Впервые опубликовано на developerWorks 31.08.2005

Учебник для экзамена LPI 201: Файловая система

Администрирование, средний уровень (LPIC-2) тема 203

Создание и конфигурирование файловой системы

Начнем с создания и конфигурирования файловых систем и их параметров.

Создание разделов

Прежде чем вы сможете пользоваться файловыми системами Linux, вам нужно создать их. Для того чтобы создать файловую систему, вам сначала необходимо создать раздел, на котором вы впоследствии будете создавать файловую систему. На жестком диске с архитектурой x86 может быть создано до четырех первичных (primary) разделов, при этом последний из этих первичных разделов может быть расширенным (extended) и включать в себя множество логических разделов.

До недавнего времени существовали ограничения на порядковый номер цилиндра, на котором может находиться загрузочный раздел, на максимальные размеры дисков, на местоположение первичного раздела на больших дисках и так далее. Однако в последние несколько лет практически все системные BIOS научились управлять дисками независимо от их размера, и современные загрузчики (по крайней мере для Linux) не имеют никаких особенных ограничений, касающихся размеров разделов или их местоположения.

Единственный оставшийся момент, о котором следует побеспокоиться, это файловые системы, отличные от Linux. Для некоторых из них время от времени все же появляется необходимость располагаться на первичном разделе в начале жесткого диска. Разделы же Linux могут прекрасно жить как на расширенном разделе, так и на любом доступном диске.

Существует несколько широко используемых инструментов в мире Linux для того, чтобы создавать и управлять разделами на жестких дисках. Самый старый из них - `fdisk` . Несколько позже становится популярным основанный на `curses` инструмент `cfdisk` . Также часто используется для разбиения жестких дисков программа `parted` сообщества GNU. Ну а, в свою очередь, программы инсталляции, предлагаемые большинством известных дистрибутивов, и/или их графическая оболочка, обеспечивают удобный интерфейс для процедуры разбиения диска и просмотра таблицы разделов.

Из всех этих инструментов, утилита fdisk остается самым гибким и нетребовательным инструментом. Но не стоит обольщаться. Случайная запись неправильной таблицы разделов влечет проблемы независимо от того, какой программой вы пользуетесь. Но если ваши разделы были созданы нестандартными способами, например, не инструментами Linux, то, возможно, fdisk будет работать там, где другие инструменты могли бы отказаться пробовать вообще. Однако,

программа cfdisk более дружественна пользователю и более интерактивна (в том случае, если она не откажется работать). Кроме того, parted обладает великолепными возможностями для изменения размера и перемещения существующего раздела не хуже чем fdisk или cfdisk.

Какую бы программу вы не использовали, чтобы создать разделы, все они работают похожим образом. Эти действия вы должны выполнять как суперпользователь, лучше всего, в однопользовательском режиме. Трудно переоценить важность этого момента, поэтому будьте очень осторожны, когда вы изменяете разделы, сделайте резервную копию всех важных данных и обратите особое внимание на то, какие изменения вы делаете.

Прежде чем вы начнете изменять таблицу разделов, неплохо было бы выяснить, какие разделы уже существуют. С помощью команды `fdisk -l /dev/hda` (или такой же, но для других дисков, например, /dev/hdb или /dev/sda) вы можете узнать, какие разделы имеются в системе. Команда mount также полезна для выяснения, как фактически используются существующие разделы. Если вы хотите создать новый раздел, имейте в виду какие-нибудь свободные секторы на последнем первичном разделе, которые можно было бы использовать для нового расширенного раздела.

Взгляните на пример таблицы разделов в моей ОС Linux:

Листинг 1. Пример обычной таблицы разделов

```
% fdisk -l /dev/sda
Disk /dev/sda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders

Device Boot   Start      End      Blocks     Id  System
/dev/sda1   *        1     1216     9767488+     7  HPFS/NTFS
/dev/sda3         1217     4255    24410767+    83  Linux
/dev/sda4         4256     9729    43969905      5  Extended
/dev/sda5         4256     4380     1004031     82  Linux swap /
Solaris
/dev/sda6         4381     5597     9775521     83  Linux
```

Из этой таблицы можно кое-что почерпнуть. Во-первых, можно увидеть, что первый раздел, вероятно, используется другой операционной системой. Далее введем mount, чтобы выяснить, как используются разделы:

```
% mount | head -1
/dev/sda3 on / type reiserfs (rw,noatime,notail,commit=600)
```

Таким образом, существующая система смонтирована как корневая файловая система на /dev/sda3. Возможно, наиболее интересным наблюдением является то, что раздел /dev/sda4 простирается до 9729 цилиндра, но этот расширенный раздел использует только часть всего имеющегося места.

