Впервые опубликовано на developerWorks 09.09.2005

Учебник для экзамена LPI 101: Установка Linux и управление пакетами

Администрирование для начинающих (LPIC-1) тема 102

Схема жесткого диска

Обзор файловых систем

Файловая система Linux состоит из файлов, которые размещаются на диске или другом блочном устройстве хранения в каталогах. Также как во многих других системах, каталоги в Linux могут содержать другие каталоги, которые называют подкаталогами. В отличие от такой системы как Microsoft® Windows® с концепцией отдельных файловых систем для каждой буквы диска (A:, C: и так далее.), файловая система Linux является одним деревом с каталогом / в качестве корневого каталога (root).

Вы можете удивиться почему же схема жесткого диска важна, если файловая система это одно большое дерево?! Дело в том, что каждое блочное устройство, такое как раздел жесткого диска, CD-ROM или дискета на самом деле уже имеют свою файловую систему. Вы создаете отдельную ветку файловой системы путем монтирования (mounting) файловых систем других устройств в точку дерева, называемой точкой монтирования (mount point).

Обычно вы начинаете процесс монтирования монтированием файловой системы одного из разделов жесткого диска как /. Вы можете смонтировать другие разделы жесткого диска как /boot, /tmp или /home. Например, вы можете монтировать файловую систему дискеты (floppy-диска) как /mnt/floppy, и файловую систему CD-ROM как /media/cdrom1. Вы можете также монтировать файлы других компьютеров, используя сетевые файловые системы, такие как NFS. Существуют и другие варианты монтирования файловых систем, но уже перечисленных достаточно, чтобы понять идею этого процесса. Для описания процесса монтирования файловой системы некоторого устройства обычно просто говорят: "монтируется устройство", что следует понимать как "монтирование файловой системы устройства".
Итак, предположим вы только что смонтировали корневую файловую систему (/) и хотите смонтировать IDE CD-ROM, /dev/hdd, в точку монтирования /media/cdrom. Точка монтирования должна существовать до того, как вы смонтируете в нее CD-ROM. Когда вы монтируете CD-ROM, файлы и подкаталоги CD-ROM становятся файлами и подкаталогами внутри /media/cdrom. Любые файлы и подкаталоги, что уже имелись в /media/cdrom становятся не видны, хотя они все еще существуют на блочном устройстве, содержащем точку монтирования /media/cdrom. Как только CD-ROM будет размонтирован, все оригинальные файлы и каталоги вновь станут видны. Вам следует избегать проблем, связанных с размещением других

файлов в каталоге, который собираетесь использовать в качестве точки монтирования.

В Таблице 1 показаны каталоги, которые должны быть в / согласно Filesystem Hierarchy Standard [Стандарту иерархии файловых систем] (более подробно об FHS, смотри Ресурсы).

Таблица 1. Каталоги FHS в /

Каталог	Описание
bin	Бинарные файлы важных программ
boot	Статические файлы загрузчика
dev	Файлы устройств
etc	Настройки этой системы
lib	Важные общие библиотеки и модули ядра
media	Точка монтирования для временных мультимедийных устройств
mnt	Точка монтирования для временного монтирования файловых систем
opt	Дополнительные пакеты для программного обеспечения
sbin	Важные бинарные файлы системы
srv	Данные служб, запущенных на компьютере
tmp	Временные файлы
usr	Вторичная иерархия
var	Часто изменяемые данные

Разделы

Первый учебник этой серии, "Учебник для экзамена LPI 101 (тема 101): Аппаратное обеспечение и архитектура" слегка затронул разделы жестких дисков, и теперь мы собираемся разобраться в них более детально.

Первый жесткий диск IDE в системе Linux это /dev/hda, а первый SCSI диск это /dev/sda. Жесткий диск разбит на сектора по 512 байт. Все сектора на пластине жесткого диска, которые могут быть прочитаны без перемещения головки, называются трэками [Прим.пер.: в русскоязычной литературе также встречается термин "дорожка"]. Диски обычно имеют более одной пластины. Набор дорожек разных пластин, которые могут быть прочитаны без перемещения головок называется цилиндром. Геометрия жесткого диска выражается в цилиндрах, дорожках (или головках) на цилиндр и секторах на дорожке.

Ограничения на возможные значения каждой из этих величин, использовавшиеся операционной системой, привели к тому, что указанные в BIOS параметры геометрии диска пришлось преобразовывать, чтобы появилась возможность работы с большими дисками. В конце концов и этих методов стало не достаточно. Большинство последних разрабатываемых технологий жестких дисков могут использоваться только с логической адресации блоков (LBA -- logical block addressing) , так что физические единицы геометрии CHS все менее важны и отображаемая геометрия может быть не совсем верна или вообще не иметь связи со структурой современных дисков. Диски больших размеров, которые используются сегодня, работают с расширением LBA, известным как LBA48 и отличающимся тем что на нумерацию секторов резервируется до 48 бит.

Пространство жесткого диска разбито (или разделено) на разделы (partition). Разделы не могут перекрываться; пространство не входящее ни в один раздел называется свободным пространством (free space). Разделы имеют имена /dev/hda1, /dev/hda2, /dev/hda3, /dev/sda1 и тому подобные. IDE диски имеют ограничение в 63 раздела, а SCSI диски до 15. Разделы обычно содержат целое число цилиндров (что связано с возможностью ошибочной ссылки на цилиндр).

Если две различные программы для разбиения по разному понимают номинальную геометрию диска, то одна из программ может сообщать об ошибке или иметь проблемы с разделами, созданными другой программой для разбиения. Вы также можете встретить проблемы такого рода если диск был перемещен из одной системы в другую, особенно если возможности BIOS различны. В Linux вы можете узнать номинальную геометрию, просмотрев соответствующий файл в файловой системе /proc, например, /proc/ide/hda/geometry. Эта геометрия используется такими инструментами разбиения диска как fdisk и parted. Листинг 1 показывает использование команды cat для отображения /proc/ide/hda/geometry, а следом идет информационное сообщение, полученное при использовании инструмента разбиения parted.

Листинг 1. Геометрия жесткого диска

```
[root@lyrebird root]# cat /proc/ide/hda/geometry
physical     19457/255/63
logical      19457/255/63
[root@lyrebird root]# parted /dev/hda
GNU Parted 1.6.3
Copyright (C) 1998, 1999, 2000, 2001, 2002 Free Software Foundation,
Inc.
This program is free software, covered by the GNU General Public
License.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY
WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS FOR A
PARTICULAR PURPOSE.  See the GNU General Public License for more
details.

Using /dev/hda
Information: The operating system thinks the geometry on /dev/hda is
19457/255/63.  Therefore, cylinder 1024 ends at 8032.499M.
(parted)
```

Заметьте, что в Листинге 1, parted вычислила номинальную позицию конца 1024 цилиндра. Цилиндр 1024 очень важен для тех систем, где BIOS может загружаться только с раздела, который размещается в первых 1024 цилиндрах диска. Наиболее вероятно, что это произойдет с BIOS, не имеющей поддержки LBA. На современных машинах это уже не проблема, хотя вам следует быть осторожными, поскольку это ограничение может существовать.

Существует три вида разделов: primary (основной), logical (логический), и extended (расширенный). The partition table (Таблица разделов) расположена в главной загрузочной записи (master boot record -- MBR) диска. MBR это первый сектор

диска, так что таблица разделов не очень большая его часть. Это ограничивает количество основных разделов числом 4. Когда требуется более четырех разделов, а это бывает часто, один из основных разделов должен быть определен как расширенный. Диск может содержать только один расширенный раздел.

Расширенный раздел это не более чем контейнер для логических разделов. Эта схема разбиения изначально использовалась в MS DOS и PC DOS, и позволяет использовать диски ПК в DOS, Windows или Linux системах.

В Linux может быть от 1 до 4 основных и расширенных разделов, то есть dev hda может иметь четыре основных раздела: /dev/hda1, /dev/hda2, /dev/hda3 и /dev/hda4. Или оно может иметь один основной раздел /dev/hda1 и один расширенный /dev/hda2. Если определены логические разделы, то их нумерация начинается с 5, то есть первый логический раздел на /dev/hda будет нумероваться /dev/hda5, даже если на диске основного раздела нет, а есть только расширенный (/dev/hda1).

На Листинге 2 приведен вывод команды parted с ключом p, отображающий информацию о разделах для того же диска, что и в Листинге 1. Заметьте, что эта система содержит несколько различных файловых систем Windows и Linux.

Листинг 2. Отображение таблицы разделов при помощи parted

```
(parted) p
Disk geometry for /dev/hda: 0.000-152627.835 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0.031  16300.327  primary   ntfs        boot
2      16300.327  25846.765  primary   fat32       lba
3      25846.765  26842.983  primary   ext3
4      26842.983 152625.344  extended              lba
5      26843.014  28898.173  logical   linux-swap
6      28898.205  48900.981  logical   ext3
7      48901.012  59655.432  logical   ext3
8      59655.463  75657.678  logical   ext3
9      75657.709  95001.569  logical   ext3        boot
10     95001.601 122997.656  logical   reiserfs
11    122997.687 152625.344  logical   ext3
```

Распределение места на диске

Как указывалось ранее, файловая система Linux имеет единственное большое дерево с корнем /. Из этого очевидно почему данные на дискете или CD-ROM должны быть смонтированы, но возможно из этого не совсем ясно зачем создавать различные разделы на жестком диске. Некоторыми вескими причинами разделения файловых систем могут являться:

- Файлы загрузки. Некоторые файлы должны быть доступны для BIOS или загрузчика во время загрузки.

- Несколько жестких дисков. Обычно каждый жесткий диск может быть разбит на один или несколько разделов, каждый со своей файловой системой,

которая должна быть смонтирована куда-нибудь в дереве файловой системы.

- Файлы совместного доступа. Некоторые системы могут подразумевать совместное использование статических файлов, таких как исполняемые файлы программ. Динамические файлы, такие как домашний каталог пользователя или файлы почтовой очереди также могут использоваться совместно, чтобы пользователи могли осуществлять вход на любой машине из нескольких, находящихся в сети и работать со своим домашним каталогом и почтовой системой.

- Возможность переполнения. Если заполненность файловой системы приближается к 100 процентам, то обычно хорошей мыслью будет хранить файлы, необходимых системе для функционирования на отдельном разделе.

- Квоты. Квоты, ограничивающие доступное пространство в файловой системе для пользователя или группы.

- Монтирование только для чтения. До изобретения журналируемых файловых систем восстановление файловых систем после системного сбоя часто занимало много времени. Поэтому редко изменяющиеся файловые системы (такие как каталоги исполняемых программ) могут быть смонтированы в режиме только для чтения, чтобы не тратить много времени на их проверку после системного сбоя.

В добавок к используемым файловым системам вы также должны выделить место на диске для раздела подкачки (swap раздела). В системе Linux под него отводится обычно один или, возможно, несколько разделов.

Определение параметров

Предположим, вы настраиваете систему, в которой есть по крайней мере один жесткий диск, и вы хотите грузиться с этого жесткого диска. (В этом учебнике не рассматривается настройка бездисковой рабочей станции, которая загружается по сети, а также не принимается во внимание использование Linux LiveCD или DVD). Несмотря на то, что размер раздела можно изменить позднее, это обычно требует некоторых усилий, поэтому важно сделать правильный выбор в начале. Итак, начнем.

Во первых вы должны быть уверены, что система будет загружаться. Некоторые старые компьютеры имеют ограничение при котором BIOS может осуществить загрузку только с раздела, который полностью расположен в первых 1024 цилиндрах диска. Если у вас подобная машина, то вы должны создать раздел, который затем будет монтироваться как /boot и хранить ключевые файлы, необходимые для загрузки системы. После загрузки, система Linux возьмет работу с диском на себя и ограничение в 1024 цилиндра больше не будет влиять на работу системы. Если вам необходимо создать раздел /boot, то обычно достаточно около 100МБ.

Следующее с чем вы должны определиться это размер swap-раздела. При сегодняшних параметрах оперативной памяти, swap представляет собой

вторичную более медленную память. Общепринято было создавать swap раздел соразмерным имеющейся оперативной памяти. В настоящее время вы можете выделить под него 500МБ для рабочей станции и около 1ГБ для сервера. Если особые обстоятельства требуют этого, то вы можете увеличить его размер, однако если вы сделает это, то ваша система может потерять в производительности, так что лучше следует увеличить реальную оперативную память. Можно использовать и файл подкачки, однако выделение отдельного раздела предпочтительнее.

Теперь мы подошли к критической точке. Требования для персональной рабочей станции менее предсказуемы, чем для сервера. Я рекомендую всем (и в особенности новичкам) размещать большинство стандартных каталогов (/usr, /opt, /var, /etc) на одном большом разделе. Это особенно полезно для пользователей, впервые устанавливающих Linux и не имеющих ясного представления, что будет получено в конечном итоге. Рабочая станция с графическим рабочим столом и разумным набором средств разработки может потребовать от 2 до 3 гигабайт плюс место, необходимое пользователю. Но некоторые более массивные инструменты разработки могут потребовать несколько гигабайт каждый. Я обычно выделяю где-то от 10 до 20 ГБ на одну операционную систему, а остальное оставляю свободным для загрузки другого дистрибутива.

Рабочая нагрузка сервера более стабильна, но и нехватка места на файловой системе может привести к большим неприятностям. Так что для них я советую создавать несколько разделов, распределённых по нескольким дискам с использованием аппаратного или программный RAID или же группы логических томов.

Вам также может потребоваться определить загрузку каждой файловой системы и будет ли файловая система доступна для нескольких операционных систем или будет использоваться только одной. Вы можете использовать ваш опыт, инструменты планирования размера, а также предположения о перспективах роста, чтобы определить наилучшее распределение дискового пространства для вашей системы.

Вне зависимости от того, что вы настраиваете -- рабочую станцию или сервер -- у вас будут некоторые файлы, уникальные для каждой операционной системы, расположенной на локальном диске. Обычно это /etc для системных параметров, /boot для файлов, необходимых во время загрузки, /sbin для файлов, необходимых для загрузки или восстановления системы, /root для домашнего каталога суперпользователя, /var/lock для lock файлов, /var/run для информации работающей системы и /var/log для файлов журналов (log-файлов) этой системы. Другие файловые системы, такие как /home для домашних каталогов пользователей, /usr, /opt, /var/mail, /var/spool/news могут располагаться на отдельных разделах или смонтированы по сети в соответствии с вашими нуждами и предпочтениями.

Менеджеры загрузки

Обзор процесса загрузки

Перед тем, как углубиться в LILO и GRUB, давайте рассмотрим как ПК начинает работу или загружается. Код, называемый BIOS (от Basic Input Output Service -- Базовая служба ввода/вывода) хранится в энергонезависимой памяти, такой как ROM, EEPROM или flash-памяти. Когда ПК включается или перезагружается, запускается этот код. Обычно он выполняет тестирование при включении питания (power-on self test -- POST) для проверки машины. В конце он загружает первый сектор главной загрузочной записи (MBR) загрузочного диска.

Как обсуждалось в предыдущем параграфе Разделы, MBR также содержит таблицу разделов, так что объём выполняемого кода в MBR меньше чем 512 байт, так что он не может содержать много инструкций. Заметим, что каждый диск, даже дискета (floppy), содержит исполняемый код в своем MBR, даже если код просто выводит сообщение вроде "Non-bootable disk in drive A:" ("В дисководе A: не загрузочный диск"). Этот код загружается BIOS из первого сектора, называемого первичный загрузчик (first stage boot loader) или загрузчик первой стадии (stage 1 boot loader).

Стандартный загрузчик в MBR жесткого диска, который использовался операционными системами MS DOS, PC DOS и Windows, проверяет таблицу разделов для определения основного раздела на загрузочном диске, помеченного как активный(active), загружает первый сектор этого раздела и передает управление в начало загруженного кода. Этот новый кусок кода также известен как загрузочная запись раздела (partition boot record). Загрузочная запись раздела тоже является загрузчиком первой стадии, но он уже достаточно интеллектуален, чтобы загрузить несколько блоков раздела. В этих блоках располагается код загрузчика второй стадии (stage 2 boot loader). В MS-DOS и PC-DOS загрузчик второй стадии непосредственно переходит к загрузке оставшейся части операционной системы. Таким образом, операционная система при загрузке проходит несколько вспомогательных шагов, пока не перейдет в рабочее состояние.

Описанная схема работает прекрасно для компьютеров с одной операционной системой. Но что произойдет, если вам понадобится несколько операционных систем. Скажем Windows 98, Windows XP и три различных дистрибутива Linux? Вы могли бы при помощи некой программой (такой как DOS FDISK) сменить активность разделов и перезагрузиться. Но это утомительно. Кроме того, диск может иметь только четыре основных раздела, а стандартный MBR загрузчик умеет выполнять загрузку только с основного раздела. Но приведенный пример подразумевает пять операционных систем, каждой из которых нужен раздел!

