Впервые опубликовано на developerWorks 24.01.2007

Подготовка к экзамену LPI 101: Устройства, файловые системы Linux и стандарт Filesystem Hierarchy Standard

Младший уровень администрирования (LPIC-1). Тема 104.

Создание разделов и файловых систем

Блочные устройства и разделы

Кратко рассмотрим блочные устройства и разделы. За более подробной информацией вернитесь к руководствам по темам 101 и 102.

Блочные устройства

Блочное устройство представляет собой уровень абстракции, описывающий любое устройство хранения информации, которое может быть разбито на блоки определенного размера; доступ к каждому блоку осуществляется независимо от доступа к другим блокам. Такой доступ часто называют произвольным доступом.

Абстрагированное представление устройств в виде блоков фиксированного размера с произвольным доступом позволяет программам использовать их независимо от того, является ли устройство жестким диском, дискетой, CD- диском, сетевым диском или каким-либо другим устройством, например файловой системой в оперативной памяти.

Примерами блочных устройств могут быть первый жесткий диск (IDE) (/dev/hda) или второй SCSI-диск (/dev/sdb). Для просмотра каталога /dev используйте команду ls -l. Первый символ b в строке указывает на блочное устройство: флоппи- или CD-дисковод, IDE- или SCSI-диск; a c – на символьное устройство, например, накопитель на магнитной ленте или терминал. См. примеры в листинге 1.

Листинг 1. Блочные и символьные устройства Linux

```
[ian@lyrebird ian]$ ls -l /dev/fd0 /dev/hda /dev/sdb /dev/st0 /dev/tty0
brw-rw----    1 ian      floppy     2,   0 Jun 24  2004 /dev/fd0
brw-rw----    1 root     disk       3,   0 Jun 24  2004 /dev/hda
brw-rw----    1 root     disk       8,  16 Jun 24  2004 /dev/sdb
crw-rw----    1 root     disk       9,   0 Jun 24  2004 /dev/st0
crw--w----    1 root     root       4,   0 Jun 24  2004 /dev/tty0
```

Разделы

Для некоторых блочных устройств, таких как дискеты, CD и DVD- диски, принято использовать одну файловую систему на всем носителе. Однако на жестких дисках больших объемов и даже на небольших USB- накопителях доступное пространство принято делить или разбивать на несколько разделов.

Разделы могут отличаться по объему, на каждом из них может быть своя файловая система, так что один диск может использоваться для различных целей, включая использование его несколькими операционными системами. Например, я использую тестовые системы под несколькими различными дистрибутивами Linux, а также иногда под Windows®, и все они используют один или два общих жестких диска.

Из руководств 101 и 102 вы помните, что жесткий диск имеет геометрию, определяемую в терминах цилиндров, головок и секторов. Даже несмотря на то, что современные диски используют логическую адресацию блоков (LBA), которая в значительной степени маскирует геометрию диска, основной единицей размещения для разделов диска остается цилиндр.

Вывод информации о разделах

Информация о разделах диска хранится в таблице разделов. Таблица разделов содержит информацию о начале и окончании каждого раздела, информацию о его типе и о том, является ли он загрузочным или нет. Чтобы создать или удалить раздел, нужно отредактировать таблицу разделов, используя специальную программу. Для экзамена LPI вам необходимо знать программу fdisk, описанную здесь, хотя существуют и другие инструменты.

Для просмотра разделов используется команда fdisk с опцией -l. Если вы хотите просмотреть разделы для конкретного диска, добавьте имя устройства, например /dev/hda. Заметьте, что инструменты для разбиения на разделы требуют административных прав доступа. Листинг 2 показывает разделы на одном из моих жестких дисков.

Листинг 2. Просмотр разделов диска с помощью команды fdisk

```
[root@lyrebird root]# fdisk -l /dev/hda

Disk /dev/hda: 160.0 GB, 160041885696 bytes
255 heads, 63 sectors/track, 19457 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1      2078  16691503+   7  HPFS/NTFS
/dev/hda2          2079      3295   9775552+   c  Win95 FAT32 (LBA)
/dev/hda3          3296      3422   1020127+  83  Linux
/dev/hda4          3423     19457 128801137+   f  Win95 Ext'd (LBA)
/dev/hda5          3423      3684   2104483+  82  Linux swap
/dev/hda6          3685      6234  20482843+  83  Linux
/dev/hda7          6235      7605  11012526   83  Linux
/dev/hda8          7606      9645  16386268+  83  Linux
/dev/hda9          9646     12111  19808113+  83  Linux
/dev/hda10        12112     15680  28667961   83  Linux
/dev/hda11        15681     19457  30338721   83  Linux
```

Замечания:

1. В заголовке отражена информация об объеме диска и его геометрии. Большинство больших дисков, использующих LBA, имеют 255 головок на цилиндр и 63 сектора в дорожке, что составляет 16065 секторов или 8225280 байт на цилиндр.
2. В данном примере первый раздел (/dev/hda1) помечен как загрузочный (или активный). Как вы видели в руководстве к теме 102, это обеспечивает загрузку раздела с помощью стандартной загрузочной записи DOS. Этот признак не имеет смысла в LILO или GRUB- загрузчиках.
3. Столбцы Start и End показывают начальный и конечный цилиндры для каждого раздела. Они не должны перекрываться, а должны следовать строго друг за другом без промежутков.
4. Столбец Blocks показывает число блоков размером 1 килобайт (1024 байт) в разделе. Максимальное количество блоков в разделе, следовательно, равняется половине произведения числа цилиндров (End + 1 - Start) на число секторов в цилиндре. Знак + в конце означает, что используются не все секторы раздела.
5. Поле Id указывает на предполагаемое использование раздела. Тип 82 – файл подкачки, 83 – раздел для хранения информации. Существует около 100 различных типов томов. Данный диск используется несколькими операционными системами, в том числе Windows/XP, поэтому на нем есть разделы с файловой системой NTFS (и FAT32).

Создание разделов с помощью команды fdisk

Только что вы узнали, как просмотреть данные о разделах диска с помощью fdisk. Эта команда также позволяет редактировать таблицу разделов с целью создания и удаления разделов.

Предупреждения

Прежде чем изменять разделы, необходимо запомнить несколько важных моментов. Если не следовать этим рекомендациям, вы рискуете потерять существующую информацию.
Не изменяйте разделы, которые используются в настоящий момент. Составьте план действий и четко его придерживайтесь.
Знайте возможности вашего инструментального средства. Fdisk не выполняет изменений без вашего подтверждения. Другие инструменты, как, например parter, могут применять изменения сразу.
Прежде чем начинать, создайте резервную копию важной информации, так как любая операция может привести к потере данных.
Инструменты для создания разделов диска оперируют таблицей разделов. Если ваш инструмент не выполняет также операций, связанных с перемещением, изменением размера, форматированием или другими способами изменения дискового пространства, ваши данные не будут повреждены. Если вы случайно допустите ошибку, как можно скорее прервите работу и обратитесь за помощью. Возможно, вы еще сможете восстановить разделы и данные.

Запуск fdisk

Для запуска fdisk в интерактивном режиме просто задайте в качестве параметра имя диска, например, /dev/had или /dev/sdb. В следующем примере показана загрузка с рабочего CD-диска Knoppix. Если вы обладаете правами администратора, то получите результат, аналогичный листингу 3.

Листинг 3. Интерактивный запуск fdisk

```
root@ttyp1[knoppix]# fdisk  /dev/hda

The number of cylinders for this disk is set to 14593.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):
```

Современные диски содержат более 1024 цилиндров, поэтому обычно вы будете получать предупреждение, как в листинге 3. Нажмите m, чтобы получить список возможных однобуквенных команд, показанный в листинге 4.

Листинг 4. Помощь в fdisk

```
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Для просмотра разделов диска нажмите p; результат - в листинге 5.

Листинг 5. Просмотр существующей таблицы разделов

```
Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        2611    20972826    7  HPFS/NTFS
```

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

Объем данного диска 120 ГБ, имеется раздел под Windows XP, занимающий около 20 ГБ. Это первичный раздел, помеченный как загрузочный, что типично для Windows-систем.

Формирование структуры диска для рабочей станции

Теперь используем часть свободного пространства для организации рабочей станции со следующими дополнительными разделами. На практике вы вряд ли будете использовать такое количество разных типов файловых систем, но здесь мы сделаем это для примера.

1. Еще один первичный раздел для наших загрузочных файлов. Он будет монтироваться как /boot и содержит файлы ядра и исходных RAM-дисков. Если вы используете загрузчик GRUB, то его файлы тоже будут располагаться здесь. Из руководства к теме 102 следует, что для этого необходимо около 100 Мбайт. Из листинга 5 видно, что объем цилиндра примерно 8 Мбайт, поэтому загрузочный раздел /boot займет 13 цилиндров. Это будет /dev/hda2.
2. Создадим расширенный раздел для размещения логических разделов, занимающий остальное свободное пространство. Это будет /dev/hda3.
3. Создадим раздел подкачки размером 500 Мбайт как /dev/hda5. Он займет 64 цилиндра.
4. Создадим логический раздел объемом около 20 ГБ для нашей Linux- системы. Это будет /dev/hda6.
5. Создадим отдельный раздел для данных пользователя размером 10 ГБ. В дальнейшем он будет монтироваться как /home, а пока это будет просто /dev/hda7.
6. И наконец, создадим маленький (2 ГБ) раздел для обмена данными между системами Linux и Windows. В дальнейшем на нем будет использоваться файловая система FAT32 (или vfat). Это будет /dev/hda8.

Создание разделов

Начнем с использования команды n для создания нового раздела; см. листинг 6.

Листинг 6. Создание первого раздела

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (2612-14593, default 2612):
Using default value 2612
Last cylinder or +size or +sizeM or +sizeK (2612-14593, default 14593):
2624

Command (m for help): p

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

```
   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        2611    20972826    7  HPFS/NTFS
/dev/hda2            2612        2624      104422+  83  Linux

Command (m for help):
```

Мы берем значение по умолчанию для первого цилиндра и задаем значение 2624 для последнего цилиндра, в результате получаем раздел из 13 цилиндров. Из листинга 6 видно, что наш раздел в действительности занимает примерно 100 Мбайт. Поскольку это первичный раздел, он должен иметь номер от 1 до 4. Рекомендуется назначать номера разделов последовательно; если этого не делать, некоторые инструменты выдают предупредительные сообщения.

Заметьте также, что наш новый раздел будет иметь тип 83, то есть раздел для хранения данных в Linux. Это можно рассматривать как указатель операционной системы, которую планируется использовать на этом разделе. Дальнейшее использование должно быть согласовано с этим, но в данный момент мы даже не будем форматировать раздел, не говоря уж о том, чтобы размещать на нем какую-либо информацию.

Теперь создадим расширенный раздел, который будет содержать логические разделы диска. Присвоим этому разделу номер 3 (/dev/hda3). Процесс и результат показан в листинге 7. Заметьте, что тип раздела назначается автоматически.

Листинг 7. Создание расширенного раздела

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
Partition number (1-4): 3
First cylinder (2625-14593, default 2625):
Using default value 2625
Last cylinder or +size or +sizeM or +sizeK (2625-14593, default 14593):
Using default value 14593

Command (m for help): p

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        2611    20972826    7  HPFS/NTFS
/dev/hda2            2612        2624      104422+  83  Linux
/dev/hda3            2625       14593    96140992+   5  Extended

Command (m for help):
```

Теперь перейдем к раздела файла подкачки как логического раздела внутри нашего

расширенного раздела. Мы задаем для конечного цилиндра значение +64 (цилиндра) вместо того, чтобы считать самим. Отметьте, что при этом мы используем команду t, чтобы задать для вновь создаваемого раздела тип 82 (раздел подкачки Linux). Иначе это будет раздел с типом 83 (данные Linux).

Листинг 8. Создание раздела подкачки

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (2625-14593, default 2625):
Using default value 2625
Last cylinder or +size or +sizeM or +sizeK (2625-14593, default 14593):
+64

Command (m for help): t
Partition number (1-5): 5
Hex code (type L to list codes): 82
Changed system type of partition 5 to 82 (Linux swap)

Command (m for help): p

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        2611    20972826    7  HPFS/NTFS
/dev/hda2            2612        2624      104422+  83  Linux
/dev/hda3            2625       14593    96140992+   5  Extended
/dev/hda5            2625        2689      522081   82  Linux swap

Command (m for help):
```

Теперь определим основной раздел для Linux и раздел /home. Для этого просто зададим объемы +20480 Мбайт и +10240 Мбайт, т.е. 20 ГБ и 10 ГБ соответственно. Предоставим fdisk самостоятельно подсчитать число цилиндров. Результаты представлены в листинге 9.

Листинг 9. Создание основного раздела Linux

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (2690-14593, default 2690):
Using default value 2690
Last cylinder or +size or +sizeM or +sizeK (2690-14593, default 14593):
+20480M
```

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (5181-14593, default 5181):
Using default value 5181
Last cylinder or +size or +sizeM or +sizeK (5181-14593, default 14593):
+10240M

Command (m for help): p

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        2611    20972826    7  HPFS/NTFS
/dev/hda2            2612        2624      104422+  83  Linux
/dev/hda3            2625       14593    96140992+   5  Extended
/dev/hda5            2625        2689      522081   82  Linux swap
/dev/hda6            2690        5180    20008926   83  Linux
/dev/hda7            5181        6426    10008463+  83  Linux

Command (m for help):
```

Последний раздел – раздел с файловой системой FAT32. Выполним уже знакомые действия для создания раздела /dev/hda9, определив объем как +2048 Мбайт, а затем изменим тип раздела на b (для FAT32 в версии Windows 95). Затем сохраним изменения.

Сохранение таблицы разделов

До настоящего времени мы редактировали таблицу разделов в оперативной памяти. Можно использовать команду q для выхода без сохранения изменений. Если что-то выполнено не так, как нужно, можно использовать d для удаления одного или более разделов и переопределить их заново. Если все сделано верно, используем v для проверки, а затем w, чтобы сохранить новую таблицу разделов и выйти. Смотрите листинг 10. Если вновь запустить fdisk -l, увидим, что изменения уже применены в Linux. В отличие от некоторых других операционных систем, для того, чтобы увидеть эти изменения, не всегда необходима перезагрузка. Перезагрузка может потребоваться, например, если раздел /dev/hda3 переназначается в /dev/hda2 из-за, того, что раздел /dev/hda2 был удален. Если перезагрузка необходима, fdisk сообщит вам об этом.

Листинг 10. Сохранение таблицы разделов.

