Впервые опубликовано на developerWorks 21.03.2006

Экзамен LPI 102: Ядро

Администрирование Linux для начинающих (LPIC-1) тема 105

Управление ядром в ходе работы системы

Формально Linux -- это ядро вашей системы. Ядро обеспечивает инфраструктуру для работы приложений и использования различных аппаратных средств. Это код низкого уровня, который взаимодействует с интерфейсами аппаратных средств, планирует и распределяет память и т.д. Выбор в пользу систем GNU/Linux часто обусловлен тем, что многие инструменты, создаваемые для большинства дистрибутивов, благодаря проекту GNU Фонда свободного программного обеспечения (Free Software Foundation) являются доступными. Однако, вы будете часто видеть вместо "GNU/Linux" только "Linux".

uname

Команда uname выдает информацию о вашей системе и ее ядре. В Листинге 1 показаны различные опции команды uname и получаемая с их помощью информация. Описание опций дается в Таблице 3.

Листинг 1. Команда uname

```
ian@pinguino:~$ uname
Linux
ian@pinguino:~$ uname -s
Linux
ian@pinguino:~$ uname -n
pinguino
ian@pinguino:~$ uname -r
2.6.12-10-386
ian@pinguino:~$ uname -v
#1 Mon Jan 16 17:18:08 UTC 2006
ian@pinguino:~$ uname -m
i686
ian@pinguino:~$ uname -o
GNU/Linux
ian@pinguino:~$ uname -a
Linux pinguino 2.6.12-10-386 #1 Mon Jan 16 17:18:08 UTC 2006 i686
GNU/Linux
```

Таблица 3. Опции команды uname

Опция	Описание
-s	Показать имя ядра. Эта информация выдается по умолчанию, если ни одна опция не указана.
-n	Показать имя узла сети или имя хоста.
-r	Показать номер выпуска ядра. Эта опция часто используется с командами управления модулями.
-v	Показать версию ядра.
-m	Показать имя аппаратной платформы (CPU).
-o	Показать имя операционной системы.
-a	Показать всю возможную информацию.

Листинг 1 получен на операционной системе Ubuntu, запущенной на процессоре Intel®. Команда uname доступна в большинстве систем UNIX® и UNIX-подобных систем, таких как Linux. Выдаваемая ею информация может быть различной в разных дистрибутивах и версиях Linux, а также на разных типах компьютеров. В Листинге 2 показаны выводы команды uname на системе Fedora Core 4, запущенной на машине AMD Athlon 64, и, для сравнения, на Apple PowerBook.

Листинг 2. Использование команды uname на других системах

```
Linux attic4 2.6.14-1.1656_FC4 #1 Thu Jan 5 22:13:55 EST 2006 x86_64
x86_64 x86_64 GNU/Linuxfilesystem

Darwin Ian-Shields-Computer.local 7.9.0 Darwin Kernel Version 7.9.0:
Wed Mar 30 20:11:17 PST 2005; root:xnu/xnu-517.12.7.obj~1/RELEASE_PPC
Power Macintosh powerpc
```

Модули ядра

Ядро управляет системой на низшем уровне, включая оборудование и интерфейсы. При большом разнообразии аппаратных средств и различных файловых систем ядро, способное поддерживать все их, будет слишком большим. К счастью, модули ядра позволяют при необходимости загрузить обеспечивающее поддержку программное обеспечение, такое как драйверы для аппаратных средств или файловые системы. Это позволяет запускать систему с небольшим ядром и затем подгружать модули по мере необходимости. Часто эта подгрузка происходит автоматически, например, при подключении устройств USB.

В оставшейся части этого раздела мы рассмотрим, как и с помощью каких команд производится конфигурирование модулей ядра.

Команды для выполнения задач загрузки и удаления модулей ядра требуют полномочий суперпользователя root. Команды, выдающие информацию о модулях, обычно могут быть выполнены обычным пользователем. Однако, в случае, если они расположены в каталоге /sbin, они будут недоступны для обычного пользователя, так как этот каталог не включается в путь поиска PATH. Таким образом, если вы не root, вам, вероятно, надо будет использовать полное наименование пути.

lsmod

Воспользуйтесь командой lsmod, чтобы узнать, какие модули загружены на вашей системе в настоящее время (см. Листинг 3). Вероятно, у вас вывод этой команды будет другим, хотя некоторые записи должны совпадать.

Листинг 3. Просмотр модулей ядра с помощью команды lsmod

```
[ian@attic4 ~]$ /sbin/lsmod
Module                  Size  Used by
nvnet                  74148  0
nvidia               4092336  12
forcedeth              24129  0
md5                     4161  1
ipv6                  268737  12
parport_pc             29189  1
lp                     13129  0
parport                40969  2 parport_pc,lp
autofs4                29637  1
sunrpc                168453  1
ipt_REJECT              5825  1
ipt_state               1985  3
ip_conntrack           42009  1 ipt_state
iptable_filter          3137  1
ip_tables              19521  3 ipt_REJECT,ipt_state,iptable_filter
dm_mod                 58613  0
video                  16069  0
button                  4161  0
battery                 9541  0
ac                      4933  0
ohci_hcd               26977  0
ehci_hcd               41165  0
i2c_nforce2             7105  0
i2c_core               21825  1 i2c_nforce2
shpchp                 94661  0
snd_intel8x0           34945  1
snd_ac97_codec         76217  1 snd_intel8x0
snd_seq_dummy           3781  0
snd_seq_oss            37569  0
snd_seq_midi_event      9409  1 snd_seq_oss
snd_seq                62801  5
snd_seq_dummy,snd_seq_oss,snd_seq_midi_event
snd_seq_device          9037  3 snd_seq_dummy,snd_seq_oss,snd_seq
snd_pcm_oss            51569  0
snd_mixer_oss          18113  1 snd_pcm_oss
snd_pcm               100553  3 snd_intel8x0,snd_ac97_codec,snd_pcm_oss
snd_timer              33733  2 snd_seq,snd_pcm
```

```
snd                    57669  11
snd_intel8x0,snd_ac97_codec,snd_seq_oss,snd_seq,
snd_seq_device,snd_pcm_oss,snd_mixer_oss,snd_pcm,snd_timer
soundcore              11169  1 snd
snd_page_alloc          9925  2 snd_intel8x0,snd_pcm
floppy                 65397  0
ext3                  132681  3
jbd                    86233  1 ext3
sata_nv                 9541  0
libata                 47301  1 sata_nv
sd_mod                 20545  0
scsi_mod              147977  2 libata,sd_mod
[ian@attic4 ~]$
```

Вы можете видеть, что в системе загружено множество модулей. Большинство из них поставляются вместе с ядром. Однако некоторые, такие как nvnet, nvidia и sata_nv от корпорации NVIDIA, содержат проприетарный код и не являются частью стандартного ядра. Таким образом, модульный подход позволяет подключать проприетарный код к ядру с открытым кодом. При условии, что лицензия производителя разрешает это, в дистрибутив Linux могут быть внесены проприетарные модули, что позволит не тратить силы на получение их непосредственно от производителя и даст гарантию, что вы сможете воспользоваться ими.

