# 6 Linux Filesystem Administration

## CERTIFICATION OBJECTIVES

| | | | |
|---|---|---|---|
| 6.01 | Storage Management and Partitions | 6.05 | Filesystem Management |
| 6.02 | Filesystem Formats | 6.06 | The Automounter |
| 6.03 | Basic Linux Filesystems and Directories | ✓ | Two-Minute Drill |
| 6.04 | Logical Volume Management (LVM)Volume Encryption with the Linux Unified Key Setup | Q&A | Self Test |

Once in production, it's rare to install a new Linux system from scratch. Besides, Linux installation is easy, at least for anyone serious about Red Hat certification. Most administrators have to maintain existing systems. Critical skills related to filesystems include adding new partitions, creating logical volumes, setting up volume encryption, and more. In many cases, you'll want to make sure these filesystems are mounted automatically during the boot process, and that requires a detailed knowledge of the /etc/fstab configuration file.

Some filesystems, such as those available from unreliable network connections, should be mounted only on a temporary basis; that is the province of the automounter.

## INSIDE THE EXAM

Some of the RHCSA objectives listed in this chapter overlap and may be in part covered in multiple sections. The objectives all relate in some way to filesystem management and should be considered as a whole in this chapter.

### Partition Management

As in the real world, it is the results that matter. It doesn't matter whether you use Disk Druid, **fdisk**, or **parted** to create partitions. You can create new partitions at the command line or use GUI front ends to these tools such as the Disk Utility. Make sure that appropriate partitions meet the requirements of the exam. Just remember Disk Druid is available only during the installation process.

The current RHCSA objectives include the following related requirements:

- Add new partitions, logical volumes, filesystems, and swap areas to a system non-destructively
- List, create, delete, and set partition type for primary, extended, and logical partitions

### Logical Volumes

Partitions are essential components of logical volumes. Related RHCSA objectives describe some of the skills required. For example, the following objective suggests that you need to know the process starting with the physical volume:

- Create and remove physical volumes, assign physical volumes to volume groups, create and delete logical volumes

Of course, a logical volume isn't fulfilling its full potential unless you can increase its size, as suggested by the following objective:

- Extend existing unencrypted ext4-formatted logical volumes

### Storage Encryption

RHEL 6 includes the Linux Unified Key Setup (LUKS) system for encrypting formatted partitions and volumes. When properly configured, LUKS makes it more difficult for a cracker to decipher critical data. To that end, the relevant RHCSA requirements include:

- Mount, unmount, and use LUKS-encrypted file systems
- Create and configure LUKS-encrypted partitions and logical volumes to prompt for password and mount a decrypted file system at boot
- Configure systems to mount ext4, LUKS-encrypted, and network file systems automatically

### Filesystem Management

Partitions and logical volumes must be formatted before they're ready to store files. To that end, you need to know how to meet the following RHCSA objectives:

- Create, mount, unmount, and use ext2, ext3 and ext4 file systems
- Mount and unmount CIFS and NFS network file systems
- Configure systems to mount file systems at boot by Universally Unique ID (UUID) or label

**CERTIFICATION OBJECTIVE 6.01**

## Storage Management and Partitions

While it's easier to create partitions, logical volumes, and RAID arrays during the installation process, not every administrator has that privilege. While this section is focused on the management of regular partitions, the techniques described in this section are also used to create the partition-based components of both logical volumes and RAID arrays. Once configured, a partition, a logical volume, and a RAID array can each be referred to generically as a volume.

In Linux, two tools still predominate for administrators who need to create and manage partitions: **fdisk** and **parted**. While these tools are primarily applied to local hard disks, they're frequently also used for other media such as drives attached over a network.

on the job

***For both fdisk and parted, partitions configured as logical volumes are given an LVM label, short for Logical Volume Management.***

## Current System State

Before using the **fdisk** or **parted** utilities to create or modify a partition, it's important to check currently available free space along with mounted filesystems. That process is made easy with the **df** and **mount** commands. The following example illustrates how the **df** command displays the total, used, and available free space on all currently mounted filesystems.

on the job

***The terms filesystem and file system are interchangeable. Both are used in official Linux documentation.***

Note the numbers under the 1k-blocks column. In this case (except for the temporary filesystem, tmpfs, and the mounted DVD), they add up to about 10GB of allocated space. If the hard drive is larger, unallocated space may be used for another partition. Partitions can be combined with others to configure additional space in logical volumes and RAID arrays. And that can be useful when you need to expand the space available to appropriate directories, such as /home, /tmp, and /var.

```
[root@server1 ~]# df
Filesystem 1k-blocks      Used Available Use% Mounted on
/dev/vda2
              8063408   2158968   5494840  29% /
tmpfs          384576         0    384576   0% /dev/shm
/dev/vda1      495844     32140    438104   7% /boot
/dev/vda5     1007896     17716    938980   2% /home
/dev/sr0      2381288   2381288         0 100% /media
```

The second command, **mount**, lists the way each filesystem is formatted. In this case, examine the partition represented by device /dev/vda5 mounted with the ext4 file type on the /home directory. It separates the directories of regular users in a dedicated partition. For the following example, I've set up the data shown from the **mount** command in columns for clarity; what you actually see from the RHEL command line is less organized.

```
[root@server1 root]# mount
/dev/vda2 on / type ext4                                  (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)
/dev/vda1 on /boot type ext4                              (rw)
/dev/vda5 on /home type ext4                              (rw)
none      on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/tmp on /tmp type none                               (rw,bind)
/var/tmp on /var/tmp type none                       (rw,bind)
/home on /home type none                             (rw,bind)
sunrpc    on /var/lib/nfs/rpc_pipefs type rpc_pipefs   (rw)
/dev/sr0  on /media type iso9660                          (ro)
```

## The fdisk Utility

The **fdisk** utility is a near-universal tool available for a variety of computer operating systems. A capable version of **fdisk** is available on Macintosh OSes. A less capable version of **fdisk** is even available on older versions of Microsoft Windows. There are many commands within **fdisk**, more in expert mode, but you need to know only the few discussed here.

Though you can modify the physical disk partition layout using many programs, this section explores the Linux implementation of **fdisk**. In contrast, the Microsoft version is based on its heritage as the Disk Operating System (DOS) and works only on Microsoft partitions.

### Start fdisk: Help and More

The following screen output lists commands that show how to start the **fdisk** program, how to get help, and how to quit the program. The /dev/vda drive is associated with the first virtual drive on a KVM-based virtual machine. As other systems may be configured with different hard drive device files, you may need to check the output from the **df** and **mount** commands for clues.

When you start **fdisk**, it now includes the following warning:

```
WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').
```

There's no requirement to make these changes. If you do, be prepared to make these changes again the next time fdisk is open.

However, such changes can help avoid annoying error messages. And the results are more precise, as sectors more closely match desired partition sizes. On production systems, it's unlikely that you're going to add or modify partitions on a daily basis. Whether or not these changes are made, **fdisk** provides the same prompt, where you can press **m** to list basic **fdisk** commands:

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

There are a wide variety of commands associated with **fdisk**—and more if you run the **x** command to access **fdisk**'s extra functionality.

## Using fdisk: A New Drive with No Partitions

After installing a new drive on Linux, that drive normally isn't configured with partitions. The **fdisk** utility can be used to configure partitions on physical or virtual disks attached to the system. For example, the baseline virtual system for this book includes three drives: /dev/vda, /dev/sda, and /dev/sdb.

on the job

***SATA, PATA, and SCSI drives are now all represented by device files like /dev/sda, /dev/sdb, and so on.***

If a newly added drive hasn't been used by the RHEL installation program (or some other disk management program), it'll return the following message the first time it's opened by **fdisk**:

```
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0xa15e5d53.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
```

In other words, even if you don't create a partition after opening it in **fdisk**, it will automatically write a DOS disk label to the drive. Yes, despite the previous warning about DOS-compatible mode, **fdisk** still uses the Disk Operating System commonly associated with Microsoft. But DOS predates Microsoft.

If you need more than four partitions on the new physical disk, configure the first three partitions as primary partitions, and then configure the fourth partition as an extended partition. That extended partition should be pretty big; all logical partitions must fit in that space.

## Using fdisk: In a Nutshell

At the **fdisk** command line prompt, start with the print command (**p**) to print the partition table. This allows you to review the current entries in the partition table. Assuming free space is available, you can then create a new (**n**) partition. Generally, partitions are either primary (**p**) or logical (**l**). If it doesn't already exist, you can also create an extended partition (**e**) to contain logical partitions. Remember that you can have up to four primary partitions, which would correspond to numbers 1 through 4. One of the primary partitions can be redesignated as an extended partition. The remaining partitions are logical partitions, numbered 5 and above. While you might think **fdisk** is old and unmaintained, it actually now supports the creation of more than 16 partitions on a drive.

If free space is available, **fdisk** normally starts the new partition at the first available sector or cylinder. The actual size of the partition depends on disk geometry.

## Using fdisk: Create a Partition

The following screen output sample shows the steps used to create (**n**) the first (/boot) partition, make it bootable (**a**), and then finally write (**w**) the partition information to the disk. (Note that although you may specify a 500MB partition, the geometry of the disk may not allow that precise size.) First, to avoid error messages described earlier, the DOS compatibility flag is unset (**c**) and display units are changed to sectors (**u**).

```
# fdisk /dev/sda

Command (m for help): c
DOS Compatibility flag is not set

Command (m for help): u
Changing display/entry units to sectors

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (2048-2047999, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-2047999, default 2047999): +500M

Command (m for help): a
Partition number (1-4): 1

Command (m for help): p
Disk /dev/sda: 1048 MB, 1048576000 bytes
.....
   Device Boot     Start         End     Blocks   Id  System
/dev/sda1   *       2048     1026047     512000   83  Linux

Command (m for help):
```

Note how the number of blocks matches the binary representation of 500MB. Repeat the commands to create any other partitions that you might need.

When partitions are added or changed, you generally don't have to reboot to get Linux to read the new partition table, unless another partition on that drive has been formatted and mounted. If so, an attempt to write the partition table with the **w** command fails temporarily with the following message:

```
WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
 The kernel still uses the old table. The new table will be used at
 the next reboot or after you run partprobe(8) or kpartx(8)
```

If you're able to unmount existing partitions or volumes on the target hard drive, you'd be able to apply the **partprobe** or **kpartx** commands to the device file of that hard drive. For example, until other applicable volumes from that drive are unmounted, the **partprobe /dev/sda** command returns the same error message.

## Using fdisk: Many Partition Types

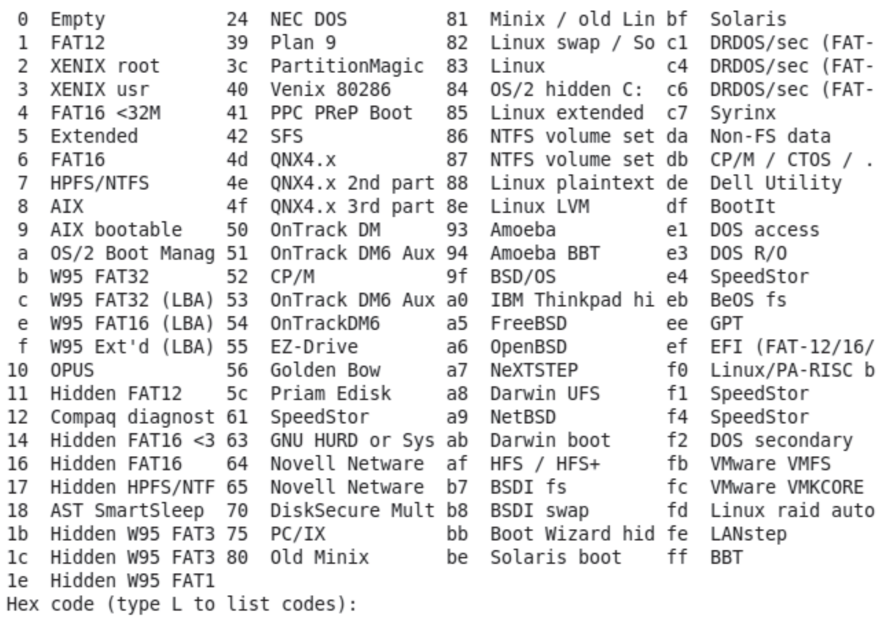
One feature of special interest is based on the **t** command, to change the partition system identifier. If you need space for logical volumes, RAID arrays, or even swap space, that command is important. After pressing **t**, you're prompted to enter the partition number (if there's more than one configured). You can then list available partition types with the **L** command, as shown here. (If there's only one partition on the drive, it is selected automatically.)

```
Command (m for help) : t
Partition number (1-4)
1
Partition ID (L to list options): L
```

The list of available partition identifiers, as shown in Figure 6-1 is impressive. Note how it's not limited to Linux partitions. But as this book covers Linux, Table 6-1 lists associated partition types.