После обнаружения свободного места, доступного на жестком диске, используем его для создания нового раздела с помощью fdisk:

```
% fdisk /dev/sda
```

Наш жесткий диск насчитывает 9729 цилиндров. Это не является нарушением, но все же больше чем 1024 и, при определенных условиях, может вызвать проблемы с:

1. Загрузочным программным обеспечением (типа старых версий LILO)
2. Загрузочными и разбивающими жесткие диски программами от других операционных систем (например, DOS FDISK или OS/2 FDISK)

Листинг 2. Создание раздела

```
Command (m for help): n
Command action
l   logical (5 or over)
p   primary partition (1-4)
l
First cylinder (5598-9729), default 5598):
Using default value 5598
Last cylinder or +size or +sizeM or +sizeK (5598-9729, default 9729):
+10000M

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

Все, что следует за двоеточием, должно быть введено пользователем (вами). Таким образом, мы создали новый раздел Linux размером 10 GB.

```
/dev/sda7 5598 6814 9775521 83 Linux
```

Далее будет рассказано, как использовать вновь созданный раздел. Возможно, понадобится перезагрузить систему, чтобы получить доступ к новому разделу.

Создание файловой системы

Одного лишь наличия раздела недостаточно; вы должны создать на нем файловую систему. Мы создали новый раздел Linux /dev/sda7 и теперь должны выбрать, какой тип файловой системы, поддерживаемой Linux, использовать на этом разделе. Может быть, мы хотим создать исторически использующуюся по умолчанию файловую систему ext2? Или более новую журналируемую расширенную файловую систему формата ext3? Возможно, мы хотим создать одну из продвинутых файловых систем, привнесенных в Linux другими компаниями, например, ReiserFS, XFS или JFS. Или нам нужно иметь файловую систему, которая может взаимодействовать с другой операционной системой, типа Minix, MS-DOS, или VFAT (некоторые другие могут быть прочитаны, если уже созданы, но не всегда могут быть созданы инструментами Linux).

Для того, чтобы создавать новые файловые системы, нужно использовать следующие принятые обозначения `mkfs.*`. Таким образом, ваша файловая система может быть создана при помощи `mkfs.ext2`, `mkfs.minix`, `mkfs.xfx`, и так далее, обычно это установлено в /sbin/. Также вы можете задать любую из них, используя синтаксис `mkfs -t <fstype>`. Для некоторых (не всех) типов файловых систем имеется короткая форма записи, например, `mke3fs`. Возможность создания определенного типа файловой системы зависит от дистрибутива, который вы

используете, и его версии, а также, от дополнительного ПО, которое вы самостоятельно установили. При этом заметим, что `mkfs.ext2` имеется практически в любом дистрибутиве.

Создать файловую систему достаточно просто. Вам нужно только применить инструмент `mkfs.*` к нужному разделу (к тому, на котором вы хотите ее создать). Например:

```
% mkfs.xfs /dev/sda7
```

Сообщения, которые вы далее увидите, зависят от файловой системы, которую вы предпочли. Вообще, эти сообщения дают вам информацию относительно числа inod'ов, блоков, типа журналирования (если имеется журналируемость), протяженности, и другую, соответствующую характеру использующейся файловой системы. Многие (но не все, к сожалению) из средств, создающих файловую систему предупредят вас, если вы решите создать новую файловую систему на разделе с уже существующей файловой системой, так что приступайте с большой осторожностью (создание новой файловой системы поверх старой может привести к потере данных).

Создание файловой системы ISO при помощи cdrecord

Передача образа ISO на записываемый компакт-диск или DVD-диск в настоящее время часто осуществляется посредством связывающих инструментов, таких как интерфейс GUI. Например, и Gnome и KDE осуществляют запись компакт-дисков через интерфейс файлового менеджера. Существуют также удобные коммерческие программы, но для системного администратора старая добрая команда cdrecord представляется наиболее надежной из средств, имеющихся в большинстве современных дистрибутивов, и намного ближе к "стандарту", чем другие программы. Вообще, нужно только определить устройство, на которое вы хотите записать, да файл ISO, который вы хотите записать.

Вы можете также определить как обычно множество параметров процесса записи, например, `-overburn` для компакт-дисков больших, чем 650 MB или определенную скорость записи для вашего записывающего устройства. Советую прочесть страницу помощи `man` для `cdrecord` для уточнения деталей.

Вы можете обнаружить записывающее устройство при помощи `-scanbus`. Устройство, которое вам нужно, имеет числовой индикатор шины, состоящий из трех цифр, и не является блочным устройством в вашей файловой системе. Например, можно увидеть нечто, похожее на следующее (сокращенно):

Листинг 3. Обнаружение записывающего устройства