В Листинге 3 вы также можете видеть, что соответствующими модулями осуществляется поддержка таких устройств как видео, SATA, SCSI, дискеты и звуковые карты, а также сетевые устройства, например, IPV6, поддержка файловых систем, такой как ext3, и Remote Procedure Call (RPC) компании Sun.

Помимо имени модуля, команда lsmod показывает также размер и число пользователей модуля. Если модуль используется более чем одним пользователем, будет представлен список этих пользователей. Так, например модуль soundcore используется модулем snd, который в свою очередь используется несколькими другими звуковыми модулями.

modinfo

Команда modinfo выдает информацию об одном или нескольких модулях. Как показано в Листинге 4, эта информация содержит полный путь до файла, имя автора, лицензию, другие параметры, которые можно передать модулю, версию, зависимости и другую информацию.

Листинг 4. Основная информация о модуле

```
[ian@attic4 ~]$ /sbin/modinfo floppy
filename:       /lib/modules/2.6.12-
1.1456_FC4/kernel/drivers/block/floppy.ko
author:         Alain L. Knaff
license:        GPL
alias:          block-major-2-*
vermagic:       2.6.12-1.1456_FC4 686 REGPARM 4KSTACKS gcc-4.0
```

```
depends:
srcversion:     2633BC999A0747D8D215F1F
parm:           FLOPPY_DMA:int
parm:           FLOPPY_IRQ:int
parm:           floppy:charp
[ian@attic4 ~]$ /sbin/modinfo sata_nv
filename:       /lib/modules/2.6.12-
1.1456_FC4/kernel/drivers/scsi/sata_nv.ko
author:         NVIDIA
description:    low-level driver for NVIDIA nForce SATA controller
license:        GPL
version:        0.6
vermagic:       2.6.12-1.1456_FC4 686 REGPARM 4KSTACKS gcc-4.0
depends:        libata
alias:          pci:v000010DEd0000008Esv*sd*bc*sc*i*
alias:          pci:v000010DEd000000E3sv*sd*bc*sc*i*
alias:          pci:v000010DEd000000EEsv*sd*bc*sc*i*
alias:          pci:v000010DEd00000054sv*sd*bc*sc*i*
alias:          pci:v000010DEd00000055sv*sd*bc*sc*i*
alias:          pci:v000010DEd00000036sv*sd*bc*sc*i*
alias:          pci:v000010DEd0000003Esv*sd*bc*sc*i*
alias:          pci:v000010DEd*sv*sd*bc01sc01i*
srcversion:     3094AD48C1B869BCC301E9F
```

В Листинге 4 обратите внимание на строки, содержащие имена файлов модулей. Эти имена оканчиваются суффиксом .ko, что является характерным отличием модулей для ядра 2.6 от других объектных файлов и от модулей для ядра 2.4 и более ранних версий, в которых и для них и для других объектных файлов использовался один и тот же суффикс .o.

Вы можете также заметить, что путь включает в себя версию ядра. Например, частью пути /lib/modules/2.6.12-1.1456_FC4/kernel/drivers/block/floppy.ko является 2.6.12-1.1456_FC4. Это запись совпадает с выводом команды uname -r. Модули являются характерными для данного ядра, и эта структура каталогов отражает их взаимозависимость.

В системах с ядром версии 2.6 с помощью команды modinfo можно ограничить количество запросов определенной информации о модуле. Чтобы извлечь информацию одного типа, например, parm, description, license, filename или alias, воспользуйтесь опцией -F. Если вам необходимо получить информацию различных типов, используйте команду несколько раз с различными опциями. В ядрах версии 2.4 информацию о параметрах можно получить с помощью опции -p. Текущая версия команды modinfo поддерживает также параметры предыдущих версий. В Листинге 5 показаны несколько примеров.

Листинг 5. Определенная информация о модуле

```
[ian@attic4 ~]$ /sbin/modinfo -F parm snd
cards_limit:Count of auto-loadable soundcards.
major:Major # for sound driver.
[ian@attic4 ~]$ /sbin/modinfo -F license nvidia floppy
NVIDIA
GPL
```

```
[ian@attic4 ~]$ /sbin/modinfo -p snd
major:Major # for sound driver.
cards_limit:Count of auto-loadable soundcards.
```

Используйте ваши навыки работы в Linux

Вы можете использовать некоторые приемы, описанные в учебном пособии "Экзамен LPI 101 (тема 103): Команды GNU и UNIX" для получения информации, например, о количестве параметров, которые можно передать модулю. В Листинге 6 показан пример.

Листинг 6. Количество параметров модуля

```
[ian@attic4 ~]$ for n in `/sbin/lsmod | tail +2 | cut -d " " -f1`;
> do echo "$n $(/sbin/modinfo -p $n |wc -l )" | grep -v " 0$"; done
nvnet 12
forcedeth 1
parport_pc 5
dm_mod 1
ohci_hcd 2
ehci_hcd 2
shpchp 3
snd_intel8x0 7
snd_ac97_codec 1
snd_seq_dummy 2
snd_seq_oss 2
snd_seq 7
snd_pcm_oss 3
snd_pcm 2
snd_timer 1
snd 2
snd_page_alloc 1
scsi_mod 6
```

rmmod

Если "use count" модуля равен 0, вы можете без опасений удалить его. Например, вы можете сделать это при подготовке к загрузке обновленной версии. Это позволит не перезагружать компьютер только из-за того, что необходимо обновить поддержку какого-то отдельного устройства. Для удаления введите команду rmmod имя модуля, как показано в Листинге 7.

Листинг 7. Удаление модуля из работающей системы

```
[root@attic4 ~]# rmmod floppy
```

Чтобы узнать о других опциях команды rmmod, обратитесь к ее странице man.

insmod и modprobe

Удаленный вами модуль может понадобиться вновь. Чтобы загрузить модуль,

введите команду insmod, полный путь для модуля, который необходимо загрузить, и все необходимые опции. При использовании этой команды вы, вероятно, захотите использовать режим подстановки вывода команды (command substitution), чтобы сгенерировать имя файла. В Листинге 8 показаны два способа загрузки модуля.