```
Command (m for help): v
127186915 unallocated sectors

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
```

```
WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
root@ttyp0[knoppix]# fdisk -l /dev/hda

Disk /dev/hda: 120.0 GB, 120034123776 bytes
255 heads, 63 sectors/track, 14593 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        2611    20972826    7  HPFS/NTFS
/dev/hda2            2612        2624      104422+  83  Linux
/dev/hda3            2625       14593    96140992+   5  Extended
/dev/hda5            2625        2689      522081   82  Linux swap
/dev/hda6            2690        5180    20008926   83  Linux
/dev/hda7            5181        6426    10008463+  83  Linux
/dev/hda8            6427        6676     2008093+   b  W95 FAT32
```

Подробнее про fdisk

Можно отметить, что мы не изменяли загрузочный раздел. Наш диск по-прежнему имеет главную загрузочную запись Windows и соответственно будет загружаться с первого раздела, который помечен как загрузочный (раздел NTFS в нашем примере).

Ни LILO, ни GRUB не используют флаг загрузочного раздела. Если какой-либо из этих загрузчиков будет установлен в главной загрузочной записи, он может загрузить раздел Windows XP. Также можно установить LILO или GRUB в раздел /boot (/dev/hda2), пометить этот раздел как загрузочный и удалить загрузочный флажок с раздела /dev/hda1. Оставить первоначальную загрузочную запись полезно, если впоследствии на машине вновь будет использоваться только Windows.

Мы рассмотрели один из способов формирования рабочей станции в Linux. Другие возможности описаны далее в руководстве, в разделе Нахождение и размещение системных файлов.

Типы файловых систем

Linux поддерживает несколько различных типов файловых систем. Каждая имеет свои достоинства, недостатки и отличительные черты. Важное свойство файловой системы – журналирование – позволяет быстро восстановить систему после сбоя. Как правило, журналируемые системы предпочтительнее нежурналируемых, если у вас есть выбор. Ниже приведен краткий обзор типов файловых систем, которые необходимо знать для экзамена LPI. Более подробную информацию см. в разделе Ресурсы

Файловая система ext2

Файловая система ext2 (также известная как вторая расширенная файловая

система) разработана для устранения недостатков в системе Minix, использовавшейся в ранних версиях Linux. Она широко использовалась в Linux в течение длительного времени. Ext2 не журналируется и в значительной степени вытеснена ext3.

Файловая система ext3

Файловая система ext3 дополняет возможности стандартной ext2 журналированием и поэтому представляет собой эволюционное развитие очень стабильной файловой системы. Она обеспечивает разумную производительность в большинстве ситуаций и продолжает совершенствоваться. Поскольку она представляет собой расширенный вариант системы ext2, есть возможность преобразовывать систему ext2 в ext3 и, в случае необходимости, обратно.

Файловая система ReiserFS

ReiserFS – это файловая система, основанная на B-дереве, с очень хорошими рабочими характеристиками, особенно для большого числа маленьких файлов. ReiserFS хорошо масштабируется и является журналируемой.

Файловая система XFS

XFS – журналируемая файловая система. Она имеет ряд эффективных функций и оптимизирована для масштабирования. XFS активно кэширует перемещаемую информацию в оперативной памяти, поэтому при использовании этой системы рекомендуется иметь источник бесперебойного питания.

Файловая система раздела подкачки

Пространство для подкачки должно быть отформатировано, но обычно оно не рассматривается как отдельная файловая система.

Файловая система vfat

Эта файловая система (также известная как FAT32) не является журналируемой и имеет множество недостатков по сравнению с файловыми системами, используемыми Linux. Она применяется для обмена данными между системами Windows и Linux, поскольку читается обеими. Не используйте эту файловую систему в Linux, за исключением случаев совместного использования данных системами Windows и Linux. Если распаковать архив Linux на диск с системой vfat, вы потеряете права доступа, например на выполнение программ, а также символические ссылки, которые могли храниться в архиве.

Как ext3, так и ReiserFS являются зрелыми файловыми системами и используются по умолчанию в ряде дистрибутивов. Обе они рекомендованы к широкому использованию.

Создание файловых систем

Для создания файловых систем в Linux используется команда mkfs, а для создания раздела подкачки – команда mkswap. Команда mkfs фактически является

интерфейсом доступа к целому ряду команд, специфичных для конкретных файловых систем, например, mkfs.ext3 для ext3, mkfs.reiserfs для ReiserFS.

Поддержка каких файловых систем имеется в вашей системе? Чтобы это выяснить, используйте команду ls /sbin/mk*. Пример представлен в листинге 11.

Листинг 11. Команды для создания файловых систем

```
root@ttyp0[knoppix]# ls /sbin/mk*
/sbin/mkdosfs      /sbin/mkfs.ext2   /sbin/mkfs.msdos     /sbin/mkraid
/sbin/mke2fs       /sbin/mkfs.ext3   /sbin/mkfs.reiserfs
/sbin/mkreiserfs
/sbin/mkfs         /sbin/mkfs.jfs    /sbin/mkfs.vfat      /sbin/mkswap
/sbin/mkfs.cramfs  /sbin/mkfs.minix  /sbin/mkfs.xfs
```

Отметьте различные формы некоторых команд. Например, команды mke2fs, mkfs.ext2 и mkfs.ext3 равнозначны, как и mkreiserfs и mkfs.reiserfs.

Существует несколько общих опций для всех mkfs-команд. Опции, которые специфичны для создаваемой файловой системы, передаются командам создания в зависимости от типа, определенного параметром -type. В наших примерах используется mkfs -type, но можно использовать и другие формы с тем же результатом. Например, можно использовать mkfs -type reiserfs, mkreiserfs или mkfs.reiserfs. Для вызова справочных страниц по конкретной файловой системе укажите в качестве имени соответствующую команду mkfs, например, man mkfs.reiserfs. Многие значения, приведенные в нижеследующих примерах вывода, управляются опциями для mkfs.

Создание файловой системы ext3

Листинг 12. Создание файловой системы ext3

```
root@ttyp0[knoppix]# mkfs -t ext3 /dev/hda8
mke2fs 1.35 (28-Feb-2004)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
251392 inodes, 502023 blocks
25101 blocks (5.00%) reserved for the super user
First data block=0
16 block groups
32768 blocks per group, 32768 fragments per group
15712 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 32 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Полезная опция, используемая при создании ext2 и ext3 – опция -L с именем, которая назначает метку тому. Метку можно использовать при монтировании файловой системы вместо имени устройства; это обеспечивает определенный уровень изоляции в отношении изменений, который необходимо отразить в различных управляющих файлах. Для просмотра и установки метки на существующую систему ext2 или ext3 используется команда e2label. Длина метки ограничена 16 символами.

Следует заметить, что ext3 ведет журнал. Если вы хотите добавить журналирование к существующей системе ext2, используйте команду tune2fs с опцией -j.

Создание файловой системы ReiserFS

Листинг 13. Создание файловой системы ReiserFS

```
.root@ttyp0[knoppix]# mkfs -t reiserfs /dev/hda6
mkfs.reiserfs 3.6.17 (2003 www.namesys.com)

A pair of credits:
Many persons came to www.namesys.com/support.html,  and got a question
answered
for $25, or just gave us a small donation there.

Jeremy  Fitzhardinge  wrote  the  teahash.c  code  for  V3.  Colin  Plumb
also
contributed to that.

Guessing about desired format. Kernel 2.4.26 is running.
Format 3.6 with standard journal
Count of blocks on the device: 5002224
Number of blocks consumed by mkreiserfs formatting process: 8364
Blocksize: 4096
Hash function used to sort names: "r5"
Journal Size 8193 blocks (first block 18)
Journal Max transaction length 1024
inode generation number: 0
UUID: 72e317d6-8d3a-45e1-bcda-ad7eff2b3b40
ATTENTION: YOU SHOULD REBOOT AFTER FDISK!
        ALL DATA WILL BE LOST ON '/dev/hda6'!
Continue (y/n):y
Initializing journal - 0%....20%....40%....60%....80%....100%
Syncing..ok

Tell your friends to use a kernel based on 2.4.18 or later, and
especially not a
kernel based on 2.4.9, when you use reiserFS. Have fun.

ReiserFS is successfully created on /dev/hda6.
```

Для задания метки тома используйте -l (или опцию --label с именем). Для добавления или просмотра метки к существующей системе ReiserFS используется команда reiserfstune. Максимальное число символов в метке – 16.

Создание файловой системы XFS

Листинг 14. Создание файловой системы XFS

```
root@ttyp0[knoppix]# mkfs -t xfs /dev/hda7
meta-data=/dev/hda7              isize=256    agcount=16, agsize=156382
blks
         =                       sectsz=512
data     =                       bsize=4096   blocks=2502112, imaxpct=25
         =                       sunit=0      swidth=0 blks, unwritten=1
naming   =version 2              bsize=4096
log      =internal log           bsize=4096   blocks=2560, version=1
         =                       sectsz=512   sunit=0 blks
realtime =none                   extsz=65536  blocks=0, rtextents=0
```

Для задания метки тома в системе XFS используется опция -L с именем. Для добавления метки к существующей файловой системе XFS используется команда xfs_admin с опцией -L. Для просмотра метки используется команда xfs_admin с опцией -l. В отличие от ext2, ext3 и ReiserFS максимальное число символов в метке составляет 12.

Создание файловой системы vfat

Листинг 15. Создание файловой системы vfat

```
root@ttyp0[knoppix]# mkfs -t vfat /dev/hda8
mkfs.vfat 2.10 (22 Sep 2003)
```

Метка тома в системе FAT32 назначается с помощью опции -n. Команда e2label отображает или устанавливает метку тома в системе vfat, а также в разделах ext. Длина метки ограничена 16 символами.

Создание пространства подкачки

Листинг 16. Создание пространства подкачки

```
root@ttyp0[knoppix]# mkswap /dev/hda5
Setting up swapspace version 1, size = 534605 kB
```

Разделы подкачки, в отличие от обыкновенных файловых систем, не монтируются, а активизируются командой swapon. Стартовые сценарии Linux автоматически активизируют разделы подкачки.

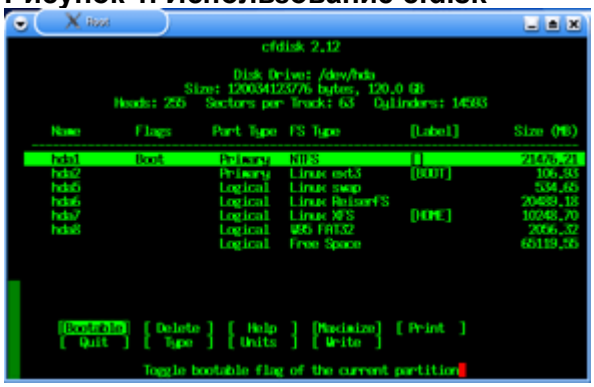
Другие инструменты и файловые системы

Следующие инструменты и файловые системы не входят в цели данного экзамена LPI. Здесь мы приводим очень краткий обзор инструментов и файловых систем, которые вам могут встретиться.

Инструменты для создания разделов

Многие дистрибутивы Linux содержат команды cfdisk и sfdisk. Команда cfdisk предоставляет более удобный графический интерфейс, чем fdisk, используя библиотеку функций ncurses, как показано на рисунке 1. Команда sfdisk предназначена для использования программистами и допускает использование сценариев. Применяйте ее, только если умеете ею пользоваться.

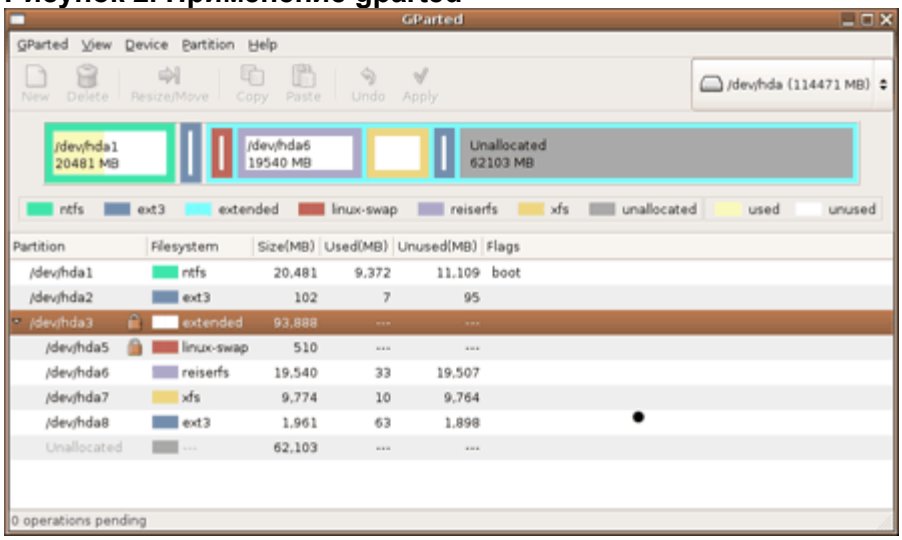
Рисунок 1. Использование cfdisk



Другим распространенным инструментом для работы с таблицей разделов является команда parted, с помощью которой можно изменять и формировать множество типов разделов, а также создавать и удалять их. Для изменения объема NTFS-раздела вместо команды parted используется ntfsresize. Команда qtparted использует графический интерфейс на базе Qt. Она выполняет как функции parted, так и ntfsresize.

Команда gparted – еще один инструмент с графическим интерфейсом, разработанный для среды GNOME. Она использует библиотеки GTK+GUI, как показано на рисунке 2. (Информацию о qtparted и gparted см. в разделе Ресурсы).

Рисунок 2. Применение gparted



Имеется также ряд коммерческих инструментов для создания дисковых разделов. Возможно, наиболее известный из них – PartitionMagic, теперь распространяемый Symantec.

Многие дистрибутивы позволяют делить диск на разделы, а иногда также сжимать существующие разделы Windows NTFS или FAT32 в процессе установки. Более точную информацию см. в руководстве по установке конкретного дистрибутива.

Диспетчер логических томов

Диспетчер логических томов (LVM) для Linux позволяет объединять несколько физических устройств хранения в единую группу томов. Например, можно добавить раздел к существующей группе томов, вместо того чтобы искать необходимое для вашей файловой системы непрерывное дисковое пространство.

RAID

RAID (резервированный массив независимых дисков) – это технология, обеспечивающая надежное хранение информации с использованием недорогих дисков, которые гораздо доступнее используемых в системах высшей ценовой категории. Существует несколько типов RAID-массивов. Технология RAID может быть реализована как на аппаратном уровне, так и на программном. Linux поддерживает оба варианта.

Другие файловые системы

Вам также могут встретиться файловые системы, не рассмотренные здесь.