```
% cdrecord -scanbus
[...]
scsibus0:
0,0,0     0) 'ATA     ' 'WDC WD800UE-00HC' '09.0' Disk
0,1,0     1) *
[...]
scsibus1:
1,0,0   100) 'Slimtype' 'DVDRW SOSW-852S ' 'PSB2' Removable
CD-ROM
```

```
[...]
```

Получив информацию с шины, вы можете записать образ:

```
% sudo cdrecord -overburn -v speed=16 dev=1,0,0
/media/KNOPPIX_V3.6-2004-08-16-EN.iso
```

В этом случае образ небольшого размера, и я знаю, что мое записывающее устройство поддерживает скорость 16x. Вывод команды будет весьма многословным из-за использования флага -v, но это помогает в понимании процесса.

Создание файловой системы ISO при помощи dd

В заключение, иногда бывает необходимо создать новый образ ISO не из каталогов в вашей главной файловой системе, а из уже существующего компакт-диска или DVD-диска. Чтобы сделать образ ISO из компакт-диска, используйте команду `dd`, но обращайтесь к физическому адресу блочного устройства для компакт-дисков, а не к точке монтирования:

```
% dd if=/dev/cdrom of=project-cd.iso
```

Вы могли бы задаться вопросом, почему не используют команду cp, если цель состоит в том, чтобы просто скопировать байты. Фактически, если вы игнорируете ошибку ввода/вывода, о которой сообщается, когда устройство исчерпывает байты, подлежащие копированию, команда cp, вероятно , может работать. Все же, команда dd обладает лучшими качествами (она не жалуется, а вместо этого сообщает об итогах своей работы).

Управление файловой системой Linux

Монтирование и отмонтирование при помощи mount и umount

Одна из особенностей гибкости систем Linux -- это прекрасная детальная настройка контроля, который имеет пользователь, над подмонтированными и отмонтированными файловыми системами. В отличие от Windows и некоторых других операционных систем, местоположения разделов не автоматически закреплены ядром Linux, а присоединены к иерархии корневой файловой системы командой mount. Кроме того, различные типы файловых систем (даже на различных устройствах) могут быть смонтированы в рамках той же самой иерархии. Вы можете отмонтировать конкретный раздел командой umount, назначать любую точку монтирования (например, /home) или адрес устройства (например, /dev/hda7).

Когда производится восстановление файловой системы, возможность управлять точками монтирования позволяет вам проводить анализ состояния разделов, используя fsck или другие инструменты, без риска дальнейшего повреждения уже поврежденной файловой системы. Вы можете также в обычном порядке монтировать файловую систему, используя различные параметры; самые важные из них монтируют файловую систему для использования только в режиме чтения с помощью одного из синонимов `-r` или `-o ro`.

В качестве примера, вы могли бы хотеть заменить местоположение каталога одного

пользователя на каталог другого, или из-за повреждения раздела, или просто хотите расширить дисковое пространство, или переместиться на более быстрый диск. Такое изменение можно выполнить, используя:

```
# umount /home # old /dev/hda7 home dir
# mount -t xfs /dev/sda1 /home # new SCSI disk using XFS
# mount -t ext3 /dev/sda2 /tmp # also put the /tmp on SCSI
```

Монтирование по умолчанию

Для повседневной работы, вам будет удобно, чтобы необходимый конкретный набор подмонтирований осуществлялся автоматически при каждой загрузке системы. Вы управляете точками монтирования, которые происходят в процессе загрузки, прописывая нужные строки конфигурации в файл /etc/fstab. Типичная конфигурация могла бы выглядеть так:

Листинг 4. Пример конфигурации для монтирования при загрузке

