогу `/home`[2]. И вот вы "привили" немного свободного пространства к вашей системе. И всё без особых усилий по пере-настройке и переносу различных программ.

Ниже вы найдёте описание основных каталогов верхнего уровня в системе Slackware.

`/bin` Здесь хранятся основные программы пользователей. Имеются в виду самые основные команды, необходимые пользователю для работы в системе. Например, такие как оболочки и команды файловой системы (**ls**, **cp** и т.д.). Каталог `/bin` обычно не претерпевает изменений после установки. Если претерпевает, то обычно лишь при обновления пакетов программ предоставленных нами.

[1] английский глагол mount – монтировать, подключать. Прим. переводчика.

[2] Для сохранения работоспособности системы, вам всё же придётся переместить все каталоги из `/home` на новый диск. Прим. переводчика.

`/boot` Файлы, используемые Загрузчиком Linux-a (LILO). Эта директория так же практически не получает изменений после установки.

`/cdrom` Помните, что все диски подключаются к определённому каталогу? Так вот, каталог `/cdrom` предоставлен вам для использования в качестве точки монтирования вашего привода компакт дисков.

`/dev` В Linux всё рассматривается, как файл. Даже различные устройства, такие как последовательные порты, жёсткие диски и сканеры. Для получения доступа к определённому устройству, необходимо чтобы существовал специальный файл, называемый device node[3] . Все эти ноды находятся в каталоге `/dev`. Аналогично устроены большинство Unix — подобных операционных систем.

`/etc` Этот каталог содержит файлы настроек. Всё, от конфигурационных файлов системы X Window, базы данных пользователей и до стартовых сценариев. Администратор системы детально ознакомится с этим каталогом, со временем, конечно.

`/home` Linux является многопользовательской системой. Каждому пользователю присваивается эккаунт и уникальная директория для персональных файлов. Эта директория называется "home" (домашним) каталогом пользователя. Каталог `/home` предоставлен для расположения домашних директорий пользователей.

`/lib` Системные библиотеки, необходимые для основных программ находятся здесь. Библиотека C, динамический загрузчик, библиотека ncurses и модули ядра — это основные обитатели этого каталога.

`/lost+found` При загрузке системы происходит проверка файловых систем на наличие ошибок. Если они обнаружены, то запускается программа **fsck** и пытается исправить их. Восстановленные части файловой системы сохраняются программой в этом каталоге.

`/mnt` Этот каталог предоставляется как временная точка монтирования для жёстких дисков, или отключаемых устройств.

`/opt` Дополнительные пакеты программ. Идея в том, что все пакеты программ, устанавливаются в этот каталог, например `/opt/<программный пакет>` и в последствии если этот пакет вам более не нужен, то достаточно всего лишь удалить соответствующий каталог. В Slackware дистрибутиве некоторые программы поставляются в `/opt` каталоге (например KDE в `/opt/kde`), но вы вольны добавить всё, что угодно в `/opt`.

`/proc` Это в своём роде уникальная директория. На самом деле, она не является частью файловой системы, это виртуальная файловая система, которая предоставляет доступ к информации ядра. Различная информация, которую ядро хочет сообщить вам, подаётся вам через "файлы" в каталоге `/proc`. Вы так же можете сообщить ядру через некоторые из этих "файлов" попробуйте выполнить **cat /proc/cpuinfo**.

[3] поискать русские синонимы. рабочая ссылка

`/root` Администратор системы известен системе, как "root". Его домашний каталог — `/root`, вместо `/home/root`. Причина этого в том, что каталог `/home` может находится в разделе, отличном от `/` и если по какой-то причине `/home` не может быть подключён, то пользователь root вынужден будет войти в систему, чтобы решить проблему. И если его домашний каталог на другом диске, это усложнит вход в систему.

`/sbin` Основные программы, выполняемые пользователем root а так же программы, выполняемые процессом загрузки хранятся здесь. Обычные пользователи не будут пользоваться этими программами.

`/tmp` Временное хранилище данных. Все пользователи имеют права чтения и записи в этом каталоге.

`/usr` Это большой каталог в Linux системе. Практически всё остальное расположено здесь. Программы, документация, исходный код ядра и X Window система. Именно в этот каталог, скорее всего, вы будете устанавливать программы.

`/var` Системные лог файлы, кэш файлы и файлы-замки программ хранятся здесь.Это каталог для часто меняющихся данных.

Теперь у вас должно появится хорошее представление о том, что содержится в каких каталогах. Следующий раздел поможет вам научится легко искать конкретные файлы, чтобы не пришлось это делать вручную.

Поиск Файлов

Теперь вы знаете, какого рода данные содержит каждый каталог, но это на самом деле не поможет вам при поиске конкретных файлов. Конечно вы можете пролистать все каталоги, в поисках того, что вам надо, но существуют более быстрые способы. В Slackware есть четыре основных инструмента для поиска файлов.

which Первый из них — это команда **which**(1). **which** обычно используется для быстрого поиска программ. Он просто ищет в каталогах, указанных в вашей PATH переменной и выдаёт первое найденное соответствие, а так же путь к этому файлу. Например:

```
$ which bash
/bin/bash
```

Таким образом, `bash` находится в `/bin` каталоге. Это очень ограниченная команда для поиска, так как она ищет только в вашем PATH.

whereis Команда **whereis**(1) работает аналогично программе **which**, но в дополнение к последней, так же ищет man страницы и исходники программ. Результат выполнения **whereis** для bash будет следующий:

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

Эта команда не только говорит, где находится программа, но так же указывает, где находится её онлайн документация. Но она всё ещё весьма ограничена. Что если вам необходимо найти определённый конфигурационный файл? **which** или **whereis** тут вам не помогут.

find Команда **find**(1) может быть использована для поиска всего, чего угодно. Я хочу задать поиск файла **xinitrc** во всей системе.

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

find потребует много времени для поиска, так как эта команда пролистает всё дерево каталогов для поиска. И если вы выполните команду, как обычный пользователь, то на экран, вероятно, неоднократно будут выведены сообщения об ошибке доступа к некоторым из каталогов (к тем, которые может просматривать только root). Но **find** нашла наш файл. Уже хорошо. Если бы только она работала чуточку быстрее...

locate Команда **locate**(1) производит поиск по всей файловой системе, в точности как и **find**, но она просматривает свою базу данных вместо того, чтобы пролистать все каталоги. База данных настроена так, что она автоматически обновляется в 4:40 утра. Вы так же можете вручную выполнить **updatedb**(1), для обновления этой базы данных (перед запуском её вручную вам следует выполнить **su nobody**). Вот пример использования locate:

```
$ locate xinitrc # нам не надо быть root
/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 больше, чем нам было необходимо и очень быстро в то же время. С этими командами вы можете найти всё, что угодно в Linux системе.

Каталог `/etc/rc.d`

Файлы, выполняемые системой при её инициализации, хранятся в каталоге `/etc/rc.d`. Slackware использует расположение сценариев инициализации в стиле BSD. Каждое задание или runlevel (уровень загрузки) имеет свой `rc` файл. Таким образом, получается организованная структура, которой легко пользоваться.

Есть несколько категорий сценариев инициализации. Начальная загрузка системы, уровни загрузки, инициализация сети и System V совместимые. По традиции, мы отнесём всё остальное к категории "других".

Начальная загрузка системы

Первая программа, запускаемая в Slackware после ядра Linux —это **init**(8). Эта программа читает `/etc/inittab`(5) файл, чтобы узнать, как загружать систему. Запускает `/etc/rc.d/rc.S` сценарий для подготовки перед переключением на выбранный runlevel. `rc.S` файл активизирует виртуальную память, подключает файловые системы, очищает определённые log каталоги инициализирует Plug and Play устройства, загружает модули ядра, настраивает PCMCIA устройства, активизирует последовательные порты и запускает System V загрузочные сценарии (если таковые присутствуют). `rc.S` выполняет много задач самостоятельно, но так же до завершения своей работы, он вызовет ниже перечисленные сценарии из каталога `/etc/rc.d`:

`rc.S` Это и есть инициализационный сценарий.

`rc.modules` Загружает модули ядра. Такие, как поддержка сетевой карты, PPP и другие. Если сценарий находит `rc.netdevice`, то он выполнит и его.

`rc.pcmcia` Проверяет наличие и настраивает все PCMCIA устройства, присутствующие в вашей системе. Это наиболее полезно для пользователей laptop компьютеров, у которых, наверняка есть PCMCIA модем или сетевая карта.

`rc.serial` Настраивает последовательные порты, запуская соответствующие **setserial** команды.

`rc.sysvinit` Ищет System V инициализационные сценарии, соответствующие выбранному runlevel и выполняет их. Это более подробно обсуждается ниже.

Инициализационные сценарии runlevel

После того, как инициализация системы завершена, **init** переходит к инициализации runlevel. Runlevel описывает, в каком режиме ваш компьютер будет работать. Звучит загадочно? Ну, runlevel сообщает **init**-у, будете ли вы в многопользовательском (multiuser logins) режиме, или только в однопользовательском (single user), хотите ли вы сетевые сервисы и будете ли вы использовать X Window или **agetty**(8), для управления входом в систему (logins). Файлы, приведённые ниже определяют разные runlevels в Slackware Linux.

`rc.0` Выключает систему (runlevel 0). По умолчанию, это символическая ссылка на rc.6.

`rc.4` Многопользовательский запуск (runlevel 4), но в X11 с KDM, GDM или XDM в качестве менеджера входа в систему.

`rc.6` Перезагружает систему (runlevel 6).

`rc.K` Запуск в одно-пользовательском режиме (runlevel 1).

`rc.M` Многопользовательский режим (runleveld 2 и 3), но со стандартным текстовым входом (login). Это runlevel, используемый в Slackware по умолчанию.

Инициализация сети

Уровни загрузки 2, 3 и 4 запустят сетевые сервисы. Следующие файлы ответственны за инициализацию сети:

`rc.inet1` Этот файл, созданный программой **netconfig**, ответственен за настройку сетевого интерфейса.

`rc.inet2` Выполняется после `rc.inet1` и запускает основные сетевые сервисы.

`rc.atalk` Запускает AppleTalk сервисы.

`rc.httpd` Запускает вэб сервер Apache.

`rc.samba` Запускает Samba сервис.

`rc.news` Запускает сервер новостей.

Совместимость с System V

Совместимость с System V была предоставлена в Slackware 7.0. Много других Linux дистрибутивов используют этот форматом вместо BSD формата. В этом стиле каждому уровню загрузки предоставляется каталог для сценариев, а в BSD-стиле каждому уровню загрузки соответствует только один сценарий.

`rc.sysinit` сценарий произведёт поиск всех System V init сценариев в `/etc/rc.d` каталоге и выполнит их, если уровень загрузки соответствующий. Это полезно, если вы пользуетесь коммерческим программным обеспечением, которое устанавливает System V init сценарии, в то же время, вы можете пользоваться и BSD сценариями.

Другие файлы

Сценарии, описанные ниже — это все остальные загрузочные сценарии. Они обычно выполняются одним из вышеперечисленных основных сценариев, таким образом, всё что вам необходимо сделать для изменения настроек, это отредактировать содержание соответствующих файлов.

`rc.cdrom` Если активизирован, то он проверит, есть ли компакт диск в приводе и если есть, то подключит его к `/cdrom`.

`rc.gpm` Запускает сервис основной поддержки мыши (general purpose mouse). Это позволит вам копировать и вставлять текст в консоли.

`rc.ibcs2` Запускает поддержку Intel Binary Compatibility. Это необходимо только в том случае, если вы планируете выполнять программы, откомпилированные на SCO Unix или на других коммерческих Intel Unix разновидностях. В этом нет необходимости для запуска Linux программ.

`rc.font` Загружает пользовательский экранный шрифт для режима командной строки.

`rc.local` Содержит всю специфическую загрузочную информацию для вашей конкретной системы. В свеже-установленном дистрибутиве этот файл пуст. Он зарезервирован для использования администратором системы. Этот сценарий выполняется самым последним при загрузке.

Для активизирования сценария всё, что вам нужно сделать, это добавить ему разрешение на выполнение при помощи **chmod** программы. Чтобы выключить сценарий, удалите разрешение на выполнение соответствующего файла. Для получения дополнительной справки по **chmod**, смотрите раздел 5.2.2.

4.1.2 Выбор ядра

Ядро это та часть операционной системы, которая обеспечивает доступ к аппаратному обеспечению компьютера, управление процессами и контроль за работой всей системы. Ядро содержит поддержку аппаратных устройств, так что выбор ядра для вашей системы — очень важный шаг при установке.

Slackware предоставляет около 60-ти прекомпилированных ядер. Так что перед вами открывается широкий выбор. Каждое из ядер включает в себя набор стандартных драйверов, плюс дополнительные специфические драйвера. Вы можете использовать одно из прекомпилированных ядер, или вы можете самостоятельно откомпилировать ядро для вашей системы. В любом случае, вам следует убедиться в том, что используемое вами ядро содержит поддержку аппаратных устройств, присутствующих в системе.

Каталог `/kernels` на Slackware CD-ROM

Прекомпилированные Slackware ядра доступны в каталоге `/kernels` на Slackware CD-ROM, а так же на FTP сайте в основном каталоге Slackware. С появлением новых версий дистрибутива, обновляются и ядра, так что документация в каталоге с ядрами — всегда наиболее полный источник информации по ним. Каталог `/kernels` содержит под-каталог для каждого из ядер. Имена под-каталогов совпадают с именами ядер. В каждом под-каталоге вы найдёте следующие фалы:

Файл	**Назначение**
`System.map`	Системный map файл для этого ядра
`bzImage` (или `zImage`)	Образ ядра
`config`	Конфигурационный файл исходника для этого ядра

Чтобы установить ядро, скопируйте `System.map` и `config` в каталог `/boot` вашей системы, а образ ядра скопируйте в файл `/vmlinuz`. Запустите **/sbin/lilo**(8), чтобы установить LILO для нового ядра, а затем пере загрузите систему. Это всё что вам необходимо проделать для установки одного из прекомпилированных ядер в вашу систему.

Ядра, заканчивающиеся на ".i" — это IDE ядра. Т.е. они не содержат SCSI поддержки в самом ядре. Ядра, заканчивающиеся с ".s" — это SCSI ядра. Они содержат поддержку IDE устройств точно так же, как и ".i" ядра, плюс поддержку SCSI.

Компилирование ядра из исходников

Новые пользователи часто спрашивают: "Следует ли мне компилировать ядро для моей системы?". Ответ, определённо: может быть. Большинству пользователей подойдёт одно из прекомпилированных ядер, с подгружаемыми модулями для устройств, не поддерживаемых самим ядром. Вы захотите откомпилировать ядро в случае, если вы обновляете его версию на более новое, которое мы ещё не предоставили в Slackware, или если вы наложили патч на исходник вашего ядра.

Сборка своего собственного ядра не такой уж сложный процесс. Первым делом необходимо убедиться, что в вашей системе установлен исходный текст ядра. Убедитесь, что вы установили пакеты из раздела K во

время установки. Вам так же понадобятся пакеты из раздела D, в частности компилятор C, GNU make и GNU binutils. В общем неплохо бы, чтобы весь раздел D был установлен, если вы собираетесь заниматься разработкой чего либо. Теперь мы готовы к сборке ядра:

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

Первым делом вернём исходный текст ядра к его первоначальному состоянию. Для этого выполним следующее:

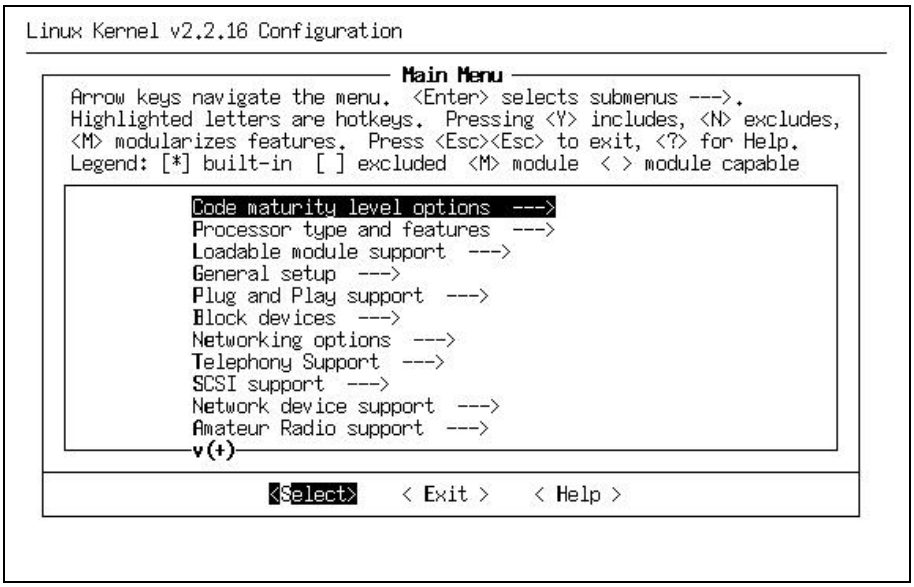
```
# make mrproper
```

Теперь вы можете настроить ядро под вашу систему. Современное ядро предлагает на ваш выбор три способа выполнения этой задачи. Первый метод — это оригинальная текстовая система вопрос-ответ. Вам будет заданно много вопросов, а после этого будет создан конфигурационный файл. Проблема с этим способом в том, что если вы ошиблись, то вы должны начать всё сначала. Метод, предпочитаемый большинством пользователей — метод, основанный на системе меню. Ну и наконец. последний — X-ориентированный инструмент настройки ядра. Выберите, какой вам больше нравится, и наберите соответствующую команду:

 # make config (текстовая вопрос-ответ версия)

#make menuconfig (menu, текстовая версия)

#make xconfig (X-ориентированная версия, вначале убедитесь, что вы в X)



Для новых пользователей, вероятно, **menuconfig** покажется наиболее простой в использовании. Вы можете найти экраны с помощью, которые объяснят предназначение каждой части ядра. После настройки ядра, выйдите из программы настройки. Она создаст необходимые конфигурационные файлы. Теперь мы можем подготовить дерево исходников ядра к сборке:

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

Следующий шаг — компиляция ядра. Вначале попробуйте задать ко-

манду **zImage**. Это не пройдёт, если ваше ядро слишком большое. Не волнуйтесь, вы всё ещё можете попробовать **bzImage**.

 # make zImage (Вначале попробуйте так)
 # make bzImage (Если предыдущая команда не сработала, попробуйте эту)

В зависимости от скорости процессора, этот процесс может быть достаточно долгим. Во время процесса сборки вы увидите на экране команды компилятора. После сборки образа ядра вы захотите собрать все те части ядра которые вы указали загружаемыми, как модули.

 # make modules [4]

Теперь мы можем установить свеже-скомпилированное ядро и модули. Чтобы установить ядро в Slackware системе, необходимо воспользоваться следующими командами:

 # mv /vmlinuz /vmlinuz.old
 # cat arch/i386/boot/zImage > /vmlinuz
 # mv /boot/System.map /boot/System.map.old
 # cp System.map /boot/System.map

Замените **zImage** на **bzImage**, если вам пришлось собрать большее ядро. Вероятно, вы захотите отредактировать `/etc/lilo.conf` файл и добавить соответствующий раздел для вашего старого ядра, на тот случай, если новое ядро не работает. После этого выполните **/sbin/lilo**, чтобы установить новый загрузочный блок. Теперь вы можете пере-загрузиться с новым ядром.

Использование модулей ядра

Модули ядра, это другое название драйверов устройств, которые могут быть вставлены в запущенное ядро. Они позволяют вам расширить список аппаратных устройств, поддерживаемых ядром, без установки другого ядра, или пере-компиляции заново.

Модули могут быть загружены или выгружены в любое время, даже во время работы системы. Это позволяет системным администраторам очень легко обновлять драйвера специфических устройств. Новый модуль может быть откомпилирован, старый выгружен, а новый загружен, и всё это без пере-загрузки компьютера.

Модули хранятся в каталоге `/lib/modules/<kernel version>` вашей системы. Они могут быть загружены во время загрузки компьютера из `rc.modules` файла. В этом файле есть очень много комментариев и примеров для типичных устройств. Для просмотра загруженных модулей воспользуйтесь командой **lsmod**(1):

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

Как видно из примера, у меня загружен только модуль параллельного порта. Для выгрузки модуля используйте команду **rmmod**(1). Модули могут быть загружены командами **modprobe**(1) или **insmod**(1). **modprobe**

[4] На самом деле, так же необходимо выполнить **make modules_install**. Прим. переводчика.

обычно безопаснее, так как автоматически загрузит модули, необходимые для модуля, который вы пытаетесь загрузить.

Большинство пользователей никогда не загружают или выгружают модули вручную. Они пользуются загрузчиком ядра для менеджмента модулей. Всё что вам надо сделать, это раз-комментировать строчку **/sbin/kerneld**(8) в `/etc/rc.d/rc.modules` и автозагрузчик запустится. Он позаботится о загрузке и выгрузке модулей по мере надобности. Автозагрузчик определяет, что нужно загрузить модуль по обращению к устройству, для которого в данный момент модуль не загружен.

Вы можете найти больше информации в man страницах для каждой из команд, а так же в `rc.modules` файле.

4.1.3 Итог

Должно быть, вы теперь знакомы с командами для поиска в файловой системе, с устройством файловой системы, а так же с файлами настроек в каталоге `/etc`. Эти знания будут вам крайне полезны при более подробном изучении системы. В добавок к этому, вы должны были узнать, как настроить и компилировать ядро из исходников.

4.2 Настройка сети

4.2.1 Сетевое оборудование

Как и большинство наиболее интересных вещей, которые вы можете делать с компьютером, подключение его к сети требует специального аппаратного оборудования. Возможно вам понадобится NIC (Network Interface Card — карта сетевого интерфейса), для подключения к LAN, возможно модем для подключения к провайдеру интернет, а может быть и оба (или несколько каждых из выше перечисленных, а может и ни одного).

При настройке удобно разделить аппаратные средства на скажем, PCMCIA (для laptop-ов) и не-PCMCIA категории. Суть этого разделения в том, что сейчас PCMCIA оборудование не поддерживается стандартной поставкой ядра, но поддерживается отдельным пакетом, включающим необходимые драйвера (как модули ядра) и некоторые программы для настройки и управления PCMCIA устройствами. Всё остальное, конечно, управляется стандартной поставкой ядра[5].

`netmods`

Драйвера сетевых устройств, поддерживаемых ядром, включены в пакет netmods (`slakware/n3/netmods.tgz`)[6]. Если вы ещё не установили этот пакет, то сейчас самое время установить его. (См. раздел 5.9 на стр. 138, для получения информации о том, как устанавливать отдельные пакеты программ.)

Модули ядра, которые должны быть загружены во время загрузки компьютера, загружаются из файла `rc.modules`, расположенного в каталоге `/etc/rc.d`. В стандартном файле `rc.modules` есть раздел "Network device support". Если вы откроете этот файл и заглянете в упомянутый раздел, вы заметите, что он вначале проверяет, существует ли исполняемый файл `rc.netdevice` в каталоге `/etc/rc.d`; `rc.netdevice` создаётся в том случае, если **setup** сумел определить ваши сетевые устройства при установке. Если это так, то вы скорее всего не читаете этот раздел (хм, парадокс, однако); ну а если не так, то продолжайте читать.

Ниже, после блока "if", находится список сетевых устройств и **modprobe** строчек, каждая из которых прокомментированна. Найдите ваше устройство и раскомментируйте соответствующую **modprobe** строчку, не забудьте сохранить изменения в файл. Если вы выполните, как root пользователь, `rc.modules`, то драйвер вашего устройства должен загрузиться (так же, как и все остальные откомментированные модули). Обратите внимание, что некоторые модули (такие, как драйвер ne2000) требуют указания параметров; убедитесь, что вы выбрали правильную строчку.

Сетевые устройства PCMCIA

Сетевые устройства PCMCIA настроить пожалуй даже проще, чем остальные. Убедитесь, что вы установили pcmcia пакет

[5] Начиная с ядра 2.4.0 PCMCIA устройства поддерживаются стандартной поставкой ядра. Прим. переводчика.

[6] Похоже, что в Slackware 7.2 такого пакета больше нет. Он является частью другого пакета. Прим. переводчика.

(`slakware/a11/pcmcia.tgz`. (См. раздел 5.9 для подробностей по установке пакетов). При установке pcmcia пакет создаст файл `rc.pcmcia` в `/etc/rc.d` каталоге и директорию `/etc/pcmcia`, а так же установит драйвера в каталог `/lib/modules/<версия ядра>/pcmcia`. Что здорово в этом пакете, так это то, что он попробует автоматически определить вставку и извлечение поддерживаемых pcmcia устройств; если вы просто вставите ваш сетевой pcmcia адаптер то услышите звуковой сигнал при загрузке необходимого модуля. При извлечении карты, её модуль должен быть автоматически выгружен.

К сожалению, если вы обновите ядро, вам скорее всего понадобится перекомпилировать pcmcia-cs, чтобы обновить драйвера. Естественно, исходный текст поставляется с дистрибутивом; поищите в `/source/a/pcmcia` каталоге исходники, сценарии и загляните в документацию.

4.2.2 Сетевые утилиты

ifconfig

Ну вот, ваше ядро уже умеет общаться с сетевым оборудованием. Теперь надо научить программы указывать ядру куда передавать информацию, и наоборот. Нам надо настроить интерфейс. Нам нужен **ifconfig**(8).

ifconfig лучше всего изучать на примерах; вы можете захотеть просто заглянуть в ваш `rc.inet1` файл (описанный в разделе 4.2.4 на стр. 54), чтобы посмотреть, как программа запускается оттуда. Типичный вариант выглядит следующим образом:

**# ifconfig eth0 192.168.1.10 broadcast 192.168.1.255 **
netmask 255.255.255.0

Эта строчка поднимает eth0 (первый сетевой интерфейс; для token ring используется tr0, для ppp — ppp0, и т.д.), с IP адресом 192.168.1.10, широковещательным адресом 192.168.1.255 (вся подсеть 192.168.1.) и сетевой маской 255.255.255.0 (показывает, что три первых части IP адреса относятся к сети, а .10 относится к вашему хосту). До тех пор, пока вы не делаете что-то очень неординарное, вы можете почти всегда использовать широковещательный адрес, состоящий из первых трёх блоков вашего IP, и завершающийся числом 255. Так же, почти всегда вы можете пользоваться маской сети 255.255.255.0. Если вы делаете что-то неординарное, вы наверное знаете достаточно, и эта часть книги вам вовсе не понадобится.

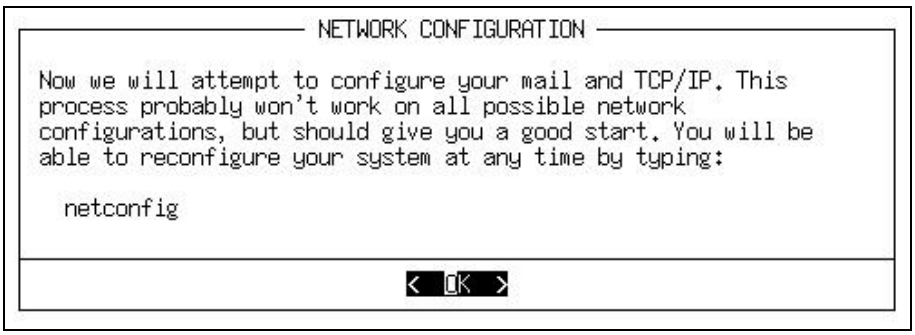
ifconfig так же может быть использован для просмотра текущих настроек. Запустите его без опций или параметров, чтобы получить список всех ваших сетевых интерфейсов и их настроек.

route

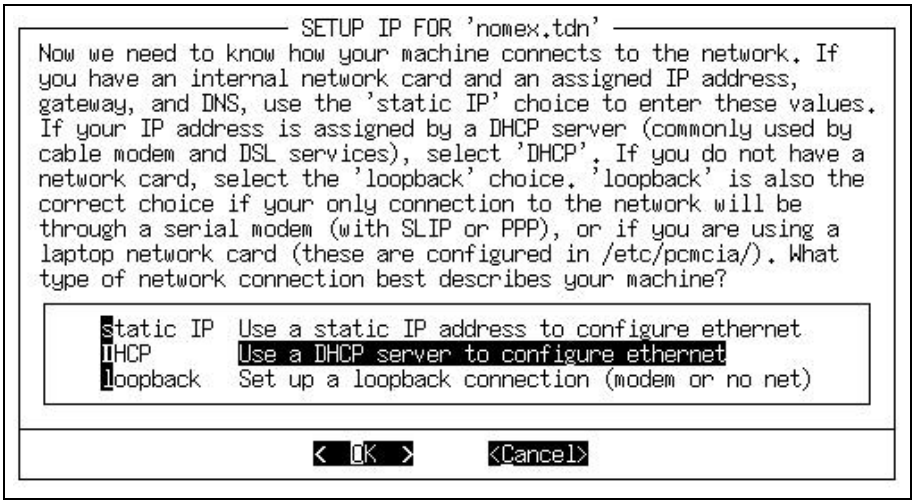
Для того, чтобы знать, какую куда посылать информацию, ядро использует роутинговую таблицу (routing table). Я не собираюсь углубляться в подробности, но вы можете просмотреть эту таблицу при помощи **/sbin/route**(8). **route -n** выдаст вам таблицу IP адресов вместо имён; это полезно в случае возникновения затруднений с вашим сервером имён или если вы просто не интересуетесь иллюзорным миром доменных имён. К счастью, если вам надо настроить простую сеть (как и большинству людей), то 2.2 ядро автоматически создаст роутинговую таблицу за вас.

netconfig

netconfig это часть программы установки Slackware, но как и большинство частей программы установки, может быть использована самостоятельно. **netconfig** достаточно проста в использовании и проведёт вас через процесс установки обычного сетевого соединения. Она особенно пригодна, если вы не очень знакомы, или вам просто не нравится копаться в сетевых `rc` файлах. После запуска **netconfig** должен появится такой экран:



Затем вам предложат ввести имя хоста и домена вашего компьютера. Скорее всего вы можете ввести просто что-нибудь, если конечно вы не настраиваете сервер, которым будут пользоваться много народа. Затем вас спросят, будете ли вы использовать статический IP, динамический DHCP, или же просто loopback.



Если вы не будете подключены к сети, выберите loopback. Если вы настраиваете компьютер, который будет подключён к университетской или большой офисной сети, то наиболее вероятно, вам придётся выбрать DHCP. Иначе выбирайте статичный адрес. Если вы не выбрали статичный адрес, то на этом программа завершиться. Если же вы выбрали статичный адрес, то вам будет необходимо указать IP вашего компьютера, маску сети, широковещательный адрес и адрес сервера имён. **netconfig** подскажет вам, как выяснить все эти параметры.

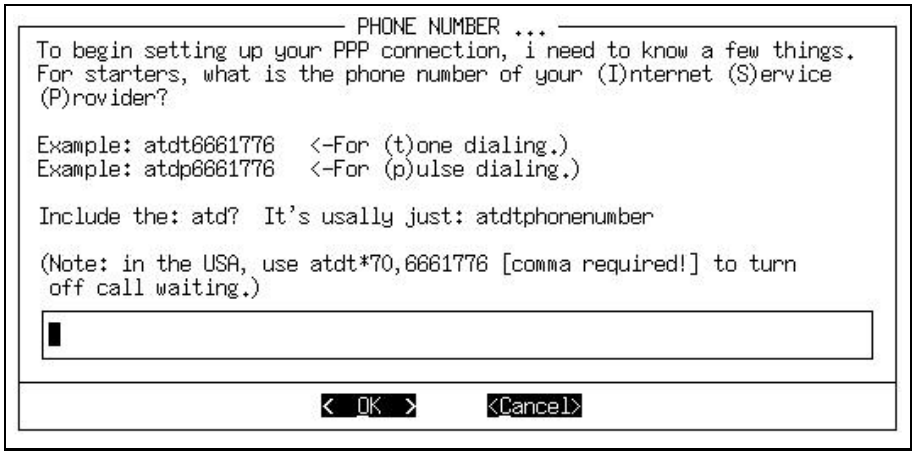
pppsetup

В Slackware есть **pppsetup** — утилита для настройки dialup соединения к ISP (Internet Server Provider). Она находится в `ppp.tgz` пакете из раздела программ N. **pppsetup** использует такой же интерфейс, как и программа установки. Если вы не помните, как пользоваться этим интерфейсом, то вернитесь к разделу 3.1.2, на стр. 22, для получения справки. **pppsetup** задаст вам много вопросов и установит несколько конфигурационных файлов в `/etc/ppp`. Как root запустите **pppsetup**; давайте пройдёмся по списку вопросов.

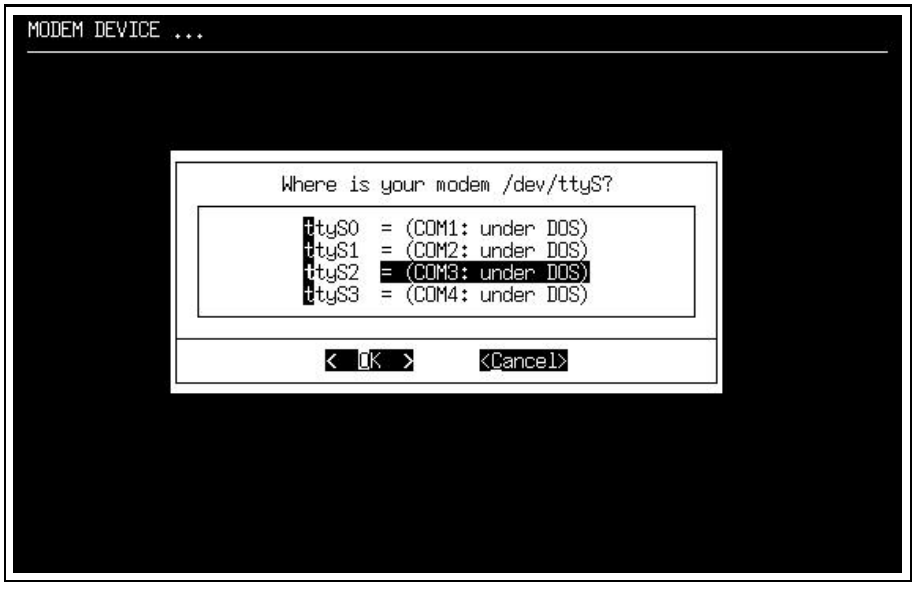
Phone number Здесь вам следует указать номер вашего ISP, с указанием в качестве префикса, типа набора номера. Для большинства людей тональный набор будет правильным выбором[7]. Если номер вашего провайдера 555-1013 и вы используете тональный набор, то вам надо ввести **atdt5551013**[8]. Если на вашей телефонной линии есть call waiting (ожидание звонка) и вы хотите отключить этот режим во время работы с провайдером, то введите что-то вроде **atdt*70,5551013**. Запятая необходима. Она вставляет 1,5 секундную паузу после *70, чтобы отключить режим ожидания звонка. Без запятой это не сработает.

[7]Для бывшего союза (за исключением Москвы и Санкт-Петербурга), надо выбрать импульсный набор. Прим. переводчика.

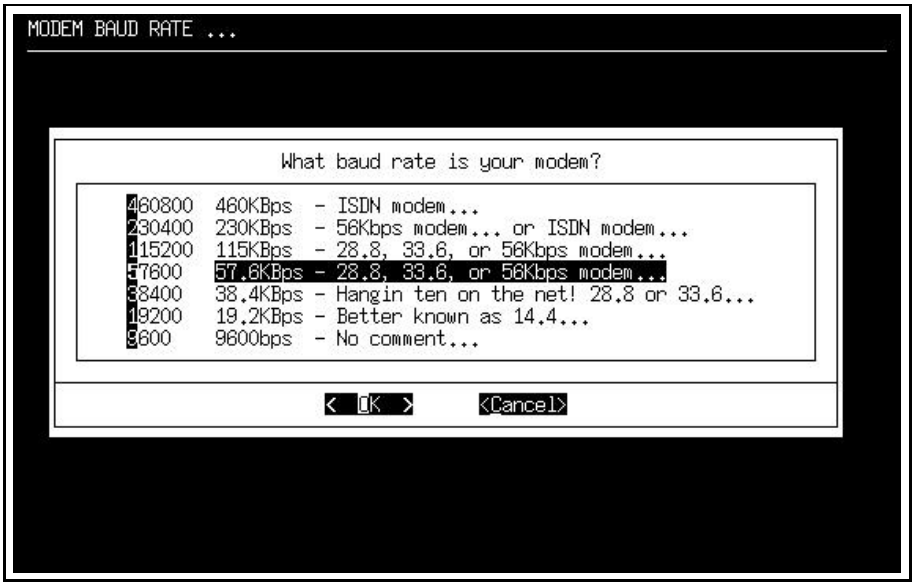
[8]atdp5551013 для импульсного набора. Прим. переводчика.



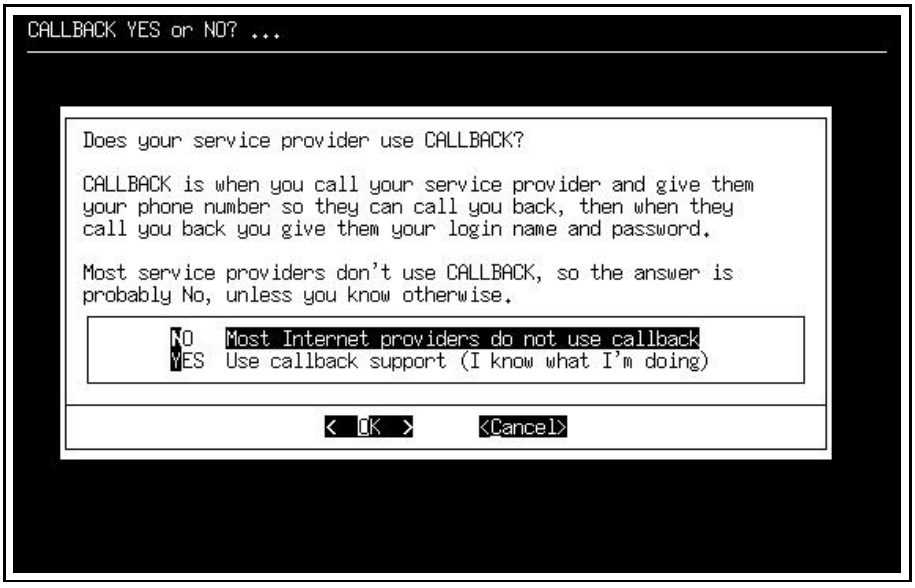
Modem device Здесь вам следует выбрать, куда подключён ваш модем. Если вы знаете на каком COM порте он был под Windows, то вы можете выбрать из списка эквивалент. Иначе вам придётся поэкспериментировать. Наиболее логично начать с `ttyS0` и пройтись по списку.



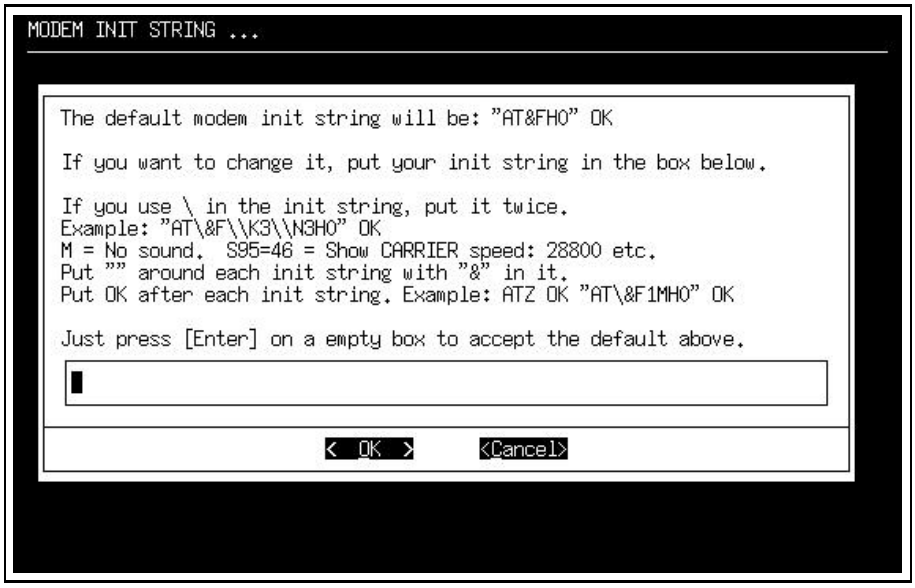
Modem baud rate Подберите baud rate, который наиболее близок вашему модему. Если вы не знаете baud rate, обратитесь к документации вашего модема.



Callback Теперь вам понадобится вся информация, предоставленная вам вашим ISP. Очень немногие ISP используют callback, так что скорее всего, вы можете спокойно ответить "No" на этот вопрос. Callback, это когда вы вначале звоните ISP, а затем они перезванивают на ваш номер, и только тогда вы можете войти в сеть. Если вам надо использовать callback, то ответьте "Yes" на этот вопрос. Тогда вам будет предложено ввести номер вашего телефона, login и пароль. Возможно, вам не понадобится вводить ваши начальные логин и пароль. Ну и наконец, последний вопрос: какую из схем идентификации (authentication) использует ваш провайдер. Если это CHAP или PAP, то ответьте "Yes". Позже вам необходимо будет настроить это. Смотрите ниже, как это сделать. Если они не пользуются вышеперечисленными методами, то ответьте "No" и смотрите раздел "Chat script" ниже.



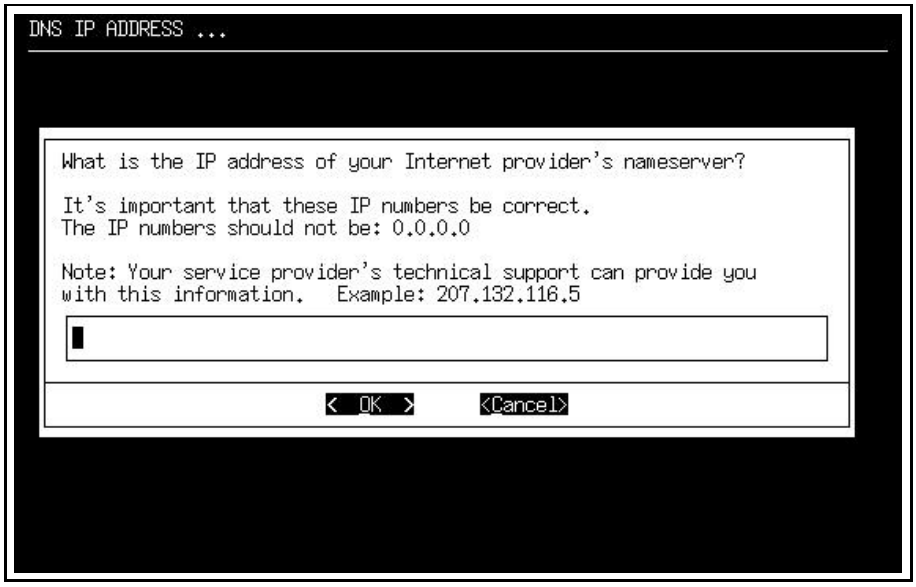
Modem init string Если у вас не какой-то необычный модем, то вам скорее всего можно просто нажать ввод, чтобы выбрать строку инициализации модема по умолчанию ("AT&FH0"). Иначе, смотрите документацию вашего модема. И найдите там строку инициализации, рекомендованную для вашего модема.



Domain name Теперь вам надо ввести доменное имя вашего провайдера. Это будет что-то вроде "primer.net", "slackware.com" или что-то похожее. (Ну ладно, почти наверняка это будет не slackware.com :)

DNS IP address Ваш ISP должен предоставить вам IP адрес их сервера имён. Если он у вас есть, введите его тут. Иначе обратитесь к ISP, чтобы узнать его[9].

[9] В Дополнении 1 будет описано, как настроить кэширующий DNS сервер на вашем



Authentication method Вам необходимо узнать у ISP, пользуются ли они CHAP, PAP, или ни одним из них при идентификации пользователя. Проще всего узнать это, позвонив вашему ISP. Тем не менее, если после набора номера вы увидите приглашение login и password после соединения, то наиболее вероятно, вам следует воспользоваться "SCRIPT". Иначе, обратитесь к ISP, чтобы выяснить, каким методом пользоваться.

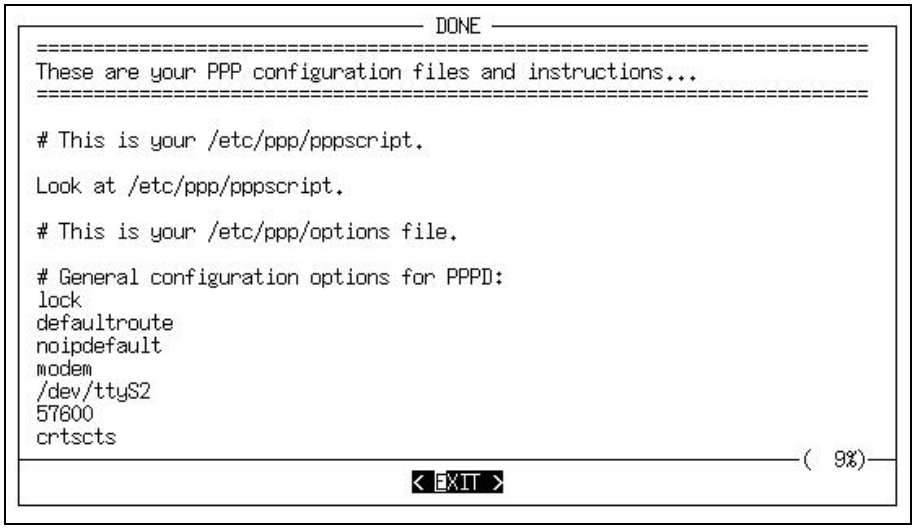
PAP или CHAP Если вы выбрали PAP или CHAP на экране выбора метода идентификации, то здесь вам необходимо будет ввести ваше имя пользователя (username). ISP должен присвоить вам username. Если

компьютере. Прим. переводчика.

это не так, то что-то определённо не правильно. Вам понадобится связаться с ними и получить username. Затем вам будет необходимо ввести пароль, предоставленный вам вашим ISP.

Chat script Если вы выбрали "SCRIPT", то вашему вниманию предоставится довольно таки долгое объяснение того, что же такое script. Прочтите его пожалуйста очень внимательно, так как там всё очень подробно описано. в общем, вам необходимо сообщить программе, какую информацию провайдер должен сообщить вашему компьютеру, и что компьютер должен ответить для входа в сеть. Вы попадёте в "петлю" вопросов и ответов, где вам надо будет ввести некоторый текст, который компьютер должен ожидать от ISP, и при получении такового, ответить на него то что вы сообщите далее. Для выхода из цикла, просто нажмите ввод на любой пустой строке.

Done В конце на экран будут выведены все данные о ppp конфигурации. Здесь вы не сможете изменить их содержание, но по крайней мере, вы можете проверить, всё ли указанно правильно. Нажмите ввод для того чтобы сохранить все настройки и выйти из программы. На этом экране вы так же найдёте много информации о том, как активизировать ваше dial-up соединение, и как разорвать его. Основная идея этого в следующем: как root, запустите **ppp-go**, для активизации соединения. Когда программа выдаст вам локальный и удалённый IP, вы подключены к интернет. Когда вы завершили всё, что хотели, выполните **ppp-off**, как root и эта программа отсоединит вас от сети.



4.2.3 Файлы `/etc`

`/etc/inetd.conf`

Для операционной системы, ориентированной на сетевое использование, вполне нормальным является запуск различных сетевых сервисов. Обычно для каждого из сервисов должна существовать программа, которая будет сидеть и слушать запросы по соединениям. Что может стать обременитель-

ным для системы, если в ней запущенно слишком много различных сервисов. Для уменьшения загрузки системы была создана программа **inetd**. **inetd** — это "интернет-суперсервер". Эта программа слушает запросы по многим socket-ам, и когда поступает запрос, **inetd** передаёт управление соответствующему серверу, для обработки запроса. Таким образом вместо нескольких ожидающих серверов, запущен только один.

Файл настроек для **inetd** — `/etc/inetd.conf`. В нём определяется, какой сервер должен быть запущен для какого соединения. В man страничке для **inetd**(8) вы, конечно, можете найти больше информации о использовании программы. Давайте быстренько пробежимся по основным сервисам:

ftp stream tcp nowait root /usr/sbin/tcpd wu.ftpd -l -i -a

Эта строчка относится к ftp серверу. Обратите внимание, вначале идёт имя протокола — "ftp", а в конце — программа, которая должна быть запущена для ответа на запрос. В приведённом примере, программа, которая должна быть запущена для ответа на поступивший запрос соединения — это `/usr/sbin/tcpd`; это программа "wraper", которая обеспечивает основные требования безопасности для того сервера, для которого она запущена. **wu.ftp** фактически и есть наш ftp сервер[10], но tcpd запускает его для вас. Больше информации вы можете найти в разделе 4.2.7.

Как и в большинстве системных файлов, строчки в **inetd.conf** комментируются символом #; вы можете активизировать и (или) остановить сервисы для **inetd**, откомментировав или закомментировав соответствующие строчки в этом файле и последующим перезапуском **inetd**.

`/etc/resolv.conf`

Этот файл сообщает всей основной системе, откуда брать DNS информацию. Все серверы имён, которыми вы пользуетесь, перечислены здесь, так же и имя домена вашего хоста. Вот вам пример этого фала (с laptop-а на котором я печатаю всё это — ninja.tdn):

domain tdn
nameserver 192.168.1.1
search tdn. slackware.com

Первая строчка описывает имя домена для ninja; это всё что идёт после имени домена в моём адресе. Вторая — DNS сервер в нашей домашней сети. Вы можете прописать их столько, сколько захотите; они будут обработаны в том порядке, в котором записаны, когда какой-то программе необходимо найти IP адрес, соответствующий какому-то доменному имени.

Последняя строчка немного интереснее. Она описывает все доменные имена, присваиваемые системе. Например, предположим, у меня есть машины zuul.tdn и hejaz.slackware.com. Я могу просто выполнить **ping zuul** и **ping hejaz**, чтобы пропинговать их, соответственно. **ping** вначале попробует добавить ".tdn" к имени zuul, и найдёт соответствующий хост. В случае с "hejaz" вначале будет опробовано имя "hejaz.tdn". Соответствующий хост не будет обнаружен, и поэтому будет произведён поиск "hejaz.slackware.com" и бинго. Следует отметить, что все домены, перечисленные в search должны заканчиваться на '.', за исключением последнего; если есть только один, то он же и является последним, и поэтому указывать '.' не надо.

[10] В Slackware-current (7.2) вместо **wu.ftp** представлен **proftp**. Прим. переводчика.

`/etc/hosts`

Файл `hosts` позволяет осуществить простейший способ поиска хостов в домене. Он представляет собой список хостов и соответствующих им IP адресов. Это полезно в небольших сетях, где использование DNS не оправдано, так же этот файл используется во время загрузки системы, когда сервер имён ещё не доступен. Мой выглядит следующим образом:

```
127.0.0.1 localhost
192.168.1.32 ninja.tdn ninja
```

Первая строчка должна быть само-достаточной. Вторая может и не быть. Вы можете указать столько имён и синонимов для одного адреса, сколько захотите, разделяя их пробелами. Итак, у меня "192.168.1.32" переводится в "ninja.tdn" (и наоборот), но синоним "ninja" так же может быть использован, когда мне лень набирать ".tdn" (как обычно и происходит).

4.2.4 `rc.inet1`

`/etc/rc.d/rc.inet1` это файл, используемый для сетевой "инфраструктуры" — он инициализирует устройства и устанавливает адреса и пути. `rc.inet1`, поставляемый Slackware, достаточно подробно прокомментирован, так что мы повторимся если повторим всё это здесь.

4.2.5 `rc.inet2`

Файл `/etc/rc.d/rc.inet2` относится к другой части сети: установке сервисов и демонов[11] и оперирует со всеми интересными настройками сети. Давайте рассмотрим блок для примера:

```
#Start the NAMED/BIND name server:
if [ -f $NET/named ]; then
  echo -n "named"
  $NET/named -u daemon -g daemon
fi
```

Важна здесь четвёртая строчка, которая запускает **named**(8). Всё остальное лишь дополнения: "if" проверяет существует ли **named** программа там, где она должна быть, а **echo** строчка выдаёт на экран, что программа **named** запускается, при загрузке системы. Большинство серверов, запускаемых из `rc.inet2`, загружаются блоками вроде этого; простая проверка того, есть ли какие-то серьёзные причины, чтобы не запускать их, вывод информации о том, что они запускаются, и затем команды, загрузки самих сервисов. Опять таки, `rc.inet2` достаточно подробно прокомментирован; погрузитесь в него на какое-то время, и вы найдёте много полезной и интересной информации.

4.2.6 NFS (Сетевая Файловая Система)

Сетевая Файловая Система используется в основном, чтобы разделять файлы в сети для общего использования. Здорово в NFS то, что она разработана

[11] демон от английского daemon, что является аббревиатурой, а не сказочным персонажем. Прим. переводчика.

таким образом, что одна машина может монтировать разделяемые ресурсы другой и обходиться с ними, как с локальными файлами.

Для настройки NFS необходимо проделать несколько действий. Первым делом соответствующие сервисы должны быть запущенны на сервер-машине: **potmap**(8), **nfsd**(8) и **mountd**(8). Второе — сервер должен соответствующим образом "экспортировать" дерево файловой системы клиенту, что осуществляется через **exports**(5) файл из `/etc`.

Первая часть выполняется установкой `tcpip1.tgz` пакета (из N раздела) и добавкой разрешения выполнения для `rc.inet2`. `/etc/exports` немного более забавен.

Предположим, у нас есть каталог с образами на battlecat.tdn, который мы хотим подключить к "ninja.tdn". На battlecat, нам нужна строчка в `/etc/exports` которая выглядит следующим образом:

/var/media/images ninja.tdn(ro)

Теперь на ninja мы можем просто задать команду:

#mount -t nfs battlecat.tdn:/var/media/images /mnt,

чтобы подключить каталог образов к локальному каталогу **/mnt**. К сожалению, я запретил сам себе запись в разделяемый каталог — указанием параметра "(ro)" в фале `/etc/exports` машины **battlecat**, который означает "только для чтения". Все такие параметры следует указывать после имени клиента, в скобках, разделённые запятыми. Например:

/var/media/images ninja.tdn(rw,no_root_squash)

"rw" означает "чтение-запись" (см.**exports**(8) man страницу для получения подробностей), пользователям с ninja разрешается запись в разделяемый каталог. Мне не нравится squash, так что я думаю, что оставлю его вам для самостоятельного изучения через его **man** страницу; если вы планируете много работать с NFS, то **exports**(8) станет вашим лучшим другом.

4.2.7 tcp_wrappers

tcp_wrappers — это основная система предотвращения (а так же разрешения) доступа к сервисам со специфических хостов. Он работает следующим образом: **inetd** (интернет суперсервер) запускает много серверов, большинство из которых "окутаны" (от английского wrap) программой **tcpd**. Другими словами, **tcpd** на самом деле запускает эти серверы, но **inetd** не знает этого (или, точнее его это не интересует). **tcpd** логирует попытки соединения, и затем проверяет фалы `/etc/hosts.allow` и `/etc/hosts.deny`, чтобы проверить, разрешать ли доступ к запрашиваемому сервису.

Правила, определённые в этих файлах могут быть довольно сложными, но давайте предположим, что pyramid.tdn действительно насаждает и не хочет оставить бедную маленькую машину mojo.tdn в покое. На mojo.tdn можно прописать следующую строчку в `/etc/hosts.deny`:

ALL: pyramid.tdn

Смысл этой строчки должен быть очевиден: она перекрывает доступ хосту pyramid ко всем сервисам на mojo, которые защищены при помощи **tcpd**. Если меня в добавок к pyramid раздражает весь домен .annoying.domain, я могу записать строчку:

ALL: pyramid.tdn, .annoying.domain

Подождите! Мой друг Hobbes застрял за компьютером в этой .annoying.domain, и я хочу предоставить ему доступ к моему компьюте-

ру (но только не его раздражающим меня дружкам). Это достаточно просто. Оставим `hosts.denny` в покое, и обратимся к **hosts.allow**. Следующая строчка откроет Hobbes-у доступ:

ALL: hobbes.annoying.domain

Для подробностей смотрите **tcpd**(8), **hosts_access**(5) и **hosts_options**(5). Система **tcp_wrappers** гораздо более гибка в настройках, чем приведённые примеры, и очень полезна и эффективна при более подробном изучении.

4.2.8 Итог

В этом разделе вы изучили, основы настройки системы для подключения к сети, как обходиться с конфигурационными файлами, а так же изучили некоторые из основ безопасности. В добавок к этому вы узнали, что такое Сетевая Файловая Система (NFS) и как включить её на вашем компьютере. Подключение системы к сети позволяет вам получить доступ ко всем видам сервисов: почта, новости и вэб-сайты. См. раздел 5.6 для информации о том, как пользоваться некоторыми из основных сетевых программ.

4.3 Система X Window

Система X Window является стандартным GUI (Графическим Интерфейсом Пользователя) для всех UNIX платформ, включая Linux. В отличие от Windows и MacOS, в Linux и Unix, GUI не имеет ничего общего с ядром операционной системы. Это независимые части. Это придаёт системе большую стабильность: если GUI зависает, это не приводит к зависанию всей системы.

Единственная проблема с X, это то, что её достаточно трудно настроить. Тем не менее, в Slackware 7 представлена безнастроечная установка X, Которая использует framebuffer драйвер. А это означает, что вам не надо проходить через процедуру настройки описанную в подразделах xf86config и XF86Setup. framebuffer будет работать на всех VESA 2.0 — совместимых видео картах. Что в свою очередь означает, что все современные видео-карты будут работать в X. Но всё же, framebuffer работает значительно медленнее, чем правильно настроенный под вашу видео-карту X сервер.

Если вы всё же решили использовать framebuffer сервер, то вам надо установить пакет `xxfb.tgz` из раздела программ X. Вам так же необходимо будет выбрать одно из разрешений консоли во время процесса настройки в программе установки. Тем не менее, для большинства пользователей мы всё же рекомендуем пройти через процедуру настройки X.

Если вы решили настроить X в вашей системе, то вам надо следовать инструкциям из раздела 4.3.1 или из раздела 4.3.2. Первый из разделов описывает использование **xf86config**(1) — программу в стиле командной строки, для настройки X. Второй из разделов относится к программе **XF86Setup**(1) — графической версии программы настройки.

4.3.1 xf86config

xf86config это одна из двух программ, для настройки X, доступных в вашей системе. Основная идея довольно проста. Вам следует ответить на серию вопросов с фиксированным числом ответов. Выбирайте ответы, наиболее подходящие вашей системе. После того, как вы пройдётесь через всю программу, она создаст файл `/etc/XF86Config`(5)[12]. И вы будете готовы к запуску X. Если вы ошиблись где-то, вам надо прервать выполнение программы при помощи **control+c** и начать всё с самого начала.

Полезно собрать как можно больше информации о ваших видео карте и мониторе, до запуска **xf86config**. Вы можете получить информацию о видео карте при помощи программы **SuperProbe**:

```
# SuperProbe
```

Её запуск выдаст вам вначале предупреждение о возможном подвисании системы. Если это испугало вас, то выйдете при помощи **control-c** до того, как пяти секундный отсчёт не закончился. Если вы всё же продолжите,то вы получите некоторую информацию о вашей видео-карте:

[12] для XFree86 4.x.x расположение `/etc/X11/XF86Config`. Прим. переводчика.

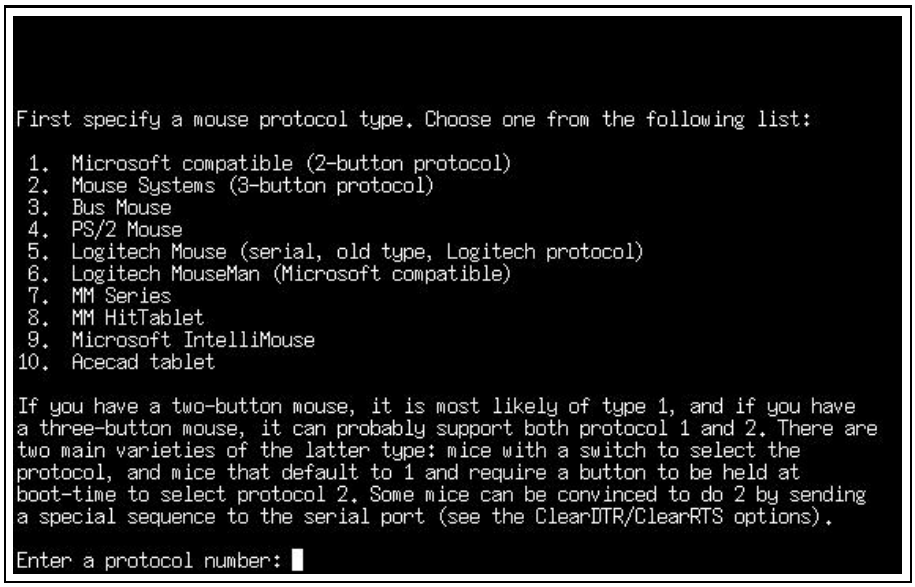
```
First video: Super-VGA
Chipset: ATI 264GT3 (3D Rage Pro) (Port Probed)
Memory: 4096 Kbytes
RAMDAC: ATI Mach64 integrated 15/16/24/32-bit
        DAC w/ clock
        (WITH 8-BIT WIDE Lookup tables)
        (programmable for 6/8-bit wide lookup tables)
Attached graphics coprocessor:
        Chipset: ATI Mach64
        Memory: 4096 Kbytes
```

Вот так выглядит информация для ATI Rage Pro видео-карты. Запишите информацию о вашей видео-карте или переключитесь на другой виртуальный терминал (используя комбинацию клавиш **alt-Fx** (функциональная клавиша), где x — число от 1 до 6) и запустите там **xf86config**. Информация о видео-карте понадобится вам позже. **xf86config** должна быть запущена пользователем root, так как она запишет фалы и создаст символические ссылки в местах, где только у root пользователя есть права на запись:

```
# xf86config
```

Сразу после запуска вы увидите полный экран текста, рассказывающего вам, что программа собирается делать. Помните, что вы не можете возвращаться к предыдущему экрану, если вы сделаете ошибку, так что отвечайте внимательно. Иначе вам придётся повториться несколько раз. Нажмите ввод, как программа вам и предложит.

Mouse protocol (протокол мыши).

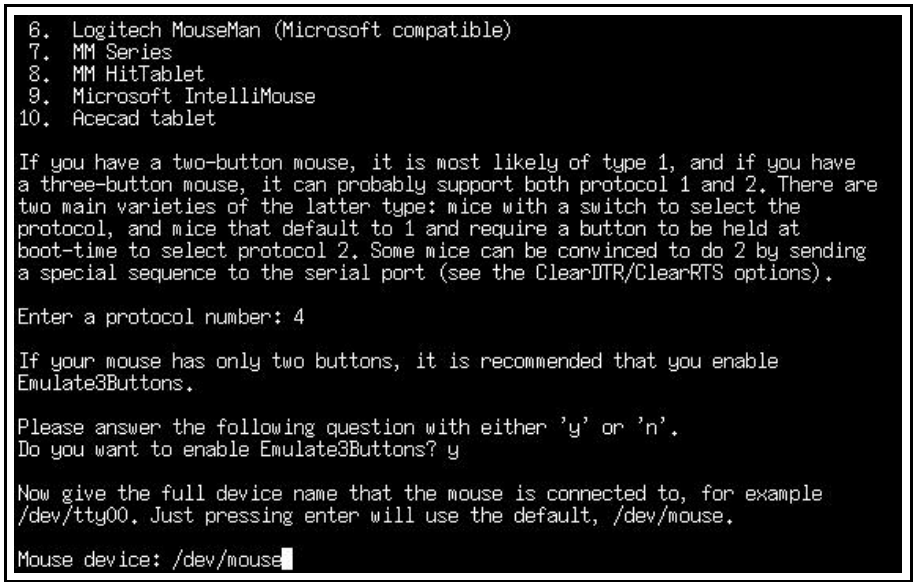


Выберите тип вашей мышки из списка. Большинство компьютеров сегодня поставляются с PS/2 или Microsoft Intellimouse. Более старые мышки, вероятно, потребуют один из более старых протоколов.

Emulate3Buttons (эммулировать 3-ю кнопку).

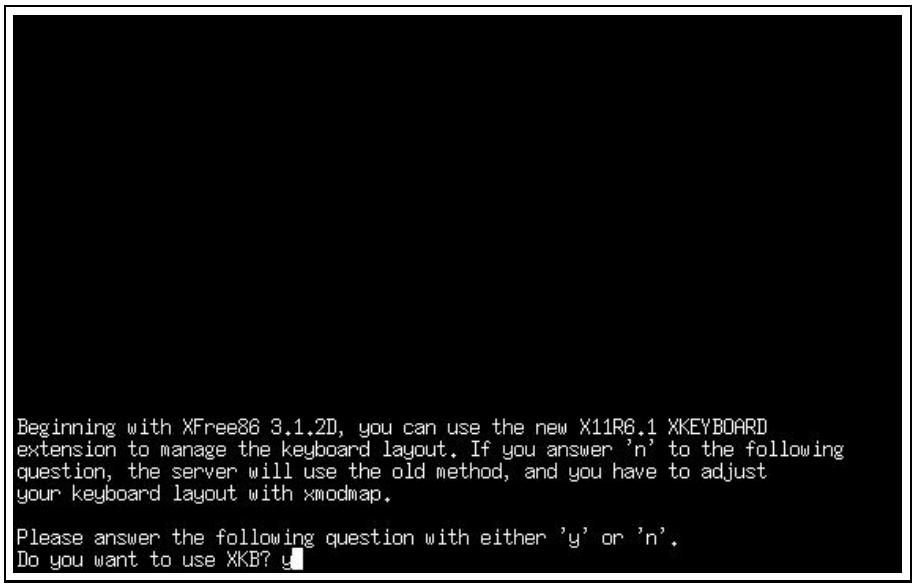
Если у вашей мыши только две кнопки, то вы можете выбрать режим эмуляции третьей кнопки. Нажим на обе кнопки одновременно будет интерпретирован, как нажатие третьей кнопки. Так как много программ используют третью кнопку, рекомендуется подключить её. Если у вас трёх-кнопочная мышь, то эта строчка то ответ на этот вопрос будет проигнорирован.

Mouse device name (имя устройства мышки).



Обычно вариант по умолчанию — `/dev/mouse` сработает здесь. Но если ваша мышь подключена к какому-то необычному порту, вам необходимо ввести что-то другое. Для большинства мышек, подключённых к последовательному или PS/2 портам, подойдёт вариант по умолчанию.

XKEYBOARD extension (расширение XKEYBOARD).

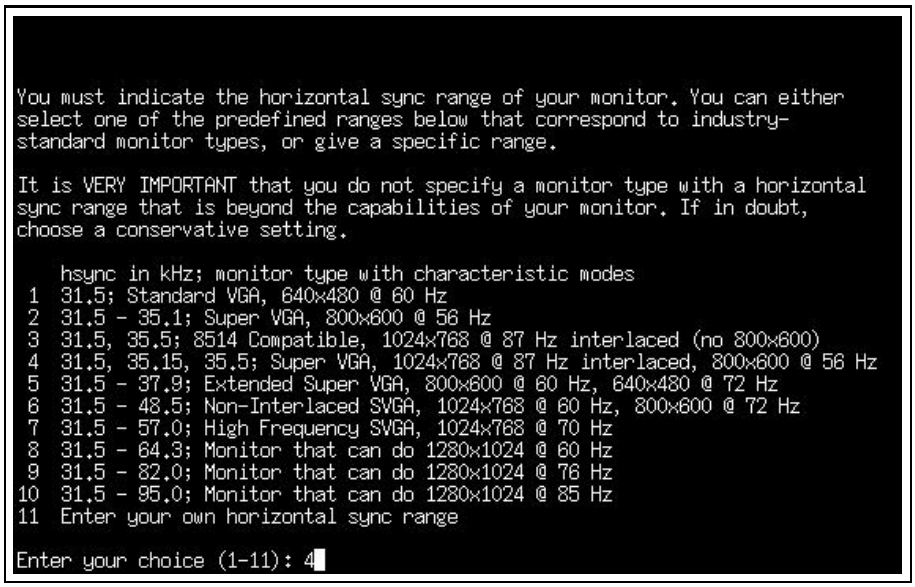


Если вы не выберете эту опцию, то вы получите какое-то странное поведение backspace и delete кнопок. Выбор расширения клавиатуры приведёт к тому, что кнопки будут работать так, как они и должны.

Bindings for alt keys (назначение дополнительных кнопок).

Если вы хотите вводить символы из различных языков, вам следует включить эту опцию. Если вы будете набирать только текст с английскими буквами, то вам нет необходимости включать её.

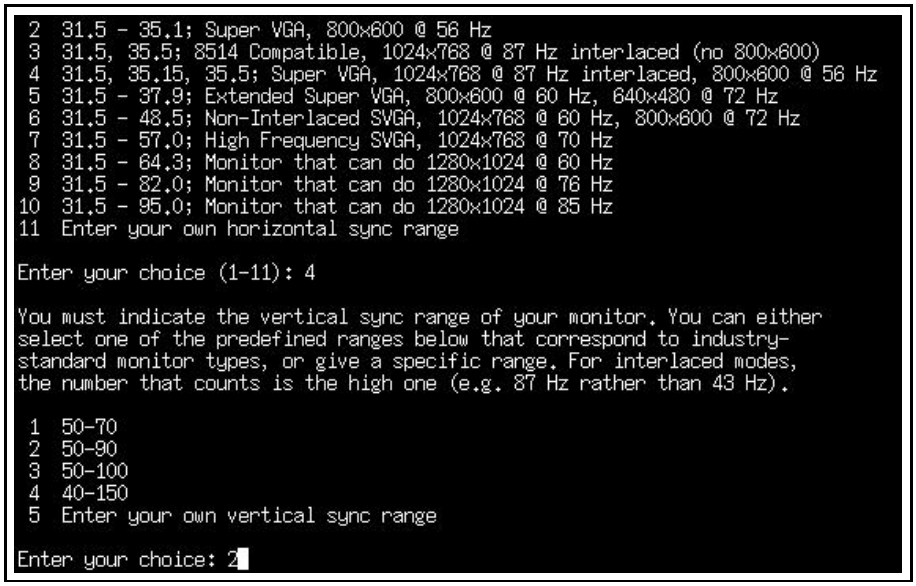
Horizontal sync range (диапазон синхронизации по горизонтали).



Это первый из вопросов о вашем мониторе. Очень важно сделать здесь правильный выбор. Не выбирайте диапазон, выходящий за пределы

спецификации вашего монитора. Это не так важно на новых мониторах, так как у них есть защита и они не вылезут за свои пределы. Более старые мониторы могут быть повреждены, при указании диапазона превосходящего возможности монитора. Так что если вы сомневаетесь, лучше перестрахуйтесь и выберите консервативный диапазон. Очень полезно иметь под рукой документацию от вашего монитора. Для большинства современных мониторов вы можете выбрать 31.5-48.5 или 31.5-57.0. Те из вас, у кого high-end мониторы, могут выбрать один из более широких диапазонов. Или вы можете ввести свой собственный диапазон частот, если вы не можете выбрать подходящий из списка.

Vertical sync range (диапазон синхронизации по вертикали).

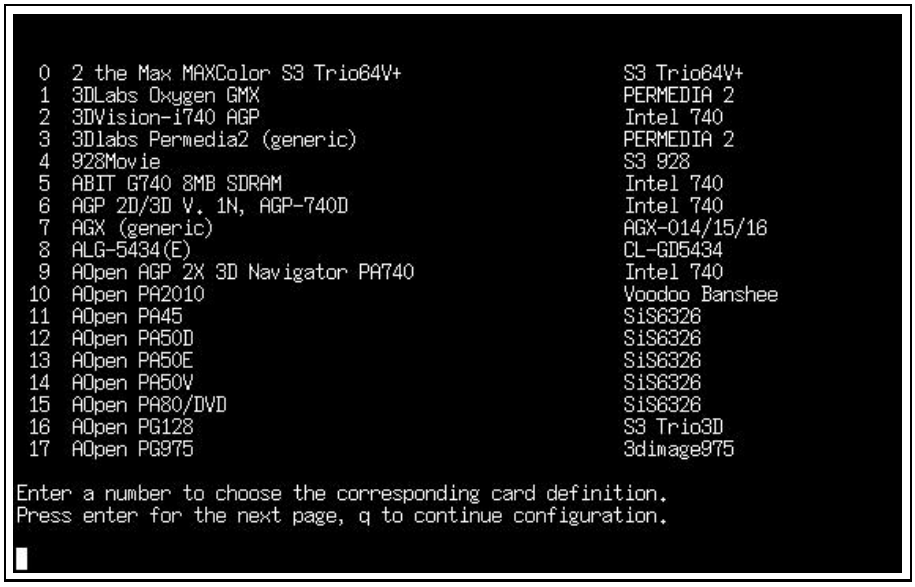


Опять таки, вам надо знать спецификацию вашего монитора для ответа на этот вопрос. Если сомневаетесь, то выбирайте меньший диапазон. Безопасный выбор, наверное, будет 50-90 или 50-100. Если вы не нашли диапазон, подходящий к вашему монитору, вы можете ввести свой собственный.

Identification strings (строка идентификации).

Теперь вам будет предложено три вопроса об идентификации вашего монитора. Эти строчки не очень важны. Вы можете просто пропустить их, путём нажатия ввода. Так же вы можете ввести имена, какие вам заблагорассудится. Эти строчки будут использованы только в конфигурационном файле для идентификации монитора.

Video card database (база данных видео-карт).

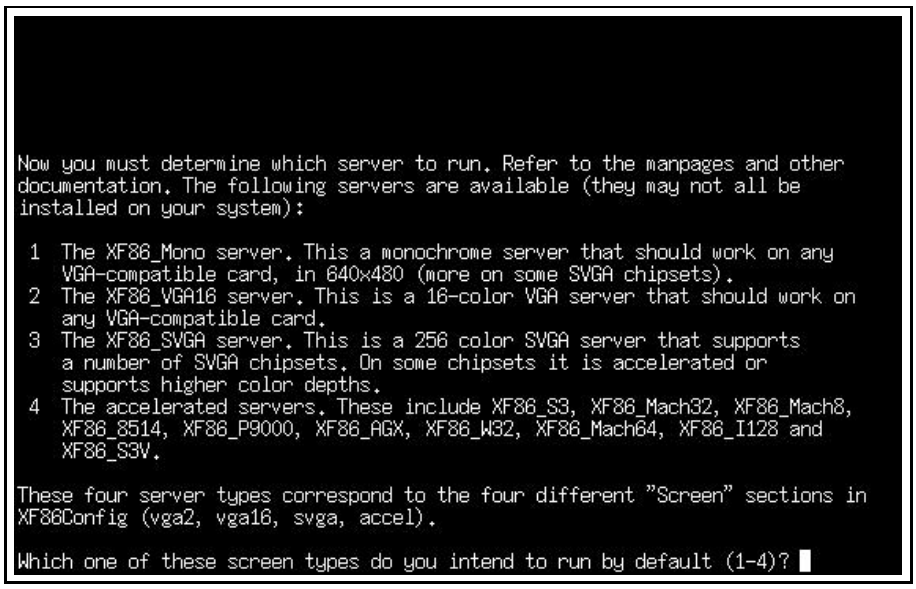


Следующий большой подраздел при настройке X относится к вашей видео-карте. Документация от видео-карты и информация, полученная при помощи **SuperProbe** помогут вам при её настройке. Скорее всего, вы захотите заглянуть в базу данных видео-карт, чтобы выбрать вашу из списка, так что отвечайте "y" здесь. Если вы просто нажмёте ввод, то вы пропустите этот экран и перейдёте к следующему. В базе данных есть около 800 карт. В левом столбце указан номер каждой из видео-карт, и их названия. В правом столбце содержится информация о чипсете, использованном в данной видео-карте. Продолжайте нажимать ввод, пока не найдёте вашу видео-карту. Когда вы найдёте её, введите соответствующий ей номер и нажмите ввод. Если вы не знаете, какой тип видео-карты у вас установлен, вам предоставляется несколько возможностей. Первое — вы можете посмотреть "Chipset" строку из вывода **SuperProbe** и поискать карту с таким чипсетом в базе данных. Так же вы можете воспользоваться "generic SVGA" типом карты. Многие карты, для которых нет их собственного X сервера, поддерживаются SVGA сервером, так что скорее всего, это и будет ваш выбор. После выбора карты, вы получите более детальную информацию. Продолжая использовать ATI Rage Pro для примера, вы получите такое сообщение:

```
Your selected card definition:
Identifier: ATI Mach64
Chipset: ATI-Mach64
Server: XF86_Mach64
Do NOT probe clocks or use any Clocks line.
```

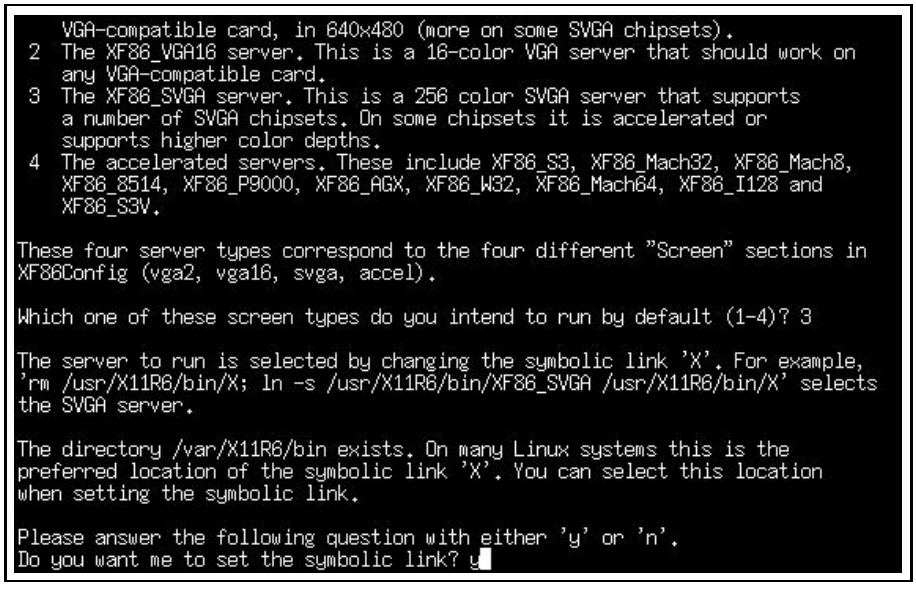
На этом этапе вам следует проверить, что вы установили пакет сервера. XF86_Mach64 сервер находится в пакете `xma64.tgz`. Убедитесь, что необходимый пакет установлен, иначе вам не удастся запустить X.

Which server to run? (какой из серверов запускать?).



Этот вопрос предлагает вам выбрать один из серверов, пригодных для использования. Если вы выбрали видео-карту правильно, просто нажмите ввод здесь. Это скажет X пользоваться сервером, специфичным для вашей видео-карты. Так же вы можете выбрать Mono сервер, VGA16 сервер, SVGA сервер, или accelerated сервер. Лучше всего использовать сервер, указанный для вашей видео-карты.

Setting the symbolic link (установка символической ссылки).



Это установит ссылку на соответствующий сервер.

Video memory (видео память).

Выберите размер имеющейся у вас видео памяти. Вы можете использовать **SuperProbe** для получения этой информации. Если в списке нет подходящего варианта, то вы можете выбрать "Other" и ввести вручную объём вашей видео памяти. Обратите внимание, что объём должен быть указан в килобайтах.

Identification strings (строчки идентификации). Вам предложат ввести три идентификационных строчки. Как и с монитором, вы можете просто нажать ввод три раза, или ввести названия, какие пожелаете.

RAMDAC Вам понадобится выбрать RAMDAC только если вы пользуетесь S3, AGX или W32 серверами. **SuperProbe** скажет вам, какой тип RAMDAC чипа присутствует на вашей видео-карте. Пройдитесь по списку, пока не найдёте нужный вам и введите соответствующий ему номер. Если вы пользуетесь не S3, AGX или W32 сервером, введите "q" для продолжения без выбора RAMDAC.

Clockchip setting (установки clockchip).

Если на вашей карте есть программируемый clockchip, то вам надо выбрать один из предложенных в списке. Учтите, что на большинстве карт нет программируемого clockchip, так что скорее всего вам нужно просто нажать ввод. **SuperProbe** выдаст вам информацию о том, есть ли у вас clockchip, и если есть, то какой именно.

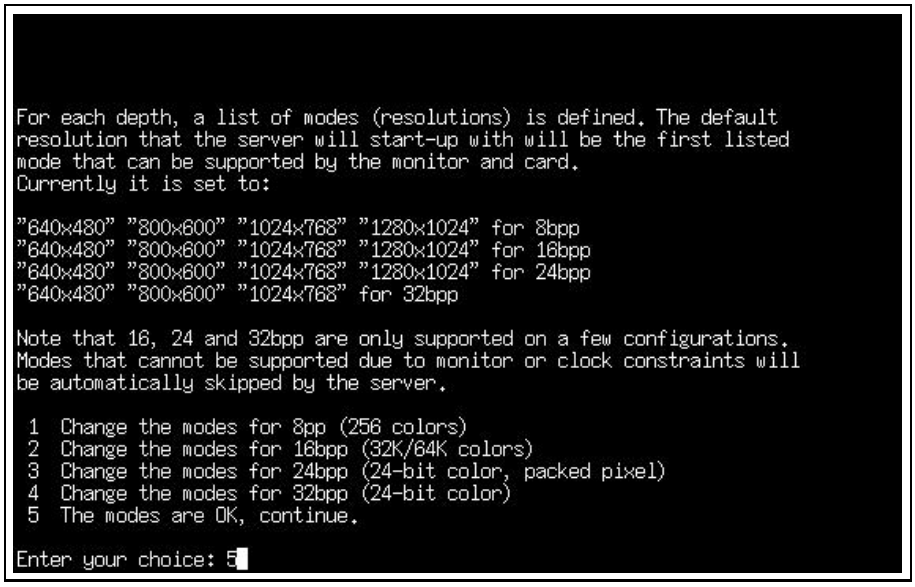
Clocks line (clock строчка).

Следующий экран полон информации о том, что такое "clock line". В соответствии с приведенным объяснением, на современных видеокартах это вам не понадобится. Затем вам будет предложено проверить clock. Вам так же будет сообщено, следует ли проверять это для вашей карты. В случае с ATI картой **xf86config** выдаст:

The card definition says to NOT probe clocks.

Если вы увидите что-то вроде этого, выберите "n". Очень старые графические карты потребуют выполнения этого процесса. **xf86config** подскажет вам, что делать в таком случае.

Video modes (видео режимы).



Ну вот, наступило время выбрать в каких режимах ваш X сервер будет работать. Вы увидите четыре различных глубины цвета — 8bpp, 16bpp,24bpp и 32bpp. Возле каждого есть список различных видео режимов, которые могут быть использованы с этой глубиной цвета. Когда вы запустите X, он запустится с глубиной цвета по умолчанию и с первым из перечисленных для этой глубины разрешений. Если вы захотите изменить разрешение по умолчанию, сейчас самое время сделать это. Если порядок режимов вас устраивает, выберите "Ok". Иначе выберите глубину цвета, для которой хотите сделать изменения. Например, вам предоставлен следующий выбор:

"640x480800x6001024x7681280x1024"for 8bpp
"640x480800x6001024x7681280x1024"for 16bpp
"640x480800x6001024x7681280x1024"for 24bpp
"640x480800x6001024x768"for 32bpp

Если вы хотите, чтобы X запускался в другом разрешении по умолчанию, то вначале выберите глубину цвета, которую хотите изменить. А затем следуйте указаниям программы. Она попросит вас, ввести цифры, соответствующие разрешениям в выбранном вами порядке. Если вы просто хотите поменять порядок разрешений, то вы введёте что-то вроде этого:

Which modes? 5432

Вы так же можете удалить те разрешения, которые захотите. Если ваша видео-карта не может работать при 1280x1024, то нет никакого смысла пробовать этот режим. Вы можете удалить этот режим, ответив:

Which modes? 432

После выбора режимов для этой цветовой глубины, вас спросят, захотите ли вы использовать виртуальный экран, который больше чем ваш физический экран. Виртуальный экран больше вашего монитора.

Когда вы двигаете мышью по виртуальному экрану, она будет "пролистывать" экран, при подводе курсора к краю. Это позволяет разместить больше окон на вашем экране. Но так как вы не можете видеть весь экран сразу, это может стать раздражительным. Но интересно попробовать поиграть с этой опцией. Так что если захотите, то вперёд. Затем вы вернётесь в список видео режимов. После изменения для 24bpp глубины он будет выглядеть следующим образом:

"640x480800x6001024x7681280x1024"for 8bpp
"640x480800x6001024x7681280x1024"for 16bpp
"1280x10241024x768800x600640x480"for 24bpp
"640x480800x6001024x768"for 32bpp

Продолжите изменения для остальных глубин цвета, если пожелаете. Когда завершите, нажмите "Ok" для продолжения.

Write the config file (запись конфигурационного фала).

Ну вот, настройка X завершена **xf86config**, должен ли он записать настройки в файл `/etc/XF86Config`. Если вы захотите пользоваться X, вы вероятно должны ответить "y" на этот вопрос, так как именно там X будет искать файл настроек.

В случае, если вы правильно ответили на все вопросы, и установили пакет X сервера, то теперь вы можете запустить X следующим образом:

```
$ startx
```

Если вы установили KDE или GNOME, то один из них загрузится. Иначе вам надо запустить **xwmconfig** и выбрать, какой из оконных менеджеров (Window managers) вы хотите, чтобы запускался по умолчанию. Оконные менеджеры будут описаны позже в этой части. **xwmconfig** устанавливает оконный менеджер по умолчанию только для пользователя, который запускает программу. Если в вашей системе есть несколько пользователей, каждому из них будет необходимо запустить программу, чтобы выбрать себе оконный менеджер.

Существует несколько комбинаций клавиш, которые весьма полезны при использовании X. Если вам надо выйти из X, и вы не можете сделать это правильно, воспользуйтесь комбинацией клавиш **control-alt-backspace**, которая "убьёт" X и выбросит вас в текстовый режим. Вы можете переключиться на терминалы командной строки, из X, нажав **control-alt-функциональная кнопка**. X запускается на 7-ом терминале, так что вы можете вернуться в X, нажав **alf-F7**. Ну и наконец, вы можете переключать видео режимы, находясь в X. **control-alt-numeric keypad +** переключит вас на следующее более высокое разрешение, в то время как **control-alt-numeric keypad -** переключит на следующее более низкое разрешение.

4.3.2 XF86Setup

Второй способ настройки X — программа **XF86Setup**, графическая программа настройки, которая является частью пакета `xset.tgz`. Вам так же необходимо установить `xvg16.tgz`.

Чтобы запустить **XF86Setup**, войдите в систему как root и выполните:

```
# XF86Setup
```

Если у вас уже есть `/etc/XF86Config` файл (если вы уже настраивали X), то вам будет задан вопрос, хотите ли вы использовать существующий файл. Иначе программа сразу перейдёт в графический режим.

XF86Setup по своей сути очень сходна с **xf86config**. Она задаст вам те же вопросы, так что обращайтесь к предыдущему разделу за разъяснениями. **XF86Setup** содержит много справочной информации сама по себе. Так что у вас не должно возникнуть существенных затруднений.

4.3.3 Сессионные файлы настроек

`xinitrc` и `~/.xinitrc`

xinit(1) — фактически является программой, запускающей X; она выполняется из **startx**(1), так что возможно, вы не заметили этого (и скорее всего это вам знать и не надо). Тем не менее, её конфигурационный файл определяет, какие программы (включая и оконный менеджер) запускать при загрузке X. **xinit** вначале проверяет, есть ли в вашем домашнем каталоге `.xinitrc` файл. Если она находит его, он выполняется, а иначе выполняется `/var/X11R6/lib/xinit/xinitrc` (системный файл по умолчанию). Вот пример простого `xinitrc` фала:

```
# !/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

# merge in defaults and keymaps
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
# start some nice programs
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 из этих "if" блоков используются для подключения разных конфигурационных фалов. Очень скоро мы вернёмся к `.Xresources`, а вот файл `.Xmodmap` мы оставим в покое. Наиболее интересная часть файла расположена в конце, это та часть, где запускаются различные программы. Эта X сессия начнётся с **twm**(1) оконным менеджером, с часами и с тремя терминалами. Обратите внимание на exec в строчке запуска последнего из терминалов. Эта команда говорит, что этот терминал (**xterm**(1)) заменит текущую оболочку (ту, которая запустила **xinit** сценарий). Когда пользователь выйдет из этого **xterm**, X сессия закончится.

Если вы хотите, определить, какие из программ должны быть запущены в X сессии, скопируйте `/var/X11R6/lib/xinit/xinitrc` в `~/.xinitrc` и отредактируйте его, разместив там строчки, запускающие те программы, которые вы пожелаете. Последние строчки моего выглядят вот так:

```
# Start the window manager:
exec startkde
```

Обратите внимание, что есть несколько `xinitrc.*` файлов в каталоге `/var/X11R6/lib/xinit`, которые соответствуют разным оконным менеджерам и GUI-ам. Вы можете пользоваться тем из них, каким пожелаете.

`.Xresources` и `.Xdefaults`

Многие из программ X, для получения различных предпочтений пользователя (цвета, шрифты, и т.д.), используют X Resource Database. Эта база данных обслуживается при помощи **xrdb**(1) программы, которую напрямую,

скорее всего вы никогда не станете запускать. В Slackware она запускается автоматически из `xinitrc`. Файл из которого `xinitrc` указывает программе **xrdb** загружать предпочтения — `/.Xresources`. **xrdb** так же загрузит `/.Xdefaults`. Минимальный файл `.Xresources` выглядит следующим образом:

```
xterm*background: black
xterm*foreground: gray
xterm*scrollBar: true
xterm*font: -*-lucidatypewriter-*-r-*-*-15-*-*-*-*-*-*-*
```

Эти четыре строчки определяют настройки для **xterm** программы. `Xresource` имеет следующую структуру:

```
program*option: setting/value
```

Таким образом содержание `.Xresources` должно быть само-достаточно для понимания. Не пугайтесь строчки с шрифтами; шрифты для X всегда описываются таким способом.

Серверы и оконные менеджеры

Изначально X Window система была разработана, для работы сквозь сеть. Один большой сервер выполняет X программы, а на экран они выводятся на других машинах-клиентах, где угодно в сети. Возможность удалённо выводить на экран программы может иметь много преимуществ. Главные недостатки этой концепции в том, что она менее безопасна, чем локально выполняемые программы, а так же она очень требовательна к сетевым ресурсам. Вы найдёте обсуждение этой концепции ниже в разделе 4.3.5 на стр. 72.

Даже если вы запускаете X на вашем собственном компьютере, вы имеете дело с клиент-сервер моделью. Сервер это часть, которая определяется видео-картой. Когда вы настроили X и сообщили ему, какая у вас видеокарта, вы определили, каким X сервером пользоваться. Клиентская часть — это все остальные программы, которые вы выполняете в X. Специфический клиент, называемый оконным менеджером, ответственен за внешний вид вашей специфической X сессии. Оконные менеджеры детально обсуждаются ниже.

Работа оконного менеджера — обрабатывать рисование окон на экране, с программами внутри этих окон, а так же обрабатывать ввод с мыши и клавиатуры. Первые оконные менеджеры только это и делали. Сегодняшние оконные менеджеры — гораздо более сложные программы, и в них можно настроить практически всё, что вы пожелаете. В них есть все мыслимые опции, которые позволяют сделать ваш рабочий стол отличным от всех других.

Большое количество оконных менеджеров действительно отличает Linux от Windows. В Windows, у вас есть одна основная оконная среда. В Linux вы можете пользоваться одним из многих оконных менеджеров, каждый с различным внешним видом и различными настройками. Некоторые люди назовут это недостатком, так как нет одного стандартного внешнего вида. Но большинство пользователей Linux назовут это достоинством, так как вы можете настроить вашу систему так, как пожелаете.

4.3.4 Выбор рабочего стола

Многие годы Unix использовался исключительно как операционная система для серверов, за исключением мощных профессиональных рабочих станций. Только люди с техническими наклонностями пользовались Unix, как операционной системой, и интерфейс пользователя соответствовал этому факту. GUI были скелетами, разработанными для запуска нескольких необходимых приложений, вроде CAD программ и image render-ов. Управление файлами и системой производилось исключительно из командной строки. Различные производители (Sun Microsystems, Silicon Graphics, и т.д.) продавали рабочие станции с попытками предоставления "look and feel", но большое разнообразие сред разработки для GUI привело к исчезновению единого стандартного рабочего стола. Полоса прокрутки может выглядеть по разному в разных программах. Меню расположены в различных частях окна. В программах встречаются разные кнопки и переключатели. до тех пор, пока пользователями были лишь технические профессионалы, это было не очень важно.

С появлением свободных Unix подобных ОС, и растущим числом различных графических приложений, X стал использоваться в качестве рабочего стола пользователями. Большинство, конечно очаровано внешним видом предлагаемым Microsoft-овским Windows и Apple-овским MacOS; недостаток такого разделения в направлениях X-ориентированных приложений, стал барьером к более широкому использованию программ. В ответ на это появились два проекта с открытым исходником: The K Desktop Environment или KDE, и GNU Network Object Model Environment, известная как GNOME. Каждая из которых имеет широкий спектр приложений, от панелей задач и менеджеров файлов, до игр и офисных пакетов, написанных с теми же GUI toolkit, и сильно внедрёнными, чтобы предоставить универсальный и завершённый рабочий стол.

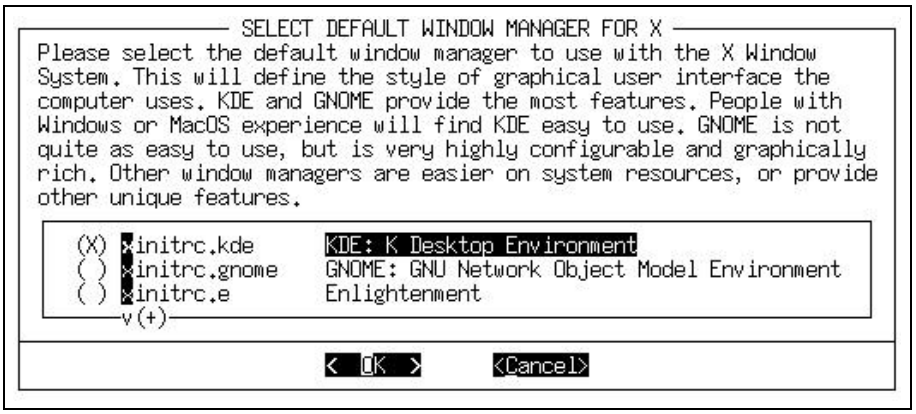
Различия между KDE и GNOME не очень велики. Они выглядят по разному, так как написаны с различными GUI toolkit. KDE основан на Qt библиотеке от Troll Tech AS, в то время, как GNOME основан на GTK, наборе инструментов, изначально разработанном для GNU Image Manipulation Program (или GIMP). Так как проекты независимы, то и разрабатываются они разными дизайнерами и программистами, с разными стилями разработки и с различной философией. Тем не менее, результат в обоих случаях, фундаментально идентичен: полная, тесно интегрированная рабочая среда и набор приложений. По функциональности и внешнему виду, обе среды предоставляют все те же функции, что и другие операционные системы.

Преимущество в том, что эти десктопы бесплатны. Т.е. вы можете получить одну из них, или даже обе на одном и том же компьютере. Выбор за вами.

В добавок к GNOME и KDE, в Slackware есть большая коллекция оконных менеджеров. Некоторые разработаны, как эмуляторы других ОС, некоторые для персональной настройки, другие для скорости. Выбор велик. Конечно, вы вольны установить столько, сколько пожелаете, поиграться с ними и выбрать какой же больше всех остальных вам больше нравится.