Решение состоит в том, чтобы использовать некоторый специальный код, который позволяет пользователю выбрать систему для загрузки. Приведем примеры:

1. Loadlin, исполняемая DOS-программа запускаемая в работающей системе DOS для загрузки с Linux раздела. Она была популярна, когда настройка множественной загрузки была сложным и рискованным делом.
2. OS/2 Boot Manager -- программа, устанавливавшаяся в небольшой специальный раздел. Раздел помечался как активный и стандартный MBR процесса загрузки запускал Менеджер загрузки (Boot Manager), который выводил меню, позволявшее пользователю выбрать систему для загрузки.
3. Умный менеджер загрузки, программа, которая может размещаться в разделе операционной системы и вызываемая или загрузочной записью активного раздела или основной загрузочной записью Примеры:
    - BootMagic™, часть Norton PartitionMagic™
    - LILO, the LInux LOader (Загрузчик Linux)
    - GRUB, the GRand Unified Boot loader (Главный Унифицированный Загрузчик)

Очевидно, что если вы можете передать управление программе содержащей более 512 байт для выполнения своих задач, то не трудно обеспечить загрузку с логических разделов или загрузку с разделов, находящихся не на загрузочном диске. Все эти решения предоставляют эти возможности или путем загрузки загрузочной записи с любого раздела или потому, что знают какой файл или файлы следует загрузить, чтобы начать процесс загрузки.

Далее мы сосредоточимся на LILO и GRUB, поскольку это загрузчики, включенные во многие дистрибутивы Linux. В процессе установки вашего дистрибутива вам, вероятно, было предложено установить один из них на выбор. Оба могут работать с большинством современных дисков. Запомните, что технологии жестких дисков развиваются стремительно, поэтому вы всегда должны проверить и убедиться, что выбранный вами загрузчик, а также дистрибутив Linux и ваша BIOS будут работать с новеньким сияющим диском. Не соблюдение этого условия может привести к потере данных.

Загрузчик второй стадии, используемый LILO и GRUB позволяет вам выбрать какую из имеющихся операционных систем или их версий следует загрузить. Однако LILO и GRUB значительно различаются в том, что изменение системы требует выполнения некой команды для создания заново настроек загрузки LILO при обновлении ядра или выполнении других подобных операций, тогда как для GRUB это можно сделать отредактировав текстовый файл настроек. LILO существует давно. GRUB новее. Оригинальный GRUB теперь стал GRUB Legacy, а GRUB 2 разрабатывается под патронажем Free Software Foundation (смотри Ресурсы).

LILO

LILO, или LInux LOader (Загрузчик Linux), один из наиболее распространенных загрузчиков Linux. LILO может быть установлен в MBR вашего загрузочного жесткого диска или в загрузочную запись раздела. Он также может быть установлен на сменных носителях, таких как дискеты (floppy-диски), CD или USB диски. Если вы еще не знакомы с LILO, то хорошей мыслью будет попрактиковаться на дискете или USB диске, что мы и будем делать в наших примерах.

При установке Linux вы обычно указываете в качестве менеджера загрузки LILO или GRUB. Если вы выбираете GRUB, то скорее всего LILO не будет установлен автоматически. В этом случае вам потребуется установить пакет LILO вручную. Если вам необходима помощь для этого, то смотрите раздел этого учебника об управлении пакетами ниже. В дальнейшем мы предполагаем, что LILO уже установлен.

Основная функция команды lilo, расположенной в /sbin/lilo, запись загрузчика первой стадии и создание файла распределения памяти (/boot/map), используя данные конфигурации, которые обычно располагаются в /etc/lilo.conf. У неё есть несколько дополнительных назначений, которые мы опишем позднее. Для начала давайте посмотрим на типичный файл конфигурации LILO, который может быть использован при двойной загрузке Windows и Linux.

Листинг 3. Пример /etc/lilo.conf

```
prompt
timeout=50
compact
default=linux
boot=/dev/fd0
map=/boot/map
install=/boot/boot.b
message=/boot/message
lba32
password=mypassword
restricted

image=/boot/vmlinuz-2.4.21-32.0.1.EL
        label=linux
        initrd=/boot/initrd-2.4.21-32.0.1.EL.img
        read-only
        append="hdd=ide-scsi root=LABEL=RHEL3"

other=/dev/hda1
        loader=/boot/chain.b
        label=WIN-XP
```

Первый набор опций, приведенных выше -- это глобальные опции, управляющие работой LILO. Второй и третий представляют опции для каждого образа двух операционных систем, которые мы хотим загружать через LILO, в данном примере Red Hat Enterprise Linux 3 и Windows XP.

Глобальные опции в нашем примере это:

prompt
: вызывает вывод сообщения загрузки.

timeout
: указывает в десятых долях секунды задержку перед автоматической загрузкой системы по умолчанию. В нашем примере timeout=50, что эквивалентно задержке в 5 секунд.

compact
: пытается объединить запросы на чтение для смежных секторов. Это ускоряет загрузку и уменьшает размер файла распределения памяти.

default

указывает какая операционная система будет грузиться по умолчанию. Если не указано, то грузится первая по списку. В нашем примере если пользователь не выберете что-либо в течении 5 секунд, то будет загружена система Linux.

boot

указывает куда будет установлен LILO. В нашем примере это дискета (floppy диск) /dev/fd0. Для установки в MBR первого жесткого диска укажите boot=/dev/hda. Наша система RHEL 3 в действительности располагается на /dev/hda11, поэтому мы должны указать boot=/dev/hda11 если хотим установить LILO в этот раздел. Если этот параметр опущен, то LILO попытается использовать загрузочный сектор устройства, смонтированного в данное время как root (/).

map

указывает расположение файла распределения памяти, импользуемый LILO, для вывода сообщений пользователю и загрузки операционных систем указанных в секциях image файла lilo.conf. По умолчанию это /boot/map

install

указывает новый файл для установки в качестве загрузочного сектора. По умолчанию устанавливается /boot/boot.b, являющийся частью пакета LILO.

message

определяет сообщение, появляющееся перед приглашением загрузки. Оно должно быть менее 65535 байт. Если ваша система отображает графическую оболочку меню LILO, то вы обнаружите, что /boot/message содержит файл картинки. На некоторых системах Red Hat это будет файл 300x200 пикселов в формате PCX. Для систем SUSE это может быть 16 цветный bitmap-файл размером 640x480 пикселов. В этом случае вы также можете обнаружить несколько дополнительных параметров. Посмотрите документацию, идущую с вашей системой. Например, моя система SUSE SLES9 хранит его в /usr/share/doc/packages/lilo/README.bitmaps.

lba32

указывает, что LILO следует использовать для дисков режим LBA32 вместо CHS или линейной адресации секторов.

password

определяет пароль, который следует ввести перед загрузкой образа. Заметим, что это обычный текст, поэтому следует установить такие атрибуты файла /etc/lilo.conf, чтобы запретить просмотр этого файла всем пользователям кроме root. Он не должен совпадать с паролем суперпользователя (root). И password, и следующая за ним опция, restricted, на самом деле являются примерами опций для каждого образа, которые для удобства могут быть указаны в глобальной секции. Если указано именно так, то одно и то же значение используется для всех образов до тех пор, пока оно не будет переопределено в разделе настроек конкретного образа.

restricted

смягчает запрос пароля так, что пароль запрашивается только если пользователь пытается использовать при загрузке дополнительные параметры. Вы можете использовать это, чтобы позволить пользователю загружаться нормально без ввода пароля, но заставить ввести пароль при загрузке в режиме единичного пользователя.

Следующая секция описывает опции специфичные для RHEL3.

image

указывает, что это секция системы Linux, которая загружается из файла. Параметром является имя файла образа ядра Linux.

label

это необязательная метка, которую вы можете ввести вместо полного имени файла образа.

initrd
: это имя инициализационного RAM-диска, который содержит модули, необходимые ядру до того, как будет смонтирована файловая система.

read-only
: указывает, что корневая файловая система должна быть изначально смонтирована в режиме только для чтения. Последующие стадии загрузки обычно перемонтируют ее в режим чтение/запись после того, как она будет проверена.

append
: указывает опции, передаваемые ядру. В нашем примере указано, что для /dev/hdd должна использоваться эмуляция SCSI (2.4 и более ранние ядра использовали таким образом оптические устройства типа CD-ROM). Также указано, что раздел с меткой RHEL3 должен монтироваться как корневой (/).

В последней секции указаны опции для нашей не-Linux системы.

other
: указывает имя устройства, содержащего устройство (или файл) в котором находится загрузочный сектор загружаемой системы.

loader
: указывает используемый загрузчик. LILO поддерживает chain.b, который просто загружает эту загрузочную запись загрузочного раздела или, как вариант, /boot/os2_d.b который может использоваться для загрузки OS/2 со второго жесткого диска.

label
: не обязательная метка, которую вы можете использовать вместо полного имени образа при выборе образа.

Теперь если мы вставим пустую дискету мы сможем выполнить команду lilo (/sbin/lilo) для создания загрузочной дискеты как показано в Листинге 4. Заметьте, что команда lilo имеет пять уровней детальности вывода. Добавьте дополнительный -v для каждого уровня.

Листинг 4. Создание загрузочной дискеты с lilo

```
[root@lyrebird root]# lilo -v -v
LILO version 21.4-4, Copyright (C) 1992-1998 Werner Almesberger
'lba32' extensions Copyright (C) 1999,2000 John Coffman

Reading boot sector from /dev/fd0
Merging with /boot/boot.b
Secondary loader: 11 sectors.
Mapping message file /boot/message
Compaction removed 43 BIOS calls.
Message: 74 sectors.
Boot image: /boot/vmlinuz-2.4.21-32.0.1.EL
Setup length is 10 sectors.
Compaction removed 2381 BIOS calls.
Mapped 2645 sectors.
Mapping RAM disk /boot/initrd-2.4.21-32.0.1.EL.img
Compaction removed 318 BIOS calls.
RAM disk: 354 sectors.
Added linux *
Boot other: /dev/hda1, on /dev/hda, loader /boot/chain.b
Compaction removed 0 BIOS calls.
Mapped 6 (4+1+1) sectors.
Added WIN-XP
/boot/boot.0200 exists - no backup copy made.
```

```
Map file size: 8192 bytes.
Writing boot sector.
```

Теперь мы получили загрузочную дискету LILO. Если LILO обнаружит ошибку, то вы увидите сообщение о ней и загрузочный сектор не будет записан. Например, если в нашем файле /etc/lilo.conf мы пропустим опцию lba32, то мы увидим нечто похожее на приведенное в Листинге 5. Это будет совет использовать опции linear или lba32. В этом случае мы используем командную строку для указания опции -l, что равнозначно указанию опции linear в lilo.conf. Если мы проделаем это еще раз с опцией -L, то lilo должен завершить все успешно и результат будет схож с приведенным ранее.

Листинг 5. Пример /etc/lilo.conf с ошибкой

```
[root@lyrebird root]# lilo
Warning: device 0x030b exceeds 1024 cylinder limit
Fatal: geo_comp_addr: Cylinder number is too big (16284 > 1023)
[root@lyrebird root]# lilo -l
Warning: device 0x030b exceeds 1024 cylinder limit
Fatal: sector 261613688 too large for linear mode (try 'lba32'
instead)
```

При тестировании вашей загрузочной дискеты, измените запись boot=/dev/fd0 в вашем файле lilo.conf, чтобы установить LILO в MBR или загрузочную запись раздела. Например, указание boot=/dev/hda установит LILO в основную загрузочную запись вашего первого жесткого диска IDE.

Вы получили некоторое представление о LILO и его конфигурационном файле, включая то как изменять некоторые опции конфигурации из командной строки lilo. Более подробную информацию вы найдете в man-страницах lilo, используя команду man lilo. Вы можете найти еще более развернутую информацию в руководстве пользователя в формате postscript, устанавливаемом вместе с пакетом lilo. Оно должно быть установлено в вашем каталоге с документацией, но точное место расположения может варьироваться от системы к системе. Одним из способов ее обнаружения является фильтрация списка пакетов при помощи grep. Листинг 6 показывает это для основанной на rpm системы RHEL3, которую мы используем в качестве примера.

Листинг 6. Обнаружение руководства пользователя при помощи rpm.

```
[ian@lyrebird ian]$ rpm -ql lilo | grep ".ps$"
/usr/share/doc/lilo-21.4.4/doc/Technical_Guide.ps
/usr/share/doc/lilo-21.4.4/doc/User_Guide.ps
```

Дополнительные параметры командной строки LILO

LILO имеет несколько дополнительных параметров командной строки.

lilo -q
: отображает информацию из файла распределения памяти

lilo -R
: настроит lilo на автоматическую загрузку определенной системы при следующей перезагрузке. Это очень удобно для автоматической перезагрузки удаленных систем.

lilo -l
: отображает информацию о пути к файлу ядра

lilo -u
удаляет lilo и восстанавливает прежнюю загрузочную запись.
При загрузке системы Linux через LILO, вам может понадобиться указать дополнительные параметры. Например, если графическая оболочка не загружается, то вы можете захотеть загрузиться в режиме mode 3 или в однопользовательском режиме для восстановления. Любой текст, набираемый вами после имени метки будет передан ядру. Например, в нашем примере мы можем выбрать систему RHEL просто набрав "linux". Для загрузки в mode 3 или однопользовательском режиме, следует набрать одну из указанных строк соответственно.

```
linux 3
linux single
```

Напомним также, что с LILO вы **должны** выполнить команду lilo каждый раз при обновлении файла настроек (/etc/lilo.conf). Вам также следует выполнять команду lilo если вы добавляете, перемещаете или удаляете разделы или производите любые другие изменения, которые могут повредить сгенерированный загрузчик.

GRUB

GRUB или GRand Unifood Boot loader (Главный Унифицированный Загрузчик), это второй из двух наиболее распространенных загрузчиков Linux. Как и LILO, GRUB может быть установлен в MBR вашего загрузочного жесткого диска или в загрузочную запись раздела. Также он может быть установлен на сменных носителях, таких как дискета, CD или USB драйв. Если вы еще не знакомы с GRUB, то хорошей идеей будет попрактиковаться на дискете или USB диске, что мы и будем делать в приводимом примере.

GRUB, или GNU GRUB, в настоящее время разрабатывается под патронажем Free Software Foundation. Новая версия, GRUB 2 находится в стадии разработки, а оригинальная версия GRUB 0.9x теперь известна как Grub Legacy.

В процессе установки Linux вы обычно выбираете в качестве менеджера загрузки или LILO, или GRUB. Если вы выбираете LILO, скорее всего GRUB не будет установлен автоматически. В этом случае вам потребуется установить пакет GRUB вручную. Смотри секцию управления пакетами ниже в этом учебнике, если для этого вам необходима помощь. В дальнейшем мы будем предополагать, что он уже установлен.

GRUB имеет файл настроек, расположенный обычно в /boot/grub/grub.conf. Если ваша система поддерживает символьные ссылки, как большинство файловых систем Linux, то вероятно у вас есть /boot/grub/menu.lst как символьная ссылка на /boot/grub/grub.conf.

Команда grub (/sbin/grub, а на некоторых системах, /usr/sbin/grub) -- это небольшая, но чрезвычайно мощная оболочка, которая поддерживает различные команды для установки GRUB, загрузки систем, размещения и отображения конфигурационных файлов и подобных задач. В этой оболочке используется тот же код, который загружается на второй стадии загрузчика GRUB, поэтому полезно изучить GRUB без выполнения перезагрузки компьютера. Стадия 2 в GRUB запускается или в командном режиме, или в меню, позволяя вам выбрать

операционную систему из меню или указать специальную команду загрузки системы. Имеется также несколько других команд, таких как grub-install, которые используют оболочку grub и помогают автоматизировать задачи вроде установки GRUB.

Листинг 7 содержит часть конфигурационного файла GRUB. Просматривая его, помните одну важную вещь -- GRUB нумерует диски, разделы и все остальное что должно быть пронумеровано, начиная с 0, а не с 1.

Листинг 7. Пример конфигурационного файла GRUB /boot/grub/menu.lst.