```
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
       proc            /proc           proc    defaults        0       0
       /dev/sda3       /               reiserfs notail         0       1
       /dev/sda5       none            swap    sw              0       0
       /dev/sda6       /home           ext3    rw              0       2
       /dev/scd0       /media/cdrom0   udf,iso9660 ro,user,noauto  0   0
       /media/Ubuntu-5.04-install-i386.iso /media/Ubuntu_5.04 iso9660
rw,loop 0 0
```

В этом листинге, первое поле (<file system>) - обычно является названием блочного устройства, подлежащего монтированию. Второе (<mount point>) – точка монтирования. В некоторых специальных случаях сначала пишется вовсе не обозначение блочного устройства. Для устройств `supermount`, вы увидите none. `/proc` - другой особый случай. Вы могли бы также подмонтировать loopback устройства, которые являются обычно обычными файлами.

Третье (<type>) и четвертое (<options>) поля являются довольно простыми; эти параметры зависят от типа файловой системы и предполагаемого использования. Пятая поле (<dump>) - обычно ноль. Шестая поле (<pass>) должно содержать 1 -- для корневой файловой системы и 2 -- для других файловых систем, которые должны быть проверены при помощи `fsck` во время загрузки системы.

Автоматическое монтирование с помощью AMD и automount

В Linux существует довольно много способов автоматического монтирования ресурсов, которые являются сменными (дискеты, компакт-диски, USB-устройства) или не находятся в состоянии готовности (например, файловых систем NFS). Цель всех этих инструментов схожа, но все они работают немного по-разному.

Инструмент AMD (демон автомонтирования) несколько старше других и работает в пространстве пользователя. AMD периодически запускается, чтобы проверить, не стали ли какие-нибудь подлежащие монтированию файловые системы доступны; в основном, для файловых систем NFS. В большинстве своем, AMD был заменен в дистрибутивах Linux на Autofs, который работает уже как ядерный процесс.

При сборке ядра, которое вы будете использовать, следует включить Autofs. После этого, поведением демона Autofs (обычно /etc/init.d/autofs) управляет файл /etc/auto.master, который, в свою очередь, ссылается на map файл. Например:

```
# Sample auto.master file
# Format of this file: mountpoint map options
/mnt /etc/auto.mnt --timeout=10
```

Файл `/etc/auto.mnt`, на который здесь ссылаются, определяет один или более подкаталогов /mnt, которые будут смонтированы (если доступ будет затребован). Отмонтирование в этом случае произойдет автоматически спустя 10 секунд после последнего доступа.

```
# Sample /etc/auto.mnt
floppy -fstype=auto,rw,sync,umask=002 :/dev/fd0
cdrom -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom
remote -fstype=nfs example.com:/some/dir
```

Автоматическое монтирование при помощи supermount и submount

Инструменты `supermount` и `submount`, работаеют на уровня ядра (либо включаются в основное ядро, либо в модули ядра), используются для автоматического монтирования сменных ресурсов при обращении к ним. submount немного поновее, но в большем количестве дистрибутивов все еще, вероятно, используется `supermount`. Ни один из них не поможет при удаленном монтировании по NFS, но нет инструментов лучше Autofs, при подключении локальных ресурсов.

Все устройства, требующие автомонтирования, перечислены в файле /etc/fstab. Инструменты используют в /etc/fstab немного разный, но достаточно простой синтаксис. Запускающийся посредством supermount /etc/fstab, мог бы содержать следующее:

```
# Example of supermount in /etc/fstab
none /mnt/cdrom supermount fs=auto,dev=/dev/cdrom 0 0
none /mnt/floppy supermount fs=auto,dev=/dev/fd0,--,user,rw 0 0
```

`submount` описывает блочное устройство в его постоянном местоположении, а не как точку монтирования. Например:

```
/dev/cdrom /mnt/cdrom subfs fs=cdfss,ro,users 0 0
/dev/fd0 /mnt/floppy subfs fs=floppyfss,rw,users 0 0
```

Что сейчас подмонтировано?

Пользователь Linux имеет возможность увидеть список текущих монтирований несколькими способами. Команда `mount` без дополнительных аргументов (или с выбором -l) перечисляет монтирования, установленные в настоящее время. Вы можете при желании фильтровать результаты выбором аргумента `-t fstype`.

Основная динамическая информация, касающаяся подмонтированных файловых систем, находится в /etc/mtab. Команды mount и umount и другие системные процессы обновляют этот файл, чтобы отразить текущий статус; вы должны

рассматривать этот файл как “файл только для чтения”. Дополнительную информацию относительно текущего состояния mount можно найти в /proc/mounts.

Специальные инструменты

Инструмент `sync` передвигает незаписанные блоки на диске. У вас нет необходимости использовать этот инструмент в нормальных ситуациях, но вы можете иногда проверять диск, but you can sometimes check for disk problems by checking for a non-zero exit status. Современные и, особенно, журналируемые файловые системы, такие как ext3, Reiser, и JFS эффективно делают синхронизирование при каждой записи.

Если захотите, вы можете вручную запретить или позволить использование свопинга или сопоставить свопинг конкретному устройству. Обычно, любое устройство, отмеченное в /etc/fstab как swap, используется для свопинга.

Поддержка файловой системы Linux

Исправление файловой системы при помощи fsck

Ваш лучший друг в восстановлении поврежденной файловой системы это `fsck`.

То, что мы называем `fsck`, является только началом множества более тонких инструментов `fsck.*`, например: `fsck.ext2`, `fsck.ext3`, или `fsck.reiser`. Можно определить тип точно, используя аргумент `-t`, но `fsck` предпримет усилие понять это самостоятельно. Прочитайте страницы помощи `man` для `fsck` или `fsck.*` для уточнения деталей. Основное, что вам нужно знать, это то, что при использовании аргумента `-a` программа будет пытаться исправить все найденные ошибки.

Вы можете проверить неподмонтированную файловую систему, упоминая местонахождение устройства, на котором она находится. Например, введите `fsck /dev/hda8`, чтобы проверить неиспользующийся раздел. Вы можете также проверить корневую файловую систему, набрав, например, `fsck /home`, но, как правило, делают это, только если файловая система уже смонтирована как "только для чтения", а не для "чтения-записи".

Проверка блоков с помощью badblocks

Утилита `badblocks` производит проверку качества блочного устройства (или раздела) на более низком уровне, чем это делает f`sck. badblocks` исследует надежность блоков на устройстве, записывая и читая тестовые образцы. Используйте аргумент -n для более медленного исследования, при котором сохраняются существующие данные. Для совершенно нового раздела без существующих файлов, вы можете (и вероятно должны), использовать аргумент `-w`. Этот вариант просто сообщит вам про плохие блоки, не восстанавливая и не отмечая их.

На практике все же более предпочтительнее использовать для проверки на плохие

блоки `fsck.*` для вашей файловой системы. Например: `e2fsck` (также можно вызывать `fsck.ext2`) имеет аргумент `-c` чтобы найти и пометить плохие блоки, которые может обнаружить `badblocks`. ReiserFS имеет аналогичные аргументы `--check` и `--badblocks` (но не совсем автоматические). Прочитайте документацию для вашей конкретной файловой системы по использованию `badblocks`.

Поиск других программ поддержки

Существует несколько инструментов для исследования и настройки файловых систем Linux. Для работы в обычном режиме вам будет достаточно настроек по умолчанию, но иногда нужно произвести более детальные исследования и масштабные действия, например, на поврежденных системах, или настроить работу в системе в точном соответствии с шаблоном.

Каждый тип файловой системы имеет свой собственный набор инструментов. Для получения более подробной информации проверьте документацию для файловой системы, которую вы используете. Большинство из них имеют сходный набор инструментов. Вот некоторые примеры:

- `dumpe2fs`: Выходная информация о файловой системе ext2/3.
- `tune2fs`: Регулировка параметров файловой системы для ext2/3.
- `debugfs`: Настройка и проверка файловой системы ext2/3 в интерактивном режиме.
-
- `debugreiserfs`: Выходная информация о файловой системе Reiser.
- `reiserfstune`: Регулировка параметров файловой системы для Reiser.
- `xfs_admin`: Регулировка параметров файловой системы для XFS.