Листинг 8. Загрузка модуля с использованием команды insmod

```
[root@attic4 ~]# insmod /lib/modules/`uname -r`
/kernel/drivers/block/floppy.ko
[root@attic4 ~]# rmmod floppy
[root@attic4 ~]# insmod $(modinfo -F filename floppy)
```

При использовании второго варианта нет необходимости помнить, в каком подкаталоге (в данном случае drivers/block) расположен модуль. Но существует и более удобный способ загрузки модуля. Команда modprobe предоставляет высокоуровневый интерфейс, оперирующий именами модулей, а не именами путей файлов. Она также управляет загрузкой дополнительных модулей, от которых зависит загружаемый модуль, и позволяет не только загружать, но и удалять модули.

В Листинге 9 показано, как использовать команду modprobe для удаления модуля vfat вместе с используемым им модулем fat. Затем показано, как будет вести себя система, если модуль будет загружен вновь, и в завершение показан результат загрузки модуля. Обратите внимание на опцию -v, которая позволяет получать подробный вывод. Если не использовать эту опцию, команда modprobe (а также команда insmod) выведет только сообщения об ошибках модуля. Между каждым действием используется команда lsmod, пропущенная через grep, чтобы проверить, загружены или нет модули vfat и fat.

Листинг 9. Загрузка модуля с помощью команды modprobe

```
[root@lyrebird root]# modprobe -r vfat
vfat: Device or resource busy
[root@lyrebird root]# lsmod | grep fat
vfat                   13132   1
fat                    38744   1  [vfat]
[root@lyrebird root]# umount /windows/D
[root@lyrebird root]# modprobe -r vfat
[root@lyrebird root]# modprobe -v --show  vfat
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/fat/fat.o
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/vfat/vfat.o
[root@lyrebird root]# lsmod | grep fat
[root@lyrebird root]# modprobe -v  vfat
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/fat/fat.o
Using /lib/modules/2.4.21-37.0.1.EL/kernel/fs/fat/fat.o
Symbol version prefix ''
/sbin/insmod /lib/modules/2.4.21-37.0.1.EL/kernel/fs/vfat/vfat.o
Using /lib/modules/2.4.21-37.0.1.EL/kernel/fs/vfat/vfat.o
[root@lyrebird root]# lsmod | grep fat
vfat                   13132   0  (unused)
fat                    38744   0  [vfat]
```

depmod

Как вы только что видели, в случае, если один модуль зависит от другого, при помощи команды modprobe можно управлять автоматической загрузкой нескольких модулей. Зависимости прописаны в файле modules.dep, хранящемся в подкаталоге /lib/modules соответствующего ядра (имя ядра определяется при помощи команды uname -r). Этот файл вместе с несколькими map-файлами генерируется при помощи команды depmod. Опция -a (от all) теперь необязательна.

Команда depmod просматривает модули в подкаталоге /lib/modules текущего ядра и обновляет информацию о зависимостях. В Листинге 10 показан пример выполнения команды depmod и полученных в результате файлов.

Листинг 10. Использование команды depmod для создания файла modules.dep

```
[root@lyrebird root]# date
Thu Mar 16 10:41:05 EST 2006
[root@lyrebird root]# depmod
[root@lyrebird root]# cd /lib/modules/`uname -r`
[root@lyrebird 2.4.21-37.0.1.EL]# ls -l mod*
-rw-rw-r--    1 root     root        54194 Mar 16 10:41 modules.dep
-rw-rw-r--    1 root     root           31 Mar 16 10:41
modules.generic_string
-rw-rw-r--    1 root     root           73 Mar 16 10:41
modules.ieee1394map
-rw-rw-r--    1 root     root         1614 Mar 16 10:41 modules.isapnpmap
-rw-rw-r--    1 root     root           29 Mar 16 10:41
modules.parportmap
-rw-rw-r--    1 root     root        65171 Mar 16 10:41 modules.pcimap
-rw-rw-r--    1 root     root           24 Mar 16 10:41
modules.pnpbiosmap
-rw-rw-r--    1 root     root       122953 Mar 16 10:41 modules.usbmap
[root@lyrebird 2.4.21-37.0.1.EL]# cd -
/root
```

Вы можете изменить поведение команд modprobe и depmod, скорректировав файл /etc/modules.conf. Обычно это делается для создания псевдонимов (alias) для имен модулей и определения команд, которые должны быть запущены после загрузки модуля или перед его отключением. Однако, могут быть произведены и другие настройки. В Листинге 11 показан пример файла /etc/modules.conf. Более подробную информацию можно найти в странице man для modules.conf.

Листинг 11. Пример файла /etc/modules

```
[root@lyrebird root]# cat /etc/modules.conf
alias eth0 e100
alias usb-controller usb-uhci
alias usb-controller1 ehci-hcd
alias sound-slot-0 i810_audio
post-install sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -L
>/dev/null 2>&1 || :
pre-remove sound-slot-0 /bin/aumix-minimal -f /etc/.aumixrc -S >/dev/null
2>&1 || :
```

Вы должны также знать, что некоторые системы используют другой конфигурационный файл, modprobe.conf, и есть системы, которые хранят информацию о конфигурации модулей в каталоге /etc/modules.d. В некоторых системах вы можете встретить файлы с названием /etc/modules. Эти файлы содержат имена модулей ядра, которые должны быть загружены в ходе загрузки системы.

Модули USB

Когда вы подключаете к работающей системе Linux устройство USB, ядро должно определить, какие модули необходимо загрузить для управления устройством. Обычно это делается при помощи hot-plug скрипта, использующего команду usbmodules для поиска соответствующего модуля. Вы можете также запустить команду usbmodules (от имени root). В Листинге 12 показан пример.

Листинг 12. Модули USB

```
root@pinguino:~# lsusb
Bus 005 Device 004: ID 1058:0401 Western Digital Technologies, Inc.
Bus 005 Device 003: ID 054c:0220 Sony Corp.
Bus 005 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 003: ID 04b3:310b IBM Corp. Red Wheel Mouse
Bus 001 Device 001: ID 0000:0000
root@pinguino:~# usbmodules --device /proc/bus/usb/005/003
usb-storage
root@pinguino:~# usbmodules --device /proc/bus/usb/001/003
usbmouse
usbhid
```

В следующем разделе показано, как собрать и настроить собственное ядро.

