Книга по Linux

(оригинал вы можете найти на http://www.opennet.ru/)

Оглавление

1. Введение в Linux.

Linux, возможно, является наиболее значительным достижением в области свободно распространяемых программ со времен Space War, или более позднего Emacs. Он превратился в операционную систему для бизнеса, образования и индивидуального программирования. Linux перестал быть системой для фанатиков-программистов, которые часами сидят перед мерцающими экранами (хотя таких и немало). Эта книга поможет вам извлечь из Linux максимальную пользу.

Linux (произносится "лИнукс") принадлежит семейству UNIX-подобных операционных систем, которая может работать на компьютерах Intel 80386 и 80486. Он поддерживает широкий спектр программных пакетов от TeX до X Windows, компиляторов GNU C/C++, протоколов TCP/IP. Это гибкая реализация ОС UNIX, свободно распространяемая по лицензии GNU (см. приложение C).

Linux может любой 386 или 486 персональный компьютер превратить в рабочую станцию. Он преподнесет всю мощь UNIX к кончикам ваших пальцев. Бизнесмены инсталлируют Linux в сетях машин, используют операционную систему для обработки данных в сфере финансов, медицины, распределенной обработки, в телекоммуникациях и т.д. Университеты по всему миру применяют Linux в учебных курсах по программированию и проектированию операционных систем. Разумеется, повсеместно программисты-энтузиасты используют Linux дома для программирования, решения своих прикладных задач и всевозможного хакерства.

Что делает Linux столь отличным от других ОС, так это создание версии UNIX "*на общественных началах*" (*free implementation*). Он был создан и продолжает совершенствоваться и развиваться группой добровольцев, первоначально в кругу пользователей сети Internet, которые обменивались кодами, информацией об обнаруженных ошибках, выявлением проблем, возникавших при расширении сферы применения. Все желающие приглашаются подключиться к этой работе. Единственное, что требуется - это интерес к семейству UNIX и желание совершенствовать свои навыки в этой сфере. Данная книга ваш путеводитель в системе.

1.1 Об этой книге.

Эта книга является руководством по инсталляции и пособием по начальному знакомству с системой Linux. Цель: приобщить новых пользователей к этой системе и собрать возможно больше существенного материала по ее использованию в одной книге. Вместо того, чтобы перегружать книгу техническими деталями, которые быстро устаревают, мы даем основы, которые помогут вам самостоятельно находить и осваивать дополнительную информацию.

Linux прост в инсталляции и использовании. Но, как и во всякой реализации UNIX, в нем присутствуют элементы "черной магии", которые необходимы при обеспечении его корректной работы. Мы надеемся, что эта книга будет вам хорошим путеводителем по Linux и покажет, насколько простой может выглядеть эта операционная система.

В этой книге мы рассматриваем следующие вопросы:

- Что такое Linux? Особенности структуры и философии этой уникальной операционной системы, и что она может вам дать.
- Все детали, необходимые для практического использования Linux, включая рекомендации по желательной конфигурации аппаратуры.
- Как получить и инсталлировать Linux. Существует много способов распространения программного обеспечения под Linux. Мы описываем общую ситуацию, связанную с его распространением, рассказываем, как его приобрести и инсталлировать. Это издание содержит также специфические инструкции по дистрибутивам Linux Debian, Red Hat и Slackware.
- Краткое учебное пособие по UNIX для тех пользователей, которые до этого не встречались с ОС UNIX. Надеемся, что это пособие дает достаточно материала для новичков, чтобы получить базовые знания и начать ориентироваться в этой ОС.
- Введение в системное администрирование Linux. Это покрывает наиболее важные задачи, с которыми следует познакомиться новым администраторам Linux, с такими задачами как регистрация новых пользователей, управление файловой системой и тому подобное.
- Информация о конфигурировании более продвинутых аспектов Linux, таких как X Window System, сетевая работа с TCP/IP и SLIP, и установке электронной почты.

Эта книга для пользователей персональных компьютеров, желающих начать работать с Linux. Мы не предполагаем предварительного опыта работы с UNIX, но надеемся, что новички будут обращаться по ходу дела к дополнительной литературе. Для незнакомых с UNIX в Приложении А приведен список полезных источников. В общем случае предполагается чтение этой книги совместно с какой-либо книгой по общим концепциям ОС UNIX.

1.2 Краткая история Linux.

UNIX одна из самых популярных в мире операционных систем благодаря тому, что ее сопровождает и распространяет большое число компаний. Первоначально она была создана как многозадачная система для миникомпьютеров и мэйнфреймов в середине 70-ых годов, но с тех пор она выросла в одну из наиболее распространенных операционных систем, несмотря на свой временами обескураживающий интерфейс и отсутствие централизованной стандартизации.

В чем реальная причина популярности UNIX? Многие хакеры нутром чувствуют, что UNIX это "настоящая вещь", Единственная Настоящая Операционная Система. Отсюда и появление Linux, как системы, разрабатываемой все более расширяющейся группой энтузиастов UNIX, которые хотят собственноручно в ней поковыряться.

Существуют версии UNIX для многих систем, начиная от персонального компьютера, до суперкомпьютеров, таких как Cray Y-MP. Фактически все серьезные системы работают именно под UNIX. Большинство версий UNIX для персональных компьютеров достаточно дороги и сложны. К моменту написания этой книги одномашинная версия AT&T's System V стоила US$1500.

Linux свободно распространяемая версия UNIX, первоначально была разработана Линусом Торвальдсом (Linus Torvalds) (torvalds@kruuna.helsinki.fi) в Университете Хельсинки (Финляндия). Linux был создан с помощью многих UNIX-программистов и энтузиастов из Internet, тех, кто имеет достаточно навыков и способностей развивать систему. Ядро Linux не использует коды AT&T или какого-либо другого частного источника, и большинство программ Linux разработаны в рамках проекта GNU из Free Software Foundation в Cambridge, Massachusetts, U.S.A. Но в него внесли лепту также программисты всего мира.

Первоначально Linux создавался Линусом Торвальдсом как хобби. Его вдохновила операционная система Minix маленькая UNIX-система, созданная Andy Tanenbaum, и впервые Linux обсуждался по компьютерной сети в рамках USENET newsgroup `comp.os.minix`. В этих обсуждениях прежде всего принимали участие пользователи Minix из учебных и научных заведений, которым хотелось чего-то большего, чем Minix.

Раннее развитие Linux прежде всего было связано с проблемой переключения задач в защищенном режиме для 80386. Все писалось на ассемблере. Линус пишет:

> ``После этого началось спокойное плавание: по-прежнему беспросветное кодирование, но у меня были различные подсобные программы и отладка была облегчена. На этом этапе я стал использовать Си и это существенно ускорило дело. В это же время я стал серьезно обдумывать маниакальную идею, как сделать Minix лучше себя самого. Я надеялся в один прекрасный день перекомпилировать gcc под Linux...``
>
> ``Два месяца ушло на написание самых базовых программ, а затем чуть больше времени на драйвер винчестера (с большим количеством ошибок, но все-таки работавшим на моей машине) и простую файловую систему. В результате я подготовил версию 0.01 (примерно конец августа 1991 г.). Она была не слишком изящной, в ней не было драйвера гибких дисков и она многое не могла делать. Но я уже не смог остановиться, пока не создал свой Minix.``

Относительно появления Linux версии 0.01 никогда не делалось никаких официальных заявлений. Исходные тексты 0.01 не давали даже нормального выполняемого кода: они фактически состояли лишь из набора заготовок для ядра и молчаливо предполагали, что вы имеете доступ к Minix-машине, чтобы иметь возможность компилировать их и совершенствовать.

5-го октября 1991 года Линус объявил первую "официальную" версию Linux, версия 0.02. В это время Linux уже мог выполнять `bash` (the GNU Bourne Again Shell) и `gcc` (the GNU C compiler), но мало еще что работало. Вновь это рассматривалось как создание некой хакерской системы. Основное внимание было обращено на создание ядра. Никакие вопросы поддержки работы с пользователем, документирования, тиражирования и т.п. даже не обсуждались. Кажется, что и сегодня сообщество Linuxистов считает эти вопросы вторичными по сравнению с "настоящим программированием" развитием ядра. Но именно с того момента мир сильно изменился.

Линус писал в `comp.os.minix`:

> ``Грустите ли вы по тем прекрасным временам Minix-1.1, когда мужчины были настоящими мужчинами и писали свои собственные драйверы на все устройства? У вас сейчас нет под рукой настоящего проекта и вы вымираете от невозможности вонзить свои зубы в какую-то ОС, которую бы можно было модифицировать под свои желания? Не находите ли вы деморализующей ситуацию, когда все в Minix работает? Нет больше бессонных ночей, которые позволяли заставить хитрые программы работать правильно? Тогда это место для вас.``
>
> ``Как я уже говорил месяц назад, сейчас я работаю над некоммерческой Minix-подобной ОС для 386-го компьютера. Она уже доведена до такого состояния, когда ею даже можно пользоваться (хотя может быть там не то, что бы вы хотели), и я хочу выложить исходные тексты для широкого распространения. Это версия 0.02, но в ней уже успешно работают `bash`, `gcc`, `gnu-make`, `gnu-sed`, `compress` и т.д."

После версии 0.03 Линус скачком перешел в нумерации к версии 0.10, так как над проектом стало работать много народу. После нескольких последовавших пересмотров версий, Линус присвоил очередной версии номер 0.95, чтобы тем самым отразить свое впечатление о том, что скоро возможна уже ``официальная`` версия. (Обычно программам не дают номер версии 1.0 до того, как она теоретически завершена и отлажена). Это было в марте 1992 г. Примерно через полтора года, в декабре 1993 версия ядра все еще была Linux 0.99.pl14, асимптотически приближаясь к 1.0. Во время написания книги текущая версия ядра 2.2 patchlevel 14, и версия для разработчиков 2.3 patchlevel 6.

Сегодня Linux это полноценная ОС семейства UNIX, способная работать с X Windows, TCP/IP, Emacs, UUCP, mail и USENET. Практически все важнейшие программные пакеты были поставлены и на Linux, т.е. для Linux теперь доступны и коммерческие пакеты. Все большее разнообразие оборудования поддерживается по сравнению с первоначальным ядром. Многие тестировали Linux на 486-ом и установили, что он вполне сравним с рабочими станциями Sun Microsystems и Digital Equipment Corporation. Кто мог предположить, что этот "маленький UNIX" вырастет настолько, что сможет делать все в мире компьютеров...

1.3 Системные характеристики.

Linux очень простая система: все, что в Windows называется "фича", в Linux называется "глюк".

Linux поддерживает большинство свойств, присущих другим реализациям UNIX, плюс ряд тех, которых больше нигде нет. Этот раздел дает поверхностный обзор характеристик ядра Linux.

Linux это полная многозадачная многопользовательская операционная система (точно также как и другие версии UNIX). Это означает, что одновременно много пользователей могут работать на одной машине, одновременно выполнять много программ.

Linux достаточно хорошо совместим с рядом стандартов для UNIX (насколько вообще можно говорить о стандартизации UNIX) на уровне исходных текстов, включая IEEE POSIX.1, System V и BSD. Он создавался имея в виду такую совместимость. Поэтому, скорее всего, вы найдете в Linux черты, присущие многим UNIX-системам. Большинство свободно распространяемых по сети Internet программ для UNIX может быть откомпилировано для LINUX практически без особых изменений. Кроме того, все исходные тексты для Linux, включая ядро, драйверы устройств, библиотеки, пользовательские программы и инструментальные средства распространяются свободно.

Другие специфические внутренние черты Linux включают контроль задач по стандарту POSIX (используемый оболочками, такими как `csh` и `bash`), псевдотерминалы (`pty`), поддержку национальных и стандартных клавиатур динамически загружаемыми драйверами клавиатур. Linux поддерживает **виртуальные консоли**, которые позволяют переключаться между различными сеансами работы на разных (или одной) системах с одной консоли, то есть позволяют "переключать экраны" на консоли в текстовом режиме. Те, кто пользовался программой "`screen`", найдут подобное в реализации виртуальных консолей в Linux.

Ядро может само эмулировать команды 387-FPU, так что системы без сопроцессора могут выполнять программы, на него рассчитывающие (т.е. с плавающей точкой).

Linux поддерживает различные типы файловых систем для хранения данных. Некоторые файловые системы, такие как файловая система *ext2fs*, были созданы специально для Linux. Поддерживаются также другие типы файловых систем, такие как Minix-1 и Xenix. Реализована также файловая система MS-DOS, позволяющая прямо обращаться к файлам MS-DOS на жестком диске. Поддерживается также файловая система ISO 9660 CD-ROM для работы с дисками CD-ROM. Подробнее о файловых системах говорится в Главах 2 и 4.

Linux обеспечивает полный набор протоколов TCP/IP для сетевой работы. Это включает драйверы устройств для многих популярных карт Ethernet, SLIP (Serial Line Internet Protocol, обеспечивающие вам доступ по TCP/IP при последовательном соединении), PLIP (Parallel Line Internet Protocol), PPP (Point-to-Point Protocol), NFS (Network File System), и так далее. Поддерживается весь спектр клиентов и услуг TCP/IP, таких как FTP, `telnet`, NNTP и SMTP. О сетевых проблемах мы будем говорить в Главе 5.

Ядро Linux сразу создано с учетом специального защищенного режима для процессоров Intel 80386 и 80486. В частности, Linux использует парадигму описания памяти в защищенном режиме и другие новые свойства процессоров. Любой знакомый с защищенным режимом процессора 80386 знает, что этот чип проектировался для многозадачных систем вроде UNIX (или Mulics). Linux использует эти свойства.

Ядро Linux поддерживает загрузку только нужных страниц. То есть с диска в память загружаются те сегменты программы, которые действительно используются. Возможно использование одной страницы, физически один раз загруженной в память, несколькими выполняемыми программами.

Для увеличения объема доступной памяти Linux осуществляет также разбиение диска на страницы: то есть на диске может быть выделено до 1 гигабайта "**пространства для**

свопинга" (**swap space**). (Swap space не совсем подходящее имя, в Linux в область свопинга выгружается не весь процесс, а только отдельные его части, в которых нет необходимости). Когда системе нужно больше физической памяти, то она с помощью свопинга выводит неактивные страницы на диск. Это позволяет выполнять более объемные программы и обслуживать одновременно больше пользователей. Однако свопинг не исключает наращивания физической памяти, поскольку он снижает быстродействие, увеличивает время доступа.

Ядро также поддерживает универсальный пул памяти для пользовательских программ и дискового кэша. При этом для кэша может использоваться вся память, и наоборот, кэш уменьшается при работе больших программ.

Выполняемые программы используют динамически связываемые библиотеки, т.е. выполняемые программы могут совместно использовать библиотечную программу, представленную одним физическим файлом на диске (иначе, чем это реализовано в механизме разделяемых библиотек SunOS). Это позволяет выполняемым файлам занимать меньше места на диске, особенно тем, которые многократно используют библиотечные функции. Есть также статические связываемые библиотеки для тех, кто желает пользоваться отладкой на уровне объектных кодов или иметь "полные" выполняемые программы, которые не нуждаются в разделяемых библиотеках. В Linux разделяемые библиотеки динамически связываются во время выполнения, позволяя программисту заменять библиотечные модули своими собственными.

Для обеспечения отладки ядро Linux выдает дампы памяти для "посмертного" анализа. Использование дампа и динамических отладчиков позволяет определить причины краха программы (и системы...).

1.4 Программные характеристики.

В этом разделе мы представим вам многие приложения, доступные в Linux, и поговорим об общих задачах вычисления. В конечном счете наиболее важным в системе является то, насколько широк спектр доступных в ней программ. А тот факт, что большая часть этих программ распространяется свободно, усиливает впечатление.

Практически любая утилита, которую вы ожидаете найти в стандартных реализациях UNIX, имеется и в Linux. Сюда включены и базовые команды, такие как `ls`, `awk`, `tr`, `sed`, `bc`, `more` и т.д. Назовите любую: она уже есть в Linux. Поэтому вы в праве ожидать знакомой рабочей UNIX-среды. В Linux есть все стандартные команды и утилиты. (Новички могут посмотреть Главу 3 для начального знакомства с базовыми командами UNIX).

В Linux имеются многие текстовые редакторы, включая `vi`, `ex`, `pico`, `jove`, также как GNU Emacs и его вариации, вроде Lucid Emacs (который содержит расширение для использования под X Windows) и `joe`. Скорее всего, любой текстовый редактор, к которому вы привыкли, перенесен в Linux.

Выбор редактора явление любопытное. Многие пользователи UNIX до сих пор используют "простые" редакторы вроде `vi` (кстати, автор писал эту книгу в Linux, используя редактор `vi`). Но `vi` имеет много ограничений по причине своего преклонного возраста. Сейчас завоевывают популярность более современные и

сложные редакторы, вроде `Emacs`. `Emacs` поддерживает базирующийся на LISP макроязык и интерпретатор, мощный командный синтаксис и другие расширения. Существуют макропакеты `Emacs`, позволяющие читать электронную почту и новости, редактировать содержимое каталогов и даже проводить сеансы психотерапии с использованием искусственного интеллекта (неоценимая возможность для измотанных Linux хакеров).

Интересное замечание: большинство утилит Linux имеют статус GNU. Эти утилиты часто поддерживают наиболее современные черты, не содержащиеся в стандартных версиях BSD или AT&T. Например, версия GNU редактора `vi`, `elvis`, содержит структурный макроязык, который отличается от исходной реализации AT&T. Но тем не менее, утилиты GNU сохраняют совместимость с их тезками из BSD и System V. Многие считают, что GNU версии лучше исходных программ.

Многие пользователи самой важной утилитой считают **shell**. **shell** это программа, которая читает и выполняет команды пользователя. Кроме того, многие оболочки (shells) имеют такие возможности, как **контроль выполнения**> (**job control**) (позволяя пользователю управлять несколькими параллельными процессами), перенаправление входа-выхода и командный язык для написания **командных файлов** (**shell scripts**). Командный файл это программа на языке оболочки, аналогичная "batch file" в MS-DOS, но язык имеет гораздо больше возможностей. Мне приходилось видеть оболочку, в которой был встроен язык C!

В Linux много типов оболочек. Наиболее важное различие между ними используемый командный язык. Например, **C Shell** (`csh`) использует командный язык, чем-то напоминающий язык программирования Си. Классический <**Борновский shell** (**Bourne Shell**) использует иной командный язык. Обычно выбор оболочки обусловлен выбором соответствующего командного языка. Выбранная оболочка в какой-то мере определяет вашу рабочую среду.

Не важно, к какой оболочке вы привыкли, та или иная ее версия есть в Linux. Наиболее популярная оболочка GNU Bourne Again Shell (`bash`), т.е. вариант Bourne shell, включающий много современных свойств и возможностей, таких как управление работами, командную историю, дописывание имен команд и имен файлов, `Emacs`-подобный интерфейс редактирования командной строки и мощное расширение стандартной оболочки (Bourne shell).

Другая популярная оболочка `tcsh`, версия C Shell с более современными функциями по сравнению с `bash`. Другие оболочки: `zsh` небольшая борноподобная оболочка; `ksh` оболочка Корна; `ash` оболочка BSD и `rc` оболочка проекта Plan 9.

Что особенно важно сказать относительно этих оболочек? Linux дает вам уникальную возможность кроить систему под ваши личные нужды. Например, если вы единственный пользуетесь этой системой и вы предпочитаете редактор `vi` и `bash` в качестве оболочки, то нет необходимости иметь прочие редакторы и оболочки. "Сделай сам, как тебе нравится" это позиция хакеров и пользователей Linux.

1.4.1 Обработка текстов и слов.

Почти все пользователи нуждаются в какой-либо системе подготовки документов. (Много ли вы знаете энтузиастов компьютерной обработки, которые все еще пользуются ручкой и бумагой? Мы догадываемся, что очень немного). В мире персональных компьютеров *обработка слов* (*word processing*) норма: она включает редактирование и манипуляции с текстом (часто в стиле WYSIWYG "What-You-See-Is-What-You-Get": "Что-Вы-Видите-То-Вы-Получаете", правда иногда WYSIWYG произносят как WYSIFYG) и получение печатных копий, содержащих рисунки, таблицы и другой гарнир.

Коммерческие редакторы от Corel, Applix и Star Division доступны в мире UNIX. Но в UNIX *обработка текстов* (*text processing*) вещь более общая, чем классическая концепция обработки слов. Вместо того, чтобы вызывать специальные средства обработки слов, исходный текст можно модифицировать любым текстовым редактором, таким как `vi` или `Emacs`. После подготовки текста пользователь форматирует текст специальными программами, преобразующими его к нужному для печати виду. Это в чем-то аналогично программированию на языке вроде Си, и "компилированию" текста в пригодную для печати форму.

В Linux много текстовых процессоров. Один из них `groff` GNU версия классического форматтера текстов `nroff`, первоначально созданного в Bell Labs и до сих пор используемого во многих UNIX. Другой современный текстовый процессор TeX, создан знаменитым в компьютерном мире Дональдом Кнутом (Donald Knuth). Доступны диалекты TeX, такие как LaTeX.

Текстовые процессоры, такие как TeX и `groff` значительно различаются по синтаксису языков форматирования. Предпочтение той или иной системы форматирования в значительной мере базируется на том, какие имеются вспомогательные утилиты и насколько система вам по вкусу.

Например, некоторые люди считают `groff` несколько заумным, поэтому они используют TeX, который более понятен для хомо сапиенс. Между тем, `groff` может давать ясный ASCII выход, легко читаемый на терминале, в то время как TeX прежде всего предназначен для вывода на печать. Но существуют многочисленные программы, позволяющие получить читаемый текст из отформатированных с помощью TeX документов или конвертирующих TeX, например, в `groff`.

Другой текстовый процессор `texinfo`, расширение TeX, используемое для подготовки программной документации в Free Software Foundation. `texinfo` может формировать печатный документ или гипертекст "Info" для просмотра на экране на основе одного исходного файла. Файлы Info это основной формат для документирования, используемый в GNU, в частности в Emacs.

Текстовые процессоры широко используются в компьютерном мире для подготовки статей, диссертаций, статей для журналов и книг (типографский вариант этой книги был подготовлен с использованием LaTeX). Возможность обрабатывать исходный язык как текстовый файл открывает двери.

Как выглядит язык форматирования? В общем случае он содержит сам текст с "управляющими кодами" для производства заказанных действий, таких как изменение фонтов, установление полей, формирование страниц и т.д.

В качестве примера возьмем следующий текст:

> Mr. Torvalds:
>
> We are very upset with your current plans to implement *post-hypnotic suggestions* in the **Linux** terminal driver code. We feel this way for three reasons:
>
> 1. Planting subliminal messages in the terminal driver is not only immoral, it is a waste of time;
> 2. It has been proven that ``post-hypnotic suggestions'' are ineffective when used upon unsuspecting UNIX hackers;
> 3. We have already implemented high-voltage electric shocks, as a security measure, in the code for `login`.
>
> We hope you will reconsider.

Этот текст на языке форматирования LaTeX будет выглядеть следующим образом:

```
\begin{quote}
Mr. Torvalds:

We are very upset with your current plans to implement
{\em post-hypnotic suggestions\/} in the {\bf Linux} terminal
driver code. We feel this way for three reasons:
\begin{enumerate}
\item Planting subliminal messages in the kernel driver is not only
 immoral, it is a waste of time;
\item It has been proven that ``post-hypnotic suggestions''
 are ineffective when used upon unsuspecting UNIX hackers;
\item We have already implemented high-voltage electric shocks, as
 a security measure, in the code for {\tt login}.
\end{enumerate}
We hope you will reconsider.
\end{quote}
```

Автор входит в "исходный" текст, приведенный выше, используя любой текстовый редактор, и генерирует форматированный выход, обрабатывая текст с помощью LaTeX. На первый взгляд такой язык может показаться достаточно заумным, но в действительности он очень прост в освоении. Использование текстового процессора обеспечивает поддержку типографских стандартов. Например, все перечисленные в рамках документа страницы будут выглядеть одинаково. Пишущий может сосредоточиться на тексте, а не на том, как это следует оформлять с точки зрения типографских требований.

Процессоры слов типа WYSIWYG привлекательны по многим причинам: они обеспечивают мощный (а иногда и сложный) визуальный интерфейс для редактирования документов. Но этот интерфейс традиционно ограничен желательными и приемлемыми для пользователя формами выдачи. Так, многие процессоры имеют средства подготовки к печати математических формул.

Преимущество текстовых процессоров состоит в том, что они позволяют описывать именно то, что вы хотите. Они позволяют также редактировать исходный текст любым текстовым редактором и затем конвертировать в различные форматы. Платой за такую гибкость является отсутствие тех качеств интерфейса, которые есть у WYSIWYG.

Есть программы, которые позволяют посмотреть на графическом дисплее вид отформатированного документа перед его печатью. Например, программа `xdvi` отображает независимый от устройства файл, сгенерированный TeX под X Window. Приложения `xfig` и `gimp` обеспечивает графический интерфейс WYSIWYG для рисунков и диаграмм, который в последующем конвертируется в команды текстового процессора, включаемые в документ.

Следует напомнить, что текстовые процессоры, вроде `nroff`, появились задолго до процессоров слов. Многие предпочитают использовать текстовые процессоры из-за их гибкости и независимости от графической среды. В любом случае, в Linux имеется процессор слов idoc, а также скоро ожидается появление коммерческих процессоров слов. Но если вы не хотите расставаться с процессором слов, к которому вы привыкли в MS-DOS, вы всегда можете использовать MS-DOS или другую операционную систему в дополнение к Linux.

Существует и много других утилит текстовых процессоров. Мощная система METAFONT используется для проектирования шрифтов в TeXe. Другие программы включают `ispell` интерактивный контролер правописания, `makeindex` используется для генерации индексов в документах, подготовленных в LaTeX. Существует много макропакетов для `groff` и TeX для форматирования документов различных типов и математических текстов, а также тьма конверторов, транслирующих TeX или `groff` во многие другие форматы.

Недавно появился новый форматтер документов YODL, написанный Karel Kubat. Язык YODL очень прост в изучении и имеет фильтры для конвертации результата в разные форматы, например, LaTeX, SGML и HTML.

1.4.2 Языки программирования и утилиты.

Linux обеспечивает полную UNIX-среду программирования, включая все стандартные библиотеки, программный инструментарий, компиляторы, отладчики, которые вы встречаете и в других UNIX-системах. В мире UNIX большинство приложений и системных программ делаются на Си или Си++. Стандартным компилятором для Си и Си++ в Linux служит GNU `gcc`, который является современным компилятором, поддерживающим много опций. Он способен компилировать Си++ (включая особенности AT&T 3.0 features) также, как Objective-C, другие объектно-ориентированные диалекты Си.

Стандарты подобные POSIX.1 поддержаны, что позволяет легко перенести программное обеспечение, написанное для Linux на другие системы. Профессиональные UNIX программисты и администраторы системы используют Linux, чтобы разработать программное обеспечение дома, затем передают программное обеспечение UNIX системам на работе. Это не только сохраняет много времени и денег, но также и позволяет Вам работать в комфорте вашего собственного дома. Один из авторов использует систему, чтобы разрабатывать и проверять дома прикладные

программы для X Window System, которые могут непосредственно компилироваться на автоматизированных рабочих местах в другом месте. Студенты, изучающие информатику, узнают программирование в UNIX и исследуют другие аспекты системы, например, архитектуре ядра.

С Linux, Вы имеете доступ к огромному набору библиотек и утилит для программирования и полному исходному тексту ядра и библиотек.

В программном мире UNIX системы и прикладные программы часто программируются в C или C++. Стандартный транслятор C и C++ для Linux GNU `gcc`, который является продвинутым, современным транслятором, который поддерживает C++, включая AT&T 3.0 возможности, а также Objective-C, другой объектно-ориентированный диалект C.

Кроме Си и Си++ многие другие компиляторы и интерпретаторы были перенесены в Linux, такие как Smalltalk, FORTRAN, Pascal, LISP, Scheme и Ada. В дополнение, существуют различные ассемблеры для написания кодов для защищенного режима 80386, а также любимые хакерами языки, вроде Perl и Tcl/Tk (shell-подобный командный язык, включающий поддержку разработки простейших приложений в X Window).

В Linux был перенесен (про)двинутый отладчик `gdb`, позволяющий пошагово выполнять программы в поисках ошибок или анализировать крах программ с помощью дампов памяти. `gprof`, утилита профилирования, показывающая, где ваша программа при выполнении тратит больше времени. Текстовый редактор `Emacs` позволяет осуществлять интерактивное редактирование. Другие инструменты, включая GNU `make` и `imake` используются для управления компиляцией больших программ; RCS: система для защиты и сопровождения исходных текстов.

Linux содержит динамические библиотеки (DLL), которые позволяют экономить место, поскольку они вызываются только во время выполнения. Эти библиотеки позволяют также прикладному программисту переопределять функции, включая свои коды. Например, если программист желает написать свою собственную версию библиотечной программы `malloc()`, компоновщик подключит новую программу вместо библиотечной.

1.4.3 Введение в X Window.

Система X Window (или кратко просто X) стандартный графический интерфейс для UNIX-машин. Это мощная среда, поддерживающая много приложений. Используя X Window, пользователь может одновременно иметь на экране несколько окон, при этом каждое имеет независимый login. Часто используется мышь, хотя она необязательна.

Было написано много специфических X-приложений, таких как игры, графические утилиты, инструментарий для программирования и документирования и т.д. С Linux и X ваш компьютер - замечательная рабочая станция. Используя протоколы TCP/IP, вы можете смотреть у себя X-приложения, выполняемые на других машинах.

Система X Window была первоначально создана в MIT и свободно распространялась. Существует много и коммерческих приложений, расширяющих возможности X

Window. Для Linux есть система X Window, известная как XFree86; версия X11R6 свободно распространяется для UNIX-систем типа Linux. XFree86 поддерживает широкий спектр видеоустройств, включая VGA, SuperVGA, различные видеоадаптеры с ускорителями. Это полный комплект X Window, содержащий сам сервер, много прикладных программ и утилит, программные библиотеки и документацию.

Стандартные X-приложения включают `xterm` (эмулятор терминала, используемый в большинстве текстовых приложений в X Window); `xdm` (X-менеджер, обслуживающий login); `xclock` (представление простых часов); `xman` (X-ориентированное руководство по Linux) и т.д. Трудно перечислить все приложения X, доступные в Linux, но базовый комплект XFree86 включает "стандартные" приложения, содержащиеся в исходной версии MIT. Но доступно и многое другое, теоретически, все написанное для X Window должно прямо компилироваться и для Linux. Доступно множество графических программ: текстовые редакторы, таблицы, графические рисовалки, web-браузеры, типа Netscape Navigator, игры.

Интерфейс X Window в большой степени контролируется **менеджером окон** (**window manager**). Эта программа отвечает за размещение окон, изменение их размеров, размещение иконок, перемещение окон, вид оконных рамок и т.д. Стандартный дистрибутив XFree86 включает twm, классический оконный менеджер MIT, но также имеются и более современные менеджеры, такие как Open Look Virtual Window Manager (`olvwm`). Среди пользователей Linux популярен `fvwm`. Это небольшой менеджер окон, требующий в два с лишним раза меньше памяти, чем `twm`. Он обеспечивает трехмерное представление обрамления окон и виртуальный рабочий стол (desktop) - если пользователь подвигает мышь к краю экрана, все изображение смещается, будто дисплей имеет большие размеры, чем на самом деле. fvwm более традиционен и позволяет реализовать все функции доступа как с клавиатуры, так и от мыши. Многие дистрибутивы Linux содержат `fvwm`, как стандартный менеджер окон. Есть версия `fvwm`, называемая `fvwm95-2`, которая имитирует Microsoft Windows 95.

Дистрибутив XFree86 содержит программные библиотеки и включает файлы для тех программистов, кто желает создавать приложения в X. Поддерживаются различные множества widget (графических представлений), такие как Athena, Open Look и Xaw3D. Включены все стандартные фонты, битмэпы и документация. Поддерживается также PEX (программный интерфейс для трехмерной графики).

Многие пользующиеся X используют и имеющиеся в Motif наборы widget. Несколько компаний продают одно и многопользовательские лицензии бинарников Motif в Linux. Поскольку Motif сам по себе сравнительно дорог, немногие владельцы Linux имеют Motif. Тем не менее, бинарники, статически связанные с библиотечными программами Motif, могут свободно распространяться. Если вы написали программы с использованием Motif и хотите их передавать, вы должны позаботиться о самодостаточности кодов.

Главные ограничения использования X Window происходят от требований к аппаратуре. Минимально необходим 386 процессор с 4 Мбайт RAM. Но для более комфортного режима желательно не менее 8 Мбайт. Желательно и процессор побыстрее, но прежде всего необходима память. Для действительно хорошего результата лучше иметь карту с акселератором (как например S3-chipset). На Linux с XFree86 был достигнут рейтинг выполнения, превосходящий 300000 xstones. На

приличном компьютере вы можете убедиться, что X под Linux работает не хуже, или даже быстрее, чем на других UNIX.

В главе 5 мы обсудим вопросы инсталляции и использования X в Вашей системе.

1.4.4 Работа в сети.

Интересует ли вас связь с миром? Linux поддерживает два базовых сетевых протокола UNIX: **TCP/IP** и **UUCP**. TCP/IP (Transmission Control Protocol/Internet Protocol) есть множество сетевых парадигм, позволяющих системам по всему миру связываться по единой сети, известной как Internet. С помощью Linux, TCP/IP и подключения к сети вы можете общаться с пользователями и машинами всего Internet через электронную почту, новости USENET, передачу файлов FTP и т.п. В Internet много машин под Linux.

Большинство сетей TCP/IP используют Ethernet, как физическое транспортное средство. Linux поддерживает многие популярные карты Ethernet и интерфейсы, в том числе PCMCIA Ethernet адаптеры.

Однако, поскольку не у всех есть дома плата Ethernet, Linux также поддерживает **SLIP** (Serial Line Internet Protocol), позволяющий связываться с Internet через модем. Для использования SLIP вы должны иметь доступ к SLIP-серверу, машине связанной с сетью и обеспечивающей вам вход в Internet. Многие фирмы и университеты предоставляют SLIP-сервис. Если ваш Linux имеет Ethernet и модем, вы можете сконфигурировать систему как SLIP-сервер для других хостов.

NFS (Network File System) позволяет вам использовать файлы совместно с другими машинами сети. FTP (File Transfer Protocol) позволяет передавать файлы между машинами. Другие приложения включают `sendmail` систему передачи и получения электронной почты с использованием протокола SMTP; базирующуюся на протоколе NNTP, системе электронных новостей типа C-News и INN; `telnet`, `rlogin` и `rsh` позволяют войти и выполнить команды на других машинах сети; `finger` позволяет получать информацию о других пользователях Internet.

Linux также поддерживает Сетевое Окружение Microsoft Windows через пакет Samba и сети Macintosh AppleTalk и LocalTalk. Также есть поддержка протокола Novell IPX.

В Linux существует полный спектр различных программ для чтения почты и новостей, это, например, `elm`, `pine`, `rn`, `nn` и `tin`.

Если у вас есть опыт работы с приложениями TCP/IP на других UNIX-системах, Linux не будет для вас чем-то новым. Система обеспечивает стандартный программный интерфейс, поэтому любая программа, использующая TCP/IP, может быть легко перенесена на Linux. X-сервер Linux также поддерживает TCP/IP, позволяя отображать выполняемые на других машинах прикладные программы на вашем дисплее.

Ниже мы обсудим конфигурацию и установку TCP/IP для Linux, включая SLIP и PPP.

UUCP (UNIX-to-UNIX Copy) старейший механизм передачи файлов, электронной почты и электронных новостей между UNIX-машинами. Классически, UUCP-машины связываются друг с другом по телефонным линиям через модем, но UUCP может

использовать в качестве транспортного средства и связь по TCP/IP. Если у вас нет доступа по TCP/IP или SLIP-сервера, вы можете сконфигурировать свою систему так, чтобы посылать и получать файлы и электронную почту с использованием UUCP. Подробнее смотрите в главе 5.

1.4.5 Телекоммуникации и BBS.

Если у вас есть модем, вы можете связываться с другими машинами, используя телекоммуникационные пакеты, имеющиеся в Linux. Многие используют программы телекоммуникации для связи с BBS (Bulletin Board Systems), а также и с коммерческими он-лайновыми системами, вроде Prodigy, CompuServe и America On-Line. Другие через модемы связываются с UNIX-системой в школе или на работе. Вы можете использовать модем и Linux для посылки и приема факсов. Телекоммуникационные пакеты Linux очень похожи на имеющиеся в MS-DOS или других операционных системах.

Один из наиболее популярных телекоммуникационных пакетов в Linux (Seyon) X-приложение, предоставляющее традиционный эргономичный интерфейс со встроенной поддержкой различных протоколов передачи файлов, таких как Kermit, ZModem и т.п. Есть также телекоммуникационные программы C-Kermit, `pcomm` и `minicom`. Это напоминает наборы телекоммуникационных программ в других системах.

Если у вас нет доступа к SLIP или PPP серверу, вы можете использовать `term` для мультиплексирования вашей последовательной линии. `term` обеспечит вам множественный доступ через модем на удаленную машину. `term` также позволит перенаправлять X-клиента на локальный X-сервер через последовательную линию, давая возможность отобразить удаленное X-приложение на вашей Linux-системе. Другой пакет, KA9Q, обеспечивает интерфейс, похожий на SLIP.

BBS хобби многих программистов. Linux поддерживает большое разнообразие программ для BBS, большинство из которых более мощные, чем в других операционных системах. С телефонной линией, модемом и Linux вы можете превратить ваш компьютер в BBS, обеспечив dial-in доступ к своей системе для пользователей с Земного шара. Программное обеспечение BBS для Linux включает XBBS и пакеты UniBoard BBS.

Большинство программ BBS ограничивают пользователя меню-системой, где имеется некоторый фиксированный набор функций. Альтернативой доступу в BBS служит полный спектр возможностей доступа UNIX, который позволяет вам работать с удаленной машиной на правах обычного пользователя.

Если у вас нет возможностей использовать TCP/IP или UUCP, Linux позволяет также связываться с рядом BBS, таких как FidoNet по телефонным линиям и обмениваться новостями и почтой. Дополнительную информацию можно найти в Главе 5.

1.4.6 World Wide Web.

Стоит заметить, что Linux включает программное обеспечение web-сервера и web-браузеры. Наиболее часто встречается сервер Apache. Тысячи Linux систем используют именно Apache, включая Linux Resources site, `www.linuxresources.com`.

Дистрибутивы Linux включают разные web-браузеры, которые можно загрузить и из сети. Доступны Lynx, Mosaic, Netscape, Arena, Amaya и другие.

Linux имеет полную поддержку Java, CGI и Perl, как стандартных инструментов программиста в Linux.

1.4.7 Интерфейс с MS-DOS.

Существуют различные утилиты для связи с миром MS-DOS. Наиболее известен Linux MS-DOS Emulator, позволяющий выполнять многие MS-DOS программы прямо на Linux. Несмотря на то, что Linux и MS-DOS абсолютно различные операционные системы, среда защищенного режима для 80386 позволяет некоторым задачам вести себя так, как это делают прикладные программы MS-DOS.

Эмулятор MS-DOS все еще в стадии совершенствования, но многие популярные пакеты в нем уже выполняются. Понятно, что некоторые приложения MS-DOS, использующие специфические или скрытые свойства системы, никогда не будут выполняться, поскольку эмулятор не знает, как их эмулировать. Например, вы не сможете без шероховатостей выполнять программы, использующие свойства защищенного режима 80386, такие как Microsoft Windows (в расширенном режиме 386-го).

Приложения, которые успешно работают под Linux MS-DOS Emulator включают: 4DOS (интерпретатор команд), Foxpro 2.0, Harvard Graphics, MathCad, Stacker 3.1, Turbo Assembler, Turbo C/C++, Turbo Pascal и WordPerfect 5.1. Стандартные команды и утилиты MS-DOS (такие как PKZIP и т.п.) также работают с эмулятором.

Эмулятор MS-DOS прежде всего предназначается для тех, кому MS-DOS нужен только для выполнения нескольких приложений, но в основном используется Linux. Эмулятор, это не полное повторение MS-DOS. Разумеется, если эмулятор не удовлетворяет вашим потребностям, вы можете использовать MS-DOS непосредственно, как и Linux, на одной и той же машине. При использовании загрузчика LILO можно во время загрузки указать, какую загрузить операционную систему. Linux может сосуществовать с другими операционными системами, с той же OS/2.

Linux обеспечивает "гладкий" интерфейс для обмена файлами между Linux и MS-DOS. Вы можете примонтировать раздел MS-DOS или гибкий диск под Linux и иметь прямой доступ к файлам MS-DOS, как и к "родным".

В настоящее время находится в работе проект, известный как **WINE**: эмулятор Microsoft Windows для X Window System под Linux. По завершению WINE, пользователи будут иметь возможность выполнять MS-Windows приложения прямо из Linux. Это похоже на эмулятор Windows от Sun Microsystems. В момент написания этих строк WINE все еще на ранней стадии создания, но имеет хорошие перспективы.

В главе 5 мы поговорим об инструментарии MS-DOS, доступном из Linux.

1.4.8 Другие приложения.

В Linux огромное количество всевозможных приложений, что и следует ожидать от такой "разносторонней" операционной системы. Основная ориентация Linux была на персональные UNIX-вычисления, но она быстро меняется. Все больше его используют в бизнесе и обучении, все больше появляется на рынке всевозможных коммерческих приложений.

В Linux доступно несколько реляционных баз данных, включая Postgres, Ingres, и Mbase. Это полномасштабные профессиональные системы управления базами данных типа клиент-сервер, похожие на имеющиеся на других платформах UNIX. Имеется также коммерческая база данных `/rdb`.

Прикладные научные пакеты включают `FELT` (Finite Element Analysis Tool); `gnuplot` (анализ данных и черчение); `Octave` (пакет символических вычислений, похожий на MATLAB); `xspread` (калькулятор типа spreadsheet); `xfractint` (X-вариант популярного рекурсивного генератора Fractint); `xlispstat` (пакет статистики) и многое другое. Другие приложения содержат `Spice` (проектирование и анализ цепей) и `Khoros` (аналого/цифровая обработка сигналов и визуализация).

Доступны коммерческие пакеты Maple и MathLab.

Разумеется, есть еще много приложений, которые были или будут перенесены на Linux. Linux обеспечивает полный программный UNIX-интерфейс, удобный в качестве исходной базы для любых приложений в любой научной области.

Как и другие операционные системы, Linux не стоит в стороне от компьютерных игр. Это и классические текстовые "подземельные" игры, вроде Nethack и Moria; игры типа MUDs (Multi-User Dungeons, которые позволяют взаимодействовать многим пользователям), вроде DikuMUD и TinyMUD; а также тьма игр в X, таких как `xtetris`, `netrek` и `Xboard` (X11-версия gnuchess). Популярные игры типа перестреляй-их-всех-в-лабиринтах (Doom и Quake*) также перенесены в Linux.

Для меломанов Linux поддерживает различные саунд-карты, вроде CDplayer (программа, которая может управлять драйвером CD-ROM, как традиционным CD-плейером), MIDI последовательности и редакторы (позволяющие создавать музыку на синтезаторе или другом, управляемом MIDI инструменте) и саунд-редакторы цифровой записи.

Вы не можете найти нужную прикладную программу? Карта программ Linux, приведенная в приложении A имеет большой список пакетов, которые были написаны для Linux или перенесены в него. Список далеко не полный, хотя и перечисляет большое количество пакетов. Другой способ поиска приложений в Linux, это просматривать файлы INDEX на FTP серверах под Linux, если вы имеете доступ в Internet. Оглянитесь вокруг и вы найдете много того, с чем интересно повозиться.

Но если вы совершенно не можете найти того, что вам надо, вы всегда можете попытаться перенести нужные приложения в Linux. Большая часть свободно распространяемых программ для UNIX могут быть откомпилированы для Linux, как правило, без больших проблем. Или, если компиляция не проходит, вы сами можете написать соответствующую программу. Если вам нужна коммерческая программа, возможно, что существует ее свободно распространяемый вариант. А может, вы

убедите компанию сделать выполняемые версии для Linux. История знает случаи, когда удавалось уговорить компании.

1.5 Относительно Copyright для Linux.

Общедоступная Лицензия GNU (**the GNU General Public License**) или кратко **GPL**. *GPL* была разработана для проекта GNU ассоциацией Free Software Foundataion. Она устанавливает некоторые положения относительно распространения и модификации "свободнораспространяемых программ". В данном случае "свобода" относится именно к Свободе, а не к стоимости. *GPL* всегда был источником недопонимания и мы надеемся, что этот обзор поможет вам понять цели и задачи *GPL* и его влияние на Linux. Полная копия *GPL* включена в Приложение С.

Первоначально Линус Торвальдс выпустил Linux под лицензией более ограничивающей, чем *GPL*, которая разрешала свободное распространение и модификацию, но запрещала любые денежные расчеты при передаче и использовании. С другой стороны *GPL* позволяет людям продавать и иметь прибыль со свободно распространяемых программ, но не разрешает ограничивать права других в распространении этих программ любым образом.

Прежде всего следует объяснить, что "свободнораспространяемые программы", под лицензией *GPL* это не *public domain*. Программы *public domain* это программы не защищенные с помощью copyright и, фигурально выражаясь, принадлежат "почтенной публике" обществу. Программы, защищаемые *GPL*, наоборот, защищают авторские права автора или авторов. Это значит, что программы защищены стандартными международными законами copyright, и что автор программ официально обозначен. Так что из факта свободного распространения программ не следует, что они *public domain*.

Программы под лицензией *GPL* не являются также *shareware*. В общем случае программы *shareware* принадлежат и копируются автором, а автор требует присылать деньги за использование программы после ее передачи. А программы под *GPL* могут распространяться бесплатно.

GPL также позволяет людям брать и модифицировать программы, а также распространять свои собственные версии программ. Однако всякая производная работа, основанная на программах, защищенных должна быть под защитой *GPL*. Другими словами, компания не может, взяв и модифицировав Linux, продавать его под ограничительной лицензией. Любые программы, производные от, должны быть также защищены *GPL*.

GPL позволяет распространять и использовать программы бесплатно. Однако, она позволяет человеку или компании распространять программы *GPL* за деньги и даже получать прибыль от продажи и распространения. Однако, при продаже программ под *GPL* дистрибутор не может лишить таких прав (свободного распространения) покупателя. То есть, если вы купили где-то программы под *GPL*, вы можете их свободно распространять, или сами заняться торговлей ими.

Сначала это может звучать как противоречие. Как это так, продавать с выгодой для себя программы, когда *GPL* позволяет любому иметь их бесплатно? Например, предположим, что какая-то компания решила собрать большое число

свободнораспространяемых программ на CD-ROM и заняться их распространением. Эта компания должна вернуть деньги для покрытия расходов на производство и дистрибуцию CD-ROM, компания может также решить сделать на этих продажах прибыль. Это разрешается в соответствии с *GPL*.

Организация, продающая свободнораспространяемые программы, должна следовать определенным ограничениям, выдвигаемым *GPL*. Первое, они не имеют права ограничить права пользователей, купивших программу. Это значит, что если вы купили CD-ROM с программами под *GPL*, вы можете бесплатно их копировать и свободно распространять этот CD-ROM, или тоже продавать. Второе, дистрибуторы должны доступным образом доводить до пользователей информацию о том, что эти программы находятся под защитой *GPL*. Третье, дистрибуторы должны поставить бесплатно полный исходный код распространяемых программ. Это позволит любому, кто купит программы под *GPL*, модифицировать их.

Позволить компаниям распространять и продавать свободнораспространяемые программы - вещь очень хорошая. Не всякий имеет доступ к Internet, чтобы скачать программы, вроде Linux, бесплатно. (прим. переводчика: Правда, сама IP-связь для многих в нашей стране очень и очень даже не бесплатно во много раз дороже, чем это обходится "среднему американцу". О качестве связи и говорить не хочется). *GPL* позволяет компаниям продавать и распространять программы среди тех, кто не имеет свободного доступа к программам. Например, многие организации продают Linux на дискетах, лентах или CD-ROM по почтовым заказам и делают на этом свою прибыль. Разработчики Linux могут никогда не увидеть какую-либо прибыль для себя от этих продаж. Вот каким образом регулируются отношения между разработчиками и дистрибуторами, когда программы находятся под лицензией *GPL*. Другими словами, Линус не спутайте автора с названием ОС Linux знал, что компании могут захотеть продавать Linux, и что он может не увидеть и пенни от этих продаж.

В мире свободнораспространяемых программ важным элементом являются деньги. Цель свободнораспространяемых программ это всегда создание и распространение фантастических программ; чтобы любой мог их свободно достать и использовать. В следующем разделе мы обсудим, как это соотносится с разработкой Linux.

1.6 Проектирование и философия Linux.

Когда новый пользователь сталкивается с Linux, часто возникают ложные ожидания. Linux уникальная операционная система, и важно понимать его философию и особенности проектирования, чтобы эффективно его использовать. Даже если вы умудренный годами UNIX-гуру, вы найдете в последующем интересное для себя.

В фирмах, разрабатывающих коммерческие UNIX, вся система создается под жестким контролем качества, существует система управления написанием программ, внесением изменений, документированием, информированием о выявленных ошибках и их устранением. Разработчикам запрещено по собственному желанию добавлять какие-то свойства или менять критически важные коды по своему желанию. Они могут вносить изменения только, как реакцию на выявленные ошибки, документировать вносимые изменения так чтобы можно было систему при необходимости "вернуть назад". Каждый разработчик закреплен за одной или несколькими частями системного кода, и только этот разработчик имеет право исправлять замеченные ошибки.

Внутри фирм департаменты контроля качества осуществляют жесткое тестирование всякой новой версии операционной системы. Разработчики обязаны под контролем устранять выявленные ошибки. Существует сложная система статистического анализа, определяющая, сколько ошибок должно быть устранено, чтобы объявить переход к новой версии.

Подход, используемый создателями коммерческого UNIX при написании и сопровождении кодов, весьма сложен и это вполне обоснованно. Фирма должна иметь постоянные и хорошо детализированные подтверждения тому, что следующая версия операционной системы созревает для выпуска на рынок. Отсюда сбор и анализ статистики о работе этой системы. Это очень большая и трудоемкая работа - создавать коммерческий UNIX. Часто настолько большая, что требуются сотни, если не тысячи программистов, специалистов по тестированию, писателей документации, административного персонала. Разумеется, никакие два производителя коммерческого UNIX не похожи друг на друга, но общие принципы именно таковы, как описаны выше.

Применительно к Linux вы можете выкинуть из головы вышеописанную концепцию организации разработки большой программной системы, отладки, контроля качества, статистического анализа и т.п. Судя по всему, Linux был и останется хакерской системой. (Что я понимаю под словом "хакер": это фанатически преданный программированию человек, которому нравится работать на компьютере и делать на нем интересные вещи. Это не соответствует пониманию некоторыми слова "хакер", как обозначения для компьютерного хулигана).

Linux первоначально создавался группой энтузиастов в Internet со всего мира. Любой, в Internet и за его пределами, имеющий достаточные знания и навыки, имеет возможность принять участие в совершенствовании и отладке ядра, переносе в Linux новых программ, написании документации, помощи новичкам. Нет определенной организации, отвечающей за развитие системы. Большей частью Linux-сообщество общается через группы по интересам USENET. Существует ряд соглашений для принимающих участие в разработках: например, любой, желающий, чтобы его код был включен в "официальное" ядро, должен написать Линусу Торвальдсу, который проведет тестирование и включит код в ядро (если предлагаемый код вписывается в систему и не противоречит ее принципам скорее всего он будет включен).

Сама по себе система проектируется по открытому принципу. Хотя число вносимых радикальных изменений в систему уменьшается, общая тенденция выдачи новой версии ядра через несколько месяцев (а иногда и чаще) сохраняется. Разумеется, это приблизительная характеристика, она зависит от многих факторов, включая число обнаруженных ошибок, число замечаний пользователей, и то, сколько часов Линус спал в этом месяце.

Естественно, не все ошибки выявляются и не все проблемы решаются между выпусками версий. Но когда создается впечатление, что существенные и часто проявляющиеся ошибки устранены и система ведет себя достаточно "стабильно", выпускается новая версия. Это не попытка выпустить безошибочную версию, а желание выпустить версию UNIX, с которой можно работать. Linux прежде всего ориентирован на разработчиков.

Все, у кого есть новые программы, которые они хотели бы добавить в систему, обычно делают их доступными для других в **альфа-версии**, т.е. на стадии тестирования теми отважными или еще не уставшими пользователями, которые хотят сокрушать возникающие проблемы первоначального кода. Поскольку Linux-сообщество в большой степени кучкуется вокруг Internet, альфа-программы выкладываются на один или более Linux FTP-сервера (смотрите Приложение B) и посылается письмо в одну из Linux-групп USENET, о том как можно получить и тестировать представленный код. Пользователи, которые скачивают и тестируют эти альфа-программы, могут по почте сообщать результаты, указывать ошибки, задавать вопросы автору.

После решения начальных проблем с альфа-кодом, код приобретает статус **бета-версии**, при котором он обычно уже достаточно стабилен, хотя и небезгрешен (он работает, но не все еще его ветви полностью работоспособны). Иначе код попадает в разряд "готового", когда он считается полным и правильным. Относительно новых кодов ядра необходимо просить Линуса включить их в стандартное ядро или добавить как факультативную опцию ядра.

Следует иметь в виду, что это лишь соглашения, а не правила. Некоторые люди так уверены в своих программах, что не считают нужным публиковать альфа-версию. Так что от разработчика зависит в известной мере и процедура.

Вас может несколько удивить столь неупорядоченная система привлечения добровольцев, программирования и отладки UNIX. Может ли она дать вообще положительный результат? Как оказывается, это один из самых эффективных и успешных проектов, когда-либо существовавших. Полностью все ядро было написано без привлечения каких либо частей ранее существовавшего кода. Большая работа была проделана добровольцами по переносу свободно распространяемых программ в Linux. Были написаны библиотеки, создана файловая система, драйверы для многих популярных устройств.

Обычно программная система Linux распространяется в виде *дистрибутива* (*distribution*), содержащего средства инсталляции и раскрутки системы. Большинству пользователей трудно самим собрать систему из разрозненных частей. Но не существует некоего стандартного дистрибутива: их много, каждый со своими преимуществами и недостатками. О различных дистрибутивах Linux речь пойдет в Разделе 2.1.

1.7 Различия между Linux и другими операционными системами.

Важно понимать различия между Linux и другими операционными системами, такими как MS-DOS, OS/2, а также другими реализациями UNIX для персональных компьютеров. Прежде всего, должно быть ясно, что Linux может счастливо сосуществовать с другими операционными системами на той же машине. Как мы увидим, существуют даже способы взаимодействия операционных систем.

Почему Linux?

Почему стоит использовать Linux вместо хорошо известных, хорошо оттестированных, хорошо документированных коммерческих операционных систем? Мы можем привести тысячи причин. Одна из наиболее важных то, что Linux отличный выбор для персональных вычислений в среде UNIX. Если вы разработчик программ в UNIX, зачем дома использовать MS-DOS? Linux позволит вам создавать и тестировать программы для UNIX на вашем персональном компьютере, включая базы данных и приложения для X Window. Если вы студент, то высока вероятность, что университетская компьютерная система работает под UNIX. Linux позволяет вам иметь свой собственный UNIX и перекраивать его по своему вкусу. Инсталляция и использование Linux также прекрасный путь изучения UNIX, если у вас нет доступа к другим UNIX-машинам.

Но не будем зарываться. Linux не только для отдельных любителей UNIX на персоналках. Это большая и достаточно сложная система для решения сложных задач и организации распределенных вычислений. Многие фирмы, особенно небольшие, двигаются в сторону Linux, предпочитая его другим UNIX. Университеты считают Linux отлично подходящим для обучения операционным системам. Крупные поставщики программ начинают понимать, какие выгоды сулит свободное распространение операционных систем.

Linux vs. MS-DOS.

Не является чем-то экзотическим одновременно держать на компьютере Linux и MS-DOS. Многие пользователи Linux работают с прикладными пакетами MS-DOS, вроде различных редакторов.

Хотя Linux имеет собственные аналоги для таких приложений (например, TeX), существуют разнообразные причины, по которым конкретный человек будет использовать MS-DOS наряду с Linux. Если вся ваша диссертация написана с использованием WordPerfect в MS-DOS, возможно вы не сможете конвертировать его в TeX или какой-то другой формат. Существует много коммерческих приложений для MS-DOS, которых нет в Linux, поэтому, если есть смысл, почему не использовать обе операционные системы.

MS-DOS не использует полностью функциональные возможности 80386 и 80486 процессоров. С другой стороны, Linux полностью работает в защищенном режиме процессора и реализует все возможности процессора. Вы можете иметь прямой доступ ко всей имеющейся в распоряжении памяти (и сверх того - используя виртуальную RAM). Linux обеспечивает полный UNIX-интерфейс, отсутствующий в MS-DOS. На Linux вы можете просто писать и отлаживать прикладные программы для UNIX, в то время, как это несложно делать под MS-DOS.

Мы можем обсуждать плюсы и минусы MS-DOS и Linux бесконечно. Между тем, давайте заметим, что Linux и MS-DOS абсолютно разные системы. MS-DOS это дешевая ОС (в сравнении с другими операционными системами), и имеет широкую поддержку в мире персональных компьютеров. Ни одна другая ОС для персональных компьютеров не может сравниться с ней по популярности, прежде всего из-за стоимости. Мало кто из владельцев персональных компьютеров может даже представить, что однажды он потратит $1000 или более только на одну операционную систему. Linux же, кстати, вообще бесплатен, так что у вас есть повод подумать.

Вы можете сами составить свое впечатление от сравнения Linux и MS-DOS, основываясь на том, насколько они отвечают вашим ожиданиям. Linux это не для всякого. Если вам всегда хотелось иметь полномасштабный UNIX дома, не совершая больших затрат, Linux может быть как раз то, что вам надо.

Linux vs. Других систем.

Ряд других продвинутых операционных систем всходит на горизонте мира персональных компьютеров. В частности, OS/2 фирмы IBM и Windows NT фирмы Microsoft становятся все более популярны по мере ухода пользователей из MS-DOS.

Обе OS/2 и Windows NT являются полными многозадачными операционными системами, как и Linux. Чисто технически, OS/2, Windows NT и Linux очень похожи: они имеют похожие интерфейсы с пользователем, систему защиты и т.п. Но главное действительное отличие состоит в том, что Linux есть разновидность UNIX, а отсюда все преимущества принадлежности к UNIX-сообществу.

Что делает UNIX столь важным? Это не только самая популярная операционная система для многопользовательских машин, это также база для большей части свободно распространяемых в мире программ. Если у вас есть доступ к Internet, почти все программы, свободно доступные там, написаны именно для UNIX. (Сам Internet в большой степени стоит на UNIX).

Существует много различных фирм, производящих UNIX и ни одной, ответственной за его распространение. Существует большая тяга к стандартизации в UNIX-сообществе, которая выражается в концепции открытых систем. Но ни одна фирма не контролирует этот процесс. Поэтому любой производитель (или как показала практика хакер) может следовать стандартам UNIX (коль скоро на них нет авторских прав).

OS/2 и Windows NT принадлежат частным компаниям. Поэтому интерфейс и проектные решения контролируются конкретными фирмами и только эти фирмы могут совершенствовать свои продукты. (Не надейтесь увидеть когда-нибудь в обозримом будущем бесплатную версию OS/2). В некотором смысле такая организация дела имеет преимущества: она обеспечивает жесткую стандартизацию программного и пользовательского интерфейсов. OS/2 есть OS/2, где бы вы ее не обнаружили, то же самое с Windows NT.

А интерфейс UNIX постоянно совершенствуется и меняется. Несколько организаций пытаются выработать стандарт программной модели, но эта задача очень сложная. Linux наилучшим образом соответствует стандарту POSIX.1 для программного интерфейса UNIX. Предполагалось, что Linux примкнет еще к ряду движений по стандартизации, но это не стало пока в число главных задач Linux-сообщества.

Linux vs. Другие реализации UNIX.

Существует ряд других реализаций UNIX для 80386 и 80486. Архитектура 80386 сама подталкивает к проектированию UNIX, поэтому многие разработчики воспользовались этим преимуществом. Другие реализации UNIX, учитывавшие особенности архитектуры процессора весьма похожи на Linux. Вы можете убедиться, что почти все коммерческие версии UNIX поддерживают практически одинаковую программную

среду и сетевые характеристики. Однако имеются и значительные отличия между Linux и коммерческими UNIX.

Другие версии UNIX для персоналок обычно проще Linux. Все коммерческие версии имеют некоторый минимальный набор программного обеспечения, поддержку сети и инструменты разработки.

Прежде всего Linux поддерживает иной спектр аппаратных средств. Linux поддерживает большинство хорошо известных устройств, но поддержка ограничена той аппаратурой, к которой пользователи действительно имеют доступ. Разработчики коммерческих UNIX обычно имеют больший список устройств, хотя Linux и отстает незначительно. Про особенности аппаратуры поговорим в Разделе 1.8.

Многие пользователи отмечают, что Linux по крайней мере не менее надежен, чем коммерческие UNIX. Linux все еще находится в стадии развития, и некоторые вещи (вроде TCP/IP) недостаточно стабильны, но постоянно совершенствуются.

Наиболее важным фактором для многих является цена. Linux распространяется свободно, если вы имеете доступ в Internet (или другую компьютерную сеть) и можете его скачать. Если у вас нет выхода в такую сеть, то вы можете купить его на дискетах или CD-ROM. Разумеется, вы можете скопировать Linux у друга или разделить с кем-то стоимость его покупки. Если вы планируете инсталлировать Linux на большом количестве машин, вам достаточно купить всего одну копию. На тиражирование нет лицензионных ограничений.

Не следует преуменьшать реальную стоимость коммерческих UNIX, поскольку наряду со стоимостью собственно программ, сюда входят также стоимость документации, сопровождение, гарантия качества. Это очень важные составляющие для больших организаций, но, может быть, не столь существенные для индивидуальных пользователей. В любом случае, во многих фирмах и в университетах считают, что использование Linux в лабораториях на недорогих персональных компьютерах предпочтительнее коммерческого UNIX в лаборатории, укомплектованной рабочими станциями.

Как пример из "реального мира", можно сказать о том, что Linux путешествовал по северу Тихого океана, выполняя работу по телекоммуникации и анализу данных на океанографическом исследовательском судне. Linux используется на исследовательской станции в Антарктике. Несколько госпиталей используют Linux для ведения историй болезни. Он доказал, что он надежен и удобен, как и другие реализации UNIX.

Существуют и другие бесплатные или недорогие реализации UNIX для 386 и 486. Одна из наиболее известных реализаций 386BSD. 386BSD совместима с Linux во многих аспектах, но какая из них "лучше" зависит от ваших личных желаний и ожиданий. Мы можем указать одно существенное отличие: Linux создавался открыто (когда каждый доброволец мог внести свой вклад в процесс создания), а 386BSD создан замкнутой группой программистов, которые и поддерживают систему. Поэтому существует заметное различие в философии и разработке между этими двумя проектами. Цели двух проектов существенно различны: цель Linux - создать полную UNIX-систему от начала (и получить большое удовольствие от этого процесса); а цель 386BSD частично состоит в модификации существующего кода BSD, применительно к 386.

NetBSD это другая версия BSD NET/2 для нескольких машин, включая 386. NetBSD имеет слегка более открытую концепцию разработки и сравнима с 386BSD по многим аспектам.

Другой проект HURD попытка Free Software Foundation создать и распространять свободную версию UNIX для многих платформ. За дополнительной информацией об этом проекте обращайтесь в Free Software Foundation (адрес дан в Приложении C). В момент написания книги проект HURD все еще на начальной стадии.

Существуют также другие недорогие версии UNIX, например Coherent (приблизительно $99) и Minix (академический, но полезный UNIX, на котором первоначально базировался Linux). Некоторые из этих реализаций представляют преимущественно академический интерес, в то время, как другие нормальные полномасштабные системы. Нет смысла говорить о том, как много индивидуальных пользователей UNIX двигаются в сторону Linux.

1.8 Требования к оборудованию.

Вы должны быть убеждены, что Linux прекрасен и что вы пополните его грандиозными вещами. Но это потом, а до того, как броситесь его инсталлировать, вы должны сориентироваться в требованиях к аппаратуре, которые диктует Linux.

Имейте в виду, что Linux был создан самими пользователями. Это означает, что большая часть поддерживаемого Linux оборудования это то, что пользователи реально у себя имеют. Как в результате оказалось большая часть популярной периферии для 80386/80486 поддерживается (действительно, Linux поддерживает оборудование, которое в ряде случаев не поддерживают некоторые коммерческие UNIX). Хотя некоторые достаточно экзотические устройства пока не поддерживаются. Если какое-то из облюбованных вами устройств пока не поддерживается в Linux, есть смысл надеяться, что оно скоро будет поддерживаться.

Многие компании лицензируют интерфейс устройств, поэтому добровольные разработчики Linux просто не могут написать эти драйверы. Иначе это будет нарушением авторских прав соответствующей компании. Мало что можно изменить в этой ситуации. Иногда хакеры пишут драйверы, основываясь на своих представлениях о конкретном интерфейсе. В других случаях разработчики с разной степенью успеха пытаются работать с соответствующими компаниями, пытаясь получить информацию об интерфейсе.

В следующем разделе мы попытаемся дать резюме технических требований Linux. *Linux Hardware HOWTO* (см. Раздел 1.9) содержит более полный перечень оборудования, поддерживаемого Linux.

Внимание: Большая часть драйверов Linux в настоящее время находится в стадии разработки. Различные дистрибутивы могут содержать разные наборы драйверов. Здесь прежде всего перечисляются те драйверы, которые уже поддерживаются определенное время и зарекомендовали себя как достаточно стабильные. Дополнительную информацию по дистрибутивам смотрите в Разделе 2.2).

Linux доступна на многих платформах в дополнение к Intel 80x86. Это Macintosh, Amiga, Sun SparcStation и системы на основе Digital Equipment Corporation Alpha. В данной книге рассматриваются в основном системы на 80386, 80486 и Pentium процах, а также их клоны AMD, Cyrix и IBM.

Требования к материнской плате и процессору.

В настоящее время Linux поддерживает системы на Intel 80386, 80486 или Pentium CPU. Это включает все вариации этих процессоров, такие как 386SX, 486SX, 486DX и 486DX2. С Linux могут работать также "неинтеловские" клоны процессоров, вроде AMD и Cyrix. Linux также портирован на DEC Alpha и Apple PowerMac.

Если у вас 80386 или 80486SX, вы можете также иметь сопроцессор, хотя это и не обязательно, (ядро Linux может эмулировать FPU). Поддерживаются все стандартные сочетания FPU, такие сопроцессоры как IIT, Cyrix FasMath и Intel.

Материнская плата должна использовать шину ISA, EISA, VLB или PCI. Это определяет, как система взаимодействует с периферией и другими компонентами главной шины. Шина MicroChannel (MCA) фирмы IBM, используемая на машинах типа IBM PS/2, пока не поддерживается. Поддерживаются системы с локальными шинами, ускоряющими доступ к видео и дискам.

Требования к памяти.

Linux требует совсем немного памяти в сравнении с другими развитыми операционными системами. Вы должны иметь как минимум 4 Мбайт RAM; хотя настоятельно рекомендуется иметь не менее 16 Мбайт. Чем больше памяти, тем быстрее работает система. Многие дистрибутивы для установки требуют много памяти.

Linux может поддерживать все 32-битовое адресное пространство процессоров 386/486; другими словами, он автоматически использует всю память.

Linux может успешно работать на 4 Мбайтах RAM, включая всяческие свистульки и погремушки, вроде X Window, `Emacs` и т.п. Хотя, иметь побольше памяти не менее важно, чем иметь помощнее процессор. 16 Мбайт более подходит для индивидуального использования; 32 Мбайт или более если вы предполагаете более серьезную загрузку системы.

Большинство пользователей Linux выделяют часть жесткого диска для области свопинга, которая используется как **виртуальная RAM**. Если даже вы имеете много реальной физической памяти, RAM, вы можете пожелать иметь область свопинга. Хотя область свопинга не заменяет действительной физической памяти, она может позволить выполнять на вашей системе более объемные приложения, удаляя неактивную часть программы на диск. Размер области свопинга, которую вы должны выделить, зависит от нескольких факторов; мы вернемся к этому в Разделе 2.

Требования к драйверам жесткого диска.

Вам не обязательно иметь драйвер жесткого диска для работы в Linux. Вы можете работать с минимальной системой с гибкого диска. Но это медленно и имеет много ограничений, да и большинство пользователей имеет доступ к памяти на жестких дисках. У вас должен быть 16-ти битный контроллер в стандарте AT. В ядре есть

поддержка 8-ми битного XT-стандарта; хотя разумеется, большинство контроллеров использует сегодня AT-стандарт. Linux может поддерживать все MFM, RLL и IDE контроллеры. Поддерживается большинство (но не все) ESDI контроллеры.

Общее правило для не-SCSI драйверов жестких дисков состоит в том, что если вы можете иметь доступ к этим устройствам из MS-DOS или другой ОС, значит вы можете работать с ними и в Linux.

Linux может также поддерживать многие популярные SCSI контроллеры, хотя поддержка SCSI ограничена из-за большого разнообразия существующих стандартов таких интерфейсов. Поддержка SCSI контроллеров включает Adaptec AHA1542B, AHA1542C, AHA1742A (BIOS version 1.34), AHA1522, AHA1740, AHA1740 (SCSI-2 controller, BIOS 1.34 in Enhanced mode); Future Domain 1680, TMC-850, TMC-950; Seagate ST-02; UltraStor SCSI; Western Digital WD7000FASST, все на чипсетах от NCR, все карты Adaptec и Buslogic. Также должны работать клоны, базирующиеся на этих картах.

Требования к дисковому пространству.

Разумеется, для инсталляции Linux вам необходимо иметь некоторое свободное пространство на жестком диске. Linux поддерживает различные драйверы жестких дисков на одной машине; вы можете выделить место для нескольких устройств, если это необходимо.

Размер необходимого пространства зависит в большой степени от ваших потребностей и программ, которые вы инсталлируете. Linux сравнительно компактный UNIX; вы можете для всей системы занять 10 - 20 Мбайт. Между тем, если вы хотите иметь место для расширения и для больших пакетов, вроде X Window, вам потребуется больше места (200-500 мегабайт). Если вы планируете множественный доступ, вам потребуется иметь место и для файлов других пользователей.

Кроме того, даже если вы имеете большое количество физической RAM (16 Мбайт или более), скорее всего вы захотите иметь область свопинга, используемую виртуальной памятью. Детали мы обсудим в разделе 2.

Каждый дистрибутив Linux обычно сопровождается какой-то литературой, которая может помочь вам определиться с объемом необходимой памяти в зависимости от того, какое программное обеспечение вы планируете поставить. Минимальную систему вы можете эксплуатировать менее, чем на 20 Мбайтах. Полная система со всеми свистульками и погремушками потребует до 250-300 Мбайт. Очень большие системы для многих пользователей, где зарезервировано место для последующих расширений, потребует 500-650 Мбайт. Но это лишь грубые прикидки, которые вы уточните исходя из своих потребностей.

Требования к монитору и видеоадаптеру.

Linux поддерживает все стандарты Hercules, CGA, EGA, VGA, IBM monochrome и SuperVGA видеокарт и текстовые мониторы. В общем случае, если видеокарта и монитор работают под другими ОС вроде MS-DOS, они будут работать и под Linux. Оригинальные карты IBM CGA дают в Linux "снег", так что их неприятно использовать.

Графическое окружение вроде X Window имеет свои требования к видеооборудованию. Вместо перечисления этих требований мы перенесем обсуждение в раздел 5.1. А кратко, для работы с X Window System на вашем Linux, вам требуется одна из видеокарт, перечисленных в этом разделе.

Прочее оборудование.

У большинства пользователей есть "специфическое" оборудование, вроде стриммера, на CD-ROM, звуковой карты и т.д., поэтому они интересуются, поддерживается ли оно в Linux. Читайте дальше.

Мышь и другие устройства, подключаемые к портам.

Большей частью мышь вы будете использовать в графическом окружении, таком как X Window System. Но некоторые приложения Linux, не ассоциируемые с графической средой, также используют мышь.

Linux поддерживает все стандарты последовательно подключенной мыши, включая Logitech, серию MM, Mouseman, Microsoft (2 кнопки) и Mouse Systems (3 кнопки). Linux также поддерживает мышь, подключенную на шину: Microsoft, Logitech и ATIXL. Поддерживается также интерфейс мыши PS/2.

Все прочие подключаемые таким же образом устройства, которые эмулируют вышеперечисленных мышей, тоже должны работать.

CD-ROM.