Чтобы упростить выбор рабочего стола, Slackware включает в себя программу **xwmconfig**, которая позволяет вам выбрать, какой из десктопов или оконных менеджеров использовать. Итак:

$ xwmconfig



Вы увидите список всех desktop и оконных менеджеров, установленных в вашей системе. Просто выберите один из них, какой захотите. Каждый пользователь в системе должен выполнить эту программу, так как разные пользователи могут использовать разные destop-ы. И возможно, не все захотят пользоваться тем, который установлен по умолчанию, в процессе установки.

А потом просто запустите X:

$ startx

4.3.5 Экспортирование экрана

Как мы отметили выше, возможно запускать X программы на одном компьютере, а выводить их на экран на другом. Это невероятно требовательная к скорости сети процедура, так что вы наверное не захотите делать это через модемное соединение, или через большие расстояния. Так же есть проблема безопасности: экспортирование экрана небезопасно, так как вы позволите всей сети наблюдать за тем, что вы делаете. Всё же это может быть полезно в локальной сети.

Здесь следует определить использование слов "клиент" и "сервер". Когда вы экспортируете экран, можно запутаться в том, кто есть клиент, а кто сервер. Машина, которая выполняет X программы и посылает изображение, будет называться сервером. Машину, на которой изображение будет выводиться на экран, назовём клиентом. Программу, показывающую вещи назовём "сервер", а работающую программу — "сервер".

В этом примере мы будем использовать два компьютера: golf — довольно мощный сервер, расположенный под столом в другом конце забитого людьми и оборудованием кабинета. На нём есть очень много ОЗУ, мощный процессор. В добавок к этому, на нём установлено много X программ, но нет монитора. В другом конце комнаты расположен couch — старый компьютер, с небольшим объёмом ОЗУ и маленьким диском. На нём нет достаточно ресурсов для запуска программы вроде Netscape. couch имеет два существенных преимущества: монитор и он расположен прямо возле шикарного кресла. Вам даже не надо вставать, чтобы пользоваться компьютером. Было бы очень здорово, если бы вы могли запустить Netscape не вставая с кресла. Выход — экспортирование экрана.

Войдите в систему на couch и запустите X. Запустите вашу любимую

терминальную программу (**xterm**, **rxvt**, **eterm**, **aterm**, или любую другую). Первый шаг при настройке удалённого отображения X программ — настройка клиента так, чтобы другие компьютеры могли показывать вывод программ. Для этого используется программа **xhost**. Она контролирует доступ. Если вы в безопасной внутренней сети, вам наверное безразлично, если кто-то может наблюдать, что вы делаете. В этом случае вы просто разрешите всем в этой сети выводить эту информацию на экран:

```
couch $ xhost +
access control disabled, clients can connect from any host
```

С другой стороны, вы можете захотеть сделать это в небезопасной сети (Internet, сеть колледжа, или что то в этом роде). Вы определённо не хотите, чтобы все видели, что вы делаете. **xhost** позволяет вам выбрать тех, кто сможет смотреть, что вы делаете:

```
couch $ xhost + golf.foc
golf.foc being added to access control list
```

Теперь только **golf.foc** (сервер указанный ранее) может выводить информацию на экран компьютера **couch**. вы можете проверить, у кого есть доступ для вывода на экран информации, запустив **xhost** без аргументов:

```
couch $ xhost
access control enabled, only authorized clients can connect
INET:golf.foc
INET:localhost
INET:couch.foc
LOCAL:
```

Ну вот, клиент готов. Следующий шаг — настроить сервер, чтобы он знал, куда выводить изображение вместо монитора. Так как на сервере нет монитора (а значит на нём и X не выполняется), он должен знать, куда выводить изображение.

Настройка сервера так же не сложна. После того, как вы соединились с сервером, вам надо изменить $DISPLAY переменную. По умолчанию, вероятно, она будет пуста. Вам надо присвоить переменной значение, равное удалённому хосту, плюс число, соответствующее тому, на какую X сессию выводить экран. Скорее всего у вас будет только одна X сессия, так что тут никаких проблем возникнуть не должно.

Вот как надо установить $DISPLAY переменную в нашем примере (если на сервере используется Bash, в других случаях суть такая же, но синтаксис будет немного отличен):

```
golf $ export DISPLAY=couch.foc:0.0
```

Это всё, что необходимо сделать для настройки сервера. Теперь, не отсоединяясь, просто запустите X программы оттуда. Весь вывод программ будет послан через сеть на ваш экран.

```
golf $ netscape &
```

Эта команда запустит **netscape** с сервера, но так как переменная DISPLAY указывает на couch, всё будет выводится туда. Вам даже не придётся вставать из-за вашего старого компьютера для того, чтобы запускать программы. Одно важное замечание, которое здесь необходимо сделать: на сервере должны быть установлены все X библиотеки, и другие файлы, необходимые для выполнения программы. В то же время, вам не понадобится `/etc/XF86Config` файл, так как ничего не выводится на экран сервера.

После этого вы можете захотеть отключить экспортирование экрана, убрав сервер из списка доступа на вашем клиенте:

```
couch $ xhost - golf.foc
golf.foc being removed from access control list
couch $
```

Вы видите, как здорово разделять ресурсы компьютеров. Но будьте осторожны, вы можете быть хостом многих X программ для многих удалённых компьютеров, и даже не знать этого.

4.3.6 Итог

В этом разделе вы научились настраивать X Window систему при помощи **xf86config** и **XF86Setup**. Вы так же должны были разобраться в том, что такое окружение рабочего стола, и что такое оконный менеджер, и как выбрать, какой из них запускать. В добавок вы должны знать, как экспортировать экран. Теперь у вас должна быть запущенна работающая графическая среда.

4.4 Загрузка

Настройка загрузчика системы Linux может быть как простой, так и сложной. Многие из пользователей устанавливают только Slackware на свой компьютер. Они просто включают компьютер и он готов к использованию. А некоторым из пользователей, необходимо так же загружать и другие операционные системы для выполнения определённых задач, в связи с чем им необходимо, чтобы обе системы сосуществовали на компьютере.

В этом разделе описывается использование LILO и Loadlin — двух загрузчиков, поставляемых в Slackware. Здесь так же будут описаны типичные варианты "двойной" загрузки и приведены примеры по её настройке.

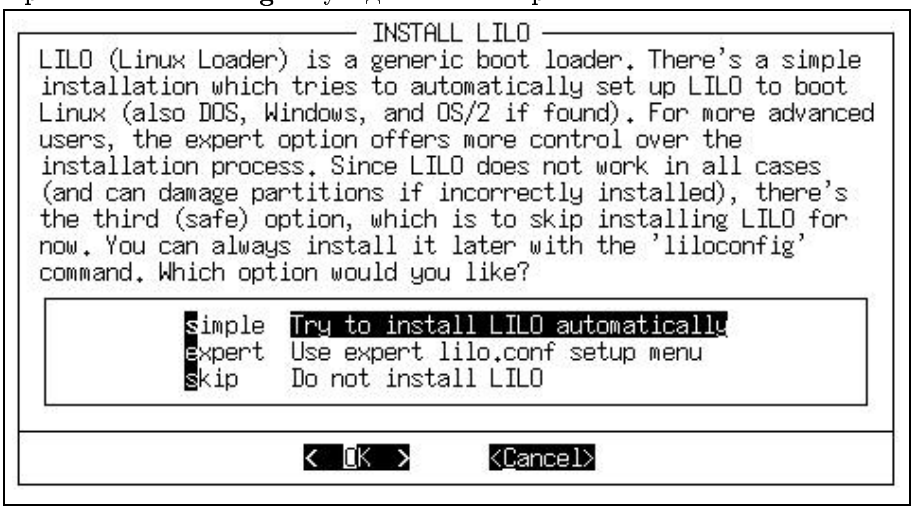
4.4.1 LILO

Linux Loader (Загрузчик Linux), или LILO, является наиболее популярным загрузчиком, используемым в системе Linux. Он достаточно гибок в настройках и может быть использован для загрузки других операционных систем.

Slackware Linux поставляется с меню управляемой программой для настройки LILO, называемой **liloconfig**. Первый раз эта программа запускается в процессе установки системы, но вы можете вызвать её и после установки, напечатав **liloconfig** в командной строке.

LILO читает свои параметры настройки из файла `/etc/lilo.conf`(5). Этот файл не используется при каждой загрузке системы, но используется при каждой установке LILO. LILO должен быть пере установлен в загрузочный сектор после изменения его настроек. **liloconfig** поможет вам построить конфигурационный файл. Если вы предпочитаете редактировать `/etc/lilo.conf` вручную, то пере установка LILO должна быть произведена путём вызова **/sbin/lilo** из командной строки.

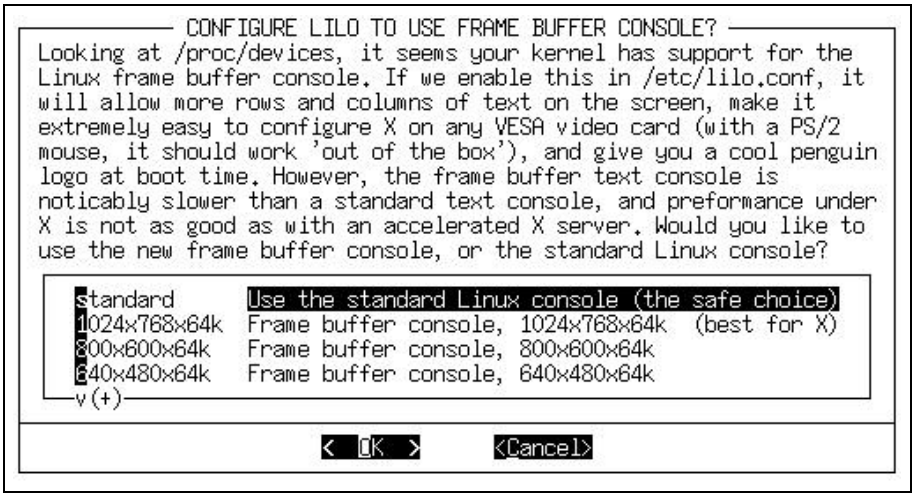
При вызове **liloconfig** вы увидите такой экран:



Если вы устанавливаете LILO впервые, то вам лучше выбрать "simple". В другом случае, вы найдёте, что режим "expert" быстрее, если вы знакомы с Linux и LILO. Выбор "simple" начнёт настройку LILO.

Если поддержка framebuffer-а включена в ядро, **liloconfig** спросит вас,

какое видео разрешение вы хотите использовать. Это же разрешение будет использовано для XFree86 frame buffer сервера. Если вы не хотите запускать консоль в специфическом видео разрешении, то выбор normal сохранит стандартный 80x25 текстовый режим.



Следующая часть настройки LILO — это выбор места, куда вы его установите. Это, вероятно, самый важный шаг. Список, приведённый ниже объясняет подробности выбора.

Root Этот вариант установки расположит LILO в начале вашего корневого каталога. Это наиболее безопасный выбор в случае, если у вас на компьютере есть другая операционная система. В этом случае вы можете быть уверены, что любые другие загрузчики не будут стёрты. Недостаток в том, что LILO может быть загружен отсюда только в случае, если ваш Linux диск является первым в вашей системе.

Floppy Этот вариант ещё более безопасен, чем предыдущий. Он создаст загрузочную дискетку, которую вы сможете использовать для загрузки вашей Linux системы. Загрузчик вообще не располагается на жёстком диске, и когда вы хотите пользоваться Slackware Linux, вы загружаетесь с этой дискеты.

MBR Этот метод рекомендуется в том случае, если Slackware единственная система на вашем компьютере, или если вы решили воспользоваться LILO для загрузки так же и остальных систем на вашем компьютере. `ВНИМАНИЕ: Этот вариант перепишет любой другой загрузчик, находящийся в MBR`

После выбора места для расположения загрузчика, **liloconfig** запишет конфигурационный файл и установит LILO. Вот и всё. Если вы выберите вариант установки "expert", вы получите особое меню. Пункты в этом меню позволят вам пройтись по файлу `/etc/lilo.conf`, добавить другие операционные системы в ваше загрузочное меню, и указать LILO передать особые параметры ядру во время загрузки. Меню "expert" выглядит следующим образом:

Какой бы не была конфигурация системы, установка загрузчика довольно проста. **liloconfig** заметно упрощает процесс установки. Тем не менее, возможны варианты, когда LILO просто не будет работать. Но LILO это не единственный способ загрузки Linux.

4.4.2 LOADLIN

Второй способ загрузки системы, предоставляемый в Slackware это LOADLIN. LOADLIN представляет собой DOS выполняемый файл, который может быть использован для загрузки Linux из работающей DOS системы. Для работы программы необходимо, чтобы ядро Linux находилось на DOS разделе, чтобы LOADLIN мог загрузить его и запустить систему.

При установке системы, LOADLIN устанавливается в домашний каталог root пользователя, как .ZIP файл. Для LOADLIN нет автоматической процедуры установки. Вам необходимо будет скопировать ядро Linux (`/vmlinuz`) и LOADLIN файл из домашнего каталога root-а на DOS раздел.

LOADLIN полезен, если вы хотите сделать загрузочное меню в DOS разделе. Меню может быть добавлено в `AUTOEXEC.BAT` файл, что позволит выбирать, какую из систем (DOS или Linux) загружать при включении компьютера. Если вы выберите Linux, меню вызовет LOADLIN, для запуска Slackware. Ниже приведён пример `AUTOEXEC.BAT` файла для Windows 95, с меню выбора системы:

```
@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=<root partition device> RO
GOTO END
:END
```

Вам необходимо будет указать имя устройства вашего корневого раздела. Что-то вроде `/dev/hda2`. Вы так же всегда можете воспользоваться **LOADLIN** из командной строки. Вам надо указать параметры программе, по аналогии с приведенным выше примером. Документация LOADLIN предлагает вам много различных примеров использования программы.

4.4.3 Двойная загрузка

Многие пользователи настраивают свои компьютеры для загрузки Slackware Linux и ещё какой-то другой операционной системы. Ниже описаны несколько типичных сценариев двойной загрузки.

Windows 9x/DOS

Настройка компьютера для загрузки, как Windows 9x, так и Linux, вероятно наиболее часто встречающийся сценарий двойной загрузки. Есть много способов реализации такой загрузки. Ниже будут приведены два из возможных вариантов.

Часто при настройке двойной загрузки системы составляется замечательный план того, как всё должно происходить, но в процессе настройки ошибаются порядком. Очень важно понимать, что операционные системы должны быть установлены в определённом порядке, для того чтобы двойная загрузка работала. Linux предоставляет больше контроля над управлением содержимого MBR. В связи с этим настоятельно рекомендуется устанавливать Linux в последнюю очередь. Вначале должна быть установлена Windows, так как она всегда переписывает MBR своим загрузчиком в процессе установки.

Использование LILO Большинство пользователей предпочитают использовать LILO для выбора системы в процессе загрузки. Как и предложено выше, Windows должен быть установлен первым.

Предположим, что у вас есть единственный 47GB IDE жёсткий диск в системе. Так же мы предположим, что вы хотите использовать половину диска для Windows, и половину для Linux. В этом случае появляется проблема с загрузкой Linux. Я не знаю, какая будет геометрия у такого диска, но скорее всего, после 23.5GB вы выйдете за предел в 1024-ого цилиндра. Лучшее расположение разделов для такой системы предложено ниже:

1GB	Windows boot	(C:)
1GB	Linux root	(/)
22.5	Windows misc	(D:)
22.5	Linux /usr	(/usr)

Не забудьте так же, что вам понадобится пространство для раздела подкачки Linux. Неписанное правило гласит, что размер раздела подкачки должен в 2 раза превышать объём оперативной памяти вашей системы. Для машины с 64Мб памяти нужен раздел подкачки в 128Мб и т.д.

После разметки диска, вам следует установить Windows. После того, как он установлен и работает, вам следует установить Linux. Установка LILO потребует дополнительного внимания. Вам надо будет выбрать "expert" вариант установки LILO.

Начните новую настройку LILO. Вам надо выбрать вариант установки в MBR, чтобы вы могли использовать LILO для выбора операционной системы. Затем из меню добавьте ваш Linux раздел и затем ваш Windows или DOS раздел. После того, как это сделано, выберите "install LILO"

Пере загрузите компьютер. LILO должен загрузиться и ждать реакции пользователя. Вы можете нажать **Alt** для получения приглашения boot:. Введите имя системы, которую вы желаете загрузить (эти имена были выбраны при установке LILO). Если вы не помните, какие имена указали, то нажмите **Tab** для получения списка доступных для загрузки ОС.

Вы можете настроить LILO более детально, путём редактирования `/etc/lilo.conf` файла. Вы можете настроить его таким образом, чтобы при загрузке выводилось текстовое меню, и чтобы всегда выводилось приглашение. Например, если я хочу, чтобы LILO выводил на экран такое приглашение:

```
System Boot Menu
================
1 - Linux
2 - Windows

LILO boot:
```

То мой `/etc/lilo.conf` будет выглядеть следующим образом:

```
# LILO configuration file
boot = /dev/hda
vga = normal
message = /boot/message
image = /vmlinuz
root = /dev/hda2
label = 1
read-only

other = /dev/hda1
label = 2
table = /dev/hda
```

А мой `/boot/message` файл будет выглядеть вот так:

```
System Boot Menu
================
1 - Linux
2 - Windows
```

LILO очень гибкий в настройках системный загрузчик. Он не ограничен загрузкой только Linux или DOS. Он может загрузить практически любую систему. Обратитесь к man страницам для **lilo**(8) и `lilo.conf`(5) для получения более подробной информации.

А что делать, если LILO не работает? Существуют определённые конфигурации, при которых LILO просто не будет работать на определённом компьютере. Но к счастью есть и другие способы реализации двойной загрузки Linux и Windows.

Использование LOADLIN Этот способ может быть использован в случае, если LILO не работает в вашей системе или просто, если вы не хотите устанавливать LILO. Этот способ идеален для тех пользователей, которые часто пере устанавливают Windows. При каждой установке Windows, он перепишет MBR, и таким образом уничтожит LILO оттуда. С LOADLIN вы избежите этой проблемы. Большой недостаток в том, что вы можете использовать LOADLIN только для загрузки Linux.

При использовании LOADLIN вы можете установить системы в любом желаемом порядке. При установке Linux будьте внимательны и не перепишите MBR. Лучше всего, просто пропустите установку LILO.

После установки операционных систем, скопируйте файл `loadlinX.zip` (где "X" — номер версии, например "16a") из домашнего каталога root пользователя на ваш Windows раздел. Так же скопируйте туда образ вашего ядра. Вам надо проделать это из Linux. Вот пример того, как проделать это:

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

Это создаст каталог `C:\LINUX` на вашем Windows разделе (мы предполо-

жили выше, что это `/dev/hda1`) и создаст там копии файлов, необходимых для LOADLIN. Затем вам следует пере-загрузиться в Windows для создания загрузочного меню.

После загрузки Windows, войдите в командную строку DOS. Нам необходимо убедиться в том, что система настроена так, чтобы не загружать графический интерфейс при включении.

```
C:\> cd \
C:\> cattrib -r -a -s -h MSDOS.SYS
C:\> cedit MSDOS.SYS
```

Добавьте туда такую строчку:

```
BootGUI=0
```

Сохраните файл и выйдите из редактора. Теперь надо добавить меню в `C:\AUTOEXEC.BAT`. Ниже приведён пример такого блока в `AUTOEXEC.BAT`:

```
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
```

Ключевая строчка здесь та, что загружает LOADLIN. Мы сообщили ей, какое ядро загружать, где находиться корневой каталог Linux, и что мы хотим, чтобы он был подключён только для чтения на начальном этапе загрузки.

Все инструменты, необходимые для обоих этих методов поставляются с Slackware Linux. На рынке существует огромное количество загрузчиков, но этих инструментов должно быть достаточно практически для любого варианта двойной загрузки.

Windows NT

Это второй из наиболее распространённых вариантов двойной загрузки. С Windows NT возникает несколько дополнительных проблем, по сравнению с Windows 9x. Самая основная в том, что если LILO записать в MBR, то Windows NT не сможет загрузиться. Поэтому приходится использовать загрузчик, поставляемый с Windows NT. Ниже приведён список необходимых

шагов по настройке Windows NT и Linux для двойной загрузки.

1. Установите Windows NT
2. Установите Linux, расположив LILO в суперблок Linux раздела
3. Прочтите первые 512 байт корневого раздела Linux и запишите их на Windows NT раздел.
4. Отредактируйте `C:\BOOT.INI` из Windows NT, добавив опцию загрузки Linux

Установка Windows NT должна быть довольно проста, как и установка Linux. Затем начинаются небольшие трюки. Получение первых 512-ти байтов вашего Linux раздела на самом деле не так сложно, как звучит. Для этого вам надо находиться в Linux. Предположив, что ваш Linux раздел это `/dev/hda2`, мы выполним команду:

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

Ну вот и готово. Теперь надо скопировать `bootsect.lnx` в ваш Windows NT раздел. Вот вам ещё одна проблема. Linux не содержит стабильной поддержки записи для файловой системы NTFS. Если при установке Windows NT вы отформатировали её раздел, как NTFS, то вам понадобится скопировать этот файл на FAT дискету, а затем в Windows NT прочесть его. Если же вы разметили диск Windows NT, как FAT, то вы можете просто смонтировать этот диск в Linux и скопировать файл на него. В любом случае, вам надо скопировать файл `/tmp/bootesct.lnx` с Linux диска в `C:\BOOTSECT.LNX` на Windows NT диске.

Последний шаг — добавление пункта в загрузочное меню Windows NT. Из Windows NT откройте режим командной строки.

**C:\WINNT> cd **
C:\> attrib -r -a -s -h boot.ini
C:\> edit boot.ini

Добавьте такую строчку в конец файла:

C:\bootsect.lnx="Slackware Linux"

Сохраните изменения и выйдите из редактора. После пере загрузки Windows NT в загрузочном меню появится пункт "Slackware Linux". Если вы выберете его, то загрузиться Linux.

Linux

Это определённо, самый простой сценарий двойной загрузки. Используя LILO, просто добавьте ещё пункт в `/etc/lilo.conf` файл. Вот и всё.

4.4.4 Итог

В этом разделе были обсуждены различные варианты загрузки системы. Использование LILO и Loadlin. Была обсуждена загрузка Linux в паре с другой операционной системой. Вы должны были усвоить, как настроить систему для загрузки, как только Linux, так и варианты двойной загрузки с системой, которую вы выберите.

Глава 5

Использование Slackware Linux

5.1 Оболочка Shell

В графической среде интерфейс представляется программой, которая рисует окна, линии прокрутки, меню, и т.д. В режиме командной строки интерфейс управляется оболочкой (англ. shell), которая интерпретирует команды и вообще делает вещи пригодными к применению. Сразу после входа в систему (который описывается в этом разделе), пользователи попадают в оболочку, и могут делать там то, что они хотят. Этот раздел является введением по оболочке в общем, и так же описывает наиболее распространённую среди пользователей Linux оболочку — Bourne Again Shell, или просто **bash**. Для получения дополнительной информации по всем рассмотренным в этом разделе вопросам, обращайтесь к man странице **bash**(1).

5.1.1 Пользователи

Вход в систему

Итак, вы загрузились, и перед вами что-то вроде:

```
Welcome to Linux 2.2.14
darkstar login:
```

Хм...никто нам ничего не говорил о login. И что такое darkstar? Не волнуйтесь; скорее всего, вы не соединились через гиперпространственную comm-link связь с искусственной луной Империи. (Боюсь, гиперпространственный comm-link ещё не поддерживается Linux ядром :) Нет, drackstar это просто имя одного из наших компьютеров, и его имя устанавливается по умолчанию. Если во время установки вы изменили имя вашего компьютера, то вы увидите его в приглашении входа, вместо drackstar.

Так что же там про login... Если это ваш первый вход в систему, вам необходимо войти в систему, как root пользователь. Вас спросят пароль; если вы указали пароль при установке, то введите его. Если нет, просто нажмите ввод. Ну вот, вы вошли в систему!

Root: суперпользователь

Итак, кто или что такое "root"? И что он имеет общего с системой?

Ну, в мире Unix и аналогичных системах (таких, как Linux), есть пользователи и ещё пользователи. Мы вернёмся к более подробному обсуждению позже, но важно знать, что пользователь root, это самый главный пользователь; это всезнающий и все-умеющий пользователь и никто не смеет ослушаться "root" пользователя. Это просто не дозволенно. Root это тот, кого мы называем суперпользователем и имеем на это полное право. И что самое лучшее, так это, что вы и есть root.

Здорово, не правда ли?

Если вы не уверенны: да, это очень здорово. Что вы должны усвоить, так это то, что root может поломать всё, что угодно, если пожелает. Возможно, вы захотите заскочить вперёд и заглянуть в раздел 5.5 на стр. 108 и посмотреть, как добавить пользователя; и войти, как этот пользователь. Мудрость гласит, что лучше всего становится суперпользователем только когда это абсолютно необходимо, чтобы избежать случайных повреждений в системе.

Кстати, если вы вошли в систему, как обычный пользователь, и вам надо выполнить пару команд, как root, вы можете воспользоваться командой **su**(1). Вас спросят пароль root-а и затем вы им станете, до тех пор, пока не выйдете. При помощи **su** вы так же можете стать любым другим пользователем вашей системы, если знаете его пароль. Например, **su logan** сделает вас мной.

5.1.2 Командная строка

Запуск программ

Трудно сделать что либо, без выполнения программ; можно использовать ваш компьютер в качестве подпорки для чего-то, например, для открытой двери, и он будет замечательно жужжать, пока включён, но не больше. И наверное, большинство согласиться со мной, что использование компьютеров в качестве жужжащей подпорки для двери не принесло бы им такой популярности, какую они сейчас приобрели.

Итак, помните, что в Linux почти всё является файлом? Так вот, для программ это тоже справедливо. Каждая команда, которую вы выполняете (если она не встроена в вашу оболочку), соответствует файлу. Вы можете запустить программу, указав полный путь к ней.

Например, помните команду **su**? Так вот, на самом деле она является файлом, расположенным в `/bin` каталоге: **/bin/su** запустит её.

Почему же тогда простой набор **su** так же работает? Вы ведь не указывали путь `/bin`. Ведь она могла находится и в `/usr/share`? откуда же компьютер узнал? Ответ в переменной PATH; большинство оболочек имеет PATH или что-то похожее. Она содержит список каталогов, в которых искать программы, которые вы пытаетесь выполнять. Таким образом, когда вы выполнили **su**, ваша оболочка прошлась по списку каталогов, ища в каждом из них выполняемый файл **su** и как только она нашла такой файл, она выполняет его. Это случается каждый раз, когда вы запускаете какую либо программу без указания полного пути к ней; если вы получаете сообщение "Command not found", это означает, что программы, которую вы пытаетесь запустить нет в каталогах, перечисленных в PATH переменной. (Это так же будет истиной, если программа вообще не существует...) Мы обсудим переменные окружения более детально в разделеthe Bourne Again Shell (bash).

Запомните так же, что "." это сокращение для "каталог, в котором я сейчас нахожусь", так что если вы в каталоге `/bin`, **./su** сработает, как полный путь к файлу.

Wildcard Matching (шаблоны имён)

Практически любая оболочка позволяет использовать некоторые символы, подразумевая, "здесь может быть всё что угодно". Такие символы называются "wildcards"; наиболее распространённые из них это "*" и "?". По соглашению, "?" обычно заменяет любой отдельный символ. Например, допустим у вас в каталоге есть файлы: `example1.txt`, `example2.txt` и `example3.txt`. Вы хотите скопировать все эти файлы (при помощи **cp** программы, которую мы опишем в разделе 5.3 на стр. 95 в другой каталог, скажем `/tmp`.

Набирать **cp example1.txt example2.txt example3.txt /tmp** потребует слишком много усилий по печатанию всего этого текста. Гораздо проще написать **cp example?.txt /tmp**; "?" будет заменён на все встретившиеся символы "1", "2" и "3".

Что вы говорите? Всё равно слишком много надо печатать? Вы правы. Это ужасно; ведь у нас есть закон о труде, который защищает нас от таких ситуаций. Но к счастью, у нас в арсенале так же есть "*". Как уже упоминалось выше, "*" заменяет любое число символов, включая их отсутствие. Так что в случае, если кроме упомянутых выше файлов в каталоге ничего нет, мы можем просто сказать **cp * /tmp** и убьём их все одним выстрелом. Предположим теперь, что в том же каталоге есть файлы `example.txt` и `hejaz.txt`. И мы хотим скопировать фалы `example`, но не `hejaz.txt`; **cp example*.txt /tmp** сделает это для нас.

cp example?.txt /tmp, конечно, скопирует только наши первые три файла; в файле `example.txt` нет символа, подходящего под "?", так что этот файл будет оставлен в покое.

Пере-направление ввода/вывода и piping

(Что-то очень интересное здесь.)

$ ps > blargh

Знаете, что это такое? Это я выполняю **ps**, чтобы посмотреть какие процессы сейчас выполняются; **ps** описана в разделе 5.4. Это не самая интересная часть. Интересная часть вот тут **> blargh**, что грубо означает, " возьми вывод **ps** и запиши его в файл, называющийся `blargh`". Но подождите, будет ещё интереснее.

$ ps | less

Эта команда берёт вывод **ps** и "pipes" (прокачивает) его через **less**, таким образом, я могу пролистывать его как захочу.

$ ps >> blargh

Это третий из наиболее используемых redirector-ов (пере-направителей); он делает то же, что и ">", только ">>" добавит вывод **ps** к файлу `blargh`, если этот файл существует. А если такого файла нет, то сработает просто как ">", и создаст файл. (">" в любом случае полностью перепишет содержимое `blargh`.) Есть так же и оператор "<", который означает "возьмите ввод из следующего источника", но он не так часто используется.

$ fromdos < dosfile.txt > unixfile.txt

Пере-направление становится действительно забавным, когда вы объединяете операторы:

$ ps | tac >> blargh

Эта команда запустит **ps**, инвертирует порядок вывода строк, и присоединит их к фалу `blargh`. Вы можете комбинировать столько операторов, сколько пожелаете; но будьте осторожны и помните, что они интерпретируются с лева направо.

Загляните в man страницу **bash**(1) для получения более детальной информации по пере-направлению вывода.

5.1.3 The Bourne Again Shell (bash)

Переменные окружения

Система Linux это сложный зверь. И есть много вещей, за которыми надо следить, много маленьких деталей, которые вступают в игру в вашем обычном взаимодействии с различными программами (о некоторых из которых вам даже не надо знать). Никто не хочет указывать большие наборы опций программе, которую хочет выполнить, указывать ей, какой тип терминала используется, имя хоста компьютера, как должно выглядеть приглашение программ...

Итак, как объединяющий механизм, пользователи получают то, что называется environment (окружение). Среда окружения определяет условия, в которых выполняется программа; некоторые из этих условий — переменные; пользователь может изменять их и играть с ними. Практически каждая оболочка имеет переменные окружения (если нет, то это наверное не очень удобная оболочка). Здесь мы приведём команды используемые в **bash** для оперирования с переменными окружения.

$ set

Сама по себе команда **set** покажет вам все переменные окружения, которые определены в данный момент вместе с их значениями. Как и большинство команд, встроенных в **bash**, она может быть использована для других целей (при указании параметров); мы оставим изучение подробностей, так как они описаны в man странице **bash**(1). Отрывок вывода команды **set** на моём компьютере выглядит следующим образом:

```
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
```

Обратите внимание, на PATH переменную, которую мы обсуждали ранее; я могу выполнять любые программы, находящиеся в каталогах, упомянутых в ней, просто набрав их имя, без указания пути.

$ unset VARIABLE

unset удалит как содержание указанной в команде переменной, так и саму переменную; **bash** забудет, что такая переменная когда либо существовала. (Не волнуйтесь, если это не что-то, что явно было определено в данной сессии оболочки, вероятно оно будет пере-определено при следующей сессии оболочки.)