Настройка и сборка ядра и его модулей

Как вы узнали из предыдущего раздела, Управление ядром в ходе работы системы, ядро обеспечивает поддержку низкого уровня для аппаратных средств и файловых систем. Современные образы ядра обычно обладают только необходимой минимальной функциональностью, но при необходимости могут быть настроены для поддержки дополнительных функций с помощью модулей ядра (kernel modules). Дополнительная поддержка включается только при необходимости, например, при подсоединении устройства или в других случаях.

Код модулей становится неотъемлемой частью ядра, динамично расширяя его функциональность. Если возможности загруженных модулей ядра в данное время не используется, ядро может самостоятельно отключить их от основной части и

выгрузить из памяти при помощи процесса, известного как autocleaning.

Ядро, загруженное с диска без модулей, как отдельный бинарный файл, должно было бы обеспечивать всю необходимую функциональность. И при необходимости добавить системе функциональности вам пришлось бы каждый раз полностью пересобирать ядро.

Однако, вы не можете поместить в модули все. Для монтирования файловой системы необходим, как минимум, образ ядра. Но, как вы могли узнать из учебного пособия "Экзамен LPI 101 (тема 102): Инсталляция Linux и управление пакетами", загрузчик может загрузить стартовый RAM-диск (initial RAM disk или initrd), который может содержать необходимые для монтирования файловой системы модули. Тем не менее, образ ядра должен как минимум включать поддержку файловой системы RAM, используемой в initial RAM disk, иначе ваша система не сможет быть загружена.

В ходе загрузки системы происходит монтирование файловой системы и затем запускаются другие процессы инициализации. Через некоторое время система загрузится полностью и будет готова к использованию. Однако, ядро будет находиться в состоянии готовности выполнять пользовательские процессы и распределять системные ресурсы между использующими их задачами.

Модульные ядра хорошо работают в современных системах с большим количеством RAM и дискового пространства. Однако, вы можете подключить новые аппаратные средства, например, видокарту или запоминающее устройство, которые не поддерживаются ядром, входящим в дистрибутив. Некоторые устройства содержат проприетарный код, который, как было сказано выше загрязняют чистое ядро Linux, поэтому некоторые дистрибутивы не включают их, даже если лицензия производителя позволяет это. В таких случаях необходимо по крайней мере собрать новые модули, а возможно, и пересобрать ядро.

Linux может использоваться как во встроенных системах, таких как мобильные телефоны, сетевых устройствах, таких как маршрутизаторы, и компьютерных приставках, так и в компьютерах большинства традиционных конфигураций. Некоторые из этих устройств используют ядра, настроенные на поддержку только тех функций, которые обеспечивают поддержку системы. Например, бездисковой системе, предназначенной для использования в качестве firewall, вероятно, не потребуется поддержка других типов файловых систем, кроме файловой системы read-only, с помощью которой она загружается, но может потребоваться поддержка продвинутых сетевых устройств, не входящая в стандартное ядро. К тому же потребуется специально настроенное ядро.

Пакеты с исходными кодами

Исходные коды ядра Linux можно найти в Linux Kernel Archives (см. Ресурсы). Для начала воспользуйтесь пакетами для ядра, входящего в ваш дистрибутив Linux, так как производитель мог добавить в него индивидуальные патчи. Если вы уже умеете находить и извлекать исходные коды ядра, просмотрите учебное пособие "Экзамен LPI 101 (тема 102): Инсталляция Linux управление пакетами". Прежде чем вносить изменения, сделайте резервную копию системы, чтобы можно было восстановить ее, если что-то пойдет не так

Если вы загружаете исходники из публичных архивов ядра, вы получаете сжатые файлы, которые необходимо декомпрессировать с помощью команды gzip или bzip2, в случае, если вы получили исходники в формате .gz, или команды bzip2 для исходников в формате .bz2. Каталог pub/linux/kernel/ на сервере содержит каталоги с исходниками для ядра версий 2.4, 2.5 и 2.6. На время написания этого пособия последняя версия ядра 2.6 содержалась в linux-2.6.15.tar.bz2.

В каталоге с исходниками ядра также содержится соответствующий файл ChangeLog-2.6.15.6, в котором описаны все изменения для текущей версии, и patch-2.6.15.bz2, позволяющий внести изменения в исходники предыдущей версии, приведя их к версии 2.6.15. Также вы заметите файлы, содержащие подписи, которые могут быть использованы для того, чтобы убедиться, что загруженные файлы случайно или умышленно не были повреждены.

Распаковка сжатых исходников обычно производится в каталог /usr/src, при этом создается подкаталог для версии ядра, например linux-2.6.15, содержащий дерево файлов, необходимых для сборки ядра. Если у вас уже есть такой каталог, вы можете создать его резервную копию или переименовать его, прежде чем приступать к распаковыванию новых исходников ядра. Это позволит вам в случае необходимости вернуться назад, а также даст гарантию того, что в дереве исходников ядра не окажется случайных файлов. Вам понадобится приблизительно 40 Мбайт для архива tar и приблизительно 350 Мбайт для разворачивания исходных кодов.

В настоящее время некоторые производители, в особенности Red Hat, распространяют файлы заголовков (headers) ядра и исходники, необходимые для сборки модулей, в виде пакета kernel development. Документация может не входить в этот пакет. Это делается и этого достаточно для того, чтобы собирать отдельные модули, например модули для проприетарных графических карт, но недостаточно для пересборки ядра. Производитель вашего дистрибутива должен иметь информацию, о том, как пересобрать ядро, и о том, где получить исходники. Обратитесь к соответствующей документации, например, посмотрите замечания к выпуску.

Предположим, вы получили через FTP или HTTP с сервера download.fedora.redhat.com из каталога pub/fedora/linux/core/updates/4/SRPMS/ исходный пакет kernel-2.6.15-1.1833_FC4.src.rpm и поместили его в каталог /root. Номер версии ядра, используемый здесь для примера, может отличаться от номера версии в вашей системе. Если используется дистрибутив Fedora Core, необходимо устновить исходный пакет RPM, затем перейти в каталог /usr/src/redhat/SPECS и в заключение пересобрать исходный RPM, для того чтобы создать дерево исходников ядра Linux, как показано в Листинге 13.