Почти все драйверы CD-ROM используют интерфейс IDE. Но некоторые все же работают со SCSI. Если у вас есть SCSI-адаптер, поддерживаемый Linux, то ваш SCSI CD-ROM должен работать. Проверена работоспособность ряда CD-ROM под Linux, включая NEC CDR-74, Sony CDU-541 и CDU-31a, Texel DM-3024 и Mitsumi.

Linux поддерживает также стандарт ISO-9660 файловой системы для CD-ROM и High Sierra file system extensions.

Стриммеры.

Сейчас на рынке имеется ряд стриммеров. Большинство из них используют SCSI-интерфейс и практически все поддерживаются Linux. Среди проверенных устройств Sankyo CP150SE; Tandberg 3600; Wangtek 5525ES, 5150ES и 5099EN с адаптером PC36. Другие QIC-02 устройства также должны поддерживаться.

Принтеры.

Linux поддерживает весь спектр параллельных принтеров. Если вы можете подключить ваш принтер на параллельный порт в MS-DOS или в другой операционной системе, то он может работать и в Linux. Программная поддержка принтера в Linux состоит из стандартных для UNIX программ `lp` и `lpr`. Эти программы позволяют также организовать удаленную печать через сеть.

Linux также имеет софт для распечатки на большинстве принтеров файлов в формате PostScript.

Модемы.

Как и с поддержкой принтеров, Linux поддерживает полный спектр последовательных модемов, как внешних, так и внутренних. В Linux много программ телекоммуникации, включая Kermit, `pcomm`, `minicom` и `Seyon`. Если ваш модем может работать с другой операционной системой, вы сможете с ним работать и в Linux без каких-либо проблем.

Карты Ethernet.

Многие популярные карты Ethernet и LAN-адаптеры поддерживаются в Linux. Также есть поддержка для FDDI, frame relay, Arcnet и token ring карт. Список поддерживаемых карт входит в исходники ядра Вашего дистрибутива Linux.

1.9 Источники информации по Linux.

Как вы, возможно, догадались, существует много источников информации по Linux помимо этой книги. В частности, есть ряд книг, не конкретно по Linux, а скорее по UNIX вообще, которые могут очень помочь, особенно тем, кто не имеет предварительного опыта в UNIX. Если вы не знакомы с миром UNIX, мы настоятельно советуем вам уделить время одной из таких книг, прежде чем храбро ринуться в джунгли Linux. В качестве хорошего начала может быть использована книга Grace Todino и John *Learning the UNIX Operating System*.

Многие из нижеупомянутых источников доступны он-лайн в электронном виде. То есть для того, чтобы ими воспользоваться, вы должны иметь доступ к Internet, USENET или Fidonet. Начните с `www.linuxresources.com` (см Приложение A). Если у вас нет он-лайн доступа к этим материалам, то следует найти добряка, который сделает вам твердую копию.

1.9.1 Документация, доступная он-лайн.

Если у вас есть доступ к Internet, вы можете найти много документации по Linux через anonymous FTP из архивов, расположенных по всему миру. Если же вы не имеете прямого доступа к Internet, эту документацию все-таки можно достать: много дистрибутивов Linux, содержащих документы, упомянутые здесь, продается на CD-ROM. Они также распространяются по многим другим сетям, вроде Fidonet и CompuServe. Если вы в состоянии послать почту в Internet, значит вы можете достать и эти файлы, используя один из ftpmail-серверов, которые пошлют вам документы или файлы электронной почтой из FTP-архивов. См. в Приложении B дополнительную информацию по использованию `ftpmail`.

Существует большое число FTP-архивов, содержащих тексты программ Linux и соответствующую документацию. Список известных архивов Linux дан в Приложении B. Для того, чтобы минимизировать сетевой трафик, всегда следует использовать FTP-сервер, находящийся в географической близости.

Приложение A содержит список документации на Linux, доступной через anonymous FTP. Имена файлов будут различаться на разных серверах. Большинство серверов хранит документацию на Linux в подкаталоге `docs` Linux-овского архивного

подпространства. Например, на FTP сервере `sunsite.unc.edu` файлы Linux помещены в каталог `/pub/Linux`, а документация по Linux в `/pub/Linux/docs`.

Примеры доступной документации - это *Linux FAQ* (Frequently Asked Questions), собрания часто задаваемых вопросов (в данном случае) по Linux; документация *Linux HOWTO*, описывающая специфические аспекты системы, включая *Installation HOWTO*, *Printing HOWTO*, *Ethernet HOWTO* и Linux META-FAQ: перечень других источников информации по Linux в Internet.

Большинство этих документов регулярно посылается в несколько относящихся к Linux групп USENET; смотрите раздел 1.9.4.

1.9.2 Linux и World Wide Web.

Начальная страница (Home Page) Документации на Linux доступна на WWW по адресу

```
http://sunsite.unc.edu/LDP
```

Эта страница содержит много HOWTO и других документов в формате HTML, а также ссылки на другие сервера, интересные для пользователей Linux, например, *Linux Journal*, который можно найти на `http://www.ssc.com/`.

1.9.3 Книги и другие публикации.

В настоящее время существует всего несколько публикаций специально посвященных Linux. Большей частью это книги из Linux Documentation Project (LDP) проекта, реализуемого в Internet по написанию и распространению "руководств" по Linux. Эти руководства аналогичны документации, поставляемой с коммерческими версиями UNIX: они покрывают все: от инсталляции Linux до его использования, программирования, работы в сети, разработки ядра и т.д.

Руководства LDP доступны через FTP-серверы Internet, а также по обычной электронной почте из нескольких источников (см. Приложение A).

Много компьютерных издательств в последнее время начали уделять внимание Linux, так что лед тронулся, следите за книжными магазинами, может что-то и появится... Много интересного есть на сайтах `http://www.ssc.com/` и `http://www.linuxjournal.com`.

И хотя собственно по Linux книг немного, тем не менее есть большое число книг по UNIX вообще, которые, без сомнения, могут использоваться применительно к Linux, поскольку Linux не имеет значительных отличий от других реализаций UNIX. Короче, все, что вы хотите узнать о программировании на Linux, может быть найдено в этих книгах, предназначенных для широкой аудитории пользователей UNIX. А здесь мы представляем наиболее важные специфические детали Linux и надеемся, что вы также будете обращаться за деталями и к другим источникам.

Вооружившись некоторым количеством хороших книжек по UNIX, а также данной книгой, вы будете способны разобраться во всем. Приложение А включает список настоятельно рекомендуемых книг по UNIX, как для новичков в UNIX, так и для асов.

Есть также ежемесячный журнал по Linux, под названием *Linux Journal*. Он распространяется по всему миру и представляет хорошую возможность быть в курсе происходящего в Linux-сообществе, особенно, если вы не имеете доступа к новостям USENET (см. ниже). Относительно подписки на *Linux Journal* см. Приложение А.

1.9.4 Новости USENET.

Usenet это мировые электронные новости и одновременно клубы по интересам, так называемые ``**newsgroups**'' (у нас принято говорить "телеконференции"). Многое по развитию Linux делается через Internet и USENET, так что не удивляет существование ряда телеконференций USENET, посвященных Linux.

Первоначально была создана телеконференция `alt.os.linux`, чтобы вынести дискуссии по Linux из comp.os.minix. Но скоро трафик `alt.os.linux` вырос настолько, что в феврале 1992 года было проведено голосование о включении конференции в иерархию "comp" и создана телеконференция `comp.os.linux`.

`comp.os.linux` быстро стала одной из наиболее популярных (и шумных) телеконференций USENET, более популярной, чем другие группы иерархии `comp.os`. В декабре 1992, в результате голосования группа была разбита, чтобы уменьшить объемы трафика, но только `comp.os.linux.announce` прошла тогда по голосованию. В июле 1993, наконец-то группа была разбита на иерархию групп. За это проголосовало около 2000 человек, одно из самых больших голосований в истории USENET.

Если у вас нет прямого доступа к USENET, но имеется возможность посылать и получать письма по e-mail, то в Internet существуют шлюзы, через которые можно получать телеконференции:

comp.os.linux.announce

> это моделируемая (контролируемая) телеконференция, содержащая объявления и важные сообщения относительно Linux (например, сообщения об ошибках, существенные исправления и т.п.). Если вы вообще собираетесь читать телеконференции Linux - эту читайте обязательно. Часто важные сообщения этой конференции не дублируются в другие. Также в этой же телеконференции бывают важные оперативные сообщения.
>
> Любое сообщение в этой группе должно быть одобрено модераторами Matt Welsh и Lars Wirzenius. Если вы хотите выставить свое сообщение в этой группе, вы обычным порядком посылаете его в телеконференцию. Оно автоматически будет направлено для одобрения модератору. Но если ваша система новостей не настроена, можно направить статью электронной почтой по адресу `linux-announce@tc.cornell.edu`. Все остальные телеконференции этой иерархии linux, перечисленные ниже немодерируемые.

comp.os.linux.alpha

Все о работе с Linux на процессорах Digital Alpha.

comp.os.linux.advocacy

Обсуждение связи между Linux и другими ОС.

comp.os.linux.answers

Для посылки Linux FAQ, How-To, README и прочих документов, которые отвечают на вопросы о Linux. Снижает трафик в других конференциях c.o.l.*.

comp.os.linux.development.apps

Разработка и портирование приложений в Linux.

comp.os.linux.hardware

Обсуждение проблем с работой различного оборудования под Linux.

comp.os.linux.m68k

Портирование Linux на процессоры типа m68k.

Следует заметить, что телеконференция comp.os.linux была заменена иерархией телеконференций. Если у вас есть доступ к `comp.os.linux` и нет доступа к другим телеконференциям этой иерархии, вдохновите администратора их завести. Приведенный здесь список очень далек от полноты. Конференции превращаются в иерархии, появляются новые конференции...

1.9.5 Списки рассылки Internet.

Если у вас есть доступ к электронной почте Internet, вы можете включиться в списки рассылки, даже если вы не имеете доступа к USENET. Даже если вы не можете работать непосредственно в Internet, но можете обмениваться с ней почтой (например, UUCP, FidoNET, CompuServe и другие сети имеют почтовую связь с Internet), вы можете подключиться к какому-то списку рассылки. Список рассылки ``Linux Activists'' предназначен для разработчиков Linux и людей, заинтересованных в содействии процессу разработки. Это "многоканальная" рассылка; можно подключиться к разным ее ветвям: NORMAL: общие вопросы Linux; KERNEL: разработка ядра; GCC: разработка gcc-компилятора и библиотек; NET: протоколы TCP/IP; DOC: документация по Linux; и другие.

За дополнительной информацией о списках рассылки, связанных с Linux, пишите на `majordomo@vger.rutgers.edu`.

Вы сможете получить действующий перечень тем рассылки, включая информацию о том, как включиться в список рассылки и как отписаться. Пошлите в теле сообщения слово `help`, и Вам будет выслано письмо с пояснениями как включиться в список рассылки и как отписаться. Слово `lists` вернет имена списков рассылки, поддерживаемых сервером `majordomo.vger.rutgers.edu`.

Существует несколько списков рассылки для Linux. Лучше всего для ориентации посмотреть объявления в телеконференциях Linux USENET, а также просмотреть списки тем рассылки, периодически публикуемые в USENET group `news.answers`.

1.10 Получение помощи.

Без сомнения, вам понадобится какая-то помощь во время ваших приключений в мире Linux. Даже самые крутые из крутых юниксистов временами спотыкаются о какие-то закорючки Linux, поэтому важно знать, как и где искать помощи, когда потребуется.

Первоочередной источник помощи списки рассылки и телеконференции USENET обсуждались в Разделе 1.9 . Если у вас нет к ним онлайнового доступа, вы можете поискать доступ к такой информации на локальных BBS, в CompuServe и т.д. Посмотрите также `http://www.linuxjournal.com/techsup.html`.

Ряд фирм предлагают коммерческую поддержку Linux. Вы можете платить за "подписку", что позволит вам созваниваться с консультантами по поводу возникающих проблем с Linux. Но при наличии доступа к USENET и почте Internet вы можете найти и бесплатное сопровождение.

Но сначала ищите ответы во всей доступной вам документации! Прежде всего в источниках, перечисленных в Разделе 1.9 и Приложении A. Эти документы тщательно написаны для людей вроде вас, которые нуждаются в помощи при работе с Linux. Даже книги, написанные про UNIX вообще, применимы к Linux и вы должны извлекать из них пользу для себя, и как правило, найдете ответы на интересующие вас вопросы.

Приучайтесь к самостоятельности. В большинстве случаев предпочтительно самостоятельно исследовать проблему насколько возможно, прежде чем обращаться за помощью. Прежде всего обращайтесь за помощью к самому Linux. Помните, что Linux некоммерческая ОС и никогда не пыталась таковой выглядеть. Вы не умрете, если сами займетесь хакерством. Это приучит вас самостоятельно решать проблемы, может через некоторое время вы сами почувствуете себя в Linux гуру.

Сохраняйте спокойствие. Ни в коем случае не приходите в отчаяние в результате общения с системой. Вы ничего не добьетесь от системы с помощью топора или, того лучше, с помощью мощного электромагнита. Linux взрослеет и мужает, становится надежнее, так что мы надеемся, что число проблем будет уменьшаться. Кстати, всяких фокусов можно ожидать и от коммерческих UNIX-ов.

Воздерживайтесь от скоропалительных решений. Многие люди бросаются рассылать письма по конференциям до того, как спокойно подумают. Часто решение находят через пять минут после того, как закончили взывать о помощи к мировому сообществу.

Если вы решили послать просьбу о помощи, постарайтесь это сделать наилучшим образом. Постарайтесь оформить просьбу максимально вежливо и предельно информативно. Не забывайте, что помощь в сети оказывается добровольно.

---

2 Приобретение и инсталляция Linux.

David Bandel переписал и откорректировал первую часть главы по установке. Другие части были выполнены авторами, которые написали части главы по отдельным дистрибутивам Linux.

Boris Beletsky написал часть о дистрибутиве Debian. Sean Dreilinger написал часть о дистрибутиве Slackware. Henry Pierce wrote написал часть о дистрибутиве Red Hat Linux. Evan Leibovitch написал часть о дистрибутиве Caldera OpenLinux. Larry Ayers написал часть о дистрибутиве S.u.S.E. Linux.

2.1 Общие принципы инсталляции.

В отличие от большинства других операционных систем, Linux может быть получена бесплатно. Благодаря GNU General Public License, согласно которой распространяется Linux (см. приложение C), никто не может продавать Вам лицензию на программное обеспечение. Вы можете использовать Linux бесплатно и поощрять к этому других.

Но это не означает, что компании не могут брать деньги на компенсацию издержек копирования плюс прибыль. Они могут также добавлять программное обеспечение, которое не распространяется свободно, но выполняется в системе.

Это дает Вам свободу выбора. Если закупка CD-ROM не входит в Ваши планы, Вы можете просто позаимствовать копию друга или загрузить программу из Internet. Вы получаете ту же самую операционную систему и пакеты программ, независимо от того, купили диск или скачали дистрибутив с FTP (см. приложение B).

2.1.1 Основные дистрибутивы Linux.

Тщательное изучение разных дистрибутивов находится здесь. Это дистрибутивы: Debian, Red Hat, Caldera, Slackware и S.u.S.E. Каждый раздел имеет большое количество информации относительно того, где получить дистрибутив. Но помните, Linux это ядро. Программное обеспечение это часть дистрибутива, а не Linux. Большинство программного обеспечения свободно доступно и может быть перенесено между различными UNIX платформами.

Каждый дистрибутив имеет собственную программу установки и утилиты сопровождения, которые упрощают администрирование системы и установку. Каждый нацелен на различную аудиторию. Я рекомендую, чтобы Вы почитали о каждом

дистрибутиве и поговорили с хорошо осведомленными друзьями. Крупные города имеют Группу пользователей Linux. В выборе дистрибутива надо быть внимательным: его смена обычно приводит к установке системы с нуля.

2.1.2 Общие положения.

Этот раздел делает предположения об уровне новичка в Linux:

- имеет компьютер с MS-DOS и Windows или OS/2;
- имеет представление о MS-DOS, но не об UNIX;
- знает или может найти сведения об аппаратуре компьютера;
- имеет желание "испытывать" Linux по любой причине; и
- не имеет несколько сот свободных мегабайт на диске.

Эти предположения не догма, и могут даже быть немного консервативными.

2.1.3 Аппаратура.

Этот раздел объясняет все шаги, которые необходимо выполнить до начала установки. Каждый дистрибутив выполняет эту подготовку немного по-своему. Несмотря на это, все процессы подготовки выполняют примерно одно и то же. Все они требуют выполнения шагов:

- Планирование;
- Сбор информации об аппаратных средствах системы;
- Резервирование Вашей старой системы (необязательно, но строго рекомендуется);
- Подготовка разделов Linux;
- Установка загрузчика (для систем с двойной загрузкой);
- Загрузка ядра Linux;
- Установка ядра;
- Выбор и установка программных пакетов;
- Загрузка программного обеспечения;
- Окончательная конфигурация системы; и
- Перезагрузка в работающую систему.

Теперь, когда я достаточно упростил процесс, давайте разбираться в деталях.

2.1.4 Планирование.

Сначала определим какое оборудование имеется. Контрольный список был включен, чтобы помочь Вам. Будьте настолько точны насколько возможно, но в разумных пределах. Например, если Вы имеете плату локальной сети Ethernet, Вы должны знать производителя (например, SMC-Ultra, 3Com 3C509...), адрес I/O (например, io=0x300), прерывание (IRQ 10), но не аппаратный адрес (00 00 a6 27 bf 3c). Не вся информация будет необходима для ваших аппаратных средств. Если Вы работаете с Windows 95 или Windows NT, скопируйте значения из окна сведений об аппаратуре. Посмотрите руководство или сайт производителя.

2.1.5 Таблица данных о системе.

General

Процессор: Тип: 386 486 Pentium PPro

Скорость (опционально):

Mhz:

Intel AMD Cyrix

Мат. плата: Производитель: Чипсет:

Пример: Производитель: неизвестен Чипсет: triton II

Мышь: Производитель: Тип: bus PS/2 serial port

Если serial: COM1 (ttyS0) COM2 (ttyS1)

Жесткие диски: Тип: IDE/MFM/RLL/ESDI SCSI

Емкость (список всех дисков):

Если SCSI контроллер: Производитель: Модель:

Пример: Производитель: BusLogic Модель: 948

Загрузка: Linux DOS/Windows OS/2 Other

Диск: Раздел: Размер: Boot:

CD-ROM: IDE/ATAPI SCSI Proprietary

Производитель: Модель:

(Proprietary only):

X-Windows:

Видеокарта: Производитель: Модель:

RAM: 1Mb 2Mb 4Mb 8Mb 16Mb

Монитор: Производитель: Модель: Max scan rate:

Сеть:

Модем: Производитель: Модель:

Порт: COM1 COM2 COM3 COM4

(ttyS0) (ttyS1) (ttyS2) (ttyS3)

Имя машины: *Например, rainier*

Следующие ответы необходимы только при использовании сетевой платы интерфейса (NIC): (не конфигурируйте работу с сетями, если Вы не имеете установленного NIC).

```
NIC тип:  ethernet  token ring  FDDI  other

NIC Производитель:   Модель:

Сетевое доменное имя:   (Например, mountains.net)

IP адрес:    (Ex: 192.168.1.2)

Network адрес:     (Ex: 192.168.1.0)

Netmask:     (Ex: 255.255.255.0)

Broadcast адрес:    (Ex: 192.168.1.255)

Gateway(s):    (Ex: none or 192.168.1.1)

DNS(s):     (Ex: 192.168.1.2)
```

В принципе не вся информация нужна немедленно. Например, прямо сейчас не нужно знать тип процессора и чипсета на плате. Но если информация доступна, лучше ее иметь.

2.1.6 Мышастики.

Если планируется использовать мышь, нужна информация о ней. Надо знать производителя мыши, поскольку разные марки выполняют внутренние функции по-разному. Будьте внимательны. Если Вы имеете мышь с маркой Microsoft, она может иметь последовательный интерфейс или PS/2. Ряд компьютеров поставляются с мышами, которые напоминают последовательных мышей и имеют разъем последовательного типа, но реально связаны с системной платой как PS/2!

Посмотрите метку на нижней стороне мышки. Если Вы имеете мышь с тремя кнопками, и переключателем на нижней части, который Вы можете установить в положения, скажем, Microsoft и PC, выберите PC. Установка Microsoft не обрабатывает среднюю кнопку, которая является полезной в UNIX. Для какого-нибудь неизвестного изготовителя выберите подходящую позицию переключателя, так как это задает используемый протокол передачи сигналов. Не существует драйверов для "Cutie" мыши, но они существуют для режимов настройки переключателя Microsoft и Mouse System на нижней части мыши.

Теперь надо узнать через какое устройство система работает с мышкой. Linux должна знать как к нему обратиться. Если Вы имеет мышь на PS/2, Вы обычно используете `/dev/psaux`, дополнительный порт для указательного устройства PS/2, или `/dev/psmouse`, его синоним, доступный для использования. Bus-мышки работают через специальный файл устройства (для каждой мыши свой), например `/dev/atibm` для ATI bus, `/dev/logibm` для Logitech bus, `/dev/inportbm` для InPort bus, или через их синонимы: `atimouse`, `logimouse`. Для последовательных мышей, если Вы знаете MS-DOS `COM:` порт, подставьте `/dev/ttyS0` для `COM1:` и `/dev/ttyS1` для `COM2:`.

2.1.7 Данные о жестких дисках и CD-ROM.

Перед инсталляцией надо разобраться сколько места на диске можно выделить для Linux. Поскольку она использует специальный раздел, оценивать надо точно.

Ваш жесткий диск может быть практически любого типа. IDE, MFM, RLL или ESDI в принципе одно и то же, и обозначаются одним названием IDE.

Если Ваш жесткий диск имеет интерфейс SCSI, загрузка будет посложнее. Вы должны знать марку и модель контроллера SCSI. Наиболее часто встречаются контроллеры Adaptec и BusLogic, но они не единственные... К тому же, от модели тоже многое зависит. Есть особые модели, например, AHA-1572 или BTC-958. Эта информация часто отображается при инициализации системы.

Для выделения места, надо оценить размер диска. Под OS/2 Вы можете использовать весь жесткий диск для OS/2, затем установить Microsoft Windows на разделе с OS/2 и выполнять Microsoft Windows под OS/2. Если Вы имеете MS-DOS, и Microsoft Windows, или OS/2 на вашем компьютере, Linux должна иметь собственный раздел. Она может быть загружена на разделе MS-DOS с UMSDOS, которая не описана здесь (прим. переводчика: по моим наблюдениям, вообще нигде не описана). В то время как Linux имеет эмуляторы DOS и может читать и даже выполнять некоторые программы DOS, DOS не может видеть, что находится на разделе Linux.

Если Вы хотите оставить MS-DOS и работать то с ней, то с Linux, прикиньте сколько места надо под нее, и вычтите его из общего объема диска. Результатом будет то, сколько можно отдать Linux.

CD-ROM обычно имеют интерфейс IDE/ATAPI, но встречается небольшое количество на SCSI; есть старые приводы на собственных интерфейсах. Если у Вас IDE или SCSI привод, все нормально. А вот если у Вас уникальный привод со своим неповторимым интерфейсом, запишите его марку и модель.

2.1.8 Диски под Linux.

Под Linux, как и под другими UNIX, **устройство** всего лишь специальный файл. Жесткие диски обрабатываются как файлы и имеют имена, как модемы, монитор и прочее оборудование. UNIX обрабатывает их как файлы, пригодные для чтения и записи. Все такие файлы сгруппированы в каталоге `/dev`.

Даже с учетом того, что отношение к устройствам такое же, как к файлам, они обрабатываются особо. Устройства с точки зрения Linux бывают двух типов: **блочные** и **символьные**, что отражает специфику общения с устройством: блоками или отдельными символами. При установке системы, все нужные устройства создаются автоматически.

Соглашения о наименовании устройств обсуждаются ниже.

2.1.9 Установка X Window System.

Для установки и настройки **graphical user interface** (GUI) потребуется знать тип видеокарты и монитора. Многие программы установки ставят графику после установки основных пакетов программ, так что данная информация может понадобиться не сразу.

Вы должны знать производителя и модель видеокарты, ее чипсет и объем видеопамяти. Про монитор надо знать частоты вертикальной и горизонтальной развертки и максимально поддерживаемое им разрешение.

Для настройки иксов также потребуется информация о мышке. Какая именно, сказано здесь.

2.1.10 Сетевое оборудование.

Если Вы имеете **network interface card** (NIC), системы Ethernet, token ring или другой, ознакомьтесь с данным разделом.

При установке Linux можно обойти сетевую часть, если Вы не имеете NIC. Но под Linux каждый компьютер должен иметь хост-имя.

Если Вы имеете модем, узнайте к какому порту он присоединен. Устройства с `/dev/ttyS0` по `/dev/ttyS3` соответствуют MS-DOS `COM:` портам 1-4. ISDN обрабатывается точно так же, но обычно выполняется дополнительная настройка со специальными, многократными, обозначениями устройств.

Теперь о памяти (RAM). Linux нормально работает на системах с 4 MB RAM, но лучше побольше. Рекомендуется 16 MB RAM.

2.1.11 Планирование, часть 2.

Советы в данном разделе (особенно по разделению диска на разделы) весьма спорны. В сообществе Linux installers единого мнения по ним нет. Тем не менее я изложу здесь свои соображения. Ваше дело использовать их или нет. Многое зависит от предполагаемых целей работы Linux-компьютера.

2.1.12 Стратегия разбиения диска на разделы.

Закаленные пользователи Linux советуют сделать один раздел Linux, один своп-раздел и один загрузочный раздел. Есть много причин для этого, но меня впечатляет одна: наступит день, когда Вы будете обновлять систему. Обновление с ядра 0.99 до 1.2.13 потребовало переформатирования раздела, как и обновление с 1.2.13 до 2.0.0, и я подозреваю, что обновление до ядра 2.2.0 будет таким же. Чего я не люблю, так это терять файлы, которые я накопил в моем основном каталоге. Да, я имею копию. Но сохранение моего каталога `/home` неповрежденным проще, тем более, что я переместил все мои специальные файлы туда в подкаталог. Пока что в дистрибутивах почти нет средств для простого обновления, и такое обновление сводится к переустановке системы. Такие средства появляются, но процесс медленный.

Другой причиной по которой надо создать несколько разделов является то, что загрузочный раздел должен располагаться в пределах первых 1024 дорожек жесткого диска. Такое ограничение накладывается многими версиями BIOS (Basic Input/Output System), которые видят только первые 1024 дорожки при запуске.

Я собираюсь описать стандарт файловой системы Linux, и как Linux обрабатывает разделы.

Под MS-DOS каждый раздел соответствует отдельному диску, и невелика разница между тем, является ли он физическим диском или логическим диском (разделом). Под Linux, физические и логические диски намного менее твердо обозначены.

При установке, Вы должны выбрать раздел, который будет корневым. Он обозначен как ``/''. Когда говорим о ``/dev'', это действительно два каталога: ``/'' и ``dev''. Ваше ядро Linux будет размещено в корневом разделе, но оно может лежать в подкаталоге, пока подкаталог находится в корневом разделе. Например, многие дистрибутивы используют каталог `/boot` для хранения ядра, системной карты и загрузочных файлов.

Следующая структура (как минимум) будет установлена в Вашем корневом разделе при установке:

```
/
|--bin
|--dev
|--etc
|--home
|--lib
|--lost+found.
|--proc
|--root
|--sbin
|--usr
|--var
```

Могут быть созданы `/boot`, `/mnt`, `/cdrom`, `/floppy`, `/opt`, но выше приведен минимальный необходимый набор каталогов.

Что относительно других разделов? Linux может использовать имя каталога (например, `/usr`) как **mount point (точку монтирования)**. То есть другой раздел на диске (или на другом диске) будет доступен как каталог (в данном случае `/usr`).

Если раздел не смонтирован, Вы ничего не увидите в каталоге монтирования. При монтировании раздела, там появятся каталоги и файлы с соответствующего раздела. Например, если Вы имеете два диска, один на 120 MB и другой на 840 MB, Вы можете создать один раздел на 120 MB диске (Например, корневой раздел) и смонтировать любой раздел с 840 MB диска в соответствующий каталог (по одному каталогу на раздел). При этом Вы создадите одну файловую систему размером в 960 MB.

Одно ограничение: Вы не можете использовать некоторые каталоги на корневом диске как точки монтирования потому, что они содержат файлы, которые необходимы, чтобы или загрузить систему или монтировать другие системы. Очевидно, что если команда, используемая, чтобы смонтировать другие разделы, размещена на другом разделе, и Вы не можете обращаться к этому разделу, пока Вы его не смонтировали, Вы будете подобны собаке, преследующей хвост.

Каталоги, которые НЕЛЬЗЯ использовать в качестве точек монтирования: `/bin`, `/dev`, `/etc`, `/lib`, `/lost+found`, `/proc`, `/root` и `/sbin`.

Детальное описание содержимого этих каталогов дано здесь.

Давайте рассмотрим маленький пример. Вы интернет провайдер. Вы имеете четыре машины, каждая имеет диск в 1 гигабайт. Вы решаете распределить место так:

```
machine A = 120MB
/usr = remainder of drive (exported)
/home = 0 - mount point (mounted from B)
/var/news = 0 - mount point (mounted from C)
/var/spool/mail = 0 - mount point (mounted from D)

machine B = 120MB

/usr = 0 - mount point (mounted from A)
/home = remainder of drive (exported)
/var/news = 0 - mount point (mounted from C)
/var/spool/mail = 0 - mount point (mounted from D)

machine C = 120MB

/usr = 0 - mount point (mounted from A)
/home = 0 - mount point (mounted from B)
/var/news = remainder of drive (exported)
/var/spool/mail = 0 - mount point (mounted from D)

machine D (reader exercise)
```

Вы вероятно отметили, что я произвольно назначил 120 МБ корневому разделу и распределил остальное место под `/usr`, `/home`, `/var/spool/mail` и прочее. Я не выделил места под своп-раздел.

Прежде чем начать изучение полученных результатов, опишу свой домашний компьютер. Он имеет два диска: `/dev/hda` (1.2 GB) и `/dev/hdb` (540 MB). Команда `df` (disk free) отображает:

```
File system   1024-blocks   Used    Available Capacity Mounted, on

/dev/hda1     150259         69605  72894       49%    /
/dev/hda3     723923        615452  71075       90%    /usr
/dev/hda2     150291         93326  49204       65%    /usr/X11R6
/dev/hdh1     499620        455044  18773       95%    /home
```

Как видите, я наполовину занял корневой раздел в 150 MB (`/`), раздел `/usr` почти полон, причем основное его содержимое приходится на `/usr/X11R6`. К тому же есть большой, но тесный раздел `/home` на 500 MB. Остаток от диска `/dev/hdb` отдан своп-разделу.

Как реалистичный минимум я предложил бы 80-100 МБ для корневого раздела, приблизительно 10 МБ на пользователя в разделе `/home`, как можно больше места для свопа (подробности чуть позже), а остальное отдать разделу `/usr`. Я имею систему с пятью пользователями дома, но я лично имею более чем 400 МБ в каталоге `/home`, в основном забитого графикой: семейными фотоальбомами и фотографиями друзей. Ваш раздел `/usr` должен, вероятно, быть по крайней мере 250 МБ, но минимум будет зависеть от того, что Вы решите устанавливать. Как Вы можете видеть, он может быстро заполниться более чем 800 МБ программ, библиотек и данных. Также не

забудьте, что разделы дают Вам гибкость управления, которую Вы теряете с одним гигантским разделом.

2.1.13 Раздел свопа.

Вы должны выделить место под своп-раздел. В отличие от Microsoft Windows, Linux использует для ускорения работы специальный своп-раздел. Хотя можно создать файл подкачки, это не рекомендуется. Linux может использовать раздел свопа объемом до 128 МБ. Я рекомендую практический минимум 16 МБ. Оптимум, вероятно, между 32 и 64 МБ: чем больше, тем лучше.

Одно последнее замечание прежде, чем Вы решите как лучше всего нарезать диск. Не забудьте, что BIOS не может видеть дальше 1024 дорожки на жестком диске (около 512 МБ). Linux ядро (файл, вероятно названный `vmlinuz` на вашем диске начальной загрузки), или любое ядро OS в этом отношении должно полностью расположиться на одном из первых двух дисководов (`/dev/hda` или `/dev/hdb`) и в пределах первых 1024 дорожек, или BIOS будет неспособна загрузить его. Спланируйте корневой раздел (также как любой другой раздел начальной загрузки) так, чтобы он оказался полностью внутри этого ограничения на первом или втором жестком диске.

2.1.14 Разбиение на разделы.

Предположим, что Вы хотите иметь на машине вместе MS-DOS с Microsoft Windows и Linux. В настоящий момент они стоят, и весь диск распределен как один большой раздел MS-DOS.

Так или иначе, мы будем иметь две операционных системы на этом компьютере. Linux удобно везде, где Вы поместите его. Ваш BIOS не сможет загрузить систему, но однажды запущенная, она не будет жаловаться, что находится на четвертом разделе четвертого жесткого диска. А вот MS-DOS и Microsoft Windows будут. Они хотят первый диск и первый раздел и могут отказываться загружаться из любой другой позиции. Я видел начальную загрузку MS-DOS из первого раздела на втором жестком диске, но первый жесткий диск не имел разделов MS-DOS, так что MS-DOS не распознавала диск. Самая лучшая стратегия часто путь наименьшего сопротивления. Если возможно, поместите MS-DOS на первом диске и первом разделе.

Вторая проблема: какая из нескольких OS загрузится первой. Если Вы хотите сначала поставить Linux (оставив `/dev/hda1` для MS-DOS), а потом поставить MS-DOS, не делайте так!. Windows 95, да и вообще все ОС от Microsoft, удалят любой предыдущий загрузчик, который хранится в master boot record (записи, которая используется BIOS для указания на загружаемое ядро). Вы могли даже слышать про это как про ``вирус Microsoft''. Это не вирус в истинном смысле слова, а только высокомерие со стороны Microsoft, которая хотела бы, чтобы загрузилась ТОЛЬКО операционная система Microsoft. Linux не вызывает такие проблемы, и фактически обеспечивает способ выбрать заданный по умолчанию образ начальной загрузки. Это также позволяет Вам вмешиваться в течение процесса начальной загрузки, чтобы определить какая операционная система загрузится. Это стандартная часть процедур установки Linux.

2.1.15 Резервирование старой системы.

Прежде чем начнете работать с таблицей разделов, зарезервируйте все Ваши данные!!! (Прим. переводчика: я однажды уверовал в "неразрушающее" перераспределение разделов, и в результате лишился своих работ по математическому анализу, что привело к тому, что я все лето сдавал экзамен вместо отдыха.)

Зарезервировав данные с жесткого диска, создайте загрузочную дискету. В MS-DOS введите:

```
C:> format a: /s
```

Данная команда отформатирует дискету и поместит на нее системные файлы. Если дискета уже отформатирована, наберите:

```
C:> sys a:
```

После создания загрузочной дискеты и проверки ее работоспособности, скопируйте из MS-DOS системы на загрузочную дискету файлы: `FDISK.EXE`, `SCANDISK.EXE` и `SYS.COM`. Также скопируйте туда файл `RESTORRB.EXE` с дистрибутива Linux или с Linux FTP archive. (См. приложение B).

Дефрагментируйте раздел DOS. Если дефрагментатор выявил ошибки, выполните программу `SCANDISK.EXE` для их исправления. Когда файлы окажутся собраны в начале диска, можно запускать `FIPS.EXE` для неразрушающего изменения размера раздела MS-DOS.

2.1.16 `FIPS.EXE`

На дистрибутивном Linux CD (или на сайте в Internet), можно найти программу `FIPS.EXE`, которая может менять размер раздела MS-DOS. `FIPS.EXE` Работает только с разделами MS-DOS. Если нужно работать с разделами других типов, может помочь программа Partition Magic, но она не бесплатна. Скопируйте `FIPS.EXE` на загрузочную дискету и загрузитесь с нее. Таким образом Вы загрузитесь в MS-DOS Real Mode и не будете использовать Microsoft Windows, что важно для работы данной программы.

На подсказку `A:>`, введите `FIPS` (большими или маленькими буквами). После приветствий Вам будет задан вопрос с каким диском (если их несколько) предполагается работать. Выберите диск. Как только Вы подтвердите выбор, позвольте `FIPS.EXE` сделать копию boot и root секторов на дискету на случай, если что-то пойдет не так.

Затем будет задан вопрос использовать ли все свободное пространство на разделе, чтобы создать второй раздел. Если Вы ответите ``yes", то у Вас не останется свободного места на разделе MS-DOS, в случае ответа ``no" Вы сможете задать сколько именно места кому достанется. Имейте в виду, что если диск не дефрагментирован, Вы не сможете использовать часть свободного места (то, которое оказалось не в конце диска). Если использовано программное обеспечение MS-DOS для зеркалирования, созданный им файл будет помещен в самый конец раздела, тогда `FIPS.EXE` сообщит, что места на разделе и вовсе нет. Устраните проблему (файл `MIRROR.FIL`) и перезапустите `FIPS.EXE`.

Вы можете менять таблицу разделов до достижения требуемого результата. Как только он будет достигнут, подтвердите изменения и запишите таблицу на диск.

После завершения `FIPS.EXE`, выньте все из дисководов и перезагрузитесь. В нашем примере мы уничтожим и создадим заново второй раздел для создания как минимум двух разделов для Linux: раздел свопа и собственно раздел Linux. Но Вы можете сделать все по-своему.

2.1.17 Подготовка к загрузке Linux.

Чтобы устанавливать Linux, мы должны начать с загрузки ядра Linux. Это выполняется точно так же, как будто Вы хотели перезагрузить MS-DOS: нужен boot-диск. Но основная масса дистрибутивов поставляются только на CD-ROM или доступны по FTP. Команда для создания дисков начальной загрузки в Linux отличается от своего аналога в MS-DOS. Если Вы купили новый компьютер с возможностью загрузки с CD-ROM, некоторые дистрибутивы позволяют Вам загружаться этим способом. Но мы пройдем процесс создания диска начальной загрузки для остальной части пользователей.

2.1.18 Создание boot-диска Linux под DOS.

Каждый дистрибутивный CD имеет MS-DOS программу, которая записывает образ диска на отформатированную дискету. Вы должны иметь дискету high density, и некоторые дистрибутивы требуют, чтобы она была именно 3.5 дюйма, 1.44 МБ. Вставьте дискету в дисковод. На CD (или на другом диске, если Вы скачали дистрибутив) найдите `RAWRITE2.EXE` (Вы можете иметь старый `RAWRITE.EXE`).

Потом `cd` в каталог, где лежат образы дисков. Может иметься только один образ, или много, которые сконфигурированы для различных аппаратных средств. Вы должны ознакомиться с дистрибутивной документацией. Запуск `RAWRITE2.EXE` без параметров приведет к запросу им пути к файлу образа и дисковода-адресата (`A:` или `B:`).

```
C:> rawrite diskimage drive
```

Повторите данный шаг для всех дополнительных дисков, если таковые есть.

Перед записью образа неплохо бы проверить дискету, например программой `SCANDISK.EXE`. Немало проблем при установке связаны с плохими дискетами, а `RAWRITE2.EXE` их не проверяет.

Все сказанное верно и для создания загрузочного диска и под Linux. man-страница `badblocks(1)` описывает как проверять диски.

2.1.19 Создание boot-диска Linux под Linux.

Вы можете создать загрузочный диск и под Linux; например, если Вы обновляете систему и Вам нужно создать образ диска, перейдите в нужный каталог с образами и введите:

```
# dd if=diskimage of=boot_floppy_device bs=512 conv=sync; sync
```

Подставьте имя образа диска вместо *diskimage*, впишите имя дисковода (почти всегда `/dev/fd0`) и повторите процедуру для каждого диска, в котором Вы нуждаетесь. Аргументы `dd`: `if` для входного файла; `of` для выходного файла, здесь мы хотим использовать дисковод; `bs` задает размер блока, в нашем случае 512 байт; `conv=sync` гарантирует, что выходной файл имеет точно тот же самый размер как и входной файл. Последний ``sync'' обеспечивает сброс буфера на диск немедленно.

Альтернативой является команда `cp` (copy):

```
cp diskimage boot_floppy_device: sync
```

Опять же, подставьте имя файла образа диска вместо *diskimage* и правильное *boot floppy device*, после чего повторите процесс для всех нужных дисков.

Теперь Вы готовы к установке системы. Большинство дистрибутивов включает Linux-версию `fdisk`, которой можно создать раздел Linux и своп-раздел. Программа установки предложит создать файловую систему (аналог форматирования в MS-DOS) на всех разделах Linux и свопа, смонтировать раздел Linux и активизировать своп.

Будет выдан запрос на проверку диска для поиска плохих блоков. Если Вы работаете со SCSI диском, ответьте ``no''. SCSI имеет встроенную систему контроля и отлова ошибок. IDE такой системы не имеет, так что нужна карта плохих блоков. На старом диске проверка нужна. При ответе ``yes'' программа установки выполнит `badblocks` для построения карты всех плохих блоков диска.

2.1.20 Разбиение жесткого диска: `fdisk` и `cfdisk`.

Каждая ОС имеет свою версию программы `fdisk`. Если нужно создать раздел MS-DOS, используйте версию `FDISK.EXE` для MS-DOS. Если создается раздел для Linux, Вы должны использовать версию `fdisk` для Linux.

Под Linux есть две программы для разбиения диска на разделы: обычный `fdisk` и дружественный `cfdisk`. Разница в том, что в `fdisk` все команды вводятся с клавиатуры как числа и символы, а в `cfdisk` используются клавиши со стрелками для выбора пунктов меню и нажимается Enter для выполнения команд. Числа вводятся только при указании размера раздела.

Перезагрузитесь с boot-дискеты. После приветствия вы увидите запрос:

```
LILO boot:
```

и мигающий курсор. Если нажмете Tab, увидите список имен. Имена зависят от дистрибутива, но там наверняка есть пункты ``rescue'' или ``expert''. Пункт ``install'' запустит после загрузки ядра программу установки. Можно передать ядру при загрузки какие-либо параметры.

Введите нужное имя и нажмите Enter. Когда ядро Linux загрузится, Вы увидите приглашение, которое зависит от дистрибутива.

Когда приглашение появится, Вы окажетесь в системе как ``root'' (регистрация пока не нужна). Подробности см. в главе 4). Введите команду:

```
# fdisk
```

Если получено сообщение об ошибке, попробуйте `cfdisk`. По умолчанию работа ведется с `/dev/hda`, но если надо работать со вторым жестким диском, введите команду:

```
# fdisk /dev/hdb
```

В `fdisk` нажмите m, чтобы увидеть меню. Команды: n создает новый раздел; d удаляет раздел; t меняет тип раздела (83 Linux, 82 Linux Swap); p выводит текущую информацию о разделах, которая не записана на диск; w записывает ее на диск; q завершает программу.

Пока не введена команда w, можно вносить любые изменения. Можно выйти из программы без сохранения изменений на диске.

Размер разделов задается числами с буквой ``k'' или ``M'' (регистр не важен) после числа, для указания KB или MB.

Вы можете создать до четырех первичных разделов. Если нужно более четырех, создайте три первичных и сколько надо расширенных. Расширенные нумеруются с 5, так что будет `/dev/hda1`, `/dev/hda2`, `/dev/hda3`, `/dev/hda5` и `/dev/hda6`, если нужно пять разделов.

Проверьте не пересеклись ли разделы. Разделы, которые выходят за пределы 1024 "магической" дорожки поймет далеко не всякий BIOS, так что загружаться с них не получится.

`cfdisk` делает то же самое, что и `fdisk`, но всегда отображает на экране состояние таблицы разделов в памяти (но не на диске!). Используйте клавиши Up и Down для выбора обрабатываемого раздела и клавиши Left и Right, чтобы выбрать действие, которое нужно выполнить. Затем нажмите Enter, чтобы выполнить действие. Вы будете должны ввести число, задающее размер раздела. `cfdisk` по умолчанию работает с `/dev/hda`, так что надо указать параметр `/dev/hdb` если надо сменить таблицу разделов на втором жестком диске. Не забудьте записать таблицу на диск прежде, чем закончите работу. Главная проблема с `cfdisk` в том, что перед выходом он не спрашивает подтверждения. Приказы пользователя не обсуждаются, они выполняются! Так что выберите Write и нажмите Enter перед выбором Quit и нажатием Enter.

2.2 Дистрибутивы Linux.

Теперь надо понять какой из дистрибутивов Linux (а их много) подходит Вам лучше всего. Дистрибутивы разные: одни имеют много программ на все случаи жизни, другие не имеют почти ничего. Самый маленький, известный мне, дистрибутив DosLinux (занимает около 10 МБ), а самый большой S.u.s.e Linux (занимает 6 дисков CD-ROM).

Linux *Distribution HOWTO* (см. приложение A) содержит полный перечень дистрибутивов Linux, доступных в интернет или по почте.

Если Вы имеете доступ к USENET, или другой системе конференций, Вы можете поспрашивать там персональное мнение людей, которые установили Linux. Полезную информацию можно также получить в *Linux Journal*, который периодически делает обзоры наиболее популярных дистрибутивов (ищите на `http://www.linuxjournal.com/selected.html` on-line версии сравнительной таблицы дистрибутивов). Многие дистрибутивы содержат примерно одинаковый комплект программ, так что во многом выбор произволен.

2.3 Debian GNU/Linux.

Раздел про Debian GNU/Linux написан Boris Beletsky.

2.3.1 Установка Debian GNU/Linux.

```
Dependencies:                   yes
Install boot methods:           floppy
Install methods:                CD, hard disk, NFS, FTP
System initialization:          Sys V init
Easy of installation:           challenging
Graphical administration tools: no
Installation utility:           dselect
Package maintenance utility:    dselect/dpkg
```

2.3.2 Получение образов дискет.

При наличии быстрого доступа в интернет лучший путь установить Debian это скачать его с anonymous FTP (см. приложение B). Основной ftp сайт Debian находится на `ftp.debian.org` в каталоге `/pub/debian`. Структура архива Debian описана в таблице.

Directory	Contents
./stable/	Latest stable Debian release.
./stable/binary-i386	Debian packages for Intel 386 architecture.

./stable/disks-i386	Boot and root disks needed for Debian installation.
./stable/disks-i386/current	The current boot floppy set.
./stable/disks-i386/special-kernels	Special kernels and boot floppy disks, for hardware.
./stable/msdos-i386	Configurations that refuse to work with our regular boot floppies. DOS short file names for Debian packages.

Таблица 2.1: Структура архива Debian GNU/Linux.

Для основной установки Debian понадобится около 12 мегабайт места на диске и соответствующее количество дискет. Для начала понадобятся образы boot и driver дискет. Debian предоставляет два набора образов boot дискет, для 1.2 и 1.44 Mb дисков и один набор образов, которые работают на всех типах дискет. Посмотрите каким приводом оснащена для загрузки Ваша система и загрузите подходящий набор.

File Name	Label	Description
rsc1440.bin	Rescue Floppy	Floppy set for systems with 1.44MB floppy drive and at least 5MB RAM.
drv1440.bin	Device Drivers	
base-1.bin	Base 1	
base-2.bin	Base 2	
base-3.bin	Base 3	
base-4.bin	Base 4	
base-5.bin	Base 5	
root.bin	Root Disk	
rsc1440r.bin	Rescue Floppy	Optional Rescue Disk: image for low memory systems (less then 5MB of RAM).
rscl200r.bin	Rescue Floppy	Floppy set for systems with 1.2MB floppy drive.
drvl200.bin	Device Drivers	
base-1.bin	Base 1	
base-2.bin	Base 2	
base-3.bin	Base 3	
base-4.bin	Base 4	
root.bin	Root Disk	

Таблица 2.2: Инсталляционные дискеты Debian GNU/Linux.

Выберите подходящий для Вашего оборудования набор дискет из таблицы и запишите образы на дискеты, как сказано выше.

2.3.3 Загрузка пакетов.

После установки Debian, скачайте файл `Packages` с `ftp://ftp.debian.org/pub/debian/stable/Packages`.

Файл хранит текущий список пакетов Debian доступных в стабильном дистрибутиве Debian. Файл имеет свой формат; каждый пакет имеет запись, отделенную от остальных пустой строкой. Информация о пакете разделена на несколько полей. В таблице описаны поля и их возможные значения. Такая информация служит материалом для построения своего списка загружаемых пакетов. После его построения, скачайте пакеты. Можно скачать вообще все и инсталлировать потом с жесткого диска, смонтировав соответствующий раздел.

2.3.4 Загрузка с дискет и установка Debian GNU/Linux.

Дискета Rescue.

Вставьте в загрузочный дисковод дискету Rescue и перезагрузитесь с нее. Через минуту или около того, Вы увидите введение в дискету Rescue и запрос `boot`.

Дискета названа *Rescue* потому, что с нее можно загрузиться в случае проблем с загрузкой с жесткого диска. После установки сохраните эту дискету.

На приглашение `boot:` можно отреагировать двумя способами: нажать клавиши от F1 до F10 для просмотра справки или загрузить систему. Если Вы имеете аппаратуру, которая некорректно работает при загрузке, можно указать параметры в командной строке загрузчика.

Для этого наберите слово ``linux" и введите пробел перед первым параметром. Простое нажатие Enter равнозначно вводу слова ``linux" без дополнительных параметров.

При первой загрузке системы нажмите Enter и посмотрите все ли работает должным образом. Если нет, позже можно перезагрузиться с указанием параметров, которые надо поискать в информации по оборудованию.

После нажатия Enter Вы увидите сообщение:

```
Loading
Uncompressing Linux
```

Затем отображается информация о Вашем оборудовании. Может быть много сообщений вида: ``can't find *что-то*", ``*что-то* not present", ``can't initialize *что-то*" или ``this driver release depends on *что-то*". Все нормально:

инсталляционный диск сделан для систем с самым разным оборудованием, так что нечего удивляться, что он чего-то не нашел. Позже можно будет создать ядро системы в котором нет драйверов для отсутствующего оборудования.

Если мало памяти.

Если в системе 4 MB RAM, вы получите предупреждение и текстовое меню. Если памяти больше, но она не видна целиком (бывает такое), спокойно продолжайте установку в текстовом режиме. Разметьте диск, запустите своп и переходите к графическому интерфейсу.

`cfdisk` используется для создания своп-раздела на диске (тип 82). Он нужен для того, чтобы программе установки хватило памяти. Задайте объем виртуальной памяти, которую Вы предполагаете использовать, как только ваша система установится. Это точно равно количеству требуемого дискового пространства. Шестнадцать мегабайт, вероятно, самое маленькое допустимое значение, но используйте 32 мегабайта, если Вы можете их выделить, или 64 мегабайта, если диск достаточно большой, и Вы не будете жалеть места.

Диалог "color or monochrome".

Как только система заканчивает загружаться, Вы должны видеть цветное или монохромное диалоговое окно. Если ваш монитор черно-белый, нажмите Enter и продолжите установку. В противном случае используйте клавишу курсора, чтобы переместить курсор в пункт меню `Color` и нажмите клавишу Enter. Дисплей должен перейти в цветной режим. Нажмите Enter и продолжите установку.

Главное меню

Вы можете увидеть диалоговое окно:

```
The installation program is determining the current state of vour system.
```

На некоторых системах это сообщение исчезает слишком быстро, чтобы его прочитать. Оно отображается между шагами в процесс установки. Программа установки проверяет состояние системы после каждого шага. Это позволяет Вам перезапускать установку без потерь работы, которую Вы уже сделали, если Вы останавливаете систему в середине установки. Если Вы должны перезапустить установку, Вам придется выбрать цветной или одноцветный режим, сконфигурировать клавиатуру, повторно активизировать своп-раздел и повторно перемонтировать любые диски, которые были инициализированы. Любая другая установка в системе будет сохранена.

Весь процесс установки управляется из главного меню. С его помощью можно и цыпленка научить ставить Debian (прим. переводчика: звучит сомнительно). Строго говоря, установка во многом сводится к клавише Enter. Обычно надо выбирать пункт `Next`, который выбран по умолчанию.

Настройка клавиатуры.

Удостоверьтесь, что выбран именно пункт `Next` и нажмите Enter. Выберите клавиатуру, которая соответствует размещению, используемому для вашего национального языка,

или выберите что-то близкое к этому, если расположение символов на клавиатуре, которое Вы хотите, отсутствует. После установки Вы можете выбирать расположение символов на клавиатуре из более широкого диапазона. Поскольку клавиши управления курсором независимы от национального языка, используйте их для выбора раскладки клавиатуры.

Shell.

Если Вы опытный пользователь UNIX или Linux, нажмите LeftAlt и F2 для перехода на вторую консоль. Там будет Bourne-подобная оболочка `ash`. В данный момент корневая файловая система находится на RAM-диске и имеется ограниченный набор UNIX утилит, доступных для использования. Вы можете посмотреть доступные команды:

```
# Is /bin /sbin /usr/bin /usr/sbin
```

Оболочка с командами понадобятся только в случае, если что-то пошло не так. Вы должны всегда использовать меню, а не оболочку чтобы активизировать своп-раздел, потому что программное обеспечение меню не может обнаружить, сделали ли Вы это из оболочки. Нажмите LeftAlt-F1, чтобы вернуться к меню. Linux обеспечивает до 64 консолей (прим. переводчика: хотя я не очень представляю, как между переключаться), но дискета Rescue использует только несколько из них.

Последний шанс! Вы зарезервировали данные? Мы подошли к первой точке их возможной потери и последнему шансу сохранить старую систему. Если резервной копии нет, выньте дискету, перезагрузитесь и зарезервируйте данные.

Разбиение жестких дисков.

Если Вы еще не создали разделы для Linux, пункт меню `Next` приведет к следующему:

```
Partition a Hard Disk
```

А если создали, то к следующему:

```
Initialise and Activate the Swap Disk Partition.
```

Данный шаг можно пропустить, если есть проблемы с памятью и своп-раздел используется с самого начала установки. При выборе пункта `Next`, Вы всегда сможете использовать клавишу "стрелка вниз" для выбора:

```
Partition a Hard Disk
```

Пункт меню `Partition a Hard Disk` предоставляет список дисков и запускает программу `cfdisk` (см. выше), которая позволяет создавать и редактировать разделы диска. Вы должны создать как минимум один раздел Linux (тип 83).

Своп-раздел может использоваться под виртуальную память и иметь размер от 16 до 128 мегабайт, в зависимости от Ваших возможностей по выделению дискового пространства. Linux не использует под своп более 128 мегабайт, так что нет смысла

делать своп больше данного значения. Своп-раздел строго рекомендуется, но можно и без него обойтись, если в системе более 16 Mb RAM.

Активизация своп-раздела.

Это следующий (`Next`) элемент меню после того, как Вы создадите один дисковый раздел. Вы имеете выбор: инициализация и активизация нового своп-раздела, активизация предварительно инициализированного раздела и выполнение без своп-раздела. Всегда допустимо заново инициализировать раздел, так что выберите `Initialize and Activate the Swap Disk Partition`, если Вы не уверены, что Вы знаете, что делаете. Этот пункт предоставит Вам опцию проверки раздела на плохие блоки, вызванные дефектами на поверхности диска. Это полезно, если Вы имеете MFM, RLL, или старые IDE диски, и проверка диска никогда не причиняет вред. Правильно работающие SCSI диски можно и не проверять: они имеют собственный внутренний механизм для обхода плохих дисковых блоков.

Своп-раздел обеспечивает виртуальную память в дополнение к физической, что используется даже при установке. Именно поэтому мы сразу запускаем своп.

Инициализация раздела Linux.

На этом этапе пункт `Next` должен выглядеть так:

```
Initialize a Linux Disk Partition
```

Если это не так, Вы не завершили процесс выделения разделов диска, или Вы не запустили своп-раздел.

Вы можете инициализировать раздел Linux или смонтировать предварительно инициализированный раздел.

Загрузочные дискеты не предназначены для обновления системы: Debian предоставляет для этого другой способ. Так что если используются старые разделы, на которых что-то есть, надо инициализировать их заново, что сотрет с них данные. Единственной причиной смонтировать раздел, не инициализируя его, может быть наличие на данном разделе файлов, которые хотелось бы сохранить, например, каталога `/home`. Вы должны инициализировать любой раздел, который Вы создали на шаге разбиения диска.

Выберите пункт `Next` для инициализации и монтирования корневого раздела (каталог ``/''). Первый раздел, который Вы инициализируете и смонтируете после своп-раздела (если он используется), будет корневым. Вам будет предложена возможность проверить его на плохие блоки, что никогда не повредит. Имейте в виду, что это может занять минут 10 или больше, если Вы имеете большой диск.

Установка основной системы.

После монтирования корневого раздела пункт `Next` будет таким:

```
Install the Base System
```

если Вы еще не выполнили некоторые шаги установки. Вы можете использовать клавиши курсора, выбрать элементы меню, чтобы инициализировать или смонтировать разделы, если Вы имеете дополнительные разделы. Если Вы создали отдельные разделы для `/var`, `/usr` или других файловых систем, Вы должны инициализировать и смонтировать их сейчас.

Теперь последует пауза, пока установщик ищет локальную копию основной системы. Поиск выполняется для CD-ROM версий и закончится провалом. Будет выведено меню дисков, чтобы установить откуда читать основные дискеты. Выберите соответствующий диск. Устанавливайте дискеты Base 1, Base 2, Base 3, Base 4 и Base 5, если Вы используете дискеты на 1.2 MB, по мере их запроса установщиком. Если одна из основных дискет нечитабельна, Вы должны создать дискету для замены и подгрузить все пять дискет в систему снова. После того, как дискеты прочитались, система устанавливает файлы, что займет некоторое время.

Установка ядра.

На данном этапе меню `Next` будет выглядеть так:

```
Install the Operating System Kernel
```

Выберите это. Вам будет выдан запрос в какой привод вставлена Rescue дискета. Ядро будет скопировано на жесткий диск. Это ядро используется позже, чтобы создать дискету начальной загрузки для Вашей системы и сделать жесткий диск загрузочным без дискеты.

Установка драйверов устройств.

Выберите элемент меню, чтобы установить драйверы устройства. Вас попросят вставить дискету Device Drivers, и драйверы будут скопированы на жесткий диск. Выберите элемент меню:

```
Configure Device Driver
```

и посмотрите какие устройства есть в системе. Драйверы будут загружаться при каждом запуске системы.

Имеется пункт меню для драйверов PCMCIA-устройств, но Вы не должны его использовать. После установки Вы можете установить пакет `pcmcia-cs`. Он обнаруживает PCMCIA платы автоматически и конфигурирует те, которые находит. Он также распознает платы, которые поддерживают горячую замену.

Настройка основной системы.

На данном этапе система читает все файлы минимальной системы Debian, но надо сконфигурировать ее перед запуском. Выберите:

```
Configure the Base System
```

Появится запрос часового пояса. Выберите из меню часовой пояс, что может привести к второму меню с уточнением данных.

Затем надо указать настроены Ваши часы на (GMT) или локальное время. Если работаете только с Linux или другой UNIX, выберите GMT. А вот если работаете с MS-DOS или Microsoft Windows, выбирайте локальное время. UNIX хранит системное время в GMT формате и конвертирует в локальное по мере необходимости. Это позволяет им следить за летним и зимним временем и правильно обрабатывать високосные годы. Пользователи, вошедшие из других часовых поясов, могут индивидуально установить часовой пояс на их терминале.

Настройка сети.

Вы должны сконфигурировать сеть, даже если Вы не имеете ее. Вы должны только ответить на первые два вопроса:

```
What is the name of your computer?
Is your system connected to a network?
```

Если Вы связаны с сетью, узнайте следующее:

- хост-имя Вашей системы;
- доменное имя Вашей системы;
- IP-адрес Вашей системы;
- маску подсети;
- IP-адрес Вашей сети;
- широковещательный адрес Вашей сети;
- если используется шлюз, нужен его IP-адрес;
- Domain Name Service (DNS) Вашей сети;
- соединяетесь ли Вы с сетью Ethernet.

Программа установки предполагает, что адрес Вашей сети является побитовым AND IP-адреса Вашей системы и маски подсети. Также предполагается, что широковещательный адрес является побитовым OR IP-адреса Вашей системы и NOT маски подсети. Предполагается также, что шлюз совпадает с DNS сервером. Если Вы не можете найти любой из этих ответов, используйте предположения системы: в случае необходимости, Вы можете изменять их после установки, редактируя файл `/etc/init.d/network`.

Сделаем жесткий диск загрузочным.

Если выбрана загрузка с жесткого диска сразу в Linux, будет задан вопрос об установке master boot record. Если Вы не используете администратора начальной загрузки (это вероятно имеет место, если Вы не знаете что такое администратор начальной загрузки), ответьте ``yes''. Следующий вопрос: хотите ли Вы загружать Linux автоматически с жесткого диска, когда Вы включаете систему. Если на этот вопрос ответить ``no'', Вы сможете сделать раздел загрузочным позже с помощью MS-DOS программы `FDISK.EXE` или Linux `fdisk` или программы `activate`.

Создание загрузочной дискеты.

Вы должны сделать дискету начальной загрузки, даже если Вы предполагаете загружать систему с жесткого диска. Причина для этого: если что-то пошло не так, с жесткого диска Вы вряд ли загрузитесь. Дискета начальной загрузки будет почти всегда работать. Выберите:

```
Make a Boot Floppy
```

из меню и вставьте чистую дискету, как будет указано. Удостоверьтесь, что дискета не защищена от записи. Программное обеспечение попытается отформатировать и записать ее. Отметьте эту дискету как ``Custom Boot'' и защитите ее от записи как только закончите запись.

Момент истины.

Выньте дискету из дисковода и выберите:

```
Reboot the System
```

из меню. Если Linux не грузится, вставьте дискету Custom Boot, которая была недавно создана, и перезагрузитесь. Linux должна загрузиться. Вы должны увидеть те же самые сообщения, как когда Вы загрузились с установочной дискеты, сопровождаемые некоторыми новыми сообщениями.

Добавление пользовательских аккаунтов и паролей.

После добавления логинов (подробности в главе 4), можно запустить `dselect`, Debian-программу управления пакетами программ.

Вы должны почитать руководство перед попыткой установить пакеты, используя `dselect`.

`dselect` позволяет Вам выбирать пакеты, которые Вы хотите установить в Вашей системе. Подробно менеджер пакетов описан ниже. Если Вы имеете CD-ROM или жесткий диск с дополнительными Debian пакетами или связаны с Internet, Вы можете прочитать этот раздел сейчас. В противном случае выйдите из `dselect`. Вы можете использовать программное обеспечение управления пакетами после того, как переместите файлы пакетов Debian в Вашу систему.

Для использования `dselect` надо зайти в систему как root.

Если Вы поставили X Window System и не используете раскладку клавиатуры для US, прочтите X11 Release note для не-US клавиатур.

Вход в систему.

После выхода из `dselect` Вы увидите приглашение `login:`. Зайдите в систему с использованием персональных логина и пароля. Система готова к использованию.

2.3.5 Запуск Debian GNU/Linux.

Здесь описаны менеджер пакетов Debian и утилиты, специфичные для Debian. Debian/GNU Linux имеет файл `Packages`, формат которого описан в таблице ниже..

Priority (приоритет)	**Важность пакета**
Required	Требуется для нормального функционирования системы.
Important	Не является необходимым, но важный.
Standard	Выполняет отдельные функции.
Optional	Полезный, но не обязательный.
Extra	Этот пакет может конфликтовать с другими, имеющими более высокие приоритеты.

Section Function	**Название раздела**
Base	Базовая система.
Devel	Средства разработки.
X11	Пакеты для X Window System.
Admin	Утилиты для администрирования.
Doc	Документация.
Comm	Различные утилиты для коммуникаций.
Editors	Различные редакторы.
Electronics	Утилиты для электроники.
Games	Игры.
Graphics	Графические утилиты.
Hamradio	Утилиты для Интернет-радио.
Mail	Программы для электронной почты (клиентские и серверные).
Math	Математика (калькуляторы и т.п.).
Net	Сетевые программы (обычно TCP/IP).
News	Программы новостей из Интернета (NNTP) (клиентские и серверные).
Shells	Оболочки типа `tcsh`, `bash`.
Sound	Звуковые приложения (например, проигрыватели аудиодисков).
TeX	Все, связанное с TeX.
Text	Приложения для обработки текстовых файлов (например, `nroff`).
Misc	Все, что не подходит под указанные категории.

Maintainer	Имя того, кто осуществляет поддержку пользователей данного пакета и его электронный адрес.
Version	Версия пакета в формате *версия Linux-версия Debian.*
Depends	Список других пакетов, от которых данный пакет зависит и без которых не будет работать.

Recommends	Имена других пакетов, которые настоятельно рекомендуется установить, если данный пакет будет использоваться.
Suggests	Перечисляются пакеты, которые могут быть полезны для данного пакета.
Filename	Имя файла, содержащего пакет на CD или на FTP-сервере.
MS-DOS-Filename	Короткое имя файла (в формате MS-DOS).
Size	Размер установленного пакета.
Md5sum	Контрольная сумма md5sum (чтобы проверить, что получена официальная версия).
Description	Здесь описывается пакет. Эту информацию надо прочесть прежде, чем файлы будут скопированы.

Таблица 2.3: Поля в записях файла `Packages` для Debian/GNU Linux.

Дистрибутивы Debian поставляются в архивах, названных **packages (пакетами)**. Каждый пакет представляет собой набор файлов, который может быть установлен программами `dpkg` или `dselect`. Пакет также может содержать дополнительные данные для программ установки.

Классификация пакетов.

Пакеты, которые включены в Debian GNU/LINUX, классифицированы согласно тому, насколько необходимыми они являются (priority) и по их функциональным возможностям (section).

priority пакета указывает насколько он необходим. Debian GNU/LINUX классифицирует все пакеты на четыре различных приоритетных уровня:

Required.

Эти пакеты должны быть установлены, чтобы система работала, и были установлены как часть основной системы.

Никогда не удаляйте такой пакет из системы, если Вы не абсолютно уверены относительно того, что Вы делаете. Иначе система, скорее всего, просто перестанет работать вообще.

Такие пакеты помечены в `dselect` как `Req`.

Important.

Эти пакеты есть почти на всех UNIX-системах. Они включают `cron`, `man` и `vi`.

Такие пакеты помечены в `dselect` как `Imp`.

Standard.

Это пакеты, которые более-менее стандартно встречаются в системах Debian GNU/Linux. Стандартная система включает довольно полную программную среду разработки и GNU Emacs.

Такие пакеты помечены в `dselect` как `Std`.

Optional.

Опциональные пакеты включают довольно полную систему. Сюда входят TeX и X Window System.

Такие пакеты помечены в `dselect` как `Opt`.

Extra

Extra-пакеты нужны ограниченному кругу лиц и ставятся для специфической цели. Например, сюда входят пакеты сетевого радио.

Такие пакеты помечены в `dselect` как `Xtr`.

По умолчанию `dselect` автоматически выбирает Стандартную систему, если пользователь не хочет сам выбирать пакеты, которые будут установлены.

Категория *section* пакета показывает функциональные возможности или его использование. Пакеты на CD-ROM и в FTP архиве размещены в подкаталогах согласно функциям. Имена каталога довольно очевидны: например, каталог `admin` содержит пакеты для администрирования системы, а каталог `devel` содержит пакеты для программирования. В отличие от приоритетных уровней, там много разделов и еще больше может быть добавлен в будущем, так что мы индивидуально не описываем их в этом руководстве.

Зависимости пакетов.

Каждый пакет имеет данные о том, как он связан с другими пакетами. Есть четыре типа зависимостей в Debian GNU/Linux: conflicts, dependencies, recommendations и suggestions.

conflict значит, что несколько пакетов нельзя установить в системе вместе. Хороший пример противоречивых пакетов: mail transfer agents (MTAs). Это программы, которые доставляют электронную почту пользователям Вашей системы и других машин сети. Debian GNU/LINUX имеет таких программы: `sendmail` и `smail`.

Но только одна из может быть установлена в данный момент времени. Они написаны для одного и того же, и не рассчитаны на сосуществование. Следовательно, пакеты `sendmail` и `smail` конфликтуют. Если попробовать поставить `sendmail`, когда `smail` уже стоит, менеджер пакетов Debian GNU/Linux откажется его устанавливать. Аналогично, если Вы пробуете установить `smail` когда `sendmail` уже установлен, `dselect` (или `dpkg`; см. ниже) откажется его устанавливать.

dependency происходит, когда один пакет требует, чтобы другой пакет функционировал правильно. При использовании нашего примера электронной почты, пользователи

читают почту программами mail user agents (MUAs). Популярные MUA: `elm`, `pine` и `emacs` RMAIL mode. Нормально установить несколько MUA сразу, потому что они не передерутся. Но MUAS не доставляют почту: это является работой MTA. Так что все пакеты mail user agent packages *depend* от mail transfer agent.

Пакет может также *рекомендовать (recommend)* или *предполагать (suggest)* другие связанные пакеты.

2.3.6 dselect.

Здесь кратко описана программа Debian `dselect`. Подробностями можно разжиться по адресу `ftp.debian.org/debian/Debian-l.2.disks-i386-current/dselect/beginner/6.html`.

`dselect` просто, управляемый меню, интерфейс для установки пакетов. Процесс установки происходит при помощи экранного меню:

```
Debian Linux dselect package handling front end.

0   [A]ccess Choose the access method, to use
1   [U]pdate Update list of available packages, if possible
2   [S]elect Request which packages you want on your system
3   [I]nstall Install and upgrade wanted packages
4   [C]onfig Configure any packages that are unconfigured
5   [R]emove Remove unwanted software
6   [Q]uit Quit dselect
```

Имеются два способа выбрать опцию из меню: выберите ее стрелками, или нажмите клавишу соответствующего символа в скобках.

Access.

В этом меню Вы выбираете метод получения и установки пакетов.

Abbreviation	Description
cdrom	install from a CD-ROM
nfs	install from an NFS server (not yet mounted)
harddisk	install from a hard disk: partition (not yet mounted)
mounted	install from a file system which is already mounted
floppy	install from a file of floppy disks
ftp	install using ftp

Update.

`dselect` читает базу данных `Packages` (описана выше) и создает базу данных пакетов, доступных на Вашей системе.

Select.

Этот раздел программы выбирает пакеты. Выберите нужный пакет и нажмите Enter. Если Вы имеете медленную машину, экран может оставаться пустым в течение 15 секунд. Потом появится страница 1 справочного файла. Вы можете обратиться к справочнику, нажав "?" в любой точке экранов Selec и можно листать справочник нажатием клавиши . (точка).

Для выхода из Select после завершения всех выборов нажмите Enter. Вы вернетесь в главное меню *если* не было проблем с выбором. Сначала надо решить проблемы.

Конфликты зависимостей совершенно нормальны и ожидаются. Если Вы выбираете пакет, А и он пакет требует, чтобы невыбранный пакет В был, `dselect` предупредит Вас относительно проблемы и наиболее вероятно предложит решение. Если пакет А конфликтует с пакетом В, Вы будете делать выбор между ними.

Install

`dselect` проходит через все 800 пакетов и устанавливает те, которые выбраны. Вы будете должны принять решения в течение этого процесса. Часто полезно обратиться к оболочке, чтобы сравнить, например, старый файл конфигурации с новым. Если старый файл назван `conf.modules`, например, новый файл будет назван `conf.modules.dpkg-new`.

Экран прокручивается довольно быстро на быстрых машинах. Вы можете остановить отображение, нажав Control-S и продолжить его, нажав Control-Q. В конце будет выдан список любых неустановленных пакетов.

Configure.

Большинство пакетов конфигурируется на шаге 3, но некоторые и здесь.

Remove.

Удаляет пакеты, в которых уже нет необходимости.

Quit.

Завершает работу программы.

2.3.7 dpkg.

Это инструмент командной строки, который устанавливает и управляет Debian пакетами. Он имеет несколько параметров, которые позволяют Вам устанавливать, конфигурировать, модифицировать, удалять и выполнять другие операции над Debian-пакетами. Вы можете даже формировать собственные пакеты. `dpkg` также позволяет Вам вносить в список доступных пакетов, файлы "находящиеся в собственности" пакетов, то есть файлы, которыми обладает пакет.

Установка новых или обновление существующих пакетов.

Скомандуйте:

```
# dpkq -i filename.deb
```

где *filename* имя файла Debian-пакета, например `tcsh_6.06-11_i386.deb`. `dpkg` частично интерактивен; в течение установки он может задавать дополнительные вопросы, например, установить ли новую версию файла конфигурации или сохранить старую версию.

Вы можете просто распаковать пакет без настройки командой:

```
# dpkg --unpack filename
```

Если пакет зависит от неустановленного пакета, или Вы уже имеете более новую версию пакета, или если происходит любая другая проблема зависимости в течение установки, `dpkg` завершится без конфигурирования пакетов.

Настройка установленных пакетов.

Если `dpkg` аварийно завершился при установке, но пакет поставить успел, то пакет останется несконфигурированным. Менеджер пакетов Debian требует, чтобы пакет был сконфигурирован, чтобы избежать проблем зависимости. Некоторые пакеты также требуют, чтобы конфигурация работала правильно.