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this
file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd1,5)
#          kernel /boot/vmlinuz-version ro root=/dev/hdc6
#          initrd /boot/initrd-version.img
#boot=/dev/hdc6
default=2
timeout=10
splashimage=(hd0,6)/boot/grub/splash.xpm.gz
password --md5 $1$/8Kl21$3VPIphs6REHeHccwzjQYO.

title Red Hat Linux (2.4.20-31.9)
        root (hd0,6)
        kernel /boot/vmlinuz-2.4.20-31.9 ro root=LABEL=RH9 hdd=ide-
scsi
        initrd /boot/initrd-2.4.20-31.9.img

title Red Hat Linux (2.4.20-6)
        root (hd0,6)
        kernel /boot/vmlinuz-2.4.20-6 ro root=LABEL=RH9 hdd=ide-scsi
        initrd /boot/initrd-2.4.20-6.img

title Red Hat Enterprise Linux WS A (2.4.21-32.0.1.EL)
        root (hd0,10)
        kernel /boot/vmlinuz-2.4.21-32.0.1.EL ro root=LABEL=RHEL3
hdd=ide-scsi
        initrd /boot/initrd-2.4.21-32.0.1.EL.img

title      Ubuntu, kernel 2.6.10-5-386
root       (hd1,10)
kernel          /boot/vmlinuz-2.6.10-5-386 root=/dev/hdb11 ro quiet
splash
initrd          /boot/initrd.img-2.6.10-5-386
savedefault
boot

title      Ubuntu, kernel 2.6.10-5-386 (recovery mode)
lock
root       (hd1,10)
kernel          /boot/vmlinuz-2.6.10-5-386 root=/dev/hdb11 ro single
initrd          /boot/initrd.img-2.6.10-5-386
boot
```

```
title Win/XP
        rootnoverify (hd0,0)
        chainloader +1

title Floppy
        root (fd0)
        chainloader +1
```

Также как и в конфигурационном файле LILO, первый набор опций определяет определяет поведение GRUB. Для GRUB он называется menu commands (команды меню) и должен быть указаны до остальных команд. Остальные секции содержат опции для каждого образа операционных систем, которые мы хотим загружать через GRUB. Заметьте, что "title" -- это команда меню. Каждому вхождению title сопутствует одна или несколько базовых команд или пунктов меню. Пример LILO был типичным простым вариантом двойной загрузки систем Windows и Linux. Этот же пример был создан на том же компьютере, что и предыдущий, но в него мы добавили несколько дополнительных операционных систем, чтобы показать вам кое-что из возможностей загрузчика. Вы можете распознать многие сходные элементы, появляющиеся в конфигурационных файлах и LILO, и GRUB. Вы можете задаться мыслью, что же следует изменить, чтобы добавить эти несколько дополнительных операционных систем к предыдущему примеру LILO.

Команды меню, применяемые ко всем остальным секциям в нашем примере это:

#

Любые строки. начинающиеся с # это комментарии и GRUB их игнорирует. Этот конфигурационный файл изначально был создан anaconda, установщиком Red Hat. Если вы установли GRUB при установке Linux, то возможно вы обнаружите комментарии, добавленные в конфигурационный файл GRUB. Комментарии обычно выступают в качестве помощи программам обновления системы, чтобы настройки GRUB оставались рабочими при обновлениях ядра. Если вы самостоятельно редактируете файл конфигурации, то обратите внимание на любые пометки, оставленные в этих целях.

default

указывает какая система будет грузиться по умолчанию, если пользователь не сделает выбор в отведенное время (timeout). В нашем примере, default=2, что означает загрузку третьей записи. напомним, что GRUB использует нумерацию с 0, а не с 1. Если ничего не указано, то по умолчанию будет грузиться первая запись, то есть запись с номером 0.

timeout

указывает в секундах время задержки перед началом загрузки системы по умолчанию. Заметьте, что LILO для указания задержки использует десятые доли секунды, тогда как GRUB использует целые секунды.

splashimage

Указывает фоновое или splash-изображение, которое будет отображаться в загрузочном меню. GRUB обращается к первому жесткому диску как (hd0) и первому разделу этого диска (hd0,0), так что указание splashimage=(hd0,6)/boot/grub/splash.xpm.gz означает использование файла /boot/grub/splash.xpm.gz, расположенного в разделе 7 первого жесткого диска. Запомните, что нумерация с нуля. Отметим также, что образ является XPM файлом, сжатым при помощи gzip. Поддержка splash-изображения это патч (patch), который может содержаться, а может и не содержаться в вашем дистрибутиве.

password

определяет пароль, который требуется ввести для доступа к меню, а также ввода команд GRUB и изменения настроек. Как и в LILO пароль может быть обыкновенным текстом. GRUB позволяет также хранить пароли в виде MD5 файлов, как показано в нашем примере. Это в некоторой степени более безопасно и большинство администраторов устанавливают пароли именно так. Если пароль не используется, то пользователь получает полный доступ к командной строке GRUB.

В нашем примере используется пять дистрибутивов Linux (три Red Hat и два Ubuntu) плюс Windows XP и возможность загрузки с дискеты. Команды, используемые в этих секциях:

title

это заголовок-описание, отображающийся в строке меню при загрузке GRUB. Вы используете клавиши стрелок для перемещения вверх и вниз по заголовкам, а затем нажимаете клавишу Enter для выбора некоторого элемента.

root

указывает раздел, с которого следует загружаться. Помните, что как и для splashimage, нумерация начинается с 0, так что первая система Red Hat, указанная как root (hd0,6) это на самом деле расположена на разделе 7 первого жесткого диска (в данном случае /dev/hda7), В то время как для первой системы Ubuntu в качестве root указан (hd1,10) расположенный на втором жестком диске (/dev/hdb11). GRUB попытается смонтировать это раздел для его проверки и обеспечения загрузки операционной системы с разными параметрами.

kernel

указывает образ ядра, который следует загрузить, а также параметры ядра. Это похоже на комбинацию команд LILO image и append. В приведенном примере у нас имеется два разных ядра Red Hat 9, плюс ядро рабочей станции Red Hat Enterprise Linux 3 Workstation, и одна и та же система Ubuntu с двумя различными наборами параметров ядра.

initrd

Имя RAM-диска, содержащего модули, необходимые ядру перед монтированием файловой системы.

savedefault

Приведено здесь для иллюстрации. Если указана команда меню default=saved, и для операционной системы используется команда savedefault, то загрузка этой операционной системы приведет к тому, что она станет выбором по умолчанию до тех пор, пока не будет загружена операционная система, также имеющая команду savedefault. В нашем примере указание default=2 приводит к игнорированию всех сохраненных умолчаний (saved default).

boot

не обязательный параметр, который указывает GRUB загружать выбранную операционную систему. Это действие по умолчанию, после того, как выполнены все команды для выделенного.

lock

в приведенном примере используется для второй системы Ubuntu. Эта система будет загружена в режиме уединенного пользователя, что позволяет пользователю производить изменения в системе, которые обычно требуют прав суперпользователя (root). Если указана эта опция, то вы должны также указать password в начальных опциях, иначе пользователь сможет изменить вашу опцию lock и загрузить систему или добавить "single" к другой из имеющихся записей. При желании можно также указать различные пароли для каждой записи.

rootnoverify

похожа на root, за исключением того, что GRUB не пытается смонтировать файловую систему или проверить ее параметры. Обычно она используется для таких файловых систем, как NTFS, которые не поддерживаются GRUB. Вы также можете использовать ее если хотите загрузить главную загрузочную запись (MBR) жесткого диска, например для доступа к другому конфигурационному файлу или для перезагрузки предыдущего загрузчика.

chainloader

указывает, что в качестве файла первой стадии (stage 1 file) будет загружен другой файл. Значение "+1" эквивалентно 0+1, что означает загрузку одного сектора, начиная с сектора 0, то есть загрузку первого сектора устройства, указанного в root или rootnoverify

Теперь вы имеете некоторое представление о том, что вы можете найти в стандартном файле /boot/grub/grub.conf (или /boot/grub/menu.lst). Существует множество других команд GRUB для предоставления обширного контроля за процессом загрузки, а также для помощи в установке grub и выполнении других задач. Вы можете узнать больше о них в руководстве GRUB, которое должно быть доступно в вашей системе по команде info grub.

Теперь, когда у нас есть файл конфигурации GRUB нам необходимо создать загрузочную дискету, чтобы протестировать его. Наипростейший способ сделать это -- использовать команду grub-install как показано в Листинге 8. Если вы устанавливаете GRUB на дискету или в раздел, то вы должны сначала размонтировать это устройство. Но это не требуется, если вы устанавливаете GRUB в MBR жесткого диска, поскольку вы монтируете только разделы (/dev/hda1, /dev/hda2 и т. д.), а не весь жесткий диск (/dev/hda).

Листинг 8. Установка GRUB на дискету.

```
[root@lyrebird root]# umount /dev/fd0
umount: /dev/fd0: not mounted
[root@lyrebird root]# grub-install /dev/fd0
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(fd0)   /dev/fd0
(hd0)   /dev/hda
(hd1)   /dev/hdc
(hd2)   /dev/sda
```

Замечание: Вы также можете использовать имя устройства GRUB (fd0) вместо /dev/fd0, но если вы это делает, то должны заключить его в кавычки, чтобы избежать его интерпретации командной оболочкой. Например:

```
grub-install '(fd0)'
```

Если вы начали работу с пустой дискетой, и уже смонтировали ее, то вы увидите, что она осталось пустой. Произошло только то, что GRUB записал измененный загрузчик первой стадии в первый сектор дискеты. Это не отображается в файловой системе. Этот загрузчик первой стадии загружает загрузчик второй стадии и конфигурационный файл с вашего жесткого диска. Попытайтесь

загрузиться с дискеты и вы увидите весьма небольшую активность работы с нею перед тем, как отобразится меню.

Карта устройств покажет вам как GRUB подгоняет свое внутреннее представление ваших дисков (fd0, hd0, hd1) к представлению Linux (/dev/fd0, /dev/hda, /dev/hdb). В системе с одним или двумя IDE жесткими дисками и, может быть, с дисководом это, возможно, будет корректным. Если карта устройств уже существует GRUB воспользуется ею без проверки. Если вы просто добавили новый диск и хотите выполнить генерацию новой карты устройств, то к команде grub-install следует добавить опцию --resize. Например,

Сразу же после тестирования вашей дискеты, вы готовы к установке GRUB в MBR вашего жесткого диска. Для первого жесткого диска IDE вам следует использовать:

```
grub-install /dev/hda
```

или

```
grub-install '(hd0)'
```

Чтобы установить его в загрузочную запись раздела 11, используйте:

```
grub-install /dev/hda11
```

или

```
grub-install '(hd0,10)'
```

Запомните, что GRUB нумерует с 0.

Обновления системы

Большинство дистрибутивов предоставляют инструменты обновления систем. Эти инструменты обычно осведомлены об установленном загрузчике и часто обновляют конфигурационный файл автоматически. Если вы собрали свое собственное ядро или предпочитаете использовать конфигурационные файлы с нестандартным именем или расположением, то вам может потребоваться обновить конфигурационный файл самостоятельно.

- Если вы используете LILO, то вы должны выполнить команду lilo, и не важно обновили ли вы конфигурационный файл или произвели такие изменения, как добавление жесткого диска или удаление раздела.
- Если вы используете GRUB, вы можете отредактировать файл /boot/grub/grub.conf чтобы произвести изменения, а загрузчик второй стадии GRUB прочитает этот файл при следующей перезагрузке. Обычно вам не нужно переустанавливать GRUB только потому, что вы добавили новое ядро. Однако, если вы передвигаете раздел или добавляете диски, то вам может понадобиться переустановить GRUB. Запомните, что загрузчик первой стадии очень мал, потому как он просто содержит список адресов блоков загрузчика второй стадии. При передвижении раздела адресация меняется, поэтому первая стадия больше не сможет обнаружить вторую стадию. Далее мы опишем некоторые стратегии восстановления, а также обсудим загрузчик стадии 1.5 GRUB.

Восстановление

Теперь мы рассмотрим что может пойти не так при тщательно подогнанных установках загрузки, особенно когда вы используйте множественную загрузку.

Первое что следует запомнить -- не паникуйте. Обычно восстановление это всего лишь несколько шагов. Мы предоставим вам несколько стратегий, которые помогут вам во многих кризисных ситуациях.

Эти стратегии и инструменты покажут вам, что любой, имеющий физический доступ к машине обладает значительными возможностями. Подобно этому любой, имеющий доступ к командной строке grub, имеет доступ к файлам вашего компьютера без учёта соглашений о правах доступа или любых других средств безопасности, предоставляемых работающей операционной системой. Имейте это ввиду при выборе загрузчика. Выбор между LILO и GRUB во многом основан на личных предпочтениях. Основываясь на том, что вы уже знали, а также на том о чем было рассказано, вы должны уже быть подготовлены к выбору загрузчика, который наиболее соответствует вашим нуждам и стилю работы.

Установка других систем может повредить ваш MBR.

Когда-нибудь при установке операционной системы вы случайно можете перезаписать ваш MBR. Некоторые системы, такие как DOS и Windows всегда устанавливают свой собственный MBR. В таком случае обычно это очень легко восстановить. Если вы выработали привычку создавать загрузочную дискету каждый раз при запуске lilo или перезагрузке (GRUB), то все очень легко. Просто загрузите Linux с вашей загрузочной дискеты и перезапустите lilo или grub-install.

Если так случилось, что у вас нет загрузочной дискеты, но есть хоть какой-то дистрибутив Linux, то обычно вы можете загрузиться с установочного диска в режиме восстановления (recovery mode). Если вы сделает это, то корневая файловая система вашего диска будет или смонтирована в некоторую точку восстановления (recovery point), или диск не будет смонтирован вообще. Вы можете использовать команду chroot чтобы сделать эту добавочную точку вашим корневым каталогом (/). Затем запустите lilo или grub-install чтобы создать новую загрузочную дискету или переписать MBR. Я обычно предпочитаю сначала создать загрузочную дискету и загрузиться с неё, чтобы перед перезаписью MBR убедиться что с моими настройками загрузчика всё в порядке, но вы можете быть более смелыми чем я. Листинг 9 показывает пример использования рабочего окружения, которое мы создали в наших предыдущих примерах конфигурирования. В этом примере, я загрузился с загрузочного диска Red Hat Enterprise Linux, который смонтировал /dev/hda11 в /mnt/sysimage. Большинство дистрибутивов в режиме восстановления выдают вам большой экран с приглашением командной строки, а не графическое окно, с которым вы привыкли работать. Можете считать, что это окно терминала, которое вы открыли от имени суперпользователя. Другими словами будьте очень осторожны записывая что-либо на жесткий диск. В Листинге 9 вводимое пользователем выделено жирным шрифтом.

Листинг 9. Использование диска для восстановления и chroot.

```
sh-3.00# chroot /mnt/sysimage
sh-2.05b# lilo
Added linux *
Added WIN-XP
sh-2.05b# grub-install '(fd0)'
Installation finished. No error reported.
This is the contents of the device map /boot/grub/device.map.
```

```
Check if this is correct or not. If any of the lines is incorrect,
fix it and re-run the script `grub-install'.

(fd0)   /dev/fd0
(hd0)   /dev/hda
(hd1)   /dev/hdc
(hd2)   /dev/sda
sh-2.05b#
```

Как только загрузочная дискета будет создана, нажмите ctrl-d, чтобы выйти из среды chroot, а затем перезагрузите компьютер, не забыв удалить установочный носитель. Если у вас нет на руках установочного CD или DVD, то существует множество CD для восстановления и Live-CD Linux, доступных в сети, а также несколько дискет и USB-дисков. Смотри Ресурсы.

Хотя это выходит за границы данного учебника, но вам может быть будет полезно узнать, что можно иметь MBR, загружающий систему Windows 2000 или Windows XP, и установить Lilo или GRUB в загрузочную запись раздела. Программа загрузки ntldr также может загружать другие цепочки загрузочных секторов, хотя ее настройка -- дело не простое. Вам понадобится скопировать загрузочный сектор на Windows-раздел и изменить скрытый файл boot.ini чтобы это сработало.

Вы переместили раздел.

Если вы переместили раздел и забыли про настройки загрузки, то могут возникнуть некоторые проблемы. Обычно LILO или GRUB отказываются загружаться. LILO возможно выводит 'L', что говорит о том, что первая стадия загрузки прошла, и на этом загрузка остановилась. GRUB выводит сообщение об ошибке. А произошло следующее: загрузчик первой стадии, у которого есть список секторов которые нужно загрузить, чтобы перейти к стадии 2, пытается загрузить сектора, расположенные по этим адресам, но они уже не содержат загрузочные сигнатуры второй стадии. Если вы создали загрузочную дискету используя описанные выше методы, то помните: все что и lilo и grub-install размещают на дискете -- это только загрузочный сектор, так что ваша загрузочная дискета скорее всего не поможет. Как и в предыдущем примере, вы можете попытаться загрузить различные окружения для восстановления и переделать загрузочную дискету с LILO или GRUB. Затем перезагрузиться, проверить вашу систему и вновь установить загрузчик в MBR.

Вы могли заметить, что наши примеры конфигурирования использовали метки разделов. Например,

```
append="hdd=ide-scsi root=LABEL=RHEL3"
```

или

```
kernel /boot/vmlinuz-2.4.20-31.9 ro root=LABEL=RH9 hdd=ide-scsi
```

Я часто использую метки подобные этим, чтобы избежать проблем при перемещении разделов. В этом случае вам все еще нужно обновить конфигурационный файл GRUB или LILO, но не нужно изменять /etc/fstab. Это может быть особенно полезно, если я создаю образ раздела на одном компьютере и восстанавливаю его в другом месте на другом компьютере.

Использование раздела /boot.

Другим подходом к восстановлению или, возможно, избежания этого является использование отдельного раздела для /boot. Этот раздел не требует очень много места, возможно 100МБ или около того. Разместите этот раздел там, где он не будет требовать перемещения и где его номер не будет меняться при добавлении или перемещении других разделов. В смешанной системе Windows и Linux хорошим выбором для раздела /boot будет /dev/hda2.

Другой причиной для создания раздела /boot может быть случай, когда ваша корневая файловая система не поддерживается загрузчиком. Например, считается общепринятым форматировать раздел /boot в ext2 или ext3, тогда как для корневого раздела (/) использовать LVM.

Если у вас установлено несколько дистрибутивов, **не используйте** единственный раздел /boot для них всех. Не забудьте настроить LILO или GRUB на загрузку с раздела, который в последствии будет смонтирован как /boot. Помните также, что программа обновления дистрибутива обычно обновляет конфигурацию GRUB или LILO именно для этой системы. В среде со множеством операционных систем вы можете пожелать связать с одной из них раздел /boot и сделать ее главной, а во всех остальных -- при необходимости править конфигурационные файлы в ручную. В качестве другого подхода можно использовать установку загрузчика для каждой системы в загрузочную запись ее раздела, а ваша главная система просто будет по цепочке загружать загрузочные записи разделов отдельных систем, и в результате при загрузке получим две стадии, каждая со своим меню.

Создание самодостаточной загрузочной дискеты.

И наконец, давайте более пристально посмотрим на настройку GRUB и то как сделать самодостаточную загрузочную дискету, которая сможет предоставить командую строку GRUB, вне зависимости от того, что произошло с вашим жестким диском.

Вспомните все, что мы говорили о цилиндрах жесткого диска. Хотя вы можете считать, что цилиндры в современных жестких дисках это некая фикция, многие части вашей файловой системы о них не забыли. В частности вы обнаружите, что разделы используют целые числа цилиндров и выравнены по границам цилиндров. В разделах многие файловые системы также работают с пространством в единицах цилиндров. Во многих UNIX и Linux системах структура файловой системы хранится в суперблоке, являющимся первой единицей размещения файловой системы. Для таких файловых систем как ext2 или ext3 и довольно больших жестких дисков пространство разбивается на несколько секций с копией суперблока в начале каждой секции. Это может помочь при восстановлении, если вы случайно испортили границы раздела в такой программе, как fdisk.

Еще одним плюсом в пользу работы с цилиндрами является то, что в начале диска имеется некоторое пространство, идущее сразу же за MBR. GRUB использует это, размещая в нем загрузчик стадии 1.5 или в другом подобном не используемом месте раздела где это возможно. Загрузчик стадии 1.5 распознает файловую систему раздела, содержащего стадию 2, так что это в чем-то более устойчиво по отношению к проблемам, связанным с перемещением файлов.

Все это хорошо и прекрасно, но как это связано с загрузочной дискетой? Что ж, на дискете не так много места и нет даже понятия цилиндров, поэтому если вы хотите загрузить и стадию 1 и стадию 2 GRUB с дискеты, вам потребуется установить стадию 1, а затем скопировать стадию 2 в сектора, следующие за загрузочным сектором. Листинг 10 содержит пример того, как это можно сделать. Используйте чистую дискету, поскольку в результате производимых действий все данные на ней будут уничтожены. Вам следует скопировать файлы поставляемые с вашим дистрибутивом grub, а не из вашего каталога /boot/grub, поскольку /boot/grub/stage2 изменяется для работы с разделами вашего жесткого диска. Вы сможете найти оригинальные файлы stage1 и stage2 в подкаталоге /usr/share/grub. В нашем примере они расположены в /usr/share/grub/i386-redhat.

Листинг 10. Создание загрузочной дискеты GRUB.