Journaled File System (JFS) от IBM, в настоящее время используемая в корпоративных серверах компании IBM, разработана для серверных сред с высокой пропускной способностью. Она реализована для Linux и входит в состав некоторых дистрибутивов. Для создания файловой системы JFS используется команда mkfs.jfs.

Существуют и другие файловые системы, например cramfs, часто используемая встроенными устройствами.

В следующем разделе рассказывается, как поддерживать целостность файловой системы и что делать при возникновении неполадок.

Целостность файловых систем

Контроль свободного пространства

Сначала - краткий обзор. Из руководства к теме 103, "Подготовка к экзамену LPI 101: команды GNU и UNIX", вы узнали, что файл или каталог размещаются в группе блоков, а информация о файле или каталоге содержится в inode.

И блоки данных, и блоки inode занимают место в файловой системе, поэтому необходимо контролировать используемое пространство, чтобы быть уверенным в наличии свободного места на диске для расширения файловой системы.

df

Команда df выводит информацию о монтированных файловых системах. (Подробнее о монтировании файловых систем – в следующем разделе Монтирование и размонтирование файловых систем). Если добавить опцию -T, к выводу будет добавлен тип файловой системы. Результат выполнения команды df в системе Ubuntu, установленной на файловые системы, созданные в предыдущем разделе, показан в листинге 17.

Листинг 17. Вывод информации об использовании файловых систем

```
ian@pinguino:~$ df -T
Filesystem    Type   1K-blocks      Used Available Use% Mounted on
/dev/hda6 reiserfs    20008280   1573976  18434304   8% /
tmpfs        tmpfs     1034188         0   1034188   0% /dev/shm
tmpfs        tmpfs     1034188     12588   1021600   2%
/lib/modules/2.6.12-10-386/volatile
/dev/hda2     ext3      101105     19173     76711  20% /boot
/dev/hda8     vfat     2004156         8   2004148   1% /dos
/dev/hda7      xfs     9998208      3544   9994664   1% /home
/dev/hda1     ntfs    20967416   9594424  11372992  46% /media/hda1
```

Заметьте, что вывод содержит общее число блоков, а также число используемых и свободных блоков. Также указывается файловая система, например, /dev/hda7, и ее точка монтирования: /home для /dev/hda7. Две записи tmpfs относятся к файловым системам в виртуальной памяти. Они существуют только в оперативной памяти или пространстве подкачки и создаются в момент монтирования без использования команды mkfs. Подробнее о команде tmpfs – в части «Общие курсы: расширенное руководство по реализации файловых систем, Часть 3» (см. ссылку в разделе

Ресурсы).

Если необходимо вывести данные об использовании inode, применяется команда df с опцией -i. Можно исключить вывод данных по определенной файловой системе, используя опцию -x, или ограничить информацию определенными типами файловых систем, используя опцию -t. При необходимости их можно использовать несколько раз. Примеры представлены в листинге 18.

Листинг 18. Просмотр inode

```
ian@pinguino:~$ df -i -x tmpfs
Filesystem            Inodes   IUsed   IFree IUse% Mounted on
/dev/hda6                  0       0       0    -  /
/dev/hda2              26208      34   26174    1% /boot
/dev/hda8                  0       0       0    -  /dos
/dev/hda7           10008448     176 10008272    1% /home
/dev/hda1              37532   36313    1219   97% /media/hda1
ian@pinguino:~$ df -iT -t vfat -t ext3
Filesystem    Type    Inodes   IUsed   IFree IUse% Mounted on
/dev/hda2     ext3     26208      34   26174    1% /boot
/dev/hda8     vfat         0       0       0    -  /dos
```

Возможно, вас не удивит то, что для системы FAT32 не отображаются inodes, но неожиданностью может стать то, что и для ReiserFS их тоже нет в выводе. ReiserFS содержит метаданные о файлах и каталогах в объектах stat items. Вследствие того, что в ReiserFS используется сбалансированная древовидная структура, в ней нет заранее определенного числа inodes, в отличие, например, от файловых систем ext2, ext3 или xfs.

Кроме того, существуют некоторые другие опции команды df, используемые для ограничения вывода локальными файловыми системами или для контроля формата вывода. Например, используйте опцию -H для вывода результатов в удобном для пользователя формате (например, 1K для 1024), или опцию -h (или - --si) для отражения размеров в десятичном представлении (1K=1000).

Если вы не знаете точно, какая файловая система используется для определенной части вашего дерева каталогов, можно применить команду df с указанием пути или даже имени файла в качестве параметра, как показано в листинге 19.

Листинг 19. Удобочитаемый формат вывода результатов df

```
ian@pinguino:~$ df --si ~ian/index.html
Filesystem             Size   Used  Avail Use% Mounted on
/dev/hda7               11G   3.7M    11G   1% /home
```

du

Команда df выводит информацию только о файловой системе в целом. Иногда необходимо узнать, сколько места занимает каталог home, или какой размер раздела потребуется, чтобы разместить каталог /usr в отдельной файловой системе.

Для решения этих задач используется команда du.

Команда du выводит информацию о файле (файлах), имена которых заданы в качестве параметров. Если задано имя каталога, то du определяет размер всех файлов и подкаталогов этого каталога на всех уровнях вложения. Результат работы команды может быть очень объемным. К счастью, существует опция -s для вывода сводной информации по каталогу. Если использовать du для получения информации о нескольких каталогах, можно добавить опцию -c для вывода суммарных данных. Также можно задавать формат вывода. Для этого применяются опции, аналогичные используемым в команде df (-h, -H, --si и т.п.). Листинг 20 показывает два варианта вывода для моего каталога home во вновь установленной системе Ubuntu.

Листинг 20. Использование du

```
ian@pinguino:~$ du -hc *
0       Desktop
16K     index.html
16K     total
ian@pinguino:~$ du -hs .
3.0M    .
```

Причина различия между результатом команды du -c *, насчитавшей 16 КБ, и команды du -s, получившей объём 3 МБ, в том, что последняя включает файлы и каталоги, начинающиеся с точки, такие как .bashrc, которые не просматриваются первой.

Еще следует отметить, что для использования du вы должны иметь права чтения каталогов, к которым вы ее применяете.

Теперь применим du для просмотра общего объема, занимаемого каталогом /usr и всеми его подкаталогами первого уровня. Результат представлен в листинге 21. Чтобы с гарантией иметь соответствующие права доступа, используйте полномочия root.

Листинг 21. Использование du для каталога /usr

```
root@pinguino:~# du -shc /usr/*
66M     /usr/bin
0       /usr/doc
1.3M    /usr/games
742K    /usr/include
0       /usr/info
497M    /usr/lib
0       /usr/local
7.3M    /usr/sbin
578M    /usr/share
0       /usr/src
14M     /usr/X11R6
1.2G    total
```

Проверка файловых систем

Иногда в системе может произойти сбой или отключиться питание. В этих случаях Linux не может аккуратно размонтировать файловые системы, и они могут оказаться в несогласованном состоянии. Работать с поврежденной файловой системой не следует, поскольку это скорее всего приведет к усугублению имеющихся ошибок.

Основной инструмент для проверки файловых систем - команда fsck, которая, аналогично mkfs, является интерфейсом доступа к командам проверки различных типов файловых систем. Несколько примеров таких команд приведено в листинге 22.

Листинг 22. Примеры программ fsck.

```
ian@pinguino:~$ ls /sbin/*fsck*
/sbin/dosfsck      /sbin/fsck.ext3   /sbin/fsck.reiser4
/sbin/jfs_fscklog
/sbin/e2fsck       /sbin/fsck.jfs    /sbin/fsck.reiserfs
/sbin/reiserfsck
/sbin/fsck         /sbin/fsck.minix  /sbin/fsck.vfat
/sbin/fsck.cramfs  /sbin/fsck.msdos  /sbin/fsck.xfs
/sbin/fsck.ext2    /sbin/fsck.nfs    /sbin/jfs_fsck
```

Процесс загрузки системы с помощью команды fsck проверяет корневую файловую систему и другие файловые системы, указанные в управляющем файле /etc/fstab. Если файловая система не была размонтирована корректно, проводится проверка целостности системы. Это определяется значением поля pass (или passno) (шестое поле записи /etc/fstab). Файловые системы со значением pass, установленным в ноль, не тестируются во время загрузки. Корневая файловая система имеет значение pass, равное 1, и тестируется первой. Другие файловые системы обычно имеют значение pass от двух и выше, которое указывает, в каком порядке их надо проверять. Несколько операций fsck могут выполняться параллельно, поэтому различные файловые системы могут иметь одинаковые значения pass, как в нашем примере системы /boot и /home.

Листинг 23. Тестирование системы при загрузке на основании данных fstab.

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/hda6       /               reiserfs defaults       0       1
/dev/hda2       /boot           ext3    defaults        0       2
/dev/hda8       /dos            vfat    defaults        0       0
/dev/hda7       /home           xfs     defaults        0       2
```

Следует отметить, что некоторые журналируемые файловые системы, такие как ReiserFS и xfs, могут иметь значение pass, установленное в 0, поскольку проверку и восстановление файловой системы производит программа журналирования, а не fsck.

Восстановление файловых систем

Если автоматическая проверка при загрузке не может восстановить согласованность файловой системы, обычно происходит переход в однопользовательскую командную оболочку и выводится сообщение с указаниями по ручному запуску fsck. В системе ext2, которая не журналируется, вам может быть представлена серия вопросов для подтверждения операций по восстановлению файловой системы. Как правило, рекомендуется следовать предложениям fsck по восстановлению системы, выбирая y (для подтверждения операции). Когда система перезагрузится, проверьте, не пропала ли какая-либо информация или файлы.

Если вы заподозрили порчу данных или хотите запустить проверку вручную, большинство программ требуют сначала размонтировать файловую систему. Поскольку размонтировать корневую файловую систему работающей системы невозможно, максимум, что можно сделать - перейти в однопользовательский режим (используя telinit 1), а затем перемонтировать корневую файловую систему в режиме «только чтение»; после этого можно провести проверку согласованности. (Монтирование файловых систем описано в следующем разделе – Монтирование и размонтирование файловых систем.) Наилучший способ проверки файловых систем – загрузиться в резервную систему с CD-диска или USB- накопителя и провести проверку ваших файловых систем в размонтированном виде.

Преимущества журналирования

Для проверки системы ext2 с помощью команды fsck может потребоваться значительное время, поскольку при этом необходимо полное чтение внутренней структуры данных (метаданных). Поскольку файловые системы становятся все больше и больше, это занимает все больше и больше времени; несмотря на то, что быстродействие дисков растет, полная проверка может занять до нескольких часов.

Эта проблема побудила к созданию журналируемых файловых систем. Такие файловые системы хранят недавние изменения в метаданных. После сбоя, чтобы определить, в каких частях файловой системы в результате сбоя могли возникнуть ошибки, драйвер файловой системы просматривает журнал. Это позволяет сократить время проверки целостности файловой системы до нескольких секунд, независимо от ее размера. Более того, драйвер файловой системы обычно проверяет файловую систему на этапе монтирования, поэтому дополнительная проверка с помощью fsck, как правило, не требуется. Фактически в файловой системе xfs команде fsck делать нечего!

Перед инициированием проверки файловой системы вручную следует уточнить параметры конкретной команды fsck по документации. В примерах, представленных в листинге 24, команда fsck запускается с рабочего компакт-диска Ubuntu.

Листинг 24. Ручной запуск fsck

```
root@ubuntu:~# fsck -p /dev/hda6
fsck 1.38 (30-Jun-2005)
Reiserfs super block in block 16 on 0x306 of format 3.6 with standard
journal
Blocks (total/free): 5002224/4608416 by 4096 bytes
Filesystem is clean
Replaying journal..
Reiserfs journal '/dev/hda6' in blocks [18..8211]: 0 transactions
replayed
```

```
Checking internal tree..finished
root@ubuntu:~# fsck -p /dev/hda2
fsck 1.38 (30-Jun-2005)
BOOT: clean, 34/26208 files, 22488/104420 blocks
root@ubuntu:~# fsck -p /dev/hda7
fsck 1.38 (30-Jun-2005)
root@ubuntu:~# fsck -a /dev/hda8
fsck 1.38 (30-Jun-2005)
dosfsck 2.11, 12 Mar 2005, FAT32, LFN
/dev/hda8: 1 files, 2/501039 clusters
```

«Продвинутые» инструменты

Также существуют более функциональные средства для проверки и восстановления файловых систем. Правила использования можно найти в документации man, а практические рекомендации – в Linux Documentation Project (см. Ресурсы). Почти все эти команды требуют, чтобы файловая система была размонтирована, хотя некоторые функции могут использоваться в файловых системах, смонтированных в режиме «только чтение». Некоторые из этих команд описаны далее.

Прежде чем предпринимать какие-либо исправления, обязательно создавайте резервную копию файловой системы.

Инструменты для файловых систем ext2 и ext3

tune2fs
: Настраивает параметры файловых систем ext2 и ext3. Используется для добавления журнала к системе ext2, делая, таким образом, из нее ext3, а также выводит или устанавливает максимальное число монтирований, после которого необходима проверка. вы также можете задать метку и назначить или запретить выполнение дополнительных опций.

dumpe2fs
: Выводит информацию о дескрипторах суперблоков и групп блоков в файловых системах ext2 и ext3.

debugfs
: Команда для интерактивной отладки файловой системы. Используйте ее для проверки или изменения состояния файловых систем ext2 или ext3.

Инструменты для файловых систем ReiserFS

reiserfstune
: Выводит и настраивает параметры файловой системы ReiserFS.

debugreiserfs
: Выполняет функции, аналогичные dumple2fs и debugfs, для файловой системы ReiserFS.

Инструменты для файловой системы XFS

xfs_info
: Выводит информацию о системе XFS.

xfs_growfs
: Расширяет файловую систему XFS (если имеется дополнительный раздел).

xfs_admin

Изменяет параметры файловой системы XFS.

xfs_repair

Восстанавливает файловую систему XFS, когда проверок при монтировании установке недостаточно для восстановления системы.

xfs_db

Проверяет или отлаживает файловую систему XFS.

Монтирование и размонтирование файловых систем

Монтирование файловых систем

Файловая система Linux представляет собой единое большое дерево с корнем /. Тем не менее мы говорим о файловых системах различных устройств и разделов. Сейчас мы разрешим это кажущееся несоответствие. Корневая файловая система монтируется в процессе инициализации. Все остальные созданные нами файловые системы не могут быть использованы системой Linux, пока они не будут смонтированы в точку монтирования.

Точка монтирования – это просто каталог в текущей совокупности смонтированных файловых систем, где файловая система данного устройства прикрепляется к общему дереву. Монтирование – это процесс, который делает файловую систему устройства частью единой файловой системы, доступной для Linux. Например, можно монтировать файловые системы на разделах жесткого диска, таких как /boot, /tmp или /home, а также на дискетах - /mnt/floppy и на CD-ROM - /media/cdrom1.

Кроме файловых систем на разделах, дискетах и CD, существуют и другие типы файловых систем. Мы вкратце упоминали файловую систему tmpfs, являющуюся файловой системой в виртуальной памяти. Также можно монтировать одну файловые системы одного компьютера на другом компьютере, используя сетевые файловые системы, такие как NFS или AFS. Можно создать файл в файловой системе, отформатировать его как файловую систему (возможно, другого типа) и смонтировать эту новую файловую систему.

Хотя процесс монтирования фактически монтирует файловую систему какого-либо устройства (или другого ресурса), принято говорить, что вы "монтируете устройство", понимая под этим "монтирование файловой системы устройства".

Базовая форма команды mount имеет два параметра: устройство (или ресурс), содержащие монтируемую файловую систему, и точка монтирования. Например, смонтируем наш раздел с системой FAT32 /dev/hda8 в точке монтирования /dos, как показано в листинге 25.

Листинг 25. Монтирование /dos