Для настройки пакета наберите:

```
# dpkg --configure package
```

где *package* имя пакета, например `tcsh`. Заметьте, что это не имя файла с пакетом!

Удаление установленных пакетов.

В менеджере пакетов Debian имеется два способа удалить пакеты: `remove (удаление)` и `purge (очистка)`. Опция `remove` удаляет определенный пакет; опция `purge` удаляет определенный пакет и файлы конфигурации. Использование:

```
# dpkg -r package
# dpkg --purge package
```

Если имеются любые установленные пакеты, которые зависят от того, который Вы желаете удалить, пакет *не будет удален*, и `dpkg` прервется с сообщением об ошибке.

Отображение статуса пакетов.

Для отображения статуса пакета (установлен, не установлен или не сконфигурирован) наберите:

```
# dpkg -s package
```

Список доступных пакетов.

Для отображения списка установленных пакетов, подходящих под заданный шаблон, наберите:

```
# dpkg -l package-name-pattern
```

где *package-name-pattern* необязательный аргумент, задающий шаблон для имени пакета, например `*sh`. Допустимы обычные символы подстановки из shell. Если шаблон не задан, выводится список всех пакетов.

Отображение файлов пакета.

Для вывода списка файлов, принадлежащих пакету, введите:

```
# dpkg -L package
```

Не будут отображены имена файлов, созданных скриптами установки пакета.

Поиск пакета, которому принадлежит файл.

Для поиска пакета, которому принадлежит файл неизвестного происхождения наберите команду:

```
# dpkg -S filename-pattern
```

где *filename-pattern* задает шаблон для поиска имен пакетов. Допустимы символы подстановки из shell.

Резюме.

`dpkg` проще использовать, чем `dselect` при небольших объемах работы с пакетами. Он также имеет некоторые функциональные возможности, которые `dselect` (интерфейс к `dpkg`) не имеет, например, выяснение какому пакету принадлежит файл. Полный список параметров есть на man-странице `dpkg(8)`.

2.3.8 Про Debian GNU/Linux.

Debian GNU/Linux Project создан Ian Murdock в 1993 году, первоначально при покровительстве проекта Free Software Foundation's GNU project. Позже Debian отделился от FSF. Debian является результатом усилий добровольца по созданию свободной высококачественной UNIX-совместимой операционной системы, основанной на ядре Linux с набором прикладных программ.

Debian community является группой из более чем 150 добровольцев со всех континентов, которые сотрудничают через Internet. Основатели проекта сформировали организацию Software in the Public Interest (SPI) для разработки Debian GNU/Linux.

Software in the Public Interest является некоммерческой организацией, созданной когда FSF прекратил спонсирование Debian. Цель организации состоит в том, чтобы разрабатывать и распространять свободное программное обеспечение. Цели очень подобны таковым FSF, и это поощряет программистов использовать GNU General Public License. Однако, SPI концентрирует усилия в несколько другом направлении: разработке и распространении системы, которая в мелких деталях отличается от GNU-системы, планируемой FSF. SPI поддерживает отношения с FSF по ряду вопросов.

SPI доступен по почте:

> Software in the Public Interest
> P.O. Box 70152
> Pt. Richmond, CA 94807-0152

2.3.9 Списки рассылки.

Имеется несколько списков рассылки, касающихся Debian:

`debian-announce@lists.debian.org`
Модерируемый. Анонсы системы. Что-то около одного сообщения в месяц.

`debian-changes@lists.debian.org`
Анонсы новых пакетов с программным обеспечением (стабильным). Несколько сообщений в день.

`debian-devel-changes@lists.debian.org`
Анонсы новых пакетов с программным обеспечением (НЕстабильным). Несколько сообщений в день.

`debian-user@lists.debian.org`
Список для вопросов о помощи и ответов на них от пользователей Debian. Около 50 сообщений в день.

`debian-sparc@lists.debian.org`
`debian-alpha@lists.debian.org`
`debian-68k@lists.debian.org`
Списки для тех, кто портирует Debian на платформы SPARC, DEC Alpha и Motorolla 680x0.

Имеется также несколько списков для разработчиков Debian.

Вы можете подписаться на список рассылки почтой или через World Wide Web. Подробности на `http://www.debian.org`.

2.3.10 Система отлова ошибок.

Debian project имеет систему отлова ошибок, которая обрабатывает отчеты пользователей об ошибках. Как только появится сообщение об ошибке, оно получает номер, со всей сопутствующей информацией сохраняется в файле и отсылается

разработчику пакета. При исправлении ошибки, она помечается как ``closed" разработчиком. Она может быть открыта снова...

Чтобы получить больше информации относительно системы отлова ошибок пошлите e-mail на `request@bugs.debian.org` со словом `help` в теле сообщения.

2.3.11 Debian-благодарности.

Большое спасибо Bruce Perens и другим авторам материалов по Debian, которые использовались, чтобы написать эту главу. Спасибо также Vadik Vygonets, моему любимому кузену, который мне немного помог. Наконец, благодарность членам Debian community за их тяжелую работу. Давайте надеяться, что Debian GNU/Linux станет еще лучше (прим. переводчика: откровенно говоря, не помешало бы!).

2.3.12 Последнее замечание.

Debian GNU/Linux меняется очень быстро, и многие фактов могут измениться быстрее этой книги. Исходный текст этого раздела (прим. переводчика: на английском языке) модифицируется регулярно. Вы можете найти его по адресу `http://www.cs.huji.ac.il/borik/debian/ligs`.

2.4 Red Hat Linux.

Глава про Red Hat Linux написана Henry Pierce.

2.4.1 Установка Red Hat Linux.

```
Dependencies:                    yes
Install boot methods:            CD, floppy,loadin (from DOS)
Install methods:                 CD, hard disk, NFS, FTP, SMB
System initialization:           Sys V init
Ease of installation:            easy
Graphical administration tools:  numerous
Installation utility:            install script that calls RPM
Package maintenance utility:     RPM, glint
```

2.4.2 Система управления пакетами RPM.

Менеджер пакетов Red Hat Linux RPM управляет программным обеспечением, определяя, как пакет программ сформирован для установки и собирает информацию относительно компонентов и методов установки в процессе формирования пакета. RPM-пакет имеет организованный набор данных в заголовке *package*`.rpm`, который может быть добавлен к базе данных, описывающей, принадлежность пакета, какое пакеты ему нужны, установлены ли требуемые пакеты и обеспечивает средства для определения программных зависимостей.

RPM дает администраторам системы возможность обновлять компоненты или всю систему при сохранении конфигурации системы или пакета; запрашивать базу данных о расположении файлов пакетов или связанной информации; выполнять проверку правильности установки пакета; храните исходные пакеты и обновления к ним отдельно. Поскольку RPM обеспечивает данные возможности, можно ставить, обновлять и удалять пакеты одной командой в текстовом режиме или несколькими щелчками мышки в X Package Management Tool. Пример работы с RPM из командной строки:

```
# rpm --install package.rpm # Установить пакет
# rpm --upgrade package.rpm # Обновить пакет
# rpm --erase package.rpm   # Удалить пакет
```

Соглашения об именах пакетов.

Правильно сформированный *package*`.rpm` имеет имя файла пакета, в которое входят: имя пакета, версия, версия обновления, архитектура и расширение `.rpm`, которое определяет файл, как RPM-пакет.

Например, `bash-1.14.7-1.i386.rpm`. Имя непосредственно содержит полезную информацию: пакет `bash` (Bourne Again SHell), версия 1.14.7, обновление 1 текущей версии для Red Hat Linux. Предназначен для Intel или совместимых 80386 (или выше) CPU и он в формате RPM. Так, если Вы видите, что пакет именован `bash-1.14.7-2.i386.rpm`, Вы знаете, что это второе обновление `bash` версии 1.14.7, и вероятно содержит исправления для предыдущей версии. В то время как внутренняя организация `*.rpm` файла здесь не рассматривается, правильно сформированный пакет содержит исполняемый файл, любые файлы конфигурации, документацию (по крайней мере man-страницы), любые файлы непосредственно связанные с пакетом, запись того, где файлы пакетов должны быть установлены и запись о любых требуемых пакетах. После успешной установки информация о пакете будет зарегистрирована в базе данных RPM системы. Более полное обсуждение системы управления пакетами RPM может быть найдено в RPM HOWTO (см. приложение А). Документ также доступен по адресу `http://www.redhat.com/support/docs/rpm/RPM-HOWTO/RPM-HOWTO.html`.

2.4.3 Апгрейд Red Hat Linux.

Обновления стали возможны только с версии Red Hat Linux 2.0 и выше, из-за глобальных изменений в двоичном формате Linux. Обновление старых версий выполняется методами установки с CD-ROM, NFS, FTP или жесткого диска. Начиная с Red Hat Linux версии 4.0 опция обновления встроена в Boot-дискету вместо отдельной программы. Если Вы обновляетесь с версий от 2.1 до 3.0.3 на версию 4.0, надо создать Boot-диск, а не искать скрипт обновления. Данный метод не переформатирует Ваши диски и оставит целыми файлы настроек.

2.4.4 Создание инсталляционных дискет.

Для создания Installation Floppy Kit надо:

1. Образ Red Hat Boot-диска, `boot.img`, который доступен по: `ftp://ftp.redhat.com/pub/redhat/current-i386/images/boot.img` или в каталоге `images` на Red Hat CD-ROM.
2. Образ Red Hat Supplemental-диска, `supp.img`, доступного на: `ftp://ftp.redhat.com/pub/redhat/current-i386/images/supp.img` или в каталоге `images` на Red Hat CD-ROM. Эта дискета требуется, если ваш метод установки основан не на CD-ROM, или Вам нужна поддержка PCMCIA. Эта дискета может также использоваться как Boot-дискета в аварийной ситуации.
3. Программа `RAWRITE.EXE`, доступная на: `ftp://ftp.redhat.com/pub/redhat/current-i386/dosutils/rawrite.exe` или в каталоге `DOS` на Red Hat CD-ROM.
4. Пользователи MS-DOS и Windows 95 устанавливающие Red Hat Linux впервые на машине, которая будет иметь Linux, установленную как вторую операционную систему, должны также получить с `ftp://ftp.redhat.com/pub/redhat/dos/fdipsll.zip` и распаковать файлы в `C:\FIPS`, если нужно свободное место на жестком диске.
5. Emergency Boot-диск для существующей операционной системы на машине, на которой Linux будет установлена как вторая операционная система.

2.4.5 Носитель для установки.

После создания установочных дискет обеспечьте правильность Вашего метода установки для дискет установки Red Hat. Для установки с CD-ROM, NFS, FTP и с жесткого диска исходным должен быть каталог `/RedHat` на верхнем уровне с каталогами `/base` и `/RPMS` в нем:

```
RedHat
|---> RPMS (contains binary the rpms to be installed)
|---> base (contains a base system and. files to set up the hard drive)
```

Установка по NFS.

Для установки по NFS, Вы должны иметь Red Hat CD-ROM на машине (например, на существующем сервере под Linux), которая поддерживает и экспортирует файловую систему ISO-9660 и Rockridge Extensions, или копию дистрибутива Red Hat с деревом каталогов, организованным как описано выше. Каталог `/RedHat` должен экспортироваться по сети. Машина должна быть подключена к сети Ethernet; установка по NFS невозможна по dialup связи.

Установка с жесткого диска.

Для установки с жесткого диска в его корневом каталоге должен быть каталог `/RedHat`, содержащий дистрибутив. Например, на первичном разделе DOS, путь к `\RedHat` должен быть `C:\RedHat`. В файловой системе MS-DOS не имеет значения, что имена пакетов *package*`.rpm` усечены. Все, что Вы должны сделать, удостоверится, что каталог `\RedHat\base` содержит основные файлы и `\RedHat\RPMS` содержит все файлы *package*`.rpm`.

Установка с FTP.

Для установки с FTP через Internet Вы должны знать IP-адрес FTP-сервера и путь к корневому каталогу дистрибутива Red Hat Linux. См. в приложение B список Linux FTP сайтов и зеркал. Если у Вас медленная линия, рекомендуется скопировать файлы к себе на жесткий диск (например, на имеющийся раздел MS-DOS) и ставить систему оттуда. Общий размер пакетов в каталоге `/RedHat/RPMS` около 170 MB и установка может занять немало времени. К тому же, при обрыве связи придется начать сначала. Если скачаете все к себе, намаетесь меньше. Для установки минимальной системы не нужно сгружать весь каталог `/RedHat/RPMS`. Подробности рассмотрены ниже.

2.4.6 Настройка установки с NFS или жесткого диска.

Вы можете настраивать процесс установки. Но такой подход только для тех, кто хорошо разбирается в Linux. В Red Hat Linux версии 4.*x,* каталог `/RedHat/RPMS` хранит примерно 170 MB файлов `.rpm`. RPM сжимает эти пакеты и получается, что пакет требует от 2 до 3 MB жесткого диска на каждый мегабайт RPM-пакета. Если *package*`.rpm` имеет размер 6 MB, нужно от 12 до 18 MB места для его установки.

Опция указания какие пакеты доступны для установки применима к установке по FTP, NFS и с жесткого диска. На CD-ROM запись невозможна, но Вы можете скопировать файлы на жесткий диск и устанавливать систему оттуда с настроенным списком пакетов. Настройка FTP и NFS установок возможны только если Вы имеет root-права на сервере. Следующие ситуации делают заказную установку желательной: при получении Red Hat Linux по FTP с медленным каналом или при проектировании набора программного обеспечения, которое нужно использовать всем рабочим станциям с Red Hat Linux в сети.

Чтобы настраивать установку, Вы должны получить файл `/base/comps`, который обеспечивает Вас списком пакетов, которые обычно включает полная установка. Пакеты, которые Вы хотите устанавливать из `/base/comps` должны быть скачаны из сети. Файл `/base/comps` должен быть отредактирован, чтобы отразить список пакетов, которые Вы получили и собираетесь устанавливать.

Если у Вас есть свои пакеты RPM, их можно добавить в файл `comps`.

Файл `comps`.

Программа установки Red Hat использует файл `/RedHat/base/comps` (здесь приведен пример для Red Hat Linux 4.0) для определения какие пакеты доступны в `/RedHat/RPMS` для установки. Файлы организованы по категориям и каждая категория имеет список пакетов, минимально необходимых для нее. ЗАМЕЧАНИЕ: упоминается только часть имени пакета, а именно имя и версия (`package-version-build.rpm`). Это сделано для совместимости файла `comps` с другими версиями Red Hat. Секция файла имеет структуру:

```
number category
package

...

end
```

Этот тэг задает число категорий, категорию, список файлов пакетов в категории и тэг завершает определение категории.

Всегда нужны пакеты, перечисленные в разделе **Base** файла. Другие разделы можно исправить или вовсе удалить для обеспечения требуемой настройки. Например, имеются три типа **Networked Stations (сетевых станций)**: "простая", управляющая и dial-up. Исследование этих разделов показывает, что многие из пакетов программ перечислены во всех трех категориях, но некоторые пакеты программ специфические для каждой категории. Если Вы создаете набор для **Dial-up Networked Station** то Вы можете безопасно устранить остальные категории вместе с программами, специфичными для них. Если нужна какая-то одна категория, нужно просто выкинуть лишние категории вместе с уникальным для них софтом. Если есть какие-то пакеты, которые не входят в дистрибутив Red Hat, допишите их в нужную категорию.

Поскольку список пакетов хранит только имя пакета (то есть, не все имя `package-name-version-build.rpm`), Вы можете подставлять обновления, доступные в каталоге `updates` на `ftp://ftp.redhat.com/pub/redhat/current/update` или любом зеркале Red Hat. Программа установки относительно нечувствительна к версии. Единственное предупреждение здесь: нужно обеспечить, чтобы зависимости пакетов были выполнены. Когда пакет RPM сформирован, RPM пробует определить какие пакеты должны быть установлены для его работы. Разработчик пакета имеет контроль над данным процессом: может добавлять связи, которые не может найти сам RPM. Здесь может понадобиться небольшое исследование. Например, один способ определять зависимости пакета (если Вы имеете пользовательский доступ к NFS-серверу на существующей машине с Red Hat Linux), зайти через `telnet` или `login` на машину (в случаее CD-ROM, вставьте его и перейдите в каталог `RedHat/RPMS`) и сделайте запрос зависимостей пакета:

```
[root@happy/RPMS] rpm -q -p -R bash-1.14.7-1.i386.rpm
libc.so.5
libtermcap.so.2
```

Опция ``-q'' переводит `rpm` в режим запроса, ``-p'' запрашивает данные на неинсталлированный пакет и ``-R'' требует, чтобы `rpm` внес в список зависимости целевого пакета. В данном примере мы видим, что нужны `libc.so.5` и `libtermcap.so.2`. Поскольку `libc` и `termcap` нужны для другого программного обеспечения (`bash`), Вы должны обеспечить, чтобы пакеты `libc` и `libtermcap` присутствовали для установки `bash`. После окончания установки основной системы можно загрузиться в ней после того, как отработает программа установки. Загрузившись Вы сможете добавлять пакеты в как Вам нужно, даже если менеджер пакетов сообщает, что пакет не устанавливается должным образом из-за конфликта зависимости.

Таблица 2.4 описывает категории программ в файле `/base/comps` для Red Hat 4.0:

Тип	**Необходим?**	**Комментарий**
BASE	Да	Базовая система; нельзя модифицировать

		при установке.
C Development	Весьма рекомендуется	Минимальный вариант требуется для компиляции ядра системы.
Development Libs	Весьма рекомендуется	Минимальный вариант требуется для компиляции ядра системы.
C++ Development	Не обязательный	Программирование на C++.
Networked Workstation	Рекомендуется (необходим для других программ)	Требуется как для работы при подключении Ethernet, так и для телефонных линий для работы в сети. Нельзя модифицировать при установке.
Anonymous FTP/Gopher Server	Не обязательный	Если система Linux будет выступать FTP-сервером или сервером Gopher.
Web Server	Не обязательный;	Полезен для программирования www-серверов и обслуживания www-страниц.
Network Management Workstation	Не обязательный	Есть дополнительные средства как для подключения по Ethernet, так и через телефонную линию.
Dialup Workstation	Рекомендуется	Необходим если связь будет через телефонную линию.
Game Machine	Не обязательный	Игры...
Multimedia Machine	Не обязательный	Нужен если есть средства Multimedia.
X Window System	Не обязательный	Если необходима X Window.
X Multimedia Support	Не обязательный	Нужен если есть средства Multimedia.
TeX Document Formatting	Не обязательный	Рекомендуется устанавливать *целиком*.
`emacs`	Рекомендуется	Истинный Редактор.
`emacs` with X support	Рекомендуется	Истинный Редактор для X, требует X.
MS-DOS and Microsoft Windows Connectivity	Не обязательный	Связь с системами MS-DOS и Microsoft Windows (без комментариев).
Extra Documentation	Рекомендуется	*Всегда* устанавливайте систему экранной документации man (manual pages)!

Таблица 2.4: Важные пакеты в Red Hat Linux.

2.4.7 Рекомендуемая минимальная установка.

Трудно определить сколько места потребуется. Однако, при установке по FTP надо получить системы **Base** и **Dialup Networked Station** и поставить их. Затем, дополнительное программное обеспечение может быть получено и добавлено по мере возникновения потребности в нем. Конечно, если Вы хотите программировать на C, Вы должны получить нужные пакеты и соответственно файл отредактировать `comps`.

Если Вы сталкиваетесь с пакетом, который требует другого пакета, которого у Вас нет, или Вы сделали ошибку в файле `comps`, Вы можете закончить установку и получить рабочую систему. Вы можете исправить проблему, вручную добавляя неудачные пакеты и их зависимости позже. В целом, получите всю **Base** систему и один из установленных пакетов **Networked Station**, после чего всегда можно добавить что угодно еще.

2.4.8 Сколько места нужно?

Таблица предоставляет примерные требования к месту на диске Red Hat Linux и различных подсистем.

Раздел	**Рекомендуемый объем**	**Пояснения**
Своп-раздел	2 X объем RAM	Если объем RAM меньше 16MB, то минимальный объем своп-раздела 16MB. Если есть 16MB RAM, а места на диске немного, рекомендуемый объем своп-раздела не менее объема RAM.
Корневой раздел, без системы X	100-200MB	Зависит от объема нужных приложений (компиляторов и т.п.).
Корневой раздел, система X установлена	250-350MB	Зависит от объема нужных приложений (компиляторов и т.п.).
`/home`	от 5MB	Зависит от количества пользователей и их потребностей.
`/var`	от 5MB	Зависит от количества пользователей, источников новостей и т.п.
`/usr/local`	25-200MB	Для программ в формате, отличном от формата RPM, и для других программ, которые нужно хранить отдельно от Red Hat.
`/usr`	от 350 MB	

Таблица 2.5: Требования Red Hat Linux к месту на диске.

2.4.9 Установка.

К настоящему времени, Вы должны были создать Installation Floppy Kit, подготовить Ваш жесткий диск и носитель дистрибутива. Сначала загрузите систему и настройте программу установки на нужный носитель дистрибутива. После чего установка будет одинаковой для всех типов носителей. Детали обсудим ниже. Для начала загрузите компьютер с дискеты, помеченной как ``Boot diskette."

2.4.10 Носитель дистрибутива.

После загрузки с boot-дискеты ядро попытается обнаружить аппаратные средства, для которых ядро начальной загрузки имеет драйверы, компилируемые непосредственно в него. Как только ядро разберется с аппаратурой, будет выдан запрос, имеется ли цветной монитор. Если да, нажмите на ``OK''. Затем появится заставка Red Hat Welcome screen. Нажмите ``OK''. Теперь будет вопрос о необходимости поддержки PCMCIA. Если она нужна, ответьте ``yes'' и по запросу вставьте дискету Supplemental. Теперь будет главное: запрос о методе установки. Следуйте инструкциям, данным в следующем разделе.

Установка с CD-ROM.

Для установки с CD-ROM выберите ``Local CD-ROM'' из списка типов установки. Нажмите ``OK''. Будет задан вопрос о том, какой у Вас CD-ROM: SCSI, IDE/ATAPI или другой. В этом месте исследование аппаратуры часто вносит путаницу: если Вы имеете 4X или более быстрый диск CD-ROM, который был сделан недавно и связан с Sound Blaster или другой звуковой платой, Вы наиболее вероятно имеете диск типа IDE/ATAPI. Это одна из наиболее путающих проблем, стоящих перед Вами.

Если выбран SCSI, надо знать производителя SCSI-карты. Прокручивайте список, пока не найдете Ваш тип карты (прим. переводчика: его там может и не быть). Когда найдете и выберите появится вопрос проводить ли поиск параметров (AUTOPROBE) или Вы их введете (SPECIFY OPTIONS). Неплохие результаты дает AUTOPROBE.

Когда программа установки Red Hat найдет CD-ROM, можно переходить к следующей главе.

Установка с жесткого диска.

Для установки с жесткого диска выберите соответствующую опцию и нажмите ``OK''. Если выбрана поддержка PCMCIA, появится запрос на установку дискеты Supplemental.

Установка с NFS.

Для установки с NFS выберите соответствующую опцию и нажмите ``OK''. Вы должны выбрать карту Ethernet на машине назначения, чтобы программа установки могла корректно загрузить драйвер. Выберите карту из списка и нажмите ``OK''. Программа установки выполнит анализ карты.

Если машина повисла, нажмите Ctrl-Alt-Delete для перезагрузки системы. Главная причина в том, что изучается карта не-Ethernet. Если это случилось, пробуйте снова, выберите SPECIFY OPTIONS, и дайте данные о Вашей платы в этой форме:

```
ether=IRQ,IO_PORT,eth0
```

Данная команда указывает провести анализ расположения, заданного значениями *IRQ* и *IO_PORT* для карты Ethernet. Если Ваша карта Ethernet сконфигурирована на IRQ 11 и IO_PORT 0x300, укажите:

```
ether=l1.0x300.eth0
```

После того, как плата будет найдена, Вас спросят относительно TCP/IP информации о Вашей машине и NFS-сервера с пакетами установки Linux. Сначала будут заданы вопросы о *IP-адресе целевой машины, маске подсети, шлюзе* и *Primary Name Server*. Например:

```
IP address 192 168 181 21
Netmask 255 255 255 0
Default Gateway 192 168 181 1
Primary Nameserver 192 168 181 2
```

После выбора `OK`, Вас спросят о имени машины (*Host name*) и имени домена (*Domain name*). Например, если Ваш домен `infomagic.com` и имя машины `vador`, введите:

```
Domainname infomagic.com
Host name vador.infomagic.com
Secondary nameserver IP (если нужен).
Tertiary nameserver IP (если нужен).
```

Последний экран запрашивает о сервере NFS и имени экспортируемого каталога с дистрибутивом Red Hat. Например, если Ваш NFS сервер `redhat.infomagic.com`, введите:

```
NFS Server name    redhat.infomagic.com
Red Hat Directory  /pub/mirrors/linuч/RedHat
```

Если не знаете данного значения, спросите у системного администратора. После ввода нажмите `OK`. Если программа установки сообщает об ошибке, возможны два варианта: Вы что-то ввели неправильно или не имеете прав доступа к этому каталогу. Обратитесь к системному администратору.

Установка с FTP.

Установка с FTP подобна установке с NFS, описанной выше. Вас спросят о плате Ethernet и TCP/IP информации о Вашей машине. Однако, в отличие от установки с NFS, Вас спросят о имени *FTP-сервера* и *каталоге с дистрибутивом Red Hat* вместо параметров для NFS-сервера. Одно предупреждение относительно FTP установки: найдите самое близкое к Вашему расположению и наименее занятое FTP-зеркало. См. в приложении B список сайтов Linux FTP.

Если ваши аппаратные средства не обнаружены, надо указать их явно. Кроме того, имеет смысл проверить на `http://www.redhat.com/pub/redhat/updates/version/images` нет ли обновлений boot-дискет для Вашей аппаратуры.

2.4.11 Остальная часть установки.

1. Дальше Вас спросят о том, новая это система или обновление старой Red Hat Linux 2.0 или старше. Если обновление, Вам не будут предлагать возможность

создать разделы жесткого диска и настроить в системе что-либо, кроме LILO. Выберите соответственно `INSTALL` или `UPGRADE`.

2. Если Вы обновляетесь, Вас спросят о корневом разделе существующей системы Red Hat. Выберите раздел и нажмите `OK`. Если Вы устанавливаете систему впервые, Вы должны создать разделы жесткого диска. После создания разделов Linux Native и Linux Swap, Вы должны инициализировать и запустить своп. Затем Вас спросят в какой раздел Вы хотите поставить Linux. При обновлении укажите корневой раздел. Вы должны выбрать минимум один раздел. Если Вы не обновляете систему, будет выдана таблица доступных разделов. Выберите соответствующие разделы и нажмите `EDIT`, чтобы указать, какие разделы для каких каталогов будут использоваться. Если Вы имеете больше чем один раздел для установки Linux, самое время, чтобы обозначить их.
3. Затем будет выведен список программных категорий для установки. Здесь можно настроить какие программы из каждой категории будут установлены. Если раньше не была установлена какая-либо Linux, просто выберите нужную категорию, а значения по умолчанию для нее установит программа установки. Если Вы нуждаетесь в пакете, который не был установлен первоначально, Вы можете всегда устанавливать его позже. В то время как программное обеспечение устанавливается, Вы будете видеть индикатор хода работы, что займет некоторое время. Установка может занять от тридцати минут до часа или больше в зависимости от выбранных программ и аппаратной конфигурации.
4. После установки программ будут вопросы о настройке мыши. Протоколы мыши и устройства обсуждались выше .
5. Затем начнется настройка X Window System. С ней лучше подождать до первого запуска системы. Если что-то с ней пойдет не так, Вам придется устанавливать систему заново.
6. Если у Вас нет карты Ethernet, *не надо* настраивать сеть сейчас. А вот если плата есть, и до сих пор не настроена, настройте сейчас. Настройка dialup-соединения будет выполнена позже, по окончании установки.
7. Затем, Вы должны сконфигурировать часы системы. UTC хороший выбор, если Вы находитесь в сети и хотите, чтобы время корректно переключалось с летнего на зимнее и назад. Местное время тоже можно использовать.
8. Если Вы не имеете US-клавиатуры, Вы должны определить конфигурацию для Вашей клавиатуры.
9. Вас спросят о пароле пользователя `root`. Не забудьте его! Восстановление пароля не тривиальный вопрос. Вы будете нуждаться в пароле, чтобы обратиться к системе, когда Вы перезагрузитесь в первый раз.
10. В заключение, Вас будут спрашивать о настройке LILO.

Если корневой раздел не принадлежит к диапазону дорожек 0-1023, *НЕ СТАВЬТЕ LILO*. При первой перезагрузке системы, если LILO не позволяет Вам загружать вашу систему правильно, используйте Emergency MS-DOS и Windows 95 boot-дискету и на приглашение `A:\>` введите `FDISK /mbr`. После этого система будет загружаться в существующей MS-DOS или Windows 95 как было до установки LILO. Затем Вы можете использовать Red Hat Boot-диск со следующими параметрами при приглашении `boot:`, чтобы загрузить Вашу систему на жестком диске:

```
boot: rescue root=dev/xxxx ro load_ramdisk=0
```

Здесь *xxxx* задает корневой раздел.

После того, как процедура установки завершена, Вы готовы перезагрузить Вашу систему и использовать Linux.

2.4.12 После установки.

Теперь перезагрузитесь с жесткого диска. Linux запустится в первый раз.

Понимание подсказки LILO.

Когда Вы включаете или перезагружаете систему, Вы можете видеть подсказку LILO, которую Вы настроили на 30-секундную задержку перед загрузкой системы. Когда LILO появляется на экране, если Вы не делаете ничего, заданная по умолчанию операционная система загрузится после предписанного времени ожидания. Однако, из LILO, Вы можете управлять несколькими аспектами как начальной загрузки Linux, так и сообщить, чтобы LILO загрузил альтернативную операционную систему. Если Вы желаете отменить заданное по умолчанию поведение LILO, нажмите клавишу SHIFT при появлении подсказки LILO ``boot:''. Нажатие Tab при появлении подсказки LILO ``boot:'' выведет список доступных операционных систем:

```
LILO boot:

dos linux
boot:
```

Это сообщает нам, что ``dos'' является заданной по умолчанию операционной системой, которая загрузится, если ничего не набрать; чтобы загрузить Linux, напечатайте ``linux''. Однако, LILO позволяет Вам передавать параметры для ядра Linux, которые отменяют заданное по умолчанию поведение. Например, Вы, возможно, экспериментировали с файлами конфигурации запуска и сделали что-то, что вывело из строя систему. Если так, Вы хотите загрузить систему только до того места, когда она читает файлы конфигурации. Для этого есть параметр ``single'':

```
boot: linux single
```

загрузит систему в однопользовательском режиме для внесения исправлений. Это также полезно, если система не загружается полностью до подсказки `login:` по каким-то причинам.

Регистрация в первый раз.

Теперь, когда перед Вами впервые появилось приглашение `login:`, Вы можете задаться вопросом, как войти в систему. В этой ситуации на недавно установленной системе, имеется только один аккаунт, администратор, ``root''. Он используется, чтобы управлять системой и делать вещи подобные выбору конфигурации системы, добавлению и удалению пользователей, программного обеспечения и так далее. Для входа введите ``root'' на приглашение `login:` и нажмите Enter. Вас спросят о пароле, который Вы ввели в течение установки. Введите его в ответ на подсказку `password:` . Подсказка системы `[root@locahost] #` появляется после того, как Вы успешно закончили вход в систему. Подсказка системы сообщает Вам две вещи: Вы вошли как

`root` и в этом случае, Ваша машина названа `localhost`. Если Вы иначе назвали Вашу машину в течение процесса установки, заданное имя появится вместо `localhost`.

2.5 Caldera OpenLinux

Глава про Caldera OpenLinux написана Evan Leibovitch.

Данный раздел дополняет Getting Started Guides, поставляемые Caldera со своими Linux-based продуктами. Ссылки на Getting Started Guide для Caldera Open Linux Base помечены в главе как ``Guide".

2.5.1 Получение Caldera OpenLinux.

В отличие от большинства других дистрибутивов Linux, Caldera OpenLinux не доступна для загрузки из Internet, и при этом не может распространяться свободно (прим. переводчика: демо-версию, которая не сильно отличается от полной, можно сгрузить с сайта Caldera. К тому же, она продается вместе с некоторыми книгами по Linux). Это из-за коммерческих пакетов, которые являются частью COL; в то время как большинство компонентов COL находится под GNU Public License, коммерческие компоненты, типа Looking Glass и Metro-X, таковыми не являются. В списке пакетов, включенных в средства COL на странице 196 Guide, коммерческие пакеты отмечены звездочкой.

COL доступен непосредственно из Caldera, или через сеть партнеров во всем мире, они обычно обеспечивают профессиональную помощь, конфигурацию и обучение для пользователей Caldera. Текущий список партнеров находится на web-сайте Caldera.

2.5.2 Подготовка к установке Caldera OpenLinux.

Caldera поддерживает те же самые аппаратные средства, что и любой другой дистрибутив, основанный на ядре Linux версии 2.0. Приложение A Guide имеет список большинства поддерживаемых SCSI-хостов и параметров конфигурации, необходимых для многих аппаратных комбинаций.

Caldera Guide имеет план установки, с описанием всех деталей системы, которые понадобятся при установке. Строго рекомендуется заполнить данный план перед установкой, он сильно облегчит Вам жизнь.

2.5.3 Создание дискет boot/modules.

Дистрибутивы COL не включают дискеты для установки. Нужно сделать две дискеты: одна для загрузки и другая ``modules" содержит много драйверов.

В то время как Guide рекомендует создать дискеты, копируя их с CD-ROM, лучше получить более новые версии дисков с сайта Caldera. Образы на старых CD-ROM имеют ошибки, которые вызывают проблемы, особенно с инсталляциями, использующими SCSI диски и большие разделы.

Чтобы получить более новые версии образов дискет, загрузите их с Caldera FTP. В каталоге `pub/col-1.0/updates/Helsinki`, вы найдете связку пронумерованных каталогов. Проверьте каталоги в порядке по убыванию, чтобы удостоверится, что Вы получаете последние версии.

Если Вы обнаружите, что один из этих каталогов имеет подкаталог, названный `bootdisk`, его содержимое как раз и есть то, что Вы хотите.

Вам нужно найти два файла:

```
install-2.0.25-XXX.img
modules-2.0.25-XXX.img
```

Замените *XXX* номером версии образов дискет.

После получения образов запишите их на дискеты, как сказано для универсальной инсталляции выше, используя MS-DOS-программу `RAWRITE.EXE` с Caldera CD-ROM или команду `dd` из Linux.

Caldera CD-ROM загрузочные. Если Ваш BIOS поддерживает такие диски, с него можно загрузиться, но лучше скачать образы дискет, если возможно. Они имеют меньше ошибок, поскольку поновее.

2.5.4 Подготовка жестких дисков.

Здесь все как обычно для других дистрибутивов. Используйте `fdisk` на загрузочным жестком диске для выделения минимум двух разделов Linux, одного для свопа и одного для корневой файловой системы. Если планируется двойная загрузка COL с другой ОС, например Microsoft Windows, MS-DOS или OS/2, COL обычно предпочтительно установить последней. Linux `fdisk` распознает ``чужие'' типы ОС лучше аналогичных программ из большинства других систем.

Для запуска Linux `fdisk`, Вы должны запустить систему с boot (и может быть modules) дискеты. Вы должны сообщить COL какой диск и контроллер диска Вы имеете. Даже `fdisk` не удастся использовать, если Linux не распознает Ваш жесткий диск!

Чтобы это сделать, выполните инструкции в Guide, с шага 2 на страницах 33-36. Не беспокойтесь об обнаружении CD-ROM или сетевой карты: в данный момент главное, чтобы Linux видела загрузочный жесткий диск, чтобы его можно было разметить программой `fdisk`. Краткое описание использования Linux `fdisk` находится на странице 28 Guide.

Не забудьте, что при работе с `fdisk`, Вы должны установить корневую файловую систему как Linux Native (тип 83) и своп-раздел (тип 82) как новые разделы. Краткое обсуждение того, сколько свопа надо предлагается на странице 10 Guide.

Сразу после создания разделов и записи новой их таблицы на диск надо перезагрузиться.

2.6 Slackware

Глава про Linux Slackware написана Sean Dreilinger.

2.6.1 Slackware не для всех.

Добро пожаловать в Slackware Linux! Этот раздел стремится помочь новому пользователю Linux, или администратору, оценивающему Slackware, планировать систему и устанавливать Slackware Linux.

Выбирать ли Slackware как разновидность Linux, которую Вы используете серьезное решение. Это может казаться тривиальным сейчас, но Linux имеет тенденцию становится все более важной. Множество экспериментов с Linux быстро развилось в очень серьезные машины, обслуживающие намного больше пользователей и задач, чем первоначально предполагалось. Slackware один из наиболее широко используемых дистрибутивов Linux.

2.6.2 Краткая история.

В 1993 году фирма Soft Landing System создала один из первых организованных дистрибутивов Linux. Хотя это было большое начало, дистрибутив SLS имел много недостатков (точно не для новичков). Slackware от Patrick Volkerding решил большинство проблем, был отзеркалирован на многих FTP, выпущен на CD-ROM во всем мире и быстро стал наиболее широко используемой разновидностью Linux. Некоторое время Slackware был единственным полным Linux-решением. Постепенно были созданы другие достойные дистрибутивы, но Slackware был первым.

К январю 1994 года Slackware достиг очень широкого распространения.

2.6.3 Что дальше?

Если Вы - администратор системы, Вы можете уже иметь один или несколько серверов, управляемых Slackware. Если Вы не имеете времени, чтобы экспериментировать на работе, то Вам нужен проверенный и надежный дистрибутив. Недостатки Slackware широко известны, главным образом обнаружены, зарегистрированы и исправлены везде, где возможно. Вы можете создать Slackware-систему, закрыть известные отверстия в защите и установить некоторые дополнительные инструментальные средства из других дистрибутивов Linux, чтобы создать превосходную UNIX-станцию или настольное автоматизированное рабочее место, все делается приблизительно за полдня.

Взгляните также на Руководство Покупателя, изданное в *Linux Journal*, который дает полное сравнение и оценку всех главных дистрибутивов. Простой перечень дистрибутивов Linux есть в Linux Distribution HOWTO (см. приложение А).

2.6.4 Upgrade? Дважды подумайте!

Обновлять Slackware очень не просто. Программа установки разработана, чтобы поместить новую операционную систему на пустые жесткие диски или освобождать дисковые разделы. Установка поверх предыдущей Slackware может стереть Ваши прикладные программы и вызвать проблемы совместимости между модифицируемыми прикладными программами и старыми файлами в той же самой системе. В старые времена для только-только развивающейся ОС, такое было вполне нормально, но сейчас на Slackware Linux выполняется много критических программ. В такой среде простая перезагрузка запланированное действие и система, перезаписывающая все файлы пользователей или прикладные программы абсолютно недопустима.

В принципе обновление реально, если Вы знаток UNIX, и озаботились резервированием своих данных. В интернете есть ресурс, где утверждается существование простой процедуры апгрейда Slackware. Посмотрите URL `ftp://ftp.wsc.com/pub/freeware/linux/update/linux`.

Или почитайте замечания по апгрейду Greg Louis в его mini HOWTO: *Upgrading Your Linux Distribution* доступного в рамках проекта LDP (прим. переводчика: в рамках проекта Russian LDP он пока недоступен) `http://sunsite.unc.edu/LDP`.

2.6.5 Выбор метода установки.

Slackware может быть установлен с ряда носителей и из сети. Каждый метод установки требует, чтобы Вы имели по крайней мере три дискеты для загрузки с них основной системы и программы установки.

CD-ROM.

Установка с CD-ROM быстрая, удобная и легкая. Несмотря на затраты на покупку диска, способ хорош тем, что диск можно копировать сколько угодно. Так как Linux и Slackware copylefted, можно делать сколько угодно копий. Установка с CD-ROM хороша и тем, что Вы не забиваете на день канал передачи с FTP. Кроме того, к CD-ROM прилагается документация и набор вспомогательных утилит, что может быть полезным.

Получение советов.

Если Вы человек увлеченный (или просто хотите поставить Slackware не один раз, прежде чем все будет работать), обратитесь в местную LUG (Linux User Group). Вообразите целую комнату, полную щедрых и хорошо осведомленных хакеров, объединившихся, чтобы совместно использовать CD-ROM и экспертизу с другими энтузиастами.

FTP.

Когда Вы скачаете Slackware из самого близкого зеркала FTP, все еще надо будет поместить наборы дисков Slackware на какой-нибудь диск (Slackware не умеет ставиться прямо с FTP-архива).

NFS.

В сетевой среде можно установить Slackware на общедоступной файловой системе и позволить каждому в локальной сети присоединяться к этому общедоступному расположению и установке. Подробности описаны здесь:

```
ftp://ftp.cdrom.com/pub/linux/slackware/MIRRORS.TXT
http://sunsite.doc.ic.ac.uk/sunsite/access-nfs.html
http://www.cs.us.es/archive/nfs.html
```

Дискеты.

Это потребует времени, но сработает: Вы можете создать груду дискет (около 50 штук), необходимых чтобы установить Slackware и затем подавать их в ваш дисковод одну за другой по запросу. Дисковые наборы Slackware фактически разработаны для использования дискет. Если Вы имеете огромное число дискет высокой плотности в Вашем распоряжении, это может быть наиболее экономичный способ установки.

Жесткий диск.

Данный подход годится, если вы скачали дистрибутив Slackware с FTP. Надо создать только boot, root и rescue-дискеты. Нужно иметь достаточно свободного места для хранения файлов дистрибутива на время установки (Вы можете стереть их впоследствии). Установка с жесткого диска также хороша, если Вы купили CD-ROM но Ваш дисковод CD-ROM не поддерживается ни одним из ядер Linux, которые поставляются на Slackware CD.

Стриммер.

Пока на стадии отладки, но может сгодится, поскольку является компромиссом между быстродействием и затратами. Детали в секции Tape файла `INSTALL.TXT` в Вашем дистрибутиве Slackware.

2.6.6 Boot-диски: их надо иметь.

Даже если у Вас прямое подключение к линии T-3 Internet, которое позволяет Вам сосать новый дистрибутив Slackware из сети, будет хорошо создать для Slackware загрузочные дискеты (boot и root). В случае неудачи (отключение электричества, кошачьи друзья, пересекающие клавиатуру, или даже человеческая ошибка), эти два диска могут быть способны восстановить систему или по крайней мере спасти личные файлы. Прим. переводчика: какова вероятность, чтобы кошка прошлась по клавиатуре и в FAR нажала последовательно + Enter F8 Enter? Но ведь нажала у меня однажды...

2.6.7 Планирование установки Slackware.

После того, как файлы все скопированы, Slackware может начать установку, если Вы готовы. Чтобы помочь Вам планировать Ваши решения, этот раздел состоит из плана установки, аналогичного тому, с каким Вам придется иметь дело в программе установки. Вы можете использовать его, чтобы записать ответы заранее (в то время как компьютер пока еще работает!).

1. **Клавиатура:** Slackware `setup` должна знать соответствует ли клавиатура типу USA 101 key.

   yes или no

2. **Конфигурация свопа:** Вы имеете один или больше разделов, подготовленных как тип 82 (Linux Swap)?

   yes или no

   Вы хотите, чтобы `setup` использовала `mkswap` на Ваших разделах свопа? Наименее вероятно ``yes'', если Вы имеете меньше чем 4 MB RAM и уже выполнили данную команду, чтобы `setup` работала лучше.

   yes или no

3. **Укажите главный раздел Linux:** `setup` выведет список список всех разделов типа 83 (Linux Native) и спросит, который из них должен быть корневым (/) для файловой системы Linux. Используйте формат типа `/dev/hda3` или имя устройства.

   имя раздела

   Последняя возможность, чтобы отступить! При использовании опции "install from scratch" Вы должны указать пустой раздел. Если Вы еще не форматировали его вручную, то надо отформатировать по запросу. Введите ``I'' для того, чтобы установить на пустом месте, или ``a'', чтобы добавить программное обеспечение к существующей системе.

   [i]nstall или [a]dd

   Переформатировать ли главный раздел Linux?

   [y]es, [n]o или [c]heck sectors

   `ext2fs` по умолчанию выделяет одному inode 4096 байт места на диске. Если вы собираетесь иметь много маленьких файлов на диске, Вы можете нуждаться в большем количестве inodes (каждый используется для одной записи о файле). Вы можете изменять расходы на один inode до 2048 или даже до 1024 байта. Введите `2048` или `1024`, или просто нажмитет Enter, чтобы принять значение по умолчанию, 4096.

   4096 (значение по умолчанию), 2048 или 1024

4. **Использовать дополнительные разделы Linux:** Вы можете смонтировать другие разделы для `/usr` или `/usr/X11` или других каталогов.

   [y]es или [n]o

   Используются разделы Linux (*отображается список разделов*). Есть, но не используются разделы Linux (*отображается список разделов*). Введите, какие

разделы еще нужно использовать, или нажмите q для выхода. Используйте формат `/dev/hda3` или имя устройства.

Имя раздела или [q]uit

Форматировать данный раздел?

[y]es, [n]o или [c]heck sections

Теперь этот новый раздел должен быть смонтирован где-нибудь в новом дереве каталогов. Например, если Вы хотите поместить его как `/usr/X11R6`, введите: `/usr/X11R6`.

Точка монтирования

Смонтировать дополнительные разделы?

[y]es или [n]o

5. **Настройка разделов DOS и OS/2:** Были найдены разделы DOS FAT или OS/2 HPFS: (*список разделов*). Вы хотите смонтировать их, чтобы видеть их из Linux?

   [y]es или [n]o

   Введите имя раздела, к которому хотите обращаться из Linux или q, чтобы выйти из процесса добавления разделов. Используйте формат: `/dev/hda3` или имя устройства.

   Имя раздела или [q]uit

   Сейчас данные разделы надо куда-то смонтировать. Например, в каталоги `/dosc`, `/dosd` или подобные. Куда именно?

   Точка монтирования

6. **Выбор носителя дистрибутива:**

```
7.
8. (1) Install from a hard drive partition
9. (2) Install from floppy disks
10.         (3) Install via NFS
11.         (4) Install from a pre-mounted directory
12.         (5) Install from CD-ROM
13.
```

   1, 2, 3, 4 или 5

14. **Установка с раздела жесткого диска:** Чтобы устанавливать непосредственно с жесткого диска, Вы должны иметь раздел с каталогом, содержащим дистрибутив Slackware. В нем должны быть подкаталоги, соответствующие дисковым наборам. Например, если дистрибутив находится в `/stuff/slack`, Вы должны иметь каталоги `/stuff/slack/a1`, `/stuff/slack/a2` и подобные, каждый содержит соответствующий дисковый набор. Вы можете устанавливать из DOS,

HPFS, или разделов Linux. Введите имя раздела, где лежит дистрибутив Slackware, или p, чтобы увидеть список разделов.

Имя раздела или [p]artition list

Какой каталог на этом разделе хранит дистрибутив Slackware. Например, `/stuff/slack`.

Имя каталога

Какой тип файловой системы на диске с дистрибутивом Slackware?

```
(1) FAT (MS-DOS, DR-DOS, OS/2)
(2) Linux Second Extended File System
(3) Linux Xiafs
(4) Linux MINIX
(5) OS/2 HPFS
```

1, 2, 3, 4 или 5

15. **Ставить из смонтированного каталога:** Система может быть установлена из каталога, который смонтирован сейчас. Это может быть обычный или NFS-каталог. Вам надо указать каталог, хранящий подкаталоги для всех дисковых наборов.

    Имя каталога

16. **Ставить с дискет:** Slackware base series (A) может быть установлена с дискет 1.2M или 1.44M. Большинство других дисков не влезает на 1.2M дискеты, media, но может быть загружено на жесткий диск и установлен оттуда позже. Откуда ставить (1/2/3/4)?

```
17.
18.          /dev/fd0u1440 (1,44M drive a:)
19.          /dev/fd1u1440 (1,44M drive b:)
20.          /dev/fd0h1200 (1,2M drive a:)
21.          /dev/fd1h1200 (1,2M drive b:)
22.
```

    1, 2, 3 или 4

23. **Ставить по NFS:** Если планируется установка по сети, не будет выполняться перенастройка платы Ethernet. Вы работаете в сети?

    [y]es или [n]o

    Нужен IP-адрес Вашей машины. Пример: 111.112.113.114.

    IP-адрес

    Нужна маска подсети. Обычно 255.255.255.0.

Сетевая маска

Используется ли шлюз (y/n)?

[y]es или [n]o

Адрес шлюза.

IP-адрес

Теперь нужно указать откуда брать дистрибутив по сети. А именно IP-адрес Вашего сервера NFS.

IP-адрес

На сервере есть каталог, в котором хранятся дисковые наборы Slackware. Программа `setup` должна знать его имя. Если диск A3 находится в `/slackware/a3`, надо ответить `/slackware`.

Имя каталога

24. **Установка с CD-ROM:** Тип Вашего CD-ROM.

```
25.
26.         (a) Works with most ATAPI IDE CD drives (dev/hd*)
27.         (b) SCSI (/dev/scd0 or /dev/scd1)
28.         (c) Sony CDU31A CDU33A (/dev/sonycd)
29.         (d) Sony 531 535 (/dev/cdu535)
30.         (e) Mitsumi, proprietary interface--not IDE (/dev/mcd)
31.         (f) New Mitsumi, also not IDE (/dev/mcdxO)
32.         (g) Sound. Blaster Pro Panasonic (/dev/sbpcd)
33.         (h) Aztech Orchid Ohano Wearnes (/dev/aztcd)
34.         (i) Phillips and some ProAudioSpectrum16 (/dev/cm206cd)
35.         (j) Goldstar R420 (/dev/gscd)
36.         (k) Optics Storage 8000 (/dev/optcd)
37.         (l) Sanyo CDR-H94+ISP16 soundcard (/dev/slcd)
38.         (m) Try to scan for your CD drive
39.
```

1, 2, 3, 4, 5, 6, 7 8, 9, 10, 11, 12 или 13

IDE CD-ROM: Введите имя устройства для Вашего IDE CD-ROM. Это будет, вероятно, одно из: `/dev/hdb /dev/hdc /dev/hdd /dev/hde /dev/hdf /dev/hdg /dev/hdh /dev/hda`

Имя устройства

SCSI CD-ROM: Какой SCSI CD-ROM используется? Если не знаете, выберите `/dev/scd0`.

1. /dev/scd0
2. /dev/scd1

Метод установки: С Slackware CD можно установить систему или протестировать ее. Выберите тип установки (slakware или slaktest).

slakware
Нормальная установка на жесткий диск
slaktest
Связать `/usr->/cdrom/live/usr`

slakware или slaktext

40. **Выбор наборов:** Укажите, какие наборы планируется ставить. Вы можете указать любую комбинацию наборов дисков, как указано ниже. Например, для установки основной системы, основных пакетов X Window System и Tcl toolkit, введите: `a x tcl`.

```
41.
42.          A   Base Linux system
43.          AP  Various applications that do not need X
44.          D   Program Development (C, C++, Kernel source, Lisp, Peri,
   etc.)
45.          E   GNU Emacs
46.          F   FAQ lists
47.          K   Linux kernel source
48.          N   Networking (TCP/IP, UUCP, Mail)
49.          Q   Extra kernels with special drivers (needed for non-SCSI
   CD)
50.          T   TeX
51.          TCL Tcl/Tk/TclX, Tel language, and Tk toolkit for developing
   X apps
52.          X   Xfree86 Base X Window System
53.          XAP X Windov; applications
54.          XD  Xfree86 X11 server development system
55.          XV  Xview (OpenLooh virtual Wiindow Manager, apps)
56.          Y   Games (that do not require X)
57.
```

Любую комбинация a ap d e f k n q t tcl x xap xd xv y и других дисковых наборов, разделенных пробелами.

58. **Установка программ:** Пакеты программ будут копироваться на Ваш диск. В процессе этого можно использовать режим PROMPT, в котором будут выдаваться сообщения о том, что происходит и запросы, надо ли ставить остальные программы. Если данный режим не используется, все будет просто установлено молча. Использовать режим PROMPT (y/n)?

[y]es или [n]o

Какие пакеты реально будут ставиться может быть настроено в файле `TAGFILE`, который есть на первом диске каждого набора. Кроме него есть еще исходный файл с именем `TAGFILE.ORG`, предназначенный для восстановления настроек по умолчанию. Файл содержит все команды для установки пакетов. Можно использовать свой файл настройки со своим расширением. Вы можете указать расширение, состоящее из ``." и 3 символов после нее, например `tgz`. Например, я указываю ``.pat", и если найден файл ``tagfile.pat" используется он, а не ``tagfile". Если установщик не может найти файл с заданным расширением, он использует файл со значениями по умолчанию. Введите нужное расширение (оно должно начинаться с ``.") или нажмите Enter для продолжения без спецрасширения.

Tagfile расширение Enter

59. **Extra Configuration:** Здесь можно перенастроить оборудование, создать boot-дискету и поставить LILO. Если Вы поставили новое ядро, эти шаги надо выполнить.

    [y]es или [n]o

60. **Создание Boot-диска:** Рекомендуется создать boot-диск. Создать?

    [y]es или [n]o

    Вставьте в загрузочный дисковод чистый отформатированный диск. Из него будет сделан Linux boot-диск. Все данные с него будут стерты. Вставьте диск и нажмите Enter, или нажмите s, если хотите пропустить этот шаг.

    Enter или [s]kip

61. **Настройка модема:** Будет создана ссылка в `/dev` от устройства `cua0`, `cua1`, `cua2` или `cua3`) на `/dev/modem`. Вы можете позже ее поменять, если модем переставлен на другой порт. Создать ссылку?

    [y]es или [n]o

    К какому порту подсоединен модем (0, 1, 2, 3)?

```
(0) /dev/ttySO (DOS COM1)
(1) /dev/ttySl (DOS COM2)
(2) /dev/ttyS2 (DOS COM3)
(3) /dev/ttvS3 (DOS COM4)
```

    0, 1, 2 или 3

62. **Настройка мыши:** Будет создана ссылка в `/dev` на устройство `/dev/mouse`. Вы можете позже ее поменять, если мышь переставлена на другой порт или сменился ее тип. Создать ссылку?

    [y]es или [n]o

    Какой тип имеет Ваша мышь (1, 2, 3, 4, 5, 6, 7)?

```
(a) Microsoft compatible serial mouse
(b) QuickPort or PS/2 style mouse (Auxiliary port)
(c) Logitech Bus Mouse
(d) ATI XL Bus Mouse
(e) Microsoft Bus Mouse
(f) Mouse Systems serial mouse
(g) Logitech (MouseMan) serial mouse
```

    1, 2, 3, 4, 5, 6 или 7

К какому порту подсоединена мышь?

```
(0) /dev/ttySO (DOS COM1)
(1) /dev/ttySl (DOS COM2)
(2) /dev/ttyS2 (DOS COM3)
(3) /dev/ttvS3 (DOS COM4)
```

0, 1, 2 или 3

63. **Настройка сети:** Сейчас надо настроить почту и TCP/IP. Данная настройка работает не во всех случаях, но дает неплохое начало. Настройки сети всегда можно поменять командой `netconfig`. Сначала нужно имя Вашей машины (хоста). Только само имя (без домена).

    Имя хоста

    Теперь нужно имя домена без точки в начале.

    Имя домена

    Если планируется использовать только loopback, Ваш IP-адрес будет 127.0.0.1, что сильно упростит настройку. Использовать *только* loopback?

    [y]es или [n]o

    Введите IP-адрес Вашей машины. Пример: 111.112.113.114.

    IP-адрес

    Введите адрес шлюза. Если шлюза нет, вы можете подправить `/etc/rc.d/rc.inet1` позже или иногда работает ввод своего IP-адреса.

    IP-адрес

    Введите маску подсети. Обычно она 255.255.255.0.

    IP-адрес

    Вы будете обращаться к серверу DNS?

    [y]es или [n]o

    Укажите IP-адрес своего сервера DNS. Можно добавлять Domain Name Servers, редактируя файл `/etc/resolv.conf`.

    IP-адрес

Теперь нажмите Ctrl-Alt-Delete и перезагрузите машину. Если Вы поставили LILO, выньте boot-диск перед перезагрузкой. Не забудьте создать файл `/etc/fstab`, если его нет. Подробности о нем см. ниже.

2.6.8 Собственно установка.

Если Вы озаботились подготовить таблицу ответов, изложенную выше, установка не вызовет никаких проблем. Изготовьте дискеты boot и root, загрузитесь с них, ответьте на кучу вопросов и все. Ответы на вопросы приведены в таблице ответов, которую Вы составили.

2.6.9 Создание boot-дисков.

Выберите Ваше ядро!

При установке Slackware Linux, Вы должны создать дискету начальной загрузки с ядром Linux, которое подготовлено для Вашего оборудования. Например, чтобы установить Slackware с IDE CD-ROM на жесткий диск SCSI, ядро, которое Вы помещаете на дискету начальной загрузки должно иметь драйверы для Вашей SCSI платы и IDE CD-ROM.

Ядра хранятся как сжатые двоичные файлы образов, так что Вы можете обращаться к ним из любой операционной системы, чтобы создать дискету начальной загрузки Slackware. На Slackware FTP, CD-ROM или NFS Вы найдете подкаталог, названный `bootdsks.144`, содержащий 1.44 MB образы дискет с ядрами для создания boot-дисков на 1.44MB дискетах высокой плотности. Если Вы работаете с 5.25" дисководом, посмотрите каталог `bootdsks.12` на предмет соответствующих образов дисков.

Таблицы представляют краткий обзор доступных ядер. Новые образы и документация доступны на `ftp://ftp.cdrom.com/pub/linux/slackware/bootdsks.144/README.TXT`

Таблица 2.6: Образы Slackware boot-дисков для IDE.

Файл	Загрузочные дискеты Slackware (интерфейс IDE)
`aztech.i`	Устройства CD-ROM: Aztech CDA268-01A, Orchid CD-3110, Okano/Wearnes CDD110, Conrad TXC, CyCDROM CR520, CR540.
`bare.i`	Поддержка только интерфейса IDE.
`cdu31a.i`	Sony CDU31/33a CD-ROM.
`cdu535.i`	Sony CDU531/535 CD-ROM.
`cm206.i`	Philips/LMS cm206 CD-ROM с адаптером cm260.
`goldstar.i`	Goldstar R420 CD-ROM (иногда продается под названием Reveal "Multimedia Kit").
`mcd.i`	Поддержка не-IDE Mitsumi CD-ROM.
`mcdx.i`	Улучшенная поддержка не-IDE Mitsumi CD-ROM.
`net.i`	Поддержка адаптера Ethernet.
`optics.i`	Оптический дисковод "Дельфин" ("DOLPHIN") Optics Storage 8000 AT CD-ROM
`sanyo.i`	Поддержка Sanyo CDR-H94A CD-ROM.
`sbpcd.i`	Поддержка не-IDE устройств CD-ROM: Matsushita, Kotobuki, Panasonic,

	CreativeLabs (Sound Blaster), Longshine, Teac.
`xt.i`	Поддержка жесткого диска с интерфейсом MFM.

Таблица 2.7: Образы Slackware boot-дисков для SCSI/IDE.

Файл	**Загрузочные дискеты Slackware (интерфейсы SCSI/IDE)**
`7000fast.s`	Поддержка Western Digital 7000FASST SCSI.
`Advansys.s`	Поддержка AdvanSys SCSI.
`Aha152x.s`	Поддержка Adaptec 152x SCSI.
`Aha1542.s`	Поддержка Adaptec 1542 SCSI.
`Aha1740.s`	Поддержка Adaptec 1740 SCSI.
`Aha2x4x.s`	Поддержка Adaptec AIC7xxx SCSI (для устройств: AHA-274x, AHA-2842, AHA-2940, AHA-2940W, AHA-2940U, AHA-2940UW, AHA-2944D, AHA-2944WD, AHA-3940, AHA-3940W, AHA-3985, AHA-3985W).
`Am53c974.s`	Поддержка AMD AM53/79C974 SCSI.
`Aztech.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка устройств CD-ROM: Aztech CDA268-01A, Orchid CD-3110, Okano/Wearnes CDD110, Conrad TXC, CyCDROM CR520, CR540.
`Buslogic.s`	Поддержка Buslogic MultiMaster SCSI.
`Cdu31a.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка устройства CD-ROM Sony CDU31/33a.
`Cdu535.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка устройства CD-ROM Sony CDU531/535.
`Cm206.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка устройства CD-ROM Philips/LMS cm206 с адаптером cm260.
`Dtc3280.s`	Поддержка DTC (Data Technology Corp) 3180/3280 SCSI.
`Eata_dma.s`	Поддержка DPT EATA-DMA SCSI (для устройств типа PM2011, PM2021, PM2041, PM3021, PM2012B, PM2022, PM2122, PM2322, PM2042, PM3122, PM3222, PM3332, PM2024, PM2124, PM2044, PM2144, PM3224, PM3334.)
`Eata_isa.s`	Поддержка DPT EATA-ISA/EISA SCSI (для устройств типа PM2011B/9X, PM2021A/9X, PM2012A, PM2012B, PM2022A/9X, PM2122A/9X, PM2322A/9X).
`Eata_pio.s`	Поддержка DPT EATA-PIO SCSI (PM2001 и PM2012A).
`Fdomain.s`	Поддержка Future Domain TMC-16x0 SCSI.
`Goldstar.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка устройства CD-ROM Goldstar R420 (иногда продается под названием Reveal "Multimedia Kit").
`In2000.s`	Поддержка Always IN2000 SCSI.
`Iomega.s`	Поддержка устройств IOMEGA PPA3, подключенных к параллельному порту SCSI (а также ZIP-дисководов, подключаемых к параллельным портам).
`Mcd.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка стандартного не-IDE устройства CD-ROM Mitsumi.

`Mcdx.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка модернизированного (enhanced) не-IDE устройства CD-ROM Mitsumi.
`N53c406a.s`	Поддержка NCR 53c406a SCSI.
`N_5380.s`	Поддержка NCR 5380b 53c400 SCSI.
`N_53c7xx.s`	Поддержка NCR 53c7xx, 53c8xx SCSI (большинство контроллеров NCR PCI SCSI используют этот драйвер).
`Optics.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка оптического дисковода DOLPHIN, Optics Storage 8000 AT CD-ROM.
`Pas16.s`	Поддержка Pro Audio Spectrum/Studio 16 SCSI.
`Qlog_fas.s`	Поддержка ISA/VLB/PCMCIA Qlogic FastSCSI! (также поддерживает адаптеры Control Concepts SCSI, основанные на микросхеме Qlogic FASXXX).
`Qlog_isp.s`	Поддержка всех контроллеров Qlogic PCI SCSI кроме PCI-basic, которые поддерживаются драйвером AMD SCSI.
`Sanyo.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка устройства CD-ROM Sanyo CDR-H94A.
`Sbpcd.s`	Все контроллеры SCSI, на которые есть поддержка, + поддержка не-IDE устройств CD-ROM Matsushita, Kotobuki, Panasonic, CreativeLabs (Sound Blaster), Longshine, Teac.
`Scsinet.s`	Все контроллеры SCSI, на которые есть поддержка, + полная поддержка адаптеров Ethernet.
`Seagate.s`	Поддержка Seagate ST01/ST02, Future Domain TMC-885/950 SCSI.
`Trantor.s`	Поддержка Trantor T128/T128F/T228 SCSI.
`Ultrastr.s`	Поддержка UltraStor 14F, 24F, 34F SCSI.
`Ustor14f.s`	Поддержка UltraStor 14F, 34F SCSI.

2.6.10 Загрузка в действии.

Теперь загрузитесь с подготовленной boot-дискеты и подождите. Ждать придется долго. Потом система устроит перекрестный допрос, описанный выше, что займет час или около того.

Зайдите в систему как `root` (пароль пока не нужен) и наберите `setup` или `setup.tty`.

2.6.11 Программа Slackware `setup`.

Slackware поставляется с двумя версиями программы `setup`. Одна управляется меню, имеет полноэкранный цветной интерфейс, кнопочки, подсказки и вообще все для чайников. Альтернатива, `setup.tty`, является чисто текстовой версией, которая на самом-то деле лучше, поскольку диалоговые окна не прячут под собой все отладочные данные и сообщения об ошибках. Если аппаратура вызывает сомнения, используйте `setup.tty`.

```
Slackware Linux Setup (version HD-3.1.0)
```

```
Welcome to Slackware Linux Setup.

Hint: If you have trouble using the arrow keys on your keyboard,
you can use '+', '-', and TAB instead. Which option would you like?

HELP        Read the Slackware Setup HELP file
KEYMAP      Remap your keyboard.
MAKE TAGS   Tagfile customization program
TARGET      Select target directory [now: /]
SOURCE      Select source media
DISK SETS   Decide which disk sets you wish to install
INSTALL     Install selected, disk sets
CONFIGURE   Reconfigure your Linus system
PKG TOOL    Install or remove packages with Pkgtool
EXIT        Exit Slackware Linux Setup

            OK
```

2.6.12 Это все?

Определенно нет! В этой точке Вы или имеете некоторое раздражающее препятствие, которое предотвращает установку от завершения, или более вероятно, вы рассматриваете подсказку `root`:

```
darkstar~#
```

и думаете ``Что дальше?"

2.6.13 Поиск неисправностей.

Не всякая установка Slackware соответствует ожиданиям администратора. Я провел не одну ночь, сидя после работы вечером, пытаясь обновить систему Slackware, и стараясь, чтобы к утру все работало к моменту, когда юзвери полезут смотреть свою почту и новости (которых нет, поскольку машина всю ночь апгрейдилась). Этот раздел рассматривает несколько общих проблем установки Slackware, их решения, и где искать дополнительную помощь.

FAQ по установке Slackware.

Patrick Volkerding, создатель Slackware, имел дело со многими вопросами новых пользователей, отвечал и получал новый град вопросов. Чтобы новые пользователи Slackware не спрашивали то же самое вопрос в 5000-ный раз, Патрик доброжелательно создал документацию и включил ее в дистрибутив Slackware. Три файла, которые Вы можете счесть очень полезными: `FAQ.TXT`, `INSTALL.TXT` и `BOOTING.TXT`.

Web-поддержка для Slackware.

Самый простой способ найти документацию по Linux: Linux Documentation Project Home Page. См. выше описание LDP Home Page.

Сейчас специфическая справка по Slackware может быть получена в Internet. Она обычно носит очень узкий характер: как решить определенную проблему.

Группы в Usenet для Slackware.

Иерархия Usenet `comp.os.linux.*` хранит море информации по Linux, не обязательно Slackware-специфической. В настоящее время, 11 отдельных Linux форумов обрабатывают большой объем обсуждения в этой иерархии, которая описана выше.

Списки рассылки для Slackware.

Специфических списков по Slackware нет. Но Вы можете обратиться на `http://www.linux.org` или поспрашивать о хороших списках рассылки в Usenet newsgroups.

Главный сервер списков рассылки по Linux `majordomo@vger.rutgers.edu` . См. выше описание того, как подписаться на списки рассылки данного сервера.

Вы получаете то, что Вы оплачиваете (коммерческая поддержка).

Коммерческая поддержка для Linux доступна у некоторых продавцов CD-ROM и у длинного списка консультантов по Linux, с кем можно входить в контакт через Linux Commercial and Consultants HOWTO:

```
http://sunsite.unc.edu/LDP/HOWTO/Consultants-HOWTO.html
http://sunsite.unc.edu/LDP/HOWTO/Commercial-HOWTO.html
```

2.6.14 Лучи славы.

Не грейтесь в лучах славы! Только поставить Slackware мало, особенно если Ваша машина является общедоступным сервером. Здесь одной программой установки не обойдешься. Ниже будет дано несколько указателей на безопасность и разделение Вашей новой Slackware-системы.

2.6.15 Задумайтесь о переустановке!

Вы только что закончили установку системы, возможно, утомительную. Но не забывайте, что в ряде случаев ее придется переустанавливать, и будьте готовы к этому! На случай неожиданностей имейте загрузочные дискеты. Friedrich Nietzsche сказал:

> A man learns what he needs to know about building his house only after he's finished (Человек поймет, что именно он хотел сделать только когда закончит).

Если, в процессе установки системы, Вы имели некоторые мысли относительно того, как Вы могли бы сделать это по-другому, теперь самое время. Если Ваша Slackware Linux система будет сервером или многопользовательской машиной, такой удобной возможности повторно установить или реконфигурировать систему радикальными способами уже не будет.

2.6.16 Безопасность системы.

Сразу отключитесь от LAN до затыкания дырок.

Slackware небезопасная система (как и вообще все системы...). Хотя Patrick Volkerding старается создать безопасный дистрибутив, несколько неизбежных отверстий в защите скоро станут известны, и заплаты или пути их блокировки скоро станут доступны администраторам систем (и хакерам...). Если Вы установили Slackware из сетевого источника (например, NFS) Вы должны временно отключиться от LAN после успешной установки, на время затыкания известных дырок в защите.

Получение пароля для `root`.

По умолчанию, новый Slackware не будет требовать пароля для пользователя `root`. После того, как Вы убедитесь, что система работает (это вопрос нескольких часов), поставьте пароль на аккаунт root. Зайдите как root и введите команду:

```
# passwd root
```

Заведите себе аккаунт.

Администратор должен всегда иметь два аккаунта: `root` для администрирования системы и какой-нибудь обычный для всего остального. Помните, что пользовательский доступ в систему пригодится для отладки! Бывают случаи, когда у `root` с его неограниченными правами все работает, а у остальных пользователей ничего не работает из-за неправильно выставленных `root`'ом прав доступа. Используйте `/sbin/adduser`, чтобы создать какого-то другого пользователя для повседневной работы и используйте для доступа к администрированию `su`. Я всегда улыбаюсь, когда вижу студентов и энтузиастов, гордо регистрирующихся в Usenet как `root@mymachine.mydomain`.

Почитайте главу 4 для выяснения вопросов о том когда и когда не надо использовать аккаунт `root`.

Блокировка входа как `root`.

Пользователь root обычно заходит в систему так: сначала заходит под своим обычным логином, а уж потом дает команду `su`. Проблема в том, что из сети можно зайти под логином `root`. Дабы не дать хакерам такой возможности, подправьте файл `/etc/securetty` и закомментируйте в нем все (поставив перед соответствующей строчкой знак #), кроме локальных терминалов:

```
console
ttyl
tty2
# ttySO
# ttvSl
```

После этого войти из сети как `root` будет уже нельзя:

```
Linux 2.0.29 (durak interactivate com) (ttyp4)

durak login: root
```

```
root login refused, on this terminal.

durak login:
```

Применение простых исправлений.

Slackware имеет несколько дыр в защите. Прежде чем их найдут и используют, надо их заткнуть. Советы по этому непростому делу есть на сайте *Slackware SimpleFixes*: `http://cesdis.qsfc.nasa.qov/linux-web/simplefixes/simplefixes.html`.

Поиск заплаток на `ftp.cdrom.com`

Обновления и заплатки для Slackware доступны на `ftp://ftp.cdrom.com/pub/linux/slackware/patches`.

Следите за новостями.

Подпишитесь на списки рассылки предупреждений администраторам о свежих проблемах в Linux и способах их устранения:

```
linux-alert-request@tarsier.cv.nrao.edu
linux-security-request@redhat.com
```

2.6.16.1 Резервируйтесь.

Все работает? Отлично, сохраните текущую систему с помощью пакета Amanda (Advanced Maryland Automatic Network Disk Archiver), который является одним из стандартных пакетов резервирования для Linux. Подробнее про пакет Amanda можно узнать по адресу `http://www.cs.umd.edu/projects/amanda/index.html`.

2.7 S.u.S.E.

Глава про S.u.S.E. Linux написана Larry Ayers.

SuSE является дальнейшим развитием Slackware. Patrick Volkerding помог разработчикам SuSE начать проект, но потом появились серьезные отличия в системах. В принципе, данный дистрибутив преследует цель улучшить и упростить Slackware. Не следует удивляться, если в новых версиях Slackware появятся некоторые свойства из S.u.S.E. Linux.

2.7.1 Начало установки.

При загрузке с одной дискеты установки, будет загружена маленькая Linux система, разработанная именно для этой цели. Установка пойдет на цветном экране с красивыми кнопочками под управлением программы YAST (Yet Another Set-up Tool), основанной

на Slackware-программе `dialog`. Данная программа дает возможность скриптам строить красивые окошки с радиокнопками, списками выбора и прочими мелочами для чайников.