**FIGURE 6-1**

Linux partition types infdisk



```
 0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
 1  FAT12           39  Plan 9          82  Linux swap / So c1  DRDOS/sec (FAT-
 2  XENIX root      3c  PartitionMagic  83  Linux           c4  DRDOS/sec (FAT-
 3  XENIX usr       40  Venix 80286     84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 4  FAT16 <32M      41  PPC PReP Boot   85  Linux extended  c7  Syrinx
 5  Extended        42  SFS             86  NTFS volume set da  Non-FS data
 6  FAT16           4d  QNX4.x          87  NTFS volume set db  CP/M / CTOS / .
 7  HPFS/NTFS       4e  QNX4.x 2nd part 88  Linux plaintext de  Dell Utility
 8  AIX             4f  QNX4.x 3rd part 8e  Linux LVM       df  BootIt
 9  AIX bootable    50  OnTrack DM      93  Amoeba          e1  DOS access
 a  OS/2 Boot Manag 51  OnTrack DM6 Aux 94  Amoeba BBT      e3  DOS R/O
 b  W95 FAT32       52  CP/M            9f  BSD/OS          e4  SpeedStor
 c  W95 FAT32 (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi eb  BeOS fs
 e  W95 FAT16 (LBA) 54  OnTrackDM6      a5  FreeBSD         ee  GPT
 f  W95 Ext'd (LBA) 55  EZ-Drive        a6  OpenBSD         ef  EFI (FAT-12/16/
10  OPUS            56  Golden Bow      a7  NeXTSTEP        f0  Linux/PA-RISC b
11  Hidden FAT12    5c  Priam Edisk     a8  Darwin UFS      f1  SpeedStor
12  Compaq diagnost 61  SpeedStor       a9  NetBSD          f4  SpeedStor
14  Hidden FAT16 <3 63  GNU HURD or Sys ab  Darwin boot     f2  DOS secondary
16  Hidden FAT16    64  Novell Netware  af  HFS / HFS+      fb  VMware VMFS
17  Hidden HPFS/NTF 65  Novell Netware  b7  BSDI fs         fc  VMware VMKCORE
18  AST SmartSleep  70  DiskSecure Mult b8  BSDI swap       fd  Linux raid auto
1b  Hidden W95 FAT3 75  PC/IX           bb  Boot Wizard hid fe  LANstep
1c  Hidden W95 FAT3 80  Old Minix       be  Solaris boot    ff  BBT
1e  Hidden W95 FAT1
Hex code (type L to list codes):
```

**TABLE 6-1**

Some Linux Partition Types

| Partition Identifier | Description |
|---|---|
| 5 | Extended partition; while not a Linux partition type, such partitions are a prerequisite for logical partitions. Also see 85. |
| 82 | Linux swap |
| 83 | Linux; applicable for all standard Linux partition formats |
| 85 | Linux extended partition; not recognized by other operating systems. |
| 88 | Linux plaintext partition table; rarely used. |
| 8e | Linux logical volume management for partitions used as physical volumes |
| fd | Linux RAID; for partitions used as components of a RAID array |

Unless you're making a change, type in identifier 83. You'll be returned to the fdisk command prompt.

## Using fdisk: Delete a Partition

The following example removes the only configured partition. The sample output screen first starts **fdisk**. Then you can print (**p**) the current partition table, delete (**d**) the partition by number (**1** in this case), write (**w**) the changes to the disk, and quit (**q**) from the program. Needless to say, *do not perform this action on any partition where you need the data.* The following output is based on a system where the aforementioned DOS compatibility flag has been disabled and system units have been changed to sectors with the **c** and **u** commands, respectively.

Assuming only one partition on this drive, it is selected automatically after running the **d** command.

```
# fdisk /dev/sda
Command (m for help): p
Disk /dev/sda: 1048 MB, 1048576000 bytes
255 heads, 63 sectors/track, 127 cylinders, total 2048000 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x654f1cda

Device      Boot     Start       End     Blocks   Id  System
/dev/sda1             2048    206847     102400   83  Linux
Command (m for help): d
Partition number (1-1): 1
```

**This is the last chance to change your mind before deleting the current partition. To avoid writing the change, exit from fdisk** with the **q** command. If you're pleased with the changes that you've made and want to make them permanent, proceed with the **w** command:

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

Unless the aforementioned error 16 message appears, that's it. You should now have an empty hard drive.

## Using fdisk: Create a Swap Partition

Now that you know how to create partitions with **fdisk**, just one additional step is required to set up that partition for swap space. Once you have a swap partition of the desired size, run the **t** command to select a partition, and then run the **l** command to show the partition ID types listed in Figure 6-1.

In this case, at the following prompt, type in **82** for a Linux swap partition:

```
Hex code (type L to list codes): 82
```

For example, I could run the following sequence of commands to set up a new swap partition on the second hard drive. The commands that I type are in boldface. The details of what you see depend on the partitions that you may have created. It'll be a 900MB swap space on the first primary partition (/dev/sdb1).

```
Command (m for help): n
Command action
     e  extended
     p  primary partition (1-4)
p
Partition number (1-4): 1
First sector (2048-2047999, default 2048): 2048
Last sector, +sectors or +size{K,M,G} (2048-2047999, default 2047999): +900M

Command (m for help): p

Disk /dev/sdb: 1048 MB, 1048576000 bytes
...

   Device Boot     Start        End     Blocks    Id  System
/dev/sdb1           2048    1845247     921600    83  Linux

Command (m for help): t
Selected partition 1
```

```
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap / Solaris)

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

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

The **fdisk** utility doesn't actually write the changes to disk until you run the write (**w**) command. Alternatively, you can cancel these changes with the quit (**q**) command. If you don't have the error 16 message described earlier, the changes are written to disk. As described later in this chapter, additional work is required to implement that swap partition.

## The parted Utility

In its different forms, the **parted** utility is becoming increasingly popular. It's an excellent tool developed by the GNU foundation. As with **fdisk**, you can use it to create, check, and destroy partitions, but it can do more. You can also use it to resize and copy partitions, as well as the filesystems contained therein. It's the foundation for multiple GUI-based partition management tools, including GParted and QtParted. For the latest information, see www.gnu.org/software/parted. As discussed later in this chapter, RHEL 6 includes the Disk Utility, available from the gnome-disk-utility package and the **palimpsest** command.

on the job

***In some ways, the parted utility may be more risky. For example, I accidentally ran the mklabel command from the (parted) prompt on an existing RHEL system. It deleted all existing partitions. Changes were written immediately, while parted was still running. Fortunately, I had a backup of this virtual system and was able to restore it with little trouble.***

exam watch

***Real Linux administrators understand that partition management tools are inexact. While fdisk is improved when sectors are used, parted does not have that option.***

During our discussion of **parted**, we'll proceed from section to section assuming that **parted** is still open with the following prompt:

```
(parted)
```

## Using parted: Starting, Getting Help, and Quitting

The next screen output lists commands that show how to start the **parted** utility, how to get help, and how to quit the program. In this case, the /dev/sdb drive is associated with the second SATA drive on a regular PC. Your computer may have a different hard drive; you can check the output from the **df** and **mount** commands for clues.

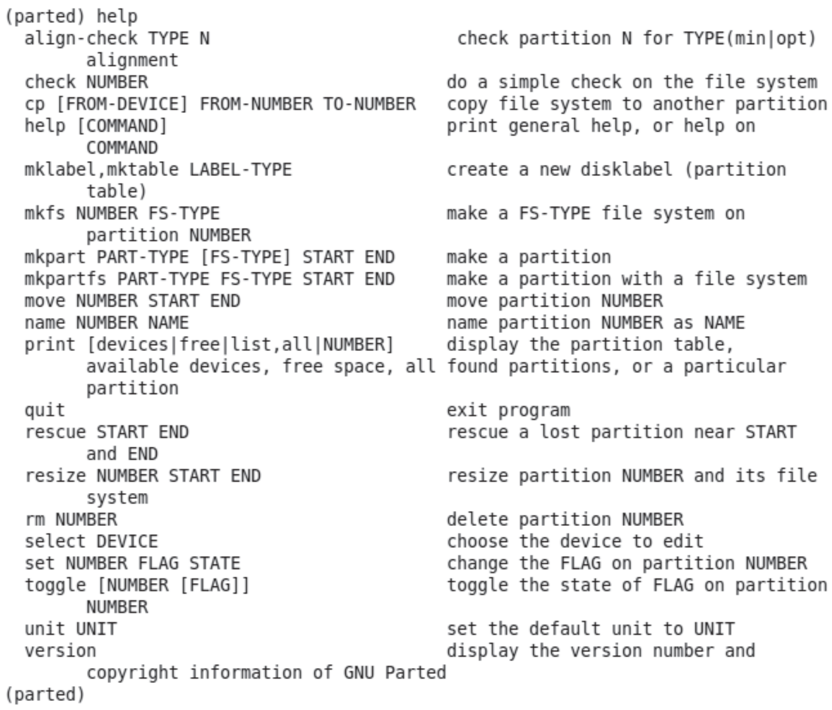
As you can see in Figure 6-2, when **parted** is run, it opens its own command line prompt. Enter **help** for a list of available commands.

There are a wide variety of commands are available at the **parted** interface. When compared to **fdisk**, **parted** can do more in some ways; it can even be used to format and resize partitions. Unfortunately, the format functionality is limited and does not allow you to create or resize ext3 or ext4 partitions, at least for RHEL 6. In fact, an attempt to resize a partition with **parted** leads to the following message:

```
WARNING: you are attempting to use parted to operate on (resize) a file system.
parted's file system manipulation code is not as robust as what you'll find in
dedicated, file-system-specific packages like e2fsprogs.
```

**FIGURE 6-2**

parted command options



```
(parted) help
  align-check TYPE N                        check partition N for TYPE(min|opt)
        alignment
  check NUMBER                             do a simple check on the file system
  cp [FROM-DEVICE] FROM-NUMBER TO-NUMBER   copy file system to another partition
  help [COMMAND]                           print general help, or help on
        COMMAND
  mklabel,mktable LABEL-TYPE               create a new disklabel (partition
        table)
  mkfs NUMBER FS-TYPE                      make a FS-TYPE file system on
        partition NUMBER
  mkpart PART-TYPE [FS-TYPE] START END     make a partition
  mkpartfs PART-TYPE FS-TYPE START END     make a partition with a file system
  move NUMBER START END                    move partition NUMBER
  name NUMBER NAME                         name partition NUMBER as NAME
  print [devices|free|list,all|NUMBER]     display the partition table,
        available devices, free space, all found partitions, or a particular
        partition
  quit                                     exit program
  rescue START END                         rescue a lost partition near START
        and END
  resize NUMBER START END                  resize partition NUMBER and its file
        system
  rm NUMBER                                delete partition NUMBER
  select DEVICE                            choose the device to edit
  set NUMBER FLAG STATE                    change the FLAG on partition NUMBER
  toggle [NUMBER [FLAG]]                   toggle the state of FLAG on partition
        NUMBER
  unit UNIT                                set the default unit to UNIT
  version                                  display the version number and
        copyright information of GNU Parted
(parted)
```

Resizing is a process normally associated with logical volumes. For more information, see the description of the **resize2fs** command later in this chapter.

## Using parted: In a Nutshell

At the **parted** command line prompt, start with the **print** the partition table command. This allows you to review the current entries in the partition table, assuming one exists. Assuming sufficient free space is available, you can then make a new (**mkpart**) partition or even make and format the filesystem (**mkpartfs**). For more information about **parted** command options, use the **help** command; for example, the following command (in bold) provides more information about **mkpart**:

```
(parted) help mkpart
  mkpart PART-TYPE [FS-TYPE] START END     make a partition

         PART-TYPE is one of: primary, logical, extended
         FS-TYPE is one of: ext3, ext2, fat32, fat16, hfsx,
         hfs+, hfs, jfs, linux-swap,ntfs, reiserfs, hp-ufs,
         sun-ufs, xfs, apfs2, apfs1, asfs, amufs5, amufs4,
         amufs3, amufs2, amufs1, amufs0, amufs, affs7, affs6,
         affs5, affs4, affs3, affs2, affs1, affs0
         START and END are disk locations, such as 4GB or 10%.
         Negative values count from the end of the disk.
         For example, -1s specifies exactly the last sector.

         mkpart makes a partition without creating a new
         file system on the partition.
         FS-TYPE may be specified to set an appropriate
         partition ID.
```

If that's too much information, just run the command. You'll be prompted for the necessary information. Remember that drives can contain up to four primary partitions, corresponding to numbers 1 through 4. One of the primary partitions can be redesignated as an extended partition. The remaining partitions are logical partitions, numbered 5 and above.

## Using parted: A New PC (or Hard Drive) with No Partitions

The first step with any truly new hard drive is to create a new partition table. For example, after I add a new hard drive to my virtual RHEL system, just about any command I run in **parted** leads to the following message:

```
Error: /dev/sdb - unrecognised disk label.
```

Before I can do anything else with this drive, I need to create a label. As shown from the list of available commands, I can do so with the **mklabel** command. As strange as it sounds, the default label to be used for Linux is **msdos**; here are the commands I run:

```
(parted) mklabel
New disk label type? msdos
```

## Using parted: Create a New Partition

Now you can create a new partition in **parted**, with the **mkpart** command. Naturally, if an extended partition already exists, the only type available will be a logical partition.