```
[root@lyrebird root]# ls /usr/share/grub
i386-redhat
[root@lyrebird root]# cd /usr/share/grub/i386-redhat
[root@lyrebird i386-redhat]# ls -l st*
-rw-r--r--    1 root     root          512 Aug  3  2004 stage1
-rw-r--r--    1 root     root       104092 Aug  3  2004 stage2
[root@lyrebird i386-redhat]# dd if=stage1 of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
[root@lyrebird i386-redhat]# dd if=stage2 of=/dev/fd0 bs=512 seek=1

203+1 records in
203+1 records out
```

Если вы отформатировали вашу дискету перед тем как выполнить это, и теперь пытаетесь смонтировать ее, то команда mount выдаст ошибку. Копирование stage2 сразу же вслед за загрузочным сектором дискеты (seek=1) уничтожит файловую систему на ней.

Если теперь вы загрузитесь с этой дискеты, то вы отметите задержку при загрузке второй стадии с дискеты. Вы можете загрузиться с этой дискеты на любом ПК, и наличие на нем системы Linux не обязательно. Когда вы загрузитесь с дискеты вы увидите приглашение командной строки GRUB. Нажмите клавишу TAB чтобы увидеть список доступных команд. Для получения справки о команде с именем commandname используйте help commandname. Листинг 11 показывает пример командной строки GRUB.

Листинг 11. Командная строка GRUB.

```
  GRUB  version 0.93  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word,
TAB
   lists possible command completions.  Anywhere else TAB lists the
possible
   completions of a device/filename.]

grub>
 Possible commands are: blocklist boot cat chainloader clear cmp color
configfi
```

```
le debug device displayapm displaymem dump embed find fstest geometry
halt help
 hide impsprobe initrd install ioprobe kernel lock makeactive map
md5crypt modu
le modulenounzip pager partnew parttype password pause quit read
reboot root ro
otnoverify savedefault serial setkey setup terminal terminfo testload
testvbe u
nhide uppermem vbeprobe

grub> help rootnoverify
rootnoverify: rootnoverify [DEVICE [HDBIAS]]
    Similar to `root', but don't attempt to mount the partition. This
    is useful for when an OS is outside of the area of the disk that
    GRUB can read, but setting the correct root device is still
    desired. Note that the items mentioned in `root' which derived
    from attempting the mount will NOT work correctly.

grub> find /boot/grub/grub.conf
 (hd0,2)
 (hd0,6)
 (hd0,7)
 (hd0,10)
 (hd1,7)

grub>
```

В этом примере мы смогли узнать, что на четырех различных разделах первого диска имеется файл конфигурации GRUB и еще один на втором жестком диске. Мы могли бы загрузить меню GRUB с одного из них, использовав команду configfile. Например:

```
configfile (hd0,2)/boot/grub/grub.conf
```

Это приведет к загрузке меню этого конфигурационного файла и мы сможем загрузить систему с этой точки. Вы можете найти эти команды grub в руководстве GRUB. Наберите info grub в окне терминала, чтобы открыть руководство.

И последнее, прежде чем мы закончим с GRUB. Мы упоминали, что файл второй стадии GRUB уничтожает файловую систему на дискете. Если вы хотите получить GRUB дискету для восстановления, загружающую файлы GRUB, включая конфигурационный файл, с дискеты, то вы можете сделать это, выполнив следующие шаги:
используйте команду mkdosfs для создания на дискете файловой системы DOS FAT и опцию -R для резервирования достаточного количества секторов для файла второй стадии.

1. Смонтируйте дискету
2. Создайте на дискете каталог /boot/grub
3. Скопируйте файлы GRUB stage1, stage2 и grub.conf в каталог boot/grub на дискете. Скопируйте также, если хотите, файл с фоновым изображением (заставкой).
4. Отредактируйте файл grub.conf на дискете так, чтобы он ссылался на файл с изображением на дискете.

5. Размонтируйте дискету
6. Используйте командную оболочку grub для установки GRUB на дискету, используя команды GRUB root и setup.

Мы проиллюстрировали это в Листинге 12.

Листинг 12. Установка GRUB на дискету с файловой системой.

```
[root@lyrebird root]# mkdosfs -R 210 /dev/fd0
mkdosfs 2.8 (28 Feb 2001)
[root@lyrebird root]# mount /dev/fd0 /mnt/floppy
[root@lyrebird root]# mkdir /mnt/floppy/boot
[root@lyrebird root]# mkdir /mnt/floppy/boot/grub
[root@lyrebird root]# cp /boot/grub/stage1 /mnt/floppy/boot/grub
[root@lyrebird root]# cp /boot/grub/stage2 /mnt/floppy/boot/grub
[root@lyrebird root]# cp /boot/grub/splash* /mnt/floppy/boot/grub
[root@lyrebird root]# cp /boot/grub/grub.conf /mnt/floppy/boot/grub
[root@lyrebird root]# umount /dev/fd0
[root@lyrebird root]# grub
Probing devices to guess BIOS drives. This may take a long time.

    GRUB  version 0.93  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word,
TAB
   lists possible command completions.  Anywhere else TAB lists the
possible
   completions of a device/filename.]
grub> root (fd0)
 Filesystem type is fat, using whole disk

grub> setup (fd0)
 Checking if "/boot/grub/stage1" exists... yes
 Checking if "/boot/grub/stage2" exists... yes
 Checking if "/boot/grub/fat_stage1_5" exists... no
 Running "install /boot/grub/stage1 (fd0) /boot/grub/stage2 p
/boot/grub/grub.c
onf "... succeeded
Done.
```

Рассмотренные инструменты это все, что необходимо вам для восстановления после различных ошибок, которые могут возникнуть при использовании загрузчика.

Сборка и установка программ

Загрузка и распаковка

Вне зависимости от вашей мотивации сборки из исходных текстов, прежде чем перейти непосредственно к самой сборке вам потребуется заполучить исходные тексты. Вы можете найти пакет на сайте, разработанном для размещения проектов, вроде сайта SourceForge.net от Open Source Technology Group (смотри Ресурсы), или на сайте посвященному только этому проекту.

В этом разделе мы в основном будем рассматривать пакеты распространяемые в виде так называемых сжатых tar-файлов (tarball). Команда tar (от Tape ARchive -- ленточный архив) используется для создания архивов с файлами из древа каталогов. Несмотря на название, tar-файлы могут располагаться на любом носителе. Хранение их на диске позволяет выполнять такие операции, как например удаление части архива, что невозможно сделать на ленте. Команда tar сама по себе не сжимает данные, она просто соединяет их в один файл в специальноми формате, позволяющем сохранить как сами файлы, так и все права доступа, запреты и структуру каталогов. Команда tar может быть использована совместно с программой сжатия, обычно gzip или bzip2 для создания сжатого архива, который экономит дисковое пространство, а также уменьшает время передачи файла. Вот этот получившийся архив и называют tarball.

Помимо простых архивов исходные тексты отдельных программ могут быть упакованы для вашего дистрибутива в пакеты с исходными текстами (source package), такие как RPM с исходными текстами (или SRPM). Мы обсудим управление пакетами в этом учебнике позднее. А сейчас просто не забудьте поискать пакет с исходными текстами для вашего дистрибутива, если вы его найдете, то это обычно значительно облегчает компиляцию программы, поскольку он уже подогнан к структуре файловой системы вашего дистрибутива.

Перед загрузкой узнайте как можно больше о пакете. Если доступна информация по сборке или установке, то просмотрите ее на предмет необходимости других пакетов, чтобы было возможно скомпилировать этот. Часто вам необходимо установить еще и несколько библиотек, а возможно и инструмент разработки, чтобы успешно скомпилировать выбранную программу. Это особенно часто бывает, если ваша программа использует любые графические инструментарии. Иногда только после запуска процесса сборки выясняется, что необходим еще какой-то пакет. Не волнуйтесь - это не редкость. Вам просто нужно найти и установить отсутствующий пакет и продолжать попытки пока все требуемые пакеты не будут установлены.

Обычно для загрузки вы можете использовать ваш браузер или, возможно, программу для ftp. Ваш пакет вероятно будет иметь имя, имеющее одно из перечисленных окончаний: tar, tar.gz, tar.Z, tgz, или tar.bz2. Иногда вы будете загружать пакеты, используя CVS (Concurrent Version System -- Система параллельных версий). Примером может быть GNU GRUB 2 от Free Software Foundation (смотри Ресурсы). В этом случае загруженные вами исходные тексты будут уже распакованы. Изредка вы можете найти файлы с расширением .zip, говорящим о том, что это файл zip.

Сжатые tar файлы

Сжатый tar файлы или tarball -- это наиболее популярная форма распространения исходных текстов, если не используется система управления пакетами вроде RPM от Red Hat или управление пакетами в Debian. Они создаются при помощи команды tar, архивирующей древо каталогов и все файлы из него в один архивный файл. Обычно результат может быть сжат при помощи некой программы сжатия, как правило используется одна из следующих compress, gzip или bzip2. Поскольку архивирование и сжатие это наиболее общие операции, команда GNU tar, имеющаяся во многих дистрибутивах Linux может также самостоятельно применять сжатие и развертывание с использованием compress,

gzip или bzip2. Если имеющаяся у вас версия tar не поддерживает указанные типы сжатия, то UNIX и Linux системы весьма хорошо используют конвейер, позволяющий нескольким командам работать над одними и теми же данными по очереди, так что двухступенчатый процесс может выполняться вручную, а tar сделает это для вас в любом случае.

Для иллюстрации предположим, что мы загрузили проект Dr. Geo interactive geometry (смотри Ресурсы). Во время создания этого урока текущая версия содержалась в файле drgeo-1.1.0.tar.gz. Расширение gz говорит о том, что этот файл сжат при помощи gzip. Сначала мы покажем, как получить tar файл из сжатого файла, а затем как извлечь оттуда отдельные файлы. Затем мы покажем вам как распаковать и извлечь файлы одной командой или при помощи конвейера

Для того, чтобы просто извлечь tar-архив мы используем команду gunzipg как показано в Листинге 13.

Листинг 13. Распаковка пакета с исходными текстами Dr Geo.

```
[ian@localhost ~]$ ls drgeo*
drgeo-1.1.0.tar.gz
[ian@localhost ~]$ gunzip drgeo-1.1.0.tar.gz
[ian@localhost ~]$ ls drgeo*
drgeo-1.1.0.tar
```

Отметим, что наш файл .tar.gz теперь заменен исходным .tar файлом. Чтобы распаковать tar-файл для других упомянутых расширений вам следует использовать следующие команды (соответственно)

```
uncompress drgeo-1.1.0.tar.Z
gunzip drgeo-1.1.0.tar.Z
gunzip drgeo-1.1.0.tar.gz
gunzip drgeo-1.1.0.tgz
bunzip2 drgeo-1.1.0.tar.bz2
```

Вы можете отметить, что gunzip может работать с .Z, .tar.gz и .tgz. В действительности же, в вашей системе может вообще не быть программ compress и uncompress.

Для извлечения файлов из tar архива используется команда tar. Стандартная форма: tar -xvf имя_файла.tar, приведена в Листинге 14. Опционально вы можете ограничить вывод, используя малый фильтр для разбивки вывода на страницы.

Листинг 14. Извлечение файлов из архива Dr Geo.

```
[ian@localhost ~]$ tar -xvf drgeo-1.1.0.tar |more
drgeo-1.1.0/
drgeo-1.1.0/po/
drgeo-1.1.0/po/ChangeLog
drgeo-1.1.0/po/Makefile.in.in
drgeo-1.1.0/po/POTFILES.in
drgeo-1.1.0/po/drgeo.pot
drgeo-1.1.0/po/az.po
drgeo-1.1.0/po/ca.po
drgeo-1.1.0/po/cs.po
```

Опция -x говорит tar, что нужно извлечь файлы из архива. Опция -v говорит tar, что нужно вывести подробный листинг обрабатываемых файлов. И опция -f вместе с именем файла (в данном случае drgeo-1.1.0.tar) говорит tar, что архивные файлы извлекаются из файла.

Правильно сделанный пакет в процессе разархивирования создаст каталог, в котором будут сохранены файлы пакета. В нашем примере это каталог drgeo-1.1.0. Иногда пакет этого не делает, и потому вы можете захотеть посмотреть структуру пакета перед тем, как выполнить его распаковку с риском получить огромное количество файлов прямо в домашнем каталоге. Чтобы выполнить проверку используйте команду tar с опцией -t, отображающей таблицу содержимого, вместо -x, производящей разархивирование. Если вы также опустите опцию -v, то все равно полученного листинга будет достаточно для того, чтобы понять какие файлы будут созданы и будет ли создан новый каталог или все будет выгружено в текущий.

Теперь, когда мы увидели как разархивировать сжатые архивы tar в два шага, вы возможно удивитесь заявлению, что все это можно сделать за один шаг. Да можно. Если вы добавите опцию -z к команде tar, то она сможет распаковать и разархивировать архивы, сжатые gzip при помощи одной команды. Например:

```
tar -zxvf drgeo-1.1.0.tgz
```

или

```
tar -zxvf drgeo-1.1.0.tar.Z
```

Для выполнения того же для архивов сжатых при помощи bzip2, вместо опции -z используйте опцию-j. Например:

```
tar -jxvf drgeo-1.1.0.tar.bz2
```

Вы может также использовать опцию -c с любой из указанных выше команд сжатия для направления распакованных файлов в стандартный вывод, который затем преобразуется в стандартный вход для команды tar. Отметим, что это оставляет ваш оригинальный файл неизменным, а не распаковывает его в большой .tar файл. Вот несколько примеров:

```
bunzip2 -c drgeo-1.1.0.tar.bz2 | tar -xvf -
uncompress -c drgeo-1.1.0.tar.Z | tar -xvf -
gunzip -c drgeo-1.1.0.tar.Z | tar -xvf -
gunzip -c drgeo-1.1.0.tar.gz | tar -xvf -
gunzip -c drgeo-1.1.0.tgz | tar -xvf -
```

Примечания:
Значение - вместо имени архивного файла заставляет tar использовать стандартный ввод для архивов. Ваша версия tar может делать это по умолчанию, в этом случае вам нет нужды указывать опцию -f вообще. Просто опустите в конце вышеперечисленных команд завершающие f -.
Команда zcat выполняет те же функции, что и gunzip -c.

Древо CVS

Иногда код необходимого вам проекта не упакован в архив, а доступен через CVS (Concurrent Version System -- Система Параллельных Версий). В качестве примера можно привести проект GRUB 2, обсуждавшийся нами в разделе Менеджеры загрузки. В Листинге 15 приведен пример.

Листинг 15. Загрузка GRUB2 через CVS.

```
[ian@attic4 ~]$ export CVS_RSH="ssh"
[ian@attic4 ~]$ cvs -z3 -d:ext:anoncvs@savannah.gnu.org:/cvsroot/grub
co grub2
cvs server: Updating grub2
U grub2/.cvsignore
U grub2/AUTHORS
U grub2/COPYING
U grub2/ChangeLog
U grub2/DISTLIST
U grub2/INSTALL
U grub2/Makefile.in
U grub2/NEWS
...
```

Команда export говорит CVS как соединиться с удаленным сервером (используя безопасную оболочку (secure shell) или ssh в данном случае). Команда cvs проверяет (опция co) проект grub2. Вы найдете все файлы проекта в каталоге grub2, который команда cvs создаст для вас.

Zip файлы

Изредка вы можете обнаружить пакеты с исходными текстами в виде zip файлов. Это может иметь место для пакетов, которые работают в Windows также как и в Linux системах. Оригинальная программа PKZIP была разработана для систем DOS компанией PKWARE, Inc., а теперь доступна для нескольких платформ. Многие системы Linux имеют версию, созданную Info-ZIP.

Листинг 16 показывает как использовать команду unzip для распаковки исходных текстов хранителя экрана, демонстрирующего выворачивание сферы на изнанку (sphere eversion).

Листинг 16. Распаковка исходных текстов sphere eversion из zip.