Разработчики из S.u.S.E. GmbH смогли обойти ряд общих проблем установки. Например, установка часто сваливается из-за того, что не может опознать CD-ROM. Конечно, копирование пакетов на жесткий диск и установка оттуда выход, но довольно неуклюжий, да и не факт, что на жестком диске хватит места. Попытка представить много ядер с разными драйверами в надежде, что хоть одно из них подойдет, тоже выход, но Вы его оцените, перебрав с десяток ядер. S.u.S.E. использует единый boot-диск, на котором есть базовое ядро, а все необходимые драйверы "для всего" сделаны в виде модулей ядра. Есть kernel daemon, который работает в фоновом режиме и обеспечивает загрузку модулей по мере необходимости. Еще одной проблемой является нехватка памяти. Здесь тоже сделано немало полезного. Скрипты установки написаны так, что их можно прервать в любом месте и возобновить установку именно с него.

При установке дистрибутива SuSE на экране программы YAST постоянно высвечивается текущее значение оставшегося объема дисковой памяти раздела. При выборе пакетов программного обеспечения, которые будут устанавливаться, можно пробовать различные комбинации и исходить при этом из того, какой объем дисковой памяти должен остаться свободным. Процессы создания разделов на диске, а также создания и активации своп-раздела в дистрибутиве SuSE не отличаются от аналогичных им в других дистрибутивах. Эти процессы основаны на одних и тех же средствах, и уже утвердился некоторый стандарт на такие процедуры.

Зависимости.

Среди дистрибутивов системы Linux быстро распространилось использование концепции **зависимости**. Зависимость это информация, включенная в пакет программ и говорящая о том, какие еще пакеты должны быть установлены для того, чтобы данный пакет нормально работал. К сожалению, не создался ни один общий формат пакетов: почти каждый дистрибутив использует свой формат. Мощным и эффективным является формат RPM дистрибутива Red Hat Linux, который используется и в SuSE. S.u.S.E. 5.1 использует формат srpm. Однако, кроме того, YaST умеет работать и с форматом tar.gz. После некоторого периода работы в системе пользователь получает представление об имеющихся в ней библиотеках и программах. Большинство пакетов программного обеспечения для системы Linux обычно содержат информацию о том, какие компоненты должны присутствовать в системе для функционирования этого пакета. Разумно будет прочесть все содержимое файла `rc.config` перед запуском конфигурационного скрипта `SuSEconfig`. Вы можете захотеть проделать вручную некоторые действия, которые скрипт будет делать по умолчанию, однако эти действия можно легко отменить посредством редактирования этого файла.

Пользователи, которые привыкли к расположению инициализационных файлов, принятому в дистрибутиве Slackware, должны будут произвести определенную перенастройку: некоторые файлы, которые обычно находятся в каталоге `/etc/rc.d`, теперь будут в каталоге `/sbin/init.d`.

2.7.2 После установки S.u.S.E.

Программа YAST предназначена также и для задач обслуживания системы. Начинающих пользователей может сбить с толку большое количество файлов, которые описывают ресурсы системы (resource files) и нужны системе Linux для установки и работы. Программа YAST дает возможность доступа к этим файлам (в том числе конфигурационному файлу программы `sendmail`, файлу `cron`, управляющему расписанием работы системы, скриптам инициализации, некоторым файлам для работы в сети) через интерфейс, основанный на системе меню. Изменения, сделанные в процессе сеанса YAST, записываются в единственный файл `rc.config`, находящийся в каталоге `/etc`. Этот файл может быть также отредактирован непосредственно. Потом эти изменения записываются в ``реальные`` конфигурационные файлы с помощью скрипта `SuSEconfig`. Этот скрипт запускается автоматически программой YAST в конце ее сессии; если файл `/etc/rc.config` редактируется непосредственно, то скрипт `SuSEconfig` должен быть запущен вручную. При описании это создает впечатление сложной процедуры, однако она гораздо проще, чем если бы пришлось отслеживать отдельные файлы, изучать точный синтаксис, который в них используется (чтобы эти файлы отредактировать) и добиваться, чтобы они работали как надо.

После того, как дистрибутив SuSE установлен и система проверена в действии, хорошей идеей будет установить исходные тексты ядра системы (они входят в дистрибутив в виде отдельного пакета; на стадии начальной установки системы этот пакет не является обязательным). Дистрибутив SuSE устанавливает общее ядро системы, и потребуется, возможно, лишь несколько дополнительных модулей. Это прекрасная возможность познакомиться с механикой компилирования исходного текста системы, и в конце концов создать ядро меньших размеров, в котором будут реализованы только действительно нужные функции. Для компиляции ядра системы необходим компилятор `gcc`; это средство почти всегда необходимо иметь установленным в системе Linux, даже если пользователь не является программистом. YAST обеспечит, чтобы были установлены все требуемые средства компиляции.

Новичка может страшить процесс компилирования ядра, однако этот процесс весьма интуитивный. На начальном этапе конфигурации доступны три интерфейса. Первый (и самый старый) это скрипт для консоли (console-mode script), вызываемый через команду `make config`. Этот скрипт задает несколько вопросов и, основываясь на ответах, формирует файл, который будет управлять процессом компилирования. Потребуются некоторые факты об аппаратном обеспечении компьютера, например, каков тип жесткого диска или устройства CD-ROM. Если потребуется поддержка звуковой карты, то надо будет узнать номер прерывания (IRQ), которое использует звуковая карта, а также некоторые дополнительные параметры, которые можно найти в инструкции к карте или с помощью утилиты `msd` в системе MS-DOS. Откровенно говоря, лучше использовать его, поскольку компиляция ядра дело важное, и здесь недопустимы ошибки графических пакетов.

Два других интерфейса `menuconfig` и `xconfig`. Первый из них использует модифицированную версию программы `dialog`, упомянутой выше. Эта версия работает на виртуальной консоли или на эмуляторе терминала `xterm` и напоминает средство установки YAST. Интерфейс программы `xconfig` выполнен с помощью средств Tk, а сама программа работает в среде X. Все три программы выполняют одну и ту же задачу, однако две последние позволяют производить меньше действий с клавиатурой. Исходные тексты ядра хорошо документированы. Файл `README` в каталоге верхнего уровня содержит достаточно информации, чтобы обеспечить успешную компиляцию и сборку ядра.

2.7.3 Установка и запуск X.

Правильная настройка X Window System (в частности системы XFree86, которая входит в дистрибутив SuSE и в большинство других дистрибутивов) может оказаться сложной задачей. Существующие типы мониторов и видеоадаптеров настолько многочисленны, что каждая установка X должна быть конфигурирована индивидуально. Проблем несколько поубавилось с версии XFree86 3.2. В новых версиях SuSE Linux для настройки XFree86 есть диалоговая программа. Она основана на `dialog` и попробует распознать имеющийся видеоадаптер и предложит обширный список мониторов, после чего будет построен файл `xf86config`. Правда, знание частот развертки монитора по-прежнему необходимо. К тому же, надо знать чипсет вашей видеоплаты, дабы иметь возможность поправить настройку в случае ошибки автоматического настройщика. Сначала Вы получите "условно работающую" систему, которая работает в минимальном разрешении. После этого нужно постепенно наращивать настройки, чтобы выжать из видеокарты все, на что она способна.

Разработчики дистрибутива SuSE потратили немало усилий на конфигурирование различных диспетчеров окон (window manager), например, `fvwm95`. При первом запуске X многие из приложений, выбранных при установке, будут доступны через меню корневого окна, управляемое мышью. Через другой пункт меню можно изменять фон окна.

В дистрибутиве SuSE имеется много хорошо исполненных пиктограмм. Это дает пользователю-новичку некоторую передышку. После того, как системы Linux и X наконец установлены, предстоит большая работа просто по изучению системы, и большим облегчением будет то, что пользователь не будет чувствовать необходимости изменять внешний вид программы, так чтобы он стал приятен для глаз!

2.7.4 Обновления.

С той минуты, как только установлена пусть даже самая последняя версия дистрибутива, она начинает постепенно устаревать. Этот процесс идет медленно, но в конце концов возникнет необходимость модернизировать некоторые части системы. В версиях SuSE, начиная с 5.1, обновление с помощью YAST возможно по FTP.

2.7.5 Поставили. А дальше?

В данный момент, поскольку система уже используется, полезно объяснить, как следует перезагружать (reboot) и завершать (shutdown) систему, готовя компьютер к выключению или перезагрузке. Никогда не следует делать это нажатием кнопки Reset на системном блоке. Также нельзя просто выключить питание. Linux, подобно большинству систем, основанных на системе UNIX, использует кэширование записи на диск. Следовательно, если вместо предусмотренной "штатной" процедуры останова системы произойдет внезапная ее перезагрузка, часть данных на диске может испортиться, что может стать причиной бесчисленных повреждений.

Самый простой способ выгрузить систему: запустить команду Например, для немедленной выгрузки и перезагрузки системы пользователю `root` надо ввести следующую команду:

```
# shutdown -r now
```

Таким образом система будет завершена корректно. Документация по программе `shutdown` описывает другие параметры, которые могут быть использованы. Для вызова документации надо ввести команду `man shutdown`.

Следует заметить, однако, что многие дистрибутивы системы Linux не содержат команды `shutdown` среди средств установки. Это значит, что при первой перезагрузке после установки системы, может быть, придется воспользоваться комбинацией клавиш Ctrl-Alt-Del.

После того, как система проверена в действии, с ней надо проделать некоторые рутинные операции по конфигурированию. Первой из них будет создание пользователя для повседневной работы (для вас самого и, при необходимости, для других пользователей, которые будут работать в этой системе). Процесс создания пользователей описан в главе 4. Обычно все, что нужно сделать, это войти в систему как `root` и запустить программу `adduser` (или, иногда, `useradd`). В процессе короткого диалога с программой будет создан новый пользователя.

Если в системе Linux используется больше одной файловой системы или если используется своп-раздел, то для того, чтобы эти файловые системы были доступны автоматически сразу после загрузки системы, может потребоваться отредактировать файл `/etc/fstab`. Если для каталога `/usr` используется отдельная файловая система и вдруг оказывается, что в нем нет ни одного файла, то возможно, что эту файловую систему надо просто смонтировать (mount). Описание файла `/etc/fstab` можно найти ниже.

2.9 Борьба с глюками.

Практический каждый влипнет в какую-нибудь историю при первой попытке инсталлировать Linux. В большей части случаев это связано с простым неправильным пониманием. прим. переводчика: То есть связано с привычкой быстро, но неправильно схватывать мысли. Но иногда может быть кое-что и посерьезнее, как зевок проектировщиков или просто ошибка.

Этот раздел описывает некоторые наиболее часто встречающиеся проблемы инсталляции и как их решать. Если инсталляция прошла успешно, но вы получили неожиданные сообщения об ошибках, они здесь также описываются.

2.9.1 Проблемы загрузки средств инсталляции.

Пытаясь первый раз загрузить средства инсталляции, вы можете столкнуться с множеством проблем. Они перечислены ниже. Заметим, что следующие проблемы не относятся к загрузке вашего вновь инсталлированного Linux. Относительно таких проблем см. ниже.

- **Ошибка дискеты или средства инсталляции.**

  Наиболее часто встречающийся случай при такого рода проблемах это запорченная загрузочная дискета. Либо дискета физически повреждена, тогда вы должны восстановить диск, используя исправную дискету, либо испорчены данные на дискете, в этом случае следует проверить правильность перенесения данных на дискету. Во многих случаях вам может помочь простая перезапись дискеты. Прим. переводчика: Однажды я столкнулся с еще одними граблями: когда Вы читаете "вставьте чистую отформатированную дискету", понимать все надо буквально! То есть, если на дискету записали некий файл, а потом его стерли, будьте добры ее отформатировать. Да, файл стерт, копирование образа ядра на дискету пройдет без накладок, но загрузиться с не получится.

  Если вы получили загрузочную дискету по почте или от какого-то другого дистрибутора, вместо самостоятельных попыток восстановить испорченную дискету свяжитесь с дистрибутором и попросите новую загрузочную дискету, но только окончательно убедившись, что именно в дискете причина.

- **Система зависает во время или сразу после загрузки.**

  После инсталляции средств загрузки вы увидите сообщения ядра, указывающие, какие устройства были распознаны и конфигурированы. После этого обычно выдается "login", позволяющий продолжать инсталляцию (некоторые дистрибутивы вместо этого помещают вас в некоторого рода инсталляционную программу). Система может зависнуть во время этих шагов. Во многих случаях система не зависает, а просто требует много времени на выполнение. Так что, прежде чем решить, что система зависла, убедитесь, что по крайней мере несколько минут дисковод и процессор бездействуют.

1. После загрузки с помощью `LILO`, система должна загрузить образ ядра с дискеты. Это может занять несколько секунд; если горит при этом лампочка обращения к дисководу, то это значит, что все идет нормально.
2. При загрузке ядра SCSI устройства должны быть проверены. Если у вас еще не было инсталлировано какого-нибудь SCSI устройства, система "зависнет" секунд на 15, пока происходит проверка SCSI устройства; обычно это происходит после строки

   ```
   3.
   4.      lp_init: lp1 exists (0), using polling driver
   5.
   ```

   появившейся на вашем экране.

6. После окончания загрузки ядра управление передается системе, загружающей файлы с дискеты. Затем вам будет выдана подсказка `login` или система выйдет в инсталляционную программу. Если вы дошли до подсказки, например, имеющей вид

   ```
   7.
   8.      Linux login:
   9.
   ```

далее вы должны войти (обычно как `root` или `install`: в разных версиях дистрибутивов по-разному). После введения имени пользователя система может задуматься секунд на 20 или более, пока программа инсталляции или shell загружается с дискеты. Опять же лампочка дисковода должна гореть. Так что не думайте, что система опять зависла.

Любой из перечисленных выше пунктов может быть источником проблем. Разумеется, система может и взаправду зависнуть при загрузке, чему может быть несколько причин. Прежде всего, у вас может быть недостаточно памяти (RAM) для загрузки средств инсталляции.

Причина многих системных зависаний - аппаратная несовместимость. Даже если ваша аппаратура поддерживается, у вас могут быть проблемы, связанные с несовместимостью конфигурации оборудования, которые тоже могут быть причиной зависания. Смотрите ниже обсуждение вопросов аппаратной несовместимости.

10. **Системные сообщения об ошибках памяти в процессе инсталляции.**

    Этот пункт относится к количеству памяти, которая имеется в вашем распоряжении. На системе с 4M RAM или менее у вас могут быть проблемы с самой загрузкой средств инсталляции. Это потому, что многие дистрибутивы используют ``ramdisk", которая является файловой системой, загружаемой прямо в RAM во время операций, использующих средства инсталляции. Полный образ инсталляционной дискеты, например, может быть загружен на ramdisk, что может потребовать более мегабайта памяти.

    Решение этой проблемы: подготовить опцию ramdisk при загрузке средств инсталляции. Каждая версия имеет процедуры реализации этого; в версии SLS, например, вы печатаете ``floppy", когда появится подсказка LILO при загрузке диска "a1". Детали посмотрите в документации на дистрибутив.

    Имейте в виду, что Linux сам по себе требует не менее 2 M RAM для минимального функционирования; некоторые версии требуют наличия 4M и даже более.

11. **Система сообщает об ошибках, таких как ``permission denied" (обращение запрещено) или ``file not found"(файл не найден) в процессе загрузки.**

    Это говорит о том, что средства инсталляции неисправны. Если вы попытаетесь загрузиться со средств инсталляции (и вы уверены, что все делаете правильно), то у вас не должно появляться сообщений, вроде вышеупомянутых. Свяжитесь с дистрибутором вашего Linux и обсудите с ним проблему. Может быть нужна новая копия. Если вы переписали

загрузочный диск сами, попробуйте пересоздать этот загрузочный диск, может это решит проблему.

12. **Система при загрузке выдает сообщение ``VFS: Unable to mount root''.**

    Это сообщение об ошибке означает, что корневая файловая система (сама находящаяся на средстве загрузки) не может быть найдена. То ли ваши средства загрузки каким-то образом испорчены, то ли вы неправильно пытаетесь загружать систему.

    Например, многие дистрибутивы на CD-ROM требуют, чтобы при загрузке диск находился в дисководе. Убедитесь также, что дисковод CD-ROM включен и что-то делает. Подробнее см. ниже.

2.9.2 Проблемы с оборудованием.

Наиболее общий случай, когда инсталляция или использование Linux приходят в противоречие с аппаратурой. Даже если вся ваша аппаратура поддерживается Linux, неправильное конфигурирование или конфликты между отдельными устройствами могут иногда приводить к странным результатам: устройства могут не распознаваться на этапе загрузки или система может зависать.

Важно локализовать эти аппаратные проблемы, если вы подозреваете, что именно они являются источником ваших неприятностей. В последующих разделах мы опишем некоторые общие проблемы, связанные с аппаратурой, и как решать их.

Локализация аппаратных проблем.

Если вы столкнулись с проблемой, которая по вашему мнению носит аппаратный характер, первое, что вы должны сделать, это попытаться локализовать проблему. Это означает, что исключая все возможные составляющие и (обычно) саму операционную систему, вы постепенно шаг за шагом выделяете неисправную часть аппаратуры.

Это не так тяжело, как иногда может казаться. Первоначально вы должны отключить от системы все несущественное оборудование, а затем определить, какое устройство в действительности является источником неприятностей, подключая шаг за шагом устройства. Это означает, что вы должны отключить все устройства кроме контроллеров гибкого диска и видео, а также, разумеется, клавиатуры. Даже такие невинные на первый взгляд устройства, как мышь, могут внести большую сумятицу в ваши мозги.

Например, предположим, что система зависла во время загрузки при распознавании платы Ethernet. Вы можете предположить, что имеет место конфликт или это проблема данной платы Ethernet. Простой и быстрый способ определиться - это вытащить плату

Ethernet и попытаться вновь загрузиться. Если все пойдет нормально, то (a) плата Ethernet не поддерживается Linux (см. выше относительно совместимых плат), или (b) существует адрес или IRQ, конфликтующие с платой.

``Address or IRQ conflict?" (``Конфликт адреса или IRQ ?"). А это-то, скажите на милость, что еще может значить? Все устройства в вашей машине используют IRQ (прим. переводчика: IRQ Interrupt ReQuest) или линию запросов прерывания, чтобы сообщить системе, что система должна для них что-то сделать. Вы можете представить себе IRQ как веревочку, за которую устройство дергает, когда ему надо, чтобы система позаботилась о выполнении какого-то поступившего запроса. Если более, чем одно устройство дергает за одну веревочку, ядро не способно определить, какое устройство нуждается в обслуживании. Вот вам и глюк.

Поэтому убедитесь, что все инсталлированные вами устройства используют уникальные линии IRQ. В общем случае IRQ для устройства может быть установлен с помощью переключения джамперов (jumpers) на плате; детали смотрите в документации на конкретное устройство. Некоторые устройства вообще не используют IRQ, но предполагается, что вы конфигурировали их, так, что они смогут им воспользоваться. Хорошие примеры тому контроллеры Seagate ST01 и ST02 SCSI.

В некоторых случаях ядро, находящееся на ваших средствах инсталляции, конфигурируется для использования конкретного IRQ для конкретного устройства. Например, в некоторых дистрибутивах Linux ядро предварительно сконфигурировано так, чтобы использовать IRQ 5 для контроллера TMC-950 SCSI, контроллера CD-ROM Mitsumi и драйвер мыши busmouse. Если вы хотите использовать два или более из этих устройств, вам необходимо будет вначале инсталлировать Linux только с одним из этих устройств, подключенным к системе, а затем перекомпилировать ядро, чтобы сменить IRQ, выделенное для другого из них по умолчанию. См. главу 4 п о поводу перекомпиляции ядра.

Другая область, где могут возникнуть конфликты аппаратуры, это каналы DMA (Direct Memory Access, каналы прямого доступа к памяти), адреса ввода-вывода (I/O) и адреса разделяемой памяти (shared memory addresses). Все вышеперечисленное есть механизмы, через которые система взаимодействует с различными устройствами. Некоторые платы Ethernet, например, используют разделяемую память также как и IRQ в качестве интерфейса с системой. И если они конфликтуют с другими драйверами, система ведет себя непредсказуемо. Вы должны быть готовы к изменению канала DMA, адресов ввода-вывода или разделяемой памяти для различных устройств с помощью переустановки джамперов. К сожалению, некоторые устройства не позволяют сделать такие переустановки.

Документация на различные устройства должна указывать IRQ, канал DMA, адрес ввода-вывода или адрес разделяемой памяти, которые используют устройства, и как их конфигурировать. И опять, простейший способ справиться с этой проблемой, это просто временно отключить конфликтующие устройства до того, как вы определите причину конфликта.

Таблица 2.8 представляет перечень IRQ и каналов DMA, используемых различными "стандартными" устройствами, стоящими во многих системах. Практически все системы имеют эти устройства, так что вам следует избегать установок IRQ и DMA других устройств на эти значения.

Device	I/O address	IRQ	DMA
ttySO (COM1)	3f8	4	n/a
ttySl (COM2)	2f8	3	n/a
ttyS2 (COM3)	3c8	4	n/a
ttyS3 (COM4)	2c8	3	n/a
Ip0 (LPT1)	378 - 37f	7	n/a
Ipl (LPT2)	278 - 27f	5	n/a
fdO, fdl (lappirs 1 and 2)	3fD-3f7	6	2
fd2, fd3 (lappirs 3 and 4)	37D-377	10	3

Таблица 2.8: Распространенные параметры устройствs.

Проблемы распознавания жесткого диска или контроллера.

При загрузке Linux вы увидите серию сообщений, выдаваемых на экран, вроде:

```
Console colour EGA+ 50x25, 8 virtual consoles
Serial driver Version 3.95 with no serial options enabled.
tty00 at Ox03f8 (irq = 4) is a 16450
tty03 at Ox02e8 (irq = 3) is a 16550A
Ip_init: Ipl exists (0), using polling driver
```

Здесь ядро распознает различные устройства, имеющиеся в системе. В некоторый момент вы увидите строчку:

```
Partition check:
```

(проверка раздела), за которой следует список распознанных разделов, например:

```
Partition check:
hda:  hda1 hda2
hdb:  hdb1 hdb2 hdb3
```

Если по какой-то причине ваши дисководы или разделы не распознаны, вы никаким образом не сможете к ним добраться.

Это может произойти по нескольким причинам:

- **Жесткий диск или контроллер не поддерживается.** Если Вы используете контроллер жесткого диска (IDE, SCSI и тому подобные), из тех, которые не поддерживаются в Linux, ядро не распознает ваш раздел на этапе загрузки.
- **Жесткий диск или контроллер неправильно конфигурированы.** Даже если ваш контроллер поддерживается в Linux, он может быть неправильно конфигурирован. Особенно эта проблема характерна для контроллеров SCSI;

большинство не-SCSI контроллеров будет хорошо работать без дополнительной конфигурации.

Для решения такого рода проблем обращайтесь к соответствующей документации на жесткие диски и/или контроллеры. В частности, многие жесткие диски потребуют переустановки джамперов, если они будут использоваться в режиме "подчиненного" (``slave") драйвера (например, в качестве второго жесткого диска). Самый железный способ проверить наличие такой ситуации, это загрузить MS-DOS или еще какую-нибудь другую операционную систему, которая заведомо должна работать с этим жестким диском и контроллером. Если вы получите доступ к диску и контроллеру из другой операционной системы, то проблемы не в конфигурировании аппаратуры.

См. выше по поводу разрешения возможных конфликтов устройств и ниже по поводу конфигурирования SCSI-устройств.

- **Контроллер конфигурирован правильно, но не распознается.** Некоторые без-BIOS-ные SCSI-контроллеры требуют от пользователя описания контроллера на этапе загрузки. Ниже описывается, как осуществить определение этих контроллеров.
- **Не распознается геометрия жесткого диска.** Некоторые системы, такие, как IBM PS/ValuePoint, не помещают информацию о геометрии жесткого диска в память CMOS, где Linux ожидает ее найти. Также, некоторым SCSI-контроллерам надо сообщать, где найти геометрию диска, чтобы Linux мог распознать формат вашего диска.

  Многие дистрибутивы имеют загрузочную опцию для описания геометрии диска. В общем случае, при загрузке средств инсталляции, вы можете описать геометрию драйвера в ответ на подсказку загрузчика LILO с помощью команды, например:

```
boot: linux hd=<cylinders>,<heads>,<sectors>
```

  где <cylinders>, <heads> и <sectors> соответствуют числу цилиндров, головок и секторов на трек у вашего диска.

  После инсталляции Linux вы будете иметь возможность инсталлировать LILO, который позволит вам загружаться с жесткого диска. В это время вы можете описать геометрию для инсталляционной процедуры LILO, что позволит не вводить геометрию при каждой загрузке. Более подробно о LILO смотрите в главе 4.

Проблемы со SCSI-контроллерами и устройствами.

Здесь описываются некоторые из наиболее типичных проблем, возникающих со SCSI-контроллерами и устройствами, такими, например, как CD-ROM, жесткие диски и ленты. Если у вас проблемы заставить Linux распознавать диск или контроллер, читайте дальше.

Linux SCSI HOWTO (см. приложение А) имеет много ценной информации о таких SCSI-устройствах, в дополнение к перечисленным здесь. Иногда требуется почти акробатическая ловкость при конфигурировании SCSI.

- **SCSI-устройство распознается всеми возможными идентификаторами (ID).** Это связано с привязкой устройств к одному и тому же адресу с контроллером. Вам следует изменить установку переключателей так, чтобы драйвер и контроллер использовали различные адреса.

- **Linux сообщает об обнаруживаемых ошибках, хотя известно, что устройство работает безошибочно.** Это может происходить из-за плохого кабеля или плохого разъема. Если ваша SCSI-шина не имеет надежных контактов с обеих сторон, может возникать ошибка доступа к SCSI-устройствам. Если у вас возникают сомнения, всегда проверяйте кабель.

- **SCSI-устройства сообщают об ошибках истечения времени.** Это обычно происходит из-за конфликтов IRQ, адресов DMA или устройств. Следует проверить также, что прерывания вашим контроллером обрабатываются корректно.

- **SCSI-контроллеры, использующие BIOS не идентифицируются.** Распознавание контроллеров, использующих BIOS, потерпит неудачу, если BIOS отключен или "подпись" вашего контроллера не распознается ядром. Дополнительную информацию можно найти в Linux *SCSI HOWTO* в приложении А.

- **Контроллеры, использующие отображаемый в память ввод-вывод, не работают.** Это происходит, когда порты отображаемого в памяти ввода-вывода буферизируются некорректно. Или определите в установках XCMOS адресное пространство контроллера, как некэшируемое, или отключите также и кэш.

- **При разбиении на разделы будет выдано сообщение, что ``cylinders > 1024'' или что вы не сможете загрузиться из раздела, имеющего номера цилиндров более 1023.** BIOS ограничивает число цилиндров числом 1024 и любой раздел, использующий большие номера цилиндров, будет неприемлем с точки зрения BIOS. Применительно к Linux это касается только загрузки; после того, как система загружена, вы сможете обращаться к разделу. Вы можете выбирать, загружать ли Linux с дискеты или из раздела, использующего цилиндры с номерами меньше 1024.

- **CD-ROM или другие устройства, которые могут дополнительно вставляться (удаляться) в компьютер, не распознаются на этапе загрузки.** Постарайтесь загрузиться с подключенным CD-ROM (или диском). Для некоторых устройств это необходимо.

Если ваш SCSI-контроллер нераспознан, возможно вам следует инициировать (force) распознавание аппаратуры на этапе загрузки. Это особенно важно для без-BIOS-ных SCSI-контроллеров. Большинство дистрибутивов позволяет описывать IRQ контроллеров и адресов разделяемой памяти во время загрузки средств инсталляции. Например, если вы используете контроллер TMC-8xx, вы можете ввести:

```
boot: linux tmx8xx=<interrupt>,<memory-address>
```

в ответ на подсказку загрузчика LILO, где <interrupt> IRQ контроллера и <memory-address> адрес разделяемой памяти. Сможете ли Вы это сделать, зависит от используемого вами дистрибутива Linux, так что относительно деталей посмотрите документацию.

2.9.3 Проблемы инсталляции программ.

Предполагается, что инсталляция программ Linux должна проходить без особых хлопот, если вы счастливый человек. Единственные проблемы, с которыми вы можете столкнуться, это испорченные средства инсталляции или отсутствие достаточного места на файловой системе Linux. Вот перечень наиболее характерных проблем:

- **Системные сообщения ``Read error'' (ошибка чтения), ``file not found'' (не найден файл) или другие ошибки во время попытки инсталлировать программы.** Это говорит о проблемах с вашими средствами инсталляции. Если вы инсталлируете с дискеты, имейте в виду, что дискеты очень склонны к такого рода недостаткам. Убедитесь, что вы используете новые исправные и свежеотформатированные дискеты. Если у вас есть на диске разделы MS-DOS, многие дистрибутивы Linux позволят вам инсталлировать с жесткого диска. Это может быть быстрее и более надежно, чем использование дискет. Встречается правда и разгильдяйство авторов дистрибутива...

  Если вы используете CD-ROM, убедитесь в отсутствии на нем царапин, пыли или других гадостей, которые могут приводить к ошибкам.

  Причиной может быть и то, что соответствующее средство инсталляции имеет неподходящий формат. Например, при использовании дискет многие дистрибутивы Linux требуют, чтобы дискета была отформатирована в формате high-density MS-DOS. Загрузочная дискета исключение; в большинстве случаев она вообще не в формате MS-DOS. Если все прочее потерпело неудачу, либо достаньте новый набор дискет с дистрибутивом или перепишите его на новые дискеты, если вы скачали дистрибутив откуда-то.

- **Системные сообщения вроде ``tar: read error'' (tar: ошибка чтения) или ``gzip: not in gzip format'' (gzip: не в формате gzip).** Часто это связано с испорченными файлами на средствах инсталляции. Другими словами, ваши дискеты могут быть нормальными, но вот данные на них каким-то образом испорчены. Например, вы каким-то образом скачали программы Linux, используя текстовый (а не бинарный) режим, тогда ваши файлы уж точно будут негодными для инсталляции.
- **Системные сообщения об ошибках, такие как ``device full'' (устройство заполнено) в процессе инсталляции.** Это верный признак того, что вы вышли за пределы отведенного пространства при инсталляции. Не все дистрибутивы способны с этим разобраться; вы не сможете прервать инсталляцию и вынуждены дождаться, когда система сама остановится.

Обычное решение в этой ситуации: пересоздание файловой системы (с помощью команды `mke2fs`), которая удаляет частично инсталлированные программы. Вы далее можете попытаться переинсталлировать программы, выбирая на этот раз меньшее количество программ, подлежащих инсталляции. В других случаях вам может понадобиться начать с полного удаления и перераспределения разделов и размеров файловой системы.

- **Системные сообщения об ошибках, такие как ``read_intr: 0x10'' при обращении к жесткому диску.** Это обычно говорит о наличии плохих блоков на диске. Однако, если вы получили это сообщение во время выполнения `mkswap` или `mke2fs`, причиной этого могло быть то, что система имела проблемы с доступом к вашему диску. Это может быть как проблема аппаратуры (см. выше, так и результат неправильного описания геометрии. Если вы применяли опцию

  ```
  hd=<cylinders>,<heads>,<sectors>
  ```

  при загрузке, чтобы инициировать определение геометрии своего жесткого диска и описали геометрию некорректно, то вы должны будете познакомиться с этой проблемой. Это также может случиться, если геометрия вашего драйвера описана некорректно в CMOS.

- **Системные сообщения об ошибках, вроде ``file not found'' или ``permission denied''.** Это может случиться, если не все необходимые файлы представлены на средствах инсталляции (смотрите следующий раздел) или существует проблема разрешения доступа. Например, про некоторые дистрибутивы Linux известно, что они сами по себе содержат ошибки. Это обычно обнаруживается очень быстро, да и случается не часто. Если вы подозреваете, что программы дистрибутива содержат ошибки и уверены, что вы ничего не сделали неправильно, свяжитесь с сопровождающими дистрибутив и сообщите об ошибке.

Если у вас появляются другие странные ошибки во время инсталляции Linux (особенно если вы сами переписали где-то эти программы), убедитесь, что вы действительно списали все необходимое. Например, некоторые используют команду FTP:

```
mget *.*
```

для скачивания программ Linux через FTP. Она скачает только те файлы, которые содержат ``." в именах файлов; если есть файлы без ``.", вы их не получите. В этом случае уместной была бы команда:

```
mget *
```

Самый лучший совет: заново пересмотреть все шаги, которые вы совершили, если у вас застопорилось дело. Вы можете наивно думать, что вы все делали правильно, когда на самом деле вы забыли сделать какой-то маленький, но важный шаг, где-то посреди

нелегкого пути инсталляции. Во многих случаях даже сама попытка заново переписать или заново инсталлировать Linux может натолкнуть на решение действительной проблемы. Не надо биться головой об стену дольше, чем надо!

Кроме прочего, если Linux завис при инсталляции, причины могут быть в аппаратуре. Смотрите по этому поводу выше.

2.9.4 Проблемы после инсталляции Linux.

Вы потратили целых полдня, инсталлируя Linux. Чтобы выделить под него место, вы стерли свои разделы с MS-DOS и OS/2 и не без утирания слез стерли свои копии "SimCity" и "Wing Commander"... Вы перезагрузили систему, а ничего не произошло. Или еще хуже того, *что-то* произошло, но не то, что должно было произойти. Ну и что делать?

Выше мы обсуждали некоторые из наиболее типичных проблем, возникающих при загрузке Linux со средств инсталляции. Многие из этих проблем могут быть и здесь. В довершение ко всему вы можете стать жертвой одной из следующих напастей.

Проблемы загрузки Linux с дискеты.

Если вы используете дискеты для загрузки Linux, вам может потребоваться описать местоположение вашего корневого раздела linux во время загрузки. Это обычно случается, когда вы используете исходную инсталляционную дискету, а не специальную загрузочную дискету, созданную в процессе инсталляции.

При загрузке дискеты, надо держать shift или ctrl. Это приведет вас к загрузочному меню; нажмите tab, чтобы получить список доступных опций. Например, многие дистрибутивы позволяют ввести:

```
boot: linux hd=<partition>
```

где *partition* задает имя корневого раздела Linux, например, `/dev/hda2`. Более детально с вопросом можно познакомиться по документации на дистрибутив.

Проблемы загрузки Linux с жесткого диска.

Если вам удалось инсталлировать LILO, вместо создания загрузочной дискеты вам следует загружать Linux с жесткого диска. Однако, автоматизированная процедура инсталляции LILO, используемая во многих дистрибутивах, не всегда безупречна. Она может сделать неправильные предположения относительно формата вашего раздела, в этом случае вы должны будете переинсталлировать LILO, чтобы все стало хорошо. Инсталляция LILO обсуждается в главе 4.

- **Системные сообщения `` `Drive not bootable---Please insert system disk.'' `` ("Устройство незагружаемо---Пожалуйста, вставьте системный диск"). Вы получите такое сообщение об ошибке, если главная загрузочная запись жесткого диска каким-то образом попорчена. Во многих случаях это безопасно и все остальное у вас на диске по-прежнему в порядке. Тут дальше есть несколько путей.
    1. При разбиении диска на разделы с использованием **fdisk** вы могли удалить раздел, который был отмечен как ``active". MS-DOS и другие операционные системы пытаются загрузить такой раздел на этапе загрузки (Linux не обращает внимания на то, является раздел ``active" или нет). Вы можете загрузить MS-DOS с дискеты и запустить **FDISK** для установки флага ``active" для раздела MS-DOS и все будет хорошо.

       Другая команда, которую можно попробовать (с MS-DOS 5.0 и выше) это:

       ```
       FDISK /MBR
       ```

       Эта команда будет пытаться заново сформировать главную загрузочную запись диска для загрузки MS-DOS, переписывая LILO. Если у вас больше нет на жестком диске MS-DOS, вам потребуется загрузить Linux с дискеты и в последующем попытаться инсталлировать LILO.

    2. Если вы создали раздел MS-DOS, используя версию команды **fdisk** из Linux или наоборот, это может быть причиной ошибки. Вам следует создавать разделы для MS-DOS, используя только версии **FDISK** для MS-DOS. (Это относится и к другим операционным системам, которые существуют наряду с MS-DOS). Здесь лучшее решение, либо начать с того, что все стереть и переразбить диск правильно, либо удалить и пересоздать плохие разделы, используя исправные версии **fdisk**.

    3. Инсталляционная процедура LILO может потерпеть неудачу. В этом случае вам следует либо загрузиться с загрузочной дискеты для Linux (если она у вас есть) или с исходного средства инсталляции. В любом случае вы будете иметь возможность для описании корневого раздела Linux, который будет использован при загрузке. Нажмите shift или ctrl во время загрузки и нажмите tab из меню загрузки, чтобы получить список опций.
- **При загрузке системы с жесткого диска MS-DOS (или другая из существующих операционных систем) стартует вместо Linux.** Прежде всего убедитесь, что вы действительно инсталлировали LILO при инсталляции программ Linux. Если это оказалось не так, система будет загружать MS-DOS (или какую-нибудь другую операционную систему из собранных вами), когда вы пытаетесь загрузиться с жесткого диска. Для того, чтобы загрузить Linux с

жесткого диска, вам необходимо инсталлировать на жесткий диск LILO (см. главу 4).

С другой стороны, если вы *все-таки инсталлировали* LILO, но другая операционная система загружается вместо Linux, то необходимо конфигурировать LILO так, чтобы она загружала другие операционные системы по умолчанию. Во время загрузки системы держите нажатой клавишу shift или ctrl, а затем нажмите tab в ответ на подсказку загрузчика. В результате вы получите список операционных систем, которые можно загрузить. Выберите соответствующую опцию (часто просто ``linux''), чтобы загрузить Linux.

Если вы хотите сделать Linux системой, загружаемой по умолчанию, вам необходимо реинсталлировать LILO. Смотрите главу 4.

Возможно также, что вы пытались инсталлировать LILO, но инсталляция потерпела неудачу. Смотрите предыдущий пункт.

Проблемы входа в систему.

После загрузки Linux на экран должна быть выдана подсказка вроде этой:

```
linux login:
```

В этот момент либо документация на дистрибутив, либо сама система скажут вам, что делать дальше. В большинстве дистрибутивов вы просто войдете в систему под именем `root` без пароля. Другие возможные имена для входа `guest` или `test`.

Большинство новоиспеченных систем Linux не требуют пароля для первоначального входа. Но если система потребует с вас пароль, могут возникнуть проблемы. Прежде всего попробуйте пароль, совпадающий с именем входа; например, если вы вошли как `root`, попробуйте ``root'' в качестве пароля.

Если вы все-таки не можете войти, то это уже проблема. Прежде всего проконсультируйтесь с документацией на дистрибутив. Может быть там где-то закопано правильное имя входа и пароль. Имя входа и пароль могут быть вам сообщены системой во время инсталляции или выведены на экран в виде подсказки.

Причиной этих неприятностей также могут быть проблемы с самой инсталляцией файлов, отвечающих за вход и инициализацию. Если в этом причина, вам может потребоваться переинсталляция (как минимум части) программ Linux или нужно загрузить ваши средства инсталляции и попытаться решить проблемы "вручную". Смотрите соображения на этот счет в главе 4.

Все в порядке, а не работает.

Если вход в систему прошел успешно, на экран будет выдана подсказка "shell", командной оболочки (например ``#'' или ``$'') и вы можете немножко поплясать вокруг системы. Но существует ряд проблем, которые могут возникнуть в начале использования системы.

Одна из наиболее типичных начальных проблем, связанных с конфигурированием, установка неверных прав доступа (защиты) файлов и каталогов. Это может выразиться в сообщении:

```
Shell-init: permission denied
```

которое будет напечатано после входа в систему (на самом деле, всегда, когда вы столкнетесь с сообщением ``permission denied'' ("обращение запрещено") вы можете быть с высокой вероятностью уверены, что это проблема защиты файлов).

Во многих случаях это простое дело для команды chmod, которая может менять права доступа к соответствующим файлам и каталогам. Например, некоторые дистрибутивы Linux использовали (ошибочный) права доступа файлов 0644 для корневого каталога (/). А следует использовать команду:

```
# chmod 755 /
```

от имени root. Но, чтобы ввести эту команду, вы обычно должны загрузиться со средства инсталляции и примонтировать вашу корневую файловую систему Linux вручную, заковыристая задача для большинства новичков.

Во время эксплуатации системы вы можете попадать в места, где неверно установлена защита файлов и каталогов или программы работают не так, как конфигурировались. Добро пожаловать в мир Linux! Хотя многие дистрибутивы и не доставляют особых хлопот, лишь немногие из них безупречны. Мы не хотим обсуждать здесь все возможные проблемы. Вместо этого по ходу всей книги мы помогаем вам решать многие проблемы конфигурирования, обучая вас обнаруживать и решать проблемы самостоятельно. В главе 1 мы детально обсуждали эту философию. В главе 4 мы даем советы относительно решения многих из типовых проблем конфигурирования.

---

3 Учебник по Linux

3.1 Введение.

Новые пользователи UNIX и Linux могут быть ошеломлены размерами и очевидной сложностью системы, которая предстала перед ними. Существует много хороших книг по использованию UNIX для всех уровней подготовки: от новичка до эксперта. Но ни одна из этих книг не обсуждает особенности Linux. Хотя 95% всего связанного с использованием Linux абсолютно аналогично другим UNIX-системам, наиболее прямой путь освоения этой системы, это по учебнику, написанному применительно к Linux. Вот эта книга и есть такой учебник.Здесь не предполагается каких-то предварительных знаний, за исключением первоначального знакомства с персональным компьютером и MS-DOS. Но даже если вы не успели побывать

пользователем MS-DOS, вы все равно все здесь поймете. На первый взгляд UNIX очень похож на MS-DOS (в конце концов фрагменты MS-DOS были спроектированы с оглядкой на операционную систему CP/M, которая, в свою очередь, проектировалась с оглядкой на UNIX). Но только при очень уж поверхностном взгляде можно говорить о похожести UNIX и MS-DOS. Если вы абсолютный новичок в мире персональных компьютеров, этот учебник вам поможет.И прежде, чем начать, призываем:

не бойтесь экспериментировать! Система вас не укусит. Работая на ней вы ничего не сможете сломать. UNIX имеет встроенные средства защиты, чтобы не дать "нормальным" пользователям (это теперь и вы) возможность испортить важные для системы файлы. Самое плохое, что вы можете натворить, это уничтожить все свои файлы, а тогда, может быть придется и переинсталлировать заново систему прим. переводчика: как правило, хотя и не всегда, чтобы довести систему до переинсталляции, надо иметь прав больше, чем у нормального пользователя.

3.2 Основы Linux.

Linux это многозадачная, многопользовательская операционная система. Это означает, что много людей может одновременно использовать один компьютер, выполняя много различных задач. (Это существенное отличие от MS-DOS, где только один человек может использовать в данный момент операционную систему). В UNIX пользователи должны себя идентифицировать при входе, что состоит из двух шагов: **ввода имени** (имя, по которому вас идентифицирует система) и **входной пароль**, который является вашим секретным словом для открытия вашего счета (регистрации в системе). Поскольку только вы знаете пароль, никто не может войти в систему под вашим именем.В традиционных UNIX-системах системный администратор присвоит вам имя и начальный пароль при вашей регистрации в системе (при заведении в системе нового пользователя). Но поскольку на своем персональном компьютере вы и системный администратор, вы должны себя (как пользователя) зарегистрировать в системе, прежде чем в нее войдете. Для дальнейших разговоров возьмем условное имя ``*larry*". Кроме прочего, каждая система UNIX имеет приписанное ей **hostname** (имя хоста). Это имя добавляет машине характера и очарования. Hostname используется для идентификации отдельных машин в сети, но даже если ваша машина не в сети, она все равно должна иметь hostname. Например, имя машины, обсуждаемой ниже ``*mousehouse*" (мышиная норка).

3.2.1 Регистрация в системе.

Прежде, чем вы сможете использовать систему, вы должны зарегистрировать себя в системе. Это необходимо потому, что неразумно использовать имя суперпользователя (*root*) для обычных нужд. Пользователь root нужен для выполнения привилегированных команд и сопровождения системы. Для того, чтобы зарегистрировать себя, вам необходимо зайти в систему под именем *root* и использовать команду *useradd* или *adduser* (зависит от дистрибутива). Об этой процедуре смотрите подробней в Разделе 4.6.

3.2.2 Вход в систему.

При входе вы увидите на экране подсказку, например, такого вида: mousehouse login: Введите регистрационное свое имя и нажмите клавишу Enter. Наш герой *larry* напечатает следующее: mousehouse login: larryPassword: Теперь введите ваш пароль (password). При вводе пароль не будет отображаться на экране, так что набирайте внимательнее. Если вы неправильно набрали пароль, то увидите на экране сообщение: Login incorrect
и вам следует попытаться еще раз.
Очень забавна ситуация, когда пароль надежно забыт. Тогда Вам придется зайти как root и стереть пароль для соответствующего пользователя (как это делается будет рассказано ниже). Еще более забавна ситуация, когда забыт root-пароль. Она забавна для всех, кроме самого root. Хотя систему даже из такой ситуации можно привести в рабочее состояние, сделать это нелегко. Когда вы наконец правильно введете имя пользователя и пароль, вы официально будете допущены в систему и можете в ней свободно путешествовать.

3.2.3 Виртуальные консоли.

Системная **консоль**, это монитор и клавиатура, связанные непосредственно с системой. Поскольку UNIX многопользовательская система, вы можете иметь дополнительные терминалы, связанные через последовательные порты с вашей системой, но они не будут консолями. Linux, как и некоторые другие версии UNIX, обеспечивает доступ к **виртуальным консолям** (или VC), которые позволяют войти в систему под несколькими именами в одно время.Для демонстрации этого войдите в систему (как было показано ранее). Теперь нажмите alt-F2. Вы должны снова увидеть подсказку *login:* , то есть перед вами вторая виртуальная консоль, а вы вошли через первую. Чтобы переключиться обратно на первую VC, нажмите alt-F1. Вы снова на первой консоли.Свежеинсталлированный Linux возможно позволит вам работать с четырьмя первыми VC, используя от alt-F1 до alt-F4. Но можно обеспечить работу с 12-ю VC: по одной на каждую функциональную клавишу. Как видите, использование VC может быть очень эффективным: вы можете работать на нескольких VC одновременно.В то время, как использование виртуальных консолей ограничено (кроме прочего, в каждый момент времени вы можете видеть только одну виртуальную консоль) оно дает вам представление о многопользовательских возможностях UNIX. Пока вы работаете на VC #1, вы можете переключиться на VC #2 и начать работу над чем-то другим.

3.2.4 Оболочки и команды.

В большинстве ваших исследований мира UNIX вы будете общаться с ним через оболочку **shell**. Shell это просто программа, которая воспринимает введенное пользователем, (т.е. команды, которые вы напечатаете) и транслирует это в команды системе. Это можно сравнить с программой *COMMAND.COM* под MS-DOS, которая делает нечто похожее. Shell лишь один из интерфейсов UNIX. Существует много различных интерфейсов, таких как X Window System, которая позволяет выполнять команды используя мышь и клавиатуру.Как только вы вошли, система запускает shell и вы можете вводить для него команды. Вот короткий пример. Как раз Larry вошел в систему и система выдала **подсказку**: mousehouse login: larryPassword: larry's passwordWelcome to Mousehouse! /home/larry #

``*/home/larry#*" это подсказка shell, показывающая, что он готов принимать команды. (Подробнее про подсказку позже). Давайте попросим систему сделать что-нибудь

интересное: /home/larry# make love make: *** No way to make target 'love' Stop /home/larry# Хм, как оказалось, make это имя существующей в системе программы и shell пытался выполнить эту команду. (Жаль, но система отнеслась к просьбе недружественно). Это подводит нас к жгучему вопросу: Что такое команды? Что происходит, когда вы вводите ``*make love*"? Первое слово командной строки ``*make*" это имя команды, которую предполагается выполнить. Все остальное в командной строке воспринимается как аргументы команды.Пример: /home/larry# cp foo bar
Здесь имя команды ``*cp*", а аргументы ``*foo*" и ``*bar*".
Когда вы вводите команду, shell делает несколько вещей. Во-первых, смотрит на то, что может (должно) быть именем команды и является ли это внутренней для shell командой (внутренняя, это команда, которую shell знает как выполнять. Существует ряд таких команд, мы о них поговорим позже). Shell также проверяет, не является ли команда синонимом другой или требуется подстановка имени. Если этого не надо делать, shell ищет соответствующую этому имени программу на диске. Если shell находит такую программу, он ее выполняет, передавая ей аргументы из командной строки.В нашем примере shell ищет программу по имени *make* и пытается выполнить ее с аргументом *love*. *make* это программа, которая часто используется при компиляции больших программ, она берет в качестве аргумента имя "целевого" файла компиляции. В случае ``*make love*" мы приказали команде make откомпилировать love. Поскольку make не смог найти файла с таким именем, он сообщил (несколько забавным образом) о невозможности выполнить команду и вернулся в подсказку. Что случится, если мы введем команду, а shell не сможет найти программу с этой командой? Давайте попробуем: /home/larry# eat dirteat: command not found. /home/larry#
Все очень просто, если shell не может найти программу с именем данным в командной строке (здесь ``*eat*"), он выдает сообщение об ошибке, которое объясняет причину невыполнения команды. Вы часто будете видеть это сообщение, если будете вводить имена команд с ошибками. (например, напечатаете ``*make love*" вместо ``*make love*").

3.2.5 Выход из системы.

Прежде, чем идти дальше, мы расскажем, как выйти из системы. При наличии подсказки shell используйте команду: /home/larry# exit
для выхода. Есть другие способы выхода, но этот самый безопасный.

3.2.6 Смена пароля.

Вы также должны представлять, как можно менять пароль. Команда "passwd" прим. переводчика: именно с пропущенными буквами она и пишется спросит вас про старый пароль и про новый Она попросит дважды ввести новый пароль для надежности. Внимание! Не забывайте свой пароль, иначе вам придется просить системного администратора уничтожить его и установить новый (Если вы и есть системный администратор, см. главу 4).

3.2.7 Файлы и каталоги.

Во многих операционных системах (включая UNIX) существует концепция **файла**, по которой его можно рассматривать просто, как набор информации, которому дано имя. Примерами файлов будут: программа, которая может выполняться, письмо, полученное по электронной почте, написанная вами статья. Существенно то, что все, что хранится

на диске, хранится в отдельных файлах. Файлы идентифицируются по именам. Например, файл, содержащий вашу статью может быть сохранен под именем *my-paper*. Эти имена обычно каким-то образом отражают содержание. Не существует стандартного формата имен файлов, как в MS-DOS и других операционных системах; в общем случае имена файлов могут содержать любые символы (кроме /: смотрите ниже обсуждение формирования "путей") и ограничены 256 символами по длине. Имя с путем (полное имя) ограничено длиной в 4096 символов. Одновременно с концепцией файла рассмотрим и концепцию каталога. *Каталог* это совокупность файлов. Его можно рассматривать как "папку", содержащую множество различных файлов. Каталоги сами по себе также получают имена, по которым вы их различаете. Каталоги организованы в древовидную структуру, т.е. каталоги могут содержать другие каталоги. К файлу можно обращаться по *пути* (*pathname*), формируемой из имени файла, которому предшествует имя каталога, содержащего файл Например, скажем, Larry имеет каталог, названный *papers*, который содержит три файла: *history-final*, *english-lit*, и *masters-thesis*. (Каждый из этих трех файлов содержит информацию о проводимых Larry работах). Для того, чтобы обратиться к файлу *english-lit*, Larry может указать маршрут: `papers`/english-lit Как вы видите, имена каталогов и файлов разделяются единичным слэшем (/). Поэтому имена файлов сами по себе не могут содержать этот символ. Пользователи MS-DOS увидят в этом что-то знакомое, поскольку в MS-DOS для этого используется бэкслэш (\). `Как уже говорилось, каталоги могут быть вставлены друг в друга. Например, пусть` `Larry` в каталоге *papers* имеет другой каталог с названием *notes*. Этот каталог содержит файлы с именами *math-notes* и *cheat-sheet*. Путь файла *cheat-sheet* будет: `papers/notes/cheat-sheet` `Итак, путь это маршрут, который надо проделать, чтобы добраться до конкретного файла. Каталог выше данного (под)каталога называется` **`родительским каталогом.`** `Здесь каталог` *papers* является родительским для каталога *notes*.

3.2.8 Дерево каталогов.

`Большинство` `Linux` систем имеет стандартную структуру каталогов, что облегчает конкретную установку системы. Структура представляет из себя дерево каталогов, начинающееся с каталога ``/'', известного под названием "корневой каталог". Каталоги ниже / относятся к числу важнейших подкаталогов: среди них */bin*, */etc*, */dev*, и */usr*. Эти каталоги в свою очередь содержат другие каталоги, которые содержат системные конфигурационные файлы, программы и т.д. В частности, каждый пользователь имеет **домашний каталог**, который выделяется пользователю для хранения его файлов. В вышеприведенном примере все файлы Larry (такие как *cheat-sheet* и *history-final*) содержались в домашнем каталоге Larry. Обычно пользовательский домашний каталог находится под каталогом */home* и называется именем пользователя. Так домашний каталог Larry будет */home/larry*.`Вот простое дерево каталогов. Оно даст вам некоторое представление о том, как организуется дерево каталогов в вашей системе.`

```
/_____bin    |_dev    |_etc    |_home_____larry    |  |_sam
|_lib    |_proc    |_tmp    |_usr__X386    |_bin    |_emacs    |_etc
|_g++-include    |_include    |_lib    |_local_____bin    |   |_emacs    |
|_etc    |   |_lib    |_man    |_spool    |_src_____linux    |_tmp
```

3.2.9 Текущий рабочий каталог.

Команды, которые вы даете shell, выдаются из вашего **текущего каталога**. Вы можете думать о вашем рабочем каталоге, как о каталоге в котором вы находитесь. При начальном входе в систему вашим рабочим каталогом автоматически становится домашний каталог (в нашем случае */home/larry*). При обращении к файлу вы можете обращаться к нему с учетом вашего местоположения, вместо того, чтобы указывать полный путь.Вот например, у Larry есть каталог *papers*, а *papers* содержит файл *history-final*. Если Larry хочет посмотреть этот файл, он может использовать команду:

```
/home/larry# more /home/larry/papers/history-final
```

Команда more просто показывает файл на экране порциями. Поскольку текущий рабочий каталог Larry */home/larry*, он вместо этого может обратиться к файлу с учетом своего текущего местоположения. Команда будет:

```
/home/larry# more papers/historv-final
```

Так что вы можете начинать имя файла (такого как *papers/final*) с символа, отличного от ``/'', система предполагает, что вы обращаетесь к файлу с учетом вашего текущего рабочего каталога. Это называют **относительным именем** (а полный маршрут, **полное (абсолютное) имя**, т.е. путь от корня до данного имени). С другой стороны, если Вы начинаете имя с символа /, система считает, что Вы используете полное имя файла с путем к нему, начиная от корневого каталога (/) файловой системы.

3.2.10 Обращение к домашнему каталогу.

Оболочки (shell), т.е. программы, которые читают и выполняют команды пользователя, могут использоваться (одновременно в одной системе) разные. В большинстве систем Linux используются *tcsh* или *bash* при начальной регистрации в системе. В *tcsh* и *bash* вы можете обратиться к своему домашнему каталогу, используя тильду (~). Например, команда: `/home/larry# more ~papers/history-final` эквивалентна

```
/home/larry# more /home/larry/papers/history-final
```

Символ ~ просто заменяет имя вашего домашнего каталога.Вы также можете обратиться к домашнему каталогу другого пользователя с помощью тильды. Имя ``~*karl/letters*'' будет интерпретировано shell, как ``*/home/karl/letters*'' (если */home/karl* домашний каталог пользователя karl). Использование тильды упрощает обращение; не существует каталога с именем ~ так что это просто "синтаксический сахар", который имеется в распоряжении shell.

3.3 Первые шаги в Linux.

Перед тем, как начать, важно заметить, что все имена файлов и команд чувствительны к большим и малым буквам (чего нет в системах типа MS-DOS). Например, команда *make* очень отличается от *Make* или *MAKE*. То же относится и к именам каталогов.

3.3.1 Первая прогулка.

Теперь мы можем войти в систему и узнать, как обращаться к файлам и менять местоположение в файловой системе, чтобы упрощать свою жизнь в ней. Команда для перемещения по дереву каталогов *cd*, (``change directory''). Вы скоро обратите внимание, что многие часто используемые команды Linux состоят из двух-трех букв. Формат команды *cd*:

cd directory
где *directory* имя каталога, в который вы желаете перейти. Как мы уже говорили, когда вы входите в систему, вы автоматически оказываетесь в своем домашнем каталоге. Если Larry желает двинуться ниже по дереву (например, в подкаталог papers), он должен использовать команду:
/home/larry# cd papers/home/larry/papers# Как видите, изменилась подсказка, отразив изменение местоположения (новый рабочий каталог). Теперь он в каталоге *papers* и может посмотреть на свой файл *history-final* с помощью команды: /home/larry/papers# more history-final Чтобы вернуться назад из подкаталога *papers*, надо использовать команду: /home/larry/papers# cd ../home/larry# Обратите внимание на пробел между ``*cd*" и ``..". Каждый каталог содержит имя ``..", которое отсылает к родительскому (для данного каталога) каталогу. Также каждый каталог имеет имя ``.", которое ссылается на него самого. Поэтому команда: /home/larry/papers# cd .
никуда не переведет.
В команде *cd* вы можете использовать маршруты. Чтобы перейти в домашний каталог Карла, вы можете воспользоваться командой: /home/larry/papers# cd /home/karl/home/harl# Используя команду *cd* без аргументов вы из любого места дерева вернетесь в свой домашний каталог: /home/karl# cd/home/larry#

3.3.2 Просмотр содержимого каталогов.

Теперь вы знаете, как ходить по каталогам, но вероятно возникает вопрос: "Ну и что дальше?" Само по себе хождение по каталогам бесполезно, давайте познакомимся с новой командой *ls*. ls (LiSt) выдает на экран перечень файлов и каталогов (по умолчанию из текущего каталога). Например, /home/larry# IsMailletterspapers /home/larry# Здесь мы видим, что у Larry три "единицы хранения" в его текущем каталоге: *Mail*, *letters* и *papers*. Но это мало, что говорит: каталоги это или файлы? Можно использовать опцию (прим. переводчика: часто в документации по UNIX используют в этом контексте слово "флаг") *-F* в команде *ls*, чтобы получить больше информации: /home/larry# Is --FMail/letters/papers/ /home/larry# Приписанные справа к именам файлов / говорят о том, что это подкаталоги. Использование *ls -F* (обратите внимание "*-F*" пишется без пробела) может дать также ``*" в конце некоторых имен файлов. Это будет говорить о том, что это **выполняемые** файлы или программы. Если, при вызове *ls -F*, ничего справа не приписано к имени, то это "нормальный" файл, т.е. не каталог и не выполняемый файл.В общем, каждая команда UNIX может иметь несколько опций в дополнение к другим аргументам. Эти опции обычно записываются со знаком ``-", как это было показано на примере *ls -F*. Опция *-F* сообщает команде *ls*, что необходимо выдать дополнительную информацию о типе файлов.Если вы напишете в команде *ls* имя каталога, то она выдаст содержимое указанного каталога: /home/larry# Is -F papersenglish-lithistory-finalmasters-thesisnotes/ /home/larry# Или, чтобы было интереснее, давайте посмотрим, что имеется в системном каталоге */etc/*:

```
/home/larry# Is /etcImages       ftpusers     lpc          rc.new
shellsadm          getty        magic        rc0.d      startconsbcheckrc
gettydefs    motd         rc1.d      swapoffbrc          group        mount
rc2.d      swaponbrc          inet         mtab         rc3.d
syslog.confcsh.cshrc    init         mtools       rc4.d
syslog.pidcsh.login    init.d       pac          rc5.d
syslogd.reloaddefault      initrunlvl   passwd       rmt
```

termcapdisktab inittab printcap rpc umountfdprm inittab.old profile rpcinfo updatefstab issue psdatabase securetty utmpftpaccess lilo rc services wtmp /home/larry# Для вышедших из MS-DOS пользователей полезно обратить внимание, что имена файлов могут быть длиннее 8 символов и содержать точку на любой позиции. Можно даже использовать несколько точек в одном имени.Давайте поднимемся вверх по дереву (прим. переводчика: так уж сложилось, что в UNIX начальной вершиной дерева является "корень (root)"), используя команду ``*cd* ..", а затем спустимся в другой каталог (*/usr/bin*): /home/larry# cd ../home# cd ../#cd usr/usr# cd bin/usr/bin# Вы, разумеется, можете передвигаться по каталогам большими шагами, например, сразу выполнить *cd /usr/bin*.Постарайтесь погулять по каталогам, используя команды ls и cd. В некоторых случаях вы можете напороться на раздражающее сообщение ``*Permission denied*"(обращение запрещено). Это всего лишь сработала система защиты UNIX, чтобы выполнять команды в тех или иных каталогах вы должны иметь на это разрешение. Подробнее об этом поговорим позже.

3.3.3 Создание новых каталогов.

Пора познакомиться с тем, как создавать каталоги. Это делается командой *mkdir*. Попробуйте следующее: /home/larry# mkdir foo/home/larry# Is -FMail/foo/letters/papers//home/larry# cd foo/home/larry/foo# Is/home/larry/foo# Наши вам поздравления! Вы только что создали новый каталог и зашли в него. Поскольку пока нет файлов в этом новом каталоге, давайте познакомимся с тем, как копировать файлы.

3.3.4 Копирование файлов.

Копирование файлов осуществляется командой *cp* (CoPy): /home/larry/foo# cp /etc/termcap/home/larry/foo# cp /etc/shells/home/larry/foo# Is -Fshells termcap /home/larry/foo# cp shells bells/home/larry/foo# Is -Fbells shells termcap /home/larry/foo# Команда *cp* копирует файлы, перечисленные в командной строке, в файл или каталог, указанный последним аргументом. (прим. переводчика: несколько файлов одной командой cp можно скопировать только в каталог; в файл можно скопировать только один файл). Обратите внимание на то, как мы используем каталог ``." для ссылки на текущий каталог.

3.3.5 Перемещение файлов.

Новая команда с именем *mv* (MoVe) перемещает файлы вместо их копирования. Синтаксис команды очевиден: /home/larry/foo# mv termcap sells/home/larry/foo# Is -Fbells sells shells /home/larry/foo# Обратите внимание, что теперь *termcap* уже не существует, а на его месте файл *sells*. Это можно использовать для переименования файлов, что мы сейчас и сделали. Но можно и переносить файлы в совсем другие каталоги.**Внимание!** Команды *mv* и *cp* уничтожат содержимое файла в который они пишут (если он существовал), не спрашивая вашего разрешения. Будьте внимательны, когда вы переносите файл в другой каталог: там уже может существовать файл с таким именем и вы его затрете.

3.3.6 Удаление файлов и каталогов.

Мы с вами насоздавали ненужных файлов, изучая работу команды *ls*. Для удаления файлов используется команда *rm* (ReMove): `/home/larry/foo# rm bells sells/home/larry/foo# Is -Fshells /home/larry/foo#` У нас ничего не осталось, кроме *shells*, но не будем переживать. Обратите внимание, что команда *rm* не будет вас переспрашивать перед удалением, так что будьте осторожны. Приказы пользователя не обсуждаются, они выполняются! Родственная *rm* команда *rmdir*. Эта команда удаляет каталоги, но только пустые каталоги. Если в каталоге есть хоть какие-нибудь файлы или подкаталоги, она распишется в бессилии.

3.3.7 Просмотр файлов.

Команды *more* и *cat* используются для просмотра содержимого файлов. *more* выдает файл на дисплей "поэкранно", в то время, как cat выдает весь файл разом. (прим. переводчика: если файл длинный, то, при использовании команды cat файл промелькнет и на экране останутся последние строки). Чтобы посмотреть файл *shells*, используем команду: `/home/larry/foo# more shells` При использовании команды *more* нажимайте клавишу пробел для перехода к следующей странице и b для возврата к предыдущей. Нажав q, вы выйдете из more.А теперь попробуйте команду *cat* etc/termcap/. Текст промелькнет слишком быстро, чтобы успеть его прочитать. На самом деле команда ``*cat*'' (conCATenate) в основном используется для других целей, для той же конкатенации нескольких файлов. Это в дальнейшем будет обсуждаться.

3.3.8 Получение оперативной помощи.

Практически каждый UNIX имеет то, что называется "Руководство" *man (``manual pages'')*. Эта команда *man* содержит документацию на различные команды системы, ресурсы, конфигурационные файлы. Например, если вы хотите найти информацию о других опциях команды *ls*, введите: `/home/larry# man Is`
и вам на экран будут выведены страницы Руководства по ls.
К сожалению, большинство страниц руководства написаны с ориентацией на пользователей, имеющих некоторые представления о работе соответствующих команд. Поэтому страницы Руководства обычно содержат справочные данные по командам, а не учебный материал.Но Руководство неоценимо для освежения памяти, если вы забыли синтаксис команды. Руководство может также много рассказать вам о командах, которые мы даже не упомянем в этой книге. Я предлагаю вам посмотреть в Руководстве те команды, которые мы уже обсуждали и все, с которыми мы будем встречаться. Вы обнаружите, что не на все команды есть Руководство. Тому несколько причин. Одна, некоторые страницы Руководства еще просто не написаны (

the Linux Documentation Project, программа подготовки документации для Linux, как бы отвечает за решение этой проблемы. Мы уже собрали большую часть документации). Во-вторых, команда может быть внутренней командой shell или синонимом (alias), что обсуждалось, в каждом из этих случаев для них нет собственных страниц. Возьмем для примера *cd*, которая является внутренней командой shell. Shell выполняет эту команду, но она не имеет своей отдельной программы.

3.4 Доступ к файлам MS-DOS.

Если, по какой-нибудь необъяснимой причине, вам необходимо обеспечить доступ к файлам MS-DOS, вы можете это легко сделать.

Обычно для получения доступа к файлам MS-DOS, вам достаточно примонтировать MS-DOS раздел или дискету и обращаться к фалам через файловую систему Linux. Например, если вы вставите дискету MS-DOS в устройство */dev/fd0* (A: в нотации MS/DOS), команда: `# mount -t msdos /dev/fd0 /mnt`
примонтирует эту дискету к каталогу */mnt*. Посмотрите раздел 4.8.4 для получения дополнительной информации о монтировании флоппи-дисков.
Точно также, вы можете примонтировать MS-DOS раздел на вашем винчестере. Если вы, например, имеете MS-DOS раздел на */dev/hda1*, команда: `# mount -t msdos /dev/hda1 /mnt`
смонтирует его. Не забудьте размонтировать DOS-раздел после окончания работы с ним. Вы можете монтировать раздел MS-DOS автоматически во время загрузки системы, если добавите строку в файл */etc/fstab* (см. раздел 4.4). Например, следующая строка в файле */etc/fstab* монтирует DOS раздел */dev/hda1* в каталог */dos*:
`/dev/hda1 dos msdos defaults` Вы также можете смонтировать файловую систему VFAT, используемую Windows 95/98/2000: `# mount -t vfat /dev/hda1 /mnt`
Таким образом вы получите возможность работы с длинными именами файлов. Данный подход применим только к разделам, которые реально хранят длинные имена. Нельзя работать с длинными именами, если раздел смонтирован как нормальный FAT16. Замечание: VFAT и FAT32 поддерживаются одним модулем ядра системы, так что если вы можете смонтировать раздел VFAT, то можете смонтировать и раздел FAT32. Подобная ситуация иногда приводит к забавным ситуациям, типа случая с Red Hat Linux, в которой не было упоминания про FAT32, хотя ее поддержка имелась (речь шла только о VFAT).

Вы можете также получить доступ к файлам MS-DOS, используя пакет Mtools. Команды *mcd*, *mdir* и *mcopy* этого пакеты работают точно также как команды MS-DOS *cd*, *dir*, *copy*. Если вы установили пакет Mtools, то он должен содержать и руководства на эти команды. Доступ к файлам MS-DOS и выполнение программ MS-DOS две разных вещи. В настоящее время в процессе разработки находится эмулятор программ MS-DOS. Он широко распространен и даже входит в состав дистрибутива SLS. Доступен он также и по FTP с многих серверов (см. приложение B). Эмулятор MS-DOS достаточно полон для выполнения большинства DOS программ, включая Wordperfect. Однако Linux и MS-DOS совершенно разные операционные системы и полнота любого MS-DOS эмулятора в любой UNIX-системе всегда ограничена. В разработке также находится несколько эмуляторов Microsoft Windows, которые будут работать под X Window System (ее иногда называют X Windows, но это неправильно).

3.5 Краткая информация о базовых командах.

Этот раздел представляет некоторые наиболее полезные базовые команды , включая те, о которых мы уже говорили. Обратите внимание, что опции обычно начинаются с ``-" и во многих случаях несколько однобуквенных опций могут следовать за одним минусом, записанные слитно. Например, вместо использования *ls -l -F*, можно использовать *ls -lF*.Вместо перечисления всех возможных опций каждой команды, мы будем говорить только о тех, которые полезны или важны в данное

время. Действительно, большинство из этих команд имеет большое число опций (большинство из которых никогда не используется). Вы можете для каждой команды с помощью man посмотреть все возможные опции.Обратите также внимание на то, что многие из команд берут список файлов или каталогов, как аргументы, обозначенные как ``*<file1>* ... *<fileN>*''. Например, команда *cp* берет в качестве аргументов список файлов, которые надо копировать, за которыми следует имя целевого файла или каталога. При копировании нескольких файлов в качестве целевого может выступать только каталог!

3.6 Исследование файловой системы.

Файловая система есть собрание файлов и иерархия каталогов. Я обещал поводить вас по файловой системе, и время настало. У вас достаточно интеллекта и знаний извлечь пользу из того, что я говорю и у вас есть карта дорог. /_____bin |_dev |_etc |_home_____larry | |_sam |_lib |_proc |_tmp |_usr__X386 |_bin |_emacs |_etc |_g++-include |_include |_lib |_local_____bin | |_emacs | |_etc | |_lib |_man |_spool |_src_____linux |_tmpПерво-наперво вернемся в корневой каталог (*cd /*) и сделаем *ls -F*. Вы, очевидно, увидите каталоги: *bin*, *dev*, *etc*, *home*, *install*, *lib*, *mnt*, *proc*, *root*, *tmp*, *user*, *usr* и *var*. Можете увидеть и несколько отличный вариант: не волнуйтесь, различные версии Linux могут иметь отличия.Присмотримся к каждому каталогу:

/bin

bin сокращенно от ``binaries" (т.е двоичные или выполняемые файлы). Здесь находится много важных системных программ. Используйте команду ``*ls -F /bin*" чтобы посмотреть имеющийся здесь список файлов. Вы можете обнаружить здесь уже знакомые вам команды, вроде *cp*, *ls* и *mv*. Это и есть программы соответствующих команд. Когда, например, вы используете команду *cp*, вы выполняете программу */bin/cp*.Используя *ls -F*, вы увидите, что большинство (если не все) файлов в */bin* имеют справа от имени звездочку (``*"). Это говорит о том, что файлы выполняемые, как описано в разделе 3.3.2.

/dev

Следующая остановка на нашем пути */dev*. Вновь посмотрите на содержимое с помощью *ls -F*."Файлы" в */dev* известны как **драйверы устройств** они используются для доступа к устройствам и ресурсам системы, таким как диски, модемы, память и т.д. Например, как вы можете читать данные из файла, точно также вы можете читать входные сигналы от мыши, имея доступ к */dev/mouse*. Имена файлов, начинающиеся на *fd* это дисководы гибких дисков. *fd0* первый дисковод, *fd1* второй. Теперь самые шустрые из вас заметят, что здесь имеется больше дисководов, чем те два, которые мною упоминались: они представляют специфические типы дисководов. Например, *fd1H1440* представляет доступ к high-density, 3.5" дискетам на дисководе 1.Вот перечень некоторых из наиболее используемых файлов устройств:

- • */dev/console/* относится к системной консоли, т.е. к монитору, напрямую связанному с системой.
- • Различные */dev/ttyS* и */dev/cua* устройства используются для доступа к последовательным портам. Например, */dev/ttyS0* относится к ``COM1" под MS-DOS. Устройства */dev/cua* относятся к "звонящим" (``callout") устройствам, которые используются совместно с модемами.

- Устройства, имена которых начинаются с *hd*, имеют доступ к жестким дискам. */dev/hda* относится ко

всему первому жесткому диску, а *hda1* только к первому разделу */dev/hda*.

- Устройства с именами, начинающимися на *sd* SCSI-драйверы. Если у вас SCSI жесткий диск, вместо доступа к нему через */dev/hda*, вы будете обращаться к */dev/sda*. SCSI ленты доступны через устройства *st*, а SCSI CD-ROM через *sr*.
- Устройства *lp* обеспечивают доступ к параллельным портам. */dev/lp0* относится к ``LPT1" в MS-DOS.
- */dev/null* используется как "черная дыра": любые данные, посланные сюда, канут в Лету. Если вы хотите подавить вывод команды на экран, вы можете перенаправить этот вывод в */dev/null*. Мы об этом позже еще поговорим.
- Устройства с именами */dev/tty* относятся к "виртуальным консолям" вашей системы (доступ путем нажатия alt-F1, alt-F2 и т.д.). */dev/tty1* соответствует первой VC, */dev/tty2* соответствует второй и т.д.
- Устройства, чьи имена начинаются на */dev/pty*, это "псевдотерминалы". Они используются для входа с удаленных "терминалов". Например, если ваша машина в сети, вход к вам по telnet будет использовать одно из устройств */dev/pty*.