```
(parted) mkpart
Partition type? primary/extended? primary
File system type? [ext2]? ext4
Start? 1MB
End? 500MB
```

For **parted**, I use 1MB as some space has to be reserved for the MBR. While only 512 bytes are required for the MBR, a 1MB entry avoids an error message. Now review the results with the **print** command:

```
(parted) print

Disk /dev/sdb: 10.7GBSector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End    Size   Type     File system  Flags
 1      1049kB  500MB  499MB  primary  ext4
```

If this is the first partition you've created, the filesystem type is empty. Unfortunately, **parted** does not work perfectly; even if you've set up an ext4 label, you still need to format it with the **mkfs.ext4** command discussed later in this chapter. But as long as there are no errors, the partition table is already written to disk. But for the purpose of this chapter, don't exit from **parted** just yet.

on the job

***The GUI parted tools (GParted, QTParted) do support formatting to a wider variety of filesystem formats, even though they're just "front ends" to parted. They*** *might* ***be available from third-party repositories such as those described in Chapter 7.***

## Using parted: Delete a Partition

It's easy to delete a partition in **parted**. All you need to do from the **(parted)** prompt is use the **rm** command to delete the target partition, by number.

Of course, before deleting any partition, you should:

- Save any data you need from that partition.
- Unmount the partition.
- Make sure it isn't configured in /etc/fstab, so Linux doesn't try to mount it the next time you boot.
- After starting **parted**, run the **print** command to identify the partition you want to delete, as well as its ID number.

For example, to delete partition /dev/sdb10 from the **(parted)** prompt, run the following command:

```
(parted) rm 10
```

## Using parted: Create a Swap Partition

Now let's repeat the process to create a swap partition. If necessary, delete the previously created partition to make room. Make the start of the new partition 1MB after the end of the previous partition. You can still use the same commands, just substitute the **linux-swap** file system type as appropriate:

```
(parted) mkpart
Partition type? primary/extended? primary
File system type? [ext2]? linux-swap
Start? 501MB
End? 1000MB
```

Now review the result with the **print** command:

```
(parted) print

Model: ATA QEMU HARDDISK (scsi)
Disk /dev/sdb: 1049MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  500MB   499MB   primary  ext4
 2      501MB   1000MB  500MB   primary
```

Now exit from **parted**. Additional work is required to implement these changes. The commands that follow, **mkswap**, **swapon**, and **mkfs.ext4**, are covered later in this chapter.

```
(parted) quit

# mkswap /dev/sdb2
# swapon /dev/sdb2
```

Now you can format the new regular Linux partition with the following command:

```
# mkfs.ext4 /dev/sdb1
```

## Using parted: Set Up a Different Partition Type

When a partition is created in parted, you can reset its purpose with the **set** command. If the partitions are still available on that second hard drive (or any other existing hard drive with unused partitions, open it with the **parted** command. For example, the following command opens up that second hard drive:

```
# parted /dev/sdb
```

Run the **print** command. The flags column for existing partitions should be empty. Now you'll set that flag with the **set** command. From the commands shown here, the flags are set to use that first partition of the second drive as an LVM partition:

```
(parted) set
Partition number? 1
Flag to Invert? lvm
New state? [on]/off on
```

Now review the result with the **print** command:

```
(parted) print

Model: ATA QEMU HARDDISK (scsi)
Disk /dev/sdb: 1049MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End    Size   Type     File system  Flags
 1      1049kB  500MB  499MB  primary  ext4         lvm
```

Similar steps can be used to configure a partition or a component of a RAID array. It's also a flag; just substitute **raid** for **lvm** in response to the Flag to Invert prompt just shown. If you're following along with a RHEL 6 system, first confirm the result. Exit from **parted**, and run the following commands:

```
# parted /dev/sdb print
# fdisk -l /dev/sdb
```

You'll see the **lvm** flag as shown previously from the **parted** command; you'll see the following confirmation in the output to the **fdisk** command:

```
  Device   Boot  Start    End    Blocks    Id  System
/dev/sda1            1    263    487424    fd  Linux LVM
```

If you've set up the baseline virtual system described in Chapter 2, this is an excellent opportunity to set up partitions as components of LVM volumes. Now that you have the tools, it does not matter whether you use **fdisk** or **parted** for the purpose. You can choose to use all free space. Just be sure to create a partition on more than one hard disk for this purpose, to help illustrate the power of logical volumes.

## Graphical Options

As suggested earlier, excellent graphical front ends are available for disk partitions. The GParted and QtParted options are based on **parted** and are designed for the GNOME and KDE desktop environments, respectively. As they are not available from the Red Hat Network, they are not supported by Red Hat and therefore won't be available for any Red Hat exams.

One graphical option available for RHEL 6 is known simply as Disk Utility, available from the gnome-disk-utility package. Once appropriate packages are installed, you can open Disk Utility from the command line interface with the **palimpsest** command.

The Disk Utility screen shown in Figure 6-3 reveals an application that's far from perfect. The screen depicts the baseline virtual machine created in Chapter 2; it lists the virtual hard drive, device /dev/vda, as if it were a peripheral hard drive. Fortunately, that is a trivial error; the functionality of the tool is not affected. It is still a very capable tool that can organize your drives or destroy all of your data in a number of ways.

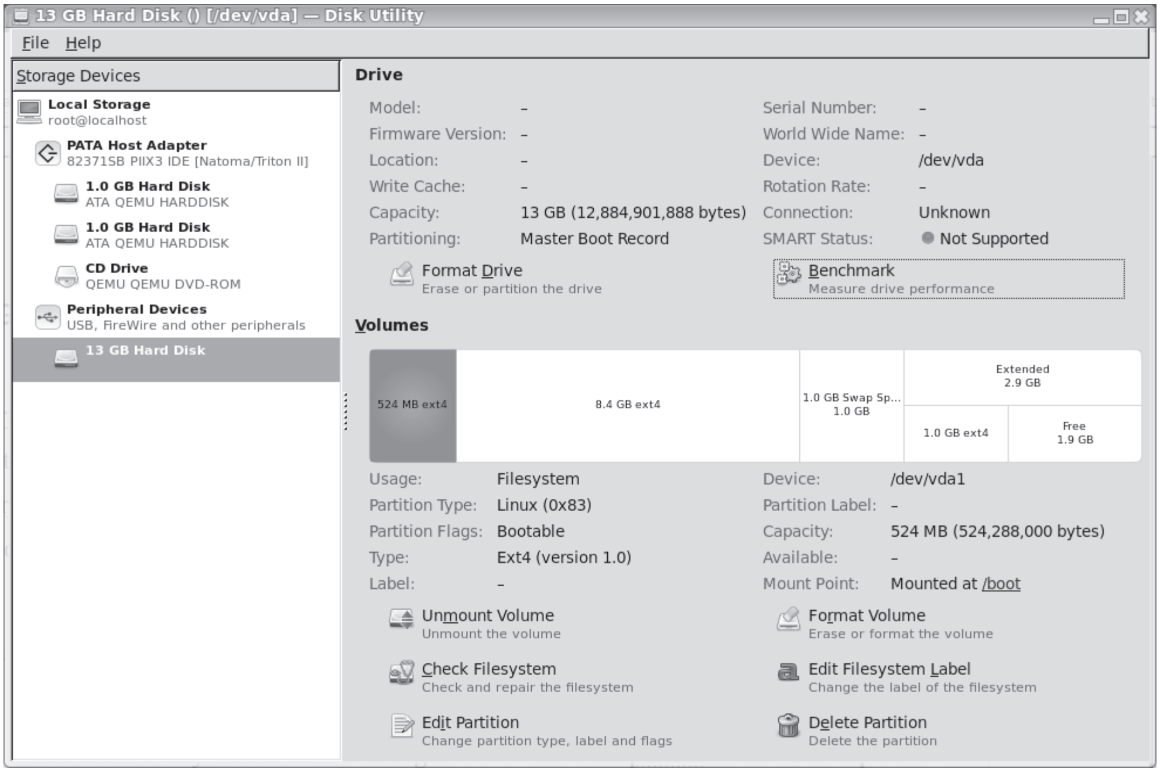
The functionality includes the following clickable options:

- **Format Drive** Supports changes to the entire drive.
- **Benchmark** Allows measurements of read and write performance.

- **Unmount Volume** Front end to the **umount** command.
- **Format Volume** Front end to the **mkfs** command for a number of filesystem formats.
- **Check Filesystem** Front end to the **fsck** command.
- **Edit Filesystem Label** Front end to the **e2label** command; labels were commonly used on RHEL 5.
- **Edit Partition** Front end to **fdisk**'s Change A Partition's System ID command for different partition types such as Linux swap and Linux LVM.
- **Delete Partition** Front end to the **fdisk** functionality to delete a partition.
- **Create Partition** Front end to the **fdisk** functionality to create a new partition.

Not all of these options appear in Figure 6-3; for example, the Create Partition option does not appear unless you've selected a "free" area of the target hard drive. In addition, you may note that the functionality of the Disk Utility goes beyond mere partitioning.

**FIGURE 6-3** The Disk Utility



Model: –
Firmware Version: –
Location: –
Write Cache: –
Capacity: 13 GB (12,884,901,888 bytes)
Partitioning: Master Boot Record
Serial Number: –
World Wide Name: –
Device: /dev/vda
Rotation Rate: –
Connection: Unknown
SMART Status: ● Not Supported

Usage: Filesystem
Partition Type: Linux (0x83)
Partition Flags: Bootable
Type: Ext4 (version 1.0)
Label: –
Device: /dev/vda1
Partition Label: –
Capacity: 524 MB (524,288,000 bytes)
Available: –
Mount Point: Mounted at /boot

## EXERCISE 6-1

### Work with fdisk and parted

In this exercise, you'll work with both the **fdisk** and **parted** utilities. It assumes a new empty drive is available. That does not require additional expense, as a virtual machine–based drive is acceptable for this purpose. For the purpose of this exercise, **fdisk** and **parted** will be used on drives /dev/sda and /dev/sdb, respectively. Feel free to substitute accordingly. Be aware, you'll save the results of this work for exercises that follow later in this chapter.

1. Run the **fdisk -l /dev/sda** command to review the current status of the /dev/sda drive.
2. Open disk /dev/sda with the **fdisk /dev/sda** command.
3. As suggested by the start message, switch off DOS-compatible mode and change display units to sectors with the **c** and **u** commands.
4. Run the **p** command to display any previously configured partitions.
5. Create a new partition with the **n** command. If there are primary partitions available, create one with the **p** command. If options for primary partition numbers are presented, select the first available.
6. When presented with a request similar to the following to specify the first sector of the new partition, specify something else. First try to specify sector 1 to see the response. Then try a sector somewhere after the default. For the purpose of this example, I specify 10,000 here:

```
First sector (2048-2047999, default 2048): 10000
```

7. When presented with a request similar to the following to specify the last sector of the new partition, enter a number somewhere in the middle of the listed range. For the purpose of this example, I specify 1,000,000 here:

```
Last sector, +sectors or +size{K,M,G} (10000-2047999, default 2047999):
1000000
```

8. Run the **p** command again to review the result. Run the **w** command to write the result to disk.
9. Review the result on the /dev/sda disk with the **parted /dev/sda print** command.
10. Open the other available free disk /dev/sdb with the **parted /dev/sdb** command.

11. From the (parted) prompt, Run the **print** command to review the current status of partitions. If you see an "unrecognized disk label" error message, run the **mklabel msdos** command and run the **print** command again.
12. Create a new partition with the **mkpart** command. Follow the prompts. It does not matter whether the partition is primary or logical (avoid an extended partition for now). Do not enter a filesystem type; start the partition at **100M** (100MB) and end it at **600M** (600MB). Run the **print** command to confirm the new partition, and identify the partition number.
13. Run the **quit** command to exit from parted.
14. Run the **fdisk -l /dev/sdb** command to review the result. Do you see a problem?
15. Run the **fdisk /dev/sdb** command to open that drive. Run the **p** command. What do you see? Is it familiar?
16. Enter the **c** and **u** commands again. Repeat the **p** command. Do you see the same problem?
17. Exit from **fdisk** with the **q** command.

## CERTIFICATION OBJECTIVE 6.02

## Filesystem Formats

The number of filesystem types may exceed the number of operating systems. While RHEL can work with many of these formats, the default is ext4. While many users enable other filesystems such as ReiserFS, Red Hat may not support them.

Linux supports a rich variety of filesystems. Linux filesystems can be somewhat inaccurately divided into two categories: "standard" formatting and journaling. While this is an oversimplification, it suffices to describe the filesystems important to Linux. To me, a standard filesystem is an older Linux filesystem that does not log changes.

on the job

***There are a large number of filesystem types well described in the Filesystems HOWTO at www.tldp.org. Strictly speaking, there is no "standard" Linux filesystem.***

The filesystems described in this book are just a small list of those that can be configured on an RHEL system. While Red Hat supports a limited list, the Linux kernel makes it possible to set up more. A list of the filesystems supported by a kernel is available in its configuration file in the /boot directory, in the **config-`uname -r`** file, where **uname -r** is a command that, with the backquotes, substitutes the version number of the currently loaded kernel.

## Standard Formatting Filesystems

Linux is a clone of Unix. The Linux filesystems were developed to mimic the functionality of Unix filesystems available at the time. The first Linux operating systems used the Extended Filesystem (ext). Until the past few years, Red Hat Linux operating systems formatted their partitions by default to the Second Extended Filesystem (ext2). For RHEL 5, the default was the Third Extended Filesystem (ext3). The new default for RHEL 6 is the Fourth Extended Filesystem (ext4). Both ext3 and ext4 are journaling filesystems, described in the next section.

Given the growth in filesystem sizes, journaling filesystems are more resilient to failure. So to some extent, the non-journaling filesystems described in Table 6-2 are legacy filesystems. Of course, filesystems such as ISO 9660 and swap are still in common use.

## Journaling Filesystems

As hard disks and partitions grow in size, Linux distributions use filesystems with journaling features. Journaling filesystems have two main advantages. First, such as filesystem is faster for Linux to check during the boot process. Second, if a crash occurs, a journaling filesystem has a log (also known as a journal) that can be used to restore the metadata for the files on the relevant partition.

For RHEL 5, the default RHEL filesystem is ext3; for RHEL 6, it's ext4. Those aren't the only journaling filesystem options available, however. I list a few of the options commonly used for RHEL in Table 6-3. From this list, Red Hat officially supports only ext3 and ext4.

**TABLE 6-2** Some Linux Standard Filesystem Types

| Filesystem Type | Description |
|---|---|
| ext | The first Linux filesystem, used only on early versions of that operating system. |
| ext2 (Second Extended) | The foundation for ext3, the default filesystem for RHEL 5. The ext3 filesystem is essentially ext2 with journaling. |
| swap | The Linux swap filesystem is associated with dedicated swap partitions. You've probably created at least one swap partition when you installed RHEL. |
| MS-DOS and VFAT | These filesystems allow you to read MS-DOS-formatted filesystems. MS-DOS lets you read pre–Windows 95 partitions, or regular Windows partitions within the limits of short filenames. VFAT lets you read Windows 9x/NT/2000/XP/Vista/7 partitions formatted to the FAT16 or FAT32 filesystems. |
| ISO 9660 | The standard filesystem for CD-ROMs. It is also known as the High Sierra File System, or HSFS, on other Unix systems. |
| /proc | A Linux *virtual* filesystem. Virtual means that it doesn't occupy real disk space. Instead, files are created as needed. Used to provide information on kernel configuration and device status. |
| /dev/pts | The Linux implementation of the Open Group's Unix98 PTY support. |

**TABLE 6-3** Journaling Filesystems

| Filesystem Type | Description |
|---|---|
| ext3 | The default filesystem for RHEL 5. |
| ext4 | The default filesystem for RHEL 6. |
| JFS | IBM's journaled filesystem, commonly used on IBM enterprise servers. |
| ReiserFS | The Reiser File System is resizable and supports fast journaling. It's more efficient when most of the files are very small and very large. It's based on the concept of "balanced trees." It is no longer supported by RHEL, or even by its former main proponent, SUSE. For more information, see www.namesys.com. |
| xfs | Developed by Silicon Graphics as a journaling filesystem, it supports very large files; as of this writing, xfs files are limited to $9 \times 10^{18}$ bytes. Do not confuse this filesystem with the X Font Server; both use the same acronym. |
| NTFS | The current Microsoft Windows filesystem |

The Red Hat move to ext4 is a testament to its use as a server operating system. One improvement with ext4 means that filesystems can be as large as 1 exabyte (EB). The former maximum filesystem size with ext3 was just 16 terabytes (TB). The ext4 filesystem reduces fragmentation, guarantees space for files, supports faster checks, and more. It even supports file timestamps in nanoseconds. As ext4 has been a part of the Linux kernel since 2008, it is proven technology. Given its speed and reliability, Red Hat even uses ext4 as the default filesystem for partitions dedicated to the /boot directory.

## Filesystem Format Commands

There are several commands that can help you create a Linux filesystem. They're all based on the **mkfs** command, which works as a front end to filesystem-specific commands such as **mkfs.ext2**, **mkfs.ext3**, and **mkfs.ext4**.

If you want to reformat an existing partition, logical volume, or RAID array, take the following precautions:

- Back up any existing data on the partition.
- Unmount the partition.

There are two ways to apply formatting on a volume. (As noted earlier in this chapter, a volume is a generic name that can describe a partition, a RAID array, or a logical volume.) For example, if you've just created a partition on /dev/sdb5, you can format it to the ext4 filesystem using one of the following commands:

```
# mkfs -t ext4 /dev/sdb5
# mke2fs -t ext4 /dev/sdb5
# mkfs.ext4 /dev/sdb5
```

You can format partitions, logical volumes, and RAID arrays to other filesystems. The options available in RHEL 6 include:

- **mkfs.cramfs** creates a compressed ROM filesystem.
- **mkfs.ext2** formats a volume to the ext2 filesystem.
- **mkfs.ext3** formats a volume to the RHEL 5 default ext3 filesystem.
- **mkfs.ext4** formats a volume to the RHEL 6 default ext4 filesystem.
- **mkfs.msdos** (or **mkfs.vfat** or **mkdosfs**) formats a partition to the Microsoft-compatible VFAT filesystem; it does not create bootable filesystems.

(The inode numbers for all three files are the same; in other words, they are three different names for the same command.)

- **mkfs.xfs** formats a volume to the XFS filesystem developed by the former Silicon Graphics.
- **mkswap** formats a volume to the Linux swap filesystem.

These commands assume that you've configured an appropriate partition in the first place; for example, before the **mkswap** command can be properly applied to a partition, the Linux swap partition ID type must be configured for that partition. If you've created a RAID array or logical volume, as described later in this chapter, similar rules apply.

on the job

***One advantage of some rebuild distributions is the availability of useful packages not supported by or available from Red Hat. For example, CentOS 6 includes the ntfsprogs package, which supports the mounting of NTFS partitions.***

## Swap Volumes

While Linux can use swap files, the swap space that's used essentially as overflow for RAM is generally configured in properly formatted partitions or logical volumes. They're generally not configured form RAID arrays, as redundancy for RAM is generally not useful. To see the swap space currently configured, run the **cat /proc/swaps** command.

As suggested in the previous section, swap volumes are formatted with the **mkswap** command. But that's not enough. First, to test the new swap volume, it must be activated with the **swapon** command. If the new swap volume is recognized, you'll see it in both the /proc/swaps file and the output to the **top** command. Second, you'll need to make sure to configure the new swap volume in the /etc/fstab file, as described later in this chapter.

## Filesystem Check Commands

The **fsck** command analyzes the specified filesystem and performs repairs as required. Assume, for example, you're having problems with files in the /var directory, which happens to be mounted on /dev/sda7. If you want to run **fsck**, unmount that filesystem first. In some cases, you may need to go into single-user mode with the

**init 1** command before you can unmount a filesystem. To unmount, analyze, and then remount the filesystem noted in this section, run the following commands:

```
# umount /var
# fsck -t ext4 /dev/sda7
# mount /dev/sda7 /var
```

The **fsck** command also serves as a "front end," depending on the filesystem format. For example, if you're formatting an ext2, ext3, or ext4 filesystem, **fsck** by itself automatically calls the **e2fsck** command. In fact, the **fsck.ext2**, **fsck.ext3**, **fsck.ext4**, and **e2fsck** files are all different names for the same command! They have the same inode number. You can confirm this by applying the **ls -i** command to all four files, which are part of the /sbin directory.

## Filesystem Conversions

If you're upgrading between versions of Linux, you may also want to upgrade filesystems. It wasn't that long ago that most Linux systems were configured to the ext2 filesystem. In fact, for RHEL 5, ext3 was considered inefficient by many for filesystems such as the /boot directory, so many administrators formatted the associated partition to ext2.
Fortunately, if you have a partition formatted to ext2, it's easy to add the journal associated with ext3. And then it's a straightforward command to add the features associated with ext4 formatting. These commands require a temporary remounting of the filesystem in read-only mode. Alternatively, you could just unmount that directory. For example, if the filesystem in question is mounted on the /dev/vda1 partition, you'd run the following commands to convert from ext2 to ext3, where the **tune2fs -j** command adds a journal.

```
# mount -o remount,ro /dev/vda1# tune2fs -j /dev/vda1# mount -o
remount,rw /dev/vda1
```

Of course, you'd then have to make appropriate changes to the /etc/fstab configuration file, namely changing the filesystem format column from ext2 to ext3. If necessary, you could convert back with the **tune2fs ^O has_journal** command. But any move from ext2 or ext3 to ext4 is one-way. You can't convert back to ext3 or ext2. If you're ready to make the move to ext4, run the following command:

```
# tune2fs -O extent,uninit_bg,dir_index /dev/vda1
```

In fact, if you try to disable some components of the ext4 filesystem, it may break features such as the **blkid** command described later in this chapter.

If you're converting the volume associated with the top-level root directory (/), it's not possible to even remount that directory in read-only mode. Thus, the process with that directory may require a detour through the rescue mode environment described in Chapter 5.

To confirm the current settings associated with a filesystem, run the following command:

```
# dumpe2fs /dev/vda1 | grep "Filesystem features"
```

You should see the features just described in the output associated with the features of the specified filesystem volume.

## EXERCISE 6-2

### Format, Check, and Mount Different Filesystems

In this exercise, you'll work with the file format and checking commands **mkfs** and **fsck**, and review the results with the **mount** command. This exercise assumes you've completed Exercise 6-1, or at least have unmounted Linux partitions with no data.

1. Review the current status of partitions on the drives discussed in Exercise 6-1 with the **parted /dev/sda print** and **fdisk -l /dev/sdb** commands.
2. Format the partition created the first drive with the **mkfs.ext2 /dev/sda1** command. Review the current status of the volume with the **dumpe2fs /dev/sda1 | grep formats** command. What features do you see in the output? Save the output, temporarily. One way to do so is open a new command line console. Check the system with the **fsck.ext2 /dev/sda1** command.
3. Mount the newly formatted partition with **mount /dev/sda1 /mnt** command. Review the output with the **mount** command, by itself. If the mount and format worked, you'll see the following output:

   ```
   /dev/sda1 on /mnt type ext2 (rw)
   ```

4. Unmount the formatted partition with the **umount /mnt** command.
5. Run the **tune2fs -j /dev/sda1** command and rerun the **dumpe2fs** command from the previous step. What's the difference between the output now, and the output when the partition was formatted to the ext2 filesystem?

6. Repeat Steps 3 and 4. What's the difference in the output to the **mount** command?
7. Run the **tune2fs -O extent,uninit_bg,dir_index /dev/sda1** command. Do you see a message in the output?
8. Apply the **fsck.ext4** command on the partition. What do you see?
9. Repeat Step 6.
10. Now on the other partition created in Exercise 6-1, apply the **mkfs.ext4 /dev /sdb1** command.
11. Mount the newly formatted partition on the **/net** directory, and run the **mount** command by itself. Can you confirm the format of the /dev/sdb1 partition?

## CERTIFICATION OBJECTIVE 6.03

## Basic Linux Filesystems and Directories

Everything in Linux can be reduced to a file. Partitions are associated with *filesystem device nodes* such as /dev/sda1. Hardware components are associated with node files such as /dev/dvd. Detected devices are documented as files in the /proc directory. The Filesystem Hierarchy Standard (FHS) is the official way to organize files in Unix and Linux directories. As with the other sections, this introduction provides only the most basic overview of the FHS. More information is available from the official FHS home page at www.pathname.com/fhs.

### Separate Linux Filesystems

There are several major directories are associated with all modern Unix/Linux operating systems. Files, drivers, kernels, logs, programs, utilities, and more are organized in these directories. They way these components are organized on storage media is known as a filesystem. It's based on the way the filesystem is formatted, and the directory where that filesystem is mounted. The FHS makes it easier for users of other Unix-based operating systems to understand the basics of Linux.

Every FHS starts with the top-level root directory, also known by its symbol, the single forward slash (/). All of the other directories shown in Table 6-4 are subdirectories of the root directory. Unless mounted separately, you can also find their files on the same partition as the root directory. You may not see some of the directories shown in the table if associated packages have not been installed. Not all directories shown are officially part of the FHS. More important, not all listed directories can or should be mounted separately.

Mounted directories are often known as *volumes*, which can span multiple partitions. However, while the root directory (/) is the top-level directory in the FHS, the root user's home directory (/root) is just a subdirectory.

on the job

***In Linux, the word filesystem has several different meanings. For example, a filesystem can refer to the FHS, an individual volume, or a format such as ext3. A filesystem device node such as /dev/sda1 represents the partition on which a directory can be mounted.***

## Directories That Can Be Mounted Separately

If space is available, several directories listed in Table 6-4 are excellent candidates to be mounted separately. As discussed in Chapter 1, it's typical to mount directories such as /, /boot, /home, /opt, /srv, /tmp, and /var on separate volumes. Sometimes, it makes sense to mount lower-level subdirectories on separate volumes, such as /var/ftp for an FTP server or /var/www for a Web server.

But first, several directories should always be maintained as part of the top-level root directory filesystem. These directories include: /bin, /dev, /etc, /lib, /root, /sbin, and /selinux. Files within these directories are essential to the smooth operation of Linux as an operating system. While the same argument can be made for the /boot directory, it is a special case. The storage of the Linux kernel, Initial RAM Disk, and bootloader files in this directory can help protect the core of the operating system when there are other problems.

This ignores directories with virtual filesystems, including /proc and /sys. Files in these directories are filled only during the boot process and disappear when a system is shut down. As there's nothing to store from these directories, there's no reason to mount them separately. Some directories listed in Table 6-4 are designed for use only as mount points. In other words, they should normally be empty. If you store files on those directories, they won't be accessible if, say, a network share is mounted on them. Typical network mount points include the /media, /mnt, /net, and /smb directories.

**TABLE 6-4** Basic Filesystem Hierarchy Standard Directories

| Directory | Description |
|---|---|
| / | The root directory, the top-level directory in the FHS. All other directories are subdirectories of root, which is always mounted on some volume. |
| /bin | Essential command line utilities. Should not be mounted separately; otherwise, it could be difficult to get to these utilities when using a rescue disk. |
| /boot | Includes Linux startup files, including the Linux kernel. The default, 500MB, is usually sufficient for a typical modular kernel and additional kernels that you might install during the RHCE or RHCSA exam. |
| /dev | Hardware and software device drivers for everything from floppy drives to terminals. Do not mount this directory on a separate volume. |
| /etc | Most basic configuration files. Do not mount this directory on a separate volume. |
| /home | Home directories for almost every user. |
| /lib | Program libraries for the kernel and various command line utilities. Do not mount this directory on a separate volume. |
| /media | The mount point for removable media, including floppy drives, DVDs, and Zip disks. |
| /misc | The standard mount point for local directories mounted via the automounter. |
| /mnt | A legacy mount point; formerly used for removable media. |
| /net | The standard mount point for network directories mounted via the automounter. |
| /opt | Common location for third-party application files. |
| /proc | Currently running kernel-related processes, including device assignments such as IRQ ports, I/O addresses, and DMA channels, as well as kernel configuration settings such as IP forwarding. As a virtual filesystem, Linux automatically configures it as a separate filesystem in RAM. |
| /root | The home directory of the root user. Do not mount this directory on a separate volume. |
| /sbin | System administration commands. Don't mount this directory separately. |
| /selinux | Currently configured settings associated with Security Enhanced Linux. Do not mount this directory on a separate volume. |
| /smb | The standard mount point for remote shared Microsoft network directories mounted via the automounter. |
| /srv | Commonly used by various network servers on non–Red Hat distributions. |
| /tftpboot | Included if the TFTP server is installed. |
| /tmp | Temporary files. By default, Red Hat Enterprise Linux deletes all files in this directory periodically. |
| /usr | Small programs accessible to all users. Includes many system administration commands and utilities. |
| /var | Variable data, including log files and printer spools. |

CERTIFICATION OBJECTIVE 6.04

## Logical Volume Management (LVM)

Logical Volume Management (LVM, also known as the Logical Volume Manager) can allow you to manage the space allocated to appropriate directories. As an example, assume the /var and /home directories are configured on separate logical volumes. If extra space is available on the volume associated with the /var directory, you can reallocate space to the volume associated with the /home directory.

Alternatively, if you are managing a server on a growing network, new users will be common. Periodically, more room may be needed on a volume associated with the /home directory. With LVM, you can add a new physical disk and allocate its storage capacity to an existing /home directory volume.

on the job

***While LVM can be an important tool to manage the space available to different volumes, it does not provide redundancy. However, you can use LVM in concert with a RAID array.***

### Definitions in LVM

To work with LVM, you need to understand how partitions configured for that purpose are used. First, with the **fdisk** and **parted** utilities, you need to create partitions configured to the LVM partition type. The commands within those utilities were described earlier in this chapter.

Once those partitions are available, they need to be set up as physical volumes (PVs). That process sets up the physical partitions into manageable chunks known as physical extents (PEs). With the right commands, you can then convert those PEs to logical extents (LEs). Those LEs can be organized into logical volumes (LVs). You can then create volume groups (VGs) from part or all of an LV. That VG can then be formatted and mounted on an appropriate directory. For those who are new to LVM, it may be important break out each definition:

- **Physical volume (PV)** A PV is a partition, configured to the LVM partition type.
- **Physical extent (PE)** A PE is a small uniform segment of disk space. PVs are split into PEs.

- **Logical extent (LE)** Every LE is associated with a PE and can be combined into a volume group.
- **Volume group (VG)** A VG is a bunch of LEs, grouped together.
- **Logical volume (LV)** An LV is a part of a VG, which can be formatted and then mounted on the directory of your choice.

You'll see this broken down in the following sections. But in essence, to create an LV system, you need to create a new PV, using a command such as **pvcreate**, assign the space from one or more PVs to a VG with a command such as **vgcreate**, and allocate the space from some part of available VGs to an LV with a command such as **lvcreate**.

To add space to an existing LVM system, you need to add free space from an existing VG with a command such as **lvextend**. If you don't have any existing VG space, you'll need to add to it with unassigned PV space with a command such as **vgextend**. If all of your PVs are taken, you may need to create a new PV from an unassigned partition or hard drive with the **pvcreate** command.

Whenever you change an active PV, LV, or VG, unmount the mounted LV first. If it's an essential filesystem such as the top-level root (/) directory, you may need to boot from the RHEL 6 installation CD/DVD into rescue mode.

## Create a Physical Volume

The first step is to start with a physical partition. Based on the discussion earlier in this chapter, you should be able to set up partitions set up to match the Linux LVM identifier. Then, to set up a new PV on a properly configured partition, such as /dev/sda1, apply the **pvcreate** command to that partition:

```
# pvcreate /dev/sda1
```

If there is more than one partition to be configured as a PV, the associated device files can all be listed in the same command:

```
# pvcreate /dev/sda1 /dev/sda2 /dev/sdb1 /dev/sdb2
```

## Create a Volume Group

From one or more PVs, you can create a volume group (VG). In the following command, substitute the name of your choice for *volumegroup*:

```
# vgcreate volumegroup /dev/sda1 /dev/sda2
```

You can include additional PVs in any VG. Assume there are existing PVs based on /dev/sdb1 and /dev/sdb2 partitions, you can add to the *volumegroup* VG with the following command:

```
# vgextend volumegroup /dev/sdb1 /dev/sdb2
```

## Create a Logical Volume

However, a new VG isn't enough since you can't format or mount a filesystem on it. So you need to create a logical volume (LV) for this purpose. The following command creates an LV. You can add as many chunks of disk space, in PEs, as you need.

```
# lvcreate -l number_of_PEs volumegroup -n logvol
```

This creates a device named /dev/*volumegroup*/*logvol*. You can format this device as if it were a regular disk partition, and then mount a directory on that new logical volume.

But this isn't useful if you don't know how much space is associated with each PE. You could use trial and error, using the **df** command to check the size of the volume after mounting a directory on it. Alternatively, you can use the **-L** switch to set a size in MB. For example, the following command creates an LV named flex of 200MB:

```
# lvcreate -L 200M volumegroup -n flex
```

## Make Use of a Logical Volume

But that's not the last step. You may not get full credit unless the directory gets mounted on the logical volume when the system is rebooted. This process is described later in this chapter in the discussion of the /etc/fstab configuration file.

## More LVM Commands

There are a wide variety of LVM commands related to PVs, LVs, and VGs. Generally, they are **pv***, **lv***, and **vg*** in the /usr/sbin directory. Physical volume commands include those listed in Table 6-5.

As you assign PVs to VGs to LVs, you may need commands to control and configure them. Table 6-6 includes an overview of most related volume group commands.

**TABLE 6-5**

Physical Volume Management Commands

| Physical Volume Command | Description |
|---|---|
| pvchange | Changes attributes of a PV: the **pvchange -x n /dev/sda10** command disables the use of PEs from the /dev/sda10 partition. |
| pvck | Checks the integrity of a physical volume. |
| pvcreate | Initializes a disk or partition as a PV; the partition should be flagged with the LVM file type. |
| pvdisplay | Displays currently configured PVs. |
| pvmove | Moves PVs in a VG from the specified partition to free locations on other partitions; prerequisite to disabling a PE. One example: **pvmove /dev/sda10**. |
| pvremove | Removes a given PV from a list of recognized volume: for example, **pvremove /dev/sda10**. |
| pvresize | Changes the amount of a partition allocated to a PV. If you've expanded partition /dev/sda10, **pvresize /dev/sda10** takes advantage of the additional space. Alternatively, **pvresize --setphysicalvolumesize 100M /dev/sda10** reduces the amount of PVs taken from that partition to the noted space. |
| pvs | Lists configured PVs and the associated VGs, if so assigned. |
| pvscan | Similar to **pvs**. |

**TABLE 6-6**

Volume Group Commands

| Volume Group Command | Description |
|---|---|
| vgcfgbackup<br>vgcfgrestore | Backs up and restores the configuration files associated with LVM; by default, they're in the /etc/lvm directory. |
| vgchange | Similar to pvchange, allows you to activate or deactivate a VG. For example, vgchange -a y enables all local VGs. |
| vgck | Checks the integrity of a volume group. |
| vgconvert | Supports conversions from LVM1 systems to LVM2: vgconvert -M2 VolGroup00 converts VolGroup00. |
| vgcreate | Creates a VG, from two or more configured PVs: for example, vgcreate vgroup00 /dev/sda10 /dev/sda11 creates vgroup00 from PVs as defined on /dev/sda10 and /dev/sda11. |
| vgdisplay | Displays characteristics of currently configured VGs. |
| vgexport<br>vgimport | Exports and imports unused VGs from those available for LVs; the vgexport -a command exports all inactive VGs. |

**TABLE 6-6**

Volume Group Commands (*continued*)

| Volume Group Command | Description |
|---|---|
| vgextend | If you've created a new PV: vgextend vgroup00 /dev/sda11 adds the space from /dev/sda11 to vgroup00. |
| vgmerge | If you have an unused VG vgroup01, you can merge it into vgroup00 with the following command: vgmerge vgroup00 vgroup01. |
| vgmknodes | Run this command if you have a problem with VG device files. |
| vgreduce | The vgreduce vgroup00 /dev/sda11 command removes the /dev/sda11 PV from vgroup00, assuming sufficient free space is available. |
| vgremove | The vgremove vgroup00 command removes vgroup00, assuming it is not assigned to any LV. |
| vgrename | Allows renaming of LVs. |
| vgs | Displays basic information on configured VGs. |
| vgscan | Scans and displays basic information on configured VGs. |
| vgsplit | Splits a volume group. |

As you assign PVs to VGs and then subdivide VGs into LVs, you may need commands to control and configure them. Table 6-7 includes an overview of related LVM commands.

**TABLE 6-7**

Logical Volume Commands

| Logical Volume Command | Description |
|---|---|
| lvchange | Similar to pvchange, changes the attributes of an LV: for example, the lvchange -a n vgroup00/lvol00 command disables the use of the LV labeled lvol00. |
| lvconvert | If there are sufficient available PVs, the lvconvert -m1 vgroup00/lvol00 command mirrors the LV. |
| lvcreate | Creates a new LV in an existing VG. For example, lvcreate -l 200 volume01 -n lvol01 creates lvol01 from 200 extents in the VG named volume01. |
| lvdisplay | Displays currently configured LVs. |
| lvextend | Adds space to an LV: the lvextend -L4G /dev/volume01/lvol01 command extends lvol01 to 4GB, assuming space is available. |
| lvreduce | Reduces the size of an LV; if there's data in the reduced area, it is lost. |
| lvremove | Removes an active LV: the lvremove volume01/lvol01 command removes all lvol01 from VG volume01. |

**TABLE 6-7**

Logical Volume Commands (*continued*)

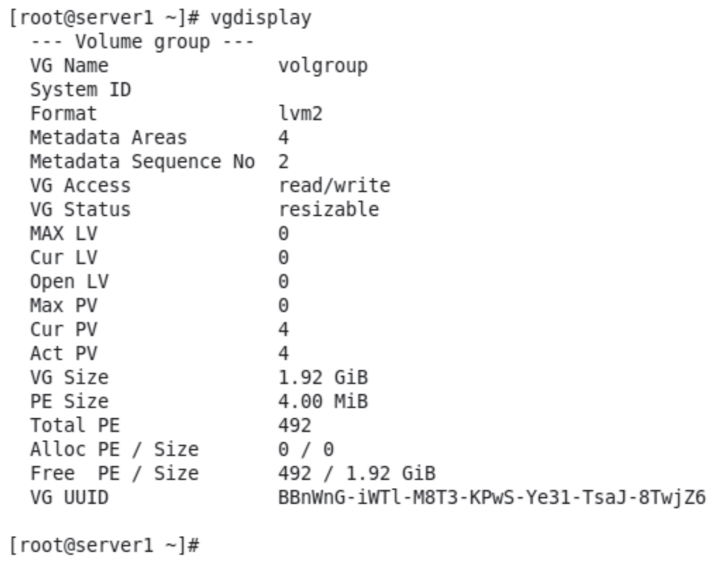
| Logical Volume Command | Description |
|---|---|
| lvrename | Renames an LV. |
| lvresize | Resizes an LV; can be done by -L for size. For example, lvresize -L 4GB volume01/lvol01 changes the size of lvol01 to 4GB. |
| lvs | Lists all configured LVs. |
| lvscan | Scans for all active LVs. |

Here's an example how this works. Try the vgscan command. You can verify configured volume groups (VGs) with the vgdisplay command. For example, Figure 6-4 illustrates the configuration of VG volgroup.

While there are a number of lvm* commands installed, just four of them are active: lvm, lvmconf, lvmdiskscan, and lvmdump. The lvm command moves to an lvm> prompt. It's rather interesting, as the help command at that prompt provides a nearly full list of available LVM commands.

The lvmconf command can modify the default settings in the related configuration file, /etc/lvm/lvm.conf. The lvmdiskscan command scans all available drives for LVM-configured partitions. Finally, the lvmdump command sets up a configuration report in the root administrative user's home directory (/root).

**FIGURE 6-4**