Листинг 13. Создание дерева исходников ядра для Fedora Core

```
[root@attic4 ~]# uname -r
2.6.15-1.1833_FC4
[root@attic4 ~]# rpm -Uvh kernel-2.6.15-1.1833_FC4.src.rpm
   1:kernel                 ########################################
[100%]
[root@attic4 ~]# cd /usr/src/redhat/SPECS
[root@attic4 SPECS]# rpmbuild -bp --target $(arch) kernel-2.6.spec
Building target platforms: x86_64
Building for target x86_64
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.23188
+ umask 022
+ cd /usr/src/redhat/BUILD
+ LANG=C
+ export LANG
+ unset DISPLAY
+ '[' '!' -d kernel-2.6.15/vanilla ']'
+ cd /usr/src/redhat/BUILD
+ rm -rf kernel-2.6.15
+ /bin/mkdir -p kernel-2.6.15
+ cd kernel-2.6.15
+ /usr/bin/bzip2 -dc /usr/src/redhat/SOURCES/linux-2.6.15.tar.bz2
+ tar -xf -
...
+ echo '# x86_64'
+ cat .config
+ perl -p -i -e 's/^SUBLEVEL.*/SUBLEVEL = 15/' Makefile
+ perl -p -i -e 's/^EXTRAVERSION.*/EXTRAVERSION = -prep/' Makefile
+ find . -name '*.orig' -o -name '*~' -exec rm -f '{}' ';'
+ exit 0
```

Исходники ядра Linux для Fedora теперь расположены в /usr/src/redhat/BUILD/kernel-2.6.15/linux-2.6.15. В соответствии с принятыми соглашениями дерево /linux-2.6.15 часто перемещают в каталог /usr/src и делают символьную ссылку в /usr/src/linux, как показано в Листинге 14. Это не обязательно, но когда исходники ядра находятся в /usr/src/linux, проще ориентироваться по ссылкам.

Листинг 14. Перемещение дерева исходников в /usr/src

```
[root@attic4 SPECS]# mv ../BUILD/kernel-2.6.15/linux-2.6.15 /usr/src
[root@attic4 SPECS]# cd /usr/src
[root@attic4 src]# ln -s linux-2.6.15 linux
[root@attic4 src]# ls -ld lin*
lrwxrwxrwx   1 root root   12 Mar 20 18:23 linux -> linux-2.6.15
drwxr-xr-x  20 root root 4096 Mar 20 18:13 linux-2.6.15
```

Прежде чем попытаться что-нибудь пересобрать, просмотрите файл Changes, расположенный в каталоге Documentation. Он среди прочего содержит список

пакетов, необходимых для сборки ядра, с указанием номеров версий. Убедитесь, что эти пакеты установлены.

В Листинге 13 вы могли заметить файлы Makefile и .config. make-файл содержит различные цели сборки для конфигурирования опций ядра, сборки ядра и его модулей, установки модулей и сборки пакетов RPM или deb. Наиболее свежие версии исходников ядра позволяют использовать make help для получения краткой справки для каждой цели. В более старых системах было необходимо обращаться к документации или просматривать make-файл. В Листинге 15 показана часть вывода make help.

Листинг 15. Справка по make-файлу для сборки ядра

```
[ian@attic4 linux-2.6.15]$ make help
Cleaning targets:
  clean           - remove most generated files but keep the config
  mrproper        - remove all generated files + config + various backup
files

Configuration targets:
  config          - Update current config utilising a line-oriented
program
  menuconfig      - Update current config utilising a menu based program
  xconfig         - Update current config utilising a QT based front-end
  gconfig         - Update current config utilising a GTK based front-end
  oldconfig       - Update current config utilising a provided .config as
base
  randconfig      - New config with random answer to all options
  defconfig       - New config with default answer to all options
  allmodconfig    - New config selecting modules when possible
  allyesconfig    - New config where all options are accepted with yes
  allnoconfig     - New minimal config

Other generic targets:
  all             - Build all targets marked with [*]
* vmlinux         - Build the bare kernel
* modules         - Build all modules
  modules_install - Install all modules
  dir/            - Build all files in dir and below
  dir/file.[ois]  - Build specified target only
...
```

Конфигурация

Файл .config содержит информацию о конфигурации ядра, включая конфигурацию целевой платформы, какие компоненты будут включены, и должны ли компоненты быть включены в ядро или собраны в виде модулей. Создание файла .config является первым шагом на пути сборки или пересборки ядра. Этот файл создается при помощи одной из конфигурационных целей make-файла.

Основные опции конфигурации:

config

Цель config использует интерфейс командной строки для получения ответов многие на вопросы, касающиеся создания или обновления файла .config.

После появления конфигурационных целей, использующих меню, интерфейс командной строки используется редко.

menuconfig

Цель menuconfig использует программу с меню-интерфейсом, построенную на базе ncurses, для создания или обновления файла .config. Вы должны только ответить на вопросы для элементов, которые хотите изменить. Этот подход заменил старую цель config. Выполняется в окне терминала удаленно или локально.

xconfig

Цель xconfig использует систему графического меню, основанную на QT front-end, используемом в KDE desktop.

gconfig

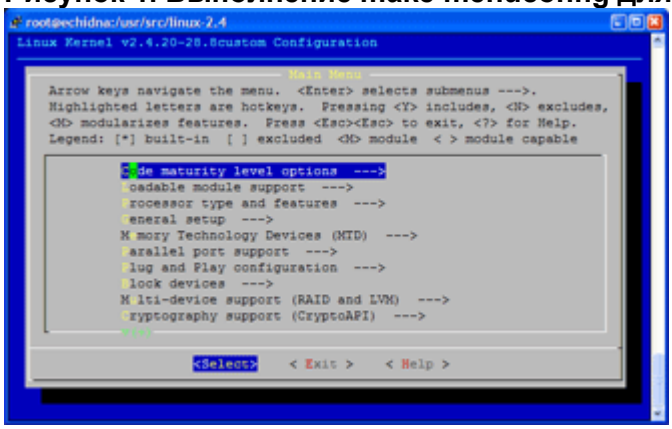
Цель gconfig использует систему графического меню, основанную на QT front-end, используемом в GNOME desktop.

oldconfig

Цель oldconfig позволяет создать конфигурацию с использованием существующего файла .config, созданного ранее или взятого из другой системы. Например, если вы устанавливали исходники ядра для Fedora, как описано выше, вы можете скопировать конфигурационный файл для вашей системы из /lib/modules/$(uname -r)/build/.config в /usr/src/linux. Сделав это, можно использовать одну из целей меню конфигурации, чтобы при необходимости внести изменения.