$ export VARIABLE=some_value

Теперь **export**, действительно очень полезная команда. С её помощью вы присваиваете переменной VARIABLE значение "**some_value**; если

VARIABLE не существовала, то она будет создана. Если VARIABLE уже имела какое-то значение, то оно будет потерянно и изменено на новое. Это не есть хорошо, если вы просто пытаетесь добавить имя каталога в PATH. В этом случае вам понадобится команда вроде этой:

$ export PATH=$PATH:/some/new/directory

Обратите внимание на $PATH: когда вы хотите, чтобы **bash** интерпретировала переменную (заменяла её на её значение), добавьте $ перед именем переменной. Например, **echo $PATH** высветит значение PATH переменной, в моём случае:

$ 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 по tab

(Опять что-то интересное.)

1. Интерфейс командной строчки требует много печатания.
2. Печатание это работа.
3. Никто не любит работу.

Из пунктов 3 и 2 мы можем заключить, что 4) никто не любит печатать. К счастью, **bash** спасает нас от 5 (никто не любит интерфейс командной строки).

Как же **bash** справляется с этим, спросите вы? В дополнение к шаблонам имён, обсуждённым выше, **bash** имеет функцию "завершение по tab".

Завершение по tab работает примерно так: Вы набираете имя файла. Может он есть в PATH, а может быть в текущем каталоге. Всё что вам надо, это напечатать достаточную часть имени файла для его однозначной идентификации. Затем нажмите tab кнопку. **bash** определит, чего вы хотели и дополнит имя файла за вас!

Время для примера. Каталог `/usr/src` содержит два подкаталога:`/usr/src/linux` и `/usr/src/sendmail`. Я хочу посмотреть, что находится в `/usr/src/linux`. Я набираю только **ls /usr/src/l**, нажимаю кнопку **TAB**, и **bash** выдаст мне **ls /usr/src/linux**.

Теперь, предположим, есть два каталога `/usr/src/linux` и `/usr/src/linux-old`; Если я напечатаю **/usr/src/l** и нажму **TAB**, **bash** дополнит столько, сколько сможет, и я получу /usr/src/linux. Я могу остановиться на этом, или нажать **TAB** опять и **bash** покажет мне список каталогов, которые подходят под то, что уже напечатано.

Итак, меньше печатания (и пользователям может понравиться интерфейс командной строки). Я же говорил, что это здорово.

5.1.4 Виртуальные терминалы

Итак, вы делаете что-то и вам надо сделать ещё что-то. По идее вы можете просто бросить то, что вы делали и перейти к выполнению другой задачи.

Но ведь это же многопользовательская система вроде? И вы можете войти столько раз, сколько пожелаете? Так зачем же выполнять только одну задачу, если параллельно можно выполнять и другую?

Проблема в том, что мы не можем подключить несколько клавиатур, мышек и мониторов к одному и тому же компьютеру; скорее даже большинство из нас просто не захочет этого. Очевидно, что дополнительное оборудование не является решением проблемы. А вот программное обеспечение может помочь, и Linux позволяет это, предлагая "виртуальные терминалы", или "VTs".

Нажав **Alt** и функциональную кнопку, вы можете переключать виртуальные терминалы; каждая функциональная клавиша соответствует терминалу. Slackware имеет 6 виртуальных терминалов по умолчанию. **Alt+F2** переключит вас на второй, **Alt+F3** на третий, и т.д.

Остальные функциональные клавиши зарезервированы для X сессий. Каждая X сессия использует свой собственный VT, начиная с седьмого (**Alt+F7**) и выше. Когда вы находитесь в X, **Alt+Function** комбинация заменяется на **Ctrl+Alt+Function**; так что если вы в X и хотите переключиться на третий терминал (не завершая X сессию), **Ctrl+Alt+F3** переключит вас туда. (**Alt+F7** вернёт вас обратно, если вы используете первую из X сессий.)

5.1.5 Итог

В этом разделе были обсуждены пользователи, оболочка, командная строка и виртуальные терминалы. Вы должны уверенно себя чувствовать при работе в режиме командной строки, выполнении программ и использовании pipe-ов и операторов пере-направления вывода. Так же вы должны усвоить, насколько велик и все-могущественен root пользователь, и почему плохо всегда работать, как root.

5.2 Структура файловой системы

Мы с вами уже обсуждали структуру каталогов в Linux Slackware. Вы умеете искать файлы и каталоги, которые вам нужны. Но файловая система это нечто большее, чем структура каталогов.

Linux это многопользовательская система. Все элементы системы многопользовательские, даже файловая система. Система хранит информацию о том, кому принадлежит файл и кто может читать его. А так же позволяет использовать ссылки и подключение NFS. В этом разделе наряду с перечисленными аспектами разъясняются аспекты многопользовательской структуры файловой системы.

5.2.1 Права собственности

Файловая система хранит информацию о правах собственности для каждого фала и каталога. Включая информацию о пользователе и группе, которым принадлежит файл.Самый простой способ просмотреть эту информацию — воспользоваться **ls**:

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

Нас интересуют третий и четвёртый столбцы. Они содержат информацию о именах пользователя и группы, которым принадлежит файл. В приведённом примере файл принадлежит пользователю root и группе bin.

Владельца файла можно изменить при помощи **chown**(1) (что значит "change owner" — изменить владельца), а группу при помощи **chgrp** ("change group" — изменить группу). Чтобы изменить владельца фала на "daemon", мы выполним команду:

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

А чтобы изменить группу на "root":

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

chown так же может быть использована для задания и пользователя и группы, которым принадлежит файл:

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

Принадлежность фалов — это очень важная часть использования Linux системы, даже если вы единственный пользователь. Иногда вам придётся исправлять права владения файлами и узлами (nodes) устройств.

5.2.2 Права доступа

Права доступа — это ещё одно проявление многопользовательских основ файловой системы. С их помощью вы можете указать, кто может читать, изменять и выполнять фалы.

Права доступа хранятся в виде четырёх восьмеричных чисел, каждое из которых устанавливает права для отдельного блока. Блоки представляют права владельца, права группы и права всего остального мира. Четвёртое восьмеричное число используется для хранения специальной информации, такой как ID владельца, ID группы и "sticky"бит. Значения этих чисел соответствуют правам доступа (им так же сопоставляются буквы, которые выводятся такими программами, как **ls** и могут быть использованы программой **chmod**:

Таблица 5.1: Соответствие восьми-битовых чисел правам

Тип	Восьми-битовое значение	Буква
"sticky" bit	1	t
set user ID (ID пользователя)	4	s
set group ID (ID группы)	2	s
read (чтение)	4	r
write (запись)	2	w
execute (выполнение)	1	x

Для каждой из групп права складываются. Например, если вы хотите, чтобы права группы были "read" и "write", вы воспользуетесь "6" в части соответствующей группе прав доступа.

Права по умолчанию для **bash**:

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

Первый дефис для каталогов принимает значение, равное "d". Три группы прав (владелец, группа и весь мир) отображаются следующими. Из примера мы видим, что владелец имеет права для чтения, записи и выполнения (rwx). Группа имеет только право на чтение и выполнения файла (r-x). И все остальные имеют только права на чтение и выполнение фала (r-x).

Как же нам присваивать права файлам? Вначале давайте создадим файл для примеров:

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

Мы воспользуемся **chmod**(1) (что означает "change mode" — изменить режим), для того чтобы установить права доступа для файла примера. Сложите восьмизначные числа, для прав, которые вы хотите. Для того, чтобы владелец имел права чтения, записи и выполнения, получится 7. Чтение и выполнение соответствует числу 5. Установим права:

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

Для задания специальных прав, сложите числа вместе, и расположите их в первом столбце:

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

Если вас смущают восьми-битовые числа, вы можете пользоваться буквами. Группы прав представляются, как:

Владелец	u
Группа	g
Мир	o
Все вышеперечисленные	a

Чтобы проделать то же, что и выше, нам понадобится выполнить команду несколько раз:

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

Некоторые предпочитают работать с буквами. Но в любом случае, результат — это тот же самый набор прав доступа.

Мы несколько раз упоминали упоминали права "set user ID" (установить ID пользователя) и "set group ID" (установить ID группы). Возможно, вы мучаетесь в догадках, что же это такое. Обычно, когда вы выполняете программу, она имеет все те же самые права, что и вы имеете, как пользователь. То же справедливо и для группы. Когда вы запускаете программу, она выполняется с правами вашей текущей группы. При помощи "set user ID" вы можете заставить программу всегда выполнятся с правами её владельца (например "root"). "Set group ID" работает так же, но для группы.

Будьте осторожны здесь, так как set user ID и set group ID могут открыть "дыры " в безопасности вашей системы. Если вы установите эти права на файл, принадлежащий root-у, вы позволяете всем запускать эту программу, и выполнять её, как root. Так как у пользователя root нет ограничений в системе, это ставит под вопрос целостность основной безопасности системы. Короче, пользоваться этими командами не плохо, но здравый смысл должен присутствовать.

5.2.3 Ссылки

Ссылки это указатели на файлы. При помощи ссылок вы можете создавать файлы, существующие в нескольких местах, и доступные под многими именами. Есть два типа ссылок: жёсткие и символические.

Жёсткие ссылки это просто разные имена для одного и того же файла. Они могут существовать только в одном и том же каталоге и удаляются тогда, когда исходный файл удаляется. В некоторых случаях они бывают полезны, но большинство пользователей находит символьные ссылки более удобными.

Символьные ссылки (или так же мягкие) могут указывать на файл, находящийся вне текущего каталога. Фактически, это маленький файл, содержащий необходимую информацию. Вы можете добавлять и удалять символьные ссылки не влияя на исходный файл.

У ссылок нет их собственных прав доступа и принадлежности. Вместо этого они отражают права файла, на который они указывают. Slackware, в основном использует символьные ссылки. Вот обычный пример:

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

Оболочка **sh** в Slackware, на самом деле, **bash**. Ссылки удаляются при помощи **rm**. А для их создания используется команда **ln**. Эти команды будут обсуждены более подробно в разделе 5.3.

5.2.4 Монтирование (подключение) устройств

Как обсуждалось ранее в разделе "Структура файловой системы", все диски и устройства в вашем компьютере являются одной большой файловой системой. Различные разделы жёсткого диска, CD-ROMы, и дисководы, все расположены в одном и том же дереве каталогов. Для подключения

всех этих устройств к файловой системе, вам надо использовать **mount** и **umount** команды.

Некоторые устройства автоматически подключаются при загрузке компьютера. Они перечислены в файле `/etc/fstab`. Всё, что вы хотите подключать автоматически, должно иметь строчку в этом файле. Для других устройств, вам придётся указывать команду каждый раз, когда вы хотите воспользоваться ими.

`fstab`

Давайте рассмотрим пример файла `/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	/cdrom	iso9660	ro	0 0

В первом столбце указанно имя устройства. В этом случае устройства, это пять разделов, разбросанных по двум SCSI жёстким дискам, два специальных файловых системы, которым не нужны файловые системы, флоппи дисковод и CD-ROM дисковод. Второй столбец указывает куда подключать устройство. Это должен быть существующий каталог, за исключением раздела подкачки. Третий столбец указывает тип файловой системы для устройства. Для обычного Linux раздела это будет **ext2** ("second extended filesystem" — вторая расширенная файловая система). Для CD-ROM это будет **iso9660**, а для Windows разделов это может быть, как **msdos**, так и **vfat**.

Четвёртый столбец перечисляет опции, которые используются при монтировании устройств. Почти всегда "defaults" вполне достаточно[1]. Тем не менее, устройствам, доступным только для чтения рекомендуется указывать параметр **ro**. Существует огромное количество различных опций. Смотрите man страницу `fstab`(5) для получения дополнительной информации. Последние два столбца используются программой **fsck** и другими командами для манипулирования дисками. Так же обращайтесь к man странице для получения разъяснений.

При установке Slackware Linux, программа установки генерирует минимальный `fstab` файл. Вам понадобится изменить этот файл только если вы хотите добавить диски, или захотите, чтобы устройства автоматически монтировались при загрузке.

mount и umount

В Linux очень просто подключать устройства. Всё что вам надо сделать, это запустить **mount** команду, указывай ей несколько опций. Использование этой команды может быть значительно упрощено, если монтируемое

[1] Для монтирования Windows разделов с кириллицей в именах файлов рекомендуется использовать опцию **codepage=866,iocharset=koi8-r**. Прим. переводчика.

устройство описано в `/etc/fstab` файле. Предположим, я хочу подключить CD-ROM и `fstab` выглядит так, как в примере предыдущего раздела. Я могу сделать это, выполнив команду:

 # mount /cdrom

Так как в `fstab` есть строчка для этого устройства, **mount** возьмёт все опции оттуда. Если бы для этого устройства не было описания в **fstab**, то пришлось бы задавать все опции вручную:

 # mount -t iso9660 -o ro /dev/cdrom /cdrom

Эта строка включает ту же самую информацию, что и строка в вышеприведённом `fstab`, но в другом формате. Давайте разберёмся. **-t iso9660** указывает тип файловой системы на устройстве. В этом случае это **iso9660** система, которая используется на большинстве CD-ROM. **-o ro** указывает, что устройство подключается только для чтения. **/dev/cdrom** это имя устройства, которое мы хотим подключить и **/cdrom** это каталог, куда мы хотим подключить устройство.

До того, как извлекать флоппи, CD-ROM, или другое извлекаемое устройство, которое в данный момент подмонтированно, вам надо размонтировать его. Для того, чтобы сделать это, нужно воспользоваться командой **umount**. Не спрашивайте, почему она называется не **unmount**, так как мы не сможем вам ответить. Вы можете использовать либо имя устройства, либо точку монтирования в качестве аргумента для команды. Например, если вы хотите размонтировать CD-ROM, подключённый в предыдущем примере, любая из нижеприведённых команд позволит вам сделать это:

 # umount /dev/cdrom
 # umount /cdrom

5.2.5 Монтирование NFS

NFS, как мы уже упоминали, это сокращение для Network Filesystem (Сетевая файловая система. На самом деле она не является частью реальной файловой системы, но может быть использована для подключения устройств через сеть.

Зачастую в больших Unix сетях возникает необходимость разделять многие программы, домашние каталоги, и почтовые очереди. Проблема предоставления идентичных копий файлов различным компьютерам разрешается при помощи NFS. Можно использовать NFS для разделения одного набора домашних каталогов между всеми рабочими станциями. В таком случае, рабочие станции монтируют этот разделяемый каталог и используют его, как если бы он был размещён локально.

Смотрите раздел 4.2.6 и man страницы для `exports`(5), **nfsd**(5) и **mountd** для дополнительной информации.

5.2.6 Итог

В этом разделе вы должны были приобрести знания о правах владения и доступа. Вы должны понимать, для чего это нужно, и как их устанавливать. Так же вы должны иметь представление о ссылках и уметь монтировать устройства. Это основные необходимые знания по файловой системе.

5.3 Управление файлами и каталогами

Slackware Linux организованна таким образом, чтобы быть настолько Unix подобной, насколько это возможно. Традиционно ОС Unix ориентирована на интерфейс командной строки. В Slackware есть так же и графический интерфейс, но командная строка остаётся основным способом управления системой. Поэтому очень важно понимать основные команды управления файлами.

Этот раздел объясняет основные команды для управления файлами и примеры их использования. Помимо команд описанных здесь существует множество других, но для начала вам хватит и этих. Команды лишь коротко описаны здесь. Для получения дополнительной информации обращайтесь к соответствующим man страницам.

5.3.1 ls

Эта команда выводит список файлов в каталоге. Пользователи Windows и DOS найдут её аналогичной команде **dir**. При простом выполнении **ls**(1) выведет список файлов в текущем каталоге. Для того, чтобы просмотреть содержимое корневого каталога, вам необходимо выполнить следующие команды:

```
$ cd /
$ ls

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

В таком выводе неудобно то, что вы не можете определить, где файлы, а где каталоги. Многие пользователи предпочитают, чтобы **ls** выводила так же идентификатор типа, что-то вроде этого:

```
$ ls -FC

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

Эта команда добавляет слэш после имён каталогов, звезду после имён выполняемых файлов, и т.д.

ls так же может быть использована для получения более подробной информации о файлах. Например, чтобы увидеть, даты создания, имена владельцев, права доступа, вы воспользуетесь подробным списком:

```
$ ls -l
```

```
drwxr-xr-x   2  root  bin     4096  May   7  1994  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  1997  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  1999  usr/
drwxr-xr-x  18  root  root    4096  Mar   8  03:32  var/
-rw-r—r—    1  root  root  461907  Feb  22  20:04  vmlinuz
```

Предположим, вы хотите увидеть список скрытых фалов в текущем каталоге. Для этого воспользуйтесь такой командой:

```
$ ls -a
```

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

Файлы, имя которых начинается с точки (так называемые "dot файлы") не показываются при простом выполнении **ls**, поэтому называются "скрытыми". Вы увидите их только, если добавите опцию **-a**.

Существует ещё множество других опций, описание которых вы можете найти в man странице. И не забудьте, что вы можете комбинировать их.

5.3.2 cd

Команда **cd** используется для смены рабочего каталога. Просто наберите **cd** и затем имя каталога, в который вы хотите 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$
```

Обратите внимание, что если вы не указываете слэш в начале пути, команда пытается переместится в под-каталог текущего каталога.

Команда **cd** отличается от остальных тем, что она встроена в оболочку. Команды, встроенные в оболочку обсуждаются в разделе 5.1. Возможно, это ничего для вас не значит сейчас. В общем, это означает, что для этой команды нет man страницы. Вместо этого вам надо обратиться к **help** вашей оболочки. Например:

```
$ help cd
```

Эта команда выдаст вам опции команды **cd**.

5.3.3 more

more(1) это то, что мы называем утилитой для разбиения на страницы. Часто вывод определённой команды больше, чем размер экрана. Отдельные команды не знают, как разбить свой вывод на несколько экранов. Они предоставляют это утилите разбиения на страницы.

Команда **more** разбивает вывод на отдельные экраны и ждёт, пока вы нажмёте пробел до того, как выводить следующий экран. Нажатие ввода сместит экран на одну строчку вниз. Вот хороший пример:

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

Вывод будет гораздо больше экрана. Чтобы разбить вывод на экраны, просто пропустите (pipe) команду через **more**:

```
$ ls -l | more
```

Это pipe символ (**Shift и бэкслэш**). В двух словах, pipe означает "возьми вывод команды **ls** и пропусти его через **more**". Вы можете пропускать практически всё через **more**, не только **ls**. pipe описана в разделе 5.1.2 на стр. 86.

5.3.4 less

Команда **more** очень удобна, но зачастую вы встретитесь с ситуацией, когда вы пропустили тот экран, который вам необходим. **more** не позволяет вернуться. Команда **less**(1) позволяет вам сделать это. Она используется так же, как и **more** команда, так что примеры, приведённые выше справедливы и для неё. В общем, less больше, чем more.

5.3.5 cat

cat(1) это сокращение от "concatenate" (сцеплять, связывать). Изначально была разработана для объединения нескольких текстовых файлов в один, но может быть использована и для других целей.

Чтобы объединить два или несколько файлов в один, просто перечислите файлы после **cat** команды и пере направьте вывод в новый файл. **cat** работает со стандартным вводом и выводом, поэтому вам надо воспользоваться символами оболочки для пере направления. Например:

```
$ cat file1 file2 file3 > bigfile
```

Эта команда возьмёт содержимое фалов `file1`, `file2` и `file3` и объединит их в файл `bigfile`.

Вы можете также использовать **cat** для просмотра содержимого фалов. Многие пользователи для просмотра текстовых фалов выполняют **cat**, имя фала, и затем пропускают вывод через **more** или **less**:

```
$ cat file1 | more
```

Так же **cat** часто используется для копирования фалов. Вы можете скопировать любой файл, выполнив:

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

Файл `/bin/bash` будет скопирован в ваш домашний каталог, под именем `mybash`.

Обсуждённые здесь примеры, это лишь несколько из возможных вариантов применения **cat**. Так как **cat** предоставляет расширенный контроль

за стандартным вводом и выводом, она идеальна для применения в shell скриптах, и для использования в качестве части более сложных команд.

5.3.6 touch

touch(1) используется для изменения временных атрибутов файла. С помощью этой команды вы можете изменить отметку времени доступа и отметку времени изменения файла. Если указанный файл не существует, то **touch** создаст пустой файл с указанным именем. Чтобы пометить файл текущим системным временем, выполните такую команду:

```
$ touch file1
```

Эта команда имеет несколько опций, включая опции для уточнения, какую временную метку изменить, какое время использовать, и ещё много других. man станица содержит описание всех этих опций.

5.3.7 echo

Команда **echo**(1) выводит указанный текст на экран. Строку, которую вы хотите вывести, следует указать после **echo** команды. По умолчанию **echo** выведет текст и символ перевода строки после него. Вы можете указать опцию **-n**, чтобы строка не переводилась. Опция **-e** укажет команде искать эскейп символы в строке и выполнить их.

5.3.8 mkdir

mkdir(1) создаёт новый каталог. Просто укажите имя каталога, который вы хотите создать после имени команды. Вот пример создания под-каталога `hejaz` в текущем каталоге:

```
$ mkdir hejaz
```

Так же вы можете указать путь:

```
$ mkdir /usr/local/hejaz
```

Опция **-p** укажет **mkdir** создавать "вложенные" каталоги. Так, если в выше приведённом примере каталог `/usr/local` не существует, то программа выдаст сообщение об ошибке. Указание параметра **-p** заставить программу создать `/usr/local` и `hejaz` каталоги:

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

5.3.9 ln

ln(1) используется для создания ссылок на файлы. Это могут быть как жёсткие — hard, так и гибкие или символические —soft, symbolic ссылки. Разница между двумя типами ссылок обсуждалась в разделе5.2.3 на стр. 92. Если вы хотите создать символическую ссылку на каталог `/var/media/mp3` и расположить ссылку в вашем домашнем каталоге, вам необходимо выполнить такую команду:

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

Опция **-s** указывает команде **ln**, что ссылка должна быть символическая. Затем задаётся цель ссылки, и в конце указывается как она будет называться. В приведённом случае будет создан файл `mp3`, указывающий

на `/var/media/mp3`. Вы можете назвать ссылку как вам заблагорассудиться, просто изменив последний из параметров команды **ln**.

Создание жёстких ссылок производится аналогично. Только опускается опция **-s**. Для примера, создадим жёсткую ссылку на тот же каталог, что и в примере выше:

```
$ ln /var/media/mp3 ~/mp3
```

5.3.10 cp

cp(1) применяется для копирования файлов. Пользователи DOS найдут её похожей на команду **copy**. Эта команда имеет очень много опций. Загляните в man страницу, если вам интересно.

Обычный вариант использования команды это копирование файла из одного места в другое. Например:

```
$ cp hejaz /tmp
```

Эта команда создать копию файла `hejaz` из текущего каталога в `/tmp` каталог.

Многие пользователи предпочитают сохранять временные отметки файлов при копировании. Вот вам пример:

```
$ cp -a hejaz /tmp
```

Это сохранит все временные отметки оригинального файла для копии.

Для того, чтобы рекурсивно скопировать содержимое каталога в другой каталог, вы воспользуетесь такой командой:

```
$ cp -R adirectory /tmp
```

Это скопирует `adirectory` в каталог `/tmp`.

cp имеет гораздо больше опций, которые детально описаны в man странице.

5.3.11 mv

mv(1) перемещает файлы из одного места в другое. Пользователи DOS найдут её похожей на **move** команду. При использовании команды надо указывать источник и назначение. Вот пример обычного использования **mv**:

```
# mv myfile /usr/local/share/hejaz
```

mv имеет несколько опций, которые так же обсуждаются в man странице.

5.3.12 rm

rm(1) удаляет файлы и деревья каталогов. Аналогия для пользователей DOS — **del** и **deltree**. Использование **rm** может быть очень опасным, если вы не отдаёте себе отчёта в том, что вы делаете. В отличие от DOS и Windows в Linux удалённые файлы восстановлению не подлежат.

Чтобы удалить отдельный файл, укажите его имя после имени команды:

```
$ rm file1
```

Если у вас нет прав записи для этого файла, то вы получите сообщение об ошибке доступа. Чтобы удалить файл, несмотря ни на что, воспользуйтесь **-f** опцией:

```
$ rm -f file1
```

Чтобы целиком удалить каталог, воспользуйтесь обеими **-r** и **-f** опциями. Вот вам хороший пример того, как удалить всё содержание вашего диска. Надеемся, вы не захотите выполнить такую команду. В любом случае, вот пример команды:

```
# rm -rf /
```

Будьте очень осторожны при использовании **rm**; вы можете подстрелить себя в ногу. За получением дополнительных опций, обращайтесь к man страничке.

5.3.13 rmdir

rmdir(1) удаляет каталоги из файловой системы. Каталог должен быть пустым, иначе команда не сможет удалить его. Синтаксис прост:

```
$ rmdir <directory>
```

Этот пример удалит `hejaz` под-каталог в текущем рабочем каталоге:

```
$ rmdir hejaz
```

Если этот каталог не существует, **rmdir** скажет вам об этом. Вы так же можете указать полный путь к каталогу, который хотите удалить:

```
$ rmdir /tmp/hejaz
```

Этот пример попробует удалить `hejaz` каталог, являющийся под-каталогом `/tmp` каталога.

Вы так же можете удалить каталог, и все его родительские каталоги, указав **-p** опцию.

```
$ rmdir -p /tmp/hejaz
```

Эта команда вначале попытается удалить каталог `hejaz`, внутри `/tmp`. Если это удастся, команда попытается удалить `/tmp` каталог. **rmdir** продолжит удаление указанных каталогов, пока не столкнуться с ошибкой, или не удалит все указанные каталоги.

5.3.14 Итог

В этом разделе описано множество программ для манипулирования фалами и каталогами. Вы должны были научиться создавать, удалять, и перемещать файлы и каталоги. Вы так же должны были научиться, отображать содержимое каталогов и файлов и изменять их временные метки. Ну и наконец, вы должны понимать, почему **rm -rf** / это очень и очень плохая идея.

5.4 Управление процессами

Любая выполняемая программа называется процессом. Всё от X Winow системы, до системных программ (демонов), которые запускаются при включении компьютера, является процессом. Каждый процесс запускается от имени какого-то пользователя. Процессы, которые стартовали при загрузке обычно выполняются от имени root пользователя, или от имени пользователя nobody. Процессы запущенные вами, обычно выполняются от вашего имени. Процессы, начатые другими пользователями, работают под их именем.

Вы можете управлять теми процессами, которые вы запустили. Вдобавок к этому, root может управлять всеми процессами в системе, даже теми, которые выполняются другими пользователями. Процессами можно управлять и наблюдать за ними при помощи специальных программ, а так же при помощи некоторых команд оболочки.

5.4.1 Перевод в фоновый режим

Программы, запущенные из командной строки обычно выполняются на переднем плане (foreground). Это позволяет вам видеть весь вывод программы и взаимодействовать с ней. Но бывают такие случаи, когда вам не хочется, чтобы программа занимала ваш терминал. Это называется выполнением программы в фоновом режиме (foreground), и существует несколько способов перевода программ в фоновый режим.

Первый способ перевода программы в фоновый режим, это добавление символа & в конце строки, запускающей программу. Предположим, вы хотите воспользоваться mp3 проигрывателем **amp** для проигрывания файлов из каталога, заполненного mp3 файлами, но вам не хочется занимать для этого терминал, так как вам надо делать что-то ещё в то же самое время. Следующая команда запустит **amp** в фоновом режиме:

```
$ amp *.mp3 &
```

Программа будет выполняться, как и должна, а вы вернётесь в приглашение командной строки.

Другой способ перевода программы в фоновый режим, позволяет проделать это уже во время выполнения программы. Вначале запустите программу. Нажмите control+z. Это приостановит процесс. Что-то вроде паузы. Программа моментально прекратит выполняться, но может в любой момент быть продолжена. Как только вы приостановили процесс, вы возвращаетесь к приглашению командной строки. Для перевода процесса в фоновый режим, наберите:

```
$ bg
```

И таким образом, приостановленный процесс перейдёт в фоновый режим.

5.4.2 Вывод из фонового режима

Если вам понадобилось взаимодействовать с фоновым процессом, вы можете вернуть его на передний план. Если у вас только один фоновый процесс, вы можете вернуть его, напечатав:

```
$ fg
```

Программа опять займёт ваш терминал и вы лишитесь приглашения командной строки. Иногда, случается, что программа, выполняемая в фоновом режиме завершает свою работу. В этом случае вы получите сообщение такого вида:

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

Это говорит вам что фоновый процесс (в данном случае ls — не очень интересно), завершился.

Возможно так же одновременно выполнять несколько процессов в фоновом режиме. Если это так, вам надо знать, какой из процессов вы хотите вернуть на передний план. Простое выполнение **fg** вернёт процесс, который последним был переведён в фоновый режим. А что если у вас целый список процессов в фоновом режиме? К счастью, bash имеет команду для перечисления всех процессов. Она называется **jobs** и её вывод выглядит примерно так:

```
$ jpbs
[1] Stopped vim
[2]- Stopped amp
[3]+ Stopped man ps
```

Это выдаст вам список всех фоновых процессов. Как видите, все они остановлены. А ещё точнее, приостановлены. Номера это что-то вроде ID для всех фоновых процессов. Если возле номера отображается знак плюс (man ps), это означает, что этот процесс будет выведен из фонового режима по команде **fg** без указания аргументов.

Если же вы захотите перевести на передний план **vim**, вам придётся напечатать:

```
$ fg 1
```

и **vim** выпрыгнет обратно на ваш экран. Переведение процессов в фоновый режим может быть очень полезно, если у вас есть всего один терминал, открытый через dialup соединение. Вы можете одновременно выполнять несколько программ и переключаться между ними в любой последовательности.

5.4.3 ps

Итак, вы знаете, как переключаться между запущенными вами из командной строки фоновыми процессами. Так же вы знаете, что есть ещё много других процессов, которые всё время выполняются. Как же посмотреть их список? Вам надо воспользоваться **ps**(1) командой. У неё есть множество различных опций, здесь будут описаны лишь самые основные. Для подробной информации смотрите man страницу для **ps**. Man страницы описаны в разделе 2.2.1 на стр. 13.

Простой набор **ps** выдаст список программ, выполняемых на терминале. Довольно часто этот список будет невелик:

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

Не смотря на то, что процессов немного, информация довольно типичная. Сколько бы у вас не было процессов, при простом использовании **ps** столбцы будут теми же. Что же означает вся эта информация?

Итак, по порядку: PID это идентификационный номер (ID) процесса. Каждый выполняющийся процесс получает уникальный идентификатор. В 2.2.x ядрах ID процессов может быть любым числом от 1 и до 32767. Каждому процессу присваивается следующий свободный PID. Когда процесс завершается, его номер освобождается. Когда достигнут максимальный PID, следующий свободный будет взят из наименьшего освобождённого. Скорее всего, это изменится в ядре 2.4, и будут представлены новые 32-х битовые PID.

TTY столбец показывает, на каком терминале процесс выполняется. Простое выполнение **ps** покажет процессы выполняемые на текущем терминале, так что для всех процессов будет выведена идентичная информация в TTY столбце. Как видно из примера, оба показанных процесса выполняются на терминале ttyp0. Это говорит нам о том, что эти процессы запущены либо удалённо, либо из какого-то X терминала.

Столбец TIME показывает, сколько процессорного времени выполняется процесс. Оно не является фактическим временем, с момента запуска процесса. Помните, что Linux это многозадачная операционная система. В любой момент времени есть несколько выполняемых процессов, и каждый из этих процессов получает небольшую порцию процессорного времени. Так вот, информация указанная в столбце TIME, показывает время, которое гораздо меньше фактического времени выполнения процесса. Если вы это время больше, чем несколько минут у одного из процессов, то скорее всего, что-то не так.

Ну и наконец, CMD столбец, показывает что же это за программа. Отображается только имя программы; опции командной строки и аналогичная информация не выводится. Для того чтобы увидеть эту информацию, вам необходимо воспользоваться одной из многих опций программы **ps**. Давайте обсудим некоторые из них вкратце.

Вы можете получить полный список выполняемых в вашей системе процессов, воспользовавшись правильным набором опций. Скорее вы захотите вывести расширенный список процессов. Давайте попробуем:

```
                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
$ ps -ax         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]
(output cut)
```

Большинство из этих процессов запускаются при старте почти любого компьютера, работающего под Linux. В своей системе я сделал несколько изменений, так что вы скорее всего увидите что-то другое. Но всё же большинство из перечисленных процессов вы увидите в вашей системе. Как вы видите, опции заданные программе в этом примере, заставляют её выводить не только имена программ, но и опций с которыми они были выполнены. А так же ещё несколько новых столбцов с интересной информацией.

Наверное вам сразу бросится в глаза, что большинство из процессов выполняются на tty "?". Это процессы, запущенные с более не активного терминала. И поэтому они больше не принадлежат определённому терминалу.

Так же вы видите новый столбец: STAT. Он показывает состояние (status) процесса. S используется для спящего (sleeping) процесса: процесс ожидает, пока что-то произойдёт. Z используется для зомбиованных процессов (zombied). Это такие процессы, родительский процесс которых умер, оставив дочерние процессы рабочими. Это не есть хорошо.

Если вы хотите увидеть ещё больше информации о выполняемых процессах, попробуйте такую 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]
(output cut)
```

Это достаточно полный набор информации. Здесь вы видите новые столбцы, которые описывают: какой пользователь запустил процесс, сколько системных ресурсов использует процесс (%CPU, %MEM, VSZ и RSS столбцы), и дату, когда процесс был запущен. Очевидно, здесь предоставлено достаточно информации о процессах, которая может быть полезна для системного администратора. Так же мы встретились с ещё одной проблемой: информация не помещается на экране. Опция "-w" исправит это.