```
root@pinguino:~# mount /dev/hda8 /dos
```

Точка монтирования должна существовать прежде, чем в нее что-либо будет

смонтировано. В результате монтирования файлы и подкаталоги монтируемой файловой системы становятся файлами и подкаталогами точки монтирования. Если каталог точки монтирования уже содержал файлы и подкаталоги, они становятся невидимыми до тех пор, пока файловая система не будет демонтирована. Хороший способ избежать этого – использовать в качестве точек монтирования только пустые каталоги.

После монтирования файловой системы файлы и каталоги, созданные или скопированные в точку монтирования или в ее подкаталог, будут располагаться в смонтированной файловой системе. Так, в нашем примере, файл /dos/sampdir/file.txt будет создан в системе FAT32, смонтированной в точке /dos.

Обычно команда mount автоматически определяет тип файловой системы. Но иногда может потребоваться явное задание типа файловой системы, для чего используется опция -t, как показано в листинге 26.

Листинг 26. Монтирование с явным заданием типа файловой системы

```
root@pinguino:~# mount -t vfat /dev/hda8 /dos
```

Чтобы увидеть, какие файловые системы смонтированы, используйте mount без параметров. В Листинге 27 приведен пример для нашей системы.

Листинг 27. Просмотр смонтированных файловых систем

```
/dev/hda6 on / type reiserfs (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
usbfs on /proc/bus/usb type usbfs (rw)
tmpfs on /lib/modules/2.6.12-10-386/volatile type tmpfs (rw,mode=0755)
/dev/hda2 on /boot type ext3 (rw)
/dev/hda8 on /dos type vfat (rw)
/dev/hda7 on /home type xfs (rw)
/dev/hda1 on /media/hda1 type ntfs (rw)
tmpfs on /dev type tmpfs (rw,size=10M,mode=0755)
```

Аналогичную информацию можно просмотреть с помощью команд /proc/mounts или /ets/mtab; обе они выводят информацию о смонтированных файловых системах.

Опции монтирования

Команда mount имеет несколько опций, которые меняют ее поведение по сравнению с поведением по умолчанию. Например, можно смонтировать файловую систему «только для чтения», указав атрибут -o ro. Если файловая система уже смонтирована – добавьте remount, как показано в листинге 28.

Листинг 28. Установка атрибута "только чтение"

```
root@pinguino:~# mount -o remount,ro /dos
```

Замечания:

- указывайте опции через запятую;
- при перемонтировании уже смонтированной файловой системы достаточно определить либо точку монтирования, либо название устройства. Указывать и то и другое не обязательно;
- нельзя перемонтировать файловую систему, созданную только для чтения, в режим чтения/записи. Неизменяемые носители, например, на CD-ROM, автоматически монтируются только для чтения.
- для перемонтирования устройства, допускающего запись, в режим чтения/записи введите -o remount,rw

Команды перемонтирования не будут выполнены, если какой-либо процесс имеет открытые файлы или каталоги в перемонтируемой файловой системе. Для нахождения открытых файлов используется команда lsof. За более подробной информацией о дополнительных опциях команды lsof обращайтесь к документации man.

fstab

Из руководства к теме 102 "Подготовка к экзамену LPI 101 (тема 102) d: Установка Linux и управление пакетами", вы узнали, как с помощью параметра root= в GRUB и LILO сообщить загрузчику о том, какая файловая система монтируется в качестве корневой. Смонтировав эту файловую систему, процесс установки запускает mount с опцией -a для автоматического монтирования набора файловых систем. Этот набор задается в файле /etc/fstab. В листинге 29 показан файл /etc/fstab для системы Ubuntu, установленной на файловые системы, созданные ранее в данном руководстве.

Листинг 29. Пример использования fstab

```
root@pinguino:~# cat /etc/fstab
# /etc/fstab: sttic file system information.
#
#<file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/hda6       /               reiserfs defaults       0       1
/dev/hda2       /boot           ext3    defaults        0       2
/dev/hda8       /dos            vfat    defaults        0       0
/dev/hda7       /home           xfs     defaults        0       2
/dev/hda1       /media/hda1     ntfs    defaults        0       0
/dev/hda5       none            swap    sw              0       0
/dev/hdc        /media/cdrom0   udf,iso9660 user,noauto     0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto  0       0
```

Строки, начинающиеся символом #, являются комментариями. Остальные строки содержат шесть полей. Поскольку эти поля позиционные, все они должны быть заполнены.

file system

Для вышеупомянутых примеров имя должно быть задано как /dev/hda1.

mount point

Это точка монтирования, рассмотренная в разделе Монтирование файловых систем. Для пространства подкачки это поле имеет значение none. Для файловых систем ext2, ext3 и xfs можно также указывать метку тома, например: LABEL=XFSHOME. Это делает систему более устойчивой при установке и удалении устройств.

type

Определяет тип файловой системы. CD/DVD-диски часто имеют разные файловые системы - ISO9660 или UDF - поэтому вы можете перечислить различные возможности в виде списка, разделенного запятыми. Если вы хотите, чтобы mount автоматически определила тип, используйте auto, как сделано в последней строке для дискеты.

option

Определяет параметры монтирования. Для монтирования со значениями по умолчанию используйте defaults. Несколько полезных опций:

- rw и ro указывают монтирование файловой системы в режиме чтения/записи или только для чтения.
- noauto указывает, что файловая система не должна автоматически монтироваться при загрузке или при выдаче команды mount -a. В нашем примере эта опция применена для съемных устройств.
- user
- определяет, что пользователь, не имеющий прав root, может монтировать или демонтировать данную файловую систему. Это особенно полезно для съемных носителей. Эта опция должна быть задана в /etc/fstab, а не в команде mount.
- exec или noexecопределяют, позволять ли исполнение файлов из данной файловой системы. Для файловых систем, монтируемых пользователем, по умолчанию устанавливается значение noexec, если только после поля user не указано exec.
- noatime отключает запись атрибута времени доступа к файлу. Это может повысить производительность.

dump

Определяет, будет ли команда dump включать данную файловую систему ext2 или ext3 в резервные копии. Значение 0 означает, что dump игнорирует данную файловую систему.

pass

Ненулевые значения pass определяют порядок проверки файловых систем во время загрузки, как описано в теме Проверка файловых систем.

Для монтирования файловых систем, перечисленных в /ect/fstab, достаточно задать либо имя устройства, либо точку монтирования. Оба параметра одновременно задавать не нужно.

За более подробным описанием функций fstab и mount, включая не рассмотренные здесь опции, обращайтесь к документации man.

Размонтирование файловых систем

Все смонтированные файловые системы обычно автоматически размонтируются системой при перезагрузке или выключении. При размонтировании файловой

системы все кэшированные данные файловой системы сохраняются на диск.

Также можно размонтировать файловую систему вручную. В действительности это необходимо делать всякий раз, когда вы удаляете записываемый съемный носитель - дискету, USB-диск или флэш-накопитель. Прежде чем размонтировать файловую систему, следует убедиться в отсутствии работающих процессов, которые имеют открытые файлы в этой файловой системе. Затем используйте команду umount, указав в качестве аргумента либо имя устройства, либо точку монтирования. Несколько примеров успешного и безуспешного размонтирования приведено в листинге 30.

Листинг 30. Размонтирование файловых систем

```
root@pinguino:~# lsof /dos
root@pinguino:~# umount /dos
root@pinguino:~# mount /dos
root@pinguino:~# umount /dev/hda8
root@pinguino:~# umount /boot
umount: /boot: device is busy
umount: /boot: device is busy
root@pinguino:~# lsof /boot
COMMAND  PID USER   FD   TYPE DEVICE   SIZE NODE NAME
klogd   6498 klog    1r   REG    3,2 897419 6052 /boot/System.map-2.6.12-
10-386
```

После размонтирования файловой системы файлы в каталоге, использовавшемся в качестве точки монтирования, снова становятся видимыми.

Пространство подкачки

Вы могли заметить, в описании команды fstab, что пространство подкачки не имеет точки монтирования. В процессе загрузки система обычно активизирует пространство подкачки, указанное в /etc/fstab, если не указана опция noauto. Для управления пространством подкачки в работающей системе, например, для добавления нового раздела подкачки, используются команды swapon и swapoff. Подробнее см. документацию man.

Для просмотра активизированных в данных момент устройств подкачки используйте cat /proc/swaps.

Дисковые квоты

Установка квот позволяет контролировать использование дисков пользователями и группами пользователей. Квоты не дают отдельным пользователям и группам использовать большую часть файловой системы, чем им разрешено, или полностью заполнять эту часть. Квоты устанавливаются и изменяются пользователем root. Чаще всего они используются в многопользовательских системах, реже – в однопользовательских рабочих станциях.

Установка режима квотирования

Для установки режима квотирования необходима поддержка ядра. Как правило, современные ядра версий 2.4 или 2.6 имеют всю необходимую поддержку. В более ранних версиях поддержка могла быть неполной, что требовало от вас сборки собственного ядра. В современных реализациях поддержка квот чаще всего реализуется в виде модулей ядра. Существует три версии поддержки квот; vfsold (версия 1), vfsv0 (версия 2) и xfs (для файловой системы XFS). Данный раздел охватывает вторую версию квотирования для файловых систем, отличных от XFS, и квоты xfs для файловой системы XFS.

Первый шаг для введения квотирования – указание опций usrquota или grpquota в определении файловой системы в /etc/fstab, соответственно тому, что вы хотите ввести: квоты для пользователей, групп или то и другое. Рассмотрим создание обоих типов квот в файловой системе XFS, используемой для каталогов home в нашем примере, а также для файловой системы /boot, чтобы видеть, как это делается в различных файловых системах. Сделайте, как показано в листинге 31.

Листинг 31. Установка режима квотирования в /etc/fstab

```
/dev/hda2     /boot           ext3    defaults,usrquota,grpquota    0   2
/dev/hda7     /home           xfs     defaults,usrquota,grpquota    0   2
```

В файловой системе XFS информация о квотах входит в состав метаданных. В других файловых системах информация о квотах для пользователей хранится в файле aquota.user в корне файловой системы, а для групп пользователей – в файле aquota.group. Для квот первой версии использовались файлы quota.user и quota.group.

Внеся изменения в /etc/fstab и добавив квоты, необходимо перемонтировать файловые системы и, для файловых систем, отличных от XFS, создать файлы квот и разрешить проверку квотирования. Команда quotacheck проверяет квотирование на всех файловых системах и создает необходимые файлы aquota.user и aquota.group, если их не существует. Также она может восстановить поврежденные файлы квот. Более подробно см. руководство man. Команда quotaon включает проверку квот. Пример показан в листинге 32. Следующие опции используются в обеих командах:

-a
: для всех файловых систем в /etc/fstab, для которых разрешено автоматическое монтирование

-u
: для пользовательских квот (установлено по умолчанию)

-g
: для групповых квот

-v
: для подробного вывода

Листинг 32. Создание файлов квот и включение квотирования

```
root@pinguino:~# quotacheck -augv
```

```
quotacheck: Scanning /dev/hda2 [/boot] quotacheck: Cannot stat old user
quota
file: No such file or directory
quotacheck: Cannot stat old group quota file: No such file or directory
quotacheck: Cannot stat old user quota file: No such file or directory
quotacheck: Cannot stat old group quota file: No such file or directory
done
quotacheck: Checked 4 directories and 23 files
quotacheck: Old file not found.
quotacheck: Old file not found.
quotacheck: Skipping /dev/hda7 [/home]
root@pinguino:~# quotaon -ugva
/dev/hda2 [/boot]: group quotas turned on
/dev/hda2 [/boot]: user quotas turned on
```

Проверка квот при загрузке

Команды quotacheck и quotaon обычно входят в состав инициализационных сценариев, поэтому квотирование включается каждый раз при перезагрузке системы. Дополнительная информация содержится в руководстве Quota Mini HOWTO (см. Ресурсы).

Команда quotaoff отключает использование квот, если необходимо.

Установка пределов квот

Как видно, квотирование управляется либо через бинарные файлы в корне файловой системы, либо через метаданные файловой системы. Для установки квоты для отдельного пользователя используется команда edquota. Эта команда извлекает информацию о квотах для данного пользователя из различных файловых систем, для которых включено квотирование, создает временный файл и открывает редактор, позволяющий изменять квоты. Информацию о том, какой именно редактор используется, см. в документации man команды edquota. Для изменения квот необходимо обладать полномочиями root. Полученная информация будет выглядеть примерно так, как в листинге 33.

Листинг 33. Запуск edquota

```
Disk quotas for user ian (uid 1000):
  Filesystem    blocks       soft       hard     inodes     soft     hard
  /dev/hda2          0          0          0          0        0        0
  /dev/hda7       2948          0          0        172        0        0
```

Как видно из примера, edquota показывает текущее использование блоков 1K и inode для каждой файловой системы, где включено квотирование. Также существуют мягкие и жесткие пределы на использование блоков и inode. В данном примере их значения установлены в 0, что означает, что пределы квот не установлены.

Мягкие пределы – это пределы, при достижении которых пользователь получает предупреждения о превышении квоты. Жесткие пределы – это границы, которые

пользователь не может превысить. Можно считать, что ограничения на блоки - это ограничения на объем сохраняемой информации, а ограничения на inode – это ограничения количества файлов и каталогов.

Изменение пределов квот

Чтобы изменить пределы квот, отредактируйте значения во временном файле и сохраните его. Чтобы не применять изменения, закройте файл без сохранения. Предположим, вы хотите установить для меня ограничения в файловой системе /home: по объему – 10 Мбайт, по количеству – 1000 файлов. Добавляя 10% запаса на жесткие пределы, устанавливаем значения, как показано в листинге 34.

Листинг 34. Установка пределов