/etc
/etc содержит множество всевозможных системных файлов конфигурации. Они включают */etc/passwd* (файл паролей), */etc/rc* (командный файл инициализации) и т.д.
/sbin
/sbin используется для хранения важных системных двоичных файлов, используемых системным администратором.
/home
home содержит домашние каталоги пользователей. Например, */home/larry* домашний каталог пользователя ``*larry*". На вновь инсталлированной системе этот каталог может быть пуст в связи с временным отсутствием зарегистрированных пользователей.
/lib
/lib содержит образы **разделяемых библиотек** (**shared library images**). Эти файлы содержат код, который могут использовать многие программы. Вместо того, чтобы каждая программа имела свою собственную копию этих выполняемых файлов, они хранятся в одном общедоступном месте в */lib*. Это позволяет сделать выполняемые файлы меньше и сэкономит место в системе.
/proc
/proc это "виртуальная файловая система", в которой файлы хранятся в памяти, а не на диске. Они связаны с различными **процессами**, происходящими в системе, и позволяют получить информацию о том, что делают программы и процессы в указанное время.
/tmp
Многие программы нуждаются в создании рабочих файлов, которые нужны короткое время. Каноническое место для этих файлов в */tmp* (там обычно чаще проводится уборка мусора).
/usr
/usr это очень важный каталог. Он состоит из ряда подкаталогов, которые в свою очередь содержат наиболее важные и полезные программы и файлы конфигурации, используемые системой.Различные каталоги, описанные выше, необходимы для

нормального функционирования системы, но большинство вещей, содержащихся в */usr* необязательны для системы. Но это такие необязательные вещи, которые делают систему полезной и интересной. Без */usr* вы бы имели достаточно занудную систему, содержащую только программы, вроде *cp* и *ls*. */usr* содержит много больших программных пакетов и конфигурационных файлов, которые их сопровождают.

/usr/X386

/usr/X386 содержит The X Window System, если вы ее инсталлировали. X Window System это мощная графическая среда, которая содержит большое количество графических утилит и программ, отображающих "окна" на вашем экране. Если вы знакомы с Microsoft Windows или Macintosh environments, то X Windows будет выглядеть весьма похоже. Каталог */usr/X11K6* содержит все выполняемые и конфигурационные файлы X Window, а также файлы поддержки.

/usr/bin

/usr/bin настоящее хранилище для различных программ UNIX. Он содержит большинство выполняемых программ, которых нет ни в каких других местах, например, в том же */bin* их нет.

/usr/etc

Точно также, как и */etc*, содержит всевозможные системные программы и конфигурационные файлы. */usr/etc* содержит даже больше утилит и файлов. В общем, файлы, находящиеся в */usr/etc* несущественны для системы, в отличие от тех, которые находятся в */etc*, и очень существенны.

/usr/include

/usr/include содержит **include-файлы** для компилятора Си. Эти файлы (большинство имен которых заканчивается на .h (от слова ``header'') объявляют имена структур данных, подпрограмм и констант, используемых при написании программ на Си. Те файлы, которые находятся в */usr/include/sys* в общем случае используются при программировании на системном уровне UNIX. Если вы знакомы с языком программирования Си, здесь вы найдете такие заголовки (фрагменты программ, вставляемые обычно в начало программы), *stdio.h*, которые описывают такие функции, как *printf()*.

/usr/g++-include

/usr/g++-include содержит include-файлы для компилятора Си++ (очень похожие на */usr/include*).

/usr/lib

/usr/lib содержит библиотеки-"заглушки" и "статические" библиотеки, эквивалентные файлам из */lib*. При компиляции программа "связывается" с библиотеками, находящимися в */usr/lib*, которые в свою очередь направляют программы в */lib*, если им нужен актуальный код. Кроме того, многие другие программы хранят в */usr/lib* свои конфигурационные файлы.

/usr/local

/usr/local в большой степени похож на */usr* он содержит различные программы и файлы, несущественные для системы, но превращающие ее в удовольствие и восторг. В общем, эти программы, находящиеся в */usr/local* специализируются на специфике вашей системы, т.е. */usr/local* сильно отличается в различных UNIX. Здесь вы найдете такие большие программные пакеты, как TeX (система форматирования документов) и Emacs (большой и мощный редактор), если вы их установите.

/usr/man

Этот каталог содержит страницы Руководства. Здесь два подкаталога для каждого "раздела" Руководства. (С помощью команды "man man" вы можете получить более подробную информацию). Например, */usr/man/man1* содержит

исходные тексты (неотформатированный оригинал) страниц Руководства в разделе 1 и */usr/man/cat1* содержит отформатированные страницы для раздела 1.

/usr/src

/usr/src содержит исходные коды (неоткомпилированные программы) для различных программ вашей системы. Наиболее важная вещь здесь, это */usr/src/linux*, содержащий исходные коды ядра Linux.

/var

/var содержит каталоги, которые часто меняются в размере или имеют тенденцию быстро расти. Многие из этих каталогов "квартировались" в */usr*, но поскольку мы стремимся сделать его достаточно стабильным, каталоги, которые часто меняются, были перенесены в */var*. К числу таких каталогов относятся:

/var/adm

/var/adm содержит различные файлы, интересные системному администратору, специфические системные файлы, фиксирующие ошибки и проблемы, возникающие в системе. Другие файлы фиксируют входы в систему, как и неудачные попытки войти.

/var/spool

/var/spool содержит файлы, которые предварительно формируются для других программ. Например, если ваша машина подключена к сети, входная почта будет помещаться в */var/spool/mail* до тех пор, пока вы не прочитаете ее или не удалите. Входящие и исходящие новости помещаются в */var/spool/news* и т.д.Различные каталоги, описанные выше необходимы для системы, но большинство элементов, найденных в /usr факультативно. Однако, эти факультативные элементы делают систему полезной и интересным. Без /usr, вы имели бы скучную систему, которая поддерживает только программы подобные cp и ls. /usr содержит большинство больших пакетов программ и файлов конфигурации, которые сопровождают их.

3.7 Типы оболочек.

Как я уже много раз говорил, UNIX многозадачная, многопользовательская операционная система. Многозадачность

очень полезна: однажды привыкнув к ней, вы будете всегда ее использовать. Прежде всего, вы сможете выполнять задачи в фоновом режиме, переключать задачи и объединять их в конвейер, достигая сложных результатов простыми средствами.

Многие из возможностей, которые мы будем обсуждать в этом разделе, обеспечиваются самой оболочкой (shell). Будьте внимательны, не путайте UNIX (фактическую операционную систему) с оболочкой: оболочка, это лишь интерфейс с находящейся за ней системой. Оболочка обеспечивает выполнение громадного числа функций помимо собственно UNIX. Оболочка не только интерпретатор интерактивных команд, которые вы можете ввести, получив от оболочки подсказку (готовности принимать команды). Это также мощный командный язык, который позволяет писать программы (**shell-scripts**), объединяющие несколько команд в **командный файл**. Пользователи MS-DOS почувствуют здесь нечто схожее с ``batch-файлами''. Использование программ на языке оболочки (shell) очень мощное средство, которое позволяет автоматизировать и и существенно повысить эффективность использования UNIX. Смотрите дополнительные сведения ниже. Существует несколько типов оболочек в мире Linux. Две главные: ``Bourne shell''(shell Борна) и ``C shell'. Shell Борна (или просто shell) использует командный синтаксис, похожий на первоначально для UNIX придуманный (вроде UNIX System III). В большинстве UNIX-

систем shell Борна имеет имя */bin/sh* (где *sh* сокращение от ``shell"). C shell использует иной синтаксис, чем-то напоминающий синтаксис языка программирования Си. В большинстве Linux-систем он имеет имя */bin/csh*. В Linux есть несколько вариаций этих оболочек. Две наиболее часто используемые, это Новый Shell Борна (Bourne Again Shell) или ``Bash" (*/bin/bash*) и Tcsh (*/bin/tcsh*). Bash это развитие прежнего shell с добавлением многих полезных возможностей, частично содержащихся в C shell. Поскольку Bash можно рассматривать как надмножество синтаксиса прежнего shell, любая программа, написанная на добром старом shell Баурна должна работать и в Bash. Для тех, кто предпочитает использовать синтаксис C shell, Linux поддерживает Tcsh, который является расширенной версией C shell.Тип оболочки, которую вы решили использовать - это почти как выбор религии. Некоторые предпочитают синтаксис shell Баурна с дополнительными возможностями, предоставляемыми Bash, а некоторые - более структурированный синтаксис C shell. Для "нормальных" команд, таких как *cp* и *ls*, тип используемого вами shell никакой роли не играет. Только когда вы начнете писать командные файлы или использовать некоторые новые свойства оболочек, различия между ними становятся существенными.При обсуждении далее некоторых свойств оболочек мы будем обращать внимание на различие между Борновским shell и C shell. (Если вам это действительно очень интересно, почитайте Руководство по поводу *bash* и *tcsh*).

3.8 Символы подстановки.

Ключевое свойство большинства оболочек Unix это способность ссылаться сразу более, чем на один файл, используя для этого специальные символы. Эти, так называемые "символы подстановки" (**wildcards**), позволяют ссылаться, скажем, на все файлы, содержащие символ "*n*". Символ ``*" относится к любому символу или строке символов в имени файла. Например, когда вы используете символ ``*" в имени файла shell заменяет ее всеми возможными именами файлов из каталога, на который вы ссылаетесь. Вот простенький пример. Предположим, что Larry имеет файлы *frog*, *joe* и *stuff* в своем текущем каталоге: /home/larry# Isfrog Joe stuff /home/larry# Для обращения сразу ко всем файлам с буквой ``*o*" в имени, мы можем использовать команду: /home/larry# Is *o*frog joe /home/larry#
Как видите, ``*" была заменена всеми возможными именами файлов из имевшихся в текущем каталоге.
Использование просто ``*"даст совпадение со всеми именами, поскольку все символы совпадают с символом подстановки: /home/larry# Is *frog Joe stuff /home/larry# Вот еще несколько примеров: /home/larry# Is f*frog /home/larry# Is *ffstuff /home/larry# Is *f*frog stuff /home/larry# Is s*fstuff /home/larry# Процесс замены ``*" на имена файлов называется расширением и выполняется shell. Это важно: конкретные команды, вроде *ls*, никогда не видят ``*" в своем списке параметров. Shell, расширяя символ подстановки, включает в список параметров все имена, прошедшие сравнение с шаблоном. Так что команда: /home/larry# Is *o*
расширяется shell до фактической:
/home/larry# Is frog joe Одно важное замечание относительно ``*". Использование ее не даст совпадения с именами файлов, которые начинаются с точки (``."). Эти файлы воспринимаются как "спрятанные", хотя на самом деле их никуда не прятали. Они не показываются в списке, выдаваемом нормальной командой *ls* и не выбираются при использовании ``*".Вот пример. Мы уже упоминали, что каждый каталог имеет два специальных файла: ``." (указание на текущий каталог) и ``.."

(указание на родительский каталог). Однако, если вы используете команду *ls*, эти два файла не будут отображены: `/home/larry# Is -a.bash-profile .bashrc frog joe stuff /home/larry# Если вы используете опцию -a` в команде *ls*, то вы сможете увидеть имена, начинающиеся на ``.": `/home/larry# Is ]?eJoe /home/larry# Is f??gfrog /home/larry# Is ????fstuff /home/larry# Как видим, два специальных файла ``." и ``..", также, как два других "спрятанных" файла - .*bash_profile* и .*bashrc*. Эти два файла используются при входе larry в систему. Более подробно о них поговорим позже. Обратите внимание, что когда мы используем ``*", ни один из файлов, с именами, начинающимися на ``." не отображается: `/home/larry# Is *frog Joe stuff /home/larry#`
Это мера предосторожности: если ``*" выбирала бы имена файлов, начинающиеся на ``.", она бы также выбрала имена ``." и ``..". Но это может быть опасно при выполнении ряда команд.

Другой символ подстановки ``**?**''. ``*?*" позволяет подставить строго один символ. Так ``*ls ?*" выдаст на только имена файлов, состоящие из одного символа, а ``*ls termca?*" выдаст ``*termcap*", но не выдаст на экран ``*termcap.backup*". Вот еще один пример: `/home/larry# Is J?eJoe /home/larry# Is f??gfrog /home/larry# Is ????fstuff /home/larry# Как видите, символ подстановки позволяет описывать много файлов за один раз. При обзоре простейших команд` мы говорили, что команды *cp* и *mv* могут копировать или перемещать множества файлов за один раз. Например, `/home/larry# cp /etc/s* /home/larry`
скопирует все файлы в */etc*, с именами начинающимися с ``s" в каталог */home/larry*. Формат команды *cp* на самом деле:

`cp` files destination

где *files* список копируемых файлов, а *destination* это файл или каталог, в который производится копирование. mv имеет аналогичный синтаксис.
Обратите внимание, что если производится копирование или перемещение более, чем одного файла, <*destination*> должен быть каталогом. В файл скопировать или переместить можно только один файл.

3.9 Каналы Linux.

3.9.1 Стандартный ввод и стандартный вывод.

Многие команды Linux получают информацию с так называемого **стандартного ввода** и посылают информацию на (опять же) так называемый **стандартный вывод**. (Для них часто используются сокращения ``**stdin**" и ``**stdout**" соответственно). Ваш shell организует дело так, что стандартным вводом служит клавиатура, а стандартным выводом экран.Вот пример использования команды *cat*. Нормально cat читает данные из файлов, чьи имена даны в командной строке и посылает эти данные прямехонько на stdout. Поэтому при выполнении команды: `/home/larry/papers# cat history-final masters-thesis`
на экран пойдет файл *history-final*, а за ним следом *masters-thesis*.
Но если команде `cat` не даны имена файлов в качестве параметров, она читает данные с stdin и опять же посылает на stdout. Вот пример: `/home/larry/papers# catHello thereHello thereByeBye Ctrl-D /home/larry/papers#`

Как видите, каждая строка, которую напечатал пользователь, немедленно выдается командой cat на экран. При вводе со стандартного входа команда знает, что ввод закончен тогда, когда она получит в каком-то виде сигнал EOT (End-Of-Text). Обычно он обеспечивается нажатием ctrl-D.Вот другой пример. Команда сортировки sort читает построчно текст (здесь опять с stdin, поскольку имена файлов в параметрах не указаны, и посылает отсортированный результат на stdout. Попробуйте так:

```
 /home/larry/papers# sortbananascarrotsapplesCtrl-
Dapplesbananas /home/larry/papers# Теперь мы можем упорядочить наш список
продуктов, подлежащих закупке, в алфавитном порядке... ну разве Linux не
```

полезная вещь?

3.9.2 Перенаправление ввода и вывода.

Теперь, предположим, что мы хотим послать результат сортировки в файл, чтобы где-то сохранить список планируемых покупок. Shell дает нам возможность **перенаправлять** стандартный выход в файл, используя символ ``>". Вот как это работает:

```
 /home/larry/papers# sort>shopping-listbananascarrotsapplesCtrl-
D /home/larry/papers#
```

Как вы можете видеть, результат работы команды *sort* не отображается на экране, вместо этого он сохраняется в файле *shopping-list* (список покупок). Давайте посмотрим на этот файл:

```
 /home/larry/papers# cat shopping-
listapplesbananascarrots /home/larry/papers#
```

Теперь мы можем не только сортировать (упорядочивать) список планируемых покупок, но и сохранять его! Но предположим, что мы хранили наш неотсортированный исходный закупочный список в файле под именем *items*. Один из способов сортировки и сохранения его, это отсортировать файл с данным именем, вместо получения файла со стандартного входа, и перенаправить стандартный выход в файл. Например так:

```
 /home/larry/papers# sort items>shopping-list/home/larry/papers# cat
shopping-listapplesbananascarrots/home/larry/papers#
```

Но это можно сделать и по-другому. Перенаправлен может быть не только стандартный вывод, но также и стандартный

ввод, используя символ ``<":

```
 /home/larry/papers#
sort<itemsapplesbananascarrots /home/larry/papers# Технически, sort<items
```

эквивалентно *sort items* , но последний вариант позволяет нам продемонстрировать сказанное: *sort* < *items* ведет себя так, словно данные файла *items* были напечатаны на клавиатуре. shell обслуживает перенаправление. *sort* не было дано имя файла (*items*) и команда читала со стандартного ввода, как будто шел ввод с клавиатуры. Это иллюстрирует концепцию **фильтра**. Фильтр, это программа, которая получает данные со стандартного ввода, обрабатывает их каким-то образом и посылает результат обработки на стандартный вывод. С помощью перенаправления стандартные ввод и вывод могут быть переведены на файлы. *sort* простейший фильтр: он сортирует входные данные и посылает результат на стандартный вывод. *cat* даже еще проще: она ничего не делает со входными данными, а только выдает все, что не поступит, на вывод.

3.9.3 Использование конвейера.

Мы уже показали, как использовать команду sort в качестве фильтра. Но эти примеры предполагали, что вы откуда-то получили данные в файл, или ввели данные с клавиатуры своими собственными руками. А что, если данные, которые вы хотите отсортировать, являются выходными данными другой программы, например, такой как *ls*? Если вы используете при сортировке опцию *-r*, данные будут расположены в порядке, обратном алфавитному. Если вы хотите получить перечень файлов вашего каталога в обратном порядке, один из способов сделать это будет:

```
 /home/larry/papers# 1senglish-listhist ory-finalmasters-thesisnotes/
```

Здесь мы сохранили результат работы команды *ls* в файле, а затем выполнили *sort -r* над этим файлом. Но это очень коряво выглядит и требует создания временного файла для хранения результата работы *ls*:

```
 /home/larry/papers# Is > file-list/home/larry/papers# sort -r file-
listnotes/masters-thesishistory-finalenglish-list/home/larry/papers#
```

Выход из положения дает трубопровод (**pipeline**) (прим. переводчика: в нашей литературе принят термин "**конвейер**", так далее и будем переводить "pipeline"). Конвейер это еще одно замечательное свойство shell, которое позволяет связывать последовательность команд в конвейер, где stdout первой команды посылается прямо на stdin второй команды и так далее. Здесь мы хотим послать stdout команды *ls* на stdin команды sort. Символ ``|" олицетворяет конвейер: `/home/larry/papers# Is sortnotes/masters-thesishistory-finalenglish-list/home/larry/papers#` Эта команда намного короче и, очевидно, проще набирается. Другой полезный пример. Команда: `/home/larry/papers# Is /usr/bin` выдает на дисплей длинный список имен файлов, большинство из которых слишком быстро промелькнет на экране, чтобы вы успели прочитать их. Давайте подключим к просмотру перечня имен файлов каталога /usr/bin команду more:

```
 /home/larry/papers# Is /usr/bin|more
```

Теперь вы можете постранично листать список файлов в свое удовольствие. Но чудеса на этом не кончаются! Мы можем связать в конвейер более, чем две команды. Команда head представляет из себя фильтр, который отображает первые строки входного потока (здесь, пришедшего по конвейеру). Если мы хотим отобразить последнее имя текущего каталога, упорядоченного по алфавиту, мы можем написать:

```
 /home/larry/papers# Is|sort -r head -1notes/ /home/larry/papers#
```

где *head -1* просто выдает первую строку получаемого входного потока (в данном случае это отсортированный в обратном порядке перечень имен файлов текущего каталога, выданных командой *ls*).

3.9.4 Перенаправление вывода с добавлением.

Использование ``>" для перенаправления выхода смертельно для файла, в который происходит перенаправление (если было, что уничтожать), другими словами:

```
 /home/larry/papers# Is>file-list
```

уничтожает прежнее содержимое файла *file-list*. Если вместо этого использовать символ перенаправления ``>>", выход будет добавлен к содержимому названного файла (вместо того, чтобы быть записанным на место старого):

```
 /home/larry/papers# Is>>file-list
```

добавит выходную информацию команды ls в файл *file-list*.

3.10 Права доступа.

3.10.1 Концепция прав доступа.

Поскольку Linux многопользовательская система, чтобы защитить файлы каждого пользователя от дурного влияния других пользователей, Linux поддерживает механизм, известный, как **система прав доступа к файлам**. Этот механизм позволяет каждому файлу приписать конкретного владельца. Как пример, поскольку Larry создал файлы в своем домашнем каталоге, именно Larry владелец этих файлов и имеет к ним доступ.Linux позволяет также совместно использовать файлы нескольким пользователям и группам пользователей. Если Larry так пожелает, он может закрыть доступ к своим файлам так, что никто другой не сможет к ним подступиться. Однако в большинстве систем по умолчанию другим пользователям разрешается читать ваши файлы, но запрещается изменять или удалять. Как объяснялось выше, каждый файл имеет конкретного владельца. Но, кроме того файлами, также владеют конкретные **группы** пользователей, которые определяются при регистрации пользователей в системе. Каждый пользователь становится членом как минимум одной группы пользователей. Системный администратор может даровать пользователю доступ более, чем к одной группе. Группы обычно определяются типами пользователей данной машины. Например, в университетском Linux пользователи могут быть разбиты на группы *student*, >*staff*,*faculty*, *guest*. Есть также несколько системно-зависимых групп (вроде *bin* и *admin*), которые используются самой системой для управления доступом к ресурсам. Очень редко обычный пользователь принадлежит к этим группам.Права доступа подразделяются на три типа:

чтение (read), запись (write) и выполнение (execute). Эти типы прав доступа могут быть предоставлены трем классам пользователей: владельцу файла, группе, в которую входит владелец, и всем прочим пользователям.

Разрешение на чтение позволяет пользователю читать содержимое файлов, а в случае каталогов - просматривать перечень имен файлов в каталоге (используя, например, *ls*). Разрешение на запись позволяет пользователю писать в файл и изменять его. Для каталогов это дает право создавать в каталоге новые файлы и каталоги, или удалять файлы в этом каталоге. Наконец, разрешение на выполнение позволяет пользователю выполнять файлы (как бинарные программы, так и командные файлы). Разрешение на выполнение применительно к каталогам означает возможность выполнять команды вроде cd.

3.10.2 Интерпретация прав доступа.

Давайте рассмотрим пример, демонстрирующий работу с правами доступа. Используя команду *ls* с опцией *-l* можно получить на экране перечень файлов данного каталога в "длинном" формате, включающем информацию о правах доступа:

```
 /home/larry/foo# Is -1 stuff-rw-r-r-   1 larry    users          505 Mar
13 J.9'05 stuff /home/larry/foo# Первое поле в выведенной строке
представляет права доступа. Третье поле владельца файла (larry) и четвертое
```

группу (*users*). Очевидно, что последнее поле есть имя файла (*stuff*), а остальные поля мы обсудим позже.Этим файлом владеет *larry*, и он принадлежит группе *users*. Давайте посмотрим на права доступа. В строке *-rw-r--r--* по порядку указаны права владельца, группы и всех прочих.Первый символ этой строки прав доступа (``-'') представляет тип файла. Символ ``-" означает, что это обычный файл (в противоположность каталогу или специальному файлу какого-то устройства).

Следующие три позиции (``*rw*-") представляют права доступа, которые имеет владелец файла *larry*. Символ ``*r*"означает ``read"(читать), ``*w*" ``write" (писать). Таким образом *larry* может читать файл *stuff* и писать в него.Как мы уже упоминали, кроме разрешений на чтение и запись существует разрешение на выполнение ``execute", представляемое символом ``*x*". Но в данном случае на этой позиции ``-", так что у Larry нет прав на выполнение этого файла. И это чудесно, файл *stuff* совсем даже не является программой. Разумеется, поскольку Larry владеет файлом, он может дать сам себе разрешение на выполнение этого файла, если захочет. Мы эту процедуру скоро обсудим.Следующие три символа *r*-- представляют права доступа группы для этого файла. Эта группа имеет имя *users*. Поскольку тут есть только ``*r*", любой пользователь этой группы может только читать файл.Последние три символа представляют ту же комбинацию *r*--, то есть для всех прочих доступно чтение этого файла и запрещены запись и выполнение.Вот еще несколько примеров на права доступа: -rwxr-xr-x Владелец файла может читать, записывать и выполнять файл. Члены его группы и все остальные пользователи могут читать и выполнять файл.-rw---- Только владелец файла имеет право на чтение и запись в него.-rwxrwxrwx Все пользователи имеют право на чтение, запись и выполнение.

3.10.3 Зависимости.

Важно заметить, что права доступа, которые имеет файл зависят также от прав доступа к каталогу, в котором этот файл находится. Например, даже если файл имеет -*rwxrwxrwx*, другие пользователи не смогут до него добраться, если у них не будет прав на чтение и выполнение каталога, в котором находится файл. Например, если Larry захочет ограничить доступ ко всем своим файлам, он может просто изменить права доступа своего домашнего каталога */home/larry* на *drwx------*. Таким образом, никто другой не будет иметь доступ в его каталог, а следовательно посторонним будут недоступны и все файлы. Так что Larry может не заботиться об индивидуальной защите своих файлов.Другими словами, чтобы иметь доступ к файлу, вы должны иметь доступ ко всем каталогам, лежащим на пути от корня к этому файлу, а также разрешение на доступ собственно к этому файлу. Обычно пользователи Linux весьма открыты всеми своими файлами. Обычно файлам устанавливается защита *-rw-r--r--*, которая позволяет другим пользователям читать файлы, но никоим образом их не менять. Каталогам обычно устанавливаются права доступа *drwxr-xr-x*, что позволяет другим пользователям ходить с правами экскурсантов по вашим каталогам. Но ничего в них не трогать и не записывать. Но многие пользователи хотят держать других пользователей подальше от своих файлов. Установив права доступа файла, *-rw-------* вы никому не покажете этот файл и не дадите записать в него. Также хорошо закрывает от всех файлы защита соответствующего каталога *drwx------* .

3.10.4 Изменение прав доступа.

Команда *chmod* используется для установки (изменения) прав доступа файла. Только владелец файла может менять права доступа к нему.Синтаксис команды имеет вид: chmod {a, u, g, o}{+/-}{r, w, x} filenames Кратко, вы выбираете из **a**ll (все), **u**ser (пользователь), **g**roup (группа) и **o**ther (другие). Далее указываете, либо вы добавляете права (+), либо лишаете прав (-). И наконец, вы указываете один или несколько режимов: **r**ead, **w**rite или **e**xecute. Несколько примеров допустимых команд: chmod a+r stuff Предоставить всем право на чтение.chmod +r stuff То же самое (a принимается по умолчанию).climod og-x stuff Запретить всем, кроме владельца,

выполнять файл.climod u+rwx stuff Предоставить владельцу полный контроль (чтение, запись и выполнение) над файлом.climod o-rwx stuff Запретить какой-либо доступ к файлу всем пользователям, кроме его владельца и членов его группы.

3.11 Управление связями.

Связи позволяют давать одному физическому файлу много имен. Системой файлы распознаются по **индексам файлов (inode)**, которые являются уникальными идентификаторами в рамках системы. Команда *ls -i* выдаст вам индексы файлов. На самом деле каталог это перечень индексов файлов с соответствующими этим индексам номерами. Каждое имя файла в каталоге привязано к конкретному индексу.

3.11.1 Жесткие связи.

Команда *ln* используется для создания множества связей для одного файла. Например, скажем, что у вас есть файл *foo*. Используя *ls -i* можно посмотреть индекс этого файла: `/home/larry# ls -i foo22192 foo /home/larry#`
Здесь файл *foo* имеет в файловой системе индекс 22192. Мы можем создать новую связь для этого файла под именем bar:
`/home/larry# In foo bar`
С помощью *ls -i* можно убедиться, что оба файла имеют один и тот же индекс:
`/home/larry# Is -i foo bar22192 bar 22192 foo /home/larry#`
Теперь, обращаясь к *foo* или *bar* мы фактически обратимся к одном у тому же файлу. Поэтому, если мы меняем что-то в файле *foo*, эти же самые изменения произойдут в файле *bar*.
Эти связи известны, как

жесткие связи (hard links), поскольку они реализуются прямой ссылкой на индекс файла. Обратите внимание, что в рамках одной файловой системы вы можете организовать только жесткие связи; символические связи (смотрите ниже) не имеют этого ограничения.

Когда вы удаляете файл командой *rm*, на самом деле вы удаляете только одну ссылку на файл. Если вы введете команду: `/home/larry# rm foo`
удаляется только связь, имеющая имя *foo*; *bar* будет как и прежде существовать. Файл только тогда действительно удаляется, когда на него больше нет связей. Обычно файлы имеют только одну связь, так что команда rm действительно приведет к удалению файла. Однако, если файл имеет много ссылок, применение rm приведет только к удалению одной связи; для того, чтобы удалить файл, вы должны удалить все связи на этот файл.
Команда *ls -l* покажет число ссылок на файл (кроме прочей информации):
`/home/larry# Is -l foo bar-rw-r--r-- 2 root root 12 Aug 5:16:51 bar-rw-r--r-- 2 root root 12 Aug 5:16:50 foo /home/larry#` Вторая колонка с цифрой ``2'' показывает число связей файла.На самом деле оказывается, что каталоги представляют из себя справочник типа "имена-индексы". Кроме прочего, каждый каталог имеет минимум две жесткие ссылки: ``." (ссылка, указывающая на самого себя) и ``.." (ссылка, указывающая на родительский каталог). В корневом каталоге (/) ссылка ``.." указывает на сам же каталог /.

3.11.2 Символические связи.

Символические связи, это другой тип связей, отличающийся от жестких связей. Символические связи позволяют давать новые имена файлам, но при этом не ссылаются на индекс файла.Команда *ln -s* создаст символическую ссылку на указанный файл. Например, если мы воспользуемся командой: `/home/larry# In -s foo bar` мы создадим символьную ссылку *bar*, указывающую на файл *foo*. Если теперь используем команду *ls -i*, то увидим, что два файла имеют различные индексы: `/home/larry# Ls -i foo bar22195 bar 22192 foo /home/larry#` Однако, используя *ls -l*, мы видим, что файл *bar* имеет символический указатель на *foo*: `/home/larry# Is -l foo barlrwxrwxrwx 1 root root 3 Aug 5 16:51 bar->foo-rw-r--r-- 1 root root 12 Aug 5:16:50 foo /home/larry#` При символической ссылке не используются биты прав доступа (они всегда отображаются, как *rwxrwxrwx*). Вместо этого, права доступа к файлу, полученному символической ссылкой, определяются правами доступа к файлу, на который он ссылается (в нашем примере определяется правами файла *foo*).Функционально, жесткие ссылки и символические ссылки похожи, но есть некоторые различия. Например, вы можете создать символическую ссылку на файл, который не существует; так нельзя сделать применительно к жесткой ссылке. Символические ссылки обрабатываются ядром иным образом, чем жесткие. Это скорее техническое отличие, но иногда важное. Символические ссылки полезны, поскольку они позволяют идентифицировать файл, на который они указывают; для жестких ссылок нет простого способа определить, какие файлы привязаны к одному и тому же индексу (inode).Ссылки используются во многих местах системы Linux. Символические ссылки особенно важны для образов разделяемых библиотек в */lib*. См. дополнительную информацию.

3.12 Управление работами.

3.12.1 Задачи и процессы.

Управление работами (**job control**) это возможность, которую предоставляют многие оболочки, включая (Bash и Tcsh). Управление работами позволяет управлять множеством команд или **работ** одновременно. Прежде, чем вы закопаетесь глубже, следует поговорить о **процессах**. Каждый раз, когда вы выполняете программу, вы начинаете то, что известно, как

процесс. Процесс это название для выполняемой программы. Команда *ps* выдает перечень имеющих место в данный момент процессов. Вот пример:

`/home/larry# psPID TT STAT TIME COMMAND24 3 S 0-03 (bash)161 3 R 0-00 ps /home/larry#` **PID** (**Process IDentificator**), перечисленные в первой колонке, это неповторяющиеся числа приписанные всем идущим процессам. Последний столбец (COMMAND) дает имя выполняемой команды. Здесь мы видим только процессы, которые инициировал Larry. (В системе выполняется и много других процессов. Команда ``*ps -aux*'' может выдать перечень всех происходящих в данный момент процессов). В выведенном перечне указаны *bash* (это оболочка, используемая Larry) и

сама команда *ps*. Как вы видите, bash выполняется параллельно с командой *ps*. bash выполнит *ps*, когда Larry введет команду. После окончания *ps* (после того, как выдана таблица процессов), управление возвращается к процессу *bash*, который выдает на экран подсказку готовности к приему новых команд. Выполняемый процесс известен shell как

работа. Термины процесс и работа взаимозаменяемы. Однако процесс обычно воспринимается, как "работа", когда речь идет об **управлении работами** (**job control**) свойстве shell, позволяющем уделять внимание нескольким независимым работам.

В большинстве случаев пользователи выполняют в каждый момент времени одну работу, ту которая соответствует последней переданной shell команде. Однако, используя управление работами, вы можете одновременно выполнять несколько работ, по необходимости переключаясь с одной на другую. Какая от этого польза? Давайте предположим, что вы редактируете текстовый файл и неожиданно хотите прерваться и сделать что-то другое. С помощью управления работами вы можете отложить редактирование и, вернувшись к подсказке shell, начать какую-то другую работу. После этого вы можете вернуться к редактированию, именно к тому месту, где вы прервали редактирование. Это всего один пример. Управление работами очень полезно на практике.

3.12.2 Выполнение работ на переднем плане и в фоне.

Работы могут выполняться как на **переднем плане**, так и в **фоне**. На переднем плане в каждый момент может быть только одна работа. Работа переднего плана, это работа, с которой вы взаимодействуете, она получает информацию с клавиатуры и посылает результаты на ваш экран. (Кроме, разумеется, случаев, когда вы сами перенаправляете ввод или вывод, как описывалось выше). С другой стороны, фоновые работы не получают информации с терминала, в общем случае они тихо выполняются, не испытывая потребности в общении с пользователем.Некоторые работы требуют очень большого времени для своего завершения и не свершают ничего внешне интересного в процессе этой работы. Компиляция программ - одна из таких работ, как и компрессия больших файлов. Нет вразумительных причин, почему вы должны при этом сидеть рядом и мучительно ждать, когда эти работы закончатся. Вы можете просто запустить их в фоне. Пока они там выполняются, вы можете заняться другими программами. Работы могут быть также **отложены**. Отложенная работа, это работа, которая в данный момент не выполняется и временно остановлена. После того, как вы остановили работу, в дальнейшем вы можете ее продолжить как на переднем плане, так и в фоне. Возобновление приостановленной работы не изменит ее состояния: при возобновлении она начнется с того места, на котором была приостановлена. Имейте в виду, что приостановка работы, это не

прерывание работы. Когда вы прерываете идущий процесс (нажимая клавиши прерывания, обычно это ctrl-C), то убиваете процесс насовсем. Клавиши прерывания можно переустанавливать командой stty. По умолчанию прерывание находится на ctrl-C, но мы не можем это гарантировать для всех систем. Если работа убита, то уж убита, и нет другого способа возобновить ее, как вновь запустить сначала, используя прежнюю команду. Заметим также, что некоторые программы могут перехватывать прерывания, тогда нажатие ctrl-C не приведет к немедленному прекращению работы. Это позволит программе выполнить необходимые операции аккуратного завершения. Некоторые программы вообще не позволят вам их прервать.

3.12.3 Работа в фоне и ликвидация работ.

Давайте начнем с простого примера. Команда *yes* вроде бы бесполезная команда, посылающая бесконечный поток "y" на стандартный вывод. Но это очень полезно. Если вы направите через конвейер эти "y" на ввод другой команды, которая требует ответов *yes* и no на вопросы, поток "y" даст подтверждение на все вопросы). Попробуйте:

```
/home/larry# yesyyyyy
```

Это закончится в

бесконечности. Вы можете убить процесс, нажав клавиши прерывания; обычно это ctrl-C. Чтобы нас больше не раздражал поток нескончаемых y, перенаправим его в */dev/null*. Как вы помните, */dev/null* выступает в качестве дыры памяти. В ней исчезают бесследно любые данные. Это хороший способ заставить замолчать слишком разговорчивые программы:

```
/home/larry# yes>/dev/null
```

Вот теперь намного лучше. Ничего не печатается, но и подсказка shell не появляется. Это потому, что программа продолжает работать, посылать y в */dev/null*. Снова нажмите клавиши прерывания, чтобы прекратить это.Давайте предположим, что мы хотим, чтобы команда *yes* продолжала работать, но также хотим получить обратно подсказку shell, чтобы выполнять другие работы. Мы можем перевести команду *yes* в фоновый режим, что позволит ей выполняться, но без выхода на взаимодействие с пользователем. Чтобы переместить процесс в фоновый режим, необходимо после команды символ ``*&*": `/home/larry# yes>/dev/null &[1] 164 /home/larry#` Вы видите, что мы вновь получили подсказку. Но что значит ``*[1] 164*"? И выполняется ли команда *yes* на самом деле?``*1*" представляет **номер работы** для программы *yes*. Shell приписывает номер каждой выполняемой работе. Поскольку yes единственная работа, которая в данный момент выполняется, ей присвоен номер 1. ``164" идентификатор процесса (PID); это номер, присвоенный системой работе. Любой из этих номеров можно использовать при обращении к работе, как это будет показано в дальнейшем. Теперь мы имеем выполняемый процесс *yes* в фоновом режиме, непрерывно посылающий поток "y"-ков в */dev/null*. Чтобы проверить состояние этого процесса, используйте внутреннюю команду shell *jobs*: `/home/larry# jobs[1]+ Running yes>/dev/null & /home/larry#` Ясно, что она выполняется. Вы можете также воспользоваться командой *ps*, показанной ранее, для проверки статуса работ. Для завершения работы используйте команду *kill*. Эта команда может брать в качестве аргумента как номер работы, так и идентификатор процесса. Это была работа номер 1, так что используя команду: `/home/larry# kill %1`
мы ликвидируем работу. При идентификации работы по номеру необходимо впереди ставить символ процента (``%").

Теперь, после ликвидации, мы можем снова использовать *jobs* для проверки:

```
/home/larry# ]obs[1]+ Terminated.
yes>/dev/null /home/larry#
```

Работа действительно уничтожена, и если мы снова воспользуемся командой *jobs*, ничего не будет выведено на экран.

Вы можете также убить работу, используя номер идентификатора процесса (PID), который выводится наряду с работой, когда вы начинаете работу (в фоновом режиме). В нашем пример PID равен 164, так что команда: `/home/larry# kill 164`
эквивалентна команде:

`/home/larry# kill %1` Вам не надо использовать ``*%*", когда вы обращаетесь к работе по номеру идентификатора процесса.

3.12.4 Остановка и возобновление работ.

Есть другой способ перевести работу в фоновый режим. Вы можете начать работу нормально (в режиме переднего плана), **остановить** работу и продолжить в фоновом режиме.Сначала начнем работу yes "нормально": /home/larry# yes > /dev/null Поскольку yes опять выполняется на переднем плане, вы не получите обратно на экран подсказку shell.
Теперь, вместо того, чтобы прерывать работу с помощью ctrl-C, мы остановим работу.

Приостановка работы не убивает ее. Чтобы осуществить приостановку работы, надо нажать соответствующие клавиши, обычно это ctrl-Z:

```
 /home/larry# yes > /dev/nullctrl-Z[1]+ Stopped
yes>/dev/null /home/larry#
```

Пока работа остановлена, она просто не выполняется. На нее не тратится время процессора. Но вы всегда можете возобновить работу, и она продолжится как ни в чем не бывало.

Для возобновления работы в режиме переднего плана используйте команду *fg* (``foreground": передний план): /home/larry# fgyes> /dev/null

Shell снова выдаст на экран имя команды, чтобы вы могли проконтролировать, какую работу вы активизировали в режиме переднего плана. Вновь остановите работу с помощью ctrl-Z. В этот раз используйте команду *bg* (``background" (задний план), фоновый режим), чтобы перевести работу в фоновый режим. Эффект будет аналогичен тому, как если бы вы набрали после команды ``*&*": /home/larry# bg[1]+ yes >/dev/null & /home/larry#
И мы получили назад подсказку. Команда jobs сообщит, что команда yes действительно выполняется, и мы можем снова ее убить с помощью команды *kill* (подходящее название, не правда ли?), как мы это уже делали.
Как же теперь остановить работу? Использование ctrl-Z не поможет, поскольку работа находится в фоновом режиме. Ответ: переместить работу на передний план, а затем остановить. Вы можете использовать *fg* как для остановленных работ, так и для работ, находящихся в фоне.Существует большая разница между фоновой работой и остановленной. Остановленная работа не выполняется и не использует время процессора, да и никакой работы, честно говоря, в этот момент не делает (но занимает память, хотя по воле свопинга может оказаться на диске). Работа в фоновом режиме и выполняется, и занимает память. Она может даже выводить что-то на экран, хотя это может раздражать вас, когда вы работаете над чем-то другим. Например, если вы использовали команду: /home/larry# yes &
без перенаправления stdout в */dev/null*, поток y будет выводиться на экран без возможности прервать это (вы не можете использовать ctrl-C для прерывания работ фонового режима). Чтобы остановить эту бесконечную выдачу, вам следует использовать команду fg для перевода работы в режим переднего плана, а затем использовать ctrl-C, чтобы ее убить.
Еще одно замечание. Команды *fg* и *bg* обычно переводят на передний план или в фоновый режим работы, которые были остановлены последними (что определяется символом ``+" после номера работы, это когда вы используете команду jobs). Если вы выполняете много работ одновременно, вы можете перевести на передний план или, наоборот, в фоновый режим конкретную работу заданием идентификатора работы в качестве аргумента команд fg или bg, как в: /home/larry# fg %2
(перевод на передний план работы номер 2) или:

```
/home/larry# bg %3
```
(перевод в фон работы номер 3). Для этих команд нельзя использовать идентификаторы процессов. Кроме того, использование только номеров работ, как в:
```
/home/larry# %2
```
эквивалентно:
```
/home/larry# fg %2
```
Помните, что управление работами, это свойство shell. Команды *fg*, *bg* и *jobs* внутренние команды shell. Если по какой-то причине вы используете shell, который не поддерживает управление работами, там вы не найдете этих команд.В дополнение к этому, есть некоторые аспекты управления работами, которые различаются в Bash и Tcsh. Некоторые оболочки не имеют управления работами, хотя большинство оболочек Linux имеют такую возможность.

3.13 Использование редактора vi.

Текстовый редактор, это программа, используемая для редактирования файлов, которые содержат текст, например письма, C-программы или системные конфигурационные файлы. Хотя в Linux много всяких разных редакторов, единственный редактор, который вы с гарантией найдете в любом UNIX, это *vi* (``visual editor''). *vi* не самый простой в использовании редактор. Но поскольку он так распространен в мире UNIX/Linux и в любой момент может вам потребоваться, он заслуживает хоть какого-то описания здесь. Выбор редактора, это дело персонального вкуса и стиля. Многие пользователи предпочитают витиеватый и мощный

Emacs редактор с самым большим набором возможностей, по сравнению со всеми другими редакторами в мире UNIX. Например, Emacs имеет свой собственный встроенный диалект языка программирования LISP и множество расширений (одно из которых, ``Eliza'', в некотором роде программа искусственного интеллекта). Однако, поскольку Emacs со всеми поддерживающими его файлами сравнительно велик, его нет на многих системах. *vi*, наоборот, маленький и удаленький, но, увы, более сложный в использовании. Но когда вы с ним освоитесь, вы поймете, что он очень простой. Правда осваивать его сложно.

Этот раздел, вразумительное введение в *vi*. Мы не будем обсуждать все его свойства, а только те, которые вы должны знать, чтобы начать работать. Если вы пожелаете знать больше деталей, обратитесь к страницам Руководства. Если вы хотите изучить его подробнее, почитайте книжку

Learning the vi Editor издательства O'Reilly and Associates, или VI Tutorial производства Specialized Systems Consultants (SSC) Inc. См. подробности в приложении A.

3.13.1 Концепции.

При использовании *vi* в любое время вы можете находиться в одном из трех режимов работы. Эти режимы известны как

командный режим, режим вставки и режим последней строки.

Когда вы начинаете работать с *vi*, вы в

командном режиме . Этот режим позволяет использовать определенные команды для редактирования файлов или перехода в другие режимы. Например, напечатав ``*x*'' при нахождении в командном режиме, удаляете символ, находящийся перед курсором. Стрелки передвигают курсор по редактируемому файлу. Большинство команд, используемых в командном режиме, состоит из одного или двух символов.

Вставку или редактирование текста вы осуществляете в

режиме вставки. При использовании *vi* вы, возможно, большую часть времени находитесь именно в этом режиме. Вы переходите в режим вставки с помощью команды ``*i*'' (``insert'' вставка) из командного режима. В режиме вставки вы вставляете текст в документ на место, указываемое курсором. Для завершения режима вставки и возврата в командный режим следует нажать Esc.

Режим последней строки, это специальный режим, используемый для расширения возможностей командного режима. При вводе таких команд они появляются в последней строке экрана. Например, если вы напечатаете ``:'' в командном режиме, вы перейдете в режим последней строки и сможете использовать такие команды, как ``*wq*'' (записать (write) файл и выйти (quit) из vi), или ``*q!*'' (выйти из vi без сохранения изменений). Режим последней строки в общем случае используется для команд *vi*, которые длиннее одного символа. В режиме последней строки вы вводите однострочные команды и нажимаете enter для их выполнения.

3.13.2 Запуск vi.

Лучший способ освоить эту концепцию, это вызвать *vi* и отредактировать файл. В примере ``*screens*'', приводимом ниже, мы собираемся только показать несколько строк текста, будто бы экран состоит всего из шести строк (вместо двадцати четырех).
Синтаксис для vi: `vi` filename
здесь *filename* задает имя редактируемого файла.
Запустите vi командой: `/home/larry# vi test`
для редактирования файла *test*. Вы увидите нечто вроде:
`"test" [New file]` Столбец символов ``~'' говорит о том, что вы стоите на конце файла. Подчеркивание представляет собой курсор.

3.13.3 Вставка текста.

Вы находитесь в командном режиме; для того, чтобы вставлять текст в файл, нажмите i (что переведет вас в режим вставки) и начинайте печатать: `Now; is the time for all good. men to come to the aid.of the party.` При вставке текста вы можете напечатать столько строк, сколько пожелаете (нажимая `Enter` после каждой строки), и можете корректировать ошибки, используя клавишу backspace. Для завершения режима вставки и возврата в командный режим нажмите `Esc`. В командном режиме вы можете использовать клавиши со стрелками для перемещения по файлу. Здесь, поскольку мы имеем только одну строку текста, попытки использовать стрелки "вверх" и "вниз" приведут лишь к тому, что vi на вас загудит.Есть несколько способов вставки текста, отличных от использования команды *i*. Например, команда *a* вставляет в текст, начиная

после текущего положения курсора, вместо текущей позиции курсора. Используйте левую стрелку для перемещения курсора между словами ``good" и ``men".:

```
Now; is the time for all good_men to come to the aid.of the party.
```

Нажмите a, для начала режима вставки, напечатайте ``*wo*", а затем нажмите esc для возврата в командный режим:

```
Now; is the time for all good, women to come to the aid.of the party. Для
того, чтобы начать вставку текста в строку ниже текущей, используйте
команду ``o". Например, нажмите o и напечатайте строчку или две: Now; is the
time for all good, humans to come to the aid of the party.Afterwards, we'll
go out for pizza and beer. Но помните, что в любое время вы находитесь либо
в командном режиме (где команды, такие как i, a или o могут применяться) или в
```

режиме вставки (где вы вставляете текст, а затем с помощью esc возвращаетесь в командный режим) или в режим последней строки (в котором вы расширяете расширяемые команды, как это обсуждается ниже).

3.13.4 Удаление текста.

В командном режиме команда "*x*" удаляет символ перед курсором. Если вы нажмете x пять раз, вы окажетесь в ситуации: `Now; is the time for all good, humans to come to the aid of the party.Afterwards, we'll go out for pizza and._` Теперь нажмите a, вставьте некоторый текст, а затем нажмите esc:

```
Now; is the time for all good, humans to come to the aid of the
party.Afterwards, we'll go out for pizza and Diet Cohe_ Вы можете удалять
целые строки, набирая команду dd (т.е. нажимая d дважды). Если ваш курсор на
второй строке, и вы напечатали dd: Now; is the time for all good, humans to
come to the aid of the party. Чтобы удалить слово, на котором находится
курсор, используйте команду dw. Поместите курсор на слово ``good" и напечатайте
```

dw: `Now; is the time for all humans to come to the aid of the party.`

3.13.5 Изменение текста.

Вы можете заменить фрагменты текста, используя команду *R*. Поместите курсор на первую букву слова ``party", нажмите R и напечатайте слово ``hungry": `Now; is the time for all humans to come to the aid of the hungry._` Использование *R* для редактирования текста очень походит на на команды *i* и *a*, но *R* заменяет прежний текст вместо вставки в него. Команда *r* заменяет один символ, отмеченный курсором. Например, переместите курсор на начало слова ``Now" и напечатайте *r*, а следом *C*, то вы получите: `Cow is the time for all humans to come to the aid of the hungry.` Команда ``~" изменяет размер буквы, отмеченной курсором: большую делает маленькой и наоборот. Например, если вы поместите курсор на ``o" в ``Cow" и затем последовательно будете нажимать ~, вы в конечном итоге получите: `COW IS THE TIME FOR ALL HUMANS TO COME TO THE AID OF THE HUNGRY.`

3.13.6 Команды перемещения курсора.

Вы уже знаете, как использовать стрелки для перемещений по документу. Вы также можете использовать команды *h*, *j*, *k*, и *l* для перемещения курсора влево, вниз, вверх и вправо соответственно. Это удобно, если (по каким-то причинам) ваши клавиши со

стрелками не работают как надо.Команда *w* перемещает курсор на начало следующего слова; *b* перемещает на начало предыдущего слова.Команда *0* (ноль) передвигает курсор на начало текущей строки, а команда *$* перемещает на конец строки.При редактировании больших файлов вы хотите перемещаться вперед и назад сразу на размер экрана. Нажатием ctrl-F курсор перемещается на экран вперед, с помощью ctrl-B на экран назад.Для того, чтобы переместить курсор в конец файла, напечатайте *G*. Можно переместиться также на любую строку, напечатав команду *10G* вы переместите курсор на десятую строку файла. Для того, чтобы встать на начало (на первую строку), используйте *1G*.Вы можете сочетать команды перемещения с другими командами, такими как удаление. Например, команда *d$* удалит все от местоположения курсора до конца строки; *dG* удалит все от курсора до конца файла и т.д.

3.13.7 Сохранение файлов и выход из vi.

Для выхода из *vi* без внесения изменений в ранее существовавший файл используйте команду *:q!*.
Когда вы напечатаете ``:'', курсор переместится на последнюю строку экрана, поскольку вы перейдете в режим последней строки:
COW IS THE TIME FOR ALL HUMANS TO COME TO THE AID OF THE HUNGRY. В режиме последней строки могут выполняться некоторые расширенные команды. Одна из них *q!*, которая позволяет выйти из *vi* без записи. Команда *:wq* сохраняет (записывает) файл, а затем выходит из *vi*. Команда *ZZ* (в режиме команд, без `:'') эквивалентна *:wq*. Помните, что вы должны нажать Enter после набора команды в режиме последней строки. Если хотите записать файл без выхода из *vi*, используйте просто *:w*.

3.13.8 Редактирование другого файла.

Для того, чтобы отредактировать другой файл, используйте команду *:e*. Например, чтобы прекратить редактирование файла *test* и перейти к редактированию файла *foo*, используйте команду: COW IS THE TIME FOR ALL HUMANS TO COME TO THE AID OF THE HUNGRY. foci Если вы используете :e без предварительного сохранения файла, то сначала вы получите сообщение об ошибке: No write since last change ("edit" overrides).
которое просто означает, что *vi* не желает редактировать другой файл, пока не будет сохранен первый. В этот момент вы можете использовать *:w*, чтобы сохранить исходный файл, а затем использовать *:e* или использовать команду:
COW IS THE TIME FOR ALL HUMANS TO COME TO THE AID OF THE HUNGRY. foci
``*!*'' говорит *vi*, что вы на самом деле имеете в виду редактировать новый файл без сохранения изменений, которые делались в первом.

3.13.9 Вставка других файлов.

Если вы используете команду *:r*, вы можете включить содержимое другого файла в текущий файл. Например, команда: :r foo.txt
вставит содержимое файла *foo.txt* в данное место текста (туда, где стоит курсор).

3.13.10 Выполнение команд Shell.

Вы можете также выполнять команды прямо из *vi*. Команда *:r!* работает как *:r*, но вместо чтения файла она вставляет вывод данной команды в буфер, в место, где находится курсор. Например, если вы используете команду: `:r! Is -F`
вы получите в результате:
`COW IS THE TIME FOR ALL WOMEN TO COME TO THE AID OF THE HUNGRYletters/misc/papers/` Вы можете выполнить команду, находясь в редакторе *vi* и вернуться в редактор после ее завершения Например, если вы используете команду: `! Is -F`
будет выполнена команда *ls -F*, а результат выдан на экран, а не вставлен в редактируемый файл. Если вы используете команду:
`shell`
vi запустит shell, который позволит временно "отложить" vi и выполнить команды. После выхода из shell (используя команду *exit*) вы вернетесь в *vi*.

3.13.11 Получение помощи по vi.

vi не слишком силен в интерактивной помощи (да и большинство UNIX-ов тоже), но вы всегда можете посмотреть страницы Руководства для *vi*. *vi* это "визуальная составляющая" редактора *ex*; это *ex* делает многое для поддержания режима последней строки и командного режима в *vi*. Так что в дополнение к чтению Руководства по *vi* посмотрите также Руководство по *ex*.

3.14 Настройка окружения.

Shell обеспечивает различные механизмы настройки вашей рабочей среды. Мы уже упоминали ранее, что shell больше, чем команда интерпретации: это также мощный язык программирования. Но обсуждение программирования на shell увело бы нас далеко в сторону, а мы бы хотели познакомить вас с некоторыми способами упрощения вашей работы в UNIX за счет использования некоторых дополнительных полезных свойств shell.Как мы упоминали ранее, различные оболочки используют различный синтаксис для написания своих программ. Например, `Tcsh` использует синтаксис, похожий на язык Си, в то время как shell Борна имеет другой синтаксис. В этом разделе мы не будем заниматься их различиями, а рассмотрим примеры, используя синтаксис shell Борна.

3.14.1 Скрипты shell.

Предположим, что вы часто используете серию команд и хотели бы сократить объем постоянной печати за счет группировки их в одну команду. Например, команды:
`/home/larry# cat chapter1 chapter2 chapter3 >book/home/larry# wc -1 book/home/larry# Ip book`
объединяют файлы, содержащие главы книги: *chapter1*, *chapter2*, *chapter3* и помещают результат в файл *book*. Затем подсчитывается число строк в книге (в файле *book*) и отображается на дисплее и, наконец, печатается командой *lp*.
Вместо введения каждый раз этих команд, вы можете собрать их в один **скрипт** или сценарий (командный файл). Сценарии `shell` мы кратко опишем позже. А сценарий, который выполнит вышеприведенные команды, будет выглядеть следующим образом: `#!/bin/sh# A shell script to create and print the boohcat chapter1 chapter2 chapter3 > bookwc -1 bookIp` book Если этот сценарий будет помещен в

файл *makebook*, то вы можете просто использовать далее команду: /home/larry# makebook
которая выполнит все команды сценария. Сценарии shell обычные текстовые файлы, которые вы можете создавать с помощью редактора вроде *emacs* или *vi*.
Давайте посмотрим на этот сценарий. Первая строка ``#!/bin/sh/" говорит о том, что этот файл есть сценарий и сообщает shell, как выполнить сценарий. В данном случае необходимо передать сценарий для выполнения команде */bin/sh*, где */bin/sh* сама программа shell. Почему это важно? В большинстве систем UNIX */bin/sh* shell Борновского типа, например *bash*. Инициируя работу сценария shell выполняется, используя */bin/sh*, при этом мы гарантируем, что сценарий будет выполняться именно под shell Борновского типа (а не, скажем, под C shell). Этот сценарий будет выполняться под shell Борна, даже если вы используете Tcsh (или какой-то другой C shell) как свою рабочую оболочку. Вторая строка представляет из себя **комментарий**. Комментарии начинаются символом ``#" и могут продолжаться до конца строки: они игнорируются shell и могут использоваться программистом для пояснений.Остальные строки сценария обычные команды в том виде, в каком бы вы их вводили прямо на выполнение. Shell читает каждую строку сценария и выполняет эту строку, как будто вы ввели эту строку в ответ на подсказку shell. Права доступа важны для сценариев. Если вы создали сценарий, вы должны убедиться, что вы имеете права на его выполнение. Если вы создавали сценарий в редакторе, то он (обычно) не получает автоматически прав на выполнение. Подробно права доступа рассмотрены здесь.
Можно использовать команду: /home/larry# chmod u+x makebook
чтобы дать самому себе разрешение на выполнение shell-сценария *makebook*.

3.14.2 Переменные shell и окружение.

Shell позволяет определять **переменные**, как и большинство языков программирования. Переменная это порция данных, которой дано имя. В языке shell переменные не определяются (в традиционном смысле), так как все они одного типа "строкового", речь может идти только об их инициировании: присваивании начальных значений). **ВНИМАНИЕ!** Имейте в виду, что Tcsh, также, как и C shell, используют различные механизмы определения переменных, отличающиеся от используемых здесь. Здесь обсуждается shell Борна, например, bash (см. man-страницу Tcsh для подробностей). Когда вы присвоите значение переменной (используя оператор ``=" ), вы сможете получить это значение, добавив перед именем переменной символ ``*$*", как это показано ниже: /home/larry# foo="hello there"
Переменной foo присвоено значение ``*hello there*". Теперь вы можете обратиться к этой переменной, добавив перед именем символ ``*$*". Команда:
/home/larry# echo $foohello there/home/larry#
дает тот же самый результат, что и:
/home/larry# echo "hello there"hello there /home/larry# Эти переменные являются внутренними для shell. Это означает, что только shell имеет доступ к этим переменным. Это может быть полезно для сценариев; если вам надо сохранить информацию о имени файла, вы, например, можете поместить его в переменную. Команда set может показать вам перечень всех определенных переменных shell. Shell позволяет **экспортировать** переменные в среду. **Среда** это множество переменных, к которым могут иметь доступ все выполняемые команды. Определив однажды переменную внутри shell командой *export* вы можете передать ее среде.
ВНИМАНИЕ! Здесь вновь есть отличие между Bash и Tcsh. При использовании Tcsh используется другой синтаксис для помещения переменных в среду (используется команда *setenv*). Дополнительную информацию можно найти в Руководстве по Tcsh.

Среда очень важна в системах UNIX. Она позволяет конфигурировать некоторые команды за счет установки переменных, о которых знают команды.Вот небольшой пример. Переменная среды *PAGER* используется командой man. Она указывает команду, которая используется в свою очередь командой man для просмотра Руководства на экране. Если вы установите в качестве значения *PAGER* имя другой команды, то эта команда вместо будет обеспечивать просмотр вместо more (которая применялась по умолчанию).Присвойте *PAGER* значение ``*cat*''. Выдача на экран руководства будет вся разом, а не поэкранно, как это делала команда more:

```
/home/larry# PAGER=cat
```

Теперь экспортируйте *PAGER* в среду:

```
/home/larry# export PAGER
```

Попробуйте команду *man ls*. Руководство промелькнет по вашему экрану без (желательных) задержек.Теперь, если присвоить *PAGER* значение ``*more*'', то для выдачи на экран будет использоваться команда more:

```
/home/larry# PAGER=more
```

Обратим внимание на то, что нам не надо заново использовать команду *export* после изменения значения *PAGER*. Необходимо только раз экспортировать переменную; любые изменения, которые будут происходить после этого, будут отражаться в среде.Страницы Руководства для конкретных команд содержат информацию о том, использует ли команда какие-то переменные среды. Например, Руководство по команде *man* говорит о том, что для определения режима выдачи страницы руководства на экран используется переменная *PAGER*. Некоторые команды совместно используют переменные среды, например, многие команды используют переменную среды *EDITOR* для указания используемого редактора.Переменные среды используются также для сохранения важной информации о процедуре входа. Например переменная *HOME* содержит имя вашего домашнего каталога:

```
/home/larry/papers# echo $HOME/home/larry
```

Часто необходимо оградить строки, чтобы оболочка не воспринимала как специальные символы в них. К таким символам относятся ``*'', ``?'', пробел и другие спецсимволы. Подробнее см. в

Bourne Shell Tutorial.

Другая интересная переменная среды *PS1*, которая определяет главную подсказки shell. Например:

```
/home/larry# PS1="Your command, please"Your
```

command, please Для переустановки подсказки обратно в нормальное состояние (когда она показывает текущий рабочий каталог, после которого следует значек ``#''), выполните следующее:

```
Your command, please PS1="\W#"/home/larry#
```

В Руководстве по *bash* есть подробное описание синтаксиса, используемого при установке подсказки.

Переменная окружения PATH.

Когда вы используете команду *ls*, как shell находит соответствующий выполняемый файл (программу) для *ls*? На самом деле в большинстве систем *ls* находится в */bin/ls*. Shell использует переменную среды *PATH* ("ПУТЬ") для указания возможного местоположения выполняемых файлов соответствующих команд.Например, ваша переменная *PATH* может иметь значение:

```
/bin; /usr/bin; /usr/local/bin
```

Это список каталогов (в которых `shell` будет искать команду), отделяемых друг от друга двоеточием ``:''. Когда вы используете команду *ls*, *shell* прежде всего проверяет наличие */bin/ls*, затем */usr/bin/ls* и т.д. Обратите внимание на то, что переменная *PATH* не помогает находить обычные файлы. Например, если вы используете команду:

```
/home/larry# cp foo bar
```

shell не использует *PATH* для нахождения местопребывания файлов *foo* и *bar*: предполагается, что эти имена однозначно определяют место. Shell использует *PATH* только для нахождения команды cp.
Это экономит вам массу времени; это означает, что вы не обязаны помнить, где находятся выполняемые файлы команд. Во многих системах выполняемые файлы разбросаны во многих местах, таких как */usr/bin*, */bin* или */usr/local/bin*. Вместо того, чтобы писать полное имя команды (вроде */usr/bin/cp*), вы просто указываете в *PATH* перечень каталогов, которые бы вы хотели, чтобы shell автоматически просматривал.Обратите внимание, что *PATH* содержит ``.", что означает "текущий рабочий каталог". Это позволяет вам создавать shell-сценарии или программы и выполнять их как команды из текущего каталога, без необходимости указывать это прямо (как в случае *./makebook*). Если каталог не указан в *PATH*, то shell не будет его просматривать в поиске команд, это касается и текущего каталога. В этом заключается принципиальное отличие от MS-DOS: в ней по умолчанию ПЕРВЫМ проверяется текущий каталог.

3.14.3 Стартовые скрипты shell.

В дополнение к shell-сценариям, которые создаете вы, существует множество сценариев, которые использует сам shell для своих целей. Наиболее важными среди них являются **сценарии запуска**, которые автоматически выполняются shell при вашем входе в систему.
Сценарии запуска сами по себе это обычные сценарии, как это описывалось выше. Но они очень полезны при установке вышей среды путем автоматического выполнения набора команд при вашем входе в систему. Например, если вы всегда используете команду *mail* для проверки своей почты в момент входа в систему, вы можете поместить эту команду в свой сценарий инициализации и она будет выполнена автоматически. Как Bash, так и Tcsh делают различие между начальным shell (вызываемым при входе в систему) и прочими вызовами shell. Начальный shell вызывается в момент входа пользователя в систему; часто это единственный экземпляр shell, который вы используете. Но если вы вызываете shell из другой программы, такой как*vi*, вы тем самым запускаете новый (экземпляр) shell. Кроме того, когда вы запускаете на выполнение shell-сценарии, вы автоматически инициируете новый экземпляр shell. Файлы инициализации, используемые в Bash: */etc/profile* (устанавливается системным администратором, выполняется всеми экземплярами начальных пользовательских *bash*, вызванными при входе пользователей в систему), *$HOME/.bash_profile* (выполняется при входе пользователя) и *$HOME/.bashrc* (выполняемый всеми прочими не начальными экземплярами *bash*). Если *.bash_profile* отсутствует, вместо него используется *.profile*. Tcsh использует следующие сценарии инициализации: */etc/csh.login* (выполняется всеми пользовательскими *tcsh* в момент входа в систему), *$HOME/.tcshrc* (выполняется во время входа в систему и всеми новыми экземплярами *tcsh*) и *$HOME/.login* (выполняется во время входа после *.tcshrc*). Если *.tcshrc* отсутствует, вместо него используется *.cshrc*.Для того, чтобы лучше понять функции этих файлов, вам следует больше узнать о shell. Программирование на shell сложный вопрос, далеко выходящий за рамки этой книги. Дополнительную информацию можно получить из Руководства на *bash* и *tcsh*.

3.15 Не хотите ли попробовать сами?

Надеемся, что мы дали достаточно информации относительно того, как использовать систему. Имейте в виду, что большая часть интересных и важных

аспектов Linux здесь не обсуждалась - здесь рассматривались только самые базовые. Но этот фундамент поможет вам быстро освоить и использовать более сложные приложения. Если все рассмотренное здесь вам не показалось волнующе интересным, не унывайте: в Linux есть еще много того, с чем следовало бы познакомиться. Один из незаменимых способов изучения системы это чтение Руководства. Хотя многие страницы Руководства могут выглядеть и достаточно сложными, если вы будете достаточно глубоко копать, вы откопаете несметные россыпи полезной информации.Мы также советуем прочитать какую-то полную книгу по системе UNIX. В UNIX значительно больше разнообразных возможностей, чем можно увидеть с первого взгляда, к сожалению, они выходят за рамки этой книги. Некоторые хорошие книги по UNIX указаны в приложении А.

---

4. Администрирование системы.

Эта глава представляет обзор функций администрирования системы Linux, включая ряд особых функций, предназначенных исключительно для администратора системы.

Как в каждой бочке меда присутствует ложка дегтя, так и каждая система имеет своего администратора. А администрирование системы это очень важная и иногда пожирающая уйму времени работа, даже если вы единственный пользователь системы.

Мы постараемся обсудить здесь наиболее важные вещи, связанные с администрированием, о котором вы должны знать при использовании Linux, чтобы не испытывали неудобств при работе с ОС. Чтобы быть не слишком болтливыми и приятными собеседниками, мы и раньше рассматривали только основные черты, пропуская многие важные детали. Вам следует прочитать книгу "Linux System Administrator's Guide" (где взять сказано в Приложении А), если у вас относительно Linux серьезные намерения. Это поможет вам лучше понять как там все происходит, и как там все взаимодействует. В крайнем случае, стоит все это просмотреть, чтобы знать что в книге содержится и какой помощи вам следует от нее ожидать.

4.1 Аккаунт root.

Обычные пользователи в общем случае ограничены так, что они не могут причинить вред кому-либо другому в системе (включая саму систему), кроме самих себя. Права доступа к файлам в системе организованы таким образом, что простой пользователь не может удалить или изменить файл, файл в каталогах, которые пользователи используют совместно (такие как /bin и /usr/bin). Большинство пользователей также защищают свои собственные файлы так, что не могут их изменить, а иногда и вообще добраться до них.

Но все эти ограничения не распространяются на пользователя root. Пользователь root может читать, модифицировать или удалять любой файл системы, изменять его права доступа или менять его владельца. Он (root) может также выполнять специальные (привилегированные) программы, такие как разбитие диска на разделы или создание

файловой системы. Основная идея состоит в том, что некто (их может быть несколько), кто выполняет регистрацию пользователей (и носит имя `root`), должен, когда это необходимо, иметь возможность выполнять работы, которые не могут быть выполнены обычным рядовым пользователем. Поскольку root может делать все, что угодно, ему легко совершить какую-то ошибку, приводящую к катастрофическим последствиям.

Например, если вы как обычный пользователь случайно попытаетесь удалить файл в `/etc`, система не разрешит вам это сделать. Но, если вы вошли как `root`, система даже не пикнет, выполняя все, что прикажете. Легко уничтожить систему, пребывая в системе в качестве root. Лучший способ избежать неприятностей, это:

- Посидеть на собственных ладошках, прежде чем нажать Enter для выполнения команды, которая может быть причиной катастрофы. Например, если вы собираетесь очистить каталог, перед нажатием Enter перечитайте всю команду и убедитесь, что она написана правильно.
- Не привыкайте использовать `root`. Чем более комфортно вам будет в роли `root`, тем больше вы будете путать ваши привилегии с привилегиями нормального пользователя. Например, вы можете подумать, что вы сейчас находитесь в системе как `larry`, хотя на самом деле будете неудержимым `root`.
- Используйте отличающуюся подсказку для `root`. Для этого следует внести изменения в root-овский `.bashrc` или `.login` файл для того, чтобы сделать подсказку для `root` отличной от других. Например, многие используют символ ``$" в подсказках обычных пользователей и оставляют символ ``#" для подсказки `root`.
- Входите под именем `root` только тогда, когда это абсолютно необходимо. И, как только вы закончите работу `root`, выйдите (выведите `root` из системы). Чем меньше используете `root`, тем меньше навредите системе.

Разумеется, есть племя хакеров, которые используют `root` практически всегда и везде. Но каждый из них когда-то по глупости уничтожил хотя бы (в лучшем случае) одну систему. Есть общее правило: пока вы не познакомились с неограниченными возможностями `root`, и не привыкли к отсутствию ограничений, входите под root в крайнем случае.

Мы детально обсудим системное администрирование чуть позже.

4.2 Загрузка системы.

Многие загружают Linux используя ``загрузочную дискету", которая содержит копию ядра Linux. В ядре есть информация о корневом разделе Linux, так что ядро знает, где искать на жестком диске корневую файловую систему. Это тип дискеты, созданной, например, Slackware в процессе инсталляции.

Для создания своей собственной загрузочной дискеты, сначала разместите образ ядра на своем жестком диске. Он должно быть в файле `/vmlinuz` или `/vmlinux`. Некоторые инсталляции используют `/vmlinuz` в качестве символической ссылки на настоящее ядро системы, благодаря чему его легко найти.

Выяснив местонахождение ядра системы, установите в нем имя раздела с корневой файловой системой Linux командой `rdev`. Формат команды:

```
rdev kernel_name root_device
```

где *kernel-name* имя образа ядра, *root-device* имя корневой файловой системы. Например, для установки имени корневой файловой системы в ядре в файле `/vmlinuz` в значение `/dev/hda2`, напишите:

```
# rdev /vmlinuz /dev/hda2
```

`rdev` может установить еще несколько параметров ядра, например, использовать по умолчанию при загрузке SVGA видеорежим. Команда:

```
# rdev -h
```

выведет на экран справку. После установки имени корневой файловой системы, скопируйте ядро на дискету. Перед копированием данных дискету надо отформатировать командой MS-DOS `FORMAT.COM` или Linux `fdformat`. Таким образом будут исключены плохие сектора дискеты.

Форматы дискет и их файлы устройств обсуждаются ниже.

Файлы устройств, как упомянуто ранее, проживают в каталоге `/dev`. Чтобы скопировать ядро из файла `/etc/Image` на дискету в `/dev/fd0`, скомандуйте:

```
# cp /vmilinuz /dev/fd0
```

Эта дискета должна теперь загрузить Linux.

4.2.1 Использование LILO или LILOвая система.

LILO это программа, которая располагается в загрузочном секторе вашего жесткого диска. Эта программа выполняется, когда система загружается с жесткого диска и может автоматически загрузить Linux из образа ядра, хранящегося на жестком диске.

LILO может быть также использована, как начальный загрузчик для нескольких операционных систем, позволяя вам выбирать во время загрузки, какую операционную систему (например, Linux или MS-DOS) загружать. Когда вы загружаетесь с использованием LILO, то загружается операционная система, установленная по умолчанию, если вы не нажмете shift во время выполнения загрузки. Если вы нажмете клавишу shift, или если в файле `lilo.conf` есть директива `prompt`, вам будет выдана подсказка загрузчика, в ответ на которую вы напечатаете имя операционной системы, которую надо загрузить (например, ``linux'' или ``msdos''). Если вы нажмете tab в ответ на подсказку загрузчика, вам будет выдан перечень доступных операционных систем.

Простой способ инсталлировать LILO: отредактировать файл конфигурации `/etc/lilo.conf`. После чего команда:

```
# /sbin/lilo
```

впишет новую конфигурацию `lilo.conf` в загрузочный сектор. Эту команду необходимо отдавать каждый раз, когда вы модифицировали файл `lilo.conf`.