На Рисунке 1 показан пример того, что вы можете увидеть, запустив make menuconfig для ядра 2.4. Используйте Enter, чтобы перейти к меню ниже уровнем и Esc для возврата. Для каждого элемента можно получить справку. Для этого выберите < Help > и нажмите Enter или просто введите h. Нажмите Esc для возврата.

Рисунок 1. Выполнение make menuconfig для ядра 2.4



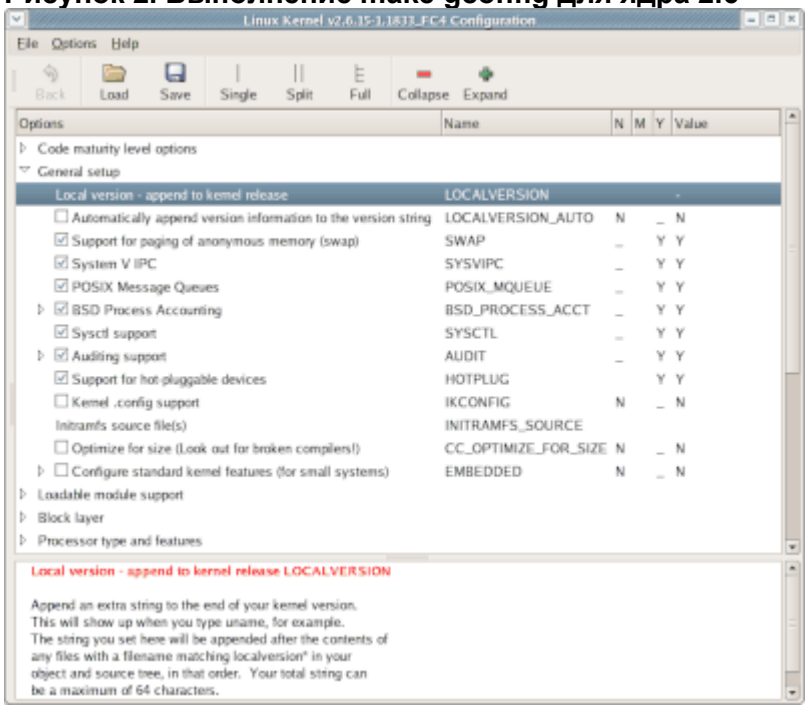
В Таблице 4 показаны различные опции, позволяющие включать компоненты в ядро или создавать специальные модули. Когда опция подсвечена, при помощи клавиши пробела можно перемещаться между возможными вариантами для данного компонента. Чтобы активировать опцию, нажмите y, чтобы отключить -- n, чтобы создать, если это возможно, модуль, нажмите m.

Таблица 4. Опции menuconfig

Опция	Описание
[*]	Компонент будет включен в ядро.
[ ]	Компонент не будет включен в ядро.
<M>	Компонент будет оформлен в виде модуля.
< >	Компонент не будет включен в ядро, но может быть оформлен в виде модуля.

На Рисунке 2 показан пример того, что вы можете увидеть, запустив make gconfig для ядра 2.6. Щелчком по стрелкам можно развернуть или свернуть элемент меню. На нижней панели отображается справочная информация.

Рисунок 2. Выполнение make gconfig для ядра 2.6



Ниже дается описание главных конфигурационных разделов для ядра 2.6. В ядре 2.4 и более ранних версиях могут присутствовать не все эти разделы. Этот список дает краткий обзор того, где и что расположено.

Code maturity level options

Этот раздел содержит опцию, определяющую, будет ли вам предоставляться возможность выбирать код, имеющий статус экспериментального. Если вы не выберете эту опцию, выбирать можно будет только опции, имеющие стабильный статус. Имейте в виду, что функции, которые вы выбрали, могут

работать, а могут и нет в текущей версии ядра, так что у вас есть шанс помочь в его отладке.

General setup

Этот раздел позволяет добавить идентификационную строку к вашему ядру, а также ряд атрибутов, которые не имеют отношения к каким-либо разделам, но тем не менее должны быть описаны.

Loadable module support

Этот раздел содержит опции, определяющие, будет ли ваше ядро поддерживать модули и будут ли они подгружаться и выгружаться автоматически. Опцию "Enable loadable module support" следует включть.

Block layer

Этот раздел управляет поддержкой дисков размером более 2TB и позволяет выбирать режимы обслуживания дисковых устройств.

Processor type and features

Этот раздел содержит специфичные для данного типа процессора конфигурационные опции. Здесь вы можете выбрать процессор и семейство процессора, которые будут поддерживаться вашим ядром. Вы можете включать или отключать поддержку ядром различных возможностей, предоставляемых данным процессором. Убедитесь, что вы включили поддержку многопроцессорных систем (symmetric multi-processing support), если в вашем системе установлено более одного процессора или процессор поддерживает технологию hyperthreading. Кроме того, для получения большей производительности графической подсистемы в системах с AGP или PCI видеокартами следует включить поддержку MTRR.

Power management options

В этом разделе помещены опции, касающиеся управления питанием. Особенно они важны для ноутбуков. Кроме контроля состояния питания, вы сможете найти там средства для контроля и мониторинга таких параметров как температура или состояние охлаждающего вентилятора.

Bus options (PCI etc.)

Этот раздел содержит опции для компьютерных шин, поддерживаемых вашей системой, таких как PCI, PCI Express и PC Card. Здесь вы можете включить поддержку файловой системы /proc/pci, которой можно пользоваться вместе с обычно используемой командой lspci.

Executable file formats / Emulations

Этот раздел содержит опции, касающиеся поддержки различных форматов бинарных файлов. Следует включить поддержку "ELF binary". Кроме того, можно включить поддержку DOS binaries для запуска их под DOSEMU, также как и других поддерживаемых соответствующими wrapper'ами бинарных файлов, таких как Java™, Python, Emacs-Lisp и т.д. Наконец, для 64-битных систем, поддерживающих 32-битную эмуляцию, вы, возможно, захотите включить поддержку 32-битных приложений.

Networking

Секция, касающаяся настроек сети, довольно велика. Здесь вы можете включить базовую поддержку сокетов, сетей TCP/IP, фильтрацию, маршрутизацию и bridging сетевых пакетов, а также поддержку различных протоколов, таких как IPV6, IPX, Appletalk и X.25. Кроме того, вы можете включить поддержку wireless, infrared и amateur radio.