```
Disk quotas for user ian (uid 1000):
  Filesystem    blocks       soft       hard     inodes     soft     hard
  /dev/hda2          0          0          0          0        0        0
  /dev/hda7       2948      10240      11264        172     1000     1100
```

Сохраните файл, чтобы применить изменения. В данном примере для пользователя ian в файловой системе /boot квоты не были изменены, поскольку пользователь ian не имеет прав для записи в эту файловую систему. Также обратите внимание, что любые изменения, внесенные в данные об используемых блоках или inodes, будут проигнорированы.

Копирование квот

Теперь предположим, что вы создаете идентификаторы для группы пользователей – слушателей учебного курса. Допустим, у вас есть пользователи gretchen, tom и greg, и вы хотите назначить им такие же квоты, как у ian. Для этого применяется опция -p команды edquota, которая использует значения квот пользователя ian в качестве прототипа для квот других пользователей, как показано в листинге 35.

Листинг 35. Установка квот по прототипу

```
root@pinguino:~# edquota -p ian gretchen tom greg
```

Квоты для групп пользователей

Команду edquota также можно использовать для ограничения выделения дискового пространства на основании принадлежности файлов группам. Пусть, например, три упомянутых выше слушателя объединены в основную группу xml-101. Чтобы задать пределы на суммарный объем, используемый всеми членами группы, на уровне 25 Мбайт и 2500 файлов, используйте команду edquota -q xml-101 и установите значения, как показано в листинге 36.

Листинг 36. Установка квот для группы пользователей

```
Disk quotas for group xml-101 (gid 1001):
```

```
  Filesystem    blocks       soft       hard     inodes     soft     hard
  /dev/hda2          0          0          0          0        0        0
  /dev/hda7         28      25600      28160         10     2500     2750
```

Льготный период

Пользователи могут превышать мягкие пределы квот в течение "льготного периода", который по умолчанию составляет 7 дней. После истечения этого периода мягкие пределы становятся жесткими. Льготные периоды устанавливаются с помощью опции -y команды edquota. Перед вами вновь окажется редактор с данными, аналогичными представленным в листинге 37. Как и раньше, сохраните изменения, чтобы обновить значения. Убедитесь, что пользователям предоставлено достаточно времени для получения предупреждений по электронной почте и удаления некоторых файлов.

Листинг 37. Установка льготных периодов

```
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem             Block grace period     Inode grace period
  /dev/hda2                     7days                  7days
  /dev/hda7                     7days                  7days
```

Проверка квот

Команда quota без указания опций выводит квоты для вызвавшего ее пользователя во всех файловых системах, где такие квоты установлены. Опция -v выводит информацию для всех файловых систем, в которых включено квотирование. Пользователь root может также добавить имя пользователя к команде, чтобы просмотреть ограничения для конкретного пользователя. Эти команды представлены в листинге 38.

Листинг 38. Просмотр квот

```
root@pinguino:~# quota
Disk quotas for user root (uid 0): none
root@pinguino:~# quota -v
Disk quotas for user root (uid 0):
     Filesystem  blocks   quota   limit   grace   files   quota   limit
grace
      /dev/hda2   19173       0       0              26       0       0
      /dev/hda7      16       0       0               5       0       0
root@pinguino:~# quota -v ian
Disk quotas for user ian (uid 1000):
     Filesystem  blocks   quota   limit   grace   files   quota   limit
grace
      /dev/hda2       0       0       0               0       0       0
      /dev/hda7    2948   10240   11264             172    1000    1100
```

Наряду с текущими уровнями использования выводятся жесткие и мягкие пределы. В листинге 39 показано, что будет, если превысить границы мягкого предела, и что

произойдет, если попытаться превысить жесткий предел. В данном примере мы создаем файл размером примерно 4 Мбайт, а затем копируем его. Вместе с первоначальным уровнем использования около 3 Мбайт этого достаточно для превышения мягкого предела. Обратите внимание, что рядом с мягким пределом выводится звездочка, показывающая, что пользователь превысил квоту. Также заметьте, что в столбце grace period теперь показано, сколько времени есть у пользователя, чтобы исправить положение.

Листинг 39. Превышение квот

```
ian@pinguino:~$ dd if=/dev/zero of=big1 bs=512 count=8000
8000+0 records in
8000+0 records out
4096000 bytes transferred in 0.019915 seconds (205674545 bytes/sec)
ian@pinguino:~$ cp big1 big2
ian@pinguino:~$ quota
Disk quotas for user ian (uid 1000):
     Filesystem  blocks   quota   limit   grace   files   quota   limit
grace
      /dev/hda7   10948*  10240   11264   7days     174    1000    1100
ian@pinguino:~$ cp big1 big3
cp: writing `big3': Disk quota exceeded
```

Создание отчета о квотах

Проверять квоты для каждого пользователя последовательно не очень удобно, поэтому для создания отчета о квотах используется команда repquota. В листинге 40 показано, как просмотреть квоты для всех пользователей и групп каталога /home.

Листинг 40. Превышение квот

```
root@pinguino:~# repquota -ug /home
*** Report for user quotas on device /dev/hda7
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --      16       0       0              5     0     0
ian       +-   11204   10240   11264  6days     175  1000  1100
tom       --       8   10240   11264              3  1000  1100
gretchen  --       8   10240   11264              3  1000  1100
greg      --      12   10240   11264              4  1000  1100


*** Report for group quotas on device /dev/hda7
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
Group           used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --      16       0       0              5     0     0
ian       --   11204       0       0            175     0     0
xml-101   --      28   25600   28160             10  2500  2750
```

Обратите внимание на знак плюс в листинге для пользователя ian. Он показывает,

что ian превысил квоту.

Как и в других командах, относящихся к квотам, опция -a создает отчет по всем файловым системам, где включено квотирование. Опция -v формирует более подробный вывод. Опция -n выводит список номеров пользователей без определения их имен. Это может повысить производительность для больших отчетов, но результат хуже читается человеком.

Предупреждение пользователей

Команда warnquota используется для отправки предупреждений по электронной почте пользователям, превысившим квоты. Если квоту превысила группа, сообщения по электронной почте отправляются пользователям, указанным в /etc/quotagrpadmins. Обычно warnquota запускается периодически как задание cron. Более подробно о cron и warnquota см. документацию man.

Полномочия доступа к файлам и управление доступом

Пользователи и группы

К этому времени вам уже должно быть известно, что Linux является многопользовательской системой, и каждый пользователь принадлежит к одной основной группе и, возможно, дополнительным группам. Кроме того, войдя в систему в качестве одного пользователя, с помощью команд su или sudo -s можно стать другим пользователем. Понятие владения файлами в Linux тесно связано с идентификаторами пользователя и группами, поэтому давайте повторим основные сведения о пользователях и группах.

Кто я такой?

Если вы не стали другим пользователем, ваш идентификатор пользователя остался таким же, каким вы его ввели при входе в систему. Если вы становитесь другим пользователем, в приглашении командной строки может содержаться ваш идентификатор пользователя, как в большинстве примеров в этом руководстве. Если в приглашении командной строки не содержится идентификатора текущего пользователя, вы можете узнать его с помощью команды whoami. В листинге 41 показано несколько примеров, в которых настройки приглашения командной строки (из переменной среды PS1) отличаются от остальных примеров в этом руководстве.

Листинг 41. Определение идентификатора текущего пользователя

```
/home/ian$ whoami
tom
/home/ian$ exit
exit
$ whoami
ian
```

В какие группы я вхожу?

Подобным же образом с помощью команды groups можно узнать, в какие группы вы входите. С помощью команды id можно получить информацию и о пользователях, и о группах. Добавив в качестве параметра к команде groups или id идентификатор пользователя, можно просмотреть информацию об этом пользователе, а не о текущем. Несколько примеров приведено в листинге 42.

Листинг 42. Определение членства в группах

```
$ su tom
Password:
/home/ian$ groups
xml-101
/home/ian$ id
uid=1001(tom) gid=1001(xml-101) groups=1001(xml-101)
/home/ian$ exit
$ groups
ian adm dialout cdrom floppy audio dip video plugdev lpadmin scanner
admin xml-101
$ id
uid=1000(ian) gid=1000(ian)
groups=4(adm),20(dialout),24(cdrom),25(floppy),
29(audio),30(dip),44(video),46(plugdev),104(lpadmin),105(scanner),106(adm
in),
1000(ian),1001(xml-101)
$ groups tom
tom : xml-101
```

Владение файлом и полномочия доступа к нему

Точно так же как у любого пользователя есть свой идентификатор, а сам он является членом основной группы, у каждого файла в системе Linux есть связанные с ним один владелец и одна группа.

Обычные файлы

Для того чтобы вывести информацию о владельцах и группах файлов, выполните команду ls -l

Листинг 43. Определение владельца файла

```
gretchen@pinguino:~$ ls -l /bin/bash .bashrc
-rw-r--r--  1 gretchen xml-101   2227 Dec 20 10:06 .bashrc
-rwxr-xr-x  1 root     root    645140 Oct  5 08:16 /bin/bash
```

В этом примере файл .bashrc пользователя gretchen принадлежит ей и входит в группу xml-101, которая является её основной группой. Точно так же, владельцем /bin/bash является пользователь root, а его основной группой – группа root. Имена пользователей и названия групп берутся из различных пространств имен, поэтому название группы может быть таким же, как имя пользователя. На самом деле по умолчанию во многих дистрибутивах для каждого нового пользователя создаётся

группа с таким же названием.

Для каждого объекта файловой системы в модели полномочий Linux есть три типа полномочий: полномочия чтения (r), записи (w) и выполнения (x). В полномочия записи входят также возможности удаления и изменения объекта. Кроме того, эти полномочия указываются отдельно для владельца файла, членов группы файла и для всех остальных.

Вернемся к первой колонке листинга 43. Обратите внимание, что она содержит строку из десяти символов. Первый символ описывает тип объекта (в этом примере - обозначает обычный файл), а оставшиеся девять символов представляют три группы по три символа в каждой. Первая группа обозначает полномочия чтения, записи и выполнения для владельца файла. Знак "-" обозначает, что соответствующего полномочия дано не было. Поэтому пользователь gretchen может читать файл .bashrc, проводить в него запись, но не может выполнять его, в то время как пользователь root может читать файл /bin/bash, проводить в него запись и выполнять его. Вторая группа обозначает полномочия чтения, записи и выполнения для группы файла. Члены группы xml-101 могут считывать файл .bashrc пользователя gretchen, но не могут производить в него запись, так же, как и все остальные. Подобным же образом, члены группы root и все остальные пользователи могут считывать и выполнять файл /bin/bash.

Каталоги

Для каталогов используются те же флаги полномочий, что и для обычных файлов, однако интерпретируются они иначе. Наличие у пользователя полномочий чтения каталога позволяет ему просматривать содержимое каталога. Пользователь, имеющий полномочия записи, может создавать и удалять файлы в этом каталоге. Полномочия выполнения позволяют пользователю входить в этот каталог и просматривать все подкаталоги. Без полномочий выполнения объекты файловой системы, находящиеся в этом каталоге, недоступны. Без полномочий чтения объекты файловой системы, находящиеся в каталоге, нельзя просматривать, однако доступ к ним можно получить, если вы знаете полный путь к этому объекту на диске. В листинге 44 приведен несколько искусственный пример, иллюстрирующий этот момент.

Листинг 44. Полномочия и каталоги

```
ian@pinguino:~$ ls -l /home
total 8
drwxr-x---   2 greg     xml-101   60 2005-12-20 11:37 greg
drwx------  13 gretchen xml-101 4096 2005-12-21 12:22 gretchen
drwxr-xr-x  15 ian      ian     4096 2005-12-21 10:25 ian
d-wx--x--x   2 tom      xml-101   75 2005-12-21 11:05 tom
ian@pinguino:~$ ls -a ~greg
.  ..  .bash_history  .bash_profile  .bashrc
ian@pinguino:~$ ls -a ~gretchen
ls: /home/gretchen: Permission denied
ian@pinguino:~$ ls -a ~tom
ls: /home/tom: Permission denied
ian@pinguino:~$ head -n 3 ~tom/.bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-
doc)
```

```
# for examples
```

Первый символ длинного листинга описывает тип объекта (d для каталога). У каталога home пользователя greg установлены полномочия на чтение и выполнение для членов группы xml-101, поэтому пользователь ian может получить список файлов, находящихся в этом каталоге. У каталога home пользователя Gretchen нет ни полномочий на чтение, ни полномочий на запись, поэтому пользователь ian не может получить доступ к нему. У каталога home пользователя tom установлены только полномочия на выполнение, но нет полномочий на чтение, поэтому пользователь ian не может просмотреть содержимое этого каталога, но может получить доступ к располагающимся в нем объектам, если он точно знает, что они существуют.

Другие объекты файловой системы

В длинном листинге могут содержаться объекты файловой системы, отличные от файлов и каталогов, что можно увидеть по первому символу листинга. Мы будет рассматривать эти объекты ниже в этом разделе, отметим на данный момент лишь возможные типы объектов.

Таблица 3. Типы объектов файловой системы

Код	Тип объекта
-	Обычный файл
d	Каталог
l	Символическая ссылка
c	Специальное символическое устройство
b	Специальное блочное устройство
p	Буфер FIFO
s	Сокет

Изменение полномочий

Добавление полномочий

Предположим, вы создали сценарий командной оболочки "Hello world". При создании сценария он обычно не является исполняемым. Чтобы добавить полномочия на выполнение, используйте команду chmod с параметром +x, как показано в листинге 45.

Листинг 45. Создание исполняемого сценария командной оболочки

```
ian@pinguino:~$ echo 'echo "Hello world!"'>hello.sh
ian@pinguino:~$ ls -l hello.sh
-rw-r--r--  1 ian ian 20 2005-12-22 12:57 hello.sh
ian@pinguino:~$ ./hello.sh
-bash: ./hello.sh: Permission denied
ian@pinguino:~$ chmod +x hello.sh
```