Вывод команды не очень красив, но весьма полезен работает. И всё же мы ещё не увидели всей доступной информации по выполняемым процессам. Есть ещё много разнообразной информации, которую вы можете получить о каждом из процессов. Загляните в man страницу для программы **ps**. Тем не менее, опции описанные в этом разделе являются наиболее часто применяемыми. И наверное вы будете пользоваться в основном ими.

5.4.4 kill

Иногда что-то не так происходит с программами и возникает необходимость восстановить порядок в системе. Программа, предоставляющая такой вид услуг, называется **kill**(1). Она может быть использована для манипулирования процессами несколькими разными способами. Наиболее часто программа применяется, чтобы убивать процессы. У вас появится необходимость в этом, если программа вышла из под контроля и забирает много системных ресурсов, или если вам просто не нравится, что эта программа осталась в памяти.

Для того, чтобы убить процесс, вам надо знать либо его PID, либо имя.

Чтобы получить PID воспользуйтесь программой **ps**, как это обсуждалось в предыдущем разделе. Например, чтобы убить процесс 4747, вы зададите такую команду:

```
$ kill 4747
```

Для того, чтобы убить процесс, вам надо быть его хозяином. Это сделано в целях безопасности. Если бы вы могли убивать процессы других пользователей, открылась бы возможность произведения множества злонамеренных вещей в системе. Разумеется, root может убить любой процесс в системе.

Есть так же другая разновидность программы **kill**, которая называется **killall**(1). Функция программы соответствует её названию. Она убивает все из запущенных программ, с указанным именем. Если вы хотите убить все выполняемые **vim** процессы, вам следует воспользоваться такой командой:

```
$ killall vim
```

Все **vim** процессы, выполняемые вами будут убиты. Если вы выполните команду, как root, то она убьёт вообще все **vim** процессы, выполняемые любым из пользователей. Это предоставляет интересный способ выбрасывания всех (включая и вас) из системы:

```
#killall bash
```

Иногда обычное выполнение **kill** не справляется с поставленной задачей. Некоторые процессы не могут быть убиты обычной командой **kill**. Вам понадобиться воспользоваться более мощной версией. Если этот нудный процесс с PID4747 не умер после выполнения **kill**, вы можете попробовать следующее:

```
$ kill -9 4747
```

Практически наверняка, это заставит процесс 4747 умереть. Вы можете воспользоваться этим же ключом с **killall** программой. Этот ключ позволяет посылать различные сигналы программам. Обычный **kill** посылает процессу сигнал SIGTERM (terminate — завершиться). **kill -9** посылает SIGKILL (kill — убить). В вашем распоряжении целый набор различных сигналов. Вы можете запросить список, выполнив:

```
$ 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
```

Номера используются для **kill**, а имя без предшествующего "SIG" используется для **killall**. Вот ещё пример:

```
$ killall -KILL vim
```

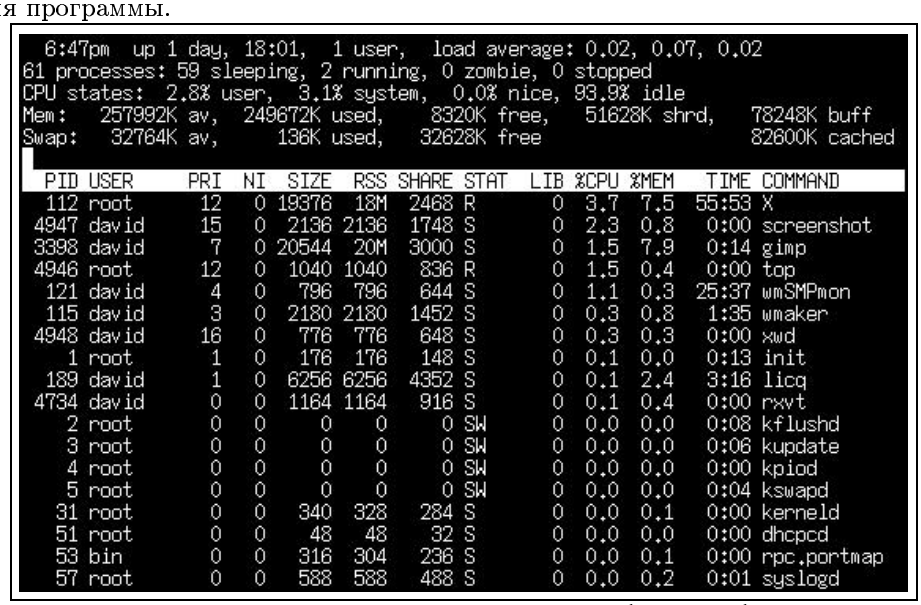
Ещё один вариант использования **kill** это перезапуск процесса. Послав SIGHUP сигнал мы заставим процесс перечитать свои конфигурационные процессы. Это очень полезно, если вы изменили содержание настроечных файлов для какой-то программы.

5.4.5 top

Есть так же и программа, которую вы можете использовать для вывода динамической информации о процессах, выполняемых в системе. Программа называется **top**(1):

```
$ top
```

Эта команда выдаст полный экран информации о выполняемых процессах, а так же некоторую общую информацию о системе. Такую, как средняя загрузка, количество выполняемых процессов, состояние процессора, информацию о свободной памяти и для каждого из процессов — PID, пользователь, приоритет, использование CPU и памяти, время выполнения и имя программы.



Называется она **top** потому, что программы, наиболее требовательные к процессору будут отражены в верху списка. **top** является довольно удобным инструментом для определения того, какая из программ вышла из под контроля и должна быть убита,

5.4.6 Итог

В этом разделе обсуждалось, что такое процессы и как вы можете ими управлять. Это включает выполнение в фоновом режиме, вывод из фонового режима, а так же использование **ps**, **top**, и **kill** для контроля за процессами. Вы должны уметь определить, какие процессы выполняются в вашей системе и знать, как вы можете от них избавиться, если они вышли из под контроля.

5.5 Основы системного администрирования

Вы администратор всех тех компьютеров, на которых вы являетесь пользователем root. Это может быть компьютер на вашем рабочем столе, с одним или двумя пользователями, а может быть и большей сервер с несколькими сотнями пользователей. Безусловно, вам необходимо знать, как манипулировать пользователями и безопасно выключать компьютер. На самом деле, это не так сложно, как кажется, но придётся привыкнуть к некоторым особенностям. В добавок к этому, вам придётся ознакомиться с некоторыми идеями, и принципами работы системы паролей.

5.5.1 Пользователи и группы

Вспомогательные программы

Самый простой способ манипулирования пользователями — при помощи поставляемых с дистрибутивом вспомогательных скриптов и программ. В Slackware для работы с пользователями есть такие программы: **adduser**, **userdel**(8), **chfn**(1),**chsh**(1) и **passwd**(1). А для работы с группами есть **groupadd**(8), **groupdel**(8) и **groupmod**(8). За исключением **chfn**, **chsh** и **passwd** перечисленные программы могут выполняться только пользователем root и поэтому расположены они в `/usr/sbin`. **chfn**, **chsh** и **passwd** могут быть выполнены кем угодно, а расположены они в каталоге `/usr/bin`.

Добавить пользователя можно при помощи программы **adduser**. Мы начнём с разбора процедуры, комментируя все вопросы, задаваемые программой и кратко описывая, что же всё это означает. Вариант ответа по умолчанию, предлагаемый программой практически на каждый из вопросов, показывается в скобках и может быть использован для ответа практически на все вопросы.

#adduser

Login name for new user (8 characters or less) []: jellyd

Это имя, которое пользователь будет использовать для входа в систему. Оно должно состоять из восьми или менее символов, так как все login утилиты рассчитаны на это. Обычно используются только маленькие буквы, но можно пользоваться и большими.

User id for jellyd [defaults to next available]:

ID (UID) пользователя, это на самом деле то, с помощью чего в системе Linux определяется принадлежность файлов. Каждый пользователь имеет уникальный номер, начиная с 1000 в Slackware. Вы можете выбрать UID для нового пользователя, или вы можете просто позволить программе **adduser** присвоить пользователю следующий свободный номер.

Initial group for jellyd [users]:

Все пользователи по умолчанию попадают в группу users. Вы можете захотеть создать отдельную группу для каждого из пользователей, но это не рекомендуется.

Additional groups for jellyd (seperated with commas, no spaces) []:

Здесь вам предлагается добавить пользователя в дополнительные группы. Один пользователь может числиться в нескольких группах. Это полезно, если у вас есть разные группы для разных задач, например, для изменения файлов вэб-сайта, для игр и т.д.

jellyd's home directory [/home/jellyd]:

Домашние каталоги по умолчанию располагаются в `/home` каталоге. Если вы работаете в очень большой системе, возможно возникнет необходимость разместить домашние каталоги в другом месте. Эта опция позволяет вам уточнить, где должен располагаться домашний каталог пользователя. Вы можете так же парализовать пользователя, указав его домашним каталогом что-то вроде `/usr/bin/false`, но мы не рекомендуем пользоваться этим методом.

jellyd's shell [/bin/bash]:

bash это оболочка по умолчанию в Slackware Linux и подойдёт для большинства людей. Если ваш новый пользователь ранее пользовался Unix, он может быть захочет какую-то другую оболочку, к которой привык больше. Вы можете поменять оболочку сейчас, или же пользователь сделает это позже самостоятельно при помощи **chsh** команды.

jellyd's account expiry date (YYYY-MM-DD) []:

Вы можете задать дату, после которой пользователь не будет больше иметь доступа к системе. По умолчанию, это бесконечность. Например, эта опция может быть полезна для ISP, если они хотят создать пользователя до определённой даты, пока не получать плату за следующий год.

OK, I'm about to make a new account. Here's
what you entered so far:

New login name: jellyd
New UID: [Next available]
Initial group: users
Additional groups: [none]
Home directory: /home/jellyd
Shell: /bin/bash
Expiry date: [no expiration]

This is it... if you want to bail out, hit Control-C.
Otherwise, press ENTER to go ahead and make the account.

Теперь вы видите всю введённую информацию и если что-то не так, вы можете остановиться при помощи Control-C и начать всё сначала. Иначе вы можете нажать ввод, и пользователь будет создан.

Making 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:

Вся эта информация дополнительна, она используется для **finger**. Вы можете оставить пустыми эти поля. Пользователь может так же изменить эту информацию в любой момент при помощи **chfn**. Но может быть полезно указать по крайней мере имя и телефон, на тот случай, если вы захотите связаться с персоной.

```
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.
Done...
```

Вам придётся ввести пароль для нового пользователя. Если новый пользователь не присутствует при создании эккаунта, просто укажите какой-то пароль по умолчанию и скажите пользователю поменять его на что-то более надёжное.

Выбор пароля Выбор надёжного пароля это первый шаг по защите системы от взлома. Пароль не должен быть легко угадываемым, так как в этом случае шансы несанкционированного проникновения в систему возрастают. Идеальный пароль это случайный набор символов, включая большие и маленькие буквы, числа и другие символы. Имейте ввиду, что tab лучше не использовать, так как он трактуется по разному разными системами, и при удалённом доступе могут возникнуть проблемы.

В общем, руководствуйтесь здравым смыслом: не выбирайте в качестве пароля чей-то день рождения, какую-то общую фразу, что-то что есть у вас на столе, что-то, что тесно связанно с вами. "secure1" так же плохой пароль.

Удаление пользователя ещё более простая процедура. Просто запустите **userdel** и имя пользователя, которого вы хотите удалить. Вам необходимо убедиться, что пользователь в этот момент не в системе, и что нет процессов, выполняемых этим пользователем. Так же помните, что если вы удалили пользователя то его больше нет.

```
#userdel jellyd
```

Выполнив эту команду, вы удалите "jellyd" из вашей системы. Так ему и надо:) Команда удаляет пользователя из файлов `/etc/passwd` и `/etc/group`, но не удалит его домашний каталог. Если вы так же хотите удалить и домашний каталог, вы должны воспользоваться следующей командой:

```
#userdel -r jellyd
```

Временная парализация эккаунта, будет описана в разделе "Изменение паролей" на стр. 112, так как это требует изменения пароля пользователя. Изменение регистрационной информации пользователя будет описано в разделах "Изменения пароля" и "Изменение информации о пользователе" на стр. 112 и 113, соответственно.

Программы для создания и удаления групп очень просты. **groupadd** просто создаст ещё одну группу в файле `/etc/group`, с уникальным group ID, а **groupdel** удалит указанную группу. Вам надо будет вручную добавить пользователей в созданную группу, путём редактирования `/etc/group`.

Давайте создадим группу:

```
#groupadd cvs
```

Или удалим:

```
#groupdel cvs
```

Вручную Конечно, возможно добавлять, изменять и удалять пользователей и группы вручную. После того, как вы ознакомитесь с процедурой, возможно, вы найдёте, что использование скриптов сложнее, чем создание вручную.

Вначале, добавим нового пользователя в файлы `/etc/passwd`(5), `/etc/shadow`(5) и `/etc/group`(5). Файл `passwd` содержит некоторую информацию о пользователе, но не содержит их паролей. Этот файл должен быть доступен для чтения всем пользователям, но вы не хотите, чтобы пароли, даже закодированные, были доступны для чтения всему миру, так как это будет огромной помощью злоумышленникам. Поэтому закодированные пароли хранятся в теневом файле, который доступен для чтения только root пользователю, а в файле `passwd` пароли пользователей отображаются, как "x". Файл `group` показывает список всех групп, и кто из пользователей к каким группам относиться.

Давайте рассмотрим `/etc/passwd` файл и и разберёмся, как добавить кого-то. Типичная строка файла выглядит таким образом:

chris:x:1000:100:Chris Lumens,Room 2„:/home/chris:/bin/bash

Каждая строка, соответствует пользователю. Поля в каждой линии разделены двоеточием. Поля в порядке слева на право: имя для входа в систему, закодированный пароль ("x" для всех в Slackware системе, так как мы пользуемся пакетом для теневых файлов паролей), ID пользователя, ID группы, дополнительная информация о пользователе, разделённая запятыми, домашний каталог и оболочка. Что вам надо сделать, так это добавить в конец файла строчку, заполнив все перечисленные поля соответствующими новому пользователю значениями.

Убедитесь, что пароль x, что ID пользователя уникален, что пользователь входит в группу 100 ("users" группа в Slackware) и что выбрана правильная оболочка.

Теперь добавим строку в `/etc/shadow`, который содержит пароли. Типичная строка этого файла выглядит вот так:

chris:1w9bsw/N9$UWLr2bRER6YyBS.CAEp7R.:11055:0:99999:7:::

Опять таки, каждая строка соответствует одному пользователю, и поля разделены двоеточием. Поля: имя входа в систему, закодированный пароль, количество дней со дня Эпохи (1 Января 1970) до дня, когда пароль был изменён последний раз. количество дней, после которых пароль может быть изменён, количество дней после которых пароль должен быть изменён, количество дней до истечения эккаунта, время когда пользователь получит сообщение о закрытии его эккаунта, дни после истечения эккаунта,после которых эккаунт полностью блокируется, дни с момента Эпохи, когда эккаунт должен быть заблокирован и зарезервированное поле.

Как вы видите, это в основном информация об истечении эккаунта. Если вы не пользуетесь информацией об истечении эккаунта, вам надо только заполнить некоторые из полей со специальными значениями. Иначе, вам понадобится произвести вначале некоторые вычисления и принять некоторые решения, до того, как вы сможете заполнить все эти поля. Для нашего нового пользователя впишите какой-то мусор в поле пароля. Не волнуйтесь о том, какой именно сейчас установлен пароль, так как через минуту мы изменим это. В пароле могут быть использованы любые символы кроме двоеточия. Оставьте "количество дней, с момента изменения пароля" пустым. Введите 0, 99999 и 7 как это указанно в примере выше (в те же поля), и

оставьте остальные поля пустыми.

Для тех из вас, кто увидев мой пароль в этом примере, и думает, что вы можете взломать его, продолжайте, пожалуйста. Вы теперь знаете пароль к тестовой системе, находящейся за firewall. Это очень вам поможет :)

Так как все являются членами "users" группы по умолчанию, вам не надо добавлять нового пользователя в неё. Если вы захотите создать новую группу или добавить нового пользователя в другие группы, вам необходимо будет отредактировать `/etc/group`. Вот типичная строка файла:

cvs::102:chris,logan,david,root

Поля: имя группы, пароль группы, ID группы и члены группы. Создание новой группы, это всего лишь добавление ещё одной строчки, с уникальным ID и указанием списка пользователей, которых вы хотите включить в группу, в этот файл. Все из перечисленных пользователей, если они в данный момент в системе, вынуждены будут выйти и снова войти в систему, чтобы изменения вступили в силу.

Теперь давайте вернёмся к команде **passwd**, чтобы создать пароль для созданного нами пользователя. Затем воспользуемся **mkdir**, чтобы создать домашний каталог там, где мы указали, что он будет расположен в `/etc/passwd` файле.

Если вы установили **sendmail**(8), вам необходимо будет создать новый файл с соответствующими правами, принадлежащий новому пользователю в `/var/spool/mail` каталоге. Вот пример:

#touch /var/spool/mail/jellyd
#chown jellyd.users /var/spool/mail/jellyd
#chmod 660 /var/spool/mail/jellyd[2]

Эти команды создадут файл очереди почты (mail spool файл) для нового пользователя "jellyd" и установят правильные права и принадлежность файла.

Для удаления пользователя, просто удалите всё, что вы вводили при создании. Удалите упоминания о пользователе из `/etc/passwd` и `/etc/group`. Удалите его login имя, из всех групп в `/etc/group`, удалите его очередь почты, если таковая имеется, и так же не забудьте удалить домашний каталог пользователя, если есть необходимость.

Удаление групп проще. Просто удалите строку, определяющую группу из файла `/etc/group`.

Изменение паролей

Программа **passwd** изменяет пароль, модифицируя `/etc/shadow` файл. Этот файл содержит все пароли системы в закодированной форме. Для того, чтобы изменить ваш пароль, выполните:

$ 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:

Как видите, вас просят ввести ваш старый пароль. Он не появится на экране, как и при входе в систему. Затем вам необходимо ввести новый пароль. **passwd** произведёт много различных проверок вашего пароля, и

пожалуется вам, если найдёт ваш пароль ненадёжным. Вы можете проигнорировать эти сообщения, если пожелаете. Затем вам предложат ввести пароль опять, для подтверждения.

Если вы root, то вы можете изменить чей угодно пароль:

```
# passwd ted
```

Вам надо будет пройти через ту же самую процедуру, за исключением того, что не понадобится указывать старый пароль. (Одно из преимуществ пользователя root...).

Если у вас в системе есть пользователи, которые насаждают вам, вы можете временно заблокировать их эккаунты. Позже вы можете опять активизировать их. Как блокировка, так и последующее включение пользователя могут быть произведены при помощи **passwd**. Чтобы отключить эккаунт, как root, выполните следующее:

```
# passwd -l david
```

Это изменит пароль david-a на что-то, что не может быть подобрано. Для возвращения пользователя, выполните следующее:

```
# passwd -u david
```

Теперь эккаунт david-a опять работает, как и раньше. Приостановка эккаунта может быть полезна, если пользователь играет не по правилам, установленным в вашей системе или если они экспортировали очень большую копию **xeyes**(1) на ваш рабочий стол X.

Изменение информации пользователя

Пользователи могут поменять в любой момент времени свою оболочку и свою finger информацию. В Slackware Linux **chsh** (change shell — изменить оболочку) и **chfn** (change finger — изменить finger) используются для этого.

Пользователь может выбрать любую из оболочек, перечисленных в `/etc/shell`. Для большинства bash будет идеальным выбором. Кто-то может быть знаком с оболочками, встречающимися в других Unix системах используемых. Выбрать оболочку можно при помощи **chsh**:

```
$ chsh
Password:
Changing the login shell for chris
Enter the new value, or press return for the default
        Login Shell [/bin/bash]:
```

После ввода пароля введите полный путь к новой оболочке. Убедитесь вначале, что она перечислена в `/etc/shells`(5). root может так же изменить оболочку пользователя, указав как аргумент программе **chsh** имя пользователя.

finger информация это необязательная информация. Она содержит такие поля, как полное имя, номера телефона и номер кабинета. Она может быть изменена при помощи **chfn**, так же, как и при создании эккаунта. Как всегда, root может изменить finger информацию для любого пользователя.

5.5.2 Правильное выключение компьютера

Очень важно выключать компьютер правильно. Простое отключение питания может вызвать серьёзные повреждения системы. Пока компьютер

включён, даже если вы ничего не делаете, некоторые файлы открыты системой. Помните, что в фоновом режиме всегда выполняются какие-то системные процессы. Эти процессы управляют системой, и много файлов открыты этими программами всё время. Если просто отключить питание, эти файлы не закрываются правильно, и из-за этого могут повредиться. В зависимости от того, какие из файлов были повреждены, система может быть серьёзно повреждена. В любом случае, вам придётся пройти через длительную процедуру проверки системы при следующей загрузке системы.

Так что если вы собираетесь выключить, или пере-загрузить компьютер, очень важно делать это правильно. Есть несколько путей сделать это; вы можете выбрать, тот, который вам понравится больше всех. Большинство способов, используемых для выключения так же могут быть использованы для пере-загрузки.

Первый метод это программа **shutdown**(8) и наверное это самый распространённый метод. **shutdown** может быть использована для выключения или пере-загрузки в определённое время, и может выдать сообщение всем пользователям системы, говорящее, что система будет выключена.

Обычный вариант использования **shutdown**:

```
# shutdown -h now
```

В этом случае мы не послали специфическое сообщение пользователям; они увидят стандартное сообщение **shutdown** программы. "now" это время, когда мы хотим выключить систему, а "-h" значит halt (остановить) систему. Это не очень вежливый способ выключения многопользовательской системы, но для вашего домашнего компьютера, этого больше чем достаточно. Более вежливо на многопользовательской системе будет заранее предупредить пользователей:

```
#shutdown -h +60
```

Эта команда пере-загрузит компьютер через час (60 минут), что вполне нормально на обычной многопользовательской системе. Действительно важные системы должны заранее создавать расписание пере-загрузки и предоставлять его в `/etc/motd`(5).

Для пере-загрузки используется та же команда, но с ключом "-r" вместо "-h":

```
#shutdown -r now
```

Вы можете использовать тот же формат для времени с **shutdown -r**, что и с **shutdown -h**. Есть много других действий, которые вы можете выполнять при помощи **shutdown**, чтобы контролировать, когда именно пере-загрузить или выключить компьютер. Смотрите man страницу для подробностей.

Второй способ выключения или пере-загрузки компьютера, это использование **halt**(8) и **reboot**(8) программ. **halt** немедленно выключит ваш компьютер, а **reboot** пере-загрузит его. **reboot** это всего лишь символическая ссылка на **halt**. Вызываются команды вот так:

```
#halt
#reboot
```

Более низко уровневый способ пере-загрузки компьютера, это общение напрямую с **init**. Все остальные методы, это лишь более удобные способы общения с **init**, но вы можете так же и напрямую сказать **init**-у, чего вы хотите, при помощи **telinit**(8) (обратите внимание на то, что в имени

команды есть только одна "l"[3]). Использование **telinit** сообщит **init**-у, в какой из уровней загрузки (runlevel) перейти, что в свою очередь, вызовет выполнение соответствующего сценария. Этот сценарий убьёт или породит процессы, в соответствии с выбранным уровнем загрузки. Это может быть использовано для выключения или пере-загрузки, так как оба этих процесса являются определёнными уровнями загрузки.

#telinit 0

Уровень загрузки 0 это режим выключения. Сказав **init** войти в режим загрузки 0, мы заставим его убить все процессы, размонтировать файловые системы и остановить систему. Это отличный способ выключения компьютера. На многих laptop-ах это так же вызовет отключение питания.

#telinit 6

Уровень загрузки 6 это режим пере-загрузки. Все процессы будут убиты, файловые системы размонтированы, и компьютер пере-загрузится. Это самый "правильный" метод пере-загрузки.

Есть ещё один способ пере-загрузки компьютера. Все остальные (вышеперечисленные) методы могут быть выполнены только пользователем root. Тем не менее, можно пере-загрузить компьютер, если вы не root, в том случая, если у вас есть физический доступ к клавиатуре. Выполнение комбинации из трёх пальцев (control-alt-delete) вызовет немедленную пере-загрузку компьютера. что действительно происходит в этом случае, так это выполнение программы **/usr/sbin/ctrlaltdel**(8). Так что если у программы какие-то странные права доступа, или если она отсутствует, нажатие этих кнопок ни к чему не приведёт[4].

5.5.3 Итог

В этом разделе обсуждены вопросы создания и удаления пользователей и групп. Вы должны уметь делать это, как при помощи поставляемых спецпрограмм, так и вручную. В добавок, вы должны знать, что происходит на различных стадиях процесса создания пользователя, иметь общие представления о том, как выбрать пароль, и как изменять информацию пользователя. Так же вы должны знать, как правильно выключать компьютер, и почему важно делать это правильно. Это самые важные части системного администрирования системы, будь то ваш домашний компьютер, или большой сетевой сервер.

[3] по английски tell to init — сказать init-у. прим. переводчика

[4] Если в целях безопасности вы не хотите, чтобы любой пользователь мог пере-загрузить компьютер, или хотите, чтобы компьютер выключался, а не пере-загружался при комбинации, отредактируйте соответствующим образом строчку `ca::ctrlaltdel:/sbin/shutdown -t5 -rf now` файла `/etc/inittab`

5.6 Основные сетевые команды

Сеть есть не что иное, как несколько компьютеров соединённых вместе. Сеть может быть простой, состоящей из нескольких компьютеров, подключённых друг к другу у вас дома или на работе, или может быть очень большой и сложной, как университетская сеть, или даже весь интернет. Если ваш компьютер является частью сети, то у вас есть доступ к остальной её части, может быть на прямую, а может быть и через сервисы, такие, как почта м вэб.

Есть очень много различных сетевых программ. Некоторые из них удобны для проверки и диагностики сетей. Другие (такие, программы чтения новостей, и вэб броузеры), полезны для выполнения вашей работы и общения с другими людьми.

5.6.1 ping

ping(8) посылает ICMP ECHO_REQUEST пакет указанному хосту. Если хост отвечает, вы получаете ICMP пакет обратно. Звучит странно? Вы можете "ping-овать" адрес, чтобы посмотреть, жив ли компьютер, ему соответствующий. Вот пример общения между двумя пользователями Linux:

> Пользователь A: Loki опять лёг.
>
> Пользователь B: Ты уверен?
>
> Пользователь A: Да, я пытался пинговать его, но никакого ответа.

Такой вариант применения **ping** и делает её очень полезной каждодневной программой. Программа позволяет очень быстро увидеть, доступен ли компьютер. Синтаксис применения прост:

$ping <ip адрес или имя хоста>

Конечно, программа так же имеет некоторые опции. Загляните в **ping**(1) man страницу, для информации.

5.6.2 finger

finger(1) запросит информацию о специфическом пользователе. Вы указываете **finger**-у имя пользователя, или email адрес, а программа попытается соединиться с нужным сервером и запросит имя пользователя, номер телефона офиса и другую подобную информацию. Вот пример:

$finger johnc@idsoftware.com

finger может выдать имя пользователя, состояние почты, телефонные номера и файлы `.plan` и `.project`. Разумеется, выдаваемая сервером информация зависит от конкретного сервера. Сервер, поставляемый в Slackware, возвращает такую информацию:

- Имя пользователя
- Номер комнаты
- Домашний телефон
- Рабочий телефон

- Login статус
- Email статуcs
- Содержимое `.plan` файла из домашнего каталога пользователя
- Содержимое `.project` файла из домашнего каталога пользователя

Первых четыре могут быть установлены или изменены при помощи **chfn** программы. Эти параметры хранятся в `/etc/passwd` файле. Для изменения информации в ваших `.plan` и `.project` файлах, просто измените их при помощи вашего любимого текстового редактора. Они должны находится в вашем домашнем каталоге и называться, соответственно, `.plan` и `.project`.

Многие пользователи пользуются **finger** для своего эккаунта с удалённых компьютеров, просто чтобы посмотреть, есть ли у них почта. Так же вы можете посмотреть план работы или текущий проект пользователя. John Carmack из id Software регулярно обновляет свой план-файл, чтобы держать сообщество пользователей в курсе того, над чем он сейчас работает.

Как и большинства программ, у **finger** есть опции. Опять таки, мы отправим вас к man странице за дополнительной информацией.

5.6.3 telnet

Кто-то когда-то сказал, что **telnet**(1) это самое крутое, из всего, что он когда либо видел в мире компьютеров. Возможность удалённо подключаться и делать что-то на другом компьютере, выделяет Unix и Unix-подобные системы от других систем.

telnet позволяет вам войти в удалённую систему точно так же, как если бы вы сидели за этим компьютером. Указав ваше имя пользователя и пароль, вам выдаётся приглашение командной строки оболочки. С этого момента вы можете делать всё то же, что обычно выполняете в текстовой консоли. Писать письма, читать группы новостей, перемещать файлы, и т.д. Если вы в X и вы соединились с другим компьютером по telnet из **xterm**, вы можете выполнять программы на удалённом компьютере, а отображать на вашем. См. раздел 4.3.5, на стр. 72.

Для подключения к удалённому компьютеру, воспользуйтесь таким синтаксисом:

$telnet <hostname>

Если хост ответит, вы получите приглашение входа в систему. Введите имя пользователя и пароль. Ну вот. Теперь вы в оболочке. Чтобы выйти из вашей сессии **telnet**, воспользуйтесь либо **exit**, либо **logout**, на ваше усмотрение.

Важно ВАЖНОЕ ЗАМЕЧАНИЕ: **telnet** не кодирует информацию, которую он посылает. Всё посылается, как обычный текст, даже пароли. Не желательно использовать **telnet** через интернет. Альтернатива — Secure Shell. Она кодирует весь трафик и доступна бесплатно. Смотрите `http://www.ssh.org/` для справки.

5.6.4 FTP клиенты

FTP расшифровывается, как File Transfer Protocol (протокол передачи файлов). Он позволяет вам отсылать и принимать файлы между двумя компьютерами. Один из них FTP сервер, а другой — FTP клиент. В этом разделе мы будем обсуждать клиента.

Для определённости, клиент это вы. "server" это компьютер, который отвечает на ваш FTP запрос и позволяет войти. Вы будете загружать файлы с (download) и на (upload) сервер. Клиент не может принимать FTP соединения, он может только соединяться с сервером.

ftp

Для подключения к FTP серверу, просто выполните **ftp**(1), указав имя хоста к которому хотите соединиться:

$ftp <имя хоста>

Если на указанном вами хосте есть FTP сервер, он спросит у вас имя пользователя и пароль. Вы можете войти с вашим собственным логином, если таковой у вас имеется, или вы можете войти, как "anonymous". Анонимные FTP сайты очень часто используются, как архивы программ. Например, чтобы скачать Slackware Linux через FTP, вы должны использовать анонимный FTP.