Файл конфигурации LILO содержит записи для каждой операционной системы, которую вы можете загрузить. Лучший способ продемонстрировать это на примере конфигурационного файла LILO config. Нижеприведенные установки для системы, которая имеет корневой раздел Linux на `/dev/hda1` и раздел MS-DOS на `/dev/hda2`:

```
# Tell LILO to modify the boot record, on /dev/hda (the first
# non-SCSI hard drive). If you boot from a drive other than
# /dev/hda, change the following line
boot = /dev/hda

# Set a sane videomode
vga = normal

# Set the delay in milli-seconds. This is the time you have to
# press the 'SHIFT' hey to bring up the LILO prompt if you

# haven't specified the 'prompt' directive
delay = 50

# Name of the boot loader. No reason to modify this unless you're
# doing some serious hacking on LILO
install = /boot/boot.b

# This forces LILO to prompt you for the OS you want to boot
# A 'TAB' hey at the LILO prompt will display a list of the OSs
# available to boot according to the names given in the 'label='
# directives belov;
prompt

# Have LILO perform some optimisation
compact

# Stanza for Linux root partition on /dev1hda1
image = vmlinuz     # Location of kernel
label = linux       # Name of OS (for the LILO boot menu)
root = /dev/hda1    # Location of root partition
read-only           # Mount read only

# Stanza for MSDOS partition on /dev/hda2
other = /dev/hda2   # Location of partition
table = /dev/hda2   # Location of partition table for /dev/hda2
label = msdos       # Name of OS (for boot menu)
```

Записи первой операционной системы в файле `config` это та ОС, которую LILO загружает по умолчанию. Вы можете выбрать другую ОС во время загрузки в ответ на подсказку LILO, как это уже обсуждалось ранее. Помните, что каждый раз, когда вы изменяете образ ядра на диске, вы должны заново выполнить `/sbin/lilo`, чтобы изменения отразились в загрузочном секторе вашего диска. Имейте также в виду, что если вы используете здесь строку ``root=", нет смысла использовать `rdev` для установки корневого раздела в образе ядра. LILO установит его сам во время загрузки.

Установщик Microsoft Windows '95 стирает менеджер загрузки LILO. Если Вы собираетесь устанавливать Windows '95 на вашей системе после установки LILO,

создайте сначала диск начальной загрузки (см. раздел 4.2). С диском начальной загрузки, Вы сможете загрузить Linux и повторно установить LILO после того, как поставите Windows '95. Это делается просто: скомандуйте как root `/sbin/lilo`. Разделы с Windows '95 могут быть конфигурированы для загрузки через LILO с использованием тех же записей в `lilo.conf`, что и для загрузки с раздела MS-DOS.

Linux FAQ (см. Приложение A) дает дополнительную информацию по тому, как использовать LILO при загрузке через Boot Manager OS/2.

4.3 Выключение системы.

Выключить Linux не так просто. Не забывайте, что никогда нельзя просто выключить питание или нажать кнопку "reset" во время работы системы. Ядро отслеживает диск при вводе-выводе с помощью буферов. Если вы перезагружаете систему, не дав шанса ядру переписать буфера на диск, вы можете попортить файловые системы. Тем не менее, переводчик лично знает прекрасных и обаятельных ребят, которые работая системными администраторами одного университета, выключают систему именно так. Должен заметить, что тот факт, что их сервер до сих пор работает, говорит не об квалификации, а о высокой устойчивости Linux, а то, что они до сих пор работают системными администраторами говорит только о том, что они нравятся администрации ВУЗа, а не об их знаниях.

Необходимы и другие меры предосторожности при выключении. Всем процессам посылается сигнал, который позволяет им красиво умереть (записав, что надо и закрыв все файлы и т.д.). Файловые системы для безопасности размонтируются. Если вы желаете, система может также предупредить пользователей, что предстоит выключение, чтобы дать им шанс тоже (красиво) выйти из системы.

Простейший способ выключения, это использование команды `shutdown`. Формат команды:

```
shutdown time warning-message
```

Здесь *time* задает время выключения системы (в формате *hh:mm:ss или на русском чч:мм:сс*), а *warning-message* задает сообщение, выдаваемое на терминалы всех пользователей перед выключением. Вы можете просто указать время (*time*) как ``now", что приведет к безотлагательному выключению. Опция `-r` приведет к перезагрузке после выключения.

Например, выключить систему в 8:00 вечера можно командой:

```
# shutdown -r 20:00
```

Команда `halt` может инициировать немедленное выключение без посылки предупреждающих сообщений или предоставления паузы перед выключением. `halt`

полезна, если вы единственный пользователь системы и хотите выключить систему и вырубить питание.

ВНИМАНИЕ! На выключайте электропитание и не перезагружайте ее, пока не увидите на консоли сообщение:

```
The system is halted.
```

Важно сделать выключение "чисто", используя команды `shutdown` или `halt`. В некоторых системах нажатие ctrl-alt-del будет перехвачено системой и приведет к ее выключению, но в других системах использование "затычки для вулкана" приведет к немедленной перезагрузке системы и может быть причиной крупных неприятностей.

4.3.1 Файл `/etc/inittab`.

Немедленно после начальной загрузки Linux, когда ядро смонтирует корневую файловую систему, оно запустит самую первую программу системы: `init`. Эта программа ответственна за старт сценариев запуска системы, и приводит систему из состояния начальной загрузки в обычное многопользовательское состояние. `init` также порождает `login:` оболочки для всех tty устройств на системе, и выполняет другие действия по запуску и парковке системы.

После запуска `init` остается спокойно работать в фоновом режиме, в случае необходимости он меняет состояние системы. Имеется много деталей, о которых программа `init` должна позаботиться. Эти задачи определены в файле `/etc/inittab`. Пример `/etc/inittab` приведен ниже.

Неправильное изменение файла `/etc/inittab` может предотвратить Вашу регистрацию в системе. По крайней мере, при изменении файла `/etc/inittab`, сделайте резервную копию оригинала, и позаботьтесь о наличии аварийно-спасательной дискеты. Впрочем, о ней надо позаботится в любом случае.

```
# inittab    Этот файл описывает, как процесс INIT конфигурирует
#            систему на каждом уровне работы (run-level).
#
# Версия:    @(#)inittab             2.04    17/05/93        MvS
#                                    2.10    02/10/95        PV
#
# Автор: Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
# Переработано: Patrick J. Volkerding, <volkerdi@ftp.cdrom.com>
# Дополнительные изменения: Robert Kiesling, <kiesling@terracom.net>
#
# Уровень по умолчанию
id:3:initdefault:

# Инициализация системы (запускается при загрузке системы)
si:S:sysinit:/etc/rc.d/rc.S

# Скрипт, который запускается в пользовательском режиме (уровень 1)
su:1S:wait:/etc/rc.d/rc.K

# Скрипт, который запускается в многопользовательском режиме
```

```
rc:23456:wait:/etc/rc.d/rc.M

# Что делать при нажатии комбинации Ctrl-Alt-Del
ca::ctrlaltdel:/sbin/shutdown -t5 -rfn now

# При уровне 0 система останавливается
l0:0:wait:/etc/rc.d/rc.0

# При уровне 6 система перезагружается
l6:6:wait:/etc/rc.d/rc.6

# Что делать при перебое в питании (выгрузка системы в режиме с
# одним пользователем)
pf::powerfail:/sbin/shutdown -f +5 "THE POWER IS FAILING"

# Если питание успело восстановиться, прекратить выгрузку системы
pg:0123456:powerokwait:/sbin/shutdown -c "THE POWER IS BACK"

# Если питание восстановилось в режиме с одним пользователем,
# то вернуться к многопользовательскому режиму
ps:S:powerokwait:/sbin/init 5

# Команды getty в многопользовательском режиме на консолях,
# подключенных к последовательным портам
#
# ВНИМАНИЕ: приведите эти команды в соответствие с
#           вашими командами getty, иначе вы не сможете
#           войти в систему!!!
#
# ВНИМАНИЕ: аргументы команды `agetty': скорость, порт
#           аргументы команды `getty_ps': порт, скорость, `gettydefs'
c1:1235:respawn:/sbin/agetty 38400 tty1 linux
c2:1235:respawn:/sbin/agetty 38400 tty2 linux
c3:1235:respawn:/sbin/agetty 38400 tty3 linux
c4:1235:respawn:/sbin/agetty 38400 tty4 linux
c5:1235:respawn:/sbin/agetty 38400 tty5 linux
c6:12345:respawn:/sbin/agetty 38400 tty6 linux

# Последовательные порты (serial lines)
# s1:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt100
s2:12345:respawn:/sbin/agetty -L 9600 ttyS1 vt100

# Подключения через модем (dialup lines)
d1:12345:respawn:/sbin/agetty -mt60 38400,19200,9600,2400,1200 ttyS0 vt100
#d2:12345:respawn:/sbin/agetty -mt60 38400,19200,9600,2400,1200 ttyS1 vt100

# Уровень 4 раньше использовался только для систем X Window.
# Потом было обнаружено, что этот уровень зацикливает скрипт init,
# так что load avg все время остается равным по меньшей мере 1.
# Таким образом, теперь есть один процесс getty, открытый на
# tty6. Надеемся, что этого никто не заметит. ;^)
# Кроме того, не так уж плохо иметь одну текстовую консоль
# на случай, если что-либо случится с X.
x1:4:wait:/etc/rc.d/rc.4

# Конец файла /etc/inittab
```

При запуске системы `/etc/inittab` запускает 6 виртуальных консолей, и выдает запрос `login:` на модем на порте `/dev/ttyS0` и на символьный терминал, подключенный через RS-232 к последовательному интерфейсу `/dev/ttyS1`.

`init` проходит через несколько уровней выполнения (**run levels**), каждый из которых соответствует определенному состоянию системы. Уровень 1 ставится немедленно после начальной загрузки системы, уровни 2 и 3 нормальные многопользовательские режимы работы системы, уровень 4 запускает X Window System с менеджером дисплея X `xdm`, уровень 6 перезагружает систему. Уровни выполнения указываются для каждой команды во втором элементе каждой строки файла `/etc/inittab`.

Например, строка:

```
s2:12345:respawn:/sbin/agettн -L 9600 ttyS1 vtlOO
```

выведет запрос `login` на последовательный терминал для уровней выполнения 1-5. ``s2'' перед первым двоеточием символический идентификатор, используемый внутри `init`. `respawn` является ключевым словом `init`, которое часто используется с последовательными терминалами. Если после некоторого времени программа `agetty`, которая ведает входом через терминал, не получает ввода с терминала, она завершается. ``respawn'' предписывает `init` повторно выполнить `agetty`, для обеспечения постоянной возможности входа в систему через терминал. Остальные параметры передаются прямо `agetty` и определяют оболочку, запускаемую при входе в систему, скорость последовательной линии, последовательное устройство и тип терминала, заданный в `/etc/termcap` или `/etc/terminfo`.

Программа `/sbin/agetty` обрабатывает много деталей, связанных с терминальным вводом-выводом в системе. Есть несколько различных версий, которые обычно используются на Linux системах. Они включают `mgetty`, `psgetty`, или простую программу `getty`.

В случае такой строки в файле `/etc/inittab`:

```
d1:12345:respawn:/sbin/agetty -mt50 35400,19200,9500,2400,1200 ttySO vtl00
```

которая позволяет пользователям войти через модем, связанный с последовательным устройством `/dev/ttyS0`, параметры ``-mt60'' для `/sbin/agetty` позволяют системе использовать все скорости модема, которые использует вызывающий абонент, и прерывать связь, если соединение не установлено через 60 секунд. Поддерживаемые модемом скорости перечисляются в командной строке так же, как скорости для последовательной линии или тип терминала. Конечно, оба модема должны поддерживать скорость до которой договорились машины в момент установления соединения.

Много важных деталей не рассмотрено в данном разделе. Задачи `/etc/inittab` занимают объем с хорошую книгу (может, я и соберусь ее написать). Подробности можно найти на man-страницах по программам `init` и `agetty`, а также в Linux Documentation Project's Serial HOWTO, поискать которое можно по адресам, приведенным в Приложении А.

4.4 Управление файловыми системами.

Другая задача системного администратора забота о файловой системе. Большая часть этой работы состоит в проверке файловой системы на наличие поврежденных или испорченных файлов; многие системы делают такие проверки во время загрузки.

4.4.1 Монтирование файловых систем.

Сначала несколько концепций, связанных с файловыми системами. Прежде, чем файловая система будет принята вашей системой, она должна быть **смонтирована** в каком-то каталоге. Например, если у вас файловая система на дискете, то вы должны примонтировать ее в каталог, скажем `/mnt`, для того, чтобы обеспечить доступ к ее файлам (смотрите подробности здесь). После монтирования файловой системы все файлы этой системы появляются в этом каталоге (и ниже). После размонтирования файловой системы каталог (в нашем случае `/mnt`) будет пуст, то же самое справедливо для файловой системы на жестком диске. Каталог `/mnt` будет пуст, если он был пуст до монтирования, иначе наоборот, станут видными файлы каталога `/mnt` (основной системы), которые становятся невидимыми, когда к этому каталогу монтируется файловая система).

Система автоматически монтирует файловые системы на ваш жесткий диск во время загрузки. Так называемая "корневая файловая система" монтируется к каталогу `/`. Если у вас отдельные файловые системы, например, для `/usr`, она монтируется в `/usr`. Если у вас только корневая файловая система, то все файлы, включая содержимое `/usr`, существуют в этой файловой системе.

Команды `mount` и `umount` (НО НЕ *unmount*!!!) используются для монтирования и демонтирования файловых систем. Команда:

```
mount -av
```

выполняется автоматически из файла `/etc/rc` или `/etc/rc.d/boot` при загрузке системы. Файл `/etc/fstab` хранит сведения о файловых системах и точках монтирования. Пример файла `/etc/fstab`:

```
# device   directory  type  options
/dev/hda2  /          ext2  default
/dev/hda3  /usr       ext2  default
/dev/hda4  none       swap  sw
/proc      /proc      proc  none
```

Первое поле, `device`, задает имя раздела для монтирования. Второе поле задает точку монтирования. Третье поле задает тип файловой системы, например для ext2fs тип `ext2`, а для Minix file system тип будет `minix`. В таблице указаны наиболее распространенные типы файловых систем. Не все файловые системы из списка поддерживаются вашей конкретной системой, поскольку в ядро должна быть скомпилирована поддержка для каждой системы, а реально туда компилируется только поддержка одной-двух нужных постоянно систем. Подробнее о построении ядра можно почитать здесь.

Файловая система	Тип для `/etc/fstab`	Описание применения
Вторая расширенная файловая система	`ext2`	Second Extended File system (наиболее распространенная файловая система для ОС Linux).
Расширенная файловая система	`ext`	Extended File system (прототип `ext2`).
Minix	`minix`	Специально разработана для ОС Minix, сейчас используется редко. Можно встретить на спасательных дискетах.
Xiafs	`xia`	Аналогична `ext2`, но редко используется (меньше возможностей).
UMSDOS	`umsdos`	Для установки системы Linux в разделе MS-DOS.
FAT *	`msdos`	Для доступа к файлам MS-DOS/Windows и других систем с поддержкой FAT12/16/32/VFAT.
`/proc`	`proc`	Обеспечивает информацию о процессах для `ps` и т.п.
ISO 9660	`iso9660`	Формат большинства дисков CD-ROM.
Xenix	`xenix`	Для доступа к файлам системы Xenix.
System V	`sysv`	Для доступа к файлам системы System V (версии для x86?).
Coherent	`coherent`	Для доступа к файлам системы Coherent.
HPFS	`hpfs`	Доступ к разделам HPFS (OS/2).
NTFS	`ntfs`	Для доступа к файлам системы NTFS (Windows NT/2000).
HFS	`hfs`	Для доступа к файлам систем HFS и HFS+ (MacOS).
PhotoCD	`photocd`	Для доступа к файлам на дисках Kodak Photo CD.
Audio	`audio`	Для доступа к дискам CD Audio, которые можно слушать на компьютере.

Таблица 4.1: Типы файловых систем в Linux.

Linux с дополнительными драйверами расширения ядра поддерживает практически все существующие файловые системы.

Последнее поле в строке файла `fstab` задает параметры для `mount`. Обычно там установлено `defaults`.

Раздел свопа включен в файл `/etc/fstab`. Он имеет точку монтирования `none` и тип `swap`. Команда `swapon -a`, которая вызывается из `/etc/rc` или из `/etc/init.d/boot`, включает свопинг для всех разделов свопа, указанных в файле `/etc/fstab`.

Файл `/etc/fstab` содержит специальную запись для файловой системы `/proc`. Как сказано выше, файловая система `/proc` используется для хранения информации о системе, процессах, доступной памяти и прочей технической информации. Фактически данная система предоставляет интерфейс к сервисам ядра. Если `/proc` не смонтирована, команды подобные `ps` не работают.

Команда `mount` может использоваться только `root`. Это сделано для обеспечения безопасности системы. Вам не захочется, чтобы монтирование и размонтирование файловых систем зависело от прихоти рядовых пользователей. Есть несколько программных пакетов, которые дают возможность тем самым рядовым пользователям монтировать и размонтировать файловые системы (особенно на дискетах) не затрагивая безопасности системы.

Команда `mount -av` фактически монтирует все файловые системы, кроме корневой файловой системы (в ранее приведенной таблице `/dev/hda2`). Корневая файловая система автоматически монтируется ядром во время загрузки.

Вместо использования `mount -av` вы можете примонтировать файловую систему вручную. Команда:

```
# mount -t ext2 /dev/hda3 /usr
```

эквивалентна монтированию файловой системы на `/dev/hda3` в примере `/etc/fstab`, рассмотренном ранее.

4.4.2 Имена устройств.

В дополнение к именам разделов, перечисленных в файле `/etc/fstab`, Linux распознает номера устройств гибких и жестких дисков. Они классифицируются по типу, интерфейсу и порядку подключения. Например, первый жесткий диск, если он имеет IDE или старый MFM интерфейс, имеет имя устройства `/dev/hda`. Первый его раздел `/dev/hda1`, второй раздел `/dev/hda2`, третий раздел `/dev/hda3`... Первый раздел второго IDE устройства `/dev/hdb1`, второй раздел `/dev/hdb2` и так далее. Схема именования для обычных x86-машин с шинами ISA и PCI дана в таблице 4.2.

Драйвер устройства	Диск
`/dev/hda`	Основной (Master) диск на первом (primary) шлейфе IDE.
`/dev/hdb`	Дополнительный (Slave) диск на первом (primary) шлейфе IDE.
`/dev/hdc`	Основной (Master) диск на втором (secondary) шлейфе IDE.
`/dev/hdd`	Дополнительный (Slave) диск на втором (secondary) шлейфе IDE.

Таблица 4.2: Имена IDE устройств.

CD-ROM и стриммеры, использующие IDE/ATAPI интерфейс, также используют данные имена устройств.

Некоторые машины, в основном крутые персоналки и рабочие станции на процессорах Digital Equipment Corporation's Alpha, используют Small Computer System Interface (SCSI). Соглашения об именах SCSI устройств сильно отличаются от приведенных выше по причине большей гибкости адресации SCSI. Первый SCSI жесткий диск в системе `/dev/sda`, второй SCSI жесткий диск `/dev/sdb`... Список обычных SCSI устройств дан в таблице 4.3.

Драйвер устройства	Устройство
`/dev/sda`	Первый жесткий диск SCSI.
`/dev/sdb`	Второй жесткий диск SCSI.
`/dev/st0`	Первый стриммер SCSI.
`/dev/st1`	Второй стриммер SCSI.
`/dev/scd0`	Первое устройство чтения SCSI CD-ROM.
`/dev/scd1`	Второе устройство чтения SCSI CD-ROM.

Таблица 4.3: Имена SCSI устройств.

SCSI CD-ROM и стриммеры именуются не так, как жесткие диски на SCSI. Сменные SCSI средства, например Iomega Zip drive, следуют соглашениям наименования для несменных дисков SCSI. Использование Iomega Zip drive для резервирования описано ниже.

Стриммеры с поддержкой форматов магнитной ленты QIC-02, QIC-40 и QIC-80 имеют собственный набор имен устройств, который описан ниже.

Гибкие диски используют другую схему именования устройств.

4.4.3 Проверка файловых систем.

Бывает полезно почаще проверять вашу файловую систему на наличие поврежденных и испорченных файлов. Некоторые системы автоматически проверяют свои файловые системы во время загрузки (с помощью соответствующих команд из `/etc/rc` или `/etc/init.d/boot`).

Для проверки файловых систем используются команды, зависящие от типа файловой системы. Для файловой системы `ext2fs` (самый широко используемый тип), такой командой служит `e2fsck`. Например, команда:

```
# e2fsck -av /dev/hda2
```

проверит файловую систему `ext2fs` на `/dev/hda2` и автоматически исправит ошибки.

Обычно бывает полезно размонтировать файловую систему перед ее проверкой. Это необходимо, если `e2fsck` должна выполнить любой ремонт файловой системы. Например команда:

```
# umount /dev/hda2
```

размонтирует файловую систему на `/dev/hda2`, после чего вы можете ее проверить. Есть одно исключение, вы не можете размонтировать корневую файловую систему. Для того, чтобы проверить размонтированную корневую файловую систему вам следует использовать специальную boot/root дискету. Вы также не можете размонтировать файловую систему, если хотя бы один из ее файлов "занят" (``busy"), т.е. используется действующим процессом. Например, вы не можете размонтировать файловую систему, если хотя бы один из текущих рабочих каталогов пользователя находится на этой файловой системе. Вы получите сообщение ``Device busy", если вы попытаетесь размонтировать используемую файловую систему.

Другая файловая система использует различные формы команды `e2fsck`, такие как `efsck` и `xfsck`. На некоторых системах вы можете просто использовать команду `fsck`, которая определит тип файловой системы и выполнит соответствующую команду.

Внимание! Необходимо немедленно перезагрузить операционную систему после проверки смонтированной файловой системы, если были внесены какие-то изменения в файловую систему. Например, если `e2fsck` сообщает, что она исправила хотя бы одну ошибку в файловой системе, вам следует немедленно выполнить `shutdown -r`, чтобы перезагрузить систему. Это позволит системе ресинхронизировать информацию о файловой системе, после модификации ее с помощью `e2fsck` (то есть, снова согласовать содержимое буферов памяти с соответствующими фрагментами файловой системы на диске).

Файловая система `/proc` никогда не нуждается в проверках такого рода. `/proc` файловая система памяти, управляемая непосредственно ядром.

4.5 Использование своп-файла.

Вместо того, чтобы резервировать специальные разделы для области свопинга, вы можете использовать файл. Однако, чтобы это сделать, вы должны инсталлировать программы Linux и предварительно сделать все, что необходимо для создания файлов свопинга.

Если у вас есть инсталлированная система Linux, вы можете использовать следующие команды для создания файла свопинга. Ниже мы собираемся создать файл свопинга размером в 8208 блоков (около 8 Мбайт):

```
# dd if=/dev/zero of=/swap bs=1024 count=8208
```

Эта команда создает файл свопинга `/swap`. Замените ``count=" размером файла свопинга в блоках:

```
# mhswap /swap 8208
```

Эта команда инициализирует swap-файл; вновь замените имя и размер своп-файла соответствующими значениями:

```
# sync
# swapon /swap
```

Теперь в свопинге будет задействован файл /swap, который мы создали, после синхронизации командой sync, которая гарантирует, что файл был записан на диск.

Главная неприятность, связанная с использованием swap-файлов, состоит в том, что доступ к ним происходит через файловую систему. Это означает, что блоки, составляющие swap-файл могут быть не смежными на диске. То есть скорость свопинга при использовании swap-файла ниже, чем при использовании swap-раздела, для которой блоки всегда смежны (последовательны) и запросы на ввод/вывод происходят прямо к устройству.

Другая проблема, связанная с использованием swap-файла, это возможность испортить информацию в файловой системе: при использовании больших файлов свопинга существует шанс, что вы попортите систему, если что-то происходит неправильно. Имея раздел свопинга отдельно от файловой системы вы страхуетесь от таких неприятностей.

Использование файла свопинга может быть очень полезным, если у вас есть временная потребность в дополнительном пространстве для свопинга. Например, если вы компилируете большую программу и хотите ускорить дело, вы можете временно создать файл свопинга и использовать его в дополнение к имеющейся области свопинга.

Для того, чтобы избавиться от файла свопинга, вначале используйте swapoff:

```
# swapoff /swap
```

А теперь вы можете смело удалить файл:

```
# rm /swap
```

4.6 Работа с пользователями.

Вне зависимости от того, много у вас пользователей или нет, важно понять проблему работы с пользователем Linux. Даже если вы единственный пользователь вы должны иметь различные account для root и для себя.

Каждый человек, использующий систему, должен иметь свой собственный account (быть индивидуально зарегистрированным в системе). Редко может быть целесообразно, чтобы несколько человек входили в систему под одним именем. Здесь дело не только в безопасности, но account используется для идентификации пользователя в системе. Необходимо иметь возможность проследить, кто что делает.

4.6.1 Концепция работы с пользователями.

Система хранит следующую информацию о каждом пользователе:

username

уникальный идентификатор, присваиваемый каждому пользователю в системе. Примеры имен пользователей `larry`, `karl` и `mdw`. Могут использоваться буквы и цифры, а также нижнее подчеркивание и точка. Обычно имена пользователей ограничиваются восемью символами.

user

ID (или UID), идентификатор пользователя, уникальный номер, присваиваемый каждому пользователю системы. Система обычно отслеживает идентификаторы пользователей, а не имена.

group

ID (или GID), идентификатор группы, это идентификатор группы пользователя. Мы уже обсуждали права группы; каждый пользователь принадлежит к одной или более группам, определенных системным администратором. Подробнее об этом ниже.

password

Система также хранит в зашифрованном виде пароль пользователя. Команда `passwd` используется для установки и изменения пароля.

full name

"**Полное имя**" или "**действительное имя**" хранится вместе с именем пользователя. Например, пользователь schmoj может иметь действительное имя ``Joe Schmo''.

home directory

Домашний каталог это каталог, в который пользователь попадает при входе в систему. Каждый пользователь должен иметь свой собственный домашний каталог, обычно в `/home`.

login shell

Исходный shell это shell, который запускается для пользователя при его входе в систему. Это, например, может быть `/bin/bash` или `/bin/tcsh`.

Информация хранится в файле регистрации пользователей `/etc/passwd`. Каждая строка этого файла описывает одного пользователе; формат строки имеет вид:

```
user name:encrypted password:UID:GID:full name:home directory:login shell
```

Например, это может выглядеть так:

```
kiwi:Xv8Q981g71oKK:102:100:Laura Poole:/home/kiwi:/bin/bash
```

Первое поле, ``kiwi", имя пользователя. Следующее поле, ``Xv8Q981g71oKK", зашифрованный пароль. Пароли в читаемом виде в системе не хранятся. Сами пароли шифруются как секретные ключи. Другими словами, вы должны знать пароль, чтобы его расшифровать. Эта форма шифрации достаточно надежна. Замечание переводчика: смотря какой пароль...

Некоторые системы Linux используют "теневой пароль", в котором информация о паролях хранится в файле `/etc/shadow`. Поскольку `/etc/passwd` всем доступен, `/etc/shadow` обеспечивает дополнительный уровень секретности своей недоступностью. Теневой пароль обеспечивает и некоторые другие свойства, вроде прекращения действия пароля (пароль выдыхается).

Третье поле, ``102", идентификатор пользователя (UID). Оно должно быть уникальным для каждого пользователя. Четвертое поле, ``100", идентификатор группы (GID). Этот пользователь принадлежит к группе номер 100. Информация по группе хранится в файле `/etc/group`. Смотрите дополнительную информацию в Разделе 4.6.5.

Пятое поле, полное имя пользователя, ``Laura Poole". Последние два поля, домашний каталог пользователя (`/home/kiwi`) и исходный shell (`/bin/bash`) соответственно. Домашний каталог пользователя не обязательно должен иметь имя, совпадающее с именем пользователя (username). Однако это помогает в идентификации.

4.6.2 Добавление пользователей.

При добавлении пользователя следует совершить несколько шагов. Пользователь должен быть занесен в файл паролей `/etc/passwd` под уникальным именем и идентификатором. Должны быть описаны идентификатор группы (GID), полное имя и другая информация. Должен быть создан домашний каталог пользователя и установлены необходимые права доступа. Домашний каталог должен быть снабжен необходимыми файлами инициализации shell. Должны быть выполнены и другие работы по конфигурации (например, создан spool для входной почты).

Хотя очень несложно добавлять пользователей вручную (я так делаю), когда вы сопровождаете систему со многими пользователями, легко что-то упустить. Самый простой способ регистрации пользователей, это использование диалоговой программы,

которая задаст вам все необходимые вопросы и автоматически скорректирует все необходимые системные файлы. Эта программа называется `useradd` или `adduser`, в зависимости от вашего дистрибутива. Страницы руководства для этих программ должны быть достаточно понятными. `adduser` использует в своей работе файл настроек `/etc/adduser.conf`, который задает настройки по умолчанию для нового пользователя системы.

Вот пример обычного файла `/etc/adduser.conf`:

```
# /etc/adduser.conf 'adduser' configuration
# See adduser (8) and adduser.conf(5) for full documentation
# The DSHELL variable specifies the default login shell on your
# system
DSHELL=/bin/bash

# The DHOME variable specifies the directory containing users' home
DHOME= home

# If GROUPHOMES is "yes". then the home directories will be created as
# /home/groupname/user
GROUP HOMES=no

# If LETTERHOMES is "yes", then the created home directories will have
# an extra directory - the first letter of the user name. For example'
# home u user
LETTERHOMES=no

# The SKEL variable specifies the directory containing "skeletal" user
# files; in other words, files such as a sample profile that will be
# copied to the new user's home directory when it is created
SKEL=/etc/shel

# FIRST_SYSTEM_UID to LAST_SYSTEM_UID inclusive is the range for UIDs
# for dynamically allocated administration and system accounts
FIRST_SYSTEM_UID=100
LAST_SYSTEM_UID=999

# FIRST_UID to LAST_UID inclusive is the range of UIDs of dynamically
# allocated user accounts
FIRST_UID=1000
LAST_UID=29999

# The USERGROUPS variable can be either "yes" or "no"  If "yes" each
# created user will be given their own group to use as a default, and
# their home directories will be g+s If "no", each created user will
# be placed in the group whose gid is USERS_GID (see below).
USERGROUPS=yes

# If USERGROUPS is "no", then USERS_GID should be the GID of the group
# 'users' (or the equivalent group) on your system
USERS_GID=100
```

В дополнение к предустановленным переменным, которые нужны команде `adduser`, `/etc/adduser.conf` определяет где по умолчанию лежат системные файлы настроек для каждого пользователя. В данном примере они лежат в каталоге `/etc/skel`, как определено в переменной `SKEL=`. Файлы в данном в каталоге, являются настройками системы по умолчанию для каждого пользователя. Это, например, `.profile`, `.tcshrc`

или .bashrc, которые автоматически копируются в домашний каталог нового пользователя командой adduser. Если Вы считаете нужным разместить в домашнем каталоге каждого нового пользователя еще какие-то файлы настроек (например, для иксов), положите их в каталог /etc/skel.

4.6.3 Удаление пользователей.

Удаление пользователей может быть выполнено с помощью команд userdel или deluser в зависимости от каждого конкретного дистрибутива.

Если вы пожелаете временно "отключить" пользователя от системы, (без удаления его аккаунта), вы можете просто приписать звездочку (``*'') в поле пароля в файле пароля /etc/passwd. Например, изменение пароля у kiwi в файле /etc/passwd:

```
kiwi:*Xv80981g71oKK:102:100:Laura Poole:/home/kiwi:/bin/bash
```

закроет для kiwi вход в систему.

4.6.4 Настройка атрибутов пользователя.

После создания пользователя вам может потребоваться изменить его атрибуты, такие как домашний каталог и пароль. Простейший способ, это прямо изменить значения в /etc/passwd. Чтобы установить пароль пользователя используйте команду passwd:

```
# passwd larry
```

сменит пароль для larry. Только root может менять пароли других пользователей. Обычный пользователь может сменить только свой пароль.

В некоторых системах имеются команды chfn и chsh, позволяющие пользователю самому устанавливать свое полное имя и исходные атрибуты shell. Если таких команд нет, то пользователи должны просить системного администратора изменить эти атрибуты для них.

4.6.5 Группы.

Как мы говорили, каждый пользователь принадлежит к одной или более группам. Единственное значение группы замыкается на права доступа к файлу, как вы помните из раздела 3.10. Каждый файл имеет "групповое владение" (``group ownership''), то есть хранит права доступа, которые определяют, как члены группы могут обращаться с файлом.

Существует несколько групп, определяемых системой, вроде bin, mail и sys. Пользователи не могут принадлежать к какой-либо из этих групп. Эти группы используются для системных файлов. А пользователи, наоборот, принадлежат к

специальной группе, например users. Если вам хочется больше проблем, вы можете поддерживать несколько групп пользователей, например student, staff и faculty.

Файл /etc/group содержит информацию о группах. Формат каждой строки следующий:

```
group name:password:GID:other members
```

Примерами групп могут быть:

```
root:*:0
users:*:100:mdw,larry
guest:*:200
other:*:250:kiwi
```

Первая группа, root, специальная системная группа зарезервированная для root. Следующая группа, users, для обычных пользователей. Она имеет идентификатор группы (GID) 100. К этой группе имеют доступ пользователи mdw и larry. Помните, что в /etc/passwd каждый пользователь получил выдаваемый по умолчанию GID. Но пользователь может принадлежать более, чем к одной группе, путем добавления имен пользователей в другие группы (в строки файла /etc/group). Команда groups перечисляет, в какие группы вы входите.

Третья группа, guest, для гостей, а other для "других" пользователей. Пользователь kiwi имеет доступ и в эту группу.

Как вы можете видеть, поле "пароль" в /etc/group редко используется. Иногда оно используется для установления пароля на доступ к группе. Это редко бывает нужно. Для защиты привилегированных групп от обычных пользователей (с помощью команды newgroup) установите в поле пароля звездочку ("*").

Команды addgroup или groupadd могут быть использованы для добавления групп в вашу систему. Обычно легче просто самому добавить запись в /etc/group, поскольку не требуется других настроек при добавлении группы. Для удаления группы просто удалите соответствующую запись в /etc/group.

4.6.6 Обязанности администратора системы.

Поскольку администратор системы имеет много возможностей и ответственности, возможны злоупотребления. Я знаю так называемых "администраторов системы", которые читают почту других пользователей, удаляют файлы пользователей без предупреждения, и вообще ведут себя подобно детям которым попалась такая мощная "игрушка".

Поскольку root имеет в системе такие привилегии, требуется определенный уровень зрелости и самоконтроля, чтобы использовать этот аккаунт так, как это было задумано: для эксплуатации системы. Существует негласный закон чести в отношениях администратора с пользователями. Как вы будете себя чувствовать, если системный

администратор читает ваши письма и просматривает ваши файлы? До сих пор нет достаточно серьезной юридической основы для неприкосновенности личной информации в многопользовательских компьютерных системах. В системах семейства UNIX пользователь root имеет возможность преодолевать все штатные механизмы защиты системы. Важно, чтобы у администратора были доверительные отношения с пользователями системы. Невозможно переоценить важность этого.

4.6.7 Взаимодействие с пользователями.

Безопасность UNIX довольно слабая от рождения. Вопросы безопасности были додуманы "в догонку": исходно система создавалась в неформальной атмосфере, когда все вмешивались в работу друг друга. Благодаря этому, даже несмотря на меры безопасности, у нормального пользователя существуют возможности причинить системе вред.

Системный администратор может выбрать две тактики взаимодействия с злоупотребляющими пользователями. Это может быть параноидная тактика и тактика доверия. Системный администратор с паранойей обычно своими действиями наносит больше вреда, чем предотвращает. Одна из моих любимых присказок: "Никогда не списывай на зловредность то, что можно списать на тупость". Взгляните с другой стороны, большинство пользователей не имеют возможностей и знаний, чтобы причинить реальный вред системе. 90% процентов того, что делает пользователь, причиняя вред системе (например, забивая пользовательский раздел огромными файлами или выполняя сразу несколько экземпляров громадной программы), он делает просто не подозревая, что он кому-то создает проблемы. Мне приходилось сталкиваться с пользователями, которые были источниками огромных неприятностей, но они действовали по простоте душевной, а не со зла.

Когда вы имеете дело с пользователями, которые опасны потенциально, не накидывайтесь на них с обвинениями. Старое правило "презумпции невиновности" все еще не отменили. Лучше всего поговорить с пользователем, поспрашивать о его проблемах, вместо того, чтобы идти на конфронтацию. Самое плохое, это пытаться отвечать ему "встречными" неприятностями. Это создаст вокруг вас, системного администратора, много подозрений, поставит под сомнение вашу способность корректно сопровождать систему. Если пользователь решит, что вы не верите ему или даже не любите, он может обвинить вас в том, что вы удаляете его файлы и вообще подсматриваете. Вряд ли вы хотите оказаться в такой ситуации.

Если вы убедились, что пользователь действительно пытается ``взломать" систему или умышленно ей вредит, старайтесь не отвечать угрозами на угрозы. Вместо этого просто предупредите его, но сохраняйте гибкость. Во многих случаях вы можете "схватить его за руку" в процессе свершения вредительства, вот тут и предупредите. Скажите, чтобы он так больше не делал. Но если вы снова его поймаете на вредительстве, то убедитесь, что это действительно намеренно. Я просто не смогу перечислить все случаи, когда оказывалось, что неприятность была либо случайной, либо я сам был виноват.

4.6.8 Установление правил.

Лучший способ управления системой, это управление без применения железного кулака. Может так вы хорошо управляли в армии, но это не для UNIX. Имеет смысл

сделать простой и гибкий свод руководств для пользователей, но чем меньше у вас будет правил, тем меньше шансов их нарушить. Даже если ваши правила использования системы очень ясны и разумны, пользователи все равно время от времени будут их без злого умысла нарушать. Это, в особенности, относится к новичкам в UNIX, которые еще только изучают основы системы. Да и вы сами можете время от времени рассылать гигабайтные файлы всем пользователям системы... Пользователям надо помочь понять правила и объяснить, зачем они нужны.

Если вы создали руководство для пользователей системы, убедитесь, что причины введения тех или иных правил им понятны. Если вы этого не сделаете, пользователи творчески подойдут к тому, как обходить эти правила. может быть и не сознавая, что они их действительно обходят.

4.6.9 Что все это значит.

Мы не можем до последней детали расписать вам, как эксплуатировать систему. Большая часть философии зависит от того, как вы используете систему. Если у вас много пользователей, то это сильно отличается от того, когда их мало, или вообще вы один. Но при любом раскладе очень полезно задуматься, что в данной конкретной системе *действительно* означают слова "системный администратор" (или "администратор системы").

Должность администратора системы не делает вас крутым юниксистом. На свете много системных администраторов, которые мало что знают о UNIX. Похоже, что существует много "нормальных" пользователей, которые, знают о UNIX больше любого системного администратора. Пребывание в должности администратора не дает вам права использовать угрозы в адрес пользователей. Именно потому, что система дает вам привилегию устроить из файлов пользователя все, что угодно, вы не имеете никакого права это делать.

Наконец, быть системным администратором, это невесть что. При этом не имеет значения, опекаете вы маленький 386-ой или суперкомпьютер Cray. Знание заветного пароля `root` не принесет вам денег и славы; оно поможет сопровождать систему и поддерживать ее работоспособность. Вот так.

4.7 Архивация и компрессирование (сжатие) файлов.

Прежде, чем мы сможем говорить о сохранении (резервировании) программ, мы должны представить инструменты, используемые для архивации файлов и программ в системах UNIX.

4.7.1 Использование `tar`.

Команда `tar` наиболее часто используется для архивации файлов. Формат команды `tar`:

```
tar opilons files
```

где *options* есть список команд и опций для tar, а *files* список файлов добавляемых в архив или извлекаемых из него.

Например, команда:

```
# tar cvf backup.tar /etc
```

упакует все файлы, содержащиеся в /etc, в архив tar под именем backup.tar. Первый аргумент команды tar, ``cvf', это (внутренняя) команда tar. ``c" указывает tar создать новый архивный файл. Опция ``v" заставляет tar выводить имя каждого архивируемого файла. Опция ``f" говорит, что следующий аргумент, backup.tar, имя созданного архивного файла. Остальные аргументы команды tar, имя файла архива и имя добавляемого в архив каталога. Команда:

```
# tar xvf backup.tar
```

распакует архивный файл в текущий каталог. Это может быть иногда и небезопасным занятием, когда извлеченные из архива файлы затирают существовавшие файлы.

Поэтому перед извлечением архивированных файлов важно знать, где файлы следует распаковать. Например, вы заархивировали следующие файлы: /etc/hosts, /etc/group и /etc/passwd. Если вы используете команду:

```
# tar cvf backup,tar /etc/hosts /etc/group /etc/passwd
```

в начало имени каждого файла добавится каталог с именем /etc. Чтобы извлечь файлы в нужное место, вам потребуется использовать следующие команды:

```
# cd
# tar wf backup tar
```

поскольку файлы извлечены с сохраненным в архиве путем.

Если вы заархивировали файлы командой:

```
# cd /etc
# tar cvf hosts group passwd
```

имя каталога не сохраняется в архивном файле. Поэтому вы должны выполнить ``cd /etc" перед извлечением файлов. Вы обратили внимание: то, как вы создали архивный файл сильно влияет на то, в каком месте его следует извлекать. Команда:

```
# tar tvf backup.tar
```

может быть использована для просмотра оглавления архивного файла перед его распаковкой. Таким образом вы можете посмотреть, к какому каталогу относятся архивированные файлы и сможете извлечь файлы из архива в нужном месте.

4.7.2 gzip и compress.

В отличие от архивирующих программ для MS-DOS, `tar` не сжимает автоматически файлы в процессе архивирования. Поэтому, если вы архивируете два одномегабайтных файла, результирующий архивный файл будет размером два мегабайта. Команда `gzip` может использоваться для сжатия файла (сжимаемый файл не обязан быть `tar`-файлом). Команда:

```
# gzip -9 backup tar
```

сожмет `backup.tar` и оставит вас наедине с `backup.tar.gz`, сжатой версией файла. Опция `-9` говорит команде `gzip`, что следует использовать максимально возможную компрессию.

Команда `gunzip` может быть использована для расжатия сжатого файла. С аналогичным эффектом вы можете использовать команду ``gzip -d''.

`gzip` сравнительно новый инструмент в кругах, приближенных к UNIX. Долгие годы вместо этого использовалась команда `compress`. Однако, по нескольким причинам (тут и патентные проблемы относительно алгоритма, и то, что `gzip` значительно эффективнее) `compress` оказался не у дел.

Обработанные командой `compress` файлы заканчивались расширением `.Z`. Например, `backup.tar.Z` это сжатая `compress` версия файла `backup.tar`, а `backup.tar.gz` сжатая `gzip` версия. Чтобы еще надежнее запутать дело, для обозначения `gzip` файлов некоторое время использовалось расширение `.z` (маленькая ``z''). В настоящее время официальное расширение .gz. Команда `uncompress` используется для развертывания файла, который был обработан командой `compress`. Функционально она равнозначна команде `compress -d`. Но команда `gunzip` тоже знает, как обращаться с такими файлами.

4.7.3 Можно вместе.

Чтобы заархивировать и сжать группу файлов, вы можете использовать команды:

```
# tar cvf backup.tar /etc
# gzip -9 backup.tar
```

Результат будет `backup.tar.gz`. Для распаковки этого файла используйте обратную последовательность команд:

```
# gunzip backup.tar.gz
# tar xvf backup.tar
```

Разумеется, всегда следует убедиться перед распаковкой файла, что вы в нужном каталоге.

Вы можете опереться на некоторую сообразительность UNIX, позволяющего сделать это одной командной строкой:

```
# tar cvf - /etc|gzip -9c>backup.tar.gz
```

Здесь мы посылаем tar-файл, сформированный из /etc, в файл ``-", который представляет стандартный вывод. Результат по конвейеру поступает на ввод команды gzip, которая сжимает этот файл и результат сохраняет в backup.tar.gz. Опция -c команды gzip говорит, что вывод команды gzip посылает результат на стандартный вывод, который перенаправляется на backup.tar.gz.

Единственная команда для распаковки этого архива:

```
# gunzip -c backup.tar.gz|tar xvf -
```

Опять, команда gunzip рассжимает содержимое файла backup.tar.gz и посылает результирующий файл на стандартный вывод. Он по конвейеру передается команде tar, которая читает файл ``-", что в данном случае олицетворяет стандартный вывод.

К счастью, команда tar также содержит опцию z, автоматически сжимая-расжимая файлы, используя алгоритм компрессии gzip.

Команда:

```
# tar cvf backup.tar.gz /etc
```

эквивалентна:

```
# tar cvf backup.tar /etc
# gzip backup.tar
```

Как и команда:

```
# tar xvf backup.tar.Z
```

может быть использована вместо:

```
# uncompress backup.tar.Z
# tar xvf backup.tar
```

За дополнительной информацией обратитесь к man-руководству по tar и gzip.

4.8 Использование дискет и осуществление резервирования.

Дискеты часто используются как средство резервирования. Если у вас нет (стриммера), можно использовать дискеты (хотя они медленнее и, в некотором смысле, менее надежны).

Как упоминалось выше, дискеты должны быть отформатированы программами MS-DOS FORMAT.COM или Linux fdformat.

Некоторые имена устройств и доступные в Linux форматы дискет приведены в таблице 4.4.

```
Floppy device driver   Format

/dev/fd0d360           Double density 360Kb 5.25
/dev/fd0h1200          High density 1.2 MB 5.25
/dev/fd0h1440          High density 1.44 MB 3.5
```

Таблица 4.4: Форматы дискет в Linux.

Устройства начинаются с `fd0`, что соответствует первому дисководу для дискет, который называется `A:` в системе MS-DOS. Имя второго дисковода `fd1`. Вообще, ядро Linux может определить формат дискеты, которая уже отформатирована: можно просто указать `/dev/fd0` и оставить определение формата на долю системы. Но когда вы используете еще не отформатированную дискету, такой фокус не пройдет.

Полный перечень устройств и их имен в Linux опубликован в *Linux Allocated Devices,*, автор H. Peter Anvin (см. приложение A).

Вы можете использовать дискеты также для хранения отдельных файловых систем, в этом случае вы должны **смонтировать** (**mount**) дискету для получения доступа к ее данным. Подробности в разделе 4.8.4. разделе 4.8.4.

4.8.1 Резервирование на дискеты.

Простейший способ резервирования на дискетах, это использование команды `tar`. Команда:

```
# tar cvfsM /dev/fd0
```

сделает полную копию вашей системы с использованием дисковода `/dev/fd0`. Опция ``M'' позволяет копировать на несколько дискет (multivolume backup); то есть, когда одна дискета заполнится, `tar` запросит следующую. Команда:

```
# tar xvfsM /dev/fdO
```

может быть использована для полного восстановления. Этот метод может быть также использован для лент (`/dev/rmt0`). Подробности о лентах в разделе 4.8.3.

Существует несколько других программ для осуществления многотомного резервирования. Вам могут пригодиться программы "`backflops`", которые можно взять на `tsx-11.mit.edu`.

Создание полной копии системы может быть весьма время-ресурсоемким. Большинство системных администраторов использует **"инкрементальную"** стратегию резервирования. Каждый месяц производится полное копирование, а каждую неделю только тех файлов, которые были модифицированы в эту неделю. В этом случае, если вы грохнете свою систему в середине месяца, вы можете просто восстановить состояние на начало месяца, а затем восстановить понедельные изменения.

Команда `find` может быть полезна в выискивании файлов, которые изменились после какой-то даты. Несколько скриптов (командных файлов на shell) для инкрементального резервирования можно найти на `sunsite.unc.edu`.

4.8.2 Резервирование на Zip диски.

Резервирование на Zip диски ничем не отличается от резервирования на дискеты, но поскольку Zip диски имеют емкость 100 мегабайт, на дискету помещается меньше. Как правило, архив может быть целиком выгружен на один Zip диск.

Zip диски доступны с тремя типами интерфейса: SCSI, IDE и параллельный порт PPA. Поддержка Zip дисков не входит в настройки скомпилированного ядра Linux по умолчанию, но она может быть указана при построении заказного ядра для вашей системы. Подробнее о компиляции ядра см. ниже.

SCSI и PPA интерфейсы Zip дисков используют SCSI интерфейс и соглашения об именах SCSI устройств, которые описаны здесь.

Zip отформатированы производителем в файловой системе a MS-DOS. Вы можете использовать имеющуюся файловую систему MS-DOS, которая должна поддерживаться Вашим ядром Linux kernel, или использовать `mke2fs` для создания на диске файловой системы Linux.

Zip диск, который смонтирован на первом SCSI устройстве, называется `/dev/sda4`.

```
# mount /dev/sda4 /mnt
```

Для доступа к диску нужна отдельная точка монтирования, например, `/zip`. Чтобы ее создать скомандуйте от имени `root`:

```
# mhdir /zip
# chmod 0755 /zip
```

Теперь Вы можете использовать `/zip` для монтирования файловой системы с Zip диска.

Запись архива на Zip диск аналогично записи на дискету. Для архивации и сжатия каталога `/etc` на смонтированный Zip диск скомандуйте:

```
# tar zcvf /zip/etc.tgz /etc
```

Данная команда может быть выполнена из произвольного каталога, поскольку указывает полное имя файла. Архив будет назван `etc.tgz`, если на Zip диске файловая система MS-DOS, в которой имя файла должно соответствовать соглашениям MS-DOS 8+3, в противном случае имя файла будет обрезано.

Для распаковки данного архива, скомандуйте:

```
# cd /
# tar zxvf /zip/etc.tgz
```

Для создания, например, файловой системы ext2 на Zip диске, скомандуйте (на *размонтированном* Zip диске):

```
# mke2fs /dev/sda4
```

Смонтировав Zip диск с файловой системой ext2, можно зарезервировать все файловые системы одной командой:

```
# tar zcvf /zip/local.tar.gz /usr/local
```

Заметьте, что резервирование программой `tar` все еще предпочтительнее, чем архивация командой `cp -a`, поскольку `tar` сохраняет время модификации файлов.

4.8.3 Резервирование на стриммер.

Архивация на ленту аналогично архивации на дискеты, только отличается драйвер устройства. Ленты также форматируются и обрабатываются по-другому, чем гибкие дискеты. Некоторые драйверы стриммера для Linux перечислены в таблице 4.5.

Драйвер стриммера	Формат
`/dev/rft0`	Лента QIC-117, перемотка при закрытии.
`/dev/nrft0`	Лента QIC-117, нет перемотки при закрытии.
`/dev/tpqic11`	Лента QIC-11, перемотка при закрытии.
`/dev/ntpqic11`	Лента QIC-11, нет перемотки при закрытии.
`/dev/qft0`	Стриммер габарита 1/2, перемотка при закрытии.
`/dev/nqft0`	Стриммер габарита 1/2, нет перемотки при закрытии.

Таблица 4.5: Имена устройств стриммера.

Некоторые стриммеры используют контроллер дискет и управляются ftape драйвером, который описан ниже. Инсталляция модуля драйвера ftape описана здесь. Имена SCSI стриммеров перечислены а таблице 4.3.

Для архивации каталога `/etc` на стриммер с применением `tar` скомандуйте:

```
# tar cvf /dev/qft0 /etc
```

Для распаковки файлов с ленты, скомандуйте:

```
# cd /
# tar xvf /dev/qft0
```

Ленты, как и дискеты, перед использованием должны быть отформатированы. Драйвер ftape может форматировать ленты под Linux. Для форматирования ленты формата QIC-40 скомандуйте:

```
# ftformat -format-parameter qic40-205ft -mode-auto --omit-erase -discard-header
```

Другие стриммеры имеют свой софт для форматирования. Посмотрите документацию на стриммер или на связанный с ним драйвер устройства.

Перед извлечением ленты из стриммера ее надо перемотать и записать на ленту буфера ввода-вывода. Это аналог размонтирования файловой системы на дискете перед ее извлечением, поскольку стриммер также буферизуется в памяти. Стандартной командой UNIX для управления стриммером является `mt`. Ваша система может и не поддерживать эту команду. Ftape драйвер имеет подобную команду, `ftmt`, которая используется, чтобы управлять операциями с лентами.

Для перемотки ленты перед ее извлечением, скомандуйте:

```
# ftmt -f /dev/qft0 retension
```

Конечно, впишите правильное имя стриммера для вашей системы.

Чтобы получить состояние стриммера с отформатированной лентой в нем, используйте команду:

```
# ftmt -f /dev/qft0 status
```

4.8.4 Использование дискет в качестве файловых систем.

Вы можете создать файловую систему на дискете точно также, как в разделе жесткого диска. Например:

```
# mke2fs /dev/fd0 1440
```

создает файловую систему на дискете на `/dev/fd0`. Размер файловой системы должен соответствовать размеру дискеты. Дискеты high-density 3.5" емкостью в 1.44 Мбайт или 1440 блоков. Дискеты high-density 5.25" емкостью в 1200 блоков. Указание размера файловой системы нужно, если система не может сама определить емкость дискеты.

Для того, чтобы иметь доступ к дискете, вы должны примонтировать содержащуюся на ней файловую систему. Команда:

```
# mount /dev/fd0 /mnt
```

примонтирует дискету, находящуюся на `/dev/fd0` к каталогу `/mnt`. Теперь все файлы, находящиеся на дискете, будут находиться в каталоге `/mnt` вашего жесткого диска.

"Точка монтирования" (каталог, к которому вы примонтируете файловую систему) должен существовать, когда вы применяете команду `mount`. Если он не существует, создайте его с помощью команды `mkdir`, как описано здесь.

Дополнительную информацию по файловым системам, монтированию и точкам монтирования смотрите здесь.

Важное замечание! Ввод/вывод на дискету буферизируется точно также, как и для жесткого диска. Когда вы меняете дискету, вы не должны видеть горящую лампочку дисковода (пока ядро работает с буферами ввода/вывода). Важно, чтобы вы не извлекали дискету из дисковода до ее размонтирования, которое можно выполнить командой:

```
# umount /dev/fd0
```

Нельзя просто взять и вытащить дискету, как в MS-DOS. При замене дискет сначала размонтируйте одну командой `umount`, а затем примонтируйте вторую командой `mount`.

4.9 Модернизация и инсталляция программ.

Другая обязанность системного администратора: модернизация и инсталляция новых программ.

Сообщество приверженцев Linux очень динамично. Новые версии ядра появляются каждые несколько недель, да и другие программы изменяются не менее часто. Поэтому новые пользователи Linux часто чувствуют необходимость в постоянной модернизации (upgrade) своей системы, чтобы поспевать за изменениями, идущими лихой поступью. Это необходимо и это и потеря времени: отслеживать все изменения в мире Linux. Просто у вас может абсолютно все время уходить на модернизацию системы и лишь оставшееся - на собственно использование системы.

Ну, так когда желаете заняться модернизацией? Некоторые нутром чувствуют, что заниматься модернизацией пристало тогда, когда появилась новая версия дистрибутива, например, когда появляется новая версия Slackware. Многие пользователи Linux каждый раз при этом полностью переинсталлируют свою систему. Это тоже потеря времени. Обычно изменения от версии к версии Slackware незначительные. Бессмысленно переписывать и переинсталлировать 30 дисков, когда только 10% программ были действительно модифицированы.

Лучший вариант модернизации системы это ручная работа: модернизируйте только те программные пакеты, про которые вы точно знаете, что их стоит менять. Это многих пугает: они хотят знать, что менять, и как, и что они теряют, если не модернизируют. Залог успеха в Linux это преодолеть боязнь принципа "сделай сам", одного из фундаментальных принципов Linux.

Действительно, благостное состояние пользователя работающей и хорошо настроенной системы враз меняется при переинсталляции, поскольку, без сомнения, приводит и к перенастройке всего и вся, к тому, что опять все не работает, как это было при первой инсталляции системы. Так что определенные сеансы самопсихотерапии необходимы, чтобы иметь деловой настрой. Все, что требуется - это немножко "ноу-хау" по модернизации системы.

Вы обнаружите, что когда вы модернизируете одну компоненту вашей системы, другие вещи не должны ломаться. Например, большая часть моей системы оставлена со времен древней 0.96 MCC Interim installation. Тем не менее, я использую новейшую версию ядра и библиотек без проблем. Большей частью бессмысленно заниматься модернизациями, чтобы "не отстать от моды". Суета все это. Это вам не MS-DOS или Microsoft Windows. У нас нет серьезных причин обязательно работать на новейшей во все времена версии системы. Если вы осознаете, что вам действительно нужны некоторые вещи из новой версии, тогда модифицируйте на здоровье. А если нет, то лучше не надо. Другими словами модернизируйте только то, что надо, и только тогда, когда надо. Не модернизируйте во имя модернизации.

Наиболее важная часть вашей системы, как возможный объект модернизации, это ядро, библиотеки и компилятор `gcc`. Это три ключевые части вашей системы, и в некоторых случаях они бывают взаимозависимыми. Большая часть остального хозяйства вашей системы и без периодических модернизаций сойдет.

4.9.1 Модернизация ядра.

Обновление ядра вопрос получения источников новой версии и компилирования их. Это безболезненная процедура, но Вы можете столкнутся с проблемами, если Вы пробуете использовать версию для разработчиков. Версия ядра имеет две части: собственно версию ядра и patchlevel. На момент написания данного документа, последнее устойчивое ядро `2.0.33`. `2.0` версия, и 33 patchlevel. Нечетные версии (например, `2.1`), версии для разработчиков. Если не хотите проблем, держитесь от них подальше! Вообще, можно без особых проблем обновлять ядро до следующего patchlevel, но обновление до новой версии потребует обновления системных утилит, которые тесно связаны с ядром.

Исходники ядра Linux можно найти на многих (Linux FTP сайтах. На `sunsite.unc.edu`, исходники лежат в каталоге `/pub/Linux/kernel` в подкаталогах, соответствующих номерам версий.

Исходники выпускаются в файле формата .tar.gz. Например, файл с исходниками ядра 2.0.33 называется `linux-2.0.33.tar.gz`.

При распаковке исходников в каталог `/usr/src`, создается подкаталог `/usr/src/linux`. Обычно `/usr/src/linux` является символической ссылкой на другой каталог с номером версии, например `/usr/src/linux-2.0.33`. Таким образом, можно ставить и тестировать исходники новой версии, не удаляя исходников старой. Ссылка создается так:

```
# cd /usr/src
# mkdir linux-2.0.33
# rm -r linux
# In -s linux-2.0.33 linux
# tar xzf linuK-2.0.33.tar.gz
```

При апгрейде на новый patchlevel имеющейся версии ядра, файлы заплаток могут сильно облегчить жизнь. Полный исходник ядра в сжатом `gzip` виде занимает около 7 MB. Для апгрейда с ядра версии `2.0.31` до версии `2.0.33`, Вам надо скачать файлы заплаток `patch-2.0.32.gz` и `patch-2.0.33.gz`, которые есть на многих FTP сайтах. Положите полученные заплатки в каталог `/usr/src`, и последовательно примените их к ядру, что обновит исходник до новой версии. Например, так:

```
# cd /usr/src
# gzip -cd patch-2.0.32.gz|patch -p0
# gzip -cd patch-2.0.33.gz|patch -p0
```

После распаковки и применения патчей (заплаток), Вы должны удостовериться, что три символических ссылки в `/usr/include` правильны для вашего дистрибутива ядра. Создать данные ссылки можно так:

```
# cd /usr/include
# rm -rf asm linus scsi
# In -s /usr/src/linux/include/asm-i386 asm
# In -s /usr/src/linux/include/linux linux
# In -s /usr/src/linux/include/scsi scsi
```

После их создания Вам уже не надо будет создавать их при установке каких-либо заплат или новых версий ядра. См. раздел 3.11 для подробной информации о символических ссылках.

Чтобы скомпилировать ядро, Вы должны иметь `gcc` компилятор C, установленный в Вашей системе. Для компиляции ядра версий `2.0` нужен `gcc` версии `2.6.3` или более старшей.

Сначала `cd` в `/usr/src/linux`. Команда `make config` запросит у Вас настройки ядра. На данном шаге Вы выбираете аппаратные средства, которые ядро будет поддерживать. Самая большая ошибка, когда не включают поддержку контроллера

жесткого диска. Без правильной поддержки жесткого диска в ядре, система не будет даже загружаться. Если Вы не уверены относительно того, что делает данная опция, есть короткое описание, доступное при нажатии ? и Enter.

Затем выполните команду `make dep` для создания всех зависимостей в ядре. Это важный шаг. Команда `make clean` удаляет старые двоичные файлы из дерева исходников ядра.

Команда `make zImage` наконец откомпилирует ядро и положит его в файл `/usr/src/linux/arch/i386/boot/zImage`. Ядра Linux на системах Intel всегда сжимаются. Иногда ядро, которое Вы хотите компилировать, слишком большое, чтобы быть сжатым системой сжатия, которую использует `make zImage`. Компиляция ядра, которое является слишком большим, завершится сообщением об ошибке: `Kernel Image Too Large`. Если это случается, попробуйте команду `make bzImage`, которая использует систему сжатия поддержкой больших ядер. Ядро будет записано в `/usr/src/linux/arch/i386/boot/bzImage`.

Как только откомпилируете ядро, Вы должны или скопировать его на дискету начальной загрузки командой наподобие ``cp zImage /dev/fd0'') или установить ядро для загрузки с жесткого диска с помощью LILO. См. подробности про LILO.

4.9.2 Добавление драйвера устройства к ядру.

Выше описано использование Iomega Zip дисковода для резервирования. Поддержка для Iomega Zip, подобно многим другим устройствам, вообще не компилируется в дистрибутивные ядра Linux: разнообразие устройств просто слишком большое, чтобы поддерживать все в пригодном для использования ядре. Однако, исходный текст для драйвера устройства параллельного порта для работы с Zip включен как часть исходного текста ядра. Этот раздел описывает, как добавить, что поддержку для Iomega Zip параллельного порта и как сделать, чтобы он сосуществовал с принтером, подключенным к другому параллельному порту.

Вы должны сформировать и поставить заказное ядро Linux kernel, как описано в предыдущем разделе.

Выбор устройства устройства Zip `ppa` требует ответа `Y` на соответствующие вопросы на шаге конфигурации, когда Вы определяете конфигурацию заказного ядра. В частности, устройство `ppa` требует ответа ``Y'' на три вопроса:

```
SCSI support? [Y/n/m] Y
SCSI disk support? [Y/n/m] Y
IOMEGA Parallel Port Zip Drive SCSI support? [Y/n/m] Y
```

После успешного выполнения `make config` выполните `make dep`, `make clean`и `make zImage` для создания ядра, Вы должны сообщить ядру, как установить драйвер. Это делается через командную строку LILO. Как описано в разделе 4.2.1, файл настройки

LILO, `/etc/lilo.conf` имеет секции для каждой ОС, которую он знает и директивы для запроса параметров у пользователя при начальной загрузке машины.

Есть директива ``append=", которая позволяет добавлять при загрузке информацию для различных драйверов устройств в командной строке. В данном случае, драйвер Iomega Zip `ppa` требует свободного адреса I/O port и свободного прерывания. Это аналогично определению отдельных устройств принтера `LPT1:` и `LPT2:` под MS-DOS.

Например, если Ваш принтер использует шестнадцатеричный (основание 16) адрес порта `0x378` (см. руководство по Вашей карте параллельных портов, если адрес неизвестен) и опрашивается (то есть не требуется линия IRQ, обычная конфигурация Linux), Вы должны вписать в `/etc/lilo.conf` строку:

```
append="lp=0x378,0"
```

Стоит заметить, что Linux автоматически распознает один порт `/dev/lp` при загрузке, но в специальных конфигурациях параметры загрузки необходимы.

``0" после адреса порта сообщает, чтобы ядро *не* использовало IRQ (запрос прерывания) для принтера. Это обычно применяется потому, что принтеры намного медленнее, чем CPU, более медленный метод обращения к устройствам ввода-вывода, известен как **polling**, где ядро периодически проверяет состояние принтера, что позволяет компьютеру не отставать от принтера.

Однако, устройства которые функционируют на более высоких скоростях, подобно последовательным линиям и дискам, требуют **IRQ**, или линию запроса прерывания. Это аппаратный сигнал, посылаемый устройством процессору всякий раз, когда устройство требует внимания процессора; например, если устройство имеет данные для передачи на процессор. Процессор останавливает любой процесс и обрабатывает запрос прерывания от устройства. Диск Zip `ppa` требует свободного прерывания, которое должно соответствовать прерыванию, которое установлено на плате принтера, к которой подключен Zip. В настоящее время драйвер устройства Linux `ppa` не поддерживает формирование цепочки устройств параллельного порта, так что для Zip `ppa` и разных принтеров должны использоваться отдельные порты.

Определить, какие прерывания уже используются в Вашей системе, можно командой:

```
# cat /proc/interrupt
```

Она отображает список устройств и линий IRQ, которые они используют. Однако, Вы также должны быть внимательным, чтобы не использовать автоматически конфигурируемое прерывание последовательного порта, которое не может быть перечислено в файле `/proc/interrupt`. Linux Documentation Project Serial HOWTO, доступное на сайтах, перечисленных в приложении A, подробно описывает конфигурацию последовательных портов.

Вы должны также проверить аппаратные параметры настройки различных плат интерфейса на вашей машине, открывая корпус машины и визуально проверяя параметры настройки устройства в случае необходимости, чтобы гарантировать, что Вы не выбрали линию IRQ, которая уже используется другим устройством. Много

устройств, борющихся за линию прерывания, возможно, одна наиболее общая причина нефункционирования Linux системы.

Типичный файл `/proc/interrupt` выглядит примерно так:

```
0  6091646 timer
1  40690   keyboard
2  0       cascade
4  284686  + serial
13 1       math error
14 192560  + ide0
```

Нам интересен первый столбец. Он перечисляет номера линий IRQ, которые использованы в Вашей системе. Для `ppa` драйвера мы хотим выбрать линию, которая здесь еще *не* перечислена. IRQ 7 часто хороший выбор, поскольку она редко используется в заданных по умолчанию конфигурациях системы. Мы также должны определить адрес порта, который использует устройство `ppa`. Этот адрес должен быть физически сконфигурирован на плате интерфейса. Параллельным портам ввода-вывода назначены специфические адреса, так что Вы должны почитать документацию для вашей платы параллельного порта. В этом примере, мы используем порт ввода-вывода, адрес которого `0x278`, который соответствует LPT2: порт принтера под MS-DOS.

Добавление IRQ и адреса порта в командной строке загрузчика дает следующую инструкцию (как будто, мы параметры появились в соответствующей секции файла `/etc/lilo.conf`:

```
append="lp=0x378.0 ppa=0x278.7"
```

Эти инструкции добавятся к параметрам запуска ядра при начальной загрузке. Они гарантируют, что любой принтер, подключенный к системе не сталкивается с Zip диском. Конечно, если ваша система не имеет принтера, директива ``lp="` может и должна быть опущена.

После того, как Вы установили заказное ядро как описано в разделе 4.2.1, прежде, чем Вы перезагрузите систему, убедитесь, что выполнили команду:

```
# /sbin/lilo
```

для установки в загрузочный сектор жесткого диска новых настроек LILO.

4.9.3 Установка модуля драйвера устройства.

Выше описано резервирование на стриммер. Linux поддерживает много типов стриммеров с интерфейсами IDE, SCSI и дисковода для гибких дискет. Linux поддерживает драйвер ftape как модуль.

На момент написания данного документа, последняя версия ftape была 3.04d. Вы можете получить дистрибутив с FTP архива `sunsite.unc.edu` (см. инструкции в приложении B). Архив ftape находится в `/pub/Linux/kernel/tapes`. Посмотрите, какая

версия там наиболее свежая. На момент написания данного документа, последняя версия ftape лежала в файле `ftape-3.04d.tar.gz`.

После распаковки архива ftape в каталог `/usr/src`, наберите `make install` в каталоге верхнего уровня дистрибутива ftape для компиляции модуля драйвера ftape и утилит. Если Вы испытываете проблемы совместимости с дистрибутивными файлами ftape вашего ядра системы или библиотеки, выполните команды `make clean` и `make install`, которые гарантируют, что модули компилируются для вашей системы.

Чтобы использовать эту версию ftape драйвера, Вы должны иметь поддержку модулей, компилируемую в ядро, и поддержку демона ядра `kerneld`. Однако, Вы *не должны* включать код ftape, как опцию ядра при его настройке, поскольку более современный ftape модуль полностью заменяет этот код.

`make install` установит драйвер устройства в правильный каталог. На стандартных Linux системах, модули лежат в каталоге `/lib/modules/kernel-version`. Если у Вас ядро версии 2.0.30, модули лежат в каталоге `/lib/modules/2.0.30`. Шаг установки `make install` также обеспечивает, чтобы эти модули было можно найти, добавляя соответствующие инструкции в файл `modules.dep`, размещенному в верхнем каталоге файлов модулей, в данном случае `/lib/modules/2.0.30`. Установка ftape добавляет следующие модули к вашей системе (в этом примере использована версия ядра `2.0.30`):

```
/lib/modules-2.0.30/misc/ftape.o
/lib/modules-2.0.30/misc/zft-compressor.o
/lib/modules-2.0.30/misc/zftape.o
```

Команды для загрузки модуля должны быть добавлены к системному файлу конфигурации модулей. Во многих системах он называется `/etc/conf.modules`. Чтобы автоматически загружать ftape модули по требованию, добавьте следующие строки к файлу `/etc/conf.modules`:

```
alias char-major-27 zftape
pre-install ftape /sbin/swapout 5
```

Первая инструкция загружает все связанные с ftape модули в случае необходимости, когда к устройству со старшим номером 27 (ftape устройство) обращается ядро. Поскольку поддержка модуля zftape (который обеспечивает автоматическое сжатие данных для стриммера) требует поддержки других ftape модулей, все они грузятся по требованию ядра. Вторая строка определяет параметры загрузки для модулей. В данном случае утилита `/sbin/swapout`, которая обеспечивается пакетом ftape, гарантирует, что для ftape драйвера доступно достаточное количество DMA-памяти.

Чтобы обращаться к ftape устройству, Вы должны сначала поместить отформатированную ленту в стриммер. Команды для форматирования лент и операций с ними даны в разделе 4.8.3.

4.9.4 Обновление библиотек.

Как упомянуто выше, многие программы откомпилированы для использования разделяемых библиотек, которые хранят общий программный код, используемый разными программами.

Если Вы получили сообщение:

```
Incompatible library version
```

при попытке выполнить программу, вам необходимо модернизировать версию ваших библиотек, которые использует программа. Библиотеки совместимы в обратном направлении, то есть программа, откомпилированная для использования с более ранней версией библиотек, должна работать с новой версией библиотек. А обратное не справедливо.

Самая последняя версия библиотек может быть найдена FTP-серверах Linux. На `sunsite.unc.edu` они расположены в `/pub/Linux/GCC`. Файлы "версии" (``release") должны описывать, какие файлы вам необходимо скачать, и как их инсталлировать. Кратко, вы должны иметь файлы `image-version.tar.gz` и `inc-version.tar.gz`, где версия указывает версию инсталлируемых библиотек, например 4.4.1. Это tar-файлы, сжатые `gzip`. Файлы образов содержат образы инсталлируемых библиотек в `/lib` и `/usr/lib`. Файл `inc` содержит include-файлы для инсталляции в `/usr/include`.

Файл `release-`*version*`.tar.gz` объясняет инсталляционную процедуру в деталях (конкретные инструкции для конкретных версий отличаются). В общем случае вы должны инсталлировать библиотечные `.a` и `.sa` файлы в `/usr/lib`. Эти библиотеки используются на этапе компиляции.

Дополнительно, разделяемая библиотека образов файлов `libc.so.`*version* инсталлируется в `/lib`. Это разделяемые библиотеки образов загружаются во время выполнения использующими их программами. Каждая библиотека имеет символическую связь, использующую старшее число версии библиотеки в `/lib`.

Например, библиотека `libc` версия 4.4.1 имеет старшую цифру версии 4. Файл, содержащий библиотеку - `libc.so.4.4.1`. Символическая связь с именем `libc.so.4`, указывающая на этот файл, также в `/lib`. Вы должны изменить эту символическую связь, когда модифицируете библиотеки. Например, когда идет смена версий, вы должны изменить символическую связь файла `libc.so.4` на новую версию.

Важное замечание! Надо менять символическую связь за один шаг, как показано ниже. Если вы каким-то образом удалили символическую связь `libc.so.4` тогда программы, которые зависят от этой связи (включая базовые утилиты вроде `ls` и `cat`) перестанут работать. Используйте следующую команду для обновления символической связи `libc.so.4`, чтобы она указывала на файл `libc.so.4.4.1`:

```
# In -sf /lib/libc.so.4.4.1 /lib/libc.so.4
```

Вы должны также изменить символическую связь `libm.so.`*version* таким же манером. Если вы переходите на отличную (от прежней) версию библиотек, замените имена

вышеупомянутых файлов. Пояснения к версии библиотеки должны прояснить детали. Дополнительную информацию про символические связи смотрите здесь.