```
ian@pinguino:~$ ./hello.sh
Hello world!
ian@pinguino:~$ ls -l hello.sh
-rwxr-xr-x  1 ian ian 20 2005-12-22 12:57 hello.sh
```

Подобным же образом можно использовать r для установки полномочий на чтение и w для установки полномочий на запись. На самом деле можно использовать вместе любую комбинацию r, w и x. Например, команда chmod +rwx установит для файла все полномочия на чтение, запись и выполнение. Использование chmod в таком виде добавляет не установленные на данный момент полномочия.

Выборочное изменение

Вы могли заметить, что в приведенном выше примере права на выполнение устанавливаются для владельца, группы и других. Чтобы действовать более избирательно, необходимо использовать префикс для выражения режима: u для установки полномочий для пользователей, g для установки полномочий для групп и o для установки полномочий для всех остальных. Указание a определяет полномочия для всех пользователей, что равнозначно отсутствию префикса. В листинге 46 показано, как добавить пользователю и группе полномочия на запись и выполнение другой копии сценария командной оболочки.

Листинг 46. Выборочное добавление полномочий

```
ian@pinguino:~$ echo 'echo "Hello world!"'>hello2.sh
ian@pinguino:~$ chmod ug+xw hello2.sh
ian@pinguino:~$ ls -l hello2.sh
-rwxrwxr--  1 ian ian 20 2005-12-22 13:17 hello2.sh
```

Снятие полномочий

Иногда вам нужно не добавить полномочия, а снять их. Просто измените + на a -, чтобы удалить все указанные и установленные полномочия. В листинге 47 показано, как снять все полномочия для остальных пользователей с двух сценариев командной оболочки.

Листинг 47. Снятие полномочий

```
ian@pinguino:~$ ls -l hello*
-rwxrwxr--  1 ian ian 20 2005-12-22 13:17 hello2.sh
-rwxr-xr-x  1 ian ian 20 2005-12-22 12:57 hello.sh
ian@pinguino:~$ chmod o-xrw hello*
ian@pinguino:~$ ls -l hello*
-rwxrwx---  1 ian ian 20 2005-12-22 13:17 hello2.sh
-rwxr-x---  1 ian ian 20 2005-12-22 12:57 hello.sh
```

Следует отметить, что можно за один раз изменить полномочия более чем у одного файла. Как и с некоторыми другими командами, с которыми вы встречались в руководстве для экзамена 103, вы даже можете использовать параметр -R (или --recursive) для рекурсивного обхода каталогов и папок.

Установка полномочий

Теперь, когда вы можете добавлять и удалять полномочия, вы можете задать вопрос – как установить только определенный набор полномочий. Это делается с помощью знака = вместо + и -. Для того чтобы установить полномочия для приведенных выше сценариев так, чтобы другие пользователи не имели прав доступа, можно вместо команд на удаление полномочий использовать команду chmod o= hello*.

Если вы желаете установить различные полномочия для пользователя, группы и остальных пользователей, вы можете разделять различные выражения запятыми; например, ug=rwx,o=rx, также вы можете использовать цифровой способ указания полномочий, описываемый ниже.

Установка полномочий в восьмеричном формате

До настоящего момента для указания полномочий вы использовали символы (ugoa и rxw). В каждой группе существует три возможных типа полномочий. Также можно указывать полномочия, используя вместо символов числа в восьмеричном формате. Для установки полномочий, таким образом, может потребоваться до четырёх восьмеричных цифр. Рассматривать первую цифру мы будем при обсуждении атрибутов. Вторая цифра определяет полномочия для пользователя, третья – полномочия для группы и четвертая – полномочия для остальных пользователей. Каждая из этих трех цифр получается путем сложения желаемых полномочий: на чтение (4), на запись (2) и на исполнение (1). В примере для hello.sh, приведенном в листинге 45, сценарий был создан с полномочиями -rw-r--r--, что соответствует восьмеричному 644. Установка прав на выполнение для всех изменит режим на 755.

Использование полномочий в цифровом виде очень удобно в случаях, когда вы хотите установить все полномочия сразу, не указывая одинаковые полномочия для каждой группы. Используйте таблицу 4 в качестве удобного справочника для установки полномочий в восьмеричном формате.

Таблица 4. Установка полномочий в числовом формате

Символический	**Восьмеричный**
rwx	7
rw-	6
r-x	5
r--	4
-wx	3
-w-	2
--x	1
---	0

Режимы доступа

Когда вы входите в систему, запускается новый процесс командной оболочки с вашим идентификатором пользователя и идентификатором группы. Эти

идентификаторы определяют полномочия на доступ ко всем файлам в системе. Обычно это означает, что вы не можете открывать файлы, принадлежащие другим пользователям, и системные файлы. На самом деле мы как пользователи полностью полагаемся на другие программы, выполняющие действия от нашего имени. Поскольку программы, которые вы запускаете, наследуют ваш идентификатор пользователя, они не могут получить доступа к объектам файловой системы, доступа к которым не имеете вы.

В качестве важного примера можно привести файл /etc/passwd, который не может быть изменен обычными пользователями напрямую, так как полномочия на запись есть только у пользователя root:. Однако обычным пользователям необходима возможность изменения файла /etc/passwd каким-либо образом всякий раз, когда им нужно изменить свой пароль. Итак, если пользователь не может изменить этот файл, как это можно сделать?

suid and sgid

В модели полномочий Linux есть два специальных режима доступа, называемых suid (установить идентификатор пользователя) и sgid (установить идентификатор группы). Если у исполняемой программы установлен режим доступа suid, она будет запущена так, как если бы это сделал владелец файла, а не пользователь, который фактически её запустил. Подобно этому, при установленном режиме доступа sgid программа будет работать так, как если бы её запустил пользователь, входящий в группу, которой принадлежит файл, а не в группу, в которой фактически состоит пользователь. Эти режимы можно установить как по отдельности, так и вместе.

В листинге 48 показан исполняемый файл passwd, владельцем которого является пользователь root:

Листинг 48. Режим доступа suid файла /usr/bin/passwd

```
ian@pinguino:~$ ls -l /usr/bin/passwd
-rwsr-xr-x  1 root root 25648 2005-10-11 12:14 /usr/bin/passwd
```

Заметьте, что вместо x в тройке символов полномочий пользователя стоит s. Это обозначает, что для этой конкретной программы установлен флаг выполнения и режим доступа suid. При запуске программы passwd она будет выполняться так, как будто бы её запустил пользователь root, со всеми правами доступа привилегированного пользователя, а не того пользователя, который её действительно запустил. Поскольку программа passwd работает с уровнем доступа пользователя root, она может изменить файл /etc/passwd.

Флаги suid и sgid занимают в длинном листинге каталога то же место, что и флаг x. Если файл исполняемый, установленные флаги suid и sgid будут отображаться как маленькая s, в противном случае они будут выводиться как большая S.

Несмотря на то, что флаги suid и sgid очень удобны, и даже необходимы во многих ситуациях, неправильное использование этих режимов доступа может привести к появлению брешей в системе обеспечения безопасности. У вас должно быть как можно меньше программ, работающих в режиме доступа suid. Команда passwd является одной из немногих, которая должна работать в режиме suid.

Установка suid и sgid

Флаги suid и sgid устанавливаются с использованием символа s; например, u+s устанавливает режим доступа suid, а g-s снимает режим доступа sgid. В восьмеричном формате режиму suid соответствует значение 4 в первой цифре (старший разряд), а режиму sgid соответствует значение 2.

Каталоги и sgid

Если для каталога установлен режим sgid, все файлы и каталоги, созданные в нем, будут наследовать идентификатор группы этого каталога. В частности, это полезно для деревьев каталогов, используемых группой людей, работающих над одним проектом. В листинге 49 показано, как пользователь greg может настроить каталог, с которым могут работать все пользователи группы xml-101, а также пример того, как пользователь gretchen может создать файл в каталоге.

Листинг 49. Режим доступа sgid и каталоги

```
greg@pinguino:~$ mkdir xml101
greg@pinguino:~$ chmod g+ws xml101
greg@pinguino:~$ ls -ld xml101
drwxrwsr-x  2 greg xml-101 6 Dec 25 22:01 xml101
greg@pinguino:~$ su - gretchen
Password:
gretchen@pinguino:~$ touch ~greg/xml101/gretchen.txt
gretchen@pinguino:~$ ls -l ~greg/xml101/gretchen.txt
-rw-r--r--  1 gretchen xml-101 0 Dec 25 22:02
/home/greg/xml101/gretchen.txt
```

Теперь любой член группы xml-101 может создавать файлы в папке xml101 пользователя greg. Как показано в листинге 50, другие члены группы не могут изменять файл gretchen.txt, но у них есть полномочия на запись в каталог и потому они могут удалить файл.

Листинг 50. Режим доступа sgid и владение файлом

```
gretchen@pinguino:~$ su - tom
Password:
~$ cat something >> ~greg/xml101/gretchen.txt
-su: /home/greg/xml101/gretchen.txt: Permission denied
~$ rm ~greg/xml101/gretchen.txt
rm: remove write-protected regular empty file
`/home/greg/xml101/gretchen.txt'? y
~$ ls -l ~greg/xml101
total 0
```

Бит закрепления в памяти

Только что вы увидели, как любой пользователь, имеющий полномочия на запись в

каталог, может удалить находящиеся в ней файлы. Такая ситуация может быть приемлемой для проекта рабочей группы, но нежелательна для файлового пространства, находящегося в общем доступе, например, каталога /tmp. К счастью, решение существует.

Оставшийся флаг режима доступа называется битом закрепления в памяти (sticky bit). Он представляется символом t и числом 1 в восьмеричной цифре старшего разряда. Он отображается в длинном листинге каталога на месте флага выполнения для остальных пользователей (последний символ), значение регистра аналогично значению регистра для suid и sgid. Если для каталога установлен этот флаг, он допускает удаление файла или ссылок только владельцем или суперпользователем (root). В листинге 51 показано, как пользователь greg может установить бит закрепления в памяти на свой каталог xml101, а также показано, что этот флаг установлен для каталога /tmp.

Листинг 51. Каталоги с закреплением в памяти

```
greg@pinguino:~$ chmod +t xml101
greg@pinguino:~$ ls -l xml101
total 0
greg@pinguino:~$ ls -ld xml101
drwxrwsr-t  2 greg xml-101 6 Dec 26 09:41 xml101
greg@pinguino:~$ ls -ld xml101 /tmp
drwxrwxrwt  13 root root    520 Dec 26 10:03 /tmp
drwxrwsr-t   2 greg xml-101   6 Dec 26 09:41 xml101
```

Исторически, в системах UNIX® бит закрепления в памяти использовался на файлах и обозначал, что исполняемые файлы необходимо хранить в области свопинга, чтобы исключить их повторную загрузку. Современные ядра системы Linux игнорируют установку бита закрепления в памяти для файлов.

Краткое резюме по режимам доступа

В таблице 5 представлено краткое описание символического и восьмеричного представления трех обсуждаемых здесь режимов доступа.

Таблица 5. Режимы доступа

Режим доступа	Символическое	Восьмеричное
suid	s c u	4000
sgid	s c g	2000
sticky	t	1000

Сочетая эти данные с приведенной ранее информацией о полномочиях, вы можете увидеть, что полномочия и режимы доступа, соответствующие каталоги xml101 пользователя greg, записываемые как drwxrwsr-t, в полном восьмеричном представлении выглядят как 1775.

Файлы только для чтения

Режимы доступа и полномочия предоставляют широкие средства управления тем, кто и что может делать с файлами и каталогами. Как бы то ни было, они не предотвращают неумышленного удаления файлов пользователем root. В различных файловых системах существуют дополнительные атрибуты, которые предоставляют дополнительные возможности. Одним из таких атрибутов является атрибут только для чтения. Если этот атрибут установлен, даже пользователь root не сможет удалить файл, пока атрибут не будет снят.

Чтобы просмотреть, установлен ли на файле или каталоги флаг "только для чтения" (или какой-либо иной атрибут), необходимо использовать команду lsattr. Чтобы сделать файл доступным только для чтения, необходимо подать команду chattr с параметром -i.

В листинге 52 показано, что пользователь root может создать файл только для чтения, но не может удалить его до тех пор, пока не снят флаг "только для чтения".

Листинг 52. Файлы только для чтения

```
root@pinguino:~# touch keep.me
root@pinguino:~# chattr +i keep.me
root@pinguino:~# lsattr keep.me
----i------------ keep.me
root@pinguino:~# rm -f keep.me
rm: cannot remove `keep.me': Operation not permitted
root@pinguino:~# chattr -i keep.me
root@pinguino:~# rm -f keep.me
```

Для того чтобы изменить флаг "только для чтения", необходимы полномочия root, или, по меньшей мере, привилегия CAP_LINUX_IMMUTABLE. Перевод файлов в режим "только для чтения" часто выполняется в ходе мероприятий по обеспечению безопасности и обнаружению вторжений. Дополнительную информацию можно найти в документации man по ключевому слову capabilities (man capabilities).

umask

При создании нового файла ему присваиваются определенные полномочия. Часто устанавливается режим 0666, что открывает возможность чтения и записи в этот файл для всех пользователей. В любом случае на процесс определения полномочий при создании файла влияет значение umask, которое определяет, какие полномочия пользователь не желает автоматически присваивать вновь создаваемым файлам и каталогам. Система использует значение umask для ограничения изначально установленных полномочий. Просмотреть значение параметра umask можно с помощью команды umask, пример использования которой показан в листинге 53.

Листинг 53. Вывод umask в восьмеричном формате

```
ian@pinguino:~$ umask
0022
```

Необходимо помнить, что umask указывает только полномочия, которые не должны быть предоставлены. По умолчанию в системах Linux параметру umask обычно присваивается значение 0022, что снимает с групп и других пользователей полномочия на запись во вновь создаваемые файлы. Для того, чтобы вывести значение параметра umask в символическом виде, отображая, какие полномочия разрешены, используйте параметр -S.

Использовать команду umask можно не только для просмотра, но и для установки значения параметра umask. Итак, если вы желаете хранить ваши файлы в конфиденциальном порядке и полностью отключить доступ группы и всех остальных пользователей, вам необходимо использовать значение umask, равное 0077. Также можно установить umask в символическом виде, umask u=rwx,g=,o=, как это показано в листинге 54.

Листинг 54. Установка umask

```
ian@pinguino:~$ umask
0022
ian@pinguino:~$ umask -S
u=rwx,g=rx,o=rx
ian@pinguino:~$ umask u=rwx,g=,o=
ian@pinguino:~$ umask
0077
ian@pinguino:~$ touch newfile
ian@pinguino:~$ ls -l newfile
-rw-------  1 ian ian 0 2005-12-26 12:49 newfile
```

В следующем разделе будет показано, как можно изменить владельца или группу существующего объекта файловой системы.

Установка владельца и группы файла

Из предыдущего раздела вы узнали, что у каждого объекта файловой системы имеется владелец и группа. В этом разделе вы узнаете, как изменить владельца и группу существующего файла, а также установить группу, назначаемую новым файлам по умолчанию.

Группа файла

Для того чтобы изменить группу файла, необходимо использовать команду chgrp, в качестве параметров для которой необходимо указать название группы и название одного или нескольких файлов. Если пожелаете, вы можете использовать номер группы. Если группу файла изменяет обычный пользователь, он должен быть членом назначаемой группы. Пользователь root может назначить файлу любую группу. В листинге 55 показан соответствующий пример.

Листинг 55. Изменение группы-владельца