Device drivers

Этот раздел также очень велик. Здесь вы можете включить поддержку большого числа аппаратных устройств, включая IDE/ATAPI или SCSI диски, или flash-диски. Включите DMA для ваших IDE устройств; иначе они будут

работать в более медленной PIO-моде. Если вы хотите иметь поддержку multiple devices, таких как RAID или LVM, соответсвующие опции также надо включить. Здесь вы также можете включить поддержку параллельного порта для работы с принтером через этот интерфейс. Здесь происходит конфигурирование широкого набора поддерживаемых сетевых устройств для различных сетевых протоколов, которые мы конфигурировали ранее. Кроме того, здесь вы найдете опции поддержки устройств аудио- и видео-захвата, устройств USB и IEEE 1384 (Firewire), а также различного рода устройств аппартного мониторинга. В разделе управления символьными устройствами (Character Devices) вы, возможно, захотите включить поддержку печати через параллельный порт и поддержку direct rendering.

Firmware drivers

Этот раздел содержит несколько опций, относящихся к установке и обновлению BIOS, таких как использование функций Dell System Management на некоторых системах производства компании Dell.

File systems

Этот раздел предназначен для конфигурирования файловых систем, поддержку которых вы хотите иметь в вашем ядре, скомпилированных в виде модулей или нет. Также вы сможете найти здесь файловые системы для съемных дисковых устройств (дискеты, CD и DVD устройства), а также сетевых файловых систем, таких как NFS, SMB или CIFS. Поддержка различных типов разделов и национальных кодировок Native Language Support также располагаются в этом разделе.

Instrumentation support

Этот раздел позволяет вам включать экспериментальную поддержку профайлинга для профилирования работы вашей системы.

Kernel hacking

Этот раздел позволяет включать режим отладки ядра и выбирать, какие дополнительные функции будут включены.

Security options

Этот раздел предназанчен для конфигурирования опций защиты, а также включения и конфигурирования SELinux (Security Enhanced Linux).

Cryptographic options

В это разделе можно сконфигурировать поддержку различных алгоритмов шифорования, таких как MD4, DES и SHA256.

Library routines

Здесь вы можете указать ряд алгоритмов вычисления контрольных сумм (CRC), которые будут включены в ядро или собраны как модули.

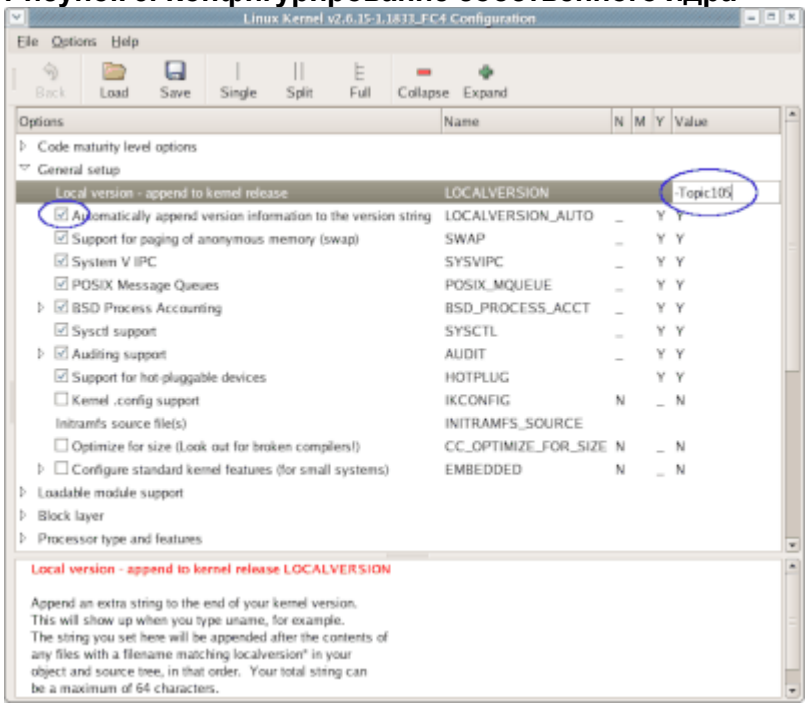
Сборка

Теперь, когда вы ознакомились с главными аспектами конфигурации ядра, вы готовы к его сборке. Если вы не знаете, каково состояние дерева для сборки, прежде чем приступать к конфигурированию нового ядра выполните make clean. Для более полной очистки выполните make mrproper. При этом будет удален файл .config, а также некоторые другие файлы, используемые в процессе сборки. Если вы сделаете это и затем захотите восстановить файл .config из резервной копии, вам надо будет выполнить make oldconfig, прежде чем приступить к конфигурированию.

В ходе эксперимента вы должны дать новому ядру специальное название, которое позволит вам легко его идентифицировать. Чтобы сделать это, установите значение

Local version и активируйте опцию Automatically append version information to the version string в соответствующей строке раздела General setup, как показано на Рисунке 3.

Рисунок 3. Конфигурирование собственного ядра



В примерах, приведенных в оставшейся части этого учебного пособия, при сборке ядра вносятся только те два изменения, которые показаны на Рисунке 3.

В принципе, для сборки ядра не требуются полномочия суперпользователя root, несмотря на то, что для установки нового ядра эти полномочия необходимы. Однако, если вы используете пакеты, установленные из дистрибутива, вам, вероятно, понадобятся привилегии суперпользователя root для доступа к необходимым файлам и каталогам. Выкачав исходники ядра из Linux kernel archives и распаковав их в своем домашнем каталоге или скопировав дерево сборки ядра и скорректировав права доступа, вы сможете поупражняться в сборке ядра, используя свою учетную запись пользователя.

Чтобы начать сборку ядра 2.6, выполните make.

Чтобы начать сборку ядра 2.4, выполните эти три команды:

```
make dep
make bzImage
make modules
```

Первая создает файлы необходимых зависимостей. Вторая собирает ядро. И последняя собирает модули.

Выполнение make на моей системе AMD Athlon 3500+ с целью полностью собрать ядро с нуля заняло около получаса. На более медленных системах выполнение этой задачи может занять до двух часов, так что сделайте перерыв или займитесь чем-нибудь другим. После запуска процесса сборки будут появляться сообщения подобные приведенным в Листинге 16.

Листинг 16. Выполнение make

```
[root@attic4 linux]# make
  CHK     include/linux/version.h
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/basic/split-include
  HOSTCC  scripts/basic/docproc
  SPLIT   include/linux/autoconf.h -> include/config/*
  CC      arch/x86_64/kernel/asm-offsets.s
  GEN     include/asm-x86_64/asm-offsets.h
...
  LD [M]  sound/usb/snd-usb-lib.ko
  CC      sound/usb/usx2y/snd-usb-usx2y.mod.o
  LD [M]  sound/usb/usx2y/snd-usb-usx2y.ko
```

Инсталляция

После того как у вас будет полностью собранное ядро, вы должны выполнить еще два действия. Сначала вам необходимо выполнить make modules_install для установки модулей ядра в новый подкаталог /lib/modules.