```
[ian@attic4 ~]$ unzip sphereEversion-0.4-src.zip
Archive:  sphereEversion-0.4-src.zip
   creating: sphereEversion-0.4-src/
  inflating: sphereEversion-0.4-src/Camera.h
  inflating: sphereEversion-0.4-src/drawutil2D.h
  inflating: sphereEversion-0.4-src/drawutil.h
  inflating: sphereEversion-0.4-src/fontdata.h
  inflating: sphereEversion-0.4-src/fontDefinition.h
  inflating: sphereEversion-0.4-src/generateGeometry.h
  inflating: sphereEversion-0.4-src/global.h
  inflating: sphereEversion-0.4-src/mathutil.h
```

```
  inflating: sphereEversion-0.4-src/Camera.cpp
  inflating: sphereEversion-0.4-src/drawutil2D.cpp
  inflating: sphereEversion-0.4-src/drawutil.cpp
  inflating: sphereEversion-0.4-src/fontdata.cpp
  inflating: sphereEversion-0.4-src/generateGeometry.cpp
  inflating: sphereEversion-0.4-src/main.cpp
  inflating: sphereEversion-0.4-src/mathutil.cpp
  inflating: sphereEversion-0.4-src/README.TXT
  inflating: sphereEversion-0.4-src/Makefile
```

Сборка (компиляция) программ

Теперь, когда исходные тексты распакованы в древо каталогов, рассмотрим как скомпилировать программу или программы.

Инспекция исходных текстов

Перед тем, как запустить компиляцию, вы должны просмотреть что было распаковано. В частности просмотреть документацию по установке. Обычно это файл README или INSTALL, возможно, что и оба, расположенные в каталоге вашего нового проекта. Если пакет разработан для нескольких платформ, вы можете найти файлы, специфичные для некой платформы, такие как README.linux или INSTALL.linux.

Конфигурирование

Весьма часто в главном каталоге исходных текстов встречается скрипт configure. Это скрипт разработан для настройки Makefile, который подгоняется под вашу систему. Он обычно генерируется разработчиком, с использованием программы GNU autoconf. Скрипт configure проверяет вашу систему на наличие всех нужных инструментов и совместимость. Полученный в результате Makefile или несколько таких файлов, скомпилируют проект на вашей конкретной системе.

Сложный конфигурационный скрипт может проверять множество аспектов вашй системы, включая такие вещи, как тип процессора, является ли он 32-х или 64-х битным и так далее. Простой конфигурационный сценарий не делает почти ничего, кроме создания файлов Makefile.

Если у вас нет файла с именем configure в главном каталоге проекта, то просмотрите документацию на предмет другого способа настройки параметров сборки. Если же такой файл у вас есть, то перейдите в главный каталог проекта и выполните.

```
./configure --help
```

Эта команда выдаст справку о доступных опциях конфигурации. Многие из них, такие как --prefix, встречаются в большинстве конфигурационных скриптов. Некоторые скорее всего будут специфичны для конкретного компилируемого проекта. Найдите те, который вам нужно изменить.

Примечание: Если ваш проект не имеет конфигурационного скрипта, то возможно он имеет файл Makefile, работающий на большинстве платформ, или установочный процесс в какой-то другой форме. Например, пакет, использующий

только скрипты Python файлы с данными может не требовать сборки, так что он может иметь просто сценарий для установки.

В учебнике для темы 104 мы рассмотрим стандарт иерархий файловых систем (FHS -- Filesystem Hierarchy Standard). А сейчас отметим, что локальные программы должны иметь исполняемые файлы, сохраненные в древе /usr/local в /usr/local/bin и man-страницы в /usr/local/man. Скрипты configure вероятно имеют опцию --prefix, указывающую место установки. Если программа не совместима с FHS, то вам может потребоваться указать эту опцию при запуске скрипта configure. Если вы компилируете программу для замены установленной версии, то вам может потребоваться установить ее, указав каталог в /opt или /usr в качестве префиксов.

В добавок к возможному указанию префиксов вы можете обнаружить другие опции, связанные с размещением специфичных компонентов, таких как --mandir или --infodir для указания расположения man и info страниц соответственно.

После просмотра возможных опций и определения того, что вам следует изменить, выполните скрипт configure, добавив все необходимые опции. Не забудьте добавить ./ перед командой configure, поскольку каталог вашего проекта вероятно не будет указан в переменной path. Например, вы можете выполнить

```
./configure
```

или

```
./configure --prefix /usr/local
```

После запуска configure обычно вы видите сообщения, рассказывающие о типе используемой вами системы и о том, какие необходимые инструменты установлены, а какие нет. Если все идет хорошо, то к концу процесса конфигурирования вы должны получить созданный Makefile.

```
config.cache
```

По завершении выполнения скрипта configure, он сохраняет информацию о конфигурации в файле с названием config.cache, расположенном в том же каталоге, что и сам скрипт configure.

Если вам необходимо запустить ./configure вновь, то убедитесь, что прежде вы удалили файл config.cache (используйте команду rm), поскольку configure будет использовать настройки из config.cache, если он существует, не производя повторной проверки вашей системы.

Листинг 17 содержит часть выведенных сообщений выполнения configure для пакета Dr Geo, который мы распаковали ранее.

Листинг 17. Конфигурирование Dr Geo.

```
[ian@localhost ~]$ cd drgeo-1.1.0
[ian@localhost drgeo-1.1.0]$ ./configure | less
checking for XML::Parser... ok
checking for iconv... /usr/bin/iconv
```

```
checking for msgfmt... /usr/bin/msgfmt
checking for msgmerge... /usr/bin/msgmerge
checking for xgettext... /usr/bin/xgettext
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of
Makefiles... no
checking for g++... g++
checking for C++ compiler default output file name... a.out
checking whether the C++ compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
...
checking for guile... /usr/bin/guile
checking for guile-config... no
configure: error: guile-config required but not found
```

Скрипт configure проверяет несколько программ преобразования графики, являющихся частью пакета netpbm. Имеется предупреждение, поскольку одна из необходимых программ преобразования не найдена в системе. Есть также предупреждение о том, что использование префикса /usr/local требует прав суперпользователя (на этапе установки). Поскольку это первый запуск, то выводится несколько ошибок, связанных с файлом Makefile, который еще не существует, но появится если мы выполним configure вновь. И наконец скрипт configure сообщает об успешном завершении.

Make и файлы Makefile

По завершении конфигурирования, вы должны получить файл в каталоге проекта с именем Makefile. Он называется сборочный файл проекта, программа с именем make используется для его обработки и сборки программы. Может также иметься несколько make-файлов в ваших подкаталогах.

Make-файл содержит правила, являющиеся инструкциями, которые сообщают программе make как собираьб различные компоненты приложения. Файл также содержит targets (цели), которые сообщают программе make, что именно компилировать. Программа make анализирует make-файл и определяет порядок в котором следует производить компиляцию (сборку). Например, если исполняемый файл создается из трех объектных файлов, то объектные файл должны быть скомпилированы до того, как они будут объединены в исполняемый код. Make-файл может выполнять как компиляцию, так и инсталляцию приложения. Назначения (targets) make-файла обычно доступны для нескольких функций, таких как:

```
make
```

без опций просто компилирует программу. Говоря технически, таким образом компилируется назначение по умолчанию, что обычно означает просто компиляцию программы из исходных текстов.

```
make install
```

устанавливает скомпилированную программу. Если вы производите установку в /usr/local, то вам могут потребоваться права суперпользователя (root).

```
make clean
```

удаляет файлы, созданные в процессе сборки.

```
make all
```

иногда используется для выполнения всех функций make-файла за раз. . Обратитесь к документации проекта, чтобы узнать, есть ли дополнительные назначения (targets) или дополнительные элементы, которые могут понадобиться вам.

Теперь, когда ваш главный Makefile создан, используйте команду make, обычно без опций, для сборки выполняемых файлов, man-страниц и других частей программы. В зависимости от быстродействия вашего компьютера и сложности процесса, сборка может занять от одной-двух минуты до нескольких часов для сложных проектов.

Иногда сборка может не работать. Наиболее общие причины, это:
Отсутствие необходимых пакетов
Не та версия необходимых пакетов
Не верные значения параметров, которые вы должно быть пропустили в configure или make.
Отсутствие компилятора.
Ошибки в скрипте configure или созданном Makefile.
Ошибки в исходном коде.
В нашем примере с Dr Geo, одна из таких проблем была обнаружена на этапе конфигурирования, но это не обычный случай. По мере накопления опыта работы в Linux, вы сможете определить и исправить эти проблемы. Иногда вам придется обратиться к FAQ [Прим.пер.: Frequently Asking Qwestions -- ЧАсто задаваемые ВОпросы. В последнее время в Рунете это преобразуется в рускоязычное ЧАВО] или списку рассылки о поддержке данного пакета. В других случаях вам может понадобиться определить, что вы пропустили и установить это.

Установка

Если при сборке все прошло хорошо, то вы готовы к установке. На этапе компиляции были собраны все необходимые файлы, но они все еще расположены не в том месте, чтобы быть готовыми к использованию. Например, бинарные файлы необходимо скопировать в /usr/local/bin, а man страницы в /usr/local/man и т.д.

Если вы не указывали опцию --prefix, то несколько файлов и каталогов скорее всего будет скопировано в древо /usr/local. Вам потребуются права

суперпользователя для записи в древо /usr/local вашей файловой системы. Если вы вошли не как суперпользователь (root), то используйте команду su для получения прав суперпользователя. Будет запрошен ввод пароля root. Затем используйте команду make install, чтобы установить только что собранную программу. Установка занимает от нескольких секунд до минут, в зависимости от размера программы. Мы привели часть выведенного при установке Dr Geo в Листинге 18.

Листинг 18. Установка Dr Geo.

```
[ian@attic4 drgeo-1.1.0]$ su
Password:
[root@attic4 drgeo-1.1.0]# make install
Making install in po
make[1]: Entering directory `/home/ian/drgeo-1.1.0/po'
if test -n ""; then \
   /usr/local/share; \
else \
  /bin/sh ../mkinstalldirs /usr/local/share; \
fi
installing az.gmo as /usr/local/share/locale/az/LC_MESSAGES/drgeo.mo
installing ca.gmo as /usr/local/share/locale/ca/LC_MESSAGES/drgeo.mo
installing cs.gmo as /usr/local/share/locale/cs/LC_MESSAGES/drgeo.mo
installing da.gmo as /usr/local/share/locale/da/LC_MESSAGES/drgeo.mo
installing de.gmo as /usr/local/share/locale/de/LC_MESSAGES/drgeo.mo
installing el.gmo as /usr/local/share/locale/el/LC_MESSAGES/drgeo.mo
installing en_CA.gmo as
/usr/local/share/locale/en_CA/LC_MESSAGES/drgeo.mo
installing en_GB.gmo as
/usr/local/share/locale/en_GB/LC_MESSAGES/drgeo.mo
...
/usr/bin/install -c drgeo /usr/local/bin/drgeo
/bin/sh ./mkinstalldirs /usr/local/share/applications
 /usr/bin/install -c -m 644 drgeo.desktop
/usr/local/share/applications/drgeo.desktop
make[2]: Leaving directory `/home/ian/drgeo-1.1.0'
make[1]: Leaving directory `/home/ian/drgeo-1.1.0'
[root@attic4 drgeo-1.1.0]# exit
exit
[ian@attic4 drgeo-1.1.0]$
```

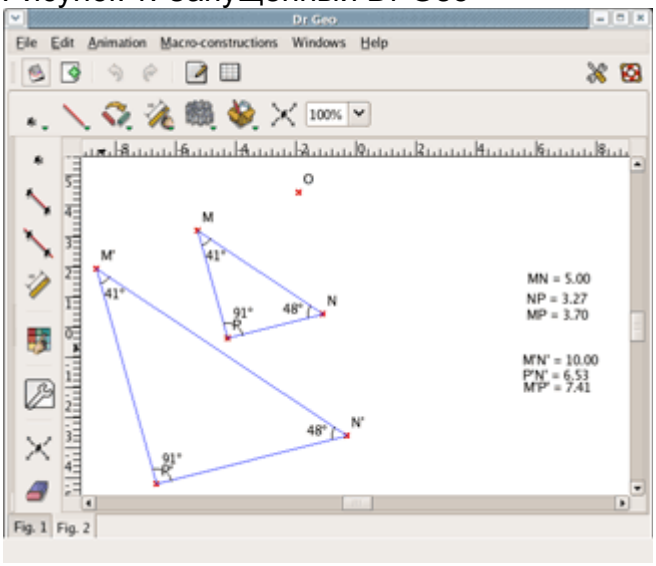
После копирования файлов, команда make install должна также убедиться, что все установленные файлы имеют корректные права доступа и разрешения. После завершения установки программа считается установленной и готовой к использованию, или готовой к предварительной настройке перед использованием.

Замечание: Если вы имеете права суперпользователя, то можно очень легко, допустив ошибку, нанести большой вред системе, поэтому не забудьте завершить режим суперпользователя командой exit или нажатием ctrl-d в командной оболочке bash.

Запуск программ

Если ваша программа готова к запуску, то вы можете попробовать запустить ее, набрав в командной строке ее имя, для нашего примера drgeo. На Рисунке 1 показано окно Dr Geo, отображающее один из примеров, поставляемых вместе с программой.

Рисунок 1. Запущенный Dr Geo



Вот что еще вам следует сделать перед запуском программы.
Прочтите man страницы если они есть в пакете. Попытайтесь выполнить man имя_программы.
Настроить конфигурационные файлы, например в /etc.
Настроить автоматический запуск для такой программы, как демон сервера.

В этом разделе мы рассмотрели процесс установки программы из исходных тектсов от самого начала и до конца. В следующих разделах мы поговорим о библиотеках, управлении библиотеками и пакетами, а также о том, как устанавливать их.

Управление библиотеками совместного доступа

Статические и динамические исполняемые файлы

В системах Linux имеется два типа исполняемых программ.

1. Исполняемые файлы статической компоновки (Statically linked) содержат все функции библиотек, которые необходимы им для работы. Все библиотечные функции включены в исполняемый файл. Это полные программы, которым не нужны внешние библиотеки для запуска. Одним из преимуществ статически скомпонованных программ является то, что они будут работать без установки зависимостей.
2. Исполняемые файлы динамической компоновки (Dynamically linked) намного меньше по размеру, поскольку они не полные, в том смысле, что для запуска им необходимы функции из внешних совместно используемых (shared) библиотек. Кроме того, что они меньше по размеру, динамическая компоновка позволяет пакету указать от каких библиотек он зависит, без необходимости включения этих библиотек в пакет. Использование

динамической компоновки также позволяет многим рабочим программам совместно использовать одну копию библиотеки, вместо того, чтобы занимать память множеством копий одного и того же кода. По этим причинам большинство программ на сегодняшний день используют динамическую компоновку.

Интересным примером типичной системы Linux является команда ln (/bin/ln), создающая связи между файлами или жесткие (hard) связи, или мягкие (soft) (или символические (symbolic)) связи (ссылки). Совместно используемые библиотеки часто используют символические ссылки между универсальным именем и конкретной версией библиотеки. При этом если нарушить некоторые ссылоки на динамические библиотеки, то команда ln (с помощью которой эти ссылки можно быть бы восстановить) тоже становится неработоспособной. Во избежание таких случаев, системы Linux содержат статически скомпонованную версию программы ln под именем sln (/sbin/sln). Листинг 19 иллюстрирует большую разницу в размере между двумя этими программами.

Листинг 19. Размеры sln и ln.

```
[ian@lyrebird ian]$ ls -l /sbin/sln; ls -l /bin/ln
-rwxr-xr-x    1 root     root       457165 Feb 23  2005 /sbin/sln
-rwxr-xr-x    1 root     root        22204 Aug 12  2003 /bin/ln
```

Команда ldd

Не принимая во внимание то, что статически скомпонованная программа имеет большой размер. Как мы можем определить является ли программа скомпонованой статически? И если она скомпонована динамически, как нам узнать какие библиотеки ей нужны? Ответ на оба вопроса это команда ldd, отображающая информацию о требуемых библиотеках для исполняемой программы. В Листинге 20 показан вывод команды ldd для исполняемых ln и sln.

Листинг 20. Вывод программы ldd для sln и ln.

```
[ian@lyrebird ian]$ ldd /sbin/sln /bin/ln
/sbin/sln:
        not a dynamic executable
/bin/ln:
        libc.so.6 => /lib/tls/libc.so.6 (0x00ebd000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00194000)
```

Поскольку ldd в действительности предназначен для динамической компоновки она сообщает нам, что sln является статически скомпонованной говоря, что это "not a dynamic executable" (не динамический исполняемый файл), в то время как для ln приводятся имена двух совместно используемых библиотек (libc.so.6 и ld-linux.so.2), которые ей необходимы, а также где их можно найти. Отметим, что .so указывает на то, что обе они являются совместно используемыми объектами (shared objects) или динамическими библиотеками. Для Листинга 21 мы воспользовались командой ls -l, чтобы показать, что это действительно символическая ссылка на конкретные версии библиотек.

Листинг 21. Символические ссылки на библиотеки.