4.9.5 Обновление gcc.

Компиляторы gcc C и C++ используются для компиляции программ вашей системы, в первую голову - ядра. Новейшую версию gcc можно найти на FTP-серверах Linux. На sunsite.unc.edu его можно найти в каталоге /pub/Linux/GCC (вместе с библиотеками). Должен существовать файл версии для дистрибуции gcc, детализирующий, какие файлы вы должны переписать и как их инсталлировать. Многие дистрибутивы позволяют обновлять версии с помощью своих средств управления пакетами программ. Вообще, такие пакеты куда удобней при установке, чем универсальные.

4.9.6 Модернизация других программ.

Модернизация других программ, это в основном проблема добычи соответствующих файлов и их инсталляции. Большинство программ для Linux распространяются как зазипованные tar-файлы, включая как исходные, так и выполняемые, или те и другие. Если выполняемые файлы не включены в версию, вам может потребоваться самостоятельно их откомпилировать. Обычно это означает запуск make в каталоге, где находятся исходники.

Чтение группы новостей USENET comp.os.linux.announce - простейший путь, чтобы выловить информацию о новых программах. Так что самый простой способ отыскать какие-то программы, это побродить по FTP-серверам, поскачивать с серверов (ls-lR) индексные файлы и, используя grep, найти желаемые файлы. Если вам доступен archie, это также может помочь. Детали смотрите в приложении A. Если у вас нет archie, вы можете по telnet выйти на сервер archie вроде archie.rutgers.edu, войти как ``archie'' и воспользоваться командой ``help''). Детали смотрите в приложении A.

4.10 Прочие задачи.

Хотите верьте, хотите нет, но существует ряд хозяйственных задач, входящих в функции системного администратора, которые не попадают ни в одну из основных категорий.

4.10.1 Системные файлы настройки.

При загрузке системы некоторые сценарии автоматически выполняются системой до входа в нее пользователей. Далее следует описание того, что в это безвременье происходит.

Во время загрузки ядро запускает процесс `/etc/init`. `init` это программа, которая читает свои настроечные файлы (`/etc/inittab`) и запускает другие процессы, базирующиеся на содержании этих файлов. Один из важных процессов, который запускается из `inittab`, это `/etc/getty`, он грузится для каждой виртуальной консоли. Процесс `getty` захватывает ВК (Виртуальную Консоль) и запускает на ней процесс `login`. Это позволяет вам входить на каждой ВК. Если `/etc/inittab` не содержит процессов `getty` для конкретной ВК, на эту ВК вы не войдете.

Другой процесс, выполняемый из `/etc/inittab`, это `/etc/rc`, главный системный файл инициализации Этот файл представляет из себя shell-сценарий, который выполняет все необходимые команды инициализации во время загрузки, такие например, как монтирование файловых систем и инициализации области свопинга. На многих системах `init` выполняет файл `/etc/init.d/rc`.

Ваша система может также выполнять и другие сценарии, например `/etc/rc.local`. `/etc/rc.local` обычно содержит команды инициализации, специфичные для вашей системы, такие как установка хост-имени (смотрите следующий раздел). `rc.local` может запускаться из `/etc/rc` или прямо из `/etc/inittab`.

4.10.2 Установка хост-имени.

В сетевой среде хост-имя используется для однозначной идентификации конкретной машины, в то время как отдельно стоящей машине хост-имя придает чувство собственного достоинства и шарма. Это, как дать имя вашей собаке: вы можете обращаться к собаке просто ``The dog'', значительно интереснее приписать собаке имя, вроде Spot или Woofie.

Хост-имя элементарно устанавливается командой `hostname`. Если вы в сети, ваше хост-имя должно быть полным хост-именем вашей машины, таким как `goober.norelco.com`. Если вы не в сети, вы можете выбрать произвольные имена для хоста и домена, например `loomer.vpizza.com`, `shoop.nowhere.edu` или `floof.org`.

При установке хост-имени оно должно быть занесено в файл `/etc/hosts`, который приписывает IP адрес каждому хосту. Даже если ваша машина не в сети, вам следует включить ваше хост-имя в `/etc/hosts`. Например, если вы не имеете выхода в сеть по TCP/IP и ваше хост-имя `floof.org`, просто включите следующую запись в `/etc/hosts`:

```
127.0.0.1       floof.org localhost
```

Данная команда припишет ваше хост-имя `floof.org` к локальному IP-интерфейсу (loopback address) 127.0.0.1 (используемому, даже если вы не в сети). Синоним `localhost` также приписывается этому адресу.

Если вы подключены к сети по TCP/IP, ваши действительные IP адрес и хост-имя должны появиться в `/etc/hosts`. Например, если ваше хост-имя `goober.norelco.com`, и ваш IP адрес 128.253.154.32, добавьте следующую строку в `/etc/hosts`:

```
128.253.154.32  goober.noreico.com
```

Если вашего хост-имени не будет в `/etc/hosts`, вы не сможете его установить. Для установки хост-имени просто используйте команду `hostname`. Например, команда:

```
# hostname -S goober.noreico.com
```

устанавливает хост-имя `goober.norelco.com`. Во многих случаях команда `hostname` выполняется из одного из системных установочных файлов, таких как `/etc/rc` или `/etc/rc.local`. Отредактируйте эти два файла и измените находящуюся там команду `hostname`, установив хост-имя своей машины; после перезагрузки машины хост-имя будет иметь новое значение.

4.11 Что делать при ЧП.

В некоторых случаях администратор системы будет сталкиваться с проблемой выкарабкивания из абсолютной катастрофы, такой например, как забытие пароля `root` или крах файловой системы. Лучший совет: *без паники*! Все делают глупые ошибки: это лучший способ освоить системное администрирование, хотя и тяжелый.

Linux не является нестабильной версией UNIX. Действительно, у меня было значительно меньше проблем с зависанием системы, чем с коммерческими версиями UNIX на многих платформах. Linux также выигрывает от большого расположения к нему крутых программистов, которые могут помочь выпутаться из сложной ситуации.

Первый шаг в исследовании любой проблемы: попытаться справиться с ней самостоятельно. Потыкайтесь туда-сюда и посмотрите, что из этого будет получаться. Слишком много времени системные администраторы тратят на рассылку во все стороны отчаянных воплей о помощи, прежде, чем вникнуть в проблему. В большем числе случаев вы обнаружите, что вы сами легко можете решить проблему. А это уже ваш прямой путь в мэтры.

Очень редки случаи, когда после краха системы требуется переинсталляция: это вам не винды... Многие начинающие пользователи случайно удаляют некоторые важные системные файлы и немедленно бегут за инсталляционным диском. Прежде чем применять такие отвратительные меры, исследуйте проблему и попросите других помочь ее решить. В большинстве случаев вы можете восстановить систему с дискеты сопровождения (maintenance diskette).

4.11.1 Восстановление с использованием дискеты сопровождения.

Одно незаменимое средство для администратора системы, это так называемый ``boot/root disk'', дискета, которая может загрузить полный Linux, вне зависимости от вашего жесткого диска. Boot/root disks в действительности очень прост: вы создаете корневую файловую систему на дискете, помещая на нее все необходимые утилиты, инсталлируя на дискете LILO и загружаемое ядро. Другой способ, это использовать одну дискету для ядра и другую для корневой файловой системы. В любом случае результат одинаков: Вы запускаете Linux полностью с дискет.

Канонический пример boot/root disk это загрузочный диск Slackware. Для этого вам не надо скачивать дистрибутив полностью: хватит только `boot` и `root` дискеты). Эти дискеты содержат загрузочную таблицу и корневую файловую систему. Предполагается, что они используются при инсталляции дистрибутивов Slackware, но они бывают очень полезны и для сопровождения системы.

Boot/root disk, созданный H.J Lu, который можно взять в `/pub/Linux/GCC/rootdisk` на `sunsite.unc.edu` другой пример такого рода диска сопровождения.

Использовать boot/root disk очень легко. Просто загрузите диск на вашей системе и войдите под `root` (обычно без пароля). Чтобы получить доступ к файлам вашего жесткого диска, необходимо смонтировать ваши файловые системы вручную. Например, команда:

```
# mount -t ext2 /dev/hda2 /mnt
```

смонтирует файловую систему ext2fs на `/dev/hda2` под `/mnt`. Помните, что `/` теперь находится на boot/root disk; вам необходимо смонтировать файловую систему вашего жесткого диска под каким-то каталогом, чтобы получить доступ к файлам. Так что `/etc/passwd` вашего жесткого диска теперь `/mnt/etc/passwd`, если вы смонтировали вашу корневую файловую систему на `/mnt`.

4.11.2 Восстановление пароля для root.

Если вы забыли пароль вашего `root`, нет проблем. Просто загрузитесь с boot/root disk, смонтируйте вашу корневую файловую систему под `/mnt` и сотрите поле пароля для `root` в `/mnt/etc/passwd`, как например:

```
root::0:0:root:/:/bin/sh
```

теперь `root` остался вообще без пароля; когда вы перезагрузитесь с жесткого диска, вы сможете войти как `root` и снова установить пароль, используя команду `passwd`.

Не правда ли, вы счастливы, что научились работать с `vi`? На вашей boot/root disk, редакторов, вроде Emacs наверняка нет, а `vi` должен быть, (Прим. переводчика: администратор должен отдавать себе отчет, что процедуру снятия пароля root умеет запросто выполнять не он один).

4.11.3 Восстановление файловой системы.

Если у вас каким-то образом грохнулась файловая система, вы можете использовать `e2fsck` (это в случае, если вы используете файловую систему типа ext2fs) для исправления запорченных данных файловой системы с дискет. Другие файловые системы используют другие формы команды `fsck`; детали смотрите здесь.

Когда вы проверяете вашу файловую систему с дискеты, лучше всего, чтобы файловая система не была смонтирована.

Частая причина неисправности файловой системы: порча суперблока. Суперблок, это заголовок (``header')'файловой системы, который содержит информацию о статусе файловой системы, размере, свободных блоках и т.д. Если вы попортили ваш суперблок (например, случайно прямо в него записали какие-то данные) операционная система может вообще не распознать файловую систему. Все попытки примонтировать файловую систему потерпят неудачу, и `e2fsck` не поможет решить проблему.

К счастью, файловая система типа ext2fs сохраняет копии суперблока в границах "группы блоков" (``block group") на диске, обычно через каждые 8 КБ блоков. Для того, чтобы приказать `e2fsck` использовать копию суперблока, вы можете использовать команду:

```
# e2fsck -b 8193 partition
```

здесь *partition*, это раздел, на котором располагается файловая система. Опция `-b 8193` приказывает `e2fsck` использовать копию суперблока, хранящуюся в блоке 8193 файловой системы.

4.11.4 Восстановление потерянных файлов.

Если вы случайно удалили важные файлы, нет способа их "разудалить" обратно. Однако, вы можете скопировать соответствующие файлы с дискеты к себе на жесткий диск. Например, если вы удалите `/bin/login` в своей системе (который обеспечивает вход в систему), просто загрузите boot/root дискету, смонтируйте корневую файловую систему на `/mnt` и используйте команду:

```
# cp -a /bin/login /mnt/bin/login
```

опция `-a` приказывает `cp` сохранить права доступа копируемых файлов.

Разумеется, если удаленные файлы не столь существенны, что они не были удостоены копирования на дискету boot/root, значит вам не повезло. Если вы создавали резервные копии, вы можете скопировать файлы оттуда.

4.11.5 Восстановление потерянных библиотек.

Если вы случайно потеряли свои библиотеки или символические связи в `/lib`, скорее всего команды, которые зависят от этих библиотек, больше не будут выполняться (подробности). Простейшее решение: загрузиться с дискеты boot/root, смонтировать вашу корневую файловую систему и восстановить библиотеки в `/mnt/lib`. Здесь рассказано, как ставить библиотеки и создавать их символические ссылки.

5 X Window System

X Window System является большой, мощной и сложной графической средой для UNIX систем. Система X-Window была разработана в Массачуссетстком технологическом институте (MIT), которая стала затем стандартом для всех UNIX систем. Практически каждая рабочая станция UNIX в мире работает на одном из вариантов X-Window.

Группа программистов, возглавляемая Дэвидом Вексельблатом (David Wexelblat), (Вы можете связаться с Девидом по E-Mail: dwex@XFree86.org) произвела перенос MIT X Window System версия 11, релиз 6 (X11R6) для 80386/80486/Pentium UNIX систем как свободно распространяемого программного продукта. Эта версия, известная как XFree86 TM, (XFree86 является торговой маркой XFree86 Project, Inc.) доступна для System V/386, 386BSD и других реализаций UNIX для процессоров x86, включая Linux. Она включает в себя все требуемые выполняемые коды, конфигурационные файлы, библиотеки и инструментарий.

Некоторые интересные возможности данной версии:

- полная поддержка версии X11R6.3 X Consortium
- новое расширение DPMS от Digital Equipment Corporation;
- расширение Low Bandwidth X (LBX) для всех X серверов;
- поддержка Microsoft IntelliMouse;
- поддержка сжатия шрифтов с помощью `gzip` font.

Полная настройка и использование X Window выходит за пределы этой книги. Вам следует обратиться к книге *The X Window System: A User's Guide* (см. приложение А). В этой главе мы опишем шаг за шагом установку и настройку XFree86 для Linux. Для более детального ознакомления вы можете обратиться к документации, поставляемой вместе с XFree86 (она обсуждается ниже). Другим полезным источником информации является *THE LINUX XFree86 HOWTO*.

5.1 X Window: требования к аппаратуре.

5.1.1 Видеокарта и монитор.

Документация, поставляемая вместе с видеоадаптером, как правило указывает тип используемых микросхем. Если вы приобрели новую видеокарту или новый компьютер, попросите поставщика уточнить изготовителя, модель и тип микросхем видеокарты. Как правило поставщики охотно дадут вам эту информацию. Большинство из них сообщит, что видеокарта является стандартной SVGA картой и будет работать в вашей операционной системе. Объясните им, что ваше программное обеспечение (Linux и XFree86!) не поддерживает всех видеокарт и вам требуется дополнительная информация.

Вы можете также определить тип микросхемы, вызвав команду `SuperProbe`, входящую в состав XFree86. Это будет описано ниже.

Как правило, все видеокарты поддерживаются как в режиме 256 цветов, так и в монохромном режиме. Если на вашей видеокарте установлено достаточно видеопамяти, многие из микросхем поддерживаются в режиме 15, 16, 24 и 32 бита на точку. Обычно видеокарты используются в режиме 8 бит на точку (256 цветов).

Монохромный сервер поддерживает основные карты VGA, монохромные карты Hercules, Hyundai HGC1280, Sigma LaserView и Apollo monochrome cards.

Этот список несомненно расширится со временем. Полный список поддерживаемых карт вы найдете в замечаниях к текущей версии XFree86.

Одной из проблем, с которой столкнулись разработчики, являлся нестандартный механизм определения частоты, используемый для управления картой. Некоторые производители либо не описывали способ программирования карты, либо требовали подписания дополнительного соглашения о нераспространении полученной информации. Это очевидно ограничило бы свободное распространение XFree86, чего естественно не могли допустить разработчики. Долгое время данная проблема была с видеокартами, производимыми фирмой Diamond, но начиная с версии 3.3 XFree86, Diamond начала сотрудничество с разработчиками с целью выпуска драйвера для этих карт.

Также поддерживаются карты с ускорителями на чипсетах S3. Посмотрите документацию на XFree86 на предмет поддержки вашей карты ДО опытов с оборудованием. Сравнение производительности карт регулярно публикуется в Usenet news groups `comp.windows.x.i386unix` и `comp.os.linux.misc`.

Замечу, что мой персональный компьютер с Linux содержит 486DX2-66, 20 мегабайт RAM, и имеет видеоадаптер VLB S3-864 с 2 мегабайтами оперативной памяти. Я протестировал X benchmarks на этой машине и на рабочей станции Sun Sparc IPX. Linux где-то раз в 7 быстрее, чем Sparc IPX. Обычно, XFree86 под Linux с графическим ускорителем показывает существенно большую производительность чем коммерческие рабочие станции (которые обычно используют неэффективные алгоритмы обработки графической информации).

5.1.2 Память, CPU и место на диске.

Лучше всего ставить XFree86 под Linux на 80486 или более быстрой машине с не менее, чем 16 МБ RAM. Имейте в виду, что чем больше физической оперативной памяти вы имеете, тем меньше операционная система использует свопинг. Так как

операция свопинга медленная (доступ к диску намного медленнее, чем к памяти), для комфортабельной работы вам следует иметь 16 или более мегабайт. Система с 4-мя мегабайтами работает намного (в десятки раз) медленнее чем с 16-ю мегабайтами.

Стандартная установка XFree86 требует 60-80 МБ на диске. Сюда входят X сервер(ы), шрифты, библиотеки и стандартные утилиты. Для нормальной работы с приложениями надо около 200 МБ.

5.2 Установка XFree86.

Дистрибутив Xfree86 в выполняемых кодах можно найти на целом ряде FTP-серверов. На `sunsite.unc.edu` он находится в каталоге `/pub/X11/XFree86`. (На момент написания текущая версия была 3.3.1; периодически появляются новые версии).

Вполне возможно, что вы имеете XFree86 как часть дистрибутива Linux, в этом случае в перекачке XFree86 нет необходимости.

Вам потребуется один из серверов:

```
Файл           Описание

X33S514.tgz    Server for 8514 based boards.
X33AGX.tgz     Server for AGX based boards.
X33li2S.tgz    Server for the Imagine 1128 boards.
X33Ma64.tgz    Server for Mach64 basedboards.
X33Ma32.tgz    Server for Mach32 basedboards.
X33Ma8.tgz     Server for Mach8 basedboards.
X33Mono.tgz    Server for monochrome video modes.
X33P9K.tgz     Server for P9000 basedboards.
X33S3.tgz      Server for S3 basedboards.
X33S3VI.tgz    Server for S3/V3rge basedboards.
X33SVGA.tgz    Server for Super VGA based boards.
X33SVGAi6.tgz  Server for VGA/EGA basedboards.
X33W32.tgz     Server for ET4000/W32 based boards.
```

Все нижеперечисленные файлы также нужны:

```
Файл           Описание

preinst.sh     Pre-installation script
postinst.sh    Post-installation script
x33bin.tgz     Clients, run-time libs, and app-defaults files
x33doc.tgz     Docunientation
x33fnts.tgz    75dpi, misc and PEX fonts
x33lib.tgz     Data files required at run-time
x33man.tgz     Manual pages
x33setup.tgz   XF86 Setup utility
x33vg16.tgz    16 colour VGA server (XF86 Setup needs this server)
```

Следующие файлы не являются обязательными для существующих инсталляций, но нужны для новых:

```
Файл          Описание

X33cfg.tgz    sample config files for xinit, xdm
```

Не ставьте `X33cfg.tgz` поверх существующих инсталляций XFree86, не зарезервировав файлы настроек! Распаковка `X33cfg.tgz` перезапишет их. Если Вы используете какие-то специальные файлы настроек, ставить данный пакет иногда и вовсе не надо.

Шрифты в версии 3.3.1 сжаты программой `gzip`. Вероятно, Вы захотите удалить старые шрифты. Но сначала зарезервируйте их! X сервера и сервера шрифтов из прошлых версий не могут читать шрифты, сжатые `gzip`, так что сохраните старые шрифты, если планируется работа и со старыми серверами.

Следующие файлы не являются обязательными:

```
Файл           Описание

X33fl00.tgz    100dpi fonts
X33fcyr.tgz    Cyrillic fonts
X33fnon.tgz    Other fonts (Chinese, Japanese, Korean, Hebrew)
X33fscl.tgz    Scalable fonts (Speedo and Type1)
X33fsrv.tgz    Font server and config files
X33prog.tgz    X header files, config files and compile-time libs
X33nest.tgz    Nested X server
X33vfb.tgz     Virtual framebuffer X server
X33prt.tgz     X Print server
X33ps.tgz      PostScript version of the documentation
X33htinl.tgz   HTML version of the documentation
X33jdoc.tgz    Do cunientation in Japanese (for version 3.2)
X33jhtm.tgz    HTML version of the do cunientation in Japanese (3.2)
X331kit.tgz    X server LinkKit
```

Каталог XFree86 должен содержать файлы `README` и замечания по инсталляции текущей версии.

Все что вам требуется для инсталляции XFree86, это получить указанные файлы, создать каталог `/usr/X11R6` (пользователем `root`), перейти в этот каталог и распаковать файлы. Затем запустите скрипт `preinst.sh`. Вы должны скопировать его и все архивные файлы дистрибутива в каталог `/var/tmp` перед запуском `preinst.sh`. `/usr/X11R6` должен быть текущим каталогом при распаковке архивов и запуске `preinst.sh`:

```
# cd /usr/X11R6
# sh /var/tmp/preinst.sh
```

Распакуйте файлы из `/var/tmp` в `/usr/X11R6` командой:

```
# gzip -d /var/tmp/X33prog.tgz|tar vxf
```

Имейте в виду, что эти файлы упакованы относительно каталога /usr/X11R6, так что необходимо распаковывать их находясь в этом каталоге.

После распаковки всех архивов, запустите скрипт postinst.sh:

```
# cd /usr/X11R6
# sh /var/tmp/postinst.sh
```

Теперь необходимо связать файл /usr/X11R6/bin/X с тем сервером, который вы намереваетесь использовать. Например, если вы желаете работать с SVGA сервером, файл /usr/bin/X11/X необходимо связать с файлом /usr/X11R6/bin/XF86_SVGA. Если же вы собираетесь использовать монохромный сервер, переустановите связь X с XF86_MONO командой:

```
# In -sf /usr/XllR6/bin/XF86-MONO /usr/X11R6/bin/X
```

Это же справедливо и для серверов других видеокарт.

Вам следует убедиться, что каталог /usr/X11R6/bin находится в переменной среды PATH. Это может быть сделано редактированием файлов /etc/profile или /etc/csh.login (в зависимости от оболочек (shell) которые вы или другие пользователи используют). Вы также можете просто добавить этот каталог с вашей переменной PATH, корректируя в вашем домашнем каталоге файлы .bashrc или .cshrc, в зависимости от типа вашей оболочки.

Вам также необходимо обеспечить загрузку динамических библиотек. Для этого добавьте строку:

```
/usr/XllR6/lib
```

в файл /etc/ld.so.conf и выполните /sbin/ldconfig как root.

5.3 Исследование аппаратной конфигурации.

Если вы не уверены какой сервер использовать, или не знаете какую микросхему содержит ваша видеокарта, вы можете запустить команду SuperProbe (включенную в /usr/X11R6/bin). Эта программа попытается определить тип микросхемы вашей видеокарты и другую полезную информацию.

Для запуска SuperProbe из командной строки наберите:

```
# SuperProbe
```

Возможно, что SuperProbe запутается в устройствах, которые используют адреса I/O портов, которые могли бы использоваться платами видео. Чтобы такого не было, используйте параметр excl со списком адресов, которые SuperProbe не должен исследовать. Например:

```
# SuperProbe -excl 0x200-0x230.0x240
```

Адреса даны как шестнадцатеричные числа, которые имеют префикс 0x.

Чтобы отобразить список видеокарт, известных SuperProbe, наберите:

```
# SuperProbe -info
```

SuperProbe может выдать много информации при запуске с параметром -verbose. Вы можете перенаправить вывод в файл:

```
# SuperProbe -verbose >superprobe.out
```

Запуск SuperProbe может подвесить систему. Убедитесь, что любые несущественные прикладные программы не запущены, или по крайней мере безопасно сохранили все их данные на диск, и позаботьтесь, чтобы любые пользователи вышли из системы. Также, загруженные системы (которые печатают в фоне, например), могут искажать вывод программного обеспечения подобного SuperProbe или X сервера, который пробует измерять спецификации синхронизации платы видео.

5.4 Автопостроение файла XF86Config.

Создание файла XF86Config вручную трудная задача, но это не невозможно. Несколько инструментальных средств в XFree86 версии 3.3.1 могут помочь Вам. Одно из них, программа XF86Setup, может автоматически генерировать файл XF86Config в правильном формате. Вы должны знать точные спецификации о вашей плате видео, вертикальных и горизонтальных значениях регенерации вашего монитора. Большинство информации может быть найдено в руководствах.

Несколько других программ конфигурации также доступны, в зависимости от дистрибутива Linux. Наиболее общие: Xconfigurator и xf86config. Последняя программа это старшая версия XF86Setup и она включена в старые выпуски XFree86. Вы должны всегда использовать XF86Setup если есть и XF86Setup, и xf86config.

5.5 Настройка XFree86.

В большинстве случаев установка XFree86 не представляет проблем. Однако, если вы желаете использовать видеокарту для которой драйвер находится в процессе

разработки или добиться лучших разрешения или производительности от карты с графическим акселератором, то вам потребуется определенное время для настройки XFree86.

В этой главе мы опишем как создать и отредактировать XF86Config файл, который настраивает сервер XFree86.

В большинстве случаев лучше всего начать с основной конфигурации XFree86, которая использует низкое разрешение, например 640x480, поддерживаемого всеми видеокартами и мониторами. Однажды настроив XFree86 на стандартное разрешение, вы можете затем подстроить файл конфигурации для для того, чтобы использовать все возможности, предоставляемые вашей аппаратурой.

В дополнение к информации приведенной здесь, вам следует ознакомиться со следующей документацией:

- Документация на XFree86 в /usr/X11R6/lib/X11/doc (из пакета XFree86-3.1-doc). Посмотрите файл README.Config, который является руководством по настройке XFree86.
- Разные видео чипсеты имеют разные README файлы в разных каталогах (например, README.Cirrus и README.S3). Почитайте файл о Вашей видеокарте.
- man-руководство по XFree86.
- man-руководство по XF86Config.
- man-руководство по используемому серверу, например XF86_SVGA или XF86_S3.

Основной файл настройки XFree86 /usr/X11R6/lib/X11/XF86Config. Он хранит данные о мышке, параметрах видеокарты, мониторе и тому подобном. Файл XF86Config.eg входит в дистрибутив XFree86 в качестве примера. Скопируйте его в XF86Config и отредактируйте.

В man-руководстве по XF86Config описывается и формат файла XF86Config. Если Вы еще не прочитали руководство, сделайте это.

Далее мы собираемся просмотреть файл XF86Config участок за участком. Этот файл может выглядеть не совсем так, как файл в вашем дистрибутиве XFree86, но структура их совпадает.

Учтите, что формат XF86Config может меняться в разных версиях XFree86. Ознакомьтесь с документацией к дистрибутиву.

Имейте также в виду, что не следует просто копировать конфигурационный файл, приведенный здесь и пытаться использовать его. Попытка использовать конфигурационный файл, не соответствующий вашему оборудованию, может заставить ваш монитор работать со слишком высокой для него частотой; **были сообщения о выходе из строя мониторов (особенно мониторов с фиксированной частотой) при использовании неверных XF86Config файлов.**

Каждая секция файла XF86Config определяется парой строк Section "<section-name>" ... EndSection.

Первая секция файла называется `Files`, и выглядит следующим образом:

```
Section "Files"
RgbPath "/usr/X11R6/lib/X11/rgb"
FontPath "/usr/X11R6/lib/X11/fonts/misc"
FontPath "/usr/XiiR6/lib/X11/fonts/75dpi"
End-Section
```

Строка `RgbPath` указывает местоположение базы данных цветов, а каждая строка `FontPath` определяет каталог, содержащий шрифты. Как правило, вам не следует изменять эти строки, вам следует только убедиться, что все каталоги шрифтов присутствуют. Обычно они лежат в `/usr/X11R6/lib/X11/fonts`.

Следующая секция имеет имя `ServerFlags` и определяет несколько глобальных параметров для сервера. Как правило эта секция пуста:

```
Section "ServerFlags"
# Uncomment this to cause a core dump at the spot where a signal is
# received. This may leave the console in an unusable state, but may
# provide a better stack trace in the core dump to aid in debugging
# NoTrapsignals

# Uncomment this to disable the Crtl-Alt-BS server abort sequence
# DontSap
EndSection
```

Все строки данной секции закомментированы.

Следующая секция `Keyboard`. Она определяет работу клавиатуры. Доступны также и другие режимы, кроме приведенных в данном примере. Описанные здесь опции работают на большинстве клавиатур:

```
Section "Keyboard"

Protocol "Standard"
AutoRepeat 500 5
Server NumLock

End-Section
```

Следующая секция, `Pointer`, определяет параметры мыши:

```
Section "Pointer"

Protocol    "MouseSystems"
Device      "/dev/mouse"

# Baudrate and SampleRate are only for some Logitech mice
#    BaudRate   9500
#    SampleRate 150

# Emulate 3 Buttons is an option for 2-button Microsoft mice
#    Emulate 3 Buttons
```

```
# ChordMiddLe is an option for some 3-button Logitech mice
#    ChordMiddLe

EndSection
```

Единственными опциями, на которые стоит обратить внимание являются `Protocol` и `Device`. `Protocol` определяет *протокол*, который использует ваша мышь. Возможными типами (для Linux есть другие опции, доступными для других ОС) являются:

- `BusMouse`
- `Logitech`
- `Microsoft`
- `MMSeries`
- `Mouseman`
- `MouseSystems`
- `PS/2`
- `MMHitTab`

Для Logitech busmouse следует использовать протокол `BusMouse`. Заметьте, что для старых мышей Logitech необходимо использовать протокол `Logitech`, а для новых или `Microsoft`, или `Mouseman` протокол.

Строка `Device` определяет устройство к которому подключена мышь. На большинстве систем Linux это `/dev/mouse`. `/dev/mouse` обычно связано с соответствующим серийным портом (например `/dev/cua0` соответствует COM1, а `/dev/cua1` COM2) или с портом `busmouse`. В любом случае убедитесь, что указанное устройство есть в каталоге `/dev` и работоспособно.

Следующая секция, `Monitor`, определяет характеристики вашего монитора. Файл `XF86Config` может содержать не одну, а несколько секций `Monitor` (это справедливо и для других секций). Это полезно в том случае, когда вы подключили к системе несколько мониторов или используете один и тот же `XF86Config` файл для различных конфигураций:

```
Section "Monitor"
Identifier "CTX 545S III"

# These values are for a CTX 545SIII only! Don't attempt to use
# them with your monitor (unless you have this model?
Bandwidth 50
HorizSync 30-38,47-50
VertRefresh 50-90

# Modes  Name      dot clock hori - vert
ModeLine "640x480"  25  540   554 750 800 480 491 493 525
ModeLine "800x600"  35  800   824 895 1024 500 501 503 525
ModeLine "1024x768" 55 1024  1088 1200 1328 758 783 789 818

EndSection
```

Строка `Identifier` используется для именования текущего описания монитора. Это может быть любая строка, на которую вы в дальнейшем ссылаться в файле `XF86Config`.

HorizSync определяет возможные скорости горизонтальной развертки для вашего монитора в килогерцах. Если у вас многочастотный (multisync) монитор, вы можете указать интервал значений (или несколько интервалов, разделенных запятой). Если у вас монитор с фиксированной частотой, то вам надо указать список фиксированных значений. Например:

```
HorizSync    31.5, 35.2, 37.9, 35.5, 48.95
```

В руководстве на ваш монитор эти значения должны быть описаны. Если вы не имеете этой информации, вам следует связаться с производителем или продавцом вашего монитора.

Строка VertRefresh описывает возможные значения частоты вертикальной развертки для вашего монитора в герцах. Как и для HorizSync вы можете указать интервал или список дискретных значений. Ваше руководство на монитор должно содержать эту информацию.

Сервер использует значения HorizSync и VertRefresh ТОЛЬКО для того, чтобы убедиться что вы верно определили разрешение монитора. Это исключает возможность разрушения монитора при попытке работы с ним на частоте превышающей максимально допустимую.

Строка ModeLine определяет один из режимов разрушения вашего монитора. Ее формат:

```
ModeLine name clock horiz-values vert-values
```

name строка, которую вы можете использовать в этом файле в дальнейшем для указания разрешения. *dot-clock* определяет частоту задающего генератора адаптера для этого разрешения. Обычно частота указывается в мегагерцах. Она определяет скорость с которой видеокарта должна посылать значения точек экрана на монитор при указанном разрешении. *horiz-values* и *vert-values* состоят из 4-х цифр каждая. Эти значения определяют, когда электронная пушка монитора во время развертки должна включиться и когда должны проходить импульсы горизонтальной и вертикальной синхронизации во время развертки луча.

Как описать строку ModeLine для вашего монитора? Файл VideoModes.doc, включенный в дистрибутив XFree86, детально описывает как определить эти значения для каждого разрешения, которое поддерживает ваш монитор. Значение clock должно соответствовать частотам, которые поддерживает ваша видеокарта. Далее в файле XF86Config вы определите эти значения.

Существует два файла modeDB.txt и Monitors в дистрибутиве XFree, которые могут содержать данные ModeLine для вашего монитора. Эти файлы располагаются в каталоге /usr/X11R6/lib/X11/doc.

Вы можете начать со значений ModeLine для мониторов стандарта VESA. Этот режим поддерживается большинством мониторов. Файл modeDB.txt включает описания синхронизации для стандартного разрешения VESA. В этом файле вы найдете такие строки:

```
# 640X48O@60Hz Non-Interlaced mode
# Horizontal Sync = 31 5kHz
# Timing- H=(0.95us, 3.81us, 1.59us). V=(0.35ms, 0.064ms, 1.02ms)
#
# name        clock   horizontal timing       vertical timing flags
"640X480"     25.175  640        664 760 800  480      491 493 525
```

Это стандартная строка синхронизации для разрешения 640x480 точек. Она устанавливает частоту 25.175 Mhz, которая должна поддерживаться большинством мониторов (более подробно об этом позже). В вашем файле эта строка должна выглядеть так:

```
ModeLine "640X480" 25.175 640 664 760 800 4SO 491 493 525
```

Заметим, что аргумент *name* в строке `ModeLine` (в нашем случае "640x480") может быть любой строкой, которая описывает для вас разрешение монитора.

Для каждой строки `ModeLine` сервер проверяет, попадают ли указанные значения в интервал указанных значений `Bandwidth`, `HorizSync` и `VertRefresh`. Если нет, то сервер выдаст предупреждение при начале работы.

Если стандартные значения синхронизации VESA не работают у вас, то просмотрите другие значения в файлах `modeDB.txt` и `Monitors` для других типов мониторов. Заметим, что многие 14 и 15 дюймовые мониторы не могут поддерживать разрешений 1024x768 точек из-за низкого значения `Bandwidth`. То есть, если вы не нашли описание режима высокого разрешения для вашего монитора, то не исключено, что ваш монитор вообще не поддерживает такое разрешение.

Если вы потерпели неудачу с подбором строки ModeLine, изучите инструкцию описанную в файле `VideoModes.doc` вашего дистрибутива. По этой инструкции вы сможете описать строку `ModeLine` по спецификациям, описанным в вашем руководстве на монитор. Здесь также может помочь файл `VideoModes.doc`, в котором также описана директива `ModeLine`.

В конце концов, если вы не можете подобрать правильные значения строки `ModeLine`, то вы можете просто слегка изменить эти значения для достижения требуемого результата. Например, если изображение на экране слегка уходит влево или вверх, вы можете по инструкции из файла `VideoModes.doc` настроить значения синхронизации. Проверьте также наличие управляющих клавиш на вашем мониторе! Частенько бывает достаточным изменить горизонтальный и вертикальный размер изображения во время работы XFree86 чтобы добиться желаемой центровки и размера изображения. Наличие этих клавиш на монитора значительно упрощает жизнь.

Не используйте значения синхронизации монитора или значения `ModeLine` для мониторов не вашей модели. Если Вы попробуете управлять монитором на частоте, для которой он не предназначен, Вы можете повредить или даже уничтожить его!

Следующая секция `Device` описывает параметры вашей видеокарты. Например:

```
Section "Device"

Identifier "#9 GXE 64"
# nothing yet; we fill in these values later

EndSection
```

Эта секция описывает возможности вашей карты. `Identifier` определяет имя этого описания для ссылки на него в дальнейшем.

Первоначально вам не стоит заполнять эту секцию, за исключением поля `Identifier`. X-сервер можно использовать в режиме определения параметров установленной видеокарты. После определения этих параметров вы занесете их в эту секцию. X-сервер способен определить тип микросхемы видеокарты, поддерживаемый интервал частот, наличие RAMDAC и размер установленной памяти на видеоадаптере. Подробности в разделе 5.6.

Прежде чем мы это сделаем, нам следует закончить описание файла `XF86Config`. Следующая секция, `Screen`, описывает возможные режимы работы X-сервера с видеокартой и монитором:

```
Section "Screen"

Driver "Accel"
Device "#9 GXE 64"
Monitor "CTX 546S III"

Subsection "Display"
Depth 16
Modes "1024X768" "800X600" "640X480"
Viewport   0 0
Virtual    1024 768
EndSubsection

EndSection
```

Строка `Driver` определяет тип сервера, который вы будете использовать. Вы можете использовать следующие сервера:

- `Accel`: для `XF86_S3`, `XF86_Mach32`, `XF86_Mach8`, `XF86_8514`, `XF86_P9000`, `XF86_AGX` и `XF86_W32`.
- `SVGA`: для `XF86_SVGA`.
- `VGA16`: для `XF86_VGA16`.
- `VGA2`: для `XF86_Mono`.
- `Mono`: для монохромных не-VGA `XF86_Mono` и `XF86_VGA16`.

Убедитесь, что файл `/usr/X11R6/bin/X` является символьной ссылкой на используемый вами сервер.

Строка Device определяет идентификатор секции `Device`, описывающей установленную видеокарту. Выше мы описали секцию `Device` с идентификатором:

```
Identifier "# GXE 64"
```

Следовательно, здесь мы используем "`#9 GXE 64`" в строке `Device`.

Точно так же строка `Monitor` определяет имя секции `Monitor` для данного сервера, в данном примере "`CTX 5468 NI`".

Подсекция `Display` определяет режим работы сервера при выводе информации на экран. Файл `XF86Config` детально описывает эти режимы. Режимы, которые вам необходимо знать:

- `Depth`. Определяет число битов на точку. Обычно `Depth` принимает значение 8 (256 цветов). Для сервера `VGA16` вам следует установить значение `Depth` 4 и для монохромного сервера 1. Если вы используете видеокарту с ускорителем и имеете достаточно памяти для поддержки большего числа бит на точку, Вы можете установить `Depth` 15, 16, 24 или 32. Если с этими значениями появились проблемы вернитесь к значению 8 и попытайтесь решить проблему позже.
- `Modes`.Указывает список видеорежимов, описанных в секции `ModeLines`. Выше мы описали режимы `Modelines` названные "`1024x768`", "`800x600`" и "`640x480`". Следовательно строка `Modes` будет выглядеть:
-
-
-
  ```
  Modes    "1024X768" "800X600" "640X480"
  ```
-
-

  Первый режим, перечисленный в этой строке устанавливается по умолчанию после начала работы сервера. Далее вы можете переключать режимы разрешения изображения, использую клавиши ctrl-alt-numeric + и ctrl-alt-numeric -.

  Лучше всего при начальном конфигурации XFree86 использовать минимальное разрешение, например 640x480, которое работает на большинстве систем. И после настройки этого режима настроить `XF86Config` на работу с большими разрешениями.

- `Virtual`. Устанавливает виртуальный размер экрана. XFree86 имеет возможность использовать дополнительную память на вашей видеокарте для расширения вашего рабочего поля. Когда указатель мыши доходит до края экрана, ваше рабочее поле сдвигается показывая новые части вашего рабочего поля. Следовательно, даже если вы работаете на мониторе с низким разрешением (например 800x600 точек), вы можете установить размер виртуального экрана насколько вам позволяет память видеоплаты (1 Мегабайтная плата может хранить рабочее поле 1024x768 с 256 цветами, 2-х Мегабайтная плата - 1280x1024 с 256 цветами или 1024x768 с 16384 цветами и т д). Конечно, вы не сможете увидеть сразу все поле на вашем мониторе, но вы можете легко просмотреть любую его часть.

  `Virtual` предоставляет вам прекрасную возможность использовать всю память вашего адаптера, но она довольно ограничена. Если вы желаете еще расширить возможности работы с экраном, вам следует использовать `fvwm`, `openwin` или другой подобный менеджер окон. `fvwm` и `openwin` позволяет вам иметь намного

больший виртуальный экран (используя механизм спрятанных окон, вместо сохранения всего экрана в видеопамяти). Ваше виртуальное рабочее поле может состоять из 16x16 реальных экранов и более. Обратитесь к руководству по указанным командам. Большинство дистрибутивов XFree86 используют по умолчанию менеджер окон `fvwm`.

- `ViewPort`. Если вы использовали опцию `Virtual`, описанную выше, `ViewPort` устанавливает координаты левого верхнего угла виртуального экрана после начала работы сервера. Часто используют значение `Virtual 0 0`. Если вы не установили этого значения сервер центрирует виртуальный экран на мониторе (что может быть не всегда желательно).

Существуют и другие опции для данной секции (см. руководство для файла `XF86Config`). На практике же другие опции не обязательны для начальной установки сервера.

5.6 Заполнение информации о видеокарте.

Теперь ваш файл `XF86Config` почти готов к использованию. Единственное, что мы не сделали, не заполнили информацию о видеокарте. Сейчас нам следует запустить X сервер в режиме определения видеокарты и дооформить файл `XF86Config`.

Эту информацию вы можете найти и в файлах `modeDB.txt`, `AccelCards` и `Devices` (все эти файлы находятся в каталоге `/usr/X11R6/lib/X11/doc`). Кроме этого существуют различные файлы `README` для конкретных микросхем. Вам следует просмотреть эти файлы и используя эту информацию (частоты, тип микросхем и другие режимы) доопределить файл `XF86Config`. Если какой то информации не хватает, вы можете определить ее путем описанным ниже.

В этом примере мы опишем настройку видеокарты `#9 GXE 64`, использующую микросхему S3 и сервер `XF86_S3`. Эта карта одна из тех, с которыми работает автор, но все описанное ниже справедливо и для другой видеокарты. Перво-наперво вам надо определить тип микросхемы, используемой видеокартой. Команда `SuperProbe` (располагающаяся в каталоге `/usr/X11R6/bin`) сообщит вам эту информацию, но вам необходимо знать под каким именем известна данная микросхема X серверу.

Чтобы определить это запустите команду:

```
X -showconfig
```

Сервер сообщит вам имена микросхем, с которыми он работает (руководство на X сервер также содержит эту информацию). Например, сервер `XF86_S3` сообщит:

```
XFree86 Version 3.1 X Window System
(protocol Version 11, revision 0, vendor release 6000)
Operating System: Linux
Configured drivers

S3 accelerated server for S3 graphics adaptors (Patchlevel 0)
```

```
mmio_928, s3_generic
```

То есть сервер работает с микросхемами `mmio_928` и `s3_generic`. Руководство на сервер `XF86_S3` описывает эти микросхемы и видеокарты, использующие их. В нашем случае видеокарта `#9 GXE 64` использует микросхему `mmio_928`.

Если вы не знаете какая микросхема стоит на видеокарте, X сервер может это определить. Запустите:

```
X -probeonly>/tmp/x.out 2>&1
```

если вы работаете в оболочке shell. Если вы используете csh запустите:

```
X-probeonly &>/tmp/x.out
```

Эту команду следует запускать при низкой загрузке компьютера: она определяет также частоту видеоадаптера и большая загрузка системы может исказить эти данные.

Выходная информация в файле `/tmp/x.out` будет содержать следующие строки:

```
XFree86 version 3.1 X Window System

(protocol, version 11, revision 0, vendor release 6000)
Operating system: Linux
Configured drivers
S3 accelerated server for S3 graphics adaptors (Patchlevel 0)

mmio_928, s3_generic
Several lines deleted

(--) S3 card type 386 486 localbus
(--) S3 chipset 864 rev 0
(--) S3 chipset driver mmio_928
```

Мы видим, что сервер (`XF86_S3`) может работать с микросхемами `mmio_928` и `s3_generic`. Сервер протестировал видеокарту и опознал микросхему `mmio_928`. Следовательно, в секцию `Device` вам следует добавить строку, содержащую имя микросхемы, найденное сервером`L`

```
Section "Device"

# We already hold Identifier here
Identifier "#9 GXE 54"

# Add this line
Chipset "mmio_928"

EndSection
```

Теперь нам требуется определить частоты, поддерживаемые видеокартой. Как мы уже видели, каждый режим разрешения на мониторе требует определенной передачи точек

от видеокарты. Нам необходимо определить какие частоты может обеспечить видеокарта.

Сначала следует просмотреть справочные файлы (`modeDB.txt`, и т.п.) описанные выше и определить, нет ли там описания частот вашей карты. Частоты, как правило представлены списком из 8 или 16-ти значений частот в мегагерцах. Например в файле `modeDB.txt` можно найти строку описания видеокарты Cardinal ET4000:

```
# chip   ram  virtual   clocks                                default-mode
flags
ET4000   1024 1024 768  25 28 38 35 40 45 32 0 "1024x768"
```

Как вы видите, данная карта поддерживает частоты 25, 28, 38, 36, 40, 45, 32 и 0 MHz.

В секции `Device` файла `XF86Config`, вам следует добавить строку `Clocks` со списком частот. В нашем случае мы добавляем строку:

```
Clocks 25 28 38 36 40 45 32 0
```

к секции `Devices`, после `Chipset`.

Заметьте, что порядок частот важен! Вам не следует дублировать или изменять порядок частот.

Если вы не можете найти список частот для вашей карты, X сервер может также определить и эти значения. После вызова команды `X -probeonly`, описанного выше, вы увидите строку:

```
(--) S3 clocks 25 18 28 32 38 02 36 15 40 33 45 32 32 00 00 00
```

Теперь вам осталось лишь добавить строку `Clocks`, перечислив указанные значения. Так как часто список содержит 8 и более значений и не помещается в одной строке, вы можете продолжить список в следующих строках, только не забывайте сохранять порядок указанных значений.

Перед запуском `X -probeonly`, убедитесь что в секции `Devices` нет строк описания `Clocks` или они закомментированы. Если эти значения уже есть, X сервер не будет проверять поддерживаемые платой частоты, а возьмет указанные в строке `Clocks`.

Заметьте, что некоторые видеокарты с акселератором используют микросхему с программируемой частотой (Смотрите руководство XF86_Accel; или XFree86 файл `README` для видеокарты.) Эти микросхемы позволяют X-серверу сообщать карте какую использовать частоту. В этом случае мы вполне вероятно не сможем найти в вышеперечисленных файлах список частот для карты. Или список частот, выдаваемых командой `X -probeonly` будет содержать одно два значения с остальными дублированными или нулевыми значениями:

```
(--) SVGA cldq5434 Specifying a Clocks line makes no sense for this driver
```

Данный пример получен от сервера XF86_SVGA на видеокарте Cirrus Logic PCI.

Для видеоплат, использующих микросхему программирования частоты, вам вместо строки Clocks следует использовать строку ClockChip. Эта строка задает имя микросхемы программирования частоты, установленной на карте. Руководства для каждого сервера описывает их имена. Например, в файле README.S3 мы определили, что несколько S3-864 видеокарт используют микросхему ``ICD2061A''. Следовательно, нам следует использовать строку:

```
ClockChip "icd2061a"
```

вместо строки Clocks. Так же как и строка Clocks, строка ClockChip должна быть в секции Devices после строки Chipset.

Некоторые карты с акселератором требуют определения в файле XF86Config строки Ramdac, описывающей тип используемой микросхемы RAMDAC. Руководство на сервер XF86_Accel описывает подробно опции этой строки. Как правило, X сервер верно определяет тип используемой микросхемы RAMDAC.

Некоторые видеокарты требуют определения нескольких дополнительных опций в секции Devices. Эти опции описаны как в руководствах на ваш X сервер, так и в справочных файлах (например README.cirrus или README.S3). Эти опции устанавливаются строкой Options. Например, видеокарта #9 GXE 64 требует установку двух опций:

```
Option "number_nine"
Option "dac_8_bit"
```

Обычно X сервер работает и без этих опций, но с ними X сервер обеспечивает большую производительность. Существует слишком много всевозможных опций, чтобы из все здесь перечислить. Эти опции зависят от типа установленной видеокарты. Если вы вынуждены использовать эти опции не волнуйтесь, руководства на X сервера и справочные файлы в каталоге /usr/X11R6/lib/X11/doc/ объяснят вам что они значат.

Итак, когда вы закончите, не забудьте завершить строкой EndSection секцию Device, которая будет выглядеть следующим образом:

```
Section "Device"

# Device selection for the #9 GXE 64 only!
Identifier "#9 GXE 64"
Chipset "rnmio_928"
ClockChip "icd2061a"

Option "number_nine"
Option "dac_8_bit"

EndSection
```

Как уже сказано выше, большинство видеокарт требуют строку `Clocks` вместо строки `ClockChip`. Вышеприведенный пример применим только к конкретной видеокарте `#9 GXE 64`.

5.7 Запуск XFree86.

Как только вы закончите описание файла `XF86Config`, вы готовы запустить X сервер и начать работу. Сначала убедитесь, что каталог `/usr/X11R6/bin` включен в ваш путь (переменную `PATH`).

Для запуска X Window наберите команду:

```
startx
```

Это "оболочка" для команды `xinit` (если вы использовали `xinit` в других UNIX-системах). Она запускает X сервер и выполняет файл `.xinitrc` в Вашем домашнем каталоге. `.xinitrc` является скриптом shell, который хранит команды для X клиентов при запуске X сервера. Если файл отсутствует, используется `/usr/X11R6/lib/X11/xinit/xinitrc`.

Простой пример файла `.xinitrc`:

```
#!/bin/sh
xterm -fn 7x13bold -geometry 50x32+10+50 &
xterm -fn 9xl5bold -geometry 50x34+30-10 &
oclock -geometry 70x70-7+7 &
xsetroot -solid, midnightblue &
exec twm
```

Этот скрипт запускает два клиента `xterm` (эмулятор терминала), `oclock` (часы) и устанавливает темно-синий цвет экрана. Затем он запускает `twm`, оконный менеджер. Заметьте, что `twm` запускается через оператор `exec`. Оболочка `/bin/sh`, выполняющая этот скрипт замещается командой `twm` и при окончании работы процесса `twm`, X-сервер также завершает свою работу. Вы можете выйти из `twm`, используя основное меню. Нажмите левую кнопку мыши, находясь на свободном месте экрана. На экране появится меню, которое позволит вам за выйти из `twm`, выбрав пункт `Exit Twm`.

Убедитесь, что последняя команда в файле `.xinitrc` запускается через exec и не запускается в фоне (нет символа `&` в конце строки). Иначе X сервер завершит свою работу, как только он запустит клиента из файла `.xinitrc`.

Кроме этого, вы можете выйти из X, нажав клавиши ctrl-alt-backspace одновременно.

Описанная выше конфигурация файла `.xinitrc` является очень простой. Если вы с ним немного поработаете вы можете получить множество отличных программ и конфигураций окон на экране. Например, оконный менеджер fvwm поддерживает виртуальные экраны, вы можете подобрать различные фонты, цвета, размеры окон, их позиции и так далее, все что вы пожелаете. Хотя система X Window может на первый

взгляд показаться простой, она чрезвычайно мощна и богата различными возможностями. Ознакомьтесь с man-руководствами по `xterm`, `oclock` и `twm`. Немало интересного по настройке можно найти в *The X Window System: A User's Guide* (см. приложение А).

5.8 Проблемы...

Частенько случается, что у вас что-то не получается. Как правило, это связано с ошибками описания вашего файла `XF86Config`. Обычно, неверно указывают временные интервалы синхронизации монитора или частоты видеоплаты. Если у вас изображение на экране сдвинуто или его границы размыты, это точный показатель, что эти значения установлены неверно. Проверьте также, верно ли определили тип микросхемы видеокарты и другие опции в секции `Device` файла `XF86Config`. Убедитесь также, что вы используете необходимый X сервер и что файл `/usr/X11R6/bin/X` является символьной ссылкой на этот сервер.

Если это не поможет, попробуйте запустить X напрямую, используя команду:

```
X>/tmp/x.out 2>&1
```

Затем остановите X сервер (нажав одновременно клавиши ctrl-alt-backspace) и проверьте содержимое файла `/tmp/x.out`. X сервер сообщит все предупреждения и ошибки, например о том, что ваша видеокарта на поддерживает необходимую для вашего монитора частоту.

Файл `VideoModes.doc`, включенный в дистрибутив XFree, содержит много советов по настройке вашего файла `XF86Config`.

Не забудьте, что вы можете использовать комбинации клавиш ctrl-alt-numeric + и ctrl-alt-numeric - для переключения режимов разрешения монитора, перечисленных в секции `Screen` файла `XF86Config`. Если режим с высоким разрешением не работает, попытайтесь установить на меньшее разрешение. Это, по крайней мере, поможет определить вам ошибочные и правильные настройки вашего конфигурационного файла.

Попытайтесь также аппаратно подстроить ваш монитор, используя клавиши управления на мониторе.

Для обсуждения вопросов по XFree86 предназначены группа `comp.windows.x.i386unix` USENET. Неплохая идея подписаться на эту конференцию и описать интересующие вас проблемы: может быть кто-то имеет такие же проблемы.

Имеется также набор файлов `XF86Config`, которые были пожертвованы пользователями. Некоторые из них доступны на архиве `sunsite.unc.edu` в каталоге `/pub/Linux/X11` и в других местах. Вы можете найти файл конфигурации, который кто-то уже написал для Ваших аппаратных средств.

6 Работа в сети

В этой главе обсуждается работа в сетях: как конфигурировать подключение, как использовать протоколы TCP/IP, SLIP и PPP.

6.1 Работа в сетях с TCP/IP.

Linux полностью поддерживает протокол TCP/IP (Transport Control Protocol/Internet Protocol). Протокол TCP/IP оказался наиболее успешным средством объединения компьютеров всего мира в единую сеть. Имея компьютер с системой Linux и адаптер Ethernet, можно подключить компьютер к локальной сети или (при наличии соответствующего подключения сети) к сети Интернет, являющейся всемирной сетью с протоколом TCP/IP.

Собрать небольшую локальную сеть из компьютеров с системой UNIX просто. Каждому компьютеру потребуется адаптер Ethernet, а кроме этого соответствующие кабели и другое аппаратное обеспечение. Если же некоторая фирма или университет предоставляют доступ в Интернет, то компьютер с системой Linux может быть легко подключен к этой сети.

Современная реализация протокола TCP/IP и связанных с ним протоколов для системы Linux называется ``NET-3'', ей предшествовала ``NET-2''. Это не имеет никакого отношения к так называемой версии NET-2 системы BSD UNIX; в действительности, ``NET-3'' в этом контексте означает вторую реализацию протокола TCP/IP для системы Linux.

Реализация NET-3 для системы Linux также поддерживает протоколы SLIP (Serial Line Internet Protocol: протокол последовательного подключения к Интернету) и PPP (Point-to-Point Protocol: протокол "точка-точка"). Протоколы SLIP и PPP позволяют подключаться к Интернету по телефонным линиям с помощью модема. Если соответствующая фирма или университет предоставляет доступ с помощью протокола SLIP или PPP, то нужно лишь дозвониться до SLIP или PPP-сервера. Если же компьютер имеет доступ в Интернет через адаптер Ethernet, то компьютер можно превратить в SLIP или PPP-сервер.

Полная информация о работе с протоколом TCP/IP в системе Linux может быть найдена в Практическом руководстве по использованию NET-3 в системе Linux (NET-3 HOWTO). Этот документ можно скопировать с анонимного доступа FTP к сайту `sunsite.unc.edu`. Документ NET-3 HOWTO является полным руководством к конфигурированию сетей с протоколом TCP/IP; в нем, в том числе, описываются

подключения Ethernet, SLIP и PPP в системе Linux. Практическое руководство по использованию подключения Ethernet в системе Linux (Ethernet HOWTO) родственный ему документ; в нем описываются конфигурации различных драйверов для адаптеров Ethernet в системе Linux. Также имеется документ *Linux Network Administrator's Guide* (Руководство администратора сети в Linux), разработанный в рамках проекта Linux Documentation Project. Более подробно эти документы описаны в приложении А.

Интересна будет также книга Крейга Ханта (Craig Hunt) под названием *TCP/IP Network Administration* (администрирование в сети TCP/IP). В этой книге имеется полная информация по использованию и конфигурированию протокола TCP/IP в системе Linux.

Требования к аппаратуре для работы в сети TCP/IP.

В системе Linux можно использовать протокол TCP/IP без каких-либо сетевых устройств, поскольку конфигурирование режима работы с сетевой "заглушкой" (loopback) позволяют вести диалог с самим собой. Это необходимо для некоторых программ и игр, в которых используется "заглушка".

Однако если предполагается использование системы Linux в сети с протоколом TCP/IP и подключением Ethernet, то адаптер Ethernet потребуется. В системе Linux имеется поддержка распространенных адаптеров Ethernet, таких, как 3com 3c503, HP PCLAN (серии 27245 и 27xxx), Western Digital WD80x3, Novell NE2000/NE1000 и многих других. Подробности можно найти в практических руководствах по адаптерам Ethernet и по аппаратному обеспечению в системе Linux (Linux Ethernet HOWTO, Linux Hardware HOWTO).

Нужно иметь в виду, что для некоторых карт поддержка пусть формально и имеется, но производительность оказывается невысокой, или присутствуют другие ограничения. Примерами могут служить адаптер 3Com 3C501, который работает, но с очень низкой производительностью, и адаптер Racal-Interlan NI6510 с микросхемой am7990, который не работает, если оперативная память превышает 16 MB. Также многие адаптеры являются клонами, совместимыми с NE1000/NE2000, и могут быть причиной различных проблем. Более полное обсуждение вопроса о совместимости сетевого оборудования в системе Linux можно найти в документе Linux Ethernet HOWTO.

Linux также поддерживает протоколы SLIP и PPP, что позволяет заходить в Интернет через телефонные линии с помощью модема. В данном случае модем должен быть совместимым со SLIP или PPP-сервером: большинство серверов требуют как минимум модема со скоростью обмена 14.4bps и протоколом V.32bis. Производительность резко повышается при использовании модема со скоростью обмена 33.6bps и выше.

6.1.1 Конфигурирование TCP/IP.

В этом разделе обсуждается конфигурирование подключения к сети Ethernet с протоколом TCP/IP. Следует отметить, что этот метод будет работать для большинства компьютеров и вариантов системы Linux, но определенно не для всех. Обсуждение будет достаточным для того, чтобы указать верный путь к успешному конфигурированию подключения, но на этом пути имеется множество опасностей и

тонких деталей, которые здесь не оговариваются. Более полная информация может быть найдена в документах *Linux Network Administrators' Guide* и NET-3-HOWTO.

Во-первых, предполагается, что на компьютере с Linux установлено программное обеспечение TCP/IP. Это обеспечение включает основные клиентские программы, такие как `telnet` и `ftp`, команды для системного администрирования, например, `ifconfig` и `route` (обычно эти файлы находятся в каталоге `/etc`), конфигурационные файлы (например, `/etc/hosts`). Если сетевое программное обеспечение еще не установлено, то сделать это помогут упомянутые выше документы по Linux.

Также предполагается, что ядро системы конфигурировано и откомпилировано так, что в нем имеется поддержка TCP/IP. Процедура компиляции ядра описывается в разделе 4.9. Для того, чтобы задать опцию, соответствующую работе в сетях, надо ответить ``yes'' на соответствующие вопросы на этапе `make config` и собрать ядро заново.

После того, как это будет сделано, нужно поменять несколько файлов, которые используются системой NET-3. По сути это простая процедура. К сожалению, разные дистрибутивы системы Linux помещают конфигурационные файлы и программы поддержки протокола TCP/IP в разные каталоги. Как правило, в каталог `/etc`, иногда в каталоги `/etc`, `/usr/etc`, `/usr/etc/inet` или другие странные каталоги. В наихудшем случае придется воспользоваться командой `find`, чтобы найти эти файлы. Следует помнить также, что не все дистрибутивы помещают конфигурационные файлы и программы NET-3 в одно и то же место; напротив, они могут быть рассеяны по разным каталогам.

Следующая информация касается прежде всего подключения Ethernet. Если планируется использовать протоколы SLIP или PPP, то этот раздел следует прочитать, чтобы узнать основные понятия, а при использовании руководствоваться последующими разделами.

Ваши параметры сети.

Прежде чем конфигурировать TCP/IP, нужно узнать определенную информацию о параметрах сети. В большинстве случаев необходимую информацию предоставит системный администратор. Потребуются следующие параметры:

- IP-адрес. Это уникальный адрес компьютера в формате четырех трехзначных десятичных чисел, разделенных точками, например, 128.253.153.54. Этот набор чисел должен предоставить системный администратор.

  Если конфигурируется только режим работы с "заглушкой" (которая вообще-то называется закольцовывающим интерфейсом, loopback mode, т.е. когда нет подключения SLIP и нет адаптера Ethernet, а только моделируется подключение TCP/IP компьютера к самому себе, то IP-адресом будет 127.0.0.1.

- Маска подсети (``netmask''). Формат маски аналогичен формату IP-адреса. Маска определяет, какая часть IP-адреса соответствует номеру локальной подсети

(subnetwork number), а какая номеру компьютера в сети. Маска представляет собой битовый шаблон. При наложении этого шаблона на адрес компьютера (хоста) можно узнать номер того участка сети, к которому относится этот адрес. Это очень важно для рассылки сообщений, поэтому если вдруг окажется, например, что можно установить связь с кем-либо вне локальной сети, а внутри сети такая связь не устанавливается, то может оказаться, что маска указана неверно.

Маска выбиралась администратором сети при ее проектировании, поэтому он должен ее точно знать. Большинство локальных сетей имеют класс C и используют маску 255.255.255.0. Сети класса B имеют маску 255.255.0.0. Программа NET-3 выбирает маску автоматически, предполагая по умолчанию, что подсети отсутствуют, если они не указаны.

То же относится и к адресу "заглушки". Поскольку адрес в этом случае всегда 127.0.0.1, то и маска всегда 255.0.0.0. Это можно указать явно или положиться на маску, даваемую по умолчанию.

- Адрес сети (network address). Этот адрес является результатом побитовой операции AND двух аргументов: IP-адреса компьютера и маски подсети. Например, если маска имеет вид 255.255.255.0, IP-адрес равняется 128.253.154.32, то адрес сети будет 128.253.154.0. Если маска равняется 255.255.0.0, то адрес сети будет равняться 128.253.0.0.

  При использовании только loopback сетевой адрес отсутствует.

- Широковещательный адрес (broadcast address). Этот адрес используется для трансляции пакетов сообщений по всем компьютерам, объединенным в подсеть. Следовательно, если IP-адрес компьютера в сети определяется последним байтом (иными словами, если маска равна 255.255.255.0), то широковещательный адрес является результатом побитовой операции OR выражения 0.0.0.255 с IP-адресом Вашего компьютера.

  Например, если IP-адрес компьютера 128.253.154.32, а маска 255.255.255.0, то широковещательный адрес будет равен 128.253.154.255.

  Исторически сложилось, что в некоторых сетях в качестве широковещательного используется адрес сети. Этот факт следует знать, и если по этому вопросу возникают сомнения, надо обращаться к системному администратору. Разумеется, во многих случаях для конфигурирования сети достаточно будет взять соответствующие файлы с других компьютеров в той же сети и изменить в них только IP-адрес компьютера.

  При использовании только loopback широковещательный адрес отсутствует.

- Адрес шлюза (gateway address). Это адрес компьютера, который действительно является ``шлюзом'' во внешний мир (т.е. к компьютерам вне локальной сети). Во многих случаях адрес шлюза получается из IP-адреса компьютера заменой последних групп цифр на .1. Например, если IP-адрес компьютера равняется 128.253.154.32, то адрес шлюза может иметь значение 128.253.154.1. Это значение может также сообщить системный администратор.

  Реально шлюзов может быть несколько. В действительности *шлюз* это компьютер, который находится в двух различных сетях (т.е. имеет IP-адреса в различных подсетях). Шлюз занимается тем, что пересылает пакеты сообщений между этими двумя сетями. Многие сети имеют единственный шлюз во внешний мир (который представляет собой непосредственно примыкающую сеть). Однако в некоторых случаях к локальной сети примыкает несколько других сетей, и для каждой из них имеется свой шлюз.

  При использовании только loopback адрес шлюза отсутствует. То же верно и в случае, когда локальная сеть изолирована от всех остальных.

- Адрес сервера имен в сети (name server address). У большинства компьютеров в сети имеется сервер, который преобразует имя компьютера в IP-адрес. Адрес этого сервера должен сообщить администратор сети. Такой сервер можно также запустить на собственном компьютере (программа называется `named`); в этом случае адрес сервера имен в сети равняется 127.0.0.1. Кроме тех случаев, когда *абсолютно необходимо* иметь собственный сервер имен, лучше воспользоваться тем, который имеется (если имеется). Совершенно отдельный вопрос конфигурирование программы `named`. Сейчас он рассматриваться не будет, поскольку цель данного раздела дать возможность приступить к работе в сети. Вопросы, касающиеся имен в сети, будут рассмотрены позднее.

  При использовании только loopback адрес сервера имен отсутствует.

Пользователям SLIP/PPP информация, приведенная выше (кроме адреса сервера имен), может не требоваться. При подключении SLIP клиентский IP-адрес определяется либо статически, когда при каждом новом соединении клиенту дается один и тот же IP-адрес, либо динамически, когда адрес в момент соединения выбирается из некоторого множества свободных IP-адресов сервера. В последующих разделах конфигурирование подключения SLIP будет обсуждаться более подробно.

В реализации NET-3 заложена полная маршрутизация (full routing), поддерживаются множественные маршруты и подсети (последние на данный момент только по границам байта), и тому подобное. Приведенное выше иллюстрирует только принципиальные основы построения сетей TCP/IP. Конкретная сеть может обладать рядом особенностей; выяснить их можно у администратора сети. Также полезно будет почитать экранную документацию к программам `route` и `ifconfig`. Конфигурирование сетей TCP/IP лежит далеко за пределами темы данной книги, однако приведенное рассмотрение должно дать достаточно информации, чтобы пользователь смог выяснить остальное самостоятельно.

Файлы `rc` для работы в сетях.

Файлы rc являются конфигурационными скриптами системы. Эти файлы выполняются скриптом init, который запускается при загрузке системы и в свою очередь запускает основные демоны системы (такие как sendmail, cron и т.д.) и конфигурирует параметры сети, устанавливает имя компьютера и т.п. Обычно файлы rc находятся в каталоге /etc/rc.d, однако в некоторых версиях их помещают в каталог /etc. Дистрибутивы Slackware используют файлы rc.inet1 и т.д. в каталоге /etc/rc.d, а дистрибутивы RedHat и другие используют набор каталогов.

Далее будут описываться файлы rc, с помощью которых конфигурируется программное обеспечение протокола TCP/IP. Таких файлов два: rc.inet1 и rc.inet2. Файл rc.inet1 служит для конфигурирования основных параметров сети (например, устанавливает IP-адрес и информацию о маршрутах), а файл rc.inet2 запускает демоны TCP/IP (telnetd, ftpd и т.д.).

Во многих версиях системы эти два файла объединяются в один, называемый rc.inet или rc.net. Имя rc-файла не имеет значения; важно, чтобы он правильно исполнял свои функции и запускался во время загрузки скриптом init. Последнее обеспечивается соответствующими строками в файле /etc/inittab (при необходимости этот файл нужно отредактировать, раскомментировав строчки, относящиеся к нужным rc-файлам). В худшем случае пользователю придется создавать самому файлы rc.inet1 и rc.inet2 и добавить запускающие их строчки в файл /etc/inittab.

Как уже говорилось выше, файл rc.inet1 производит основное конфигурирование сетевого интерфейса. Сюда входит задание IP-адреса компьютера и информации о таблице маршрутов сети. Таблицы маршрутов используются для того, чтобы направлять входящие в сеть и выходящие из сети датаграммы (datagrams). В самом простом случае имеется всего три маршрута: один для пакетов, посылаемых на данный компьютер, второй на другие компьютеры в той же сети, и третий на компьютеры вне локальной сети (через шлюз). Для конфигурирования этих параметров используются две программы: ifconfig и route. Обе программы обычно находятся в каталоге /etc.

Программа ifconfig используется для конфигурирования интерфейса сетевых устройств. При этом задаются такие параметры, как IP-адрес, маска, широковещательный адрес и т.п. Программа route используется для создания и изменения элементов в таблице маршрутов.

Для большинства конфигураций файл rc.inet1 может выглядеть примерно следующим образом. Этот пример, разумеется, следует приспособить к конкретному компьютеру и *не* использовать те IP-адреса, которые указаны в этом примере (они соответствуют некоторым конкретным компьютерам в сети Интернет).

```
#!/bin/sh
# Это пример файла /etc/rc.d/rc.inet1 - конфигурирование
#     интерфейсов TCP/IP

# Сначала конфигурируем loopback

HOSTNAME=`host name`

/etc/ifconfig lo 127.0.0.1      # используется маска по умолчанию 255.0.0.0
/etc/route add 127.0.0.1        # маршрут к loopback
```

```
# Теперь конфигурируем устройства Ethernet. Если используются только
# loopback и/или SLIP, закомментируйте остальные строки.

# Измените на значения для своего компьютера
IPADDR="128.253.154.32"         # ЗАМЕНИТЕ на ВАШ IP-адрес
NETMASK="255.255.255.0"         # ЗАМЕНИТЕ на ВАШУ маску
NETWORK="128.253.154.0"         # ЗАМЕНИТЕ на адрес ВАШЕЙ сети
BROADCAST="128.253.154.255"     # ЗАМЕНИТЕ на широковещательный
                                # адрес ВАШЕЙ сети, если он есть.
                                # Если нет, оставьте свободное
                                # место и продолжайте редактировать дальше.
GATEWAY="128.253.154.1"         # ЗАМЕНИТЕ на адрес ВАШЕГО шлюза

/etc/ifconfig eth0 ${IPADDR} netmask ${NETMASK} broadcast ${BROADCAST}

# Если у вас нет широковещательного адреса, то замените
# предыдущую строчку на следующую:
# /etc/ifconfig eth0 ${IPADDR} netmask ${NETMASK}

/etc/route add ${NETWORK}

# Следующая строчка нужна только в случае, если есть шлюз
# (т.е. сеть связана с внешним миром)
/etc/route add default gw ${GATEWAY} metric 1
```

Еще раз отметим, что для конкретной конфигурации компьютера этот файл, возможно, придется в чем-то изменить. Приведенных в данном примере инструкций достаточно для сети простой конфигурации, однако для всех возможных конфигураций этого примера будет определенно недостаточно.

Файл `rc.inet2` запускает различные серверы, входящие в обеспечение протокола TCP/IP. Наиболее важным из них является сервер `inetd`. Этот сервер находится в фоновом режиме и наблюдает за работой сетевых портов. Когда к компьютеру пытаются обратиться через какой-либо порт (например, на этот порт приходит команда `telnet`), сервер `inetd` создает для этого порта копию соответствующего демона (в случае прихода команды `telnet` это будет демон `in.telnetd`). Это оказывается проще, чем иметь сразу несколько отдельных работающих демонов (т.е. отдельных копий `telnetd`, `ftpd` и т.д.). В принятой схеме они запускаются по мере надобности программой `inetd`.

`Syslogd` демон протоколирования системы. Он собирает сообщения о работе от разных программ и складывает их в файлы протоколов согласно конфигурации, содержащейся в файле `/etc/syslogd.conf`. Функции сервера `routed` поддерживать динамическую информацию о маршрутах. Если компьютер пытается послать пакеты сообщений на другую сеть, то для этого ему могут потребоваться дополнительные данные в таблице маршрутов. Заботы о действиях с таблицей маршрутов берет на себя сервер `routed`, и пользователь может не вмешиваться в этот процесс.

В приведенном ниже примере файл `rc.inet2` запускает лишь необходимый минимум серверов. Помимо указанных, существует еще много других серверов, и многие из них относятся к конфигурированию NFS. Однако при попытке конфигурировать программное обеспечение протокола TCP/IP лучше всего начинать с минимальной конфигурации, а более сложные элементы (такие, как NFS) добавлять, когда часть программ уже работает.

Также отметим, что в приведенном ниже файле предполагается, что демоны работы в сетях располагаются в каталоге /etc. Разумеется, для конкретной конфигурации этот файл может быть отредактирован.

```
#!/bin/sh
# Пример файла /etc/rc.d/rc.inet2

# Запуск демона syslogd
if [ -f /etc/syslogd ] then
   /etc/syslogd
fi

# Запуск сервера inetd
if [ -f /etc/inetd ] then
   /etc/inetd
fi

# Запуск сервера routed
if [ -f /etc/routed ] then
   /etc/routed -q
fi
```

Среди дополнительных серверов, которые могут быть запущены скриптом rc.inet2, имеется сервер named. Это сервер имен в сети, и его функция состоит в том, чтобы преобразовывать (локальные) IP-адреса в имена компьютеров и обратно. Если в сети нет другого сервера имен или если требуется дать (локальные) имена другим компьютерам того же домена, можно воспользоваться этим сервером (однако для большинства конфигураций это оказывается ненужным). Конфигурирование сервера named является довольно сложным и требует предварительного планирования; тем, кому это интересно, можно порекомендовать обратиться к книгам по администрированию сетей TCP/IP.