Подключившись, вы получите приглашение **ftp>**. Для FTP используются отдельные команды, но они похожи на стандартные команды. Вот список основных команд и для чего они используются:

Command	**Purpose**
ls	Выдаёт список файлов
cd <dirname>	Для перехода в другой каталог
put <filename>	Скачать файл
put <filename>	Закачать (upload) файл
hash	Если включить, то для каждого скачанного килобайта будет показываться # символ
prom	Включает интерактивный режим загрузки
mget <mask>	Скачивает файл или группу файлов; можно использовать шаблоны имён
mput <mask>	Закачать файл или группу файлов можно использовать шаблоны имён
quit	Отключиться от FTP сервера

FTP это довольно простая программа и её явный недостаток это интерфейс пользователя, непривычный для современных пользователей. В **man** странице обсуждаются некоторые из команд **ftp**(1).

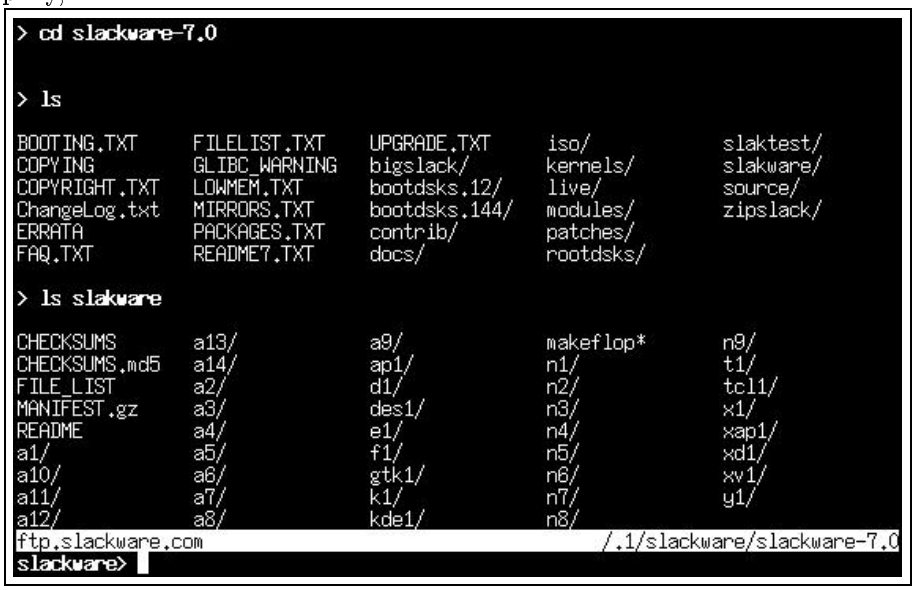
ncftp

ncftp(1) (произносится "Nik-F-T-P") это альтернатива стандартному **ftp**, поставляемая с Slackware. Это так же программа с текстовым интерфейсом, но предлагает много дополнительных функций в сравнении с **ftp**, включая:

- Завершение по tab

- Файл закладок
- Passive and non-passive FTP transfer modes
- More liberal wildcard uses
- Command history

По умолчанию, **ncftp** попробует войти анонимно на указанный вами сервер. Вы можете заставить программу выдать вам логин приглашение, задав "-u" ключ. После входа в систему, вы можете пользоваться теми же командами, что и в **ftp**, только интерфейс немного отличается в лучшую сторону, больше похож на bash.



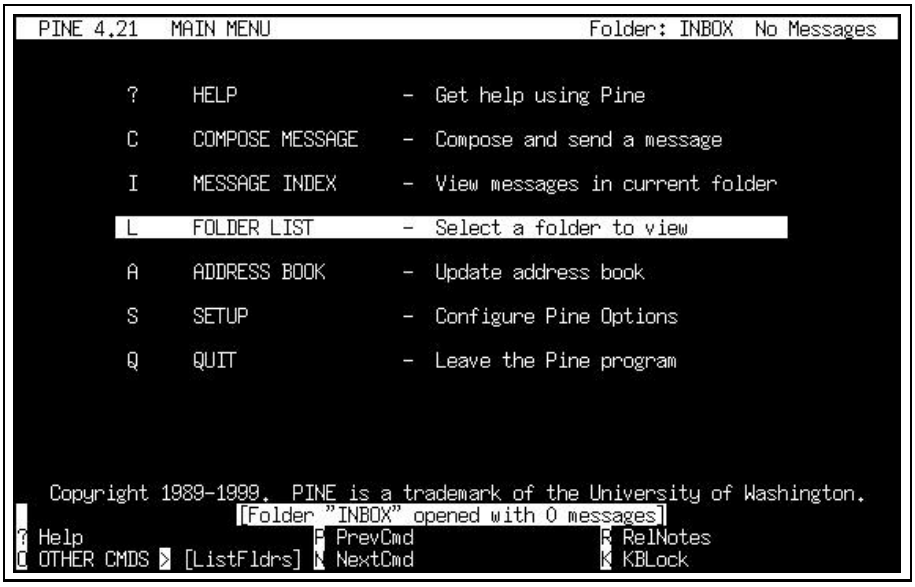
5.6.5 Электронная почта

Электронная почта это самый распространённый сервис в интернет. В 1998 году сообщалось, что было послано больше электронной почты, чем обычной. Это конечно, удобный и полезный, а так же необходимый сервис.

В Slackware, мы поставляем стандартный почтовый сервер и несколько почтовых клиентов. Все обсуждаемые здесь клиенты работают в текстовом режиме. Многие Windows пользователей могут возразить, но вы обнаружите, что клиенты, работающие в текстовом режиме довольно удобны в использовании, в особенности, для удалённой проверки почты.

pine

pine(1) (англ. сосна) это вам не elm (англ. вяз). По крайней мере так говорят. Вашингтонский университет создал свою программу для интернет, новостей и электронной почты, изначально предназначавшуюся для внутреннего использования студентами, как простого mail-клиента. Сегодня **pine** это один из самых популярных почтовых клиентов и доступен в любой из разновидностей Unix и даже в Windows.



Вы увидите меню команд, и ряд командных клавиш внизу. **pine** это довольно сложная программа, так что мы не будем слишком углубляться в детали здесь.

Чтобы посмотреть что у вас в папке "inbox" (входящие), нажмите **i**. Сообщения представлены информацией о дате, авторе и теме. Наведите курсор на то сообщение, которое хотите посмотреть и нажмите ввод. Нажав **r** вы можете ответить на письмо. Как только вы завершили с набором ответа, нажмите **Ctrl+X**, чтобы послать сообщение. Вы можете нажать **i**, чтобы вернуться к списку сообщений.

Если вы хотите удалить сообщение, нажмите **d**. Это пометит сообщение, на котором находится курсор для удаления. **pine** удаляет письма, когда вы выходите из программы. **pine** так же позволяет сохранять письма в папках. Вы можете получить список папок, нажав **l**. В списке сообщений нажмите **s**, чтобы сохранить подсвеченное сообщение в другую папку. Вам надо будет ввести имя папки, в которую вы хотите сохранить сообщение.

pine предлагает много, много разных функций; вам определённо следует заглянуть в **man** страничку. Там вы найдёте свежую информацию о программе.

elm

elm(1) это другой популярный текстовый клиент электронной почты. Его интерфейс не так дружественен, как интерфейс **pine**, эта программа явно гораздо старше.

По умолчанию вы попадаете в папку входящих сообщений. Сообщения высвечиваются с их номером, датой, отправителем и темой. Используйте стрелки, чтобы подсветить нужное вам сообщение. Нажмите ввод, чтобы просмотреть его.

Чтобы написать новое сообщение, нажмите **m** с главного экрана. **d** пометит сообщение для удаления. А **r** используется для ответа на сообщение, которое вы читаете. Все эти кнопки перечислены внизу экрана.

man страничка более детально обсуждает **elm**, так что вам наверное стоит заглянуть туда перед тем, как пользоваться программой.

mailx

mailx(1) это почтовый клиент, управляемый командной строчкой. Он очень примитивен и практически ничего не предложит вам в качестве интерфейса пользователя. Тем не менее программа очень полезна, когда вам надо быстро отправить что-то, либо если вы хотите написать сценарий отправки почты, или что-то в этом роде.

Обычный вариант запуска вот такой:

$mailx -s <тема> <кому>

mailx читает тело письма из стандартного ввода. Так что вы можете **cat** любой файл в эту команду, чтобы отправить его, или же вы можете просто напечатать текст и нажать **ctrl+D**, когда закончите набор.

Вот пример отправления по почте файла исходника программы другой персоне.

$cat randomfunc.c | mailx -s "Вот эта функция"
asdf@example.net

man страница объясняет, более подробно возможности **mailx**, так что вы наверное загляните туда перед использованием программы.

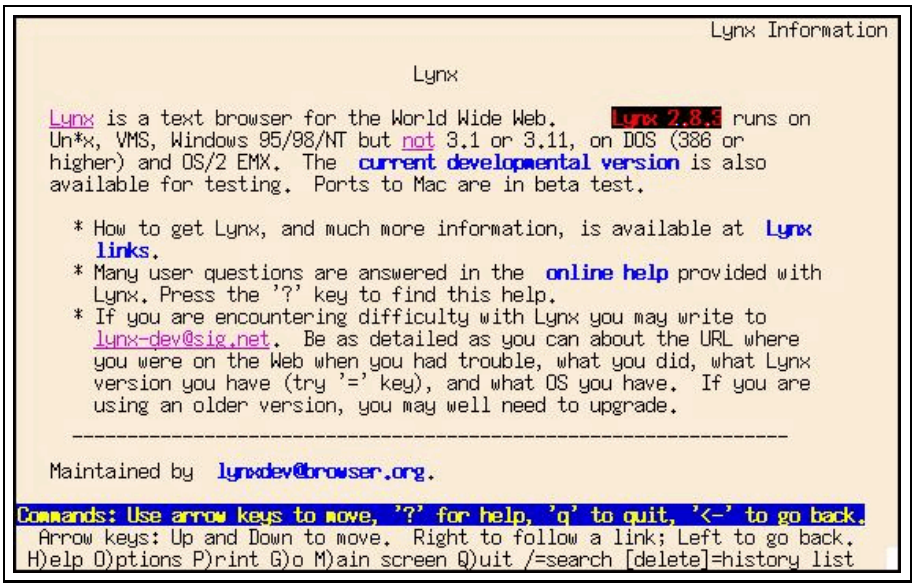
5.6.6 lynx

lynx(1) это вэб броузер текстового режима. Это очень быстрый способ для

поиска чего-то в интернет. Иногда графика это лишнее, если вы знаете, чего вы ищите.

Для запуска **lynx**, просто напечатайте lynx в приглашении командной строки:

$lynx



Вы так же можете захотеть указать сайт, который хотите открыть с **lynx**:

$lynx http://www.slackware.com

lynx печатает командные клавиши и их функции внизу экрана. Кнопки вверх и вниз позволяют пролистывать документ, ввод выбирает подсвеченную ссылку, а кнопка "влево" возвращает вас к предыдущей странице. Нажав **d** вы загрузите выделенный файл. Клавиша **g** покажет строку, где вы можете ввести URL, который хотите открыть.

Есть так же много других команд. Вы можете обратиться к man странице, или нажать **h** чтобы получить экран помощи.

5.6.7 wget

wget(1) это утилита командной строки, которая скачивает файлы с указанного URL. Она полезна при скачивании целых вэб сайтов, для просмотра в offline, или для более надёжной загрузки файлов с HTTP или FTP серверов, чем из Netscape. Основной синтаксис:

$wget <url>

Вы так же можете указать опции. Например, такая команда загрузит Slackware вэб сайт:

$wget –recursive http://www.slackware.com

wget создаст каталог `www.slackware.com` и запишет в него файлы, как они расположены на сайте.

wget так же может загружать файлы с FTP сайтов; просто укажите FTP URL вместо HTTP URL-а.

У **wget** есть гораздо больше опций, что делает её пригодной для использования в скриптах (зеркалирование вэб сайтов, и т.д.). **man** страничка будет вашей следующей остановкой.

5.6.8 traceroute

Slackware включает 4.4BSD **traceroute**(8) программу. Это полезный инструмент для диагностики системы, **traceroute** показывает каждый хост, через который пакеты путешествуют на пути достижения цели. Вы можете посмотреть, сколько "прыжков" пакеты совершают, путешествую от вас до Slackware вэб сайта, воспользовавшись такой командой:

$traceroute www.slackware.com

Каждый из хостов будет показан, вместе со временем ответа. Вот пример вывода:

```
$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
```

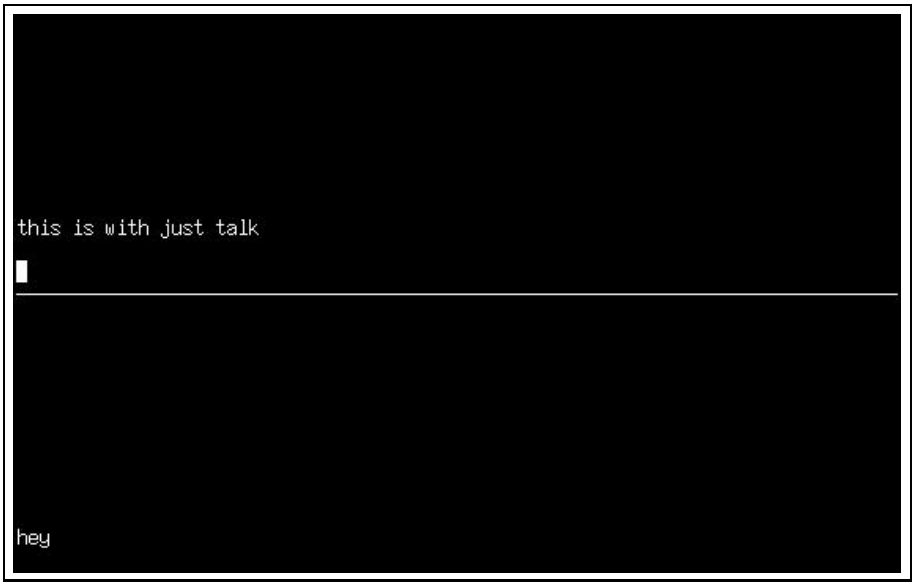
traceroute сходна с **ping** в том, что так же использует ICMP пакеты. И у этой программы есть несколько опций. По умолчанию максимальное число хостов 30, но его можно изменить, указав "-m" опцию. Другие опции детально описаны на вэб страничке.

5.6.9 Общение с другими людьми

talk

talk(1) позволяет пользователям переговариваться между собой при помощи текстовых сообщений. Экран разделяется горизонтально на две половины. Чтобы запросить общение с другой персоной, воспользуйтесь такой командой:

$talk <персона> [имя tty]



Если вы укажете только имя пользователя, запрос чата рассматривается как локальный, так что только локальные пользователи могут принять участие. Имя tty надо указывать, если вы хотите "позвонить" пользователю на специфический терминал, например, если у него несколько открытых терминалов. Необходимая для **talk** информация может быть получена при помощи команды **w**(1).

talk так же может быть использован для вызова пользователей удалённо. Для имени пользователя просто укажите его e-mail адрес, и **talk** попробует соединиться с этим удалённым пользователем на указанном удалённом хосте.

talk довольно ограничен в возможностях. Он работает только с двумя пользователями и наполовину дуплексен.

ytalk

ytalk(1) это замена **talk**. В Slackware она поставляется, как **ytalk** команда. Синтаксис аналогичен, но имеет несколько отличий:

$ytalk <username>[#ttyname]

Имя пользователя и терминал указываются так же, как и в **talk**, только указать вы их должны вместе, разделив символом хэш (#).

ytalk предлагает несколько преимуществ:

- Поддержка более чем двух пользователей.
- Меню доступных опций всегда может быть получено нажатием **Esc**.
- Вы можете выйти из вашей оболочки, оставаясь при этом в **talk** сессии.
- И даже больше...

Если вы администратор сервера, вам следует убедиться, что **ntalk** порт разрешён в `/etc/inetd.conf`. Это необходимо для нормальной работы **ytalk**.

5.6.10 Итог

Вы должны знать некоторые основные команды диагностики сети. Используя их вы можете определить,что именно является источником проблемы при связи, удалённый компьютер, или сеть между ним и вашим компьютером. Так же вы должны были ознакомиться с несколькими программами чтения новостей, вэб броузерами, ftp клиентами и коммуникационными программами.

5.7 Архиваторы

В Slackware Linux есть несколько программ, которые могут быть использованы для сжатия и архивирования файлов. Эти программы особенно полезны для создания резервных копий документов (backups) и для обмена файлами между компьютерами по сети. Вы можете найти, как программы для работы со стандартными Unix так и со стандартными Windows архиваторами.

5.7.1 gzip

gzip(1) это GNU программа сжатия. Она берёт один файл и сжимает его. Пример обычного использования выглядит вот так:

```
$ gzip infile
```

Выходной файл будет назван `infile.gz` и почти всегда будет меньше входного. Обратите внимание, что `infile.gz` заменит `infile`. Это значит, что `infile` прекратит своё существование. Останется только его сжатая копия. Обычные текстовые файлы существенно сожмутся, в то время как jpeg картинки, mp3, и другие подобные файлы почти не сожмутся, так как они уже сжаты. Приведённый выше пример, это нечто среднее между качеством сжатия и затраченным временем. Максимальное сжатие может быть получено при помощи такой команды:

```
$ gzip -9 infile
```

Это займёт больше времени, но выходной файл будет настолько сжатым, насколько **gzip** вообще может его сжать. Использование меньших значений займёт меньше времени, но соответственно и качество компрессии будет хуже.

Распаковывание gzipped (запакованных GNU zip) файлов может быть выполнено при помощи двух команд, которые на самом деле являются одной и той же программой. **gzip** распакует любой файл с узнаваемым им расширением. Вот список расширений, которые узнаёт команда: `.gz`, `-gz`, `.z`, `-z`, `.Z`, или `-Z`. Первый метод — применить команду **gunzip**(1) к файлу:

```
$ gunzip infile.gz
```

Выполнение этой команды приведёт к тому, что вместо указанного файла в этом же каталоге появится его распакованная версия и `.gz` часть его имени исчезнет.

Второй метод распаковывания gziped файла, это вызвать **gzip** в применении к файлу:

```
$ gzip -d infile.gz
```

Это приведёт к точно такому же результату, как и вызов **gunzip**. Объяснение очень просто: **gunzip** это всего лишь символическая ссылка на `/bin/gzip`:

```
$ cd /usr/bin
$ ls -l gunzip
lrwxrwxrwx 1 root root 9 Feb 2 09:45 gunzip -> /bin/gzip
```

Так что выполнение **gunzip** на самом деле лишь вызов **gzip** хоть и под другим именем. Программа может определить, по какому имени к ней обратились, и работает в соответствии с этим. В этом случае, **gzip** увидит, что его вызвали как **gunzip** и распакует файл. Поэтому вы можете на ваше усмотрение пользоваться любой из команд, для распаковки gziped файлов.

5.7.2 bzip2

bzip2(1) это альтернативная программа сжатия, установленная в Slackware Linux. Она использует алгоритм отличный от **gzip**, который имеет как преимущества, так и недостатки. Главное преимущество **bzip2** это размер сжатых файлов. **bzip2** почти всегда сожмёт лучше, чем **gzip**. Иногда файлы получаются гораздо меньше, чем фалы сжатые **gzip**-ом. Это может быть значительным преимуществом для людей, с медленным модемным соединением.

Недостаток **bzip2** в том, что она более интенсивно использует CPU, чем **gzip**. А это значит, что использование **bzip** займёт больше времени и будет более требовательно к процессору, чем **gzip**. Когда вы решаете, каким архиватором пользоваться, надо взвесить это соотношение скорость – сжатие, и выбрать, что важнее.

Использование **bzip2** очень похоже на использование **gzip**, так что мы не станем много времени тратить на её обсуждение. Просто вызовите **bzip2**, указав имя файла:

```
$ bzip2 infile
```

Вывод обычно будет меньше, чем входной файл, и получит название **infile.bz2**. Как и с **gzip**, входной файл будет заменён сжатым.

Вы можете так же указывать числовой аргумент, чтобы балансировать скоростью и качеством сжатия, как и с **gzip**. Следующий пример показывает, как достигнуть максимального сжатия при помощи **bzip2**:

```
$ bzip2 -9 infile
```

Есть два способа распаковывания файлов, заканчивающихся `.bz2` расширением, как и с **gzip**. Вы можете использовать **bzip2** или **bunzip2**(1) для распаковки bzipped файлов. Использование **bzip2** потребует указания аргумента:

```
$ bzip2 -d infile.bz2
```

Эта команда распакует bziped файл и заменит его распакованной копией. Этот результирующий файл потеряет `.bz2` расширение. Аналогично, вы можете использовать **bunzip2** для распаковки файла:

```
$ bunzip2 infile.bz2
```

все произведённые программой действия будут абсолютно идентичными, так как опять мы имеем дело с символьной ссылкой. Проверка `/bin/bunzip2` показывает, что это просто символьная ссылка на `/bin/bzip2`. Используется тот же трюк, что и с **gzip**. Вы увидите, что вызов программ при помощи нескольких различных имён, для получения разного их поведения, это любимый трюк Linux программистов.

```
$ cd /bin
$ ls -l bunzip2
lrwxrwxrwx 1 root root 5 Feb 2 09:45 /bunzip2 -> bzip2
```

5.7.3 tar

tar(1) это GNU ленточный архиватор. Он берёт несколько файлов или каталогов и создаёт один большой файл. Это позволяет вам сжимать целое дерево каталогов, чего нельзя достигнуть при простом использовании **gzip** или **bzip2**. **tar** имеет много параметров командной строки, которые описаны в man странице программы. В этом разделе будут рассмотрены наиболее

распространённые варианты использования **tar**.

Чаще всего **tar** используется для распаковки и раз-архивирования пакетов, скачанных с вэб или ftp сайтов. Большинство файлов будут иметь `.tar.gz` расширение. Это так называемый "tarball". Это означает, что несколько файлов были помещены в архив при помощи **tar** и затем этот архив был сжат при помощи **gzip**. Иногда они так же имеют расширение `.tar.Z`. Это означает то же самое, но обычно встречается на более старых Unix системах.

Так же вы можете встретить иногда `.tar.bz2` файлы. Исходный текст ядра поставляется в таком виде, потому что так вам придётся скачивать меньший файл. Как вы уже вероятно догадались, это несколько файлов, объединённых в архив при помощи **tar** и сжатых при помощи **bzip2**.

Вы можете получить файлы из таких архивов, при помощи **tar** команды с определёнными аргументами командной строки. Разархивирование tarbal-а требует указания ключа **-z**, что фактически вызовет вначале выполнение **gunzip**, для распаковки файла. Обычно tarbal-ы распаковываются такой командой:

```
$ tar -xvzf hejaz.tar.gz
```

Довольно много опций. Что же все они означают? "-x" значит извлечь (extract). Это важно, так как именно этот параметр говорит **tar**-у, что именно делать с входным файлом. В этом случае мы опять разобьём архив на все те файлы, из которых он был составлен. "-v" скажет программе быть "многословной" (verbose). Указание этого ключа приведёт к тому, что в процессе извлечения будет выводиться список файлов, которые извлекаются. Вы можете смело отказаться от использования этого ключа, если вас раздражает вывод подобной информации. Так же вы можете использовать "-vv", чтобы программа стала совсем многословной, и отображала ещё больше информации о файлах, которые извлекаются. опция "-z" говорит **tar**-у, вначале пропустить файл`hejaz.tar.gz` через **gunzip**. Ну и наконец, "-f" опция указывает, что далее в командной строке будет указанно имя файла, с которым надо работать.

Есть так же несколько других способов написания той же самой команды. На более старых системах, в которых отсутствует приличная версия GNU tar, вы можете встретить такой синтаксис:

```
$ gzip -dc hejaz.tar.gz | tar -xvf -
```

Эта строчка команд вначале распакует файл и затем пошлёт вывод **tar**-у. Так как **gzip** запишет вывод на стандартный вывод, если его попросить, эта команда запишет распакованный файл в стандартный вывод. Через pipe файл будет послан **tar**-у для раз-архивирования. "-" означает работать со стандартным вводом. Программа разархивирует поток данных, который она получит от **gzip** и запишет вывод на диск.

Другой способ записи первой команды это запись без тире перед опциями:

```
$ tar xvzf hejaz.tar.gz
```

Возможно вам так же понадобится работать с bziped архивами. Версия **tar**, поставляемая со Slackware Linux может работать с ними так же как и с gziped архивами. Вместо "-z" опции вам надо воспользоваться "-y":

```
$ tar -xvyf foo.tar.bz2
```

Следует отметить, что **tar** поместит извлечённые файлы в текущем каталоге. Так что если ваш архив находится в `/tmp` каталоге, а распаковать

вы его хотите в домашний каталог, есть два варианта решения проблемы. Первая — архив можно переместить в домашний каталог и затем обработан с **tar**. Или же вы можете указать путь к архиву в командной строке:

```
$ tar -xvzf /tmp/bar.tar.gz
```

Содержимое архива будет вывалено в ваш домашний каталог, а исходный архив так и останется в `/tmp` каталоге.

Второй наиболее распространённый вариант использования **tar** это создание ваших собственных архивов. Создание архива не более сложная процедура, чем разархивирование других файлов; просто требует другого набора опций командной строки.

Чтобы создать сжатый **tar** архив всех файлов текущего каталога (включая поддиректории и их файлы), вам следует воспользоваться командой:

```
$ tar -cvzf archive.tar.gz .
```

В этой командной строке "-c" опция указывает, что должен быть создан архив, а "-z" сжимает получившийся архив при помощи **gzip**. `archive.tar.gz` это файл, который вы хотите создать. Вы можете назвать его как вам заблагорассудится, а если вы так же укажете полный путь, то файл будет создан в указанном каталоге. Вот пример:

```
$ tar -cvzf /tmp/archive.tar.gz .
```

В этом случае архив будет создан в `/tmp` каталоге. Вы можете так же указать все файлы и каталоги, которые вы хотите включить в архив, перечислив их в конце команды. В этом случае `.` это каталог, который будет включён в архив. Вы можете заменить его на список всевозможных файлов или всего того, что захотите включить в архив.

5.7.4 zip

Ну и наконец, есть две утилиты для работы с `zip` файлами. Которые являются очень популярными в мире WIndows, итак в Linux есть программы для работы с ними. Программа для сжатия называется **zip**(1), а программа для распаковки называется **unzip**(1).

Сжимать довольно легко:

```
$ zip foo *
```

Эта команда создаст файл `foo.zip`, который будет содержать все файлы в текущем каталоге. **zip** автоматически добавит `.zip` расширение, так что вам не надо указывать его в командной строке. Вы так же можете пробежаться по текущему каталогу, пакуя все каталоги, которые там есть:

```
$ zip -r foo *
```

Распаковывать файлы тоже не сложно.

```
$ unzip foo
```

Это распакует все файлы из файла `foo.zip`, включая все каталоги, присутствующие в архиве.

zip утилиты имеют несколько расширенных опций для создания самораспаковывающихся (self-extracting) архивов, для пропуска некоторых из файлов, управления размером сжатого файла, вывода на экран отчётов и гораздо больше. Смотрите man страницы для **zip** и **unzip**, чтобы узнать как использовать эти опции.

5.7.5 Итог

В этом разделе обсуждались программы, которые были использованы для сжатия и распаковывания архивных файлов. Вы должны знать, что такое файл-архив, как создать таковой при помощи **tar** и вид сжатия на ваш выбор, как извлечь файлы из архива и как работать с Windows-овскими архивами. Почти всегда при скачивании и закачивании вы будете иметь дело с архивами, так что важно уметь работать с ними.

5.8 vi

vi(1) это стандартный текстовый редактор Unix, который используется в основном системными администраторами. Есть несколько версий (клонов) **vi**: **vi**, **elvis**, **vile**, и **vim**. Один из них есть практически в любой версии Unix, и Linux. Все эти версии включают идентичный основной набор команд, так что изучив основные команды одного из клонов, вам будет легко перейти к использованию другого.

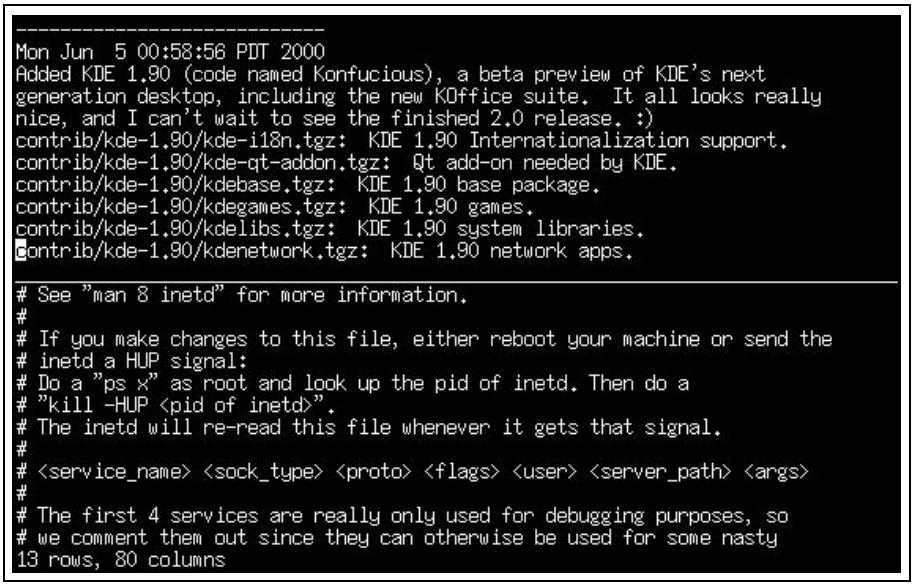
vi включает множество мощных вспомогательных инструментов, включая подсветку синтаксиса, форматирование кода, мощный поиск-замена механизм, макросы и многое другое. Эти инструменты делают программу особенно привлекательной для программистов, вэб разработчиков и т.д. Системные администраторы оценят возможный автоматизм и интегрированность с оболочкой.

В Slackware Linux, версия **vi** по умолчанию, это **elvis**. Другие доступные версии, такие как **vim** и **gvim**, доступны, если вы установили соответствующие пакеты. **gvim** это X Window версия **vim**, с панелями инструментов, всплывающими меню и диалогами.

5.8.1 Запуск vi

vi может быть запущен из командной строки множеством различных способов:

$ vi



Такая команда запустит **vi** с пустым буфером. Вы увидите почти пустой экран. Программа сейчас в "командном режиме" и ожидает, пока вы сделаете что -либо. Для обзора различных режимов работы редактора смотрите подраздел 5.8.2. Чтобы выйти из **vi** напечатайте:

:q

Если вы ничего не изменяли в открытом файле, это приведёт к выходу из программы. Если же вы изменили что либо, программа скажет вам, что

содержимое файла было изменено и подскажет вам, что напечатать, чтобы выйти, проигнорировав все изменения.

Вы можете так же открыть при помощи **vi** уже существующий файл. Например, чтобы открыть `/etc/resolv.conf`, вам надо выполнить:

```
$ vi /etc/resolv.conf
```

Так же, **vi** может открыть файл с указанной строчки. Например, вы можете запустить **vi** со строки 47 `/usr/src/linux/init/main.c`, выполнив такую команду:

```
vi +47 /usr/src/linux/init/main.c
```

vi выведет на экран указанный файл, и установит курсор на указанную строку. В случае, если номер строки больше номера последней строки файла, **vi** поместит курсор на последнюю строку. Это очень удобно для программистов, так как они могут открыть файл со строки, в которой была обнаружена ошибка, и таким образом, отпадает необходимость поиска строки вручную.

5.8.2 Режимы vi

vi оперирует в разных режимах, которые используются для выполнения различных задач. Когда вы только запустили **vi**, вы попадаете в командный режим. Отсюда вы можете выполнять разные команды для манипулирования текстом, перемещаться внутри файла, сохранить, выйти и это ещё не всё. Для изменения текста используется режим вставки (insert). Вы можете переключаться в различные режимы, при помощи комбинаций клавиш, которые описаны ниже.