Если вам необходимы проприетарные модули для видеокарты или сетевой карты, как это понадобилось мне для графической карты nVidia и для чипсета материнской платы nForce 4, сейчас подходящее время для того, чтобы собрать эти модули, используя предоставленные производителем средства.

И в заключение, вам необходимо выполнить make install для установки нового ядра и стартового RAM-диска (initial RAM disk) в каталог /boot и обновления конфигурации загрузчика. Листинг 17 иллюстрирует эти действия.

Листинг 17. Инсталляция ядра и модулей

```
[root@attic4 linux]# make modules_install
  INSTALL arch/x86_64/crypto/aes-x86_64.ko
  INSTALL arch/x86_64/kernel/cpufreq/acpi-cpufreq.ko
  INSTALL arch/x86_64/kernel/microcode.ko
  INSTALL arch/x86_64/oprofile/oprofile.ko
  INSTALL crypto/aes.ko
  INSTALL crypto/anubis.ko
  INSTALL crypto/arc4.ko
...
[root@attic4 linux]# ls -lrt /lib/modules | tail -n 3
drwxr-xr-x  5 root root 4096 Mar  4 14:48 2.6.15-1.1831_FC4
drwxr-xr-x  5 root root 4096 Mar 20 18:52 2.6.15-1.1833_FC4
drwxr-xr-x  3 root root 4096 Mar 20 21:38 2.6.15-prep-Topic105
[root@attic4 linux]# sh  /root/NFORCE-Linux-x86_64-1.0-0310-pkg1.run -a \
>  -n -K -k 2.6.15-prep-Topic105
Verifying archive integrity...OK
Uncompressing NVIDIA nForce drivers for Linux-x86_64 1.0-
0310...................
[root@attic4 linux]# sh  /root/NVIDIA-Linux-x86_64-1.0-8178-pkg2.run -a \
```

```
> -n -K -k 2.6.15-prep-Topic105
Verifying archive integrity... OK
Uncompressing NVIDIA Accelerated Graphics Driver for Linux-x86_64 1.0-
8178.................
[root@attic4 linux]# make install
  CHK     include/linux/version.h
  CHK     include/linux/compile.h
  CHK     usr/initramfs_list
Kernel: arch/x86_64/boot/bzImage is ready  (#2)
sh /usr/src/linux-2.6.15/arch/x86_64/boot/install.sh 2.6.15-prep-Topic105
arch/x86_64/boot/bzImage System.map "/boot"
[root@attic4 linux]# ls -lrt /boot | tail -n 6
-rw-r--r--  1 root root 1743149 Mar 20 21:45 vmlinuz-2.6.15-prep-Topic105
lrwxrwxrwx  1 root root      28 Mar 20 21:45 vmlinuz -> vmlinuz-2.6.15-
prep-Topic105
-rw-r--r--  1 root root  980796 Mar 20 21:45 System.map-2.6.15-prep-
Topic105
lrwxrwxrwx  1 root root      31 Mar 20 21:45 System.map -> System.map-
2.6.15-prep-Topic105
-rw-r--r--  1 root root 1318741 Mar 20 21:45 initrd-2.6.15-prep-
Topic105.img
drwxr-xr-x  2 root root    4096 Mar 20 21:45 grub
```

Стартовый RAM-диск

Обратите внимание, что в процессе сборки автоматически создается необходимый стартовый RAM-диск (initial RAM disk или initrd). Если у вас возникнет необходимость создать его вручную, это можно сделать при помощи команды mkinitrd. Подробную информацию вы найдете в странице man для этой команды.

Загрузчики

Если процесс идет без сбоев, в ходе выполнения make install будет также обновлена конфигурация загрузчика. В Листинге 18 приведены несколько строк из моего загрузчика.

Листинг 18. Обновленный конфигурационный файл GRUB

```
default=1
timeout=10
splashimage=(hd0,5)/boot/grub/splash.xpm.gz
password --md5 $1$y.uQRs1W$Sqs30hDB3GtE957PoiDWO.
title Fedora Core (2.6.15-prep-Topic105)
        root (hd0,11)
        kernel /boot/vmlinuz-2.6.15-prep-Topic105 ro root=LABEL=FC4-64
rhgb quiet
        initrd /boot/initrd-2.6.15-prep-Topic105.img
title Fedora Core -x86-64 (2.6.15-1.1833_FC4)
```

Запись для нового ядра расположена самой первой, но в качестве записи по умолчанию осталась та, которая была выбрана ранее. Если вы используете не

GRUB, а LILO, выполните команду grubby, которая используется в скрипте, осуществляющем сборку ядра, для обновления конфигурации загрузчика LILO. Если по какой-то причине обновление конфигурации прошло некорректно обратитесь к учебному пособию "Экзамен LPI 101 (тема 102): Инсталляция Linux и управление пакетами". Там вы можете найти подробные инструкции по настройке загрузчика.

В заключение одно замечание. Вас могло удивить, что в примере конфигурации была использована запись -Topic105, а в результате все созданные файлы содержали вместо нее -prep-Topic105. Это одна из применяемых в дистрибутивах Fedora мер по обеспечению безопасности, которая предотвращает неумышленное разрушение имеющегося ядра. Контроль осуществляется путем установки переменной EXTRAVERSION в начальной части главного make-файла, как показано в Листинге 19. Если вы хотите отменить эту возможность, отредактируйте make-файл.

Листинг 19. Начало главного make-файла

```
[root@attic4 linux]# head -n 6 Makefile
VERSION = 2
PATCHLEVEL = 6
SUBLEVEL = 15
EXTRAVERSION = -prep
NAME=Sliding Snow Leopard
```

Перезагрузка

Если все прошло успешно, теперь вы сможете загрузить вашу новую систему. Вы должны выбрать конфигурационную запись для нового ядра, поскольку она не является (еще) записью по умолчанию. Если вас все устроит, можно будет сделать ее записью по умолчанию. После перезагрузки, чтобы проверить, какое загружено ядро, выполните команду uname, как показано в Листинге 20.

Листинг 20. Проверка новой системы

```
[ian@attic4 ~]$ uname -rv
2.6.15-prep-Topic105 #2 Mon Mar 20 21:13:20 EST 2006
```