```
[ian@lyrebird ian]$ ls -l /lib/tls/libc.so.6; ls -l /lib/ld-linux.so.2
lrwxrwxrwx    1 root   root       13 May 18 16:24 /lib/tls/libc.so.6
-> libc-2.3.2.so
lrwxrwxrwx    1 root   root       11 May 18 16:24 /lib/ld-linux.so.2
-> ld-2.3.2.so
```

Динамическая загрузка

Исходя из предыдущего, вы можете удивиться, узнав, что ld-linux.so, которая выглядит как библиотека совместного использования, на самом деле по своей природе является исполняемым файлом. Это код, отвечающий за динамическую загрузку. Он читает информацию заголовка исполняемого файла, приведенный в формате Executable and Linking Format (формат ссылок и исполняемых) или (ELF). Из этой информации можно определить какие библиотеки необходимы и какие следует загрузить. Затем он осуществляет динамическое связывание для согласования указателей на адреса в вашем исполняемом файле и загруженных библиотеках, для того, чтобы программа запустилась.

Вы не сможете найти man страницы для ld-linux.so, но вы можете обнаружить их для ld.so, выполнив man ld.so. Листинг 22 иллюстрирует использование опции --list для ld-linux.so, чтобы показать ту же информацию для команды ln, что мы выводили для команды ldd в Листинге 20.

Листинг 22. Использование ld-linux.so для отображения требований для библиотеки.

```
[ian@lyrebird ian]$ /lib/ld-linux.so.2 --list /bin/ln
        libc.so.6 => /lib/tls/libc.so.6 (0x00a83000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00f2c000)
```

Отметим что шестнадцатиричные адреса для двух листингов отличаются. Скорее всего они будут различны даже если вы запустите ldd дважды.

Настройка динамических библиотек

Итак, как же динамический загрузчик узнает где искать исполняемые файлы? Как и для многого в Linux в /etc имеется конфигурационный файл (configuration file). Фактически два конфигурационных файла: /etc/ld/so/conf и /etc/ld.so.cache. Листинг 23 показывает содержимое /etc/ld.so.conf для двух различных систем. Отметим, что в системе attic4 (под управлением Fedora Core 4), /etc/ld.so.conf указывает, что в подкаталоге ld.so.conf.d должны учитываться все файлы .conf. Реальное содержимое /etc/ld.so.conf для вашей системы может отличаться.

Листинг 23. Содержимое /etc/ld.so.conf.

```
[ian@lyrebird ian]$ cat /etc/ld.so.conf
/usr/kerberos/lib
/usr/X11R6/lib
/usr/lib/qt-3.1/lib
[
[ian@attic4 ~]$ cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
```

Необходимо, чтобы загрузка программ проходила быстро, поэтому файл ld.so.conf обрабатывается командой ldconfig для обработки всех библиотек из ld.so.conf.d, а также из надежных каталогов, /lib и /usr/lib. Динамический загрузчик использует файл ld.conf.cache для определения файлов, которые необходимо динамически загрузить и связать. Если вы измените ld.co.conf (или добавите новые включаемые (included) файлы в ld.so.conf.d, то вы должны выполнить команду ldconfig (от имени суперпользователя) чтобы перестроить файл ld.conf.cache.

Обычно вы используете команду ldconfig без параметров для перестройки ld.so.cache. Имеется также несколько параметров, которые вы можете указать для переопределения этого поведения по умолчанию. Как всегда выполните man ldconfig для получения большей информации. Мы проиллюстрировали использование параметра -p для отображения содержимого ld.so.cache в Листинге 24.

Листинг 24. Использование ldconfig для отображения ld.so.cache.

```
[ian@lyrebird ian]$ /sbin/ldconfig -p | more
768 libs found in cache `/etc/ld.so.cache'
        libzvt.so.2 (libc6) => /usr/lib/libzvt.so.2
        libz.so.1 (libc6) => /usr/lib/libz.so.1
        libz.so (libc6) => /usr/lib/libz.so
        libx11globalcomm.so.1 (libc6) =>
/usr/lib/libx11globalcomm.so.1
        libxsltbreakpoint.so.1 (libc6) =>
/usr/lib/libxsltbreakpoint.so.1
        libxslt.so.1 (libc6) => /usr/lib/libxslt.so.1
        libxmms.so.1 (libc6) => /usr/lib/libxmms.so.1
        libxml2.so.2 (libc6) => /usr/lib/libxml2.so.2
        libxml2.so (libc6) => /usr/lib/libxml2.so
        libxmltok.so.0 (libc6) => /usr/lib/libxmltok.so.0
        libxmlparse.so.0 (libc6) => /usr/lib/libxmlparse.so.0
        libxml.so.1 (libc6) => /usr/lib/libxml.so.1
        libxerces-c.so.24 (libc6) => /usr/lib/libxerces-c.so.24
        ...
        lib-gnu-activation-20030319.so (libc6) => /usr/lib/lib-gnu-
activation-20030319.so
        ld-linux.so.2 (ELF) => /lib/ld-linux.so.2
```

Если вы запускаете старое приложение, которому требуется некая старая версия библиотеки или если вы разрабатываете новую библиотеку или новую версию библиотеки, то вам может понадобиться переопределить путь для поиска по умолчанию, используемый загрузчиком. Это может также понадобиться скриптам, использующим специфичные для продукта библиотеки, устанавливаемые в древо /opt.

Также как вы устанавливаете переменную PATH, вы можете указать путь поиска для исполняемых файлов, вы можете задать переменную LD_LIBRARY_PATH со списком каталогов, разделенных двоеточием, в которых следует искать библиотеки перед тем, как система станет искать их в ld.so.cache. Например, вы можете использовать команду вроде
export LD_LIBRARY_PATH=/usr/lib/oldstuff:/opt/IBM/AgentController/lib

В оставшихся разделах этого учебника мы рассмотрим управление пакетами.

Управление пакетами от Debian

Обзор управления пакетами

В примере с Dr Geo предыдущего раздела, наши конфигурационные шаги сначала привели к ошибке, потому что у нас не была установлена требуемая программа. Инструменты управления пакетами формализуют указание требований и версий, стандартизируют их размещение в системе, а также обеспечивают механиз отслеживания, помогающий определить установленные пакеты. В результате получаем облегчение установки программного обеспечения, его поддержки и удаления.

Хотя вы все еще можете иметь желание устанавливать программы из исходных текстов по перечисленными в предыдущем разделе причинам, но вы, вероятно, большую часть поддержки системы и установки программ выполняете с использованием менеджера пакетов, который имеется в вашем дистрибутиве.

С точки зрения пользователя, основные функции управления пакетами обеспечиваются на уровне команд. Поскольку Linux-разработчики стараются сделать использование Linux легче, то основные инструменты снабжаются другими надстройками, включая графический интерфейс, скрывающий сложность базовых средств от конечного пользователя. В этих двух разделах мы сосредоточимся на базовых средствах, хотя и упомянем некоторые другие инструменты, чтобы вы имели стартовую платформу для их изучения.

Установка пакетов Debian

Давайте вернемся к проблемам, с которыми мы столкнулись при работе с исходными текстами Dr Geo. Так получилось, что эти проблемы возникли в системе Fedora Core 4, использующей управление пакетами RPM. К счастью в этом разделе учебника, я также не досчитался нескольких компонентов guile в системе Ubuntu, основанной на Debian, в которой я пытался установить Dr Geo. Возникшие ошибки для этого случая приведены в Листинге 25.

Листинг 25. Отсутствие функции guile.

```
ian@attic4:~$ cd drgeo-1.1.0
ian@attic4:~/drgeo-1.1.0$ ./configure
checking for perl... /usr/bin/perl
checking for XML::Parser... ok
checking for iconv... /usr/bin/iconv
checking for msgfmt... /usr/bin/msgfmt
...
checking for guile... no
configure: error: guile required but not found
i
```

Пакет, который необходим нам -- это пакет guile. Мы можем установить его используя команду apt-get, как показано в Листинге 26. Отметим, что использование команды sudo является обычным для Ubuntu способом работать с правами суперпользователя (root).

Листинг 26. Установка guile с использованием apt-get.

```
ian@attic4:~$ sudo apt-get install guile
Reading package lists... Done
Building dependency tree... Done
Note, selecting guile-1.6 instead of guile
Suggested packages:
  guile-1.6-doc
The following NEW packages will be installed:
  guile-1.6
0 upgraded, 1 newly installed, 0 to remove and 24 not upgraded.
Need to get 31.5kB of archives.
After unpacking 209kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com hoary/main guile-1.6 1.6.7-1ubuntu1
[31.5kB]
Fetched 31.5kB in 0s (37.4kB/s)

Preconfiguring packages ...
Selecting previously deselected package guile-1.6.
(Reading database ... 84435 files and directories currently
installed.)
Unpacking guile-1.6 (from .../guile-1.6_1.6.7-1ubuntu1_i386.deb) ...
Setting up guile-1.6 (1.6.7-1ubuntu1) ...
i
```

Из приведенного листинга мы видим, что apt-get откуда-то читает список пакетов (подробнее об этом далее), строит древо зависимостей, определяет, что guile-doc рекомендовано устанавливать вместе с guile, и загружает пакет guile из сети Интернет. Затем пакет guile распаковывается, устанавливается и настраивается. Отметим, что расширение, используемое для пакетов Debian, это .deb. Полное имя нашего пакета guile выглядит так guile-1.6_1.6.7-1ubuntu1_i386.deb.

Если apt-get обнаруживает, что пакет, который вы пытаетесь установить зависит от других пакетов, то он автоматически загружает и устанавливает также и их. В нашем примере устанавливается только guile, потому что все зависимости уже удовлетворены. Следуя выведенному совету мы можем установить guile-doc (или guile-1.6.doc).

Предположим, что вместо установки guile-doc, мы хотим узнать зависит ли пакет guile-doc от других пакетов. Мы можем использовать опцию apt-get -s (для симулирования). Имеется и несколько других опций со сходными функциями, такие как --just-print и --dry-run. Обратитесь к man страницам за подробностями. Не удивительно, что документация для пакета, который мы только что установили не имеет каких-либо зависимостей, поэтому в Листинге 27 мы привели несколько больше полезных примеров симулировав установку пакета ssl-cert, которому требуется пакет openssl.

Листинг 27. Симуляция или пробный прогон установки ssl-cert.

```
ian@attic4:~$ sudo apt-get -s install ssl-cert
Reading package lists... Done
Building dependency tree... Done
The following extra packages will be installed:
```

```
  openssl
Suggested packages:
  ca-certificates
The following NEW packages will be installed:
  openssl ssl-cert
0 upgraded, 2 newly installed, 0 to remove and 24 not upgraded.
Inst openssl (0.9.7e-3 Ubuntu:5.04/hoary)
Inst ssl-cert (1.0-11 Ubuntu:5.04/hoary)
Conf openssl (0.9.7e-3 Ubuntu:5.04/hoary)
Conf ssl-cert (1.0-11 Ubuntu:5.04/hoary)
```

Мы видим, что требуется еще два новых пакета и вследствии этого они были установлены и настроены.

Список пакетных ресурсов: apt-setup

Мы говорили, что apt-get откуда-то читает список пакетов. Читает его он из /etc/apt/sources.list. Это список, который вы можете изменять самостоятельно, но вероятно вы предпочтете настроить его, используя команду apt-setup. Команда apt-setup это интерактивный инструмент, который знает расположение главных APT репозиториев. Вы можете иметь доступ к источникам пакетов на CD-ROM, в вашей локальной файловой системе или в сети с использованием http или ftp.

Если ваш дистрибутив установил для вас файл /etc/apt/sources.list, то он может и не содержать ваш CD-ROM в качестве источника пакетов. Это может быть неудобно особенно на начальной стадии экспериментирования с новой системой, когда вы можете захотеть добавить много пакетов, большинство которых еще не обновились. В этом случае вы можете воспользоваться командой

```
apt-cdrom add
```

чтобы добавить ваш CD-ROM к списку источников пакетов.

Apt-get и другие средства о которых мы будем говорить используют локальную базу данных для определения установленных пакетов. Они могут сверять установленные версии с доступными. Для этого информация о доступных версиях загружается с указанного в списке /etc/apt/sources.list источника и сохраняется на компьютере локально. Если вы обновляете ваш файл /etc/apt/sources.list, то вам следует выполнить

```
apt-get update
```

Это приведет сохраненные данные о доступных пакетах в актуальное состояние. Вообще же вы должны всегда делать это перед тем как установить новый пакет.

Удаление или обновление пакетов.

перед тем как покинуть apt-get, мы рассмотрим две другие полезные опции.

Если вы установили пакет и позднее хотите удалить его используйте опцию apt-get remove. Листинг 28 показывает как удалить пакет guile, который мы установили ранее.

Листинг 28. Удаление пакета guile.

```
ian@attic4:~$ sudo apt-get remove guile
Reading package lists... Done
Building dependency tree... Done
Note, selecting guile-1.6 instead of guile
The following packages will be REMOVED:
  guile-1.6
0 upgraded, 0 newly installed, 1 to remove and 24 not upgraded.
Need to get 0B of archives.
After unpacking 209kB disk space will be freed.
Do you want to continue [Y/n]? Y
(Reading database ... 84455 files and directories currently
installed.)
Removing guile-1.6 ...
```

Другой опцией, которую мы хотим упомянуть является опция upgrade Эта опция обновляет все установленные пакеты в вашей систем до новейших версий. Не путайте ее с опцией update, которая просто обновляет информацию о доступных пакетах.

За более подробной информацией о возможностях и опциях apt-get обратитесь к man странице.

Файл apt.conf

Если вы просмотрите man страницу apt-get, то обнаружите множество опций. Если вы используете команду apt-get постоянно и поняли, что опции по умолчанию вас не устраивают, то вы можете создать новые умолчания в /etc/apt/apt.conf. Программа apt-config доступна в виде скриптов для опроса файла apt.conf. За более подробной информацией обратитесь к man страницам для apt.conf и apt-config.

Информация пакета Debian

Теперь мы обратимся к нескольким инструментам для получения информации о пакете. Некоторые из этих инструментов также выполняют и другие функции, но мы сосредоточимся на информационных аспектах.

Статус пакета с dpkg

Еще один инструмент являющийся частью системы APT это dpkg. Это инструмент среднего уровня для управления пакетами, который может устанавливать и удалять пакеты, а также отображать их информацию. Конфигурирование dpkg может выполняться при помощи /etc/dpkg/dpkg.cfg. Отдельные пользователи могут также найти файл .dpkg.cfg в своем собственном домашнем каталоге, что обеспечивает дополнительное конфигурирование. Если у вас нет ни одного из этих файлов, то проверьте, например, /usr/share/doc/dpkg/dpkg.cfg.

Инструмент dpkg использует многие файлы в ветке /var/lib/dpkg вашей файловой системы. В частности, файл /var/lib/dpkg/status содержит статус информации о пакетах вашей системы. Листинг 29 показывает использование dpkg -s для

отображения статуса пакета guile после его установки. Напомним, что мы действительно установили guile-1.6. Из Листинга 29 мы видим, что нам нужно указывать полное имя, а не сокращенное.

Листинг 29. Статус пакета guile.

```
ian@attic4:~$ dpkg -s guile
Package `guile' is not installed and no info is available.

Use dpkg --info (= dpkg-deb --info) to examine archive files,
and dpkg --contents (= dpkg-deb --contents) to list their contents.
ian@attic4:~$ dpkg -s guile-1.6
Package: guile-1.6
Status: install ok installed
Priority: optional
Section: interpreters
Installed-Size: 204
Maintainer: Rob Browning <rlb@defaultvalue.org>
Architecture: i386
Version: 1.6.7-1ubuntu1
Provides: guile
Depends: guile-1.6-libs, libc6 (>= 2.3.2.ds1-4), libguile-ltdl-1
Suggests: guile-1.6-doc
Conflicts: libguile-dev (<= 1:1.4-24)
Description: The GNU extension language and Scheme interpreter
 Guile is a Scheme implementation designed for real world programming,
 providing a rich Unix interface, a module system, an interpreter, and
 many extension languages.  Guile can be used as a standard #! style
 interpreter, via #!/usr/bin/guile, or as an extension language for
 other applications via libguile.
```

Пакеты и файлы в них

Часто мы хотим знать, что находится в пакете или к какому пакету принадлежит тот или иной файл. Обе эти задачи для dpkg. Листинг 30 иллюстрирует использование dpkg -L для вывода списка файлов (включая каталоги) установленных из пакета guile.

Листинг 30. Что находится в пакете guile?

```
root@attic4:~# dpkg -L guile-1.6
/.
/usr
/usr/bin
/usr/bin/guile-1.6-snarf
/usr/bin/guile-1.6-tools
/usr/bin/guile-1.6
/usr/bin/guile-1.6-config
/usr/share
/usr/share/guile
/usr/share/guile/1.6
/usr/share/guile/1.6/scripts
/usr/share/guile/1.6/scripts/autofrisk
/usr/share/guile/1.6/scripts/display-commentary
/usr/share/guile/1.6/scripts/doc-snarf
/usr/share/guile/1.6/scripts/frisk
/usr/share/guile/1.6/scripts/generate-autoload
```

```
/usr/share/guile/1.6/scripts/lint
/usr/share/guile/1.6/scripts/PROGRAM
/usr/share/guile/1.6/scripts/punify
/usr/share/guile/1.6/scripts/read-scheme-source
/usr/share/guile/1.6/scripts/snarf-check-and-output-texi
/usr/share/guile/1.6/scripts/snarf-guile-m4-docs
/usr/share/guile/1.6/scripts/use2dot
/usr/share/doc
/usr/share/doc/guile-1.6
/usr/share/doc/guile-1.6/copyright
/usr/share/doc/guile-1.6/changelog.Debian.gz
/usr/lib
/usr/lib/menu
/usr/lib/menu/guile-1.6
```

Чтобы найти пакет, содержащий какой-то конкретный файл, используйте опцию dpkg -S как показано в Листинге 31. Имя пакета выводится слева.

Листинг 31. Какой пакет содержит файл?