```
ian@pinguino:~$ touch file1 file2
ian@pinguino:~$ ls -l file*
-rw-r--r--  1 ian ian 0 2005-12-26 14:09 file1
-rw-r--r--  1 ian ian 0 2005-12-26 14:09 file2
ian@pinguino:~$ chgrp xml-101 file1
ian@pinguino:~$ chgrp  1001 file2
ian@pinguino:~$ ls -l file*
-rw-r--r--  1 ian xml-101 0 2005-12-26 14:09 file1
-rw-r--r--  1 ian xml-101 0 2005-12-26 14:09 file2
```

Как и у многих команд, описываемых в этом руководстве, у команды chgrpp есть параметр -R, который позволяет рекурсивно применять изменения ко всем выбранным файлам и подкаталогам.

Группа, назначаемая по умолчанию

В предыдущем разделе вы узнали, как установка режима sgid для каталога может привести к тому, что файлы, создаваемые в этом каталоге, будут принадлежать группе этого каталога, а не той группе, в которую входит пользователь, создающий файл.

Также для временного изменения вашей основной группы на другую, членом которой вы являетесь, вы можете воспользоваться командой newgrp. Будет создана новая командная оболочка, при выходе из которой будет восстановлена ваша прежняя основная группа, что продемонстрировано в листинге 56.

Листинг 56. Использование команды newgrp для временного изменения группы по умолчанию

```
ian@pinguino:~$ newgrp xml-101
ian@pinguino:~$ groups
xml-101 adm dialout cdrom floppy audio dip video plugdev lpadmin scanner
admin ian
ian@pinguino:~$ touch file3
ian@pinguino:~$ ls -l file3
-rw-r--r--  1 ian xml-101 0 2005-12-26 14:34 file3
ian@pinguino:~$ exit
ian@pinguino:~$ groups
ian adm dialout cdrom floppy audio dip video plugdev lpadmin scanner
admin xml-101
```

Владелец файла

Пользователь root может изменить владельца файла с помощью команды chown. В простейшей форме синтаксис этой команды схож с синтаксисом команды chgrp, за исключением того, что вместо названия или идентификатора группы используется имя или идентификатор пользователя. Одновременно можно изменить группу файла, добавив справа от имени или идентификатора пользователя двоеточие и название или идентификатор группы. В случае если установлено только двоеточие, будет использоваться группа пользователя по умолчанию. И, конечно же, параметр -R приведет к рекурсивному внесению изменений. В листинге 57 показан

соответствующий пример.

Листинг 57. Использование команды chown для изменения владельца файла

```
root@pinguino:~# ls -l ~ian/file4
-rw-r--r--  1 ian ian 0 2005-12-26 14:44 /home/ian/file4
root@pinguino:~# chown greg ~ian/file4
root@pinguino:~# ls -l ~ian/file4
-rw-r--r--  1 greg ian 0 2005-12-26 14:44 /home/ian/file4
root@pinguino:~# chown tom: ~ian/file4
root@pinguino:~# ls -l ~ian/file4
-rw-r--r--  1 tom xml-101 0 2005-12-26 14:44 /home/ian/file4
```

В существовавшей ранее форме указания пользователя и группы вместо двоеточия использовалась точка. Использовать такой способ указания не рекомендуется, так как в случае, если в имени пользователя содержится точка, возникнет ошибка.

Жесткие и символические ссылки

Жесткие ссылки

В руководстве для раздела 103, "Подготовка к экзамену LPI 101 (раздел 103): команды GNU и UNIX" вы узнали, что файлы и каталоги хранятся в наборе блоков, а информация о файлах и каталогах хранится в узлах inode.

Жесткими ссылками называются указатели на inode. Так, фактически наименование файла является ссылкой на узел inode, содержащий информацию о файле. Как вы уже знаете, с помощью команды ls с параметром -i можно отобразить номера inode для записей файлов и каталогов.

Вы можете использовать команду ln для создания дополнительных жестких ссылок на существующие файлы (но не на каталоги, даже несмотря на то, что в системе . и .. определены как жесткие ссылки). Если на один узел inode указывает несколько ссылок, этот узел будет удален только тогда, когда количество ссылок на него станет равным нулю.

В листинге 58 показано, как создать файл и жесткую ссылку на него. Также там показано, что даже при удалении исходного наименования файла вторая жесткая ссылка предотвращает стирание при этом узла inode.

Листинг 58. Жесткие ссылки

```
ian@pinguino:~$ echo testing > file1
ian@pinguino:~$ ls -l file*
-rw-r--r--  1 ian ian 8 2005-12-26 15:35 file1
ian@pinguino:~$ ln file1 file2
ian@pinguino:~$ ls -l file*
-rw-r--r--  2 ian ian 8 2005-12-26 15:35 file1
-rw-r--r--  2 ian ian 8 2005-12-26 15:35 file2
ian@pinguino:~$ rm file1
ian@pinguino:~$ ls -l file*
```

```
-rw-r--r--  1 ian ian 8 2005-12-26 15:35 file2
ian@pinguino:~$ cat file2
testing
```

Жесткие ссылки могут существовать только в рамках определенной файловой системы. Они не могут связывать несколько файловых систем, так как ссылка на файл происходит по номеру inode, который уникален только в рамках файловой системы.

Поиск жестких ссылок

Если вам необходимо узнать, какие файлы ссылаются на определенный узел inode, вы можете использовать команду find и параметр -samefile с указанием имени файла или параметр -inum с указанием номера inode, как показано в листинге 59.

Листинг 59. Поиск жестких ссылок

```
ian@pinguino:~$ ln file2 file3
ian@pinguino:~$ ls -il file2
172 -rw-r--r--  2 ian ian 8 2005-12-26 15:35 file2
ian@pinguino:~$ find . -samefile file2
./file2
./file3
ian@pinguino:~$ find . -inum 172
./file2
./file3
```

Символические ссылки

Другой формой ссылок, используемых в файловой системе Linux, является символические ссылки (чаще называемых просто symlink). В этом случае ссылка указывает на наименование другого объекта, а не на его узел inode. Символические ссылки могут указывать на каталоги и на файлы, расположенные в других файловых системах. Они часто используются для присвоения альтернативных имен системным командам. С помощью длинного листинга каталога можно увидеть объекты, являющиеся символическими ссылками; они обозначены маленькой буквой l в первом символе, как показано в листинге 60.

Листинг 60. Примеры символических ссылок

```
ian@pinguino:~$ ls -l /sbin/mkfs.*
-rwxr-xr-x  1 root root  14160 2005-09-20 12:43 /sbin/mkfs.cramfs
-rwxr-xr-x  3 root root  31224 2005-08-23 09:25 /sbin/mkfs.ext2
-rwxr-xr-x  3 root root  31224 2005-08-23 09:25 /sbin/mkfs.ext3
-rwxr-xr-x  2 root root  55264 2005-06-24 07:48 /sbin/mkfs.jfs
-rwxr-xr-x  1 root root  13864 2005-09-20 12:43 /sbin/mkfs.minix
lrwxrwxrwx  1 root root      7 2005-12-14 07:40 /sbin/mkfs.msdos ->
mkdosfs
-rwxr-xr-x  2 root root 241804 2005-05-11 09:40 /sbin/mkfs.reiser4
-rwxr-xr-x  2 root root 151020 2004-11-25 21:09 /sbin/mkfs.reiserfs
lrwxrwxrwx  1 root root      7 2005-12-14 07:40 /sbin/mkfs.vfat ->
mkdosfs
-rwxr-xr-x  1 root root 303788 2005-04-14 01:27 /sbin/mkfs.xfs
```

В дополнение к типу l вы можете видеть справа стрелку ->, за которой следует имя файла, на которое указывает ссылка. Например, команда mkfs.vfat является символической ссылкой на команду mkdosfs. Вы найдете множество подобных ссылок в /sbin и других системных каталогах. Ещё одной отличительной особенностью символических ссылок является размер файла, равный длине имени файла, на который указывает эта ссылка.

Вы можете создать символическую ссылку с помощью команды ln с параметром -s, как показано в листинге 61.

Листинг 61. Создание символических ссылок

```
ian@pinguino:~$ touch file5
ian@pinguino:~$ ln -s file5 file6
ian@pinguino:~$ ln -s file5 file7
ian@pinguino:~$ ls -l file*
-rw-r--r--  2 ian ian 8 2005-12-26 15:35 file2
-rw-r--r--  2 ian ian 8 2005-12-26 15:35 file3
-rw-r--r--  1 ian ian 0 2005-12-26 17:40 file5
lrwxrwxrwx  1 ian ian 5 2005-12-26 17:40 file6 -> file5
lrwxrwxrwx  1 ian ian 5 2005-12-26 17:40 file7 -> file5
```

Заметьте, что количество ссылок в листинге каталога не обновилось. Удаление ссылки не влияет на файл, на который она ссылается. Символическая ссылка не защищает файл от удаления; в случае, если файл, на который указывает ссылка, перемещается или удаляется, эта ссылка будет испорчена. По этой причине во многих системах для обозначения символических ссылок в листинге каталогов используются различные цвета, чаще всего – светло-голубой для хороших ссылок и красный для испорченных.

Поиск символических ссылок

Если вам необходимо узнать, какие файлы символически ссылаются на определенный файл, вы можете использовать команду find и параметр -lname с указанием имени файла, как показано в листинге 62. В ссылках могут использоваться относительные и абсолютные пути, поэтому полезно поставить подстановочный символ «звездочка» перед именем искомого файла.

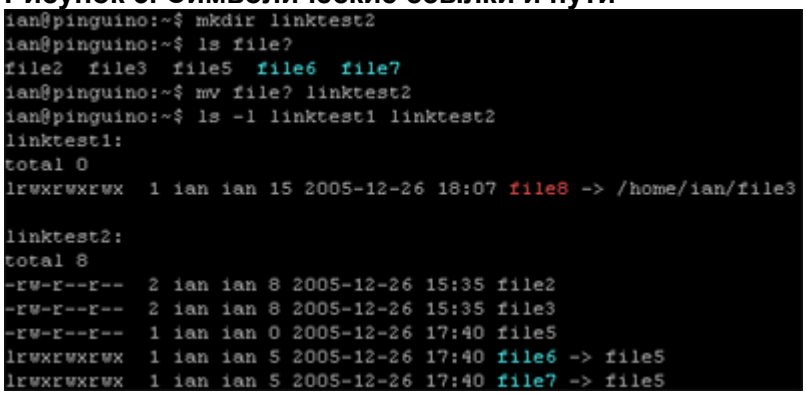
Листинг 62. Поиск символических ссылок

```
ian@pinguino:~$ mkdir linktest1
ian@pinguino:~$ ln -s ~/file3 linktest1/file8
.ian@pinguino:~$ find . -lname "*file3"
./linktest1/file8
ian@pinguino:~$ find . -lname "*file5"
./file7
./file6
```

Пути и символические ссылки

В большинстве рассмотренных ранее примеров символические ссылки находились в той же каталоги, что и файлы, на которые они указывают, а пути в ссылках, соответственно, были относительные. В листинге 62 мы создали ссылку в подкаталоге linktest1, которая указывала на абсолютное расположение файла which (~/file3). При создании символических ссылок вам необходимо решить, какие пути вы будете использовать, относительные или абсолютные, так как вам доступны оба варианта. На рисунке 3 показано действие перемещения нескольких файлов и символических ссылок в подкаталог.

Рисунок 3. Символические ссылки и пути



Красный цвет означает, что ссылка linktest1/file8 теперь испорчена. Это неудивительно, поскольку файла ~/file3 больше не существует. Однако две символические ссылки на file5 остаются хорошими, так как файл находится по тому же относительному пути, даже несмотря на то, что сам файл и обе ссылки были перемещены. Жестких и точных правил, которые позволяют выбрать, какие пути использовать в символических ссылках, абсолютные или относительные, не существует; отчасти это определяется вероятностью перемещения ссылок и файлов, на которые они указывают. Просто не забывайте учитывать этот вопрос при создании символических ссылок.

Испорченные символические ссылки

Одно завершающее замечание по нашей испорченной ссылке. Попытка чтения из файла завершится ошибкой, так как файл не существует. Однако попытка записи сработает, если у вас есть достаточно полномочий для записи в файл, на который указывает ссылка, как показано в листинге 63.

Листинг 63. Чтение и запись в испорченную символическую ссылку