Файл /etc/hosts.

Файл /etc/hosts содержит список IP-адресов и соответствующих им имен компьютеров. Как правило, файл /etc/hosts содержит информацию только о данном компьютере и, возможно, о других важных компьютерах (таких, как сервер имен или шлюз). Для других компьютеров в локальной сети преобразования IP-адресов в имена и обратно будет обеспечиваться локальным сервером имен в сети.

Например, если имя компьютера loomer.vpizza.com, а IP-адрес 128.253.154.32, то файл /etc/hosts на этом компьютере может выглядеть так:

```
127.0.0.1               localhost
128.253.154.32          loomer.vpizza.com loomer
```

Если используется только loopback, то единственная строка в файле /etc/hosts должна начинаться с чисел 127.0.0.1, за которыми должны следовать localhost и затем имя компьютера (прим. переводчика: так должно быть вообще в любой конфигурации: дело в том, что если loopback будет первым в этом файл, системе не надо будет просматривать остальные строки).

Файл /etc/networks.

Файл /etc/networks содержит список имен и адресов как собственной сети, так и других сетей. Этот список используется командой route и дает возможность (при желании) для обозначения сетей использовать имена, а не адреса.

Каждая сеть, к которой предполагается создать отдельный маршрут командой route (обычно вызываемой скриптом rc.inet1), *должна* иметь соответствующую строку в файле /etc/networks.

Приведем пример:

```
default       0.0.0.0         # маршрут по умолчанию - обязателен
loopnet       127.0.0.0       # сеть для loopback - обязательна
mynet         128.253.154.0   # ИЗМЕНИТЕ на адрес ВАШЕЙ сети
```

Файл /etc/host.conf.

В этом файле определяется, как в системе будут разрешаться имена компьютеров. В нем должны содержаться две строки:

```
order hosts,bind
multi on
```

Эти строки сообщают соответствующим процедурам, что сначала все имена должны искаться в файле /etc/hosts, а затем делается обращение к серверу имен (если он есть). Строка, начинающаяся с multi, позволяет в файле /etc/hosts для данного компьютера иметь несколько IP-адресов.

Файл /etc/resolv.conf.

В этом файле содержится конфигурация для программы разрешения имен. Здесь задается имя сервера имен (если таковой имеется) и имя домена. Имя домена получается, если из полного имени компьютера (например, для компьютера, зарегистрированного в сети Интернет) вычеркнуть имя компьютера (хоста). Например, если полное имя выглядит как loomer.vpizza.com, именем домена будет vpizza.com.

Пусть, например, имя компьютера goober.norelco.com, а сервер имен для него имеет IP-адрес, равный 128.253.154.5. Тогда файл /etc/resolv.conf может выглядеть так:

```
domain      norelco.com
nameserver  127.253.154.5
```

Можно указать и несколько серверов имен, тогда для каждого из них в файле /etc/resolv.conf будет строка, начинающаяся с nameserver.

Установка имени хоста.

Имя компьютера устанавливается командой hostname. Она обычно вызывается из скрипта /etc/rc или /etc/rc.local; чтобы определить, где именно, надо просто

просмотреть содержимое этих файлов и найти ее имя. Например, если (полное) имя компьютера должно быть `loomer.vpizza.com`, то надо отредактировать соответствующий `rc`-файл и вставить туда строку

```
/bin/hostname loomer.vpizza.com
```

Отметим, что исполняемый файл с именем `hostname` может и не присутствовать в каталоге `/bin`.

Проверка работоспособности.

После того, как все сконфигурировано, можно попробовать перезагрузить систему с новым ядром и попытаться использовать сеть. Неисправность может случиться во многих местах, так что разумно будет проверять правильность конфигурации сети по частям. К примеру, не следует начинать проверку работы сети с запуска программы Mosaic через сетевые функции системы X.

Для вывода на экран таблиц маршрутов можно использовать программу `netstat`. Именно эти таблицы являются наиболее частым источником неполадок. Точный синтаксис команды `netstat` подскажет man-страница. Для проверки подключения к сети мы предлагаем использовать клиентские программы, такие, как `telnet`, и попытаться подключится к компьютерам как в своей локальной (под)сети, так и за ее пределами. Такой подход сузит область поиска возможных неполадок. Например, если не удается соединиться с компьютерами в своей сети, но соединение с компьютерами вне этой сети устанавливается, то скорее всего ошибка связана с маской и с конфигурированием таблицы маршрутов. Для непосредственной работы с содержимым таблиц маршрутов можно вызвать (как пользователь `root`) команду `route` для работы с записями в Вашей таблице маршрутизации.

Можно также проверить соединение через сеть путем указания точного IP-адреса вместо обращения по имени. Например, если не работает команда

```
telnet shoop.vpizza.com
```

то проблема может заключаться в неверной конфигурации сервера имен. Если при указании точного IP-адреса этого компьютера соединение устанавливается, то можно заключить, что почти наверняка в основном конфигурирование сети выполнено верно, а проблема связана с указанием адреса сервера имен.

Поиск ошибок в конфигурировании сети может оказаться трудной задачей, и она не может быть здесь освещена целиком. Если у вас под рукой нет специалиста, который помог бы вам, мы предлагаем обратиться к книге *Linux Network Administrators' Guide*, выполненной в рамках проекта Linux Documentation Project.

6.1.2 Настройка SLIP.

Протокол SLIP (Serial Line Internet Protocol) позволяет использовать протокол TCP/IP при подключении через последовательный порт. К этому порту может быть подключен либо модем, либо выделенная асинхронная телефонная линия (leased asynchronous line) какого-либо вида. Разумеется, при таком подключении требуется

близко расположенный SLIP-сервер с соответствующими телефонными номерами. Многие университеты и коммерческие организации предоставляют доступ к SLIP-серверу за умеренную плату.

В настоящий момент имеется две основные программы, работающие с протоколом SLIP: `dip` и `slattach`. Обе программы предназначены для того, чтобы установить соединение SLIP через последовательное устройство. Для осуществления соединения SLIP *необходимо* воспользоваться одной из этих программ; просто позвонив на номер сервера одной из коммуникационных программ (например, `kermit`) и выполнив команды `ifconfig` и `route`, того же результата достичь не удастся. Дело в том, что программы `dip` и `slattach` выдают специальное обращение обращение *ioctl()* к системе, которое перехватывает управление последовательным устройством и начинает использовать его как интерфейс SLIP.

Программа `dip` может позвонить на номер SLIP-сервера, произвести необходимый диалог для входа в систему (например, ввести по приглашению имя пользователя и пароль) и инициализировать подключение SLIP через открытый последовательный порт. Напротив, программа `slattach` не делает почти ничего помимо захвата последовательного устройства для использования его для подключения SLIP. Она полезна в случае, если имеется постоянное соединение со SLIP-сервером через выделенную телефонную линию, и для установления связи не надо звонить на сервер, вводить пароль и т.п. Большинству пользователей, работающих с протоколом SLIP, нужнее оказывается программа `dip`.

Программа `dip` может быть использована также для конфигурирования системы Linux в качестве SLIP-сервера. При этом другие компьютеры могут звонить на данный компьютер и подключаться к сети через вторичное подключение Ethernet (secondary Ethernet connection). Для более полной информации об этом следует обратиться к соответствующим документам и man-документации к `dip`.

Подключение SLIP кардинально отличается от подключения Ethernet тем, что в "сети" присутствует всего два компьютера: SLIP-клиент (ваш компьютер) и SLIP-сервер. По этой причине соединение SLIP часто называется соединением "точка-точка" (point-to-point connection). Обобщение этой идеи под названием "протокол PPP" (Point to Point Protocol) также реализовано в системе Linux.

При инициализации подключения к SLIP-серверу последний дает клиенту IP-адрес. При выборе адреса используется два основных метода. Некоторые SLIP-серверы дают статические IP-адреса, так что один и тот же клиент будет всякий раз при подключении получать один и тот же адрес. Однако многие SLIP-серверы дают динамические IP-адреса, которые могут быть новыми при каждом новом подключении. Как правило SLIP-сервер при соединении сообщает клиенту его IP-адрес и IP-адрес шлюза. Программа `dip` может прочесть эти данные, сообщаемые сервером при входе в систему, и использовать их затем при конфигурировании устройств SLIP.

По сути настройка SLIP напоминает настройку сетевой заглушки или адаптера Ethernet. Основные отличия обсуждаются ниже. Для ознакомления надо прочесть предыдущий раздел, посвященный конфигурированию основных файлов, относящихся к протоколу TCP/IP, и затем прочесть про отличия в данном разделе.

Соединение SLIP программой `dip` со статическими IP-адресами.

Если SLIP-сервер дает статические IP-адреса, их можно вписать в файл `/etc/hosts`. Можно также конфигурировать файлы, упомянутые в предыдущих разделах: `rc.inet2`, `host.conf` и `resolv.conf`.

Также можно сконфигурировать файл `rc.inet1` как описано выше. Для сетевой заглушки (вырожденной сети) надо выполнить только программы `ifconfig` и `route`. При использовании программы `dip` она сама выполнит команды `ifconfig` и `route` для устройства SLIP. Однако если используется программа `slattach`, то команды `ifconfig`/`route` *надо будет* включить в файл `rc.inet1` для устройства SLIP (см. ниже).

Во время соединения программа `dip` *должна* конфигурировать должным образом таблицы маршрутов (routing tables) для устройства SLIP. Однако для некоторых конфигураций поведение программы `dip` может оказаться неправильным, и после того, как программа `dip` произведет соединение с сервером, придется выполнять команды `ifconfig` и `route` вручную. Проще всего это сделать из скрипта командной оболочки, который запускается из программы `dip` и немедленно исполняет соответствующие команды конфигурирования. В большинстве случаев адрес шлюза совпадает с адресом SLIP-сервера. Этот адрес может быть известен заранее; кроме того, этот адрес будет выдан на экран SLIP-сервером при соединении. Диалоговый скрипт программы `dip` (см. ниже) может воспринимать и использовать эту информацию.

Если программа `dip` сконфигурировала интерфейс неправильно, то программа `ifconfig` может затребовать значение аргумента `pointopoint`. Например, если адрес SLIP-сервера равен 128.253.154.2, а IP-адрес клиента 128.253.154.32, то может потребоваться войти в систему как `root` и после соединения с помощью программы `dip` выполнить команду

```
ifconfig sl0 128.253.154.32 pointopoint 128.253.154.2
```

При выяснении деталей может оказаться полезной man-документация к команде `ifconfig`.

Следует напомнить, что используемые командами `ifconfig` и `route` имена устройств имеют вид `sl0`, `sl1` и т.д., в отличие от устройств Ethernet, называемых `eth0`, `eth1` и т.д.

В разделе 6.1.2, ниже, будет объяснено, как конфигурировать программу `dip` для соединения со SLIP-сервером.

Соединение SLIP программой `slattach` со статическими IP-адресами.

При наличии выделенной телефонной линии (или кабеля), ведущей непосредственно к SLIP-серверу, нет необходимости использовать программу `dip` для инициализации соединения. Вместо нее для конфигурирования устройств SLIP может использоваться программа `slattach`.

В этом случае файл `/etc/rc.inet1` должен выглядеть примерно так:

```
#!/bin/sh
IPADDR="128.253.154.32"   # ЗАМЕНИТЕ на ВАШ IP-адрес
REMADDR="128.253.154.2"   # ЗАМЕНИТЕ на адрес ВАШЕГО SLIP-сервера
```

```
# Замените данные в следующих строках на параметры ваших
# последовательных устройств, используемых для соединения SLIP
slattach -p cslip -s 19200 /dev/ttyS0
/etc/ifconfig sl0 $IPADDR pointopoint $REMADDR up
/etc/route add default gw $REMADDR
```

`slattach` размещает первое свободное устройство SLIP (`sl0`, `sl1` и т.д.) на указанную последовательную линию.

Первым параметром программы `slattach` является тип используемого протокола SLIP. На настоящее время такими могут быть `slip` и `cslip`. Первый из них обычный протокол SLIP, а второй протокол SLIP со сжатием заголовка датаграммы (datagram header compression). В большинстве случаев нужно использовать `cslip`; однако если возникают проблемы, можно попробовать `slip`.

Если имеется более одного интерфейса SLIP, то надо создать некоторый принцип маршрутизации (routing considerations). Надо будет решить, какие нужно создать маршруты; такие решения принимаются только на основе знаний о конкретной структуре сети и подключений. При этом большую помощь окажет книга по конфигурированию сетей TCP/IP, а также man-страница по программе `route`.

Соединение SLIP программой `dip` с динамическими IP-адресами.

Если SLIP-сервер назначает IP-адреса динамически, то никогда нельзя знать IP-адрес компьютера заранее. Следовательно, включать соответствующую запись в файл `/etc/hosts` нельзя (при этом IP-адрес для loopback 127.0.0.1 должен присутствовать).

Многие SLIP-серверы выдают IP-адрес, выданный клиенту (а также собственный), во время соединения. Например, некоторые типы SLIP-серверов выводят такие сообщения:

```
Your IP address is 128.253.154.44.
Server address is 128.253.154.2.
```

Программа `dip` может перехватить эти цифры из выдачи сервера и использовать их для конфигурирования устройств SLIP.

Информация по составлению файлов конфигурации протокола TCP/IP для использования протоколом SLIP дана выше в разделе 6.1.2. Далее описывается конфигурирование программы `dip` для соединения со SLIP-сервером.

Использование `dip`.

`dip` заметно упрощает процесс подключения к SLIP-серверу, входа в систему и конфигурирования устройств SLIP. Программа `dip` будет весьма полезна кроме, может быть, случаев, когда для подключения к SLIP-серверу используется выделенная телефонная линия (leased line).

Для использования программы dip требуется написать диалоговый скрипт (chat script). В этом скрипте будет содержаться список команд, с помощью которых будет осуществляться диалог со SLIP-сервером во время входа в систему. Эти команды могут автоматически посылать на сервер имя пользователя и пароль, а также получить от сервера информацию о вашем IP-адресе.

Ниже дается пример диалогового скрипта программы dip для работы с сервером с динамическими IP-адресами. Для серверов со статическими адресами надо будет в начале скрипта установить значения переменных $local и $remote, равных соответственно вашему (local) IP-адресу и IP-адресу сервера. Подробно об этом написано в man-документации к dip.

```
main:
  # Установка максимального размера передаваемого блока
  # (Maximum Transfer Unit). Это максимальный размер пакета,
  # передаваемого через устройство SLIP device. Многие
  # SLIP-серверы используют либо 1500 либо 1006; при сомнениях
  # следует спросить администратора сети.
  get $mtu 1500

  # Установить маршрут SLIP как маршрут по умолчанию
  default

  # Установить нужный последовательный порт и скорость передачи
  port cua03
  speed 38400

  # Выполнить начальную установку модема и терминальной линии.
  # Если эта команда создает проблемы, закомментируйте ее.
  reset

  # Подготовка к набору номера. Подставьте в следующей строке
  # инициализационную команду (initialization string) для
  # вашего модема.
  send ATT&C1&D2\\N3&Q5%M3%C1N1W1L1S48=7\r
  wait OK 2
  if $errlvl != 0 goto error

  # Набираем номер SLIP-сервера
  dial 2546000
  if $errlvl != 0 goto error
  wait CONNECT 60
  if $errlvl != 0 goto error

  # Связь установлена. Входим в систему.
login:
  sleep 3
  send \r\n\r\n

  # Ждем приглашения в систему
  wait login: 10
  if $errlvl != 0 goto error

  # Посылаем имя пользователя
  send USERNAME\n

  # Ждем приглашения ввести пароль
  wait ord: 5
  if $errlvl != 0 goto error
```

```
  # Посылаем пароль
  send PASSWORD\n

  # Ждем сигнала готовности SLIP-сервера
  wait annex: 30
  if $errlvl != 0 goto error

  # Посылаем команды SLIP-серверу для инициализации соединения
  send slip\n
  wait Annex 30

  # Получаем значение IP-адреса SLIP-сервера. Команда
  # "get...remote" читает текст в формате xxx.xxx.xxx.xxx,
  # и присваивает его переменной, указанной вторым аргументом
  # (в данном случае переменной $remote).
  get $remote remote
  if $errlvl != 0 goto error
  wait Your 30

  # Получаем значение своего IP-адреса от SLIP-сервера и
  # присваиваем его переменной $local.
  get $local remote
  if $errlvl != 0 goto error

  # Запускаем связь через соединение SLIP
done:
  print CONNECTED to $remote at $rmtip
  print GATEWAY address $rmtip
  print LOCAL address $local
  mode SLIP
  goto exit
error:
  print SLIP to $remote failed.

exit:
```

`dip` автоматически исполняет команды `ifconfig` и `route` с учетом значений переменных `$local` и `$remote`. В данном примере этим переменным присваивает значение команда `get...remote`, которая получает текстовую строку от SLIP-сервера и присваивает ее указанной переменной.

Если запускаемые программой `dip` команды `ifconfig` и `route` не работают, то можно либо после выполнения программы `dip` запустить правильные команды из скрипта программной оболочки, либо непосредственно изменить текст скрипта для программы `dip`. Запуская программу `dip` с опцией `-v` можно получать протокол процесса подключения. Эта информация поможет выяснить источник неудачи, если связь не будет устанавливаться.

Для того, чтобы запустить программу `dip` и установить подключение SLIP, можно использовать команду следующего типа:

```
/etc/dip/dip -v /etc/dip/mychat 2>&1
```

При этом различные файлы программы `dip` и диалоговый скрипт `mychat.dip` хранятся в каталоге `/etc/dip`.

Вышеприведенное обсуждение должно быть достаточным для вашего хорошего самочувствия на славном пути в сетевое сообщество через Ethernet или SLIP. И вновь мы настоятельно рекомендуем заглянуть в книгу по TCP/IP, особенно, если ваша сеть имеет специфику в маршрутизации, отличающую ее от рассмотренных здесь.

6.2 Сети на основе телефонных линий и PPP.

Linux поддерживает полную реализацию протокола PPP (Point-to-Point Protocol). Этот протокол является механизмом создания и эксплуатации Интернет-протокола (Internet Protocol, или IP) и других сетевых протоколов для связи через последовательное соединение (через нуль-модемный кабель), через связь, установленную посредством программы `telnet`, или через связь посредством модемов и телефонных линий (разумеется, включая и цифровые линии, такие, как ISDN). В данном разделе описывается только конфигурирование протокола PPP для клиента, осуществляющего связь через аналоговый модем с удаленным компьютером, предоставляющим доступ к себе через телефонные линии с помощью протокола PPP.

Для того, чтобы получить полную информацию по конфигурированию протокола PPP в системе Linux, читателю предлагается обратиться к Практическому руководству по протоколу PPP в системе Linux (Linux PPP HOWTO), который можно получить через анонимный доступ FTP к сайту `sunsite.unc.edu`. Этот документ является полным руководством по конфигурированию протокола PPP в системе Linux, включая конфигурирование модемов, линий ISDN, нуль-модемных кабелей. Большая часть информации в этом разделе была взята из этого документа. Также доступен документ *Linux Network Administrator's Guide*, являющийся частью проекта Linux Documentation Project. Более подробно об этих документах можно прочесть в приложении A.

6.2.1 Что нужно для начала работ.

Предполагается, что ядро системы Linux сконфигурировано и откомпилировано с включением в него функций поддержки протокола TCP/IP. Информация о компиляции ядра приведена в разделе 4.9 . Для того, чтобы обеспечить сетевые функции в ядре системы, надо ответить ``yes'' на соответствующие вопросы во время шага `make config` и затем собрать ядро заново. Предполагается также, что сам пакет ppp откомпилирован и установлен, и что с ядром Linux версии 1.2.x используется версия PPP 2.1.2, а с ядром Linux версий 1.3.X/2.0.x версия PPP 2.2.0. На время написания книги последняя официальная версия пакета PPP имеет номер ppp-2.2f. Если планируется использовать модули для того, чтобы загрузить программы PPP в ядро системы, то следует прочесть kerneld mini-HOWTO. *Крайне рекомендуется использовать только те сочетания версий ядра системы и программного обеспечения PPP, про которые известно, что они стабильны в совокупности.*

Надо также прочесть следующую документацию:

- документацию, сопровождающую пакет программного обеспечения PPP;
- man-страницы к программам `pppd` и `chat` (используйте команды `man chat` и `man pppd`);
- Linux Network Administration Guide (NAG);
- Net-2/3 HOWTO;
- документацию к ядру системы, которая поставилась в `/usr/src/linux/Documentation` при установке исходных текстов Linux;
- www-страницу, посвященную конфигурированию модема (Modem Setup Information), по адресу `http://www.in.net/info/modems/index.html`;
- замечательные книги по системам UNIX и Linux, выходящие в издательстве O'Reilly and Associates. Электронный каталог в системе Интернет (O'Reilly and Associates On-Line catalog) находится на сайте `http://www.ora.com/`. Новичку в системах UNIX и Linux следует приобрести некоторые из этих книг!
- документ PPP-FAQ (Часто задаваемые вопросы по протоколу PPP), который поддерживает Эл Лонгейр (Al Longyear). Этот документ находится по адресу `ftp://sunsite.unc.edu/pub/Linux/docs/faqs` (см. приложение В). Здесь можно найти много полезной информации в формате вопрос-ответ; эта информация будет очень полезна при выяснении, почему связь PPP неправильно работает.

6.2.2 Обзор этапов настройки.

При конфигурировании системы для использования протокола PPP будет пройдено несколько этапов. Читателю предлагается внимательно прочитать, что придется делать на каждом из этих этапов, прежде чем пытаться совершить подключение PPP. Каждый из этих этапов будет подробно рассмотрен далее.

1. Убедиться, что в ядре откомпилирован блок поддержки протокола TCP/IP.
2. Убедиться, что в ядре откомпилирован (либо статически, либо как загружаемый модуль) блок поддержки протокола PPP.
3. Убедиться, что программное обеспечение протокола PPP откомпилировано и установлено.
4. Убедиться, что модем сконфигурирован и установлен (подключен) к известному последовательному порту компьютера.
5. Убедиться, что известна следующая информация от провайдера PPP (обычно это провайдер Интернет):
    - Телефонный номер, по которому следует звонить на сервер.
    - Тип назначения клиентских IP-адресов (статический или динамический). В первом случае надо знать статический IP-адрес.
    - IP-адрес сервера имен (DNS server), который будет разрешать имена компьютеров при соединении.

Убедиться, что в ядре откомпилирован блок поддержки протокола TCP/IP.

Программное обеспечение протокола PPP в системе Linux состоит из двух частей: демон протокола PPP и поддержка протокола PPP ядром. В большинстве дистрибутивов в ядре по умолчанию имеется поддержка протокола PPP, однако для некоторых дистрибутивов это не так. Для того, чтобы убедиться, что в ядре имеется блок поддержки протокола TCP/IP, можно ввести следующую команду:

```
grep -i "TCP/IP" /var/adm/messages
```

Если будет получена строчка, напоминающая:

```
Jun  8 09:52:08 gemini kernel: Swansea University Computer Society TCP/IP
for NET3.019
```

то значит, поддержка протокола TCP/IP откомпилирована. Эту же информацию можно увидеть на экране консоли во время загрузки. Однако на быстрых компьютерах это сообщение пролетает слишком быстро, чтобы его увидеть. Для того, чтобы пролистать сведения и найти это сообщение, можно использовать комбинацию клавиш Shift-PageUp.

Убедиться, что в ядре откомпилирован блок поддержки протокола PPP.

Если при загрузке ядро системы выдает сообщения типа:

```
  PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
  PPP line discipline registered.
```

то значит, поддержка протокола PPP есть. Можно также ввести команду:

```
grep -i "PPP" /var/adm/messages
```

Строка, напоминающая:

```
Jun  8 09:52:08 gemini kernel: PPP: version 0.2.7 (4 channels)
NEW_TTY_DRIVERS O PTIMIZE_FLAGS
```

также обозначает, что протокол PPP поддерживается.

Убедиться, что модем сконфигурирован.

Надо убедиться, что модем сконфигурирован верно и что известен последовательный порт, к которому он подключен. Соотношение здесь таково:

- DOS com1: = Linux /dev/cua0 (/dev/ttyS0)
- DOS com2: = Linux /dev/cua1 (/dev/ttyS1),
- и т.д.

Исторически сложилось так, что в системе Linux устройства cua*x* используются для исходящих звонков, а устройства ttyS*x* для входящих. Начиная с версии ядра 2.0.*x* такое положение изменилось, и теперь как для входящих, так и для исходящих звонков нужно использовать устройство ttyS*x*. Возможно, что в последующих версиях ядра устройства cua*x* вообще исчезнут.

Если используется высокоскоростной модем (14400 бод и выше, не имеет значения, внешний или внутренний), то последовательный порт должен уметь справляться с таким потоком данных, в частности, когда используется сжатие данных.

Это требует, чтобы последовательный порт использовал современную версию UART (Universal Asynchronous Receiver Transmitter), например, 16550A. На старом компьютере (или на компьютере со старым контроллером последовательных портов)

может быть всего лишь 8250 UART, что создаст значительные трудности при использовании высокоскоростных модемов.

Для выдачи характеристики UART порта надо ввести команду:

```
# setserial -a /dev/ttySx
```

Если имеется высокоскоростной модем, а характеристики порта недостаточные, следует позаботиться о приобретении нового адаптера последовательных портов (их цена менее $50). Для использования протокола PPP нужно будет сконфигурировать модем, поэтому *нужно прочесть руководство к модему*. Большинство модемов выпускаются с именно теми установками по умолчанию, которые требуются для протокола PPP. Рекомендуем следующую конфигурацию (в стандартных командах протокола Hayes):

- Hardware flow control (RTS/CTS) (&K3 в большинстве модемов)
- E1 (Echo ON) в `/usr/src/linux-2.0.27/include/linux/serial.h` (команда требуется для функционирования программы `chat`.)
- Q0 Report result codes (требуется для функционирования программы `chat`.)
- S0=0 Auto Answer OFF (если модем не должен отвечать на телефонные звонки).
- &C1 Carrier Detect ON только после соединения.
- &S0 Data Set Ready (DSR) всегда ON.
- Data Terminal Ready (команда зависит от типа модема).

Информация по адресу `http://www.in.net/info/modems/index.html` содержит примеры конфигураций для многочисленных модемов и может оказаться полезной.

Для выяснения информации о конфигурации модема и ее установки можно использовать соответствующие программы коммуникаций (например, minicom или seyon). Многие модемы выдают данные о своих установках в ответ на запрос AT&V, однако этот вопрос нужно уточнить в инструкции к модему.

Если вы при установке окончательно запутались, то вернуться к установкам производителя можно командой AT&F. Как уже говорилось, в большинстве модемов эти установки включают все, что нужно для PPP, однако в этом следует убедиться.

После того, как выработана строка конфигурирования модема, ее следует записать. Теперь предстоит решить, записать ли эти установки в постоянную память (NV-RAM) модема так, что потом их можно будет вызывать соответствующей AT-командой, или сообщать правильные установки модему в процессе соединения PPP.

Если модем используется только в системе Linux и только для соединения с каким-либо сервером, то проще всего записать конфигурацию в постоянную память (NV-RAM) модема.

Если же модем используется другими программами или операционными системами, то безопаснее будет сообщать установки при каждом соединении, гарантируя, что при каждом обращении к модему он будет находиться в правильной конфигурации. Такой подход имеет также преимущества, что конфигурирующая строка модема не будет потеряна, если вдруг пропадет содержимое постоянной памяти (NV-RAM) модема.

Информация о сервере.

Прежде, чем устанавливать PPP-связь с удаленным сервером, требуется получить о нем следующую информацию (ее может дать системный администратор сервера или служба технической поддержки).

- Телефонный номер сервера (один или несколько). Если модем вашего компьютера установлен в офисной АТС, то нужно знать номер, который выводит во внешнюю телефонную сеть. Обычно это либо цифра 0 или 9.
- Назначает ли сервер IP-адреса *статически* или *динамически*. Если используется статическое назначение IP-адресов, то нужно знать этот адрес для данного пользователя. Если сервер дает сразу несколько разрешенных IP-адресов, то надо знать их все, а также маску подсети (netmask).

  Большинство серверов в интернет используют динамическое назначение IP-адресов. Как указывалось выше, это будет иметь последствия в том, какие услуги можно будет получать от сервера.

  Однако, даже если используется статическое назначение IP-адресов, большинство PPP-серверов никогда (из соображений безопасности) не разрешают клиентам указывать IP-адрес. Поэтому знать эту информацию необходимо.

- IP-адрес DNS-сервера (или DNS-серверов) (Domain Name Server, серверы имен) на PPP-сервере. Таких должно быть по крайней мере два, хотя требуется только один.

  Этот пункт может вызвать проблему. Дело в том, что установка для протокола PPP в системе Microsoft Windows 95 позволяет передавать адрес DNS клиенту в процессе соединения. Поэтому служба технической поддержки сервера может заявить, что клиенту не нужно знать этот адрес.

  Однако при использовании системы Linux этот адрес *требуется*. Реализация протокола PPP в системе Linux не позволяет производить установку IP-адреса DNS динамически во время соединения (и возможно, что никогда не позволит).

- Использует ли сервер протоколы PAP (Password Authentication Protocol) или CHAP (Challenge/Handshake Authentication Protocol). Если да, то в этом случае нужно знать идентификатор (id) и пароль соединения (secret), которые следует использовать при соединении. Возможно, что ими являются соответственно имя пользователя и пароль в операционной системе сервера.
- Запускает ли сервер протокол PPP автоматически, или для этого требуется выдать специальную команду после входа в систему. Если надо ввести команду, то ее нужно знать.
- Является ли операционная система сервера системой Microsoft Windows NT, а если да, то используется ли в ней система MS PAP/CHAP. Многие корпоративные локальные сети используют такие возможности системы MS Windows NT для большей безопасности.

Каждый компьютер, подключенный к сети интернет, должен иметь свой уникальный IP-адрес. Эти адреса назначаются централизованно специальными уполномоченными в

каждой стране. Следовательно, для использования связи посредством протокола PPP нужно, чтобы имелся такой назначенный IP-адрес. Система динамического назначения адресов возникла по причине растущего количества компьютеров в сети Интернет (и отчасти из-за большого числа пользователей PPP). При динамическом назначении IP-адресов, когда адрес назначается при соединении с сервером, при каждом новом соединении могут назначаться различные IP-адреса. Этот метод является наиболее употребительным для большинства провайдеров. В некоторых случаях используется статическое назначение адресов. Выбрать себе IP-адрес нельзя; он присваивается специально уполномоченной на то организацией. Это предотвращает ситуацию, когда два компьютера будут иметь один и тот же IP-адрес, что создаст проблемы в Интернете. Удаленные PPP-серверы сообщают тип присваиваемых IP-адресов и, при динамическом присвоении адресов, фактический адрес, назначенный при соединении.

Важно знать, что при динамическом назначении IP-адресов будет очень сложно обеспечить любые постоянные услуги Интернета, такие, как www-серверы, услуги gopher, серверы Internet Relay Chat. Можно будет использовать такие услуги, расположенные на других компьютерах, но нельзя будет создать подобное на своем компьютере без приложения чрезвычайных усилий. Подобные приемы лежат вне целей данного рассмотрения.

PAP и CHAP это два различных распространенных метода идентификации. Оба эти метода поддерживаются Linux.

Тестирование модема и соединения с удаленным сервером.

Теперь, когда улажены вопросы с последовательным портом и модемом, можно посмотреть, как эти установки будут работать при попытке установить соединение с сервером.

Для этого нужно, используя коммуникационные программы (например, minicom или seyon), задать требуемую для протокола PPP инициализационную строку модема и набрать номер PPP-сервера.

Замечание: на этом этапе мы *не* пытаемся установить соединение PPP, а только проверяем, правильно ли указан телефонный номер сервера, и пытаемся точно определить, какого рода информацию сервер посылает клиенту для входа в систему и установки связи PPP.

Поэтому на данном этапе следует либо захватывать сообщения на экране и сохранять их в файле, либо *очень аккуратно* переписывать все приглашения сервера, которые он выдает.

Если сервер использует протокол PAP, то вы не должны увидеть приглашения, а вместо этого на экран будет выдаваться текстовое представление протокола установления связи (link control protocol), который будет выглядеть бессодержательным (иногда его еще называют "garbage": мусор).

Несколько предупреждений:

- Некоторые серверы позволяют пользователю входить как с помощью текстового имени и пароля, так и через протокол PAP. Так что если известно, что сервер

использует PAP, но при соединении на экран не поступает сразу же бессодержательная выдача, то это не значит, что сделано что-то неправильное.
- Некоторые серверы требуют, чтобы клиент вначале ввел некоторый текст, а потом запускают стандартную последовательность PAP.
- Некоторые PPP-серверы являются пассивными. Это значит, что они ничего не посылают до тех пор, пока клиент не пошлет им некоторый lcp-пакет. При соединении с таким сервером бессодержательная выдача на экране не появится, как впрочем и какая-либо другая.
- Некоторые серверы не запускают протокол PPP пока клиент не нажмет клавишу Enter, так что это всегда стоит попробовать, если соединение произошло, а "мусор" на экране не появился!

Следует пытаться дозвониться до сервера минимум дважды, так как некоторые серверы меняют приглашение со временем. Два основных приглашения, которые должны всегда быть распознаны клиентским компьютером, это:

- приглашение ввести имя пользователя.
- приглашение ввести пароль.

Если для запуска программы PPP на сервере надо еще запустить некоторую команду, то необходимо также выяснить, какое приглашение дает сервер, когда клиент вошел в систему и сервер ждет от него команды на запуск PPP.

Если при входе в систему сервер автоматически запускает PPP, то клиент сразу увидит на экране бессодержательные сообщения. Это сервер посылает клиенту информацию для запуска и конфигурирования соединения PPP.

Выглядит это примерно так:

```
y}#.!}!}!} }8}!}}U}"}\&} } } } }}\& ...}'}"}(}"} .~~y}
```

На некоторых серверах программу PPP требуется явно запустить. Обычно так сделано потому, что конфигурация сервера позволяет входить в систему через протокол PPP и через командную оболочку, используя одну и ту же пару имени пользователя и пароля. В таком случае после входа в систему нужно выдать команду на запуск PPP.

Если непосредственно после соединения (а также, если требуется, после входа в систему и запуска PPP-сервера) на экран не выдается бессодержательное сообщение, можно попробовать запустить PPP-сервер, нажав клавишу Enter.

На этом этапе можно повесить трубку модема (обычно для этого надо быстро ввести с клавиатуры `+++`, и после того, как модем откликнется сообщением `OK`, послать команду ATH0).

Если модем не заработал, следует вернуться к руководству к данному модему, экранной документации к коммуникационным программам и документу Serial HOWTO.

Работа с PPP-серверами с динамическим назначением IP-адресов.

Если сервер динамически назначает IP-адреса клиентов (так и делают большинство Интернет-провайдеров, если не заплатить им существенно большую сумму за подключение), то нужно осознать, какие это накладывает ограничения.

Сразу скажем, что ограничения не коснутся исходящих запросов к ресурсам системы Интернет. Иными словами, можно посылать электронную почту программой sendmail (если она будет правильно сконфигурирована), осуществлять FTP-соединение с удаленными сайтами и копировать с них файлы, пользоваться командой `finger`, бродить по www-страницам и т.п.

В частности, на электронные послания можно отвечать и при отключенном соединении. Почта будет просто храниться в компьютере, а действительно будет отправлена при следующем соединении с сервером.

Ограничения обусловлены тем, что компьютер не подключен к Интернету 24 часа в день, и что при подключении каждый раз ему дается новый IP-адрес. Таким образом, невозможно будет получать почту, направленную непосредственно на данный компьютер, и будет очень трудно организовать www- или FTP-сервер, который остальные могли бы посещать. С точки зрения Интернета, компьютер как единица, постоянно подключенная к нему и имеющая постоянный IP-адрес, не существует (так как тот же IP-адрес будет временами использоваться другими компьютерами).

Если на таком компьютере организован www-сервер (или любой иной сервер), то он будет неизвестен для всех пользователей Интернета кроме тех, кто знает, что компьютер в это время подключен, и к тому же знает его текущий IP-адрес. Есть несколько способов оповестить этих пользователей: от простого телефонного звонка или сообщения по электронной почте этим пользователям и вплоть до хитроумного использования файлов `.plan` при доступе к серверу через командную оболочку (если сервер разрешает доступ к себе через командную оболочку и команду `finger`).

Для большинства пользователей все это не является проблемой, поскольку им нужно лишь отправлять и получать электронную почту (через имя пользователя на сервере) и производить соединение с удаленными www, FTP и другими серверами Интернета. Однако для того, чтобы иметь свой сервер, доступный с других компьютеров, в действительности требуется статический IP-адрес.

Файлы программ PPP.

Для того, чтобы создать каталоги и редактировать файлы, нужные для конфигурирования PPP, надо войти в систему как `root`. Местоположение файлов, обеспечивающих конфигурирование и работу PPP-соединения, различается для версий PPP 2.1.2 и 2.2.

Для версии PPP 2.1.2 это следующие файлы:

```
/usr/sbin/pppd            исполняемый файл PPP.
/usr/sbin/ppp-on          скрипт для набора номера и установления
соединения.
/usr/sbin/ppp-off         скрипт для разрыва соединения.
/etc/ppp/options          опции программы pppd для всех соединений.
/etc/ppp/options.ttyXX    специальные опции для соединения через данный
порт.
```

Для версии PPP 2.2 используются такие файлы:

```
/usr/sbin/pppd                    исполняемый файл PPP
/usr/sbin/ppp-on                  скрипт для набора номера и установления
соединения
/etc/ppp/scripts/ppp-on-dialer    первая часть скрипта для набора номера
/etc/ppp/scripts/ppp-off          собственно скрипт диалога (chat script)
/etc/ppp/options                  опции программы pppd для всех соединений
/etc/ppp/options.ttyXX            специальные опции для соединения через
данный порт
```

Пользователи Red Hat Linux должны иметь в виду, что Red Hat 4.X устанавливает эти скрипты в каталог `/usr/doc/ppp-2.2.0f-2/scripts`.

В каталоге `/etc` должен быть каталог `ppp`:

```
drwxrwxr-x   2 root     root         1024 Oct  9 11:01 ppp
```

Если такого каталога нет, его следует создать с такими принадлежностью и правами доступа.

Если же этот каталог уже существует, то в нем должны находиться шаблоны опций в файле с именем `options.tpl`. Пример этого файла, если такого файла не окажется, приведен ниже.

В этом примере объясняются почти все опции программного обеспечения протокола PPP, поэтому с ним полезно будет ознакомиться (вкупе с экранной документацией к программе `pppd`). Хотя этот файл можно рассматривать как основу для создания собственного файла `/etc/ppp/options`, возможно, что лучше будет создать такой файл самостоятельно, потому что так файл не будет содержать большого числа закомментированных строк, как следствие, он окажется намного короче, и его будет легче читать и поддерживать.

В некоторых дистрибутивах, по-видимому, потерян файл `options.tpl`. Его полная версия должна находиться в документе PPP-HOWTO.

Какие опции нужны?

Набор опций зависит от многих факторов. Приведенный здесь список должен работать с большинством серверов.

Однако, если таким образом протокол PPP не удалось заставить работать, следует *прочесть образец файла* `/etc/ppp/options.tpl` и man-страницу к программе `pppd`. Полезно также проконсультироваться с системным администратором соответствующего сервера, либо обратиться в службу технической поддержки пользователей.

Отметим также, что скрипты для установления соединения, приведенные здесь, используют также некоторые опции программы pppd, задаваемые в командной строке. Это сделано для того, чтобы было легче производить изменения.

```
# ОБРАЗЕЦ ФАЙЛА /etc/ppp/options (нет поддержки PAP/CHAP)
#
# Для того, чтобы предотвратить раздвоение pppd в фоновый режим,
# используется опция -detach
#
# использовать управляющие строки модема
modem
# использовать стиль захвата uucp для того, чтобы обеспечить
#   исключительный доступ к последовательному устройству
lock
# использовать аппаратный контроль потока данных
crtscts
# в таблице маршрутов создать маршрут по умолчанию для
# этого соединения
defaultroute
# НЕ посылать никаких управляющих Esc-последовательностей
asyncmap 0
# использовать максимальный размер передаваемого пакета 552 байта
mtu 552
# использовать максимальный размер принимаемого пакета 552 байта
mru 552
#
# КОНЕЦ ОБРАЗЦА ФАЙЛА /etc/ppp/options (нет поддержки PAP/CHAP)
```

Установка PPP-соединения вручную.

Теперь, когда созданы файлы `/etc/ppp/options` и `/etc/resolv.conf` (и, если необходимо, файл `/etc/ppp/pap|chap-secrets`), все установки можно проверить, попытавшись произвести соединении PPP вручную. После того, как это удастся, можно будет автоматизировать этот процесс.

Для того, чтобы это было можно сделать, нужно, чтобы коммуникационные программы могли завершить свою работу, *не* перезагружая модем. Программа minicom может это сделать при нажатии последовательности клавиш Control-A Q.

- Убедиться, что вы в системе как пользователь `root`.
- Запустить коммуникационные программы (например minicom), набрать номер сервера и войти в систему. Ввести команду для запуска PPP, если это требуется. При этом на экран начинают поступать бессодержательные сообщения (как раньше).
- Если используются PAP или CHAP, то само по себе соединение с удаленным сервером должно запускать программы PPP на удаленном компьютере, и бессодержательные сообщения будут выдаваться без входа в в систему. Для ряда серверов это будет не так; тогда надо нажать клавишу Enter и посмотреть на результат.
- Теперь надо выйти из коммуникационной программы, не перезагружая модем. Теперь на приглашения Linux в качестве пользователя `root` можно ввести команду:
-
-
  - `# pppd -d /dev/ttyS0 38400 &`

- •

  При этом, разумеется, надо подставить имя устройства, к которому подсоединен модем.

  При опции `-d` возможен поиск ошибок: начальные диалоги процесса установления соединения PPP будут протоколироваться в специальных файлах, что будет полезно позднее при отслеживании ошибок.

- При установке связи PPP индикаторы модема должны мигать. Процесс установки связи PPP длится некоторое время.

В данное время можно посмотреть на интерфейс PPP командой:

```
# ifconfig
```

В дополнение к сетевым заглушкам (loopback) и адаптерам Ethernet можно увидеть информацию следующего типа:

```
ppp0     Link encap:Point-Point Protocol
    inet addr:10.144.153.104  P-t-P:10.144.153.51 Mask:255.255.255.0
    UP POINTOPOINT RUNNING  MTU:552  Metric:1
    RX packets:0 errors:0 dropped:0 overruns:0
    TX packets:0 errors:0 dropped:0 overruns:0
```

Здесь:

- inet addr:10.144.153.10 IP-адрес Вашего конца PPP-соединения.
- P-t-P:10.144.153.5 IP-адрес PPP-сервера.

Программа `ifconfig` сообщает не эти IP-адреса, а те, которые используются PPP-сервером. Отметим, что команда `ifconfig` также сообщает, что соединение установлено и работает ("up and running").

Можно также увидеть маршрут к удаленному компьютеру (и далее). Для этого надо ввести команду

```
# route -n
```

Будет выдана информация следующего вида:

```
Kernel routing table
Destination  Gateway      Genmask         Flags MSS  Window Use Iface
10.144.153.3 *            255.255.255.255 UH    1500 0        1 ppp0
127.0.0.0    *            255.0.0.0       U     3584 0       11 lo
10.0.0.0     *            255.0.0.0       U     1500 0       35 eth0
default      10.144.153.3 *               UG    1500 0        5 ppp0
```

Отметим здесь две важные строки, указывающие на IP-адрес PPP-клиента.

Первая является маршрутом HOST (помечена флагом "H"). Она позволяет увидеть компьютер, с которым произведено соединение, но не дальше.

Вторая строка является маршрутом по умолчанию, который установлен посредством опции `defaultroute` программы `pppd`. По этому маршруту отправляются все пакеты, которые имеют пункт назначения, недоступный через локальную сеть Ethernet (т.е. вне локальной сети, для которой есть специальные сетевые маршруты). Таким образом, эти пакеты направляются PPP-серверу, который затем должен отправить их дальше в Интернет, а приходящие (возвратившиеся) пакеты направить (вернуть) клиенту.

Отсутствие таблицы с такими двумя строками говорит о том, что не все установки сделаны верно. В частности, если в протоколе имеется сообщение, что программа `pppd` не замещает существующего маршрута по умолчанию, то значит, что маршрут по умолчанию указывает на интерфейс Ethernet. Это *должно быть* заменено на необходимый нам сетевой маршрут, поскольку можно иметь ТОЛЬКО ОДИН МАРШРУТ по умолчанию.

Надо будет исследовать инициализационные файлы системы и найти, где же устанавливается этот путь по умолчанию. Искать надо команду `route add default...`, и заменить ее на что-либо вроде `route add net...`.

Теперь можно проверить связь, введя команду `ping` с IP-адресом, указанном в выводе команды `ifconfig`:

```
# ping 10.144.153.51
```

В ответ будут приходить строки следующего вида:

```
PING 10.144.153.51 (10.144.153.51): 56 data bytes
64 bytes from 10.144.153.51: icmp_seq=0 ttl=255 time=328.3 ms
64 bytes from 10.144.153.51: icmp_seq=1 ttl=255 time=190.5 ms
64 bytes from 10.144.153.51: icmp_seq=2 ttl=255 time=187.5 ms
64 bytes from 10.144.153.51: icmp_seq=3 ttl=255 time=170.7 ms
```

Эти строчки будут идти бесконечным потоком; чтобы остановить этот поток, надо нажать комбинацию клавиш Control-C. При этом будет получена дополнительная информация:

```
--- 10.144.153.51 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 170.7/219.2/328.3 ms
```

(сообщается количество переданных, принятых и потерянных пакетов, а также минимальное, среднее и максимальное время реакции).

Теперь можно попытаться ввести команду `ping`, указав некоторый компьютер (но не сам PPP-сервер) по имени (желательно знать, что этот сервер в данный момент включен). Например:

```
# ping sunsite.unc.edu
```

Далее должна последовать пауза, во время которой компьютер получает IP-адрес от сервера DNS, указанного в файле `/etc/resolv.conf`. В этот время индикаторы модема должны мигать. Через некоторый промежуток времени будет получено примерно следующее сообщение:

```
PING sunsite.unc.edu (152.2.254.81): 56 data bytes
64 bytes from 152.2.254.81: icmp_seq=0 ttl=254 time=190.1 ms
64 bytes from 152.2.254.81: icmp_seq=1 ttl=254 time=180.6 ms
64 bytes from 152.2.254.81: icmp_seq=2 ttl=254 time=169.8 ms
64 bytes from 152.2.254.81: icmp_seq=3 ttl=254 time=170.6 ms
64 bytes from 152.2.254.81: icmp_seq=4 ttl=254 time=170.6 ms
```

Остановите вывод, нажав Control-C. Будет получена статистика:

```
--- sunsite.unc.edu ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 169.8/176.3/190.1 ms
```

Если отклик от удаленного сервера не был получен, то можно отдать команду `ping`, указав IP-адрес DNS-сервера. Если в этот раз отклик получен, то похоже, что ошибка связана с файлом `/etc/resolv.conf`.

Если же отклик не был получен ни разу, то либо имеется ошибка в составлении маршрутов клиента, либо сервер почему-то не может послать пакеты сообщений, адресованные клиенту. В этом случае нужно проверить таблицу маршрутов, как это показано выше, и если она в порядке, нужно обратиться к Интернет-провайдеру. Хороший тест для сервера -- попытаться установить связь с ним из другой операционной системы. Если из нее удается установить контакт с внешним (по отношению к серверу провайдера) сервером, значит, проблемы нужно искать у клиента.

Если оказалось, что связь работает, ее надо закрыть командой

```
# ppp-off
```

После короткой паузы модем сам разорвет связь.

Если это не сработало, то надо либо выключить модем, либо запустить коммуникационную программу и послать модему прерывание командой `+++`, и затем, после приглашения от модема `OK` разорвать связь командой ATH0.

Может потребоваться также очистить файл захвата (lock file), созданный программой `pppd`, с помощью команды:

```
# rm -f /var/lock/LCK..ttySx
```

6.2.3 Создание скриптов установления связи.

Хотя и в дальнейшем можно устанавливать связь вручную, как это делалось выше, намного удобнее создать несколько скриптов, которые будут делать все операции автоматически. При этом пользователю (из группы, которой доступны команды поддержки PPP, либо пользователю `root`) будет достаточно ввести одну команду, чтобы эти скрипты автоматически установили связь, вошли в систему сервера и запустили программы PPP.

Ниже даются скрипты для случая, когда сервер провайдера *не* использует PAP/CHAP.

Если программное обеспечение PPP установлено правильно, то среди файлов должны присутствовать два файла с примерами. Для версии PPP 2.1.2 эти файлы находятся в каталоге `/usr/sbin`, а для версии PPP 2.2 в каталоге `/etc/ppp/scripts`. Названия файлов:

- для PPP 2.1.2:
-
- `ppp-on`
- `ppp-off`
-
- для PPP 2.2:
-
- `ppp-off`
- `ppp-on`
- `ppp-on-dialer`
-

Если используется версия PPP 2.1.2, то мы советуем удалить файлы с примерами, поскольку их использование чревато проблемами (несмотря на то, что они длительное время использовались и даже рекомендовались в ранних версиях документа PPP-HOWTO).

Для пользователей версии PPP 2.1.2 здесь даются более удачные образцы, взятые из дистрибутива версии PPP 2.2. Их предлагается скопировать и использовать вместо старых.

Скрипт `ppp-on` является первым из пары, которая осуществляет соединение.

```
#!/bin/sh
#
# Это скрипт для инициирования PPP-соединения и первый из
# пары предназначенных для этого скриптов. Они не отвечают
# требованиям безопасности, поскольку коды можно увидеть
# командой ps. Преимущество этих скриптов в их простоте.
#
# Далее идут параметры. Их нужно заменить на параметры
# для конкретного клиентского компьютера.
#
TELEPHONE=555-1212      # Телефонный номер сервера
ACCOUNT=george          # Имя пользователя для входа в
```

```
                        # систему сервера
PASSWORD=gracie         # Пароль
LOCAL_IP=0.0.0.0        # IP-адрес клиента (если известен).
                        # При динамическом назначении
                        # IP-адрес=0.0.0.0
REMOTE_IP=0.0.0.0       # IP-адрес удаленного компьютера
                        # (если нужен). Обычно также 0.0.0.0
NETMASK=255.255.255.0   # Соответствующая маска (если нужна)
#
# Параметры надо экспортировать, чтобы они стали
# доступны скрипту 'ppp-on-dialer'
#
export TELEPHONE ACCOUNT PASSWORD
#
# Здесь указывается местоположение скрипта, который набирает
# номер и входит в систему сервера. Надо указывать абсолютный
# путь, поскольку переменная $PATH при соединении не
# используется. Делать это пользователю root не следует,
# так как это нарушит систему безопасности.
#
DIALER_SCRIPT=/etc/ppp/ppp-on-dialer
#
# Инициируем соединение
exec /usr/sbin/pppd debug /dev/ttySx 38400 \
     $LOCAL_IP:$REMOTE_IP \
     connect $DIALER_SCRIPT
```

Скрипт `ppp-on-dialer` (второй из этой пары):

```
#!/bin/sh
#
# Это вторая часть скрипта ppp-on, предназначенная для
# осуществления протокола соединения
#
/usr/sbin/chat -v                                       \
        TIMEOUT         3                               \
        ABORT           '\nBUSY\r'                      \
        ABORT           '\nNO ANSWER\r'                 \
        ABORT           '\nRINGING\r\n\r\nRINGING\r'    \
        ''              \rAT                            \
        'OK-+++\c-OK'   ATH0                            \
        TIMEOUT         30                              \
        OK              ATDP$TELEPHONE                  \
        CONNECT         ''                              \
        ogin:--ogin:    $ACCOUNT                        \
        assword:        $PASSWORD
```

Для версии PPP 2.2 скрипт `ppp-off` такой:

```
#!/bin/sh
########################################################
#
# Определить устройство, на котором надо прервать связь.
#
if [ "$1" = "" ]; then
   DEVICE=ppp0
else
```

```
   DEVICE=$1
fi

################################################################
#
# Если имеется pid-файл ppp0, то программа работает,
# и ее надо остановить.
if [ -r /var/run/$DEVICE.pid ]; then
   kill -INT `cat /var/run/$DEVICE.pid`

# Если команда kill не сработала, то значит, этому pid
# не соответствует никакой процесс, или что был оставлен
# файл захвата (lock file). Теперь можно удалить это файл.
#
        if [ ! "$?" = "0" ]; then
           rm -f /var/run/$DEVICE.pid
           echo "ERROR: Removed stale pid file"
           exit 1
        fi

# Процесс уничтожен. Теперь пусть программа pppd удалит
# последствия своей деятельности.
#
        echo "PPP link to $DEVICE terminated."
        exit 0
fi
#
# Нет процесса ppp, соответствующего ppp0
echo "ERROR: PPP link is not active on $DEVICE"
exit 1
```

6.2.4 Редактирование скриптов запуска PPP.

Поскольку новые скрипты состоят из двух частей, они будут редактироваться по очереди.

Скрипт `ppp-on`.

Скрипт `ppp-on` надо отредактировать, вставив правильные имена пользователя и пароль на сервере, а также телефонный номер сервера.

Каждая строка скрипта, начинающаяся, например, с `TELEPHONE=`, в действительности устанавливает значение некоторой переменной командной оболочки, причем это значение стоит справа от символа '=' (разумеется, комментарии не учитываются). Редактироваться должны именно эти значения.

Поскольку IP-адрес устанавливается в файле `/etc/ppp/options` (если он требуется), то надо *удалить* следующую строчку:

```
$LOCAL_IP:$REMOTE_IP \
```

Также следует убедиться, что переменная оболочки DIALER_SCRIPT указывает на полный путь к файлу, содержащему именно этот скрипт набора номера, который будет

использоваться. Если файл с исходным скриптом был перенесен на другое место или переименован, проверьте правильность этой строки в скрипте `ppp-on`.

Скрипт `ppp-on-dialer`.

Это второй из скриптов, который устанавливает PPP-соединение.

Замечание: скрипт диалога (chat script) обычно целиком помещается на одной строке. Обратные слэши используются для того, чтобы объединить несколько коротких строк, которые можно выводить сразу на экран, так что командная оболочка будет воспринимать их как одну очень длинную строку. Таким образом, эти символы не являются частью скрипта.

Будет, тем не менее, полезно познакомиться с текстом этого скрипта и понять, что он должен делать.

Скрипт диалога (chat script) является последовательностью пар 'строка ожидания'-'посылаемая строка' ('expect string'-'send string'). Заметим, что любой посылке *всегда* предшествует ожидание некоторого сигнала.

Если некая строка посылается без предварительного ожидания прихода какого-либо сигнала (или наоборот), то надо использовать пустую строку ожидания (или посылаемую строку), обозначаемую как "". Кавычки должны обрамлять сообщение и в том случае, когда оно состоит из нескольких слов (пример: сообщение 'NO CARRIER'), так что скрипт воспринимает его как единую строку.

В приведенном примере диалоговая строка выглядит так:

```
exec /usr/sbin/chat -v
```

При вызове диалогового скрипта использована опция `-v`, указывающая, что все сообщения ввода и вывода будут сохраняться в системном протоколе (обычно это файл `/var/log/messages`). После того, как этот скрипт показал свою работоспособность, эту опцию можно убрать, сократив тем самым объем информации в системных протоколах.

Следующая строка:

```
TIMEOUT         3
```

устанавливает время ожидания для прихода ожидаемых строк, равное трем секундам. Для очень медленного модема это время можно увеличить до 5 или 10 секунд.

Строка:

```
ABORT           '\nBUSY\r'
```

говорит, что при приходе строки 'BUSY' надо прервать работу.

Аналогично нужно поступить, если придет строка 'NO ANSWER':

```
ABORT           '\nNO ANSWER\r'
```

или строка 'RINGING':

```
ABORT           '\nRINGING\r\n\r\nRINGING\r'
```

(кто-то уже звонит по телефону клиента).

Посылаем модему строку 'AT'; в ответ ничего не ожидается:

```
''              \rAT
```

В следующей строке:

```
OK-+++\c-OK   ATH0
```

использованы некоторые возможности восстановления после сообщений об ошибке.

Здесь от модема ожидается строка 'OK'; если она не пришла (из-за того, что модем не находится в командном режиме), то посылается строка '+++' (эта строка возвращает стандартный Hayes-совместимый модем в командный режим) и ожидается 'OK', а затем посылается команда ATH0. Эта команда позволяет скрипту справляться с ситуацией, когда модем завис в режиме on-line.

Следующая строка:

```
TIMEOUT         30
```

устанавливает время ожидания, равное 30 секундам, на весь остаток скрипта. Если это время окажется недостаточным, его можно увеличить до 45 секунд и более.

Строка набора номера:

```
OK              ATDP$TELEPHONE
```

ожидает 'OK' от модема в ответ на команду ATH0, и набирает требуемый номер.

Дальше ждем сообщения 'CONNECT':

```
CONNECT         ''
```

Наш модем также посылает 'CONNECT', когда отвечает удаленный модем.

Теперь надо войти в систему:

```
ogin:--ogin:    $ACCOUNT
```

В эту команду встроен механизм, который, когда время ожидания будет исчерпано, а приглашение войти в систему (...ogin:) получено не будет, посылает пустую строку и еще раз ждет приглашения. Когда приглашение получено, то посылается имя

пользователя, которое хранится в переменной командной оболочки по имени $ACCOUNT.

Теперь ожидается приглашение ввести пароль:

```
  assword:        $PASSWORD
```

и он вводится.

В данном скрипте предусмотрены возможности возникновения различных ошибок. Однако программа `chat` имеет гораздо больше возможностей, чем здесь продемонстрировано. Более полную информацию можно получить на ее man-странице (man 8 chat).

6.2.5 Запуск PPP на сервере.

Приведенный выше скрипт `ppp-on-dialer` хорошо работает для тех серверов, которые после входа в систему автоматически запускают программу `pppd`. Однако некоторые серверы требуют, чтобы эта программа была явным образом запущена. Для этого нужно отредактировать скрипт `ppp-on-dialer`.

В конце скрипта (после строки, где ожидается и вводится пароль) надо ввести еще одну строку. В ней будет ожидаться приглашение командной оболочки сервера. Следует особо обходится с символами, которые имеют особый смысл в командных оболочках, например, для оболочки Bourne shell это символы:

```
$  [  ]
```

После того, как диалоговый скрипт обнаружил приглашение, он должен выдать команду, запускающую программу PPP на сервере.

Сервер, которым пользуется автор этих строк, использует стандартное приглашение оболочки Bash системы Linux:

```
[hartr@kepler hartr]$
```

на что требуется ответить:

```
# ppp
```

и на сервере будет запущен протокол PPP.

Можно предусмотреть различные ситуации следующей строкой в скрипте:

```
hartr--hartr    ppp
```

Это означает, что если приглашение в установленное время не получено, то надо послать пустую строку и еще раз подождать приглашения.

Когда приглашение получено, посылается строка 'ppp'.

К концу предыдущей строки скрипта надо не забыть добавить символ \, так чтобы скрипт рассматривался как одна длинная строка!

К сожалению, список различных приглашений серверов очень велик. Для того, чтобы выяснить устойчивую закономерность и внести ее в ожидаемую строку, надо войти в систему сервера несколько раз с помощью программы minicom.

6.2.6 Если Ваш PPP-сервер использует PAP (Password Authentication Protocol).

Если сервер использует для опознавания клиента протоколы PAP или CHAP, то требуется еще некоторая работа.

В дополнение к уже указанным, в файл /etc/ppp/options нужно добавить следующие строки:

```
#
# указать программе pppd использовать имя пользователя
# на сервере в качестве имени компьютера в процессе
# идентификации
#
name <имя пользователя в системе сервера>  # отредактировать
#                                                # эту строку
#
# Одну из следующих строк надо раскомментировать тем, у кого
# работает PPP-сервер. ВНИМАНИЕ: не надо раскомментировать
# эти строки при соединении в качестве клиента, даже если
# сервер использует PAP или CHAP (поскольку, работая на
# клиентском компьютере, эти строки дадут команду серверу
# использовать для опознавания себя протоколы PAP или CHAP;
# эту команду сервер почти наверняка не сможет выполнить,
# и связь не установится).
#
#+chap
#+pap
#
# Если используются зашифрованные пароли в файле
# /etc/ppp/pap-secrets, то надо раскомментировать следующую
# строку. ВНИМАНИЕ: это НЕ то же самое, что зашифрованный
# пароль в системе Microsoft Windows NT, который может быть
# установлен программой MS RAS
#
#+papcrypt
```

6.2.7 Использование MSCHAP.

Программа RAS в операционной системе Microsoft Windows NT может быть установлена так, что в ней будет использоваться модификация протокола CHAP. Среди исходных текстов программ PPP можно найти файл README.MSCHAP80, в котором обсуждается этот вопрос. Узнать о том, использует ли сервер для опознавания этот протокол, можно, установив опцию поиска ошибок (debugging) программы pppd. Если сервер использует протокол MS CHAP, то на экране будет примерно такая строка:

```
rcvd [LCP ConfReq id=0x2 <asyncmap 0x0> <auth chap 80> <magic 0x46a3>]
```

Индикатором здесь является строка `auth chap 80`.

Для того, чтобы использовать MS CHAP, надо будет заново откомпилировать программу `pppd`, чтобы включить туда поддержку этого протокола. Инструкции для компилирования и использования такой версии программы `pppd` можно также найти в файле `README.MSCHAP80`.

Если используется опознавание протоколом PAP или CHAP, то нужно создать файлы с секретными ключами (secrets file). Их имена будут соответственно `/etc/ppp/pap-secrets` и `/etc/ppp/chap-secrets`, принадлежность пользователю `root`, группе `root` с правами доступа 740.

Первое, что нужно знать о протоколах PAP и CHAP, это что они созданы для опознавания хостов, а не пользователей. Другими словами, после того, как компьютер совершил PPP-соединение с сервером, *любой* пользователь на этом компьютере сможет использовать это соединение.

Протокол PAP может, а протокол CHAP должен потребовать двустороннего опознавания, т.е. имя и ключ потребуются с каждой стороны. Однако большинство PPP-серверов, использующих PAP, этого не требуют.

Предоставляя услуги, провайдер обычно дает клиенту имя и пароль для входа в систему, и не интересуется тем, какое имя компьютера установлено в операционной системе на клиентском компьютере. Поэтому при соединении с сервером надо использовать имя пользователя, данное провайдером. Для этого используется опция `name` программы `pppd`. В файл `/etc/ppp/options` надо добавить строку:

```
name your_user name_at_your_ISP
```

(первый аргумент "локальное" имя пользователя, второй имя на сервере).

С технической точки зрения, на самом деле надо использовать команду `user your_user name_at_your_ISP` для PAP, однако программа `pppd` может использовать это имя в качестве имени пользователя, если оно требуется для PAP. Преимущество использования опции `name` в том, что она применима и к протоколу CHAP.

Поскольку протокол PAP предназначен для опознавания компьютеров, с технической точки зрения требуется также указать имя удаленного компьютера. Однако, поскольку в большинстве случаев используется один сервер, в файле с секретным ключом можно использовать для имени удаленного компьютера шаблон "`*`".

Файл `/etc/ppp/pap-secrets` устроен следующим образом:

```
# Секретные ключи для опознавания протоколом PAP
# клиент   сервер   ключ   приемлемые_IP-адреса_клиента
```

Четыре поля в файле разделены пробелами; последнее поле может быть пустым (для динамического и, возможно, статического назначения IP-адреса сервером).

Пусть провайдер предоставил имя пользователя `fred` и пароль `flintstone`. Значение fred для опции `name` будет занесено в файл `/etc/ppp/options`, а файл `/etc/ppp/pap-secrets` будет выглядеть так:

```
# Секретные ключи для опознавания протоколом PAP
# клиент     сервер    ключ     приемлемые_IP-адреса_клиента
fred         *         flintstone
```

В данном файле указывается, что клиентский компьютер по имени `fred` (это имя будет использоваться программой `pppd`, хотя настоящее имя компьютера может быть иным) с *каждым* сервером будет использовать пароль (секретный ключ) `flintstone`.

Заметим, что указывать IP-адрес клиента не требуется, если только сервер не *требует*, чтобы статический IP-адрес клиента обязательно указывался. Иначе, даже если адрес будет указан, для большинства PPP-серверов это не будет работать, так как они (по соображениям безопасности) не позволяют клиентам указывает те IP-адреса, которые им будут даны.

Процедура требует, чтобы опознавание было двусторонним, т.е. чтобы не только сервер опознавал клиента, но и клиент опознавал сервер.

Допустим, что имя клиентского компьютера `fred`, а имя сервера `barney`. Тогда клиент в файле `/etc/ppp/options.ttySx` должен указать значение опций `name` и `remotename` соответственно `fred` и `barney`, а сервер наоборот: `barney` и `fred`.

На компьютере `fred` файл `/etc/chap-secrets` будет выглядеть так:

```
# Секретные ключи для опознавания протоколом PAP
# клиент     сервер    ключ     приемлемые_IP-адреса_клиента
fred         barney    flintstone
barney       fred      wilma
```

а на компьютере `barney`:

```
# Секретные ключи для опознавания протоколом PAP
# клиент        сервер    ключ         приемлемые_IP-адреса_клиента
barney          fred      flintstone
fred            barney    wilma
```

Заметим в частности, что для двустороннего опознавания эти файлы для обоих компьютеров должны иметь строки, соответствующие опознаванию клиентом сервера и сервером клиента.

Скрипт для связи с опознаванием по PAP/CHAP.

Если сервер провайдера использует PAP/CHAP, то диалоговый скрипт выглядит гораздо проще. Все, что должен делать этот скрипт, это набрать номер, подождать соединения и передать управление для входа в систему программе pppd.

```
#!/bin/sh
#
# Это 2 часть скрипта ppp-on. Она осуществляет протокол
# соединения.
#
exec /usr/sbin/chat -v                                  \
        TIMEOUT         3                               \
        ABORT           '\nBUSY\r'                      \
        ABORT           '\nNO ANSWER\r'                 \
        ABORT           '\nRINGING\r\n\r\nRINGING\r'    \
        ''              \rAT                            \
        'OK-+++\c-OK'   ATH0                            \
        TIMEOUT         30                              \
        OK              ATDP$TELEPHONE                  \
        CONNECT         ''                              \
```

Как и раньше, протоколирование сообщений можно включить опцией -d программы pppd. Это эквивалентно опции 'debug'.

В течение некоторого времени, устанавливая соединение с помощью нового скрипта, нужно использовать протоколирование. Предупреждение: если свободного места на диске немного и файл протокола будет быстро расти, могут возникнуть проблемы, но для этого нужно, чтобы в течение многих минут повторялись неудачные попытки установить связь.

Когда скрипт несколько раз подтвердит свою работоспособность, протоколирование можно будет отменить.

```
exec /usr/sbin/pppd debug file options.myserver /dev/ttyS0 38400 \
```

Проверка работы скрипта.

Для проверки работы откройте новый Xterm как root (если используется X Window System), или откройте новую виртуальную консоль. В этом новом сеансе надо ввести команду:

```
# tail -f /var/log/messages
```

Во многих системах протокол помещается именно в файл /var/log/messages. Если в данной системе для этого используется файл с другим именем, то команду, приведенную выше, надо соответственно изменить.

В первом окне (или с первой виртуальной консоли) надо ввести команду:

```
# ppp-on &
```

(если имя отредактированного скрипта `/usr/sbin/ppp-on` не было изменено).

Если этот скрипт не запускается в фоновом режиме путем приписывания символа & в конце команды, то приглашение консоли (терминала) пропадет до тех пор, пока программы PPP не закончат работу (до завершения связи).

После этого можно переключиться в окно (консоль), в котором будет вестись наблюдение за протоколом системы (system log).

6.2.8 Завершение PPP-соединения.

После того, как работа с PPP-связью завершена, ее надо завершить стандартной командой `ppp-off` (это может сделать пользователь `root` или пользователь группы `ppp`).

6.2.9 Наиболее распространенные проблемы с соединением.

Одна проблема может заключаться в том, что многие провайдеры поддерживают только те пакеты программного обеспечения, которые они предоставляют новым пользователям сервера. Обычно эти программы предназначены для операционной системы Microsoft Windows (и многие службы технической поддержки ничего не знают о системах UNIX и Linux). Так что можно рассчитывать на весьма ограниченную помощь провайдера.

Можно, разумеется, оказать частную услугу работникам технической поддержки, познакомив их с системой Linux (они могут иметь начальное представление о системе UNIX из работы в Интернете), которую они могут установить на собственных домашних компьютерах.

Проблемы с адресами.

Итак, PPP-соединение установлено и работает, и можно получить отклик от PPP-сервера командой `ping`, указав в качестве IP-адреса второй (удаленный) IP-адрес, выданный командой `ifconfig ppp0`.

Прежде всего, следует попытаться указать команде `ping` те IP-адреса, которые указаны в файле `/etc/resolv.conf` IP-адресами серверов имен. Если серверы откликаются и их IP-адреса отличны от IP-адреса сервера, то должна иметься возможность установить связь через сервер с внешним миром. Попробуйте задать в качестве аргумента команды `ping` *полное имя* сервера провайдера:

```
ping my.isp.net
```

(в эту команду надо подставить нужное имя). Если это не сработало, то имеются проблемы с разрешением имен. Возможно, имеется опечатка в файле `/etc/resolv.conf`, и его нужно тщательно сверить с образцом, приведенном в разделе 6.1.1.

Если файл проверен, а связь не устанавливается (и при этом провайдер подтверждает, что сервер работает нормально), то следует проверить общую конфигурацию системы Linux, обращая особое внимание на права доступа к файлам.

Если на команду `ping` не дают отклики серверы имен провайдера, то они могут быть выключены (об этом можно узнать по телефону), или имеются проблемы с маршрутами на сервере.

Одной из возможностей может быть та, когда на сервере стоит система Linux, где среди опций ядра системы не указано перенаправление сообщений в Интернет-протоколах (IP forwarding)!

Поиск ошибок при неудачной попытке установить соединение.

Причин неудачи при соединении может быть много (ошибки в работе диалогового скрипта, сильные шумы в телефонных линиях и т.п.). Поэтому при неудачах надо исследовать системные протоколы.

Весьма распространена ситуация, когда ядро системы Linux откомпилировано и в него включены функции поддержки PPP, но при запуске программы `pppd` ядро выдает сообщение, что поддержки PPP нет. Для этого имеется целый набор возможных причин.

- Загружено не новое, а старое ядро без поддержки PPP.
- Модули поддержки PPP откомпилированы, но не установлены.
- Ожидается, что модули будут загружаться автоматически, однако это не так.
- Используется не та версия PPP, которая нужна для данной версии ядра.
- `pppd` запускается *не* пользователем `root`.
- Имеется опечатка в стартовых скриптах.
- Вход в систему сервера осуществляется неправильно.
- Не запускается протокол PPP на сервере.
- Процесс PPP на сервере начинается слишком медленно.
- Не установлен маршрут по умолчанию.

Наилучшим средством решения проблем будет обратиться к документу PPP FAQ (который действительно представляет собой набор вопросов и ответов). Это весьма полезный документ, и там наверняка есть ответы на все важные вопросы. Скорее всего, если там нет ответа на конкретный вопрос, то проблема не связана с протоколом PPP!

Ничего не получается...

Если ничто не помогает наладить PPP-соединение, следует вернуться к началу этой главы и проверить все с начала и в соответствии с выдачей в файлах системных протоколов, созданных командами `chat -v` и `pppd -d`.

Следует также обратиться к документации по протоколу PPP, к документу PPP FAQ, а также к другим указанным в этом разделе источникам!

Если проблема остается, можно обратиться за помощью в телеконференции `comp.os.linux.misc` и `comp.os.linux.networking`, а также в `comp.protocols.ppp`. Эти телеконференции часто посещают специалисты, которые могут оказать помощь.

Если принято решение искать помощь в системе Usenet, не следует посылать очень длинные сообщения, содержащие системные протоколы, поскольку это перегружает сеть. Более продуктивно будет описать проблему словами и, возможно, включить несколько строк выдачи (объемом не более одного экрана).

О документе...

Установка Linux и первые шаги (Linux Installation and Getting Started)

Документ сгенерирован транслятором **LaTeX**2HTML, версия 96.1-h (Сентябрь 30, 1996)

Командная строка запуска: **latex2html** `-split 1 -show_section_numbers -dir html gs.tex`.

Трансляция выполнена Clarica Wed Mar 4 10:46:42 PST 1998
Некоторые конструкции добавлены вручную.