Командный режим

Вначале вы попадаете в командный режим. Из этого режима вы не можете напрямую вводить текст, или редактировать уже существующий текст. Но вы можете манипулировать текстом, искать, выйти, сохранить, загружать другие файлы, ... Это всего лишь обзор командного режима. Для рассмотрения разных команд, сотрите подраздел 5.8.7.

Вероятно, наиболее часто используемая команда командного режима, это переход в режим вставки. Нажмите **i** и вы попадёте в режим вставки. Курсор изменит свою форму и "– INSERT –" высветится в нижней части экрана (в клонах программы это не произойдёт). В этом режиме вы можете вводить текст, и он будет отображаться на экране. Чтобы вернуться в командный режим, нажмите **Esc** кнопку.

Командный режим так же удобен для перемещения внутри файла. На некоторых системах вы можете использовать стрелки для перемещения. В других системах вам придётся воспользоваться более традиционными "hjkl". Вот список того, как эти кнопки используются для перемещений:

- **h** перейти влево на один символ
- **j** перейти вниз на один символ
- **k** перейти вверх на один символ
- **l** перейти вправо на один символ

Просто нажмите на соответствующую букву, для перемещения необходимом направлении. Как будет показано позже, эти команды могут использованы в комбинации с числом, для более эффективного перемещения.

Многие команды, используемые в командном режиме начинаются с двоеточия. Например, для выхода используется **:q**, как уже упоминалось выше. Двоеточие просто показывает, что это команда, в то время, как "q" говорит **vi** что вы хотите покинуть программу. Другой тип команд представлен комбинацией числа (необязательного) и буквы. Перед такими командами не надо указывать двоеточие, они используются в основном для манипулирования текстом.

Для примера, чтобы удалить одну строку из файла, нажмите **dd**. Что приведёт к удалению строки, на которой находится курсор. Выполнение команды **4dd** укажет **vi** удалить строку, на которой находится курсор, и три строки после неё. В общем, число указывает **vi**, сколько раз выполнить команду.

Вы можете комбинировать число с командами перемещения. Например, **10k** переместит вас на 10 строк вверх по тексту.

Командный режим так же может быть использован для вырезания и вставки текста, вставки текста, для считывания других файлов в текущий буфер. Копирование текста осуществляется при помощи **y** кнопки (y от yank). Копирование текущей строки выполняется нажатием **yy** и может быть использовано с предшествующим числом, для копирования нескольких строк. Затем перейдёте к месту, куда хотите вставить скопированный текст, и нажмите **p**. Текст будет вставлен в строку, следующую за текущей.

Вырезание текста выполняется при помощи **dd** и **p** используется для вставки вырезанного текста обратно в файл. Считывание текста с другого файла, это довольно простая процедура. Просто наберите **:r**, пробел и имя файла, содержащего текст, который вы хотите вставить. Содержимое файла будет помещено в текущий буфер, со строки следующей за той, на которой находится курсор. Более усовершенствованные клоны **vi** завершают имена файла, на подобие того, как это работает в оболочке.

Последний вариант использования режима, который будет описан здесь, это поиск. Режим команд позволяет выполнять как простой поиск, так и расширенный поиск с заменой. В этом разделе будет описан только команда простого поиска.

Для поиска нажмите на кнопку / и введите текст, который вы хотите найти. **vi** будет искать от того места, где находится курсор, в направлении к концу файла, и остановится при нахождении первого совпадения. Заметьте, что не точные совпадения так же заставят **vi** остановиться. Например, поиск "the" заставит **vi** остановиться на "then", "therefore" и т.д. Это произойдёт потому, что все эти слова содержат "the", но только в начале.

После того, как **vi** нашёл первое совпадение, вы можете продолжить, поиск следующего, простым нажатием / и ввод. Можно так же задавать поиск в направлении от курсора, к началу файла, для этого вместо слэш вам надо воспользоваться **?**. Например, для поиска в обратном направлении "the" вам необходимо выполнить команду **?the**.

Режим вставки

Вставка и замена текста осуществляется в режиме вставки. Как сообщалось выше, для перехода в этот режим из командного, вам надо нажать **i**. Далее, весь напечатанный вами текст вводится в текущий буфер. Нажатие **Esc** вернёт вас в командный режим.

Замена текста осуществляется несколькими способами. Нажатие **r** из командного режима позволит вам заменить один символ, подсвеченный курсором. Просто введите новый символ и он займёт место подсвеченного курсором. Сразу же после этого вы вернётесь в командный режим. Нажатие **R** позволяет вам заменить столько символов, сколько вы пожелаете. Чтобы выйти из режима замещения, просто нажмите **Esc** и вы попадёте в командный режим.

Есть так же и другой способ переключения между вставкой и заменой. Нажатие кнопки **Insert** из командного режима переведёт вас в режим вставки. Если вы уже в режиме вставки, кнопка **Insert** работает, как переключатель между режимом вставки и замены.

5.8.3 Открытие файлов

vi позволяет открывать файлы как из командного режима, так и из командной строки, указав имя файла после имени программы. Чтобы открыть файл `/etc/lilo.conf`:

```
:e /etc/lilo.conf
```

Если вы произвели изменения в текущем буфере и не сохранили их, **vi** скажет вам об этом. Вы всё же можете открыть файл, без записи текущего буфера, набрав **:e!**, пробел и имя файла, который вы хотите открыть. Обычно предупреждения программы могут быть проигнорированы указанием восклицательного знака после команды.

Если вы хотите перечитать текущий файл, вы можете сделать это, набрав **e!**. Это удобно, если вы каким-то образом что-то напортили и хотите перечитать файл.

Некоторые **vi** клоны (например, **vim**) позволяют открыть одновременно несколько буферов. Например, чтобы открыть файл `09-vi.sgml` в моём домашнем каталоге, в то время, как другой файл уже открыт, я наберу:

```
:split  /09-vi.sgml
```

Новый файл отображается в верхней половине экрана, а старый файл отображается в нижней половине. Есть много команд для управления разделённым экраном, и многие из них напоминают EMACS. Лучшей ссылкой по использованию этих команд будет man страница для вашего **vi** клона.

5.8.4 Сохранение файлов

Для сохранения файлов в **vi** может быть использовано несколько разных команд. Если вы хотите сохранить текущий буфер в файл `randomness`, вам надо набрать:

```
:w randomness
```

После того, как вы однажды сохранили файл, для повторной записи в него, вы можете просто набрать **:w**. Все изменения сохранятся в файл. После сохранения файла, вы попадаете в командный режим. Если вы хотите сохранить изменения и выйти, из **vi** (что случается довольно часто), вам надо набрать **:wq**. Это укажет **vi** сохранить изменения и выйти.

Иногда необходимо сохранить файл, помеченный только для записи. Вы можете сделать это, указав восклицательный знак после команды записи, вот так:

:w!
:wq!

Все же может случится, что вы не сможете записать изменения, которые вы произвели в файл (например, если файл принадлежит другому пользователю). Если это случится, **vi** сообщит вам, что он не может записать файл. Если вы действительно хотите изменить файл, вам придётся вернуться и изменить его, как root.

5.8.5 Выход из vi

Один из методов выхода из **vi** это использование **:wq**, что приведёт к сохранению текущего буфера перед выходом. Вы так же можете выйти без охранения изменений, при помощи **:q** или **:q!**. Последний вариант используется когда вы изменили файл, но не хотите сохранить эти изменения.

Может случиться, что произойдёт сбой при работе вашего компьютера, или программы **vi**. Тем не менее, оба клона (и **elvis** и **vim**) предпримут шаги по минимизированию потерь всех открытых буферов. Оба редактора сохраняют открытые буфера во временный файл. Этот файл обычно называется по аналогии с открытым файлом, но с точкой вначале. Это делает файл скрытым.

Этот автоматически удаляется, как только вы нормально завершили редактирование файла. Это значит, что если произошёл сбой, то резервный файл всё ещё будет существовать. И если вы захотите опять редактировать файл, программа спросит вас, как поступить. В большинстве случаев большая часть вашей не записанной работы может быть восстановлена. **elvis** так же пошлёт вам письмо (из Graceland, что достаточно странно :) сообщающее о существующей резервной копии.

5.8.6 Настройка vi

Выбранный вами **vi** клон может быть настроен несколькими способами.

Множество команд может быть введено в командном режиме для настройки практически всего, что угодно. В зависимости от вашего редактора, вы можете активизировать функции, упрощающие программирование (такие, как подсветку синтаксиса, авто-отступы, и более), установите макрос для автоматизирования задач, активизации подстановки текста, и более.

Почти все из этих команд могут быть помещены в настроечный файл в вашем домашнем каталоге. **elvis** ищет `.exrc` файл, а **vim** ищет `.vimrc` файл. Большинство конфигурационных команд, которые могут быть введены из командной строки, могут быть размещены в настроечный файл. Включая установочную информацию, подстановку текста, макросы, и более.

Обсуждение всех этих опций и различий между редакторами, не является целью данной книги, так что если вы заинтересовались расширенными функциями редакторов, загляните в man страницу или зайдите на вэб сайт предпочитаемого вами редактора. Некоторые редакторы (как **vim**) содержат мощную справочную систему, доступную внутри редактора, по команде **:help**. Вы так же можете заглянуть в книгу "Изучаем Редактор vi" Ламба и Роббинса.

Операция	**Кнопка**
влево, вниз, вверх, вправо	**h, j, k, l**
перейти в конец строки	**$**
перейти в начало строки	^
перейти в конец файла	**G**
перейти в начало файла	**:1**
перейти к 47-й строке	**:47**

Таблица 5.2: Перемещения

Операция	**Кнопка**
удалить строку	**dd**
удалить пять строк	**5dd**
заменить символ	**r**
удалить символ	**x**
удалить десять символов	**10x**
отменить последнее действие	**u**
объединить текущую строку со следующей	**J**

Таблица 5.3: Редактирование

Многие программы в Linux по умолчанию откроют текстовые файлы в **vi**. Например, редактирование ваших crontab задач загрузит **vi** по умолчанию. Если вам не нравится **vi** и хотите, чтобы другая программа запускалась по умолчанию, всё что вам надо сделать, так это установить значение VISUAL переменной окружения равное имени вашего любимого редактора. Для информации по установке переменных окружения смотрите раздел 5.1.3 на стр. 87. Если вы хотите, чтобы ваш любимый редактор был редактором по умолчанию при каждом входе в систему, добавьте установку VISUAL переменной в ваш `.bash_profile` или `.bashrc` файл.

5.8.7 Кнопки vi

Этот раздел является чем то вроде быстрого справочника многих основных **vi** команд. Некоторые из которых уже упоминались выше, а некоторые будут для вас новыми.

5.8.8 Итог

Теперь у вы должны быть некоторые основные знания по стандартному текстовому редактору Unix **vi**. **vi** это довольно сложная программа, с мно-

Операция	**Кнопка**
найти "asdf"в тексте после курсора	**/asdf**
искать "asdf"в тексте до курсора	**?asdf**
повтор последнего поиска в тексте после курсора	/
повтор последнего поиска в тексте до курсора	**?**

Таблица 5.4: Поиск

Операция	**Кнопка**
выход	**:q**
выход без записи	**:q!**
записать и выйти	**:wq**
записать, без выхода	**:w**
пере-загрузить текущий файл	**:e!**
записать содержимое буфера в файл `asdf`	**:w asdf**
открыть файл `hejaz`	**:e hejaz**
загрузить файл `asdf` в буфер	**:r asdf**
загрузить вывод **ls** в буфер	**:r !ls**

Таблица 5.5: Сохранение и выход

жеством команд и настроечных опций. Всё же, вы должны уметь открыть файл, перемещаться по тексту, редактировать текст и выходить. Это всё, что вам понадобится для выполнения каждодневных задач. По мере возникновения необходимости в более мощном редакторе, вы можете использовать документацию **vi** для изучения его возможностей.

5.9 Управление пакетами Slackware

Пакет программ это упакованный набор связанных программ, которые готовы для установки. Когда вы скачиваете архив исходных текстов программ, вам надо конфигурировать, компилировать и установить всё вручную. С пакетом программ это уже было проделано за вас. Всё, что вам надо сделать, это установить пакет. Другое очень удобное преимущество пакетов в том, что их можно очень легко удалить или обновить, если вы пожелаете. Slackware поставляется со всеми необходимыми пакетами для управления пакетами. Вы можете устанавливать, удалять, обновлять, создавать и проверять пакеты очень легко.

5.9.1 Обзор формата пакетов

Перед тем, как приступить к изучению утилит, вам надо ознакомиться с форматом пакетов Slackware. Пакет, это всего навсего **tar** архив, сжатый при помощи **gzip**. Пакет устроен таким образом, что он может быть развёрнут в корневом каталоге файловой системы.

Здесь приведена вымышленная программа и пример 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 управления пакетами развернёт этот файл в корневом каталоге, чтобы установить его. Будет сделана соответствующая запись в базе данных пакетов, включающая содержание этого пакета, чтобы он мог быть обновлён или удалён позже.

Обратите внимание на под-каталог `install/`. Это специальный каталог, который может содержать скрипт, который будет выполнен после копирования файлов. Скрипт называется `doinst.sh`. Если система обработки пакета найдёт такой скрипт, он будет выполнен после установки пакета.

Другой сценарий может быть использован в пакете, но мы обсудим это более детально в подразделе 5.9.3.

5.9.2 Утилиты пакетов

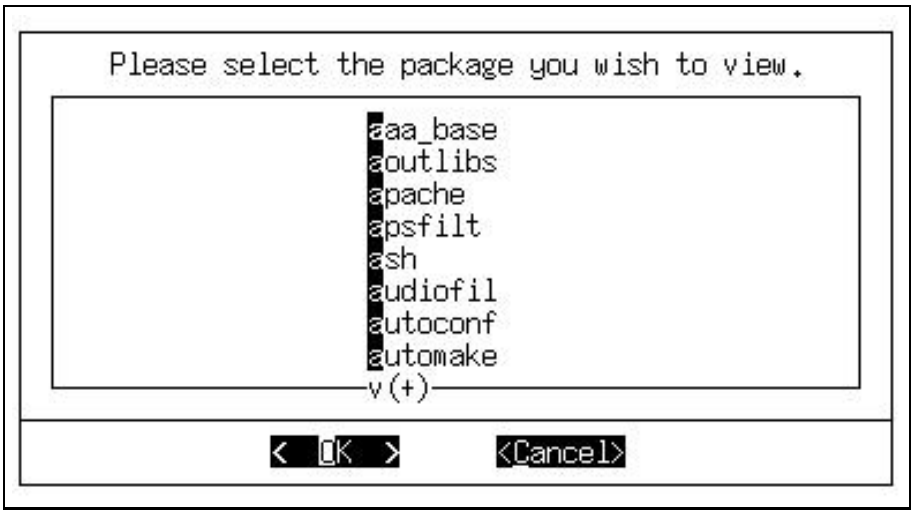
Есть четыре основных утилиты для управления пакетами. Они могут устанавливать, удалять и обновлять пакеты.

pkgtool

pkgtool(8) это управляемая меню программа, позволяющая установку и удаление пакетов. Основное меню выглядит вот так:

Установка может быть произведена из текущего каталога, другого каталога, или с флоппи дисков. Просто выберите метод, который вы хотите использовать и **pkgtool** просмотрит выбранное вами расположение, на факт наличия действительных пакетов для установки.

Вы так же можете просмотреть список пакетов, который будет выглядеть вот так:



Если вы хотите удалить пакеты, выберите remove опцию, и вы увидите список всех установленных пакетов, с возможностью выбора нескольких из них для удаления. Пометьте те, которые вы хотите удалить, и нажмите **OK**. **pkgtool** удалит их.

Некоторые пользователи предпочитают эту команду утилитам командной строки. Тем не менее, следует отметить, что утилиты командной строки предоставляют гораздо больше опций. Так же, возможность обновлять пакеты реализована только, как утилита командной строки.

Опция	Действие
-m	Производит **makepkg** операцию с текущим каталогом.
-warn	Показывает, что случится, если вы установите пакет. Это довольно полезно, так как вы видите в точности, что случится при установке пакета.
-r	Установит рекурсивно все пакеты из текущего каталога. <package name> может так же быть маской имён.

Таблица 5.6: Опции **installpkg**

Опция	Действие
-copy	Пакет будет скопирован в резервный каталог. Это создаст дерево каталогов оригинального пакета, без удаления его.
-keep	Сохраняет временные файлы, которые создаются при удалении. Полезно только для тестирования правильности работы.
-preserve	Пакет удаляется из системы, но в то же время, его копия сохраняется в резервный каталог.
-warn	Показывает, что произойдёт, если вы удалите пакет.

Таблица 5.7: Опции **removepkg**

installpkg

installpkg(8) обеспечивает установку новых для вашей системы пакетов. Синтаксис таков:

 # [ROOT=<path>] installpkg [option] <package name>...

installpkg имеет три опции. Только одна из них может использована при вызове программы.

Если вы указали переменную окружения ROOT, до выполнения **installpkg**, то путь, указанный в переменной будет использован вместо корневого каталога. Это полезно для настройки новых дисков для вашего корневого каталога. Они обычно будут смонтированы к `/mnt` или что-то другое, отличное от `/`.

База данных установленных пакетов есть не что иное, как набор файлов, расположенных в `/var/log/packages`. Для каждого из пакетов там создаётся текстовый файл. Если в пакете есть после-установочный скрипт, он записывается в `/var/log/scripts/<packagename>`.

Вы можете указать несколько пакетов, или даже использовать шаблоны для имён пакетов. Обратите внимание, что **installpkg** не скажет вам, если вы переписываете уже установленный пакет. Она просто установит новый пакет поверх старого. Если вы хотите быть уверенны, чтобы файлы от старого пакета были удалены из вашей системы, воспользуйтесь **upgradepkg**.

removepkg

removepkg(8) обеспечивает удаление пакетов, уже установленных в вашей системе. Синтаксис таков:

 # [ROOT=<path>] removepkg [option] <package name>...

Для **removepkg** есть четыре опции. Только одна из них может быть использована при вызове программы.

Если вы указали переменную окружения ROOT, до выполнения **removepkg**, то путь, указанный в переменной будет использован вместо корневого каталога. Это полезно для настройки новых дисков для вашего корневого каталога. Они обычно будут смонтированы к `/mnt` или что-то другое, отличное от `/`.

removepkg просматривает так же и остальные пакеты, и удаляет только файлы, являющиеся уникальными для удаляемого пакета. Программа так же просмотрит пост-установочный скрипт пакета, и удалит все символические ссылки, которые были созданы им.

Во время процесса удаления отображается статус. После удаления, база данных пакета перемещается в `/var/log/removed_packages`, а пост-установочный скрипт, в `/var/log/removed_scripts`.

Как и в случае с **installpkg**, вы можете указывать несколько пакетов, или использовать шаблоны для имён пакетов.

upgradepkg

upgradepkg(8) обновит пакет, уже присутствующий в вашей системе.Синтаксис программы таков:

 # [ROOT=<path>] upgradepkg <package name>...

или

**# [ROOT=<path>] upgradepkg **
<old package name>%<new package name>

upgradepkg вначале устанавливает новый пакет, а затем удаляет старый пакет, таким образом, файлы от старого пакета не будут мёртвым грузом висеть в ваших каталогах. Если имя обновляемого пакета изменилось, воспользуйтесь вторым вариантом обращения к программе, указав имя старого пакета (который установлен в системе) и после знака процента имя нового пакета (который вы устанавливаете) .

Если вы указали переменную окружения ROOT, до выполнения **removepkg**, то путь, указанный в переменной будет использован вместо корневого каталога. Это полезно для настройки новых дисков для вашего корневого каталога. Они обычно будут смонтированы к `/mnt` или что-то другое, отличное от `/`.

upgradepkg не безупречна. Вам всегда следует создавать копии ваших конфигурационных файлов. Если они будут переписаны, у вас всё ещё будет копия оригиналов.

Как и с **installpkg** и **removepkg**, вы можете указать несколько пакетов, или использовать шаблоны для имён пакетов.

rpm2tgz/rpm2targz

Red Hat Package Manager это популярный сегодня тип системы пакетов. Многие производители программ предоставляют свои программы в RPM формате. Так как это не наш родной формат, мы не рекомендуем нашим пользователям использовать их. Тем не менее, некоторые вещи доступны только в RPM (даже исходники).

Мы предоставляем программы, которая конвертирует RPM пакеты в

наш родной `.tgz` формат[5]. Это позволит вам распаковать пакет (скажем, при помощи **explodepkg**) в временный каталог и исследовать содержимое.

Программа **rpm2tgz** создаст Slackware пакет с `.tgz` расширением, а **rpm2targz** создаст архив с `.tar.gz` расширением.

5.9.3 Создание пакетов

Создание Slackware пакетов может быть, как простым, так и сложным. Нет специфического метода создания пакетов. Единственное требование это то, что пакет должен быть **tar gziped** файл, и если нужен после-установочный скрипт, он должен быть в каталоге `/install/doinst.sh`.

Если вы заинтересованы в создании пакетов для вашей системы, или для сети, которую вы обслуживаете, вам следует посмотреть на различные build скрипты в дереве каталогов исходников Slackware. Есть несколько методов, которые мы используем для создания пакетов.

explodepkg

explodepkg (8) проделает то же, что **installpkg** делает при распаковке пакета, но на самом деле не установит сам пакет, и не произведёт никаких изменений в базе данных пакетов. Она просто развернёт содержимое пакета в текущий каталог.

Если вы посмотрите На дерево каталогов исходников Slackware, вы увидите, как мы используем эту команду для "структурных" пакетов. Эти пакеты содержат скелет того, как конечный пакет будет выглядеть. Они хранят все необходимые имена файлов (нулевой длины), права доступа и принадлежности. build скрипт проделает cat с содержанием пакета из исходного каталога в build каталог пакета.

makepkg

makepkg(8) запакует текущий каталог в формат Slackware package. Она найдёт все символьные ссылки в дереве и добавит блоки их создания в пост-установочный скрипт пакета. Так же программа выдаст сообщение, если обнаружит файлы нулевой длины в дереве пакета.

Эта программа обычно выполняется после того, как вы создали дерево вашего пакета.

5.9.4 Создание tags и tagfiles (для программы установки)

Программа установки Slackware производит установку пакетов программ в вашу систему. Существуют так же файлы, которые говорят программе установки, какие из файлов должны быть установлены, какие из них необязательны, и какие по умолчанию должны быть выбраны программой установки.

[5] для работы этих программ cpio.tgz должен быть установлен (наблюдение переводчика)

Опция	**Действие**
ADD	Пакет необходим для нормальной работы системы
SKP	Пакет будет автоматически пропущен
REC	Пакет не является необходимым, но рекомендован
OPT	Пакет необязателен

Таблица 5.8: Опции статуса `tagfile`

`tagfile` находится в каталоге первой серии программ и называется `tagfile`. Он перечисляет пакеты различных дисков и их статус. Статус может быть:

Формат прост:

<package name>: <status>

Один пакет на строчку. Оригинальные `tagfiles` для каждого раздела программ хранятся в `tagfile.org`. Так что если вы запутались со своими, вы можете восстановить оригинальные.

Многие администраторы предпочитают писать свои собственные `tagfiles` и затем при установке просто нажимают "full". Программа установки прочтёт `tagfiles` и произведёт установку в соответствии с их содержанием.Если вы используете REC или OPT, то появится окно диалога, которое спросит, следует ли устанавливать этот отдельный пакет. Поэтому рекомендуется использовать ADD и SKP, если вы пишите `tagfiles` для автоматизированной установки.

Просто убедитесь, что ваши `tagfiles` будут расположены в тех же местах, что и оригинальные. Или вы можете так же выбрать "указать специфический путь к пользовательскому `tagfile`".

5.9.5 Итог

Теперь вы должны быть знакомы с идеей пакетов программ и как они используются в Slackware. Вы должны быть знакомы с различными утилитами управления пакетами и знать, как использовать их. Наиболее важные моменты этого раздела, это установка, удаление и обновление пакетов. Это и есть наиболее частое применение утилит пакетов. Так же вы должны иметь хоть общее представление о том, как создаются и проверяются пакеты.

5.10 ZipSlack и BigSlack

5.10.1 Что такое ZipSlack/BigSlack?

"ZipSlack" это специальная версия Slackware Linux. Это уже установленная копия Slackware, готовая к запуску с DOS или Windows раздела. Это она содержит основные компоненты системы. Там нет всех пакетов, входящих в Slackware. Если вы хотите получить все пакеты в виде Zipslack-а, вам надо попробовать "BigSlack".

ZipSlack получил своё имя от формы, в которой он поставляется — большой `.ZIP` файл. Работающие в DOS и Windows скорее всего знакомы с такими файлами. Это сжатые архивы. ZipSlack архив содержит необходимый набор программ для нормальной работы Slackware.

Важно отметить, что ZipSlack и BigSlack значительно отличаются от обычной установки. Даже несмотря на то, что они функционируют одинаково, и содержат одинаковые программы, они предназначаются для разных аудиторий пользователей, и имеют различные предназначения. Несколько преимуществ и недостатков ZipSlack и BifSlack обсуждены ниже.

Всегда заглядывайте в документацию, содержащуюся в ZipSlack и BigSlack каталогах. Там всегда можно найти наиболее свежую информацию о установке, загрузке и основах использования продуктов.

Преимущества

- Не требует пере разметки вашего жёсткого диска.
- Отличный способ изучения Slackware Linux без столкновения с установкой системы

Недостатки

- Uses the DOS filesystem, which is slower than a native Linux filesystem.
- Will not work with Windows NT.

5.10.2 Получение ZipSlack/BigSlack

Получение ZipSlack или BigSlack это довольно простая процедура. Если вы приобрели официальный Slackware Linux CD set, то у вас уже есть ZipSlack и BigSlack. Просто найдите CD, содержащий выбранный вами вариант и вставьте его в ваш привод CD-ROM. Обычно это третий или четвёртый диск.

Если вы хотите скачать ZipSlack или BigSlack, то зайдите вначале на нашу "Get Slack" страничку, для получения последней информации по загрузке:

http://www.slackware.com/getslack/

ZipSlack и BigSlack являются частью каждого релиза Slackware. Найдите там тот из релизов, который вы хотите, и отправляйтесь к соответствующему каталогу на FTP сайте. Каталог, соответствующий последнему релизу, всегда находится по адресу:

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

Вы найдёте ZipSlack в `/zipslack` под-каталоге и BigSlack в `/bigslack` каталоге. ZipSlack поставляется, как один большой `.ZIP` файл или флоппи

ориентированную "нарезку". Нарезка находится в `/zipslack/split` каталоге. BigSlack поставляется только в нарезанном виде.

Не останавливайтесь на скачивании только `.ZIP` файлов. Вам так же надо скачать документацию и образы загрузочных дисков, которые есть в это каталоге.

5.10.3 Установка

После того, как вы скачали все необходимые компоненты, вам надо развернуть `.ZIP` файл (или файлы, если вы загрузили нарезку). Вам необходимо использовать 32-bit unzipper. Размер и имена файлов в архиве слишком велики для 16-bit unzipper. Например, вы можете воспользоваться **WinZip** или **PKZIP** для Windows.

Оба ZipSlack и BigSlack настроены для развёртывания в корневом каталоге (таком, как `C:` или `D:`). Будет создан `\LINUX` каталог, который фактически содержит Slackware. Так же в этом каталоге вы найдёте файлы, необходимые для загрузки системы.

После того, как вы развернули файлы на выбранном вами диске, там появится каталог `\LINUX` (для определённости мы будем везде далее предполагать, что выбран диск `C:`).

5.10.4 Загрузка ZipSlack/BigSlack

Вы можете воспользоваться несколькими методами для загрузки ZipSlack и BigSlack. Наиболее обычный метод, это использование `LINUX.BAT` из DOS (или DOS режима из под Windows 9x). Файл должен быть изменён соответствующим образом для соответствия вашей системе.

Начните с открытия`C:\LINUX\LINUX.BAT` в вашем любимом текстовом редакторе. В шапке файла вы увидите большой комментарий. Он объясняет, что вам надо изменить в файле (а так же, как быть, если вы загружаетесь с внешнего Zip диска). Не волнуйтесь, если root= установка вне вашего понимания. В файле приведено несколько примеров, так что свобода выбора за вами. Если ваша попытка оказалась неудачной, вы можете попробовать опять изменить содержание файла, закомментировав не сработавшую строку, и разкомментировав другую.

После того, как вы разкомментировали нужную вам строку, путём удаления "rem " из начала строки, сохраните изменения и выйдите из редактора. Переведите ваш компьютер в режим DOS.

ПРЕДУПРЕЖДЕНИЕ окно DOS prompt в Windows 9x НЕ сработает.

Наберите `C:\LINUX\LINUX.BAT`, чтобы загрузить систему. Если всё в порядке, перед вами появится приглашение входа в систему (login).

Войдите, как root, без пароля. Скорее всего вы захотите установить пароль для root пользователя, и создать эккаунт для вас самих. Тогда вам надо посмотреть соответствующий раздел этой книги об основах использования системы.

Если файл `LINUX.BAT` не работает в вашей системе, вам следует заглянуть в `C:\LINUX\README.1ST` файл для ознакомления с другими методами загрузки..

5.10.5 Добавка, удаление и обновление программ

В ZipSlack и BigSlack могут быть использованы те же самые пакеты программ, что и в нормальной установке Slackware. Таким образом. вы можете пользоваться стандартными инструментами для управления пакетами. Вы можете даже добавить пакеты прямо со Slackware CD-ROM.

Смотрите Ref Управление пакетами для дополнительной информации.

5.10.6 Основные проблемы

Здесь приведены наиболее часто встречающиеся проблемы, при использовании ZipSlack или BigSlack. Так же мы предлагаем несколько других способов получения помощи, если проблема с которой вы столкнулись не описана здесь. В следующем разделе описаны разные способы получения поддержки.

Не получается открыть начальную консоль

Это обычно случается при указании неверного раздела диска для root= параметра в `LINUX.BAT` файле. Измените `LINUX.BAT` опять, выбрав другой раздел в качестве root=. Если вы не имеете понятия о том, что это значит, вы можете просто заняться перебором, пока не доберётесь до работающего.

Так же это может быть вызвано распаковкой `.ZIP` файлов в каталоге, отличном от корневого. Архив должен быть распакован в корневом каталоге диска, а не в каком-то из под-каталогов.

Kernel panic: VFS : Unable to mount root fs

(Это означает, что вы указали неверное устройство для root= параметра в `LINUX.BAT` файле. Вам необходимо изменить файл опять и выбрать другое root= устройство. Если вы не знаете, какое выбрать, займитесь перебором, пока не найдёте работающее значение.

5.10.7 Получение помощи

Перед тем как обращаться за помощью, вам следует прочесть документацию, в `C:\LINUX` каталоге, вполне возможно ответ на ваш вопрос уже ждёт вас там.

Если вы прочли документацию и всё ещё не решили вашу проблему, один из приведённых ниже методов наверняка поможет вам.

ZipSlack FAQ

Список наиболее часто задаваемых вопросов и ответов на них, может быть найден как в `FAQ.TXT` файле в `C:\LINUX` каталоге, так и на нашем вэб сайте:

http://www.slackware.com/faq/

Форум обсуждения ZipSlack

Доступный в он-лайн форум для ZipSlack позволяет вам общаться с другими пользователями ZipSlack и BigSlack. Вы можете опубликовать свой

вопрос тут, а так же можете помочь другим пользователям. Это очень хороший способ получения прямой помощи от других пользователей.

http://www.slackware.com/forum/

Поддержка через электронную почту

Команда поддержки Slackware попытается помочь вам, если у вас возникли проблемы с ZipSlack или BigSlack. Постарайтесь как можно понятнее сформулировать проблему и предоставить любую информацию, которая может быть полезна при решении проблемы.

<support@slackware.com>

5.10.8 Итог

Вы должны понимать, что такое ZipSlack and BigSlack. Если вы решили воспользоваться одним из них, вы должны обладать достаточной информацией для установки, загрузки и получения поддержки. ZipSlack и BigSlack Могут быть очень полезны если вы просто хотите попробовать, что такое Slackware, без удаления существующих Windows разделов.