```
ian@pinguino:~$ cat linktest1/file8
cat: linktest1/file8: No such file or directory
ian@pinguino:~$ echo "test file 8" >> linktest1/file8
ian@pinguino:~$ cat linktest1/file8
test file 8
ian@pinguino:~$ find . -name file3
./linktest2/file3
./file3
```

Поскольку я могу создавать файлы в моем каталоге home, запись в символическую ссылку приведет к созданию отсутствующего файла, на который она указывает.

Поиск и расположение системных файлов

Стандарт Filesystem Hierarchycal Standard

Стандарт Filesystem Hierarchycal Standard – это документ, который определяет структуру каталогов в Linux и других UNIX-подобных системах. Он был создан для формирования общей структуры, которая помогает упростить разработку программного обеспечения, независимого от дистрибутивов, путем расположения файлов на всех дистрибутивах Linux в одни и те же общие каталоги. Этот документ также используется в базе стандартов Linux (см. "Ресурсы").

Две независимые категории стандарта FHS

В основе стандарта Filesystem Hierarchycal Standard лежат две характеристики файлов:

Файлы общего или локального использования
: Файлы общего использования могут быть расположены на одной системе и использоваться на другой, а файлы локального использования должны храниться на той системе, где они используются.

Переменные или статические
: В число статических файлов входит документация, библиотеки и двоичные файлы, которые не изменяются без вмешательства системного администратора. Все остальные файлы являются переменными.

Такое разделение позволяет хранить файлы с разными характеристиками в различных файловых системах. В таблице 6 приведен пример из документа стандарта Filesystem Hierarchycal Standard, где показана структура, соответствующая этому стандарту.

Таблица 6. Пример реализации стандарта Filesystem Hierarchycal Standard

	Общего использования	Локального использования
Статический	/usr/opt	/etc/boot
Переменный	/var/mail /var/spool/news	/var/run/var/lock

Корневая файловая система

Целью стандарта Filesystem Hierarchycal Standard является максимально возможное сокращение корневой файловой системы. В ней должны содержаться все файлы, необходимые для загрузки, восстановления или исправления системы, в том числе утилиты, которые могут понадобиться опытным администраторам для выполнения их задач. Следует отметить, что для загрузки системы необходимо, чтобы в корневой файловой системе находились все средства, необходимые для монтирования других файловых систем.

Каталоги корневой файловой системы

В таблице 7 показано назначение каталогов, наличие которых в корневой файловой системе (/) обуславливает стандарт Filesystem Hierarchycal Standard. В ней должны присутствовать перечисленные каталоги или символические ссылки на них, за исключением отмеченных как необязательные, которые необходимы только в случаях, когда имеется соответствующая подсистема.

Таблица 7. Корневая файловая система стандарта Filesystem Hierarchycal Standard

Каталог	Назначение
bin	Двоичные коды необходимых команд
boot	Статические файлы загрузчика
dev	Файлы устройств
etc	Конфигурация системы для этого узла
lib	Необходимые модули ядра и библиотеки общего пользования
media	Точка монтирования для съемных носителей
mnt	Точка временного монтирования файловых систем
opt	Дополнительные пакеты программных приложений
sbin	Двоичные коды необходимых системных файлов
srv	Данные служб, предоставляемых этой системой
tmp	Временные файлы
usr	Дополнительная иерархия
var	Переменные данные
home	Домашние каталоги пользователей (необязательно)
lib<qual>	Важные библиотеки общего пользования в альтернативном формате (необязательно)
root	Домашний каталог пользователя root (необязательно)

/usr u /var

Иерархии /usr и /var достаточно сложны, для их описания выделены целые разделы стандарта Filesystem Hierarchycal Standard. Файловая система /usr является вторым важным разделом файловой системы, в котором содержатся данные только для чтения, находящиеся в общем доступе. Они могут использоваться различными системами, однако в современной практике это делается нечасто. В файловой системе /var содержатся переменные файлы данных, в том числе каталоги и файлы подкачки, данные администрирования и журналов, а также промежуточные и

временные файлы. Некоторая часть /var не может использоваться другими системами, однако другие части, например, /var/mail, /var/cache/man, /var/cache/fonts и /var/spool/news, могут находиться в общем доступе.

Для того чтобы полностью понять стандарт Filesystem Hierarchycal Standard, ознакомьтесь его описанием (см. раздел "Ресурсы").

Где искать файл?

В системах Linux зачастую содержатся сотни тысяч файлов. Только что установленная система Ubuntu, которую мы используем в этом руководстве, насчитывает около 50000 файлов только в иерархии /usr. В системе Fedora, с которой я работал одно время, насчитывается около 175000 файлов. В оставшейся части этого раздела мы рассмотрим инструменты, которые помогут вам найти файлы (в частности, программы) в этом необозримом море данных.

Ваш PATH

Если вы работали с несколькими системами Linux, вы могли отметить, что если вы входите как пользователь root, вы можете запускать такие команды, как fdisk, которые недоступны вам, если вы обычный пользователь. Когда вы запускаете программу в командной строке, командный процессор bash (или любой другой) производит поиск запрошенной вами программы по списку каталогов. Список каталогов указывается в переменной среды PATH, и нет ничего необычного во включении каталога /sbin в путь пользователя root, в то время как для остальных пользователей этого каталога в пути нет. В листинге 64 показаны два различных примера путей пользователя, а также пример пути для пользователя root.

Листинг 64. Несколько примеров PATH

```
ian@pinguino:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11
:/usr/games
[ian@attic4 ~]$ echo $PATH
/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/ian/b
in
[ian@attic4 ~]$ su -
Password:
[root@attic4 ~]# echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin
:/bin:
/usr/sbin:/usr/bin:/usr/X11R6/bin:/root/bin
```

Как вы можете видеть, переменная PATH представляет собой просто список наименований каталогов, разделенных двоеточиями. Поскольку команда fdisk фактически расположена по пути /sbin/fdisk, только первый и последний примеры путей позволят пользователю запускать эту команду, просто вводя fdisk, без указания полного имени (/sbin/fdisk).

Как правило, путь устанавливается в файле инициализации, например, в .bash_profile или .bashrc. Вы можете изменить его для текущей сессии путем указания нового пути. Для того чтобы новое значение было доступно другим

процессам, которые вы запускаете, нужно не забывать экспортировать переменную PATH. Пример приведен в листинге 65.

Листинг 65. Изменение переменной PATH

```
[ian@attic4 ~]$ fdisk
-bash: fdisk: command not found
[ian@attic4 ~]$ export PATH=/sbin:$PATH
[ian@attic4 ~]$ fdisk

Usage: fdisk [-l] [-b SSZ] [-u] device
E.g.: fdisk /dev/hda  (for the first IDE disk)
  or: fdisk /dev/sdc  (for the third SCSI disk)
  or: fdisk /dev/eda  (for the first PS/2 ESDI drive)
  or: fdisk /dev/rd/c0d0  or: fdisk /dev/ida/c0d0  (for RAID devices)
```

which, type, и whereis

В предыдущем примере мы узнали, что команда fdisk недоступна, только когда попытались её запустить. Существует несколько команд, которые помогут вам сделать это.

which

Вы можете использовать команду which для поиска по вашему пути программы, которая будет запускаться (если будет) при вводе команды. В листинге 66 показан пример поиска команды fdisk.

Листинг 66. Использование команды which

```
[ian@attic4 ~]$ which fdisk
/usr/bin/which: no fdisk in
(/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:
/usr/X11R6/bin:/home/ian/bin)
[ian@attic4 ~]$ export PATH=/sbin:$PATH
[ian@attic4 ~]$ which fdisk
/sbin/fdisk
```

Команда which показывает вам первую найденную команду в вашем пути. Если вы хотите узнать, нет ли нескольких соответствий, вы можете добавить параметр -a, как показано в листинге 67.

Листинг 67. Использование команды which для поиска нескольких файлов

```
[root@attic4 ~]# which awk
/bin/awk
[root@attic4 ~]# which -a awk
/bin/awk
/usr/bin/awk
```

Здесь мы нашли команду awk в каталоге /bin (который содержит команды, которые могут быть использованы и пользователями, и системным администратором, но не обязательные при отсутствии других смонтированных файловых систем), а также в каталоге /sbin (в котором содержатся исполняемые файлы, необходимые для загрузки, восстановления и исправления системы).

type

Существует ряд команд, которые не сможет найти команда which. К их числу, например, относятся встроенные команды командной оболочки. Встроенная команда type сообщит вам, как поданная вами команда будет оцениваться при выполнении. В листинге 68 показан пример использования команды type для самой себя.

Листинг 68. Использование команды type

```
[root@attic4 ~]# which type
/usr/bin/which: no type in
(/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:
/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/root/bin)
[root@attic4 ~]# type type
type is a shell builtin
```

whereis

Если вам нужна более подробная информация, а не просто место расположения программы, можно использовать команду whereis. Например, можно искать документацию man и иную информацию, как показано в листинге 69.

Листинг 69. Использование команды whereis

```
[root@attic4 ~]# whereis awk
awk: /bin/awk /usr/bin/awk /usr/libexec/awk /usr/share/awk
/usr/share/man/man1p/awk.1p.gz /usr/share/man/man1/awk.1.gz
```

Заметьте, что копия awk, расположенная в каталоге /sbin, не была найдена командой whereis. Число каталогов, используемых whereis, зафиксировано, поэтому команда не всегда может найти то, что вы ищете. Кроме того, команда whereis позволяет выполнять поиск исходных файлов, указывать альтернативные пути поиска, а также искать необычные элементы. Чтобы узнать, как изменить такое поведение или изменить фиксированные пути, используемые whereis, обратитесь к документации man.

find

Из руководства к теме 103 "Подготовка к экзамену LPI 101 (раздел 103): команды GNU и UNIX" вы узнали, как искать файлы по имени (в том числе – с использованием подстановочных знаков), пути, размеру и дате создания. В

предыдущем разделе, посвященном жестким и символическим ссылкам, вы узнали, как найти ссылки на определенный файл или узел inode.

Команда find – своего рода «швейцарский армейский нож» среди инструментов поиска файлов в системах Linux. У нее есть еще две возможности, которые могут быть вам полезны – это поиск файлов по имени пользователя и группы, а также поиск файлов по полномочиям.

В листинге 70 показан листинг каталога для рабочей группы ~greg/xml101, которую мы уже рассматривали ранее, а также пример того, как можно найти все файлы, владельцем которых является пользователь ian, и все файлы, не принадлежащие группе xml-101. Заметьте, что в команде find восклицательный знак ! инвертирует критерий поиска.

Листинг 70. Поиск файлов по пользователю и группе

```
ian@pinguino:~$ ls -l ~greg/xml101/*
-rw-r--r--  1 greg xml-101 0 2005-12-27 07:38 /home/greg/xml101/file1.c
-rw-r-----  1 greg xml-101 0 2005-12-27 07:39 /home/greg/xml101/file2.c
-rw-r--r--  1 tom  xml-101 0 2005-12-27 07:41 /home/greg/xml101/file3.c
-rw-r--r--  1 ian  ian     0 2005-12-27 07:40 /home/greg/xml101/file4.c
-rw-r--r--  1 tom  xml-101 0 2005-12-27 07:41 /home/greg/xml101/file5.c
-rw-r--r--  1 ian  xml-101 0 2005-12-27 07:40 /home/greg/xml101/file6.c
-rw-r--r--  1 tom  xml-101 0 2005-12-27 07:43 /home/greg/xml101/file7.c
-rwxr-xr-x  1 tom  xml-101 0 2005-12-27 07:42 /home/greg/xml101/myprogram
ian@pinguino:~$ find ~greg/xml101 -user ian
/home/greg/xml101/file4.c
/home/greg/xml101/file6.c
ian@pinguino:~$ find ~greg/xml101 ! -group xml-101
/home/greg/xml101/file4.c
```

Чтобы найти файлы по полномочиям, можно использовать критерий -perm вместе с символическим выражением, подобным используемому в командах chmod и umask. Вы можете выполнять поиск по точным полномочиям, однако чаще более полезным является использование в выражении полномочия префикса в виде дефиса, что означает указание искать файлы с установленными данными полномочиями, не обращая внимания на остальные полномочия. В листинге 71 показано с использованием файлов из листинга 70, как найти исполняемые файлы, и приведены два различных способа поиска файлов, которые не могут прочитать другие пользователи.

Листинг 71. Поиск файлов по полномочиям

```
ian@pinguino:~$ find ~greg/xml101 -perm -ug=x
/home/greg/xml101
/home/greg/xml101/myprogram
ian@pinguino:~$ find ~greg/xml101 ! -perm  -o=r
/home/greg/xml101/file2.c
ian@pinguino:~$ find ~greg/xml101 ! -perm  -0004
/home/greg/xml101/file2.c
```

Мы рассмотрели несколько основных видов поиска, которые можно выполнить с помощью команды find. Чтобы сузить результаты поиска, вы можете сочетать несколько расширений, а также добавлять регулярные выражения. Чтобы узнать больше об этой разносторонней команде, используйте документацию man, или лучше, если у вас установлена система info, введите команду info find.

В листинге 72 показан пример поиска с помощью команды find. В этом примере выполняется переход в каталог /usr/include, чтобы сохранить длину листинга в разумных пределах, после чего выполняется поиск всех файлов, содержащих в наименовании пути буквы xt с учетом регистра. Во втором примере вывод ограничивается ещё больше, до файлов, которые не являются каталогами и минимальный размер которых составляет 2 килобайта. Информация, которая будет выведена на вашей системе, может отличаться от этой в зависимости от того, какие пакеты у вас установлены.

Листинг 72. Заключительный пример использования команды find

```
ian@pinguino:/usr/include$ find . -iregex ".*xt.*"
./X11/xterm
./X11/xterm/ptyx.h
./irssi/src/fe-common/core/printtext.h
./irssi/src/fe-common/core/hilight-text.h
ian@pinguino:/usr/include$ find . -iregex ".*xt.*" ! -type d -size +2k
./X11/xterm/ptyx.h
./irssi/src/fe-common/core/printtext.h
```

Следует помнить, что регулярные выражения должны соответствовать полному пути, возвращаемому командой find, а также не забывать о разнице между регулярными выражениями и подстановочными символами.

locate и updatedb

Команда find выполняет поиск по всем каталогам, которые вы укажете, при каждом запуске. Для ускорения процесса вы можете использовать другую команду, locate, которая использует базу данных сохраненной информации о путях, а не выполняет всякий раз поиск по файловой системе.

locate и slocate

Команда locate выполняет поиск файлов по базе данных, которая обычно обновляется ежедневно с помощью задания cron. В современных системах Linux эта команда обычно замещается командой slocate, которая хранит полномочия вместе с путями, таким образом, не давая пользователям просматривать каталоги, права на просмотр которых у них нет. В этих системах вы можете увидеть, что locate является символической ссылкой на slocate, поэтому вы можете использовать любую команду.

Команда locate выполняет поиск по любой части пути, а не только по имени файла. В листинге 73 показано, что locate является символической ссылкой на locate, после чего приведен пример поиска пути, содержащего строку bin/ls.

Листинг 73. Использование команды locate

```
[ian@attic4 ~]$ ls -l $(which locate)
lrwxrwxrwx  1 root slocate 7 Aug 24 23:04 /usr/bin/locate -> slocate
[ian@attic4 ~]$ locate bin/ls
/bin/ls
/usr/bin/lsb_release
/usr/bin/lskatproc
/usr/bin/lspgpot
/usr/bin/lsattr
/usr/bin/lskat
/usr/bin/lshal
/usr/bin/lsdiff
/usr/sbin/lsof
/sbin/lsmod
/sbin/lsusb
/sbin/lspci
```

updatedb

База данных, используемая командой slocate, хранится в файловой системе /var, в файле /var/lib/slocate/slocate.db. Если вы увидите вывод, похожий на листинг 74 – значит, это задание в вашей системе не выполняется.

Листинг 74. Отсутствие базы данных slocate

```
[ian@attic4 ~]$ locate bin/ls
warning: locate: could not open database: /var/lib/slocate/slocate.db: No
such file or
directory
warning: You need to run the 'updatedb' command (as root) to create the
database.
Please edit /etc/updatedb.conf to enable the daily cron job.
```

База данных создается и обновляется с помощью команды updatedb. Обычно она выполняется ежедневно как задание cron. Файлом конфигурации updatedb является /etc/updatedb.conf. Для того чтобы включить ежедневное обновление, пользователь root должен изменить файл /etc/updatedb.conf и установить параметр DAILY_UPDATE=yes. Чтобы создать базу данных, запустите команду updatedb под пользователем root.