```
ian@attic4:~$ dpkg -S /usr/share/guile/1.6/scripts/lint
guile-1.6: /usr/share/guile/1.6/scripts/lint
```

Вы можете отметить, что список в Листинге 30 не содержит /usr/bin/guile, тогда как команда which guile говорит, что это программа, которая запустится если набрать guile. Когда такое случается вам может понадобиться выполнить некоторую поисковую работу, чтобы найти откуда пришел пакет. Например, на этапе установки могут выполняться такие задачи, как создание символьных ссылок, которые не указываются в качестве составляющих пакета. Последним добавлением в систему Linux является система alternatives которая управляется командой update-alternatives. В Листинге 32, мы показали как использовать команду ls, чтобы увидеть с чем связана команда guile при помощи символьной ссылки. Ссылка на каталог /etc/alternatives подсказывает, что мы используем систему alternatives, и потому мы используем команду update-alternatives, чтобы найти больше информации и в конце концов мы сможем использовать команду dpkg -S для подтверждения, что команда guile получена из пакета guile-1.6. Настройка системы alternatives может быть сделана постустановочным скриптом, являющимся частью пакета guile-1.6.

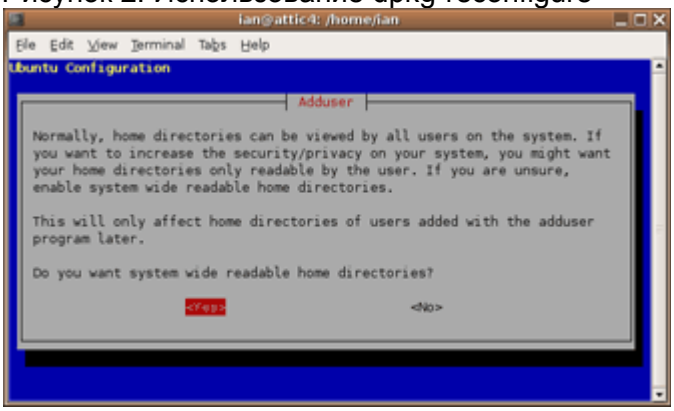
Листинг 32. Более сложное использование dpkg -S

```
ian@attic4:~$ ls -l $(which guile)
lrwxrwxrwx  1 root root 23 2005-09-06 23:38 /usr/bin/guile ->
/etc/alternatives/guile
ian@attic4:~$ update-alternatives --display guile
guile - status is auto.
 link currently points to /usr/bin/guile-1.6
/usr/bin/guile-1.6 - priority 160
 slave guile-config: /usr/bin/guile-1.6-config
 slave guile-snarf: /usr/bin/guile-1.6-snarf
 slave guile-tools: /usr/bin/guile-1.6-tools
Current `best' version is /usr/bin/guile-1.6.
ian@attic4:~$ dpkg -S /usr/bin/guile-1.6
guile-1.6: /usr/bin/guile-1.6
```

Перенастройка пакетов Debian.

APT содержит функцию с именем debconf, которая используется для настройки пакетов после их установки. Пакеты, использующие эту функцию (а используют не все) могут быть перенастроены после того, как они уже установлены. Самый легкий способ сделать это -- использовать команду dpkg-reconfigure. Например, команда adduser может создать домашние каталоги, которые сможет просматривать любой пользователь системы. Вам это может не понравиться по соображениям безопасности. На Рисунке 2 приведены вопросы по настройке, соответствующие пакету adduser. Выполните dpkg-reconfigure adduser (от имени root) для вывода этого окна.

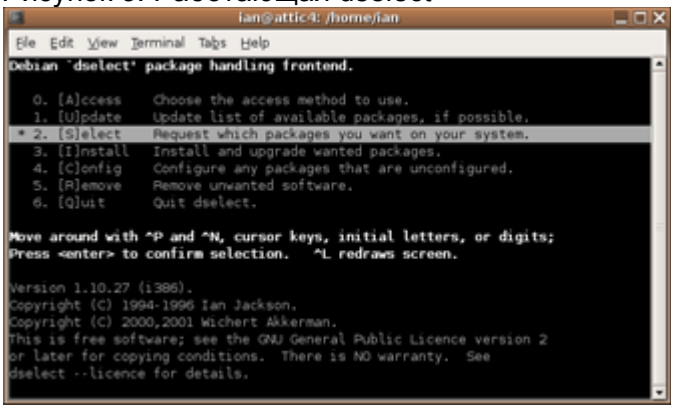
Рисунок 2. Использование dpkg-reconfigure



Использование dselect

Ранее мы отмечали, что статус пакетов хранится в /var/lib/dpkg/status. Мы также указывали, что dpkg может делать больше, чем просто отображать информацию о пакете. Теперь мы кратко рассмотрим команду dselect, предоставляющую текстовый полноэкранный интерфейс (используя ncurses) к функциям dpkg для управления пакетами. Вы можете использовать dselect для установки или удаления пакетов, а также для управления флагами статуса, показывающими должен ли пакет поддерживаться в актуальном состоянии или оставить его в текущем состоянии. Если вы выполните команду dselect (от имени root), то вы увидите экран, похожий на приведенный на Рисунке 3.

Рисунок 3. Работающая dselect

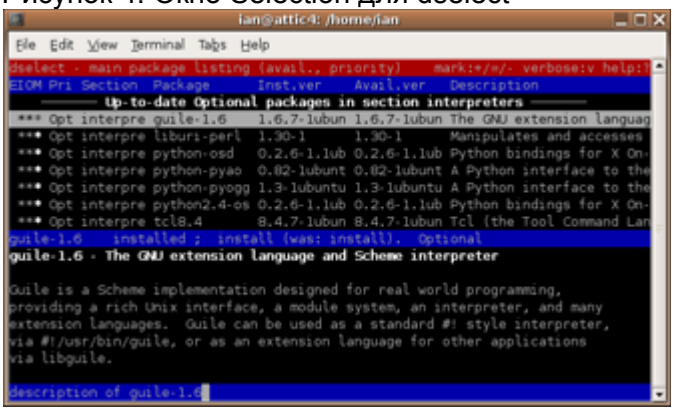


Использование режима Select в dselect

Вы можете просматривать и изменять статус каждого пакета выбрав опцию Select подсвеченную на Рисунке 3. Вы увидите окно справки. Для выхода из справки в любой момент нажмите Пробел. Затем вы увидите список пакетов и групп пакетов.

Вы можете искать пакеты, используя / в конце строки поиска. На Рисунке 4 показан пример результат поиска для "guile".

Рисунок 4. Окно Selection для dselect



Состояния выбора пакета

Состояние выбора для каждого пакета может быть определено по скрытому заголовку EIOM. Эти буквы расшифровываются как Error (ошибка), Installed state (состояние Установлено), Old mark (помечен как старый) и Mark (помечен). Вы можете использовать клавишу "v" для переключения между краткой формой отображения этой информации и отображением этого в виде слов.

Четвертую колонку, или M, мы рассмотрим повнимательнее. Она описывает что случится после того, как мы закончим работу с окном выделения и перенесемся в окно установки. Пометки имеют следующее значение:
*
Установить или обновить до последней версии
=
Оставить пакет с текущим статусом и версией
- (дефис)
Удалить пакет, но оставить его настройки для случая повторной переустановки позднее
_ (подчеркивание)
Удалить пакет вместе с настройками.

Для изменения пометок нажмите соответствующую клавишу, за исключением того, что следует нажимать клавишу "+" чтобы пометить пакет для установки или обновления. По завершении нажмите Enter для подтверждения сделанных изменений или нажмите "X" (заглавную X) для отмены изменений без сохранения. Это вернет вас к окну, изображенному на Рисунке 3, с выбранной опцией Install (Установка). Нажмите Enter для установки или обновления вашей системы.

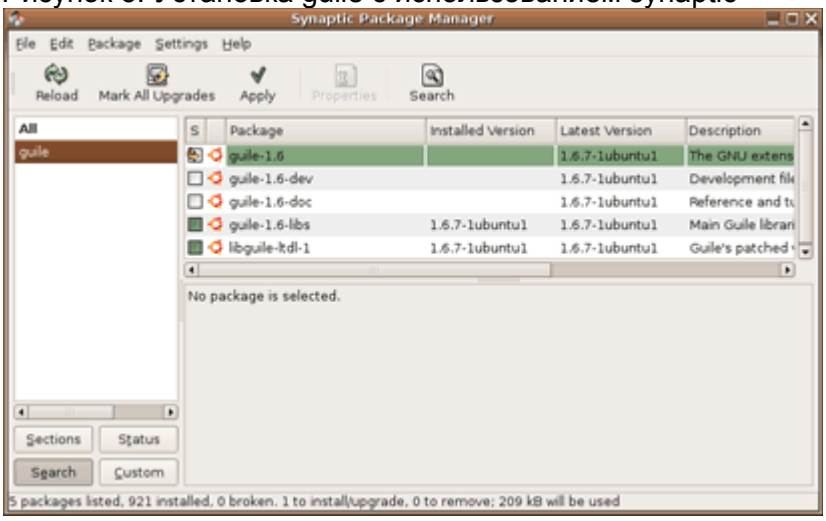
Если вам понадобится помощь, то в любой момент нажмите "?" (знак вопроса). Нажмите Пробел для выхода из справки

Обновление Debian другими средствами

Мы увидели, что dselect может помочь установить или удалить отдельные пакеты, а также обновить все пакеты вашей системы до последних версий. Если вы хотите сделать это из командной строки, то можете использовать apt-get dselect-upgrade, что обработает пометки, которые, как мы видели, настраиваются при помощи dselect.

Кроме dselect существует несколько других интерактивных интерфейсов управления пакетами для систем Debian, включая aptitude, synaptic, gnome-apt и wajig. Synaptic это графическое приложение для использования в оконной системе X. На рисунке 5 показан интерфейс пользователя synaptic с нашим старым другом -- пакетом guile, помеченным для установки.

Рисунок 5. Установка guile с использованием synaptic



Кнопка Apply (Применить) установит guile и обновит все остальные пакеты, включенные в список на обновление. Кнопка Reload (Перезагрузить) обновит список пакетов. Если вы используете графический интерфейс, то возможно synaptic будет для вас более удобна в использовании нежели apt-get, dpkg или dselect.

Поиск пакетов Debian

В нашем последнем пункте об управлении пакетами в Debian, мы рассмотрим способы поиска пакетов. Обычно apt-get и другие рассмотренные нами инструменты, уже знают все пакеты, которые могут понадобиться вам из списка доступных пакетов. Команда, которую мы еще не использовали это apt-cache, она полезна для поиска информации о пакете в вашей системе. Apt-cache может искать регулярные выражения (о которых мы поговорим подробно в учебнике по теме 103). Предположим вам нужно найти имя пакета, содержащего загрузчик Linux. Листинг 33 показывает как можно сделать это.

```
Листинг 33. Поиск загрузчика Linux при помощи apt-cache

ian@attic4:~$ apt-cache search "linux loader"
lilo - LInux LOader - The Classic OS loader can load Linux and others
lilo-doc - Documentation for LILO (LInux Loader)
```

Ранее мы видели, что dselect и synaptic также предоставляют средства поиска. Если вы используете synaptic, то заметьте, что в меню поиска имеются опции, при помощи которых вы можете указать область поиска: только имена или также описания пакетов.

Если вы все еще не можете найти пакет, то поищите в списке пакетов на сайте Debian (смотри Ресурсы), или еще где-то в Интернет.

Если вы выполнили поиск и загрузили .deb файл, то вы можете установить его, используя dpkg -i. Например, Dr Geo может быть найден в виде .deb пакета в официальном репозитории пакетов Debian.

Листинг 34. Установка Dr Geo из .deb пакета

```
ian@attic4:~$ ls drg*.deb
drgeo_1-1.0.0-1_i386.deb
ian@attic4:~$ sudo dpkg -i drgeo_1-1.0.0-1_i386.deb
Password:
Selecting previously deselected package drgeo.
(Reading database ... 84435 files and directories currently installed.)
Unpacking drgeo (from drgeo_1-1.0.0-1_i386.deb) ...
Setting up drgeo (1.0.0-1) ...
```

Заметьте, что архив с исходными текстами имеет большую версию (1.1.0), нежели deb пакет (1.0.0-1). Если вы установили Dr Geo и по некоторым причинам он не работает, то возможно вам придется установить его из исходных текстов

Если ничего не помогло, то существует еще один возможный источник пакетов. Предположим, что вы нашли программу упакованную в RPM, а не .deb. Вы можете использовать программу alien, которая может осуществлять преобразование из одного формата пакетов в другой. Вам следует тщательно прочитать документацию alien поскольку не все возможности систем управления пакетами alien может преобразовать в другой формат.

Существует намного больше систем управления пакетами в Debian, нежели описано здесь. А также в Debian имеется многое кроме системы управления пакетами. Дополнительные ссылки смотри в Ресурсах.

Менеджер пакетов Red Hat (RPM)

Установка и удаление пакетов RPM

Как и в предыдущем разделе мы рассмотрим проблемы, обнаружившиеся при установке Dr Geo в системе Fedora Core 4 в разделе Компиляция и установка программ. Вы можете вернуться к Листингу 17, в котором мы пропустили команду guile-config.

.

Введение в rpm

Команда rpm может устанавливать пакеты из локальной файловой системы или из Интернет, используя http или ftp. В Листинге 35 показана установка пакета guile-devel с использованием команды rpm -ivh и сетевого источника пакета.

Листинг 35. Установка guile-devel при помощи rpm

```
[root@attic4 ~]# rpm -ivh
http://download.fedora.redhat.com/pub/fedora\
> /linux/core/4/i386/os/Fedora/RPMS/guile-devel-1.6.7-2.i386.rpm
Retrieving
http://download.fedora.redhat.com/pub/fedora/linux/core/4/i386/os/Fedo
ra/
RPMS/guile-devel-1.6.7-2.i386.rpm
Preparing...
########################################### [100%]
   1:guile-devel
########################################### [100%]
```

Отметим, что опция -v предоставляет подробный вывод, а опция -h показывает знак "решетка" (#), для индикации прогресса. Если вы хотите перед установкой из сети проверить пакет, то возможно вам придется сначала загрузить его и только потом установить. Мы поговорим о проверке пакетов чуть позже, а сейчас давайте используем команду wget для получения пакета с его последующей установкой из локальной файловой системы без использования опций -vh. Вывод показан в Листинге 36.

Листинг 36. Установка guile-devel из файла

```
[root@attic4 ~]# wget http://download.fedora.redhat.com/pub/fedora/\
> linux/core/4/i386/os/Fedora/RPMS/guile-devel-1.6.7-2.i386.rpm
--22:29:58--
http://download.fedora.redhat.com/pub/fedora/linux/core/4/i386/os/Fedo
ra/
RPMS/guile-devel-1.6.7-2.i386.rpm
           => `guile-devel-1.6.7-2.i386.rpm'
Resolving download.fedora.redhat.com... 209.132.176.221
Connecting to download.fedora.redhat.com[209.132.176.221]:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 481,631 [application/x-rpm]

100%[====================================>] 481,631      147.12K/s
ETA 00:00

22:30:02 (140.22 KB/s) - `guile-devel-1.6.7-2.i386.rpm' saved
[481,631/481,631]

[root@attic4 ~]# ls guil*
guile-devel-1.6.7-2.i386.rpm
[root@attic4 ~]# rpm -i guile-devel-1.6.7-2.i386.rpm
```

Ни решеток, ни сообщений.

Переустановка rpm-пакета

Если вы самостоятельно выполнили указанные выше команды, то на втором шаге вы вероятно увидели ошибку (или на первом, если в вашей системе пакет guile-devel уже установлен), сообщающую, что guile-devel уже установлен. Для того, чтобы обойти это, вам следует использовать опцию -e для удаления (или стирания) rpm-пакета перед его повторной установкой, как показано в Листинге 37. Это также следует сделать, если вам необходимо переустановить rpm-пакет, из-за того, что вы случайно удалили его некоторые файлы.

Листинг 37. Удаление guile-devel

```
[root@attic4 ~]# rpm -e guile-devel
```

Принудительная установка rpm-пакета

Иногда удаление rpm-пакета не удобно, особенно если имеются некоторые программы, зависящие от него. Например, если вы попытаетесь удалить пакет guile вместо guile-devel, то можете увидеть нечто подобное Листингу 38, где многие установленные пакеты зависят от пакета guile, а потому удаление не разрешается.

Листинг 38. Попытка удалить guile.

```
[root@attic4 ~]# rpm -q -R guile-devel
/bin/sh
/usr/bin/guile
guile = 5:1.6.7
rpmlib(CompressedFileNames) <= 3.0.4-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
[root@attic4 ~]# rpm -e guile
error: Failed dependencies:
        libguile-ltdl.so.1 is needed by (installed) g-wrap-1.3.4-
8.i386
        libguile-ltdl.so.1 is needed by (installed) gnucash-1.8.11-
3.i386
        libguile.so.12 is needed by (installed) g-wrap-1.3.4-8.i386
        libguile.so.12 is needed by (installed) gnucash-1.8.11-3.i386
        libqthreads.so.12 is needed by (installed) g-wrap-1.3.4-8.i386
        libqthreads.so.12 is needed by (installed) gnucash-1.8.11-
3.i386
        guile is needed by (installed) g-wrap-1.3.4-8.i386
        guile = 5:1.6.7 is needed by (installed) guile-devel-1.6.7-
2.i386
        /usr/bin/guile is needed by (installed) guile-devel-1.6.7-
2.i386
```

Не стоит говорить, что в этом случае не нужно удалять все пакеты, зависящие от этого, решением данной проблемы является принудительная установка rpm-пакета при помощи опции --force. В Листинг 39 мы проиллюстрировали принудительную переустановку guile-devel из файла загруженного в Листинге 36.

Листинг 39. Установка guile-devel с опцией --force.

```
[root@attic4 ~]# rpm -ivh --force guile-devel-1.6.7-2.i386.rpm
Preparing...
########################################### [100%]
   1:guile-devel
########################################### [100%]
```

Принудительное удаление rpm-пакета

Существует альтернатива принудительной установке при помощи опции --force, что может понадобиться в некоторых случаях. Вы можете удалить rpm-пакет, используя опцию --nodeps, отключающую внутреннюю проверку зависим□стей. Вообще вам следует делать это только если вы знаете что делаете и только если вы собираетесь исправить проблемы с зависимостями путем переустановки пакета. Примером может служить необходимость по некоторым причинам сменить версию пакета на более раннюю и желание убедиться, что все следы поздней версии были удалены. Команда, которую вам следует использовать для удаления пакета guile без проверки зависимостей выглядит так
rpm -e --nodeps guile

Опцию --nodeps можно также использовать и при установке rpm-пакета. И опять это не рекомендуется, но иногда необходимо:

Обновление RPM пакетов

Теперь, когда вы знаете как устанавливать и удалять RPM-пакеты, обратим свой взор на обновление до новых версий. Это похоже на установку, за тем исключением, что мы используем опцию -U или -F вместо опции -i. Различие между двумя этими опциями состоит в том, что опция -U обновляет существующие пакеты или устанавливает пакет, если он не был установлен, тогда как опция -F только обновляет или freshen (освежает) уже установленные пакеты. Вследствии этого опция -U используется чаще, особенно когда командная строка содержит список RPM-пакетов. Таким образом, удаленные пакеты устанавливаются, а установленные -- обновляются. Листинг 40 показывает результат попытки обновления guile-devel до текущей версии, а затем он удаляется и попытка обновления повторяется (теперь это срабатывает как установка).

Листинг 40. Обновление guile-devel.

```
[root@attic4 ~]# rpm -Uvh  guile-devel-1.6.7-2.i386.rpm
Preparing...
########################################### [100%]
        package guile-devel-1.6.7-2 is already installed
[root@attic4 ~]# rpm -e guile-devel
[root@attic4 ~]# rpm -Uvh  guile-devel-1.6.7-2.i386.rpm
Preparing...
########################################### [100%]
   1:guile-devel
########################################### [100%]
```

Опрос RPM-пакетов

Как вы могли понять из приведенных нами примеров установка rpm-пакетов требует указания полного имени файла (или URL), вроде guile-devel-1.6.7-2.i386.rpm. С другой стороны, удаление rpm-пакета требует только имени пакета, вроде guile-devel. Также как и с APT, RPM хранит внутреннюю базу установленных пакетов, позволяющую вам манипулировать установленными пакетами, используя имя пакета. В этой части учебника мы рассмотрим какая же информация из этого хранилища доступна для вас, при помощи опции -q (от query) для команды rpm.

Базовый запрос просто спрашивает пакет установлен или нет. Добавьте опцию -i и вы получите информацию о пакете. Отметим, что для установки, обновления и удаления пакетов вам потребуются права суперпользователя (root), но не-root пользователи могут выполнять запросы к базе данных rpm-пакетов.

Листинг 41. Отображение информации о guile-devel.

```
[ian@attic4 ~]$ rpm -q guile-devel
guile-devel-1.6.7-2
[ian@attic4 ~]$ rpm -qi guile-devel
Name        : guile-devel                  Relocations: (not
relocatable)
Version     : 1.6.7                             Vendor: Red Hat, Inc.
Release     : 2                             Build Date: Wed 02 Mar
2005 11:04:14 AM EST
Install Date: Thu 08 Sep 2005 08:35:45 AM EDT      Build Host:
porky.build.redhat.com
Group       : Development/Libraries         Source RPM: guile-1.6.7-
2.src.rpm
Size        : 1635366                          License: GPL
Signature   : DSA/SHA1, Fri 20 May 2005 01:25:07 PM EDT, Key ID
b44269d04f2a6fd2
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary     : Libraries and header files for the GUILE extensibility
library.
Description :
The guile-devel package includes the libraries, header files, etc.,
that you will need to develop applications that are linked with the
GUILE extensibility library.

You need to install the guile-devel package if you want to develop
applications that will be linked to GUILE. You also need to install
the guile package.
```

RPM пакеты и файлы в них

У вас будет часто появляться желание узнать что же находится внутри пакета или к какому пакету принадлежит некоторый файл. Для вывода списка файлов пакета guile-devel, используйте опцию -ql, как показано в Листинге 42. В этом пакете много файлов, но мы покажем только часть этого вывода.

Листинг 42. Отображение информации о guile-devel.

```
[ian@attic4 ~]$ rpm -ql guile-devel
/usr/bin/guile-config
/usr/bin/guile-snarf
/usr/include/guile
/usr/include/guile/gh.h
/usr/include/guile/srfi
/usr/include/guile/srfi/srfi-13.h
/usr/include/guile/srfi/srfi-14.h
/usr/include/guile/srfi/srfi-4.h
/usr/include/libguile
/usr/include/libguile.h
...
```

Вы можете ограничить список выводимых файлов, до списка настроечных файлов, добавлением опции -c к вашему запросу. Подобно этому, опция -d ограничивает вывод только файлами документации.

Опрос файла пакета

Приведенная выше команда это запрос к базе данных RPM установленных пакетов. Если вы только что загрузили пакет и хотите получить ту же информацию, то вы можете воспользоваться опцией -p (для файла пакета) в вашем запросе с указанием имени файла пакета (также как при установке пакета). Листинг 43 повторяет запросы из Листинга 41 к файлу пакета вместо базы данных RPM.

Листинг 43. Отображение информации для файла пакета guile-devel.

```
[ian@attic4 ~]$ rpm -qp guile-devel-1.6.7-2.i386.rpm
guile-devel-1.6.7-2
[ian@attic4 ~]$ rpm -qpi guile-devel-1.6.7-2.i386.rpm
Name        : guile-devel                  Relocations: (not
relocatable)
Version     : 1.6.7                             Vendor: Red Hat, Inc.
Release     : 2                          Build Date: Wed 02 Mar
2005 11:04:14 AM EST
Install Date: (not installed)              Build Host:
porky.build.redhat.com
Group       : Development/Libraries        Source RPM: guile-1.6.7-
2.src.rpm
Size        : 1635366                         License: GPL
Signature   : DSA/SHA1, Fri 20 May 2005 01:25:07 PM EDT, Key ID
b44269d04f2a6fd2
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Summary     : Libraries and header files for the GUILE extensibility
library.
Description :
The guile-devel package includes the libraries, header files, etc.,
that you will need to develop applications that are linked with the
GUILE extensibility library.

You need to install the guile-devel package if you want to develop
applications that will be linked to GUILE. You also need to install
the guile package.
```

Опрос всех установленных пакетов

Опция -a применяет ваш запрос ко всем установленным пакетам. Это приводит к очень большому выводу информации, поэтому обычно его используют вместе с одним или несколькими фильтрами, таким как sort для сортировки листинга, more или less для разбивки на страницы, wc для получения количества пакетов или файлов, или grep для поиска пакетов имя которых вы знаете не точно. Листинг 44 показывает следующие запросы:
Отсортированный список всех пакетов в системе.
Количество всех пакетов в системе.
Количество всех файлов во всех пакетах системы.
Количество всех файлов документации установленных при помощи RPM.
Поиск всех пакетов с именем, содержащим "guile" (с учетом регистра).

Листинг 44. Запросы ко всем пакетам.

```
[ian@attic4 ~]$ rpm -qa | sort | more
4Suite-1.0-8.b1
a2ps-4.13b-46
acl-2.2.23-8
acpid-1.0.4-1
alchemist-1.0.36-1
alsa-lib-1.0.9rf-2.FC4
alsa-utils-1.0.9rf-2.FC4
...
[ian@attic4 ~]$ rpm -qa | wc -l
874
[ian@attic4 ~]$ rpm -qal | wc -l
195681
[ian@attic4 ~]$ rpm -qald | wc -l
31881
[ian@attic4 ~]$ rpm -qa | grep -i "guile"
guile-devel-1.6.7-2
guile-1.6.7-2
```

Используя rpm -qa можно облегчить администрирование нескольких систем. Если вы перенаправите отсортированный листинг в файл на одной из машин, то выполнив то же самое на другой машине, вы, при помощи программы diff, сможете найти различия.

Поиск владельца файла.

Учитывая, что теперь вы можете вывести список всех пакетов и всех файлов пакета, можно сказать, что теперь у вас есть все, чтобы обнаружить к какому пакету принадлежит некий файл. Однако команда rpm предоставляет опцию -f, для помощи в обнаружении пакета, являющегося владельцем файла. В нашем примере с Dr Geo в разделе Компиляция и установка программ, нам потребовалась guile-config. Теперь у нас есть установленный пакет guile-devel, этоn исполняемый файл то что нам нужно. В Листинге 45 показано как использовать команду which для получения полного пути к команде guile-config, и полезный пример использования вывода в качестве входных параметров для команды rpm -qf. Отметим, что апострофы, окружающие `which guile-config`

наклонены в обратную сторону. Другим способом использования в bash является использование $(which guile-config)

Листинг 45. К какому пакету принадлежит guile-config.

```
[ian@attic4 ~]$ which guile-config
/usr/bin/guile-config
[ian@attic4 ~]$ rpm -qf `which guile-config`
guile-devel-1.6.7-2
[ian@attic4 ~]$ rpm -qf $(which guile-config)
guile-devel-1.6.7-2
```

RPM зависимости

Ранее мы видели, что не можем стереть пакет guile из-за зависимостей. В добавок к файлам RPM пакет может содержать различные функции, от которых могут зависеть другие пакеты. В нашем примере многим другим пакетам требуются функции, предоставляемые пакетом guile. И мы не сможем установить guile-devel, если мы еще не установили в системе guile. А как только guile-devel будет установлен, он станет еще одной причиной почему guile не может быть удален.

Обычно все это работает хорошо. Если вам необходимо установить несколько пакетов одновременно, некоторые из которых могут зависеть от других, просто предоставьте полный список вашей команде rpm -Uvh и она проанализирует зависимости и выполнит установку в верном порядке.

Вместо попыток стереть установленный пакет и получения сообщения об ошибке, команда rpm предоставляет опцию для опроса установленных пакетов или файлов пакетов для поиска от чего они зависят или что требуют. Это опция --requires, которая может быть сокращена до -R. В Листинге 46 перечислены функции, необходимые для guile-config. Добавьте опцию -p и используйте полные RPM-имена, если хотите опросить файл пакета вместо базы данных RPM.

Листинг 46. Требования guile-config.

```
[ian@attic4 ~]$ rpm -qR guile-devel
/bin/sh
/usr/bin/guile
guile = 5:1.6.7
rpmlib(CompressedFileNames) <= 3.0.4-1
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
```

В добавок к поиску функциональности, необходимой пакету, вам может потребоваться найти, какому файлу требуется данная функциональность (что делается при попытке удаления пакета). Листинг 47 иллюстрирует это для двух функций, требующихся guile-devel. Поскольку вывод может включать повторы, мы также показали как можно отфильтровать вывод при помощи sort и uniq для вывода каждого требуемого пакета только один раз.

Листинг 47. Что необходимо /usr/bin/guile и guile.

```
[ian@attic4 ~]$ rpm -q --whatrequires /usr/bin/guile guile
guile-devel-1.6.7-2
g-wrap-1.3.4-8
guile-devel-1.6.7-2
[ian@attic4 ~]$ rpm -q --whatrequires /usr/bin/guile guile | sort|uniq
guile-devel-1.6.7-2
g-wrap-1.3.4-8
```

Целостность RPM пакетов.

Чтобы убедиться в целостности RPM-пакетов, они содержат MD5 или SHA1 дайджест, а также могут иметь цифровую подпись. Пакеты с цифровой подписью обычно требуют публичный ключ для проверки. Чтобы проверить целостность файла RPM-пакета используйте опцию rpm --checksig (сокращенно -K). Обычно вы найдете полезным добавление опции -v для подробного вывода.

Листинг 48. Проверка целостности файла пакета guile-devel.

```
[ian@attic4 ~]$ rpm --checksig guile-devel-1.6.7-2.i386.rpm
guile-devel-1.6.7-2.i386.rpm: (sha1) dsa sha1 md5 gpg OK
[ian@attic4 ~]$ rpm -Kv guile-devel-1.6.7-2.i386.rpm
guile-devel-1.6.7-2.i386.rpm:
    Header V3 DSA signature: OK, key ID 4f2a6fd2
    Header SHA1 digest: OK (b2c61217cef4a72a8d2eddb8db3e140e4e7607a1)
    MD5 digest: OK (cf47354f2513ba0c2d513329c52bf72a)
    V3 DSA signature: OK, key ID 4f2a6fd2
```

Вы можете получить строку вывода, похожую на :

```
V3 DSA signature: NOKEY, key ID 16a61572
```

Это означает, что этот пакет подписан, но у вас в базе данных RPM нет необходимого публичного ключа. Отметим, что ранние версии RPM могли обеспечивать другую верификацию.

Если пакет подписан и вы хотите проверить соответствует ли он подписи, то вы должны найти файл соответствующей подписи и импортировать его в базу данных RPM. Сначала вам следует загрузить ключ, а затем проверить его отпечатки пальцев перед тем как импортировать его с использованием команды rpm --import. Для более подробной информации смотри Man страницы для rpm. Вы также можете найти большое количество информации об подписанных бинарных файлах на www.rpm.org.

Проверка установленного пакета

Подобно проверке целостности rpm-пакета вы можете захотеть проверить целостность установленных файлов, используя rpm -V. Эти шаги необходимы, чтобы убедиться в том, что файлы не изменились с тех пор, как были установлены из rpm-пакета. Как показано в Листинге 49 если пакет все еще не был изменен, то ничего не выводится.

Листинг 48. Проверка установленного пакета guile-devel.

```
[ian@attic4 ~]$ rpm -V guile-devel
```

Давайте войдем как root и удалим /usr/bin/guile-config вместе и заменим /usr/bin/guile-snarf копией /bin/bash и попытаемся сделать то же самое вновь. Результат приведен в Листинге 49.

Листинг 49. Подделка пакета guile-devel.

```
[root@attic4 ~]# rm /usr/bin/guile-config
rm: remove regular file `/usr/bin/guile-config'? y
[root@attic4 ~]# cp /bin/bash /usr/bin/guile-snarf
cp: overwrite `/usr/bin/guile-snarf'? y
[root@attic4 ~]# rpm -V guile-devel
missing     /usr/bin/guile-config
S.5....T    /usr/bin/guile-snarf
```

Этот листинг показывает нам, что /usr/bin/guile-snarf имеет не верную MD5 сумму, размер файла и не прошел проверку mtime. Вы можете исправить это стерев пакет и переустановив его, или принудительно переустановив, как было показано ранее. Если вы удаляте пакет, то ждите сообщения об ошибке, поскольку один из его файлов отсутствует.

Настройка RPM

RPM редко требует настройки. В старых версиях rpm вы могли производить изменения в /etc/rpmrc для управления операциями во время работы. В последних версиях этот файл был перенесен в /usr/lib/rpm/rpmrc, где он автоматически заменяется, при обновлении rpm-пакета, что приводит к потере любых сделанных вами изменений. Если требуется некая специфичная для системы конфигурация, то она все еще может быть добавлена в /etc/rpmrc, а индивидуальная конфигурация для пользователей должна находиться в .rpmrc.in в домашнем каталоге пользователя. Вы можете найти описание формата этих файлов в книге Maximum RPM (Смотри Ресурсы).

Если вы хотите просмотреть конфигурацию rpmrc, то для этого имеется специальная опция команды rpm. Используйте команду:

```
rpm rpm –showrc
```

Репозитории и другие средства

Теперь вы можете удивиться откуда же берутся все эти rpm пакеты, как их можно найти и как управлять обновлением своей системы. Если у вас дистрибутив, основанный на RPM (RPM-based), то скорее всего ваш дистрибутив имеет репозиторий (repository) пакетов. Ваш дистрибутив может также предоставлять инструменты для установки пакетов из репозитория или обновления вашей исходной системы. Эти инструменты могут быть как графическими, так и работать в командной строке. Примерами могут быть:

- YaST (SUSE)
- up2date (Red Hat)

- yum - Yellow Dog Updater Modified (Fedora и другие)
- Mandrake Software Management (Mandriva)

Обычно эти инструменты выполняют обновления многих пакетов в автоматическом или полу-автоматическом режиме. Они могут также обеспечивать возможности отображения содержимого репозиториев или поиска пакетов. Обратитесь к документации вашего дистрибутива за детальной информацией.

Если вы не можете найти какой-то особенный RPM при помощи встроенных средств вашей системы, другим прекрасным ресурсом для поиска RPM-пакетов является сервер Rpmfind.Net (смотри Ресурсы).