```
[root@server1 ~]# vgdisplay
  --- Volume group ---
  VG Name               volgroup
  System ID
  Format                lvm2
  Metadata Areas        4
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                4
  Act PV                4
  VG Size               1.92 GiB
  PE Size               4.00 MiB
  Total PE              492
  Alloc PE / Size       0 / 0
  Free  PE / Size       492 / 1.92 GiB
  VG UUID               BBnWnG-iWTl-M8T3-KPwS-Ye31-TsaJ-8TwjZ6

[root@server1 ~]#
```

Configuration of a volume group (VG)

Before logical volumes are useful, you need to know how to add another LV. For example, if you've added more users, and they need more room than is available on the /home directory, you may need to add more LVs.

on the job

***Linux can't read files from the /boot directory if it's configured on a logical volume. As those files are essential to the boot process, don't set up a logical volume for that directory.***

## Remove a Logical Volume

The removal of an existing LV is straightforward, with the **lvremove** command. This assumes that any directories previously mounted on LVs have been unmounted. At that point, the basic steps are simple:

1. Save any data in directories that are mounted on the LV.
2. Unmount any directories associated with the LV. Based on the example in the previous section, you would use the following command:

   ```
   # umount /dev/vg_01/lv_01
   ```

3. Apply the **lvremove** command to the LV with a command such as:

   ```
   # lvremove /dev/vg_01/lv_01
   ```

4. You should now have the LEs from this LV free for use in other LVs.

## Resize Logical Volumes

If you need to increase the size of an existing LV, you can add the space from a newly created PV to it. All it takes is appropriate use of the **vgextend** and **lvextend** commands. For example, to add the PEs to the VG associated with a /home directory mounted on a LV, take the following basic steps:

1. Back up any data existing on the /home directory. (This is a standard precaution that isn't necessary if everything goes right. You might even skip this step on Red Hat exams. But do you really want to risk user data in practice?)
2. Unmount the /home directory from the current LV.
3. Extend the VG to include new partitions configured to the appropriate type. For example, to add /dev/sdd1 to the /home VG, run the following command:

   ```
   # vgextend vg_00 /dev/sdd1
   ```

4. Make sure the new partitions are included in the VG with the following command:

   ```
   # vgdisplay vg_00
   ```

5. Now you can extend the space given to the current LV. For example, to extend the LV to 2000MB, run the following command:

   ```
   # lvextend -L 2000M /dev/vg_00/lv_00
   ```

6. The **lvextend** command can increase the space allocated to an LV in KB, MB, GB, or even TB. For example, you could specify a 2000M LV with the following command:

   ```
   # lvextend -L 2G /dev/vg_00/lv_00
   ```

7. Resize the formatted volume with the **resize2fs** command. If you're using the entire extended LV, the command is simple:

   ```
   # resize2fs /dev/vg_00/lv_00
   ```

   To use just part of the new volume, you can specify the amount of space at the end of the command; for example, the following command suggests a format of 1900MB:

   ```
   # resize2fs /dev/vg_00/lv_00 1900M
   ```

8. Alternatively, you can reformat the LV, using commands described earlier, so the filesystem can take full advantage of the new space—and then restore data from the backup. (If you've already successfully resized an LV, *don't* reformat it. It isn't necessary, and would destroy existing data!)

   ```
   # mkfs.ext4 /dev/vg_00/lv_00
   ```

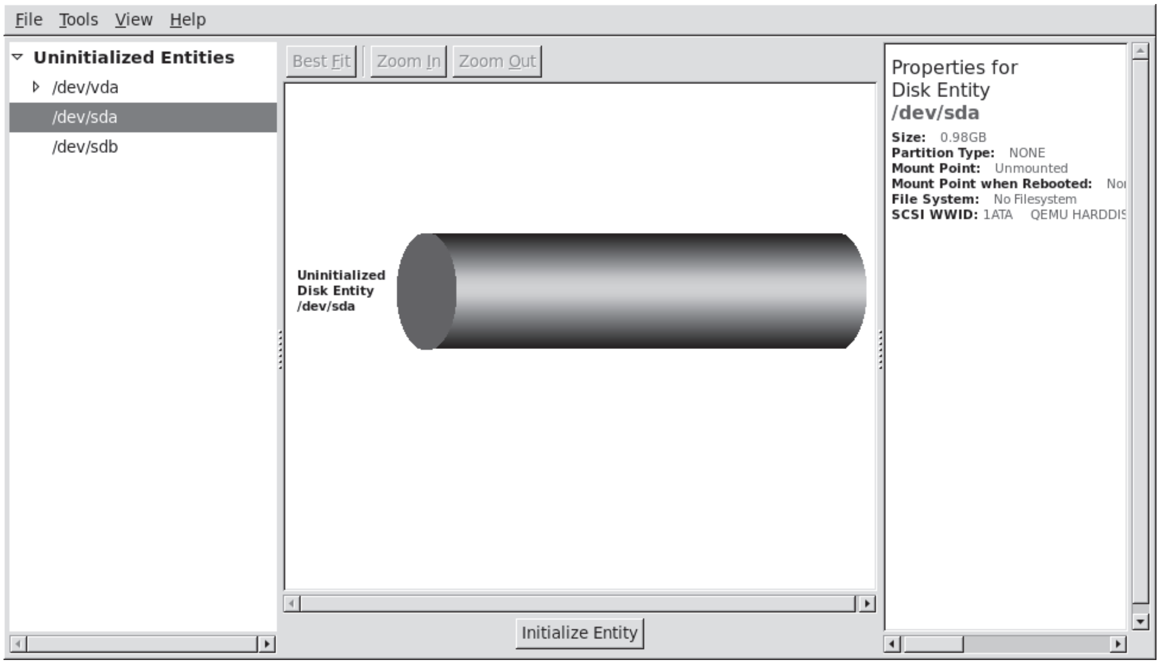
9. In either case, you'd finish the process by remounting the newly resized LV.

   ```
   # mount /dev/vg_00/lv_00 /home
   ```

## The GUI Logical Volume Management Tool

If this is all confusing, you might try the GUI Logical Volume Management tool, as shown in Figure 6-5. In general, command line tools are superior. While command line tools can still do more, the Red Hat Logical Volume Management tool is quite capable. In the GNOME desktop, you can start it by running **system-config-lvm** from a GUI command line. In the following sections, I'll show you how to use it to add, remove, and resize logical volumes.

**FIGURE 6-5** The Logical Volume Management tool



Properties for Disk Entity /dev/sda

**Size:** 0.98GB
**Partition Type:** NONE
**Mount Point:** Unmounted
**Mount Point when Rebooted:** No
**File System:** No Filesystem
**SCSI WWID:** 1ATA QEMU HARDDIS

If needed, you can install the system-config-lvm package with the following command:

```
# yum install system-config-lvm
```

If possible, test the options in these three subsections in one sitting. If you test the commands in the first subsection, retain them for the next subsection and so on. These sections effectively repeat the descriptions shown earlier in this chapter based on command line tools.

## Add an LV

Assume you've added a new hard drive. For the purpose of this section, I'm using the /dev/sda and /dev/sdb drives associated with the baseline virtual machine used for this book. In addition, I've deleted those partitions. That provides up to 2000MB of space

Remember that if the GUI commands in this tool don't work, you can always use the associated regular text commands described earlier. Based on the tool shown in Figure 6-5, take the following steps:

1. Create a /test directory. (You should already know how to use the **mkdir** command for this purpose.)
2. Open the GUI Logical Volume Management tool; one method is to run the **system-config-lvm** command inside the GUI.
3. Navigate to the Uninitialized Entities section in the left-hand pane. Based on the standard VM installation, select drive /dev/sda, or a partition configured on that drive.
4. Click the Initialize Entity button that appears at the bottom of the window. This action applies the **pvcreate** command, in this case, to the entire drive.
5. You'll see a warning that all data on the partition will be lost. Assuming that's not a problem for you, click Yes.
6. If you're initializing an entire disk, you'll see a suggestion to create a single partition encompassing the entire drive. Click Yes.
7. Confirm the result from a command line; run the **fdisk -l** command to confirm the new partition of the appropriate LVM type, and the **pvs** command to confirm the creation of the physical volume.
8. If you haven't seen it before, you'll see an Unallocated Volumes category. Select the PV that you've just initialized. You'll see three options:
   - The Create New Volume Group option allows you to create a new VG for another filesystem, which corresponds to the **vgcreate** command.
   - Add To Existing Volume Group allows you to increase the space associated with the VG of your choice. This corresponds to the **vgextend** command.
   - Remove Volume From LVM reverses the process.

   As this section is based on adding a new VG, select Create New Volume Group. You'll see the window shown in Figure 6-6, where you can assign a name and set the size for the new VG; the default uses all available space. For the purpose of this section, I've named the new VG NewVol. Name the volume and click OK.
9. You'll see the new VG in the Volume Groups category. In the left-hand pane, navigate to the name of the new VG, in this case, NewVol, and select the Logical View for this group. Click the Create New Logical Volume button

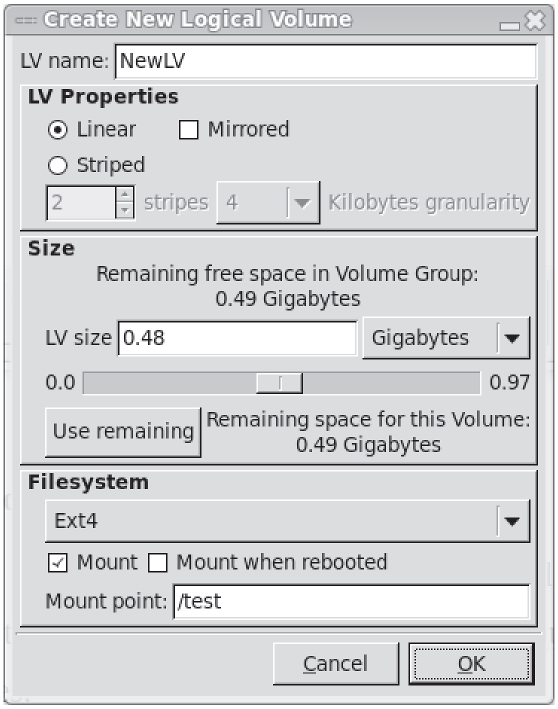
that appears at the bottom of the window, which opens the window shown in Figure 6-7.

10. In the Create New Logical Volume window, you can configure the amount of space assigned to the LV, the format, and the mount point. In Figure 6-7, I've allocated half the space to NewLV, formatted it to ext4, and set it to be mounted on the previously created /test directory. If you were to also select the Mount When Rebooted option, the tool would also configure the LV in the /etc/fstab file.

This new LV is now ready for a new filesystem; you can copy desired files to the /test directory, unmount it, and change /etc/fstab to mount it on a different directory. But for the purpose of this section, complete Step 10, but don't make any further changes to the /test directory or the /etc/fstab file. If you want to see the changes reflected in the Logical Volume Management tool, click View | Reload.

**FIGURE 6-7**

Create a New Logical Volume


Create New Logical Volume

LV name: NewLV

LV Properties
(● Linear) ☐ Mirrored
○ Striped
2 stripes 4 Kilobytes granularity

Size
Remaining free space in Volume Group:
0.49 Gigabytes
LV size 0.48 Gigabytes
0.0 0.97
Remaining space for this Volume:
0.49 Gigabytes

Filesystem
Ext4
☑ Mount ☐ Mount when rebooted
Mount point: /test

## Remove an LV

This section assumes that you've created an LV using the steps described in the preceding section.

1. Unless it's already open, start the GUI Logical Volume Management tool.
2. In the left-hand pane, navigate to NewVol, select Logical View, and click NewLV. This is the LV assigned to and mounted on the /test directory.
3. Click the Remove Logical Volume button, which corresponds to the **lvremove** command. Confirm when prompted.

on the job

***Before removing the LV, look at the other options. The Create Snapshot option supports a mirror of the current LV. It works only if there is sufficient unallocated space from available VGs. The Edit Properties option supports changes to the LV settings described in the previous subsection.***

   You'll see a warning about NewLV containing data from directory /test. Any data copied to that directory will be lost if you click Yes. Go ahead. Technically, that's all you need to remove the LV. But you can do more.

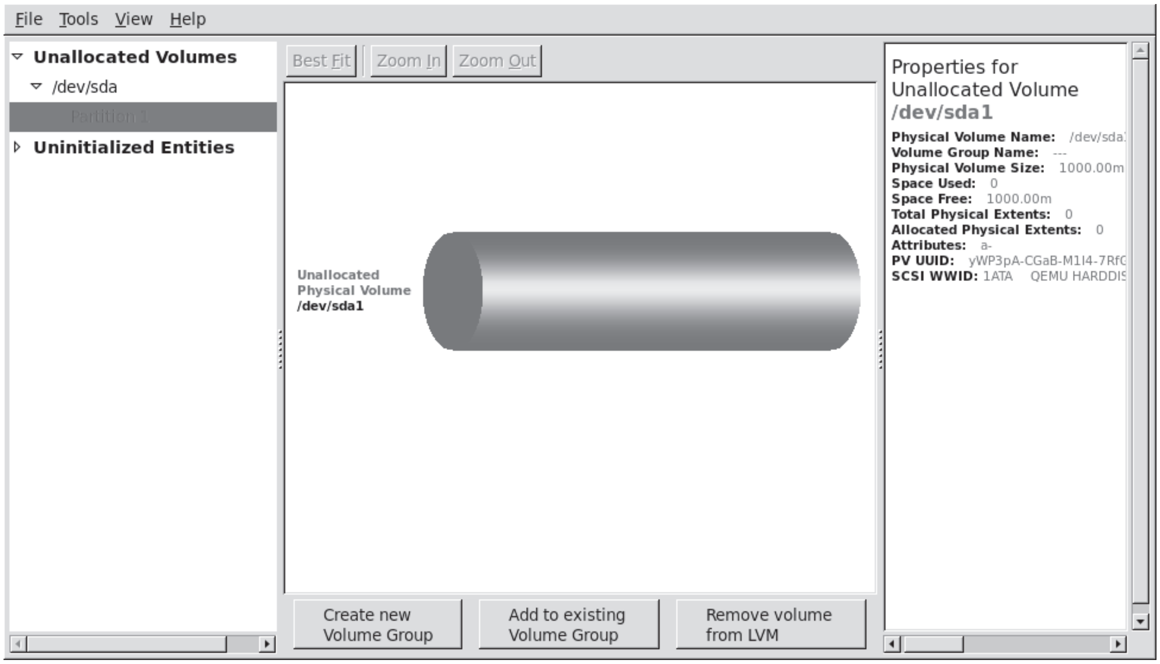
4. Move up a bit in the left-hand pane. Click NewVol and then select Physical View. Navigate down to the partition you created in the previous subsection, and then select Remove Volume From Volume Group, and select Yes when the warning appears.
5. You'll now see the partition in the Unallocated Volumes category, as shown in Figure 6-8. Navigate to and select the subject partition. Do not click Remove Volume From LVM.
6. In preparation for the next subsection, reverse the process. Rerun Steps 9 and 10 from the Add an LV section. Refer to Figure 6-7 if and as needed.

## Add to and Increase the Size of an LV

In this subsection, you'll redo the first steps to add an LV from a newly available partition or hard drive, as described in an earlier section. Instead of creating a new LV, you'll add the new space to an existing LV. To do so, take the following steps:

1. Unless it's already open, start the GUI Logical Volume Management tool.
2. Navigate to the Uninitialized Entities associated with the other spare drive, /dev/sdb.

**FIGURE 6-8** You could remove a logical volume.



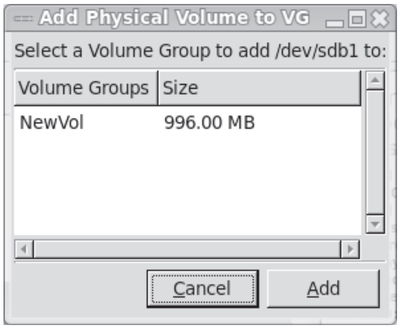
Properties for Unallocated Volume /dev/sda1

**Physical Volume Name:** /dev/sda1
**Volume Group Name:** ---
**Physical Volume Size:** 1000.00m
**Space Used:** 0
**Space Free:** 1000.00m
**Total Physical Extents:** 0
**Allocated Physical Extents:** 0
**Attributes:** a-
**PV UUID:** yWP3pA-CGaB-M1I4-7RfG
**SCSI WWID:** 1ATA QEMU HARDDIS

3. Select a partition or a drive; in this case, /dev/sdb1 or /dev/sdb. It should be different from the partition or drive created earlier. Click the Initialize Entity button that appears at the bottom of the window, and confirm when the warning message appears. If you're working with a drive, accept the prompt to create a partition. This action applies the **pvcreate** command to the new partition.
4. In the Unallocated Volumes category, select the PV just initialized. In this case, select Add To Existing Volume Group. You'll see the Add Physical Volume To VG window shown in Figure 6-9.
5. Select the existing volume group of your choice and click Add. If you've followed the instructions so far, the volume group should be named NewVol. Under the Volume Groups heading, navigate to the name of the new LV and click Logical View; for the work done in earlier sections, the name is NewLV, as shown in Figure 6-10.

**FIGURE 6-9**

Add a PV to a logical volume.



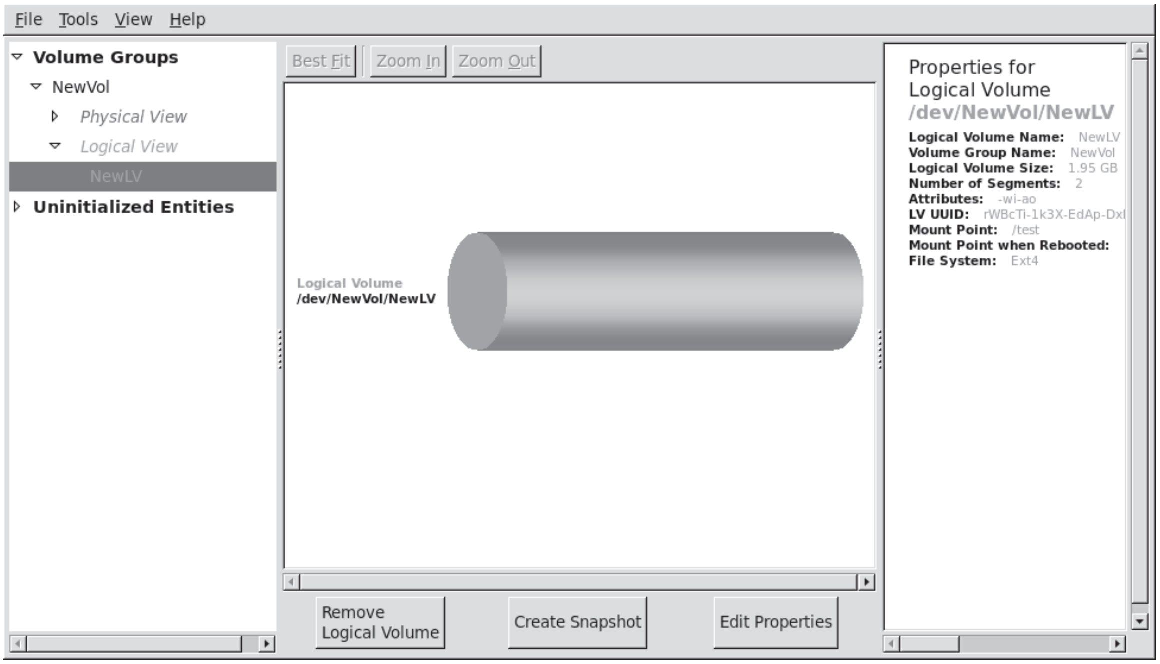
Select a Volume Group to add /dev/sdb1 to:

| Volume Groups | Size |
|---|---|
| NewVol | 996.00 MB |

6. Click Edit Properties. You'll see an Edit Logical Volume window similar (but not identical to) Figure 6-7. You'll be able to change the amount of space allocated from the VG to the NewLV Logical Volume. The advantage here is that it automatically resizes the formatted volume, with the **resize2fs** command. The process may take several minutes.

   Check the result; rerun the **df** command.

**FIGURE 6-10** Increase the size of a logical volume.



Properties for Logical Volume /dev/NewVol/NewLV

**Logical Volume Name:** NewLV
**Volume Group Name:** NewVol
**Logical Volume Size:** 1.95 GB
**Number of Segments:** 2
**Attributes:** -wi-ao
**LV UUID:** rWBcTi-1k3X-EdAp-Dxl
**Mount Point:** /test
**Mount Point when Rebooted:**
**File System:** Ext4

**exam watch**

***If you resize a volume during an exam, be prepared to work at other tasks while the volume is being resized.***

**CERTIFICATION OBJECTIVE 6.05**

## Volume Encryption with the Linux Unified Key Setup

Encryption can help secure data on key devices, such as the volume that may contain critical personal information. One way to encrypt devices on a system is with the Linux Unified Key Setup (LUKS). Encryption with LUKS works on a block level; in other words, it's applied to block device files such as the partitions and LVs associated with storage. If a key computer is lost, the data within is at least a bit more secure, as the LUKS-protected system requires either a passphrase or a keyfile.

### Passwords, Passphrases, and More

The discussion of LUKS is the first place in this book where you have to define a passphrase. Thus, it's the first opportunity to explore the difference between a passphrase and a password. So what happens if a laptop with a critical database of credit card numbers, or even social security numbers, is lost? The company that owns the laptop may then have to provide identity protection services, and even more for any users who subsequently suffer financial loss as a result.

Computer users are used to passwords. They're short and they're easy to remember. They have been adequate for networked systems that have been properly secured in other ways. While there are security issues associated with many passwords, users are resistant to change. But given the importance of data stored on many systems, a password is not enough. (Nevertheless, usernames and passwords are still part of the Red Hat exam requirements.)

However, given just a little time, most passwords can be cracked. In fact, the dictionary words used by most people as passwords can be cracked in a matter of seconds. With this fact in mind, computer professionals encourage users to create passwords with some combination of uppercase and lowercase characters, mixed with numbers and punctuation. Perhaps the most difficult passwords to crack are based on the first alphanumeric characters of a favorite phrase. For example, a password like ET,Ie3ppsiR,NC. could stand for "Every Tuesday, I eat 3 pulled pork sandwiches in Raleigh, North Carolina." But given enough time, even passwords of that complexity can be cracked. In general, spaces are not allowed in a password.

Many networks are moving to more secure systems. To that end, it's more common to find partitions and entire drives encrypted with a password, or preferably a passphrase. Passwords are typically 6 to 10 characters. Longer passwords are difficult for users to remember, unless they're based on a favorite phrase.

One advantage of a passphrase is that it can also include spaces, and even complete sentences. Such longer entries, such as "Every Tuesday, I eat 3 pulled pork sandwiches in North Carolina." are more difficult to crack than just a password. But you can go further. Computer security professionals encourage the use of a combination of words and acronyms in a passphrase, perhaps something like "On Tuesday, Ia3pps in North Carolina." Such a combination of words and acronyms in a sentence is even more difficult for a cracker to decrypt. In any case, the use of passphrases is covered in both the RHCSA and RHCE exams. This section covers the use of passphrases to protect LUKS-encrypted storage volumes.

Of course, there are more secure authentication systems available. For example, one RHCE skill discussed in Chapter 11 combines the advantages of passphrases with encryption key cryptography. In addition, RHEL 6 (though not the RHCSA/RHCE exams) includes the drivers associated with fingerprint readers, which can also be set up for user logins.

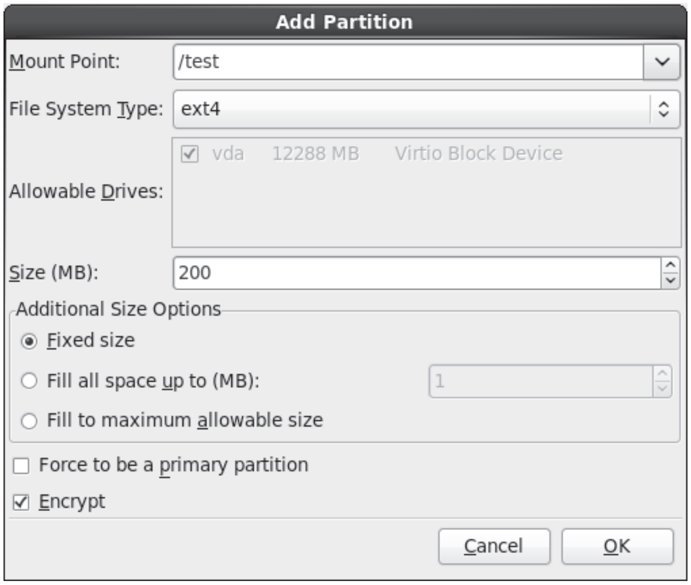
## Encryption During Installation

The easiest way to prepare a LUKS-encrypted volume is during the installation process. Figure 6-11 illustrates the RHEL 6 installation screen where a LUKS-encrypted volume is created on the /test directory.

If you've paid careful attention during the installation process, you may remember that the entire system can be encrypted, as shown back in Chapter 1, in Figure 1-6.

But as suggested at the beginning of this chapter, for the RHCSA, you need to know how to create, configure, mount, and unmount LUKS-encrypted filesystems. While this is normally done with a passphrase entered during the boot process, it also can be set up with a passkey, which can be set up on remote media such as a USB stick.

**FIGURE 6-11**



Mount Point: /test
File System Type: ext4
Allowable Drives: [✓] vda 12288 MB Virtio Block Device
Size (MB): 200
Additional Size Options
(●) Fixed size
( ) Fill all space up to (MB): 1
( ) Fill to maximum allowable size
[ ] Force to be a primary partition
[✓] Encrypt

Create a LUKS-encrypted volume with Disk Druid.

## Prepare and Initialize Encryption

So in this chapter, you need to learn how to set up an encrypted filesystem mounted on a specific directory. It requires the dm_crypt module, which is installed as part of the baseline RHEL 6 kernel package. It should be shown in the following output to the **lsmod | grep dm_crypt** command:

```
dm_crypt      12860  0
dm_mod        76856  7 dm_crypt,dm_mirror,dm_log
```

If you don't see this output, run the following command:

```
# modprobe dm_crypt
```

You'll also need the cryptsetup-luks RPM package. Install it if needed with the following command:

```
# yum install cryptsetup-luks
```

When a passphrase is configured on a LUKS filesystem, the user that boots a system is prompted for that passphrase during that boot process, with a prompt similar to the following:

```
Password for /dev/vda2 (luks-45b...):**************************
```

Each character of the passphrase, including spaces, is shown with asterisks. A cracker who gets a hold of a LUKS-encrypted system won't be able to boot the system, or at least any filesystems that are encrypted during the boot process.

## Prepare the New Filesystem

Of course, before creating an encrypted filesystem, you need a partition. While it can be processed into a logical volume or RAID array first, it's best to keep things simple when learning something new. So for the purpose of this section, start with a regular partition. Use a tool like **fdisk** or **parted** to create a regular partition from existing empty space on a hard drive.

If desired, it's possible to create a more secure filesystem by first filling it with random data. Just be aware that these options take time. While they are excellent ways to create a more secure filesystem, don't do it on an exam unless you're specifically asked to do so! One way to so is with the following **badblocks** command on device /dev/sda1. If the partition you're using is different, substitute accordingly.

```
# badblocks -c 10240 -s -w -t random -v /dev/sda1
```

This particular command tests 10240 blocks at a time (2^10 * 10) with the number of blocks (**-c**) switch. The **-s** illustrates the progress of the command. The **-w** does the actual writing of data, in a random test pattern (**-t random**), in verbose mode (**-v**). This command took a couple of minutes to add random data to a 1GB partition on my server1.example.com VM system.

An alternative is to use the Linux random number generator device, /dev/urandom. The following command is simpler in a way, as it starts by dumping random data, block by block, on the /dev/sda1 device:

```
# dd if=/dev/urandom of=/dev/sda1
```

This command took about five minutes to add random data to a 1GB partition on my server1.example.com VM system.

## Create the New Filesystem

The command that creates a LUKS-based filesystem is **cryptsetup**. The first step is to actually set up the passphrase for the filesystem with the following command. You're prompted for confirmation and a passphrase:

```
# cryptsetup luksFormat /dev/sda1
WARNING!
========
This will overwrite data on /dev/sda1 irrevocably.
Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase:
Verify passphrase:
```

The passphrases that you type in are not shown at the console. The command shown in Figure 6-12 displays header information from the encrypted device. Once again, case matters. If you type in **yes** in lowercase, the command does not prompt for a passphrase, and the volume is not encrypted.

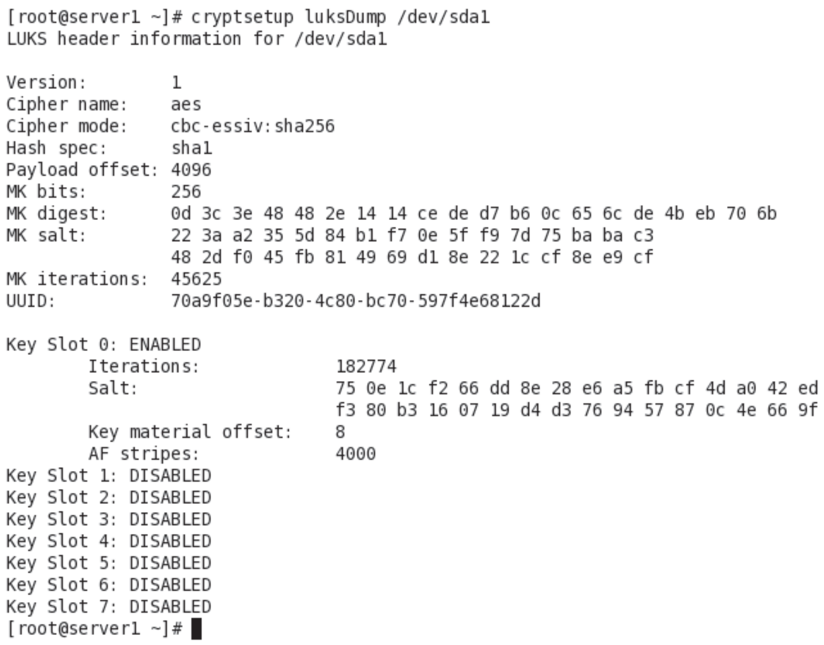
A couple of more steps are needed before the encrypted device is ready for formatting: it must be mapped. In other words, since the /dev/sda1 volume is now encrypted, it can't be read. However, you can create a map to a different device, which is essentially the decrypted version of the device.

First, you need a UUID for the device. The following **cryptsetup** command creates a UUID for the newly encrypted /dev/sda1 device:

```
# cryptsetup luksUUID /dev/sda1
```

**FIGURE 6-12**

The header of a LUKS-encrypted volume



```
[root@server1 ~]# cryptsetup luksDump /dev/sda1
LUKS header information for /dev/sda1

Version:        1
Cipher name:    aes
Cipher mode:    cbc-essiv:sha256
Hash spec:      sha1
Payload offset: 4096
MK bits:        256
MK digest:      0d 3c 3e 48 48 2e 14 14 ce de d7 b6 0c 65 6c de 4b eb 70 6b
MK salt:        22 3a a2 35 5d 84 b1 f7 0e 5f f9 7d 75 ba ba c3
                48 2d f0 45 fb 81 49 69 d1 8e 22 1c cf 8e e9 cf
MK iterations:  45625
UUID:           70a9f05e-b320-4c80-bc70-597f4e68122d

Key Slot 0: ENABLED
        Iterations:             182774
        Salt:                   75 0e 1c f2 66 dd 8e 28 e6 a5 fb cf 4d a0 42 ed
                                f3 80 b3 16 07 19 d4 d3 76 94 57 87 0c 4e 66 9f
        Key material offset:    8
        AF stripes:             4000
Key Slot 1: DISABLED
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
[root@server1 ~]#
```

Naturally, that command returns a 128-bit UUID number. Red Hat documentation recommends that you use that number as part of device file for the encrypted filesystem. You can copy that number to the local buffer by highlighting it in a console. In a GUI console, you should be able to right-click and select Copy in the pop-up menu that appears. In a text console, if the mouse is active, that UUID number should already be in the buffer.

Now type in the following command:

```
# cryptsetup luksOpen /dev/sda1 uuidnumber
```

Paste the UUID number in place of *uuidnumber*. If it's in the GUI console, right-click at the end of the command and click Paste in the pop-up menu that appears. If it's in the command line console, click the middle mouse button (or the left and right mouse buttons together). Alternatively, if mouse buttons are unavailable, you could run the following commands (Of course, the two commands can be combined.):

```
# cryptsetup luksUUID /dev/sda1 > sda1uuid
# cryptsetup luksOpen /dev/sda1 `cat sda1uuid`
```

**exam watch**

***Unless required on an exam, there's no reason to use the UUID number of an encrypted device. For example, you could run the following alternative cryptographic encryption command:*** **cryptsetup luksOpen /dev/sda1 test.**

The mapped device should now appear in the /dev/mapper directory. You should now be able to format that device with a command like **mkfs.ext4**. The following command formats the device I created in my server1.example.com system:

```
# mkfs.ext4 /dev/mapper/test
```

You should now be able to apply the **mount** command on that /dev/mapper device file. Ideally, you should now set that up in the /etc/fstab file to make sure that encrypted filesystem is mounted the next time that system is booted.

If you're already jumping ahead to configuration of the encrypted filesystem in the /etc/fstab file, this UUID corresponds to the original partition and is not associated with the encrypted filesystem. To find the UUID associated with this particular encrypted filesystem, run the following command:

```
# dumpe2fs /dev/mapper/test | grep UUID
```

That UUID number can then be used to represent the encrypted volume in /etc/fstab. The following are two ways such a volume could be configured in that file:

```
/dev/mapper/test       /test      ext4      defaults      1 2
UUID=uuidnumber        /test      ext4      defaults      1 2
```

Alternatively, the aforementioned mkfs.ext4 command would change the associated UUID, as confirmed by the output to the **blkid** command. That UUID in the **blkid** command output can then be used to represent the noted volume in a similar fashion.

If successful, you'll be prompted for the passphrase for any LUKS-encrypted filesystems that may have been created. One common related error is the unloading of the dm_crypt module during the system reboot process.

## CERTIFICATION OBJECTIVE 6.06

## Filesystem Management

Before you can access the files in a directory, that directory must be mounted on a partition formatted to some readable filesystem. Linux normally automates this process using the /etc/fstab configuration file. When Linux goes through the boot process, directories specified in /etc/fstab are mounted on configured volumes, with the help of the **mount** command. Of course, you can run that command with any or all appropriate options. So that's an excellent place to start this section.

The remainder of this section focuses on options for /etc/fstab. While it starts with the default using the baseline configuration for the standard virtual machine, it includes options for how you can customize that file for local, remote, and removable filesystems.

## The /etc/fstab File

The standard configuration file for filesystems mounted during the boot process is /etc/fstab. To look it over safely, run the **less /etc/fstab** command. From the example shown in Figure 6-13, different filesystems are configured on each line.

The default use of UUIDs is new for RHEL 6, as RHEL 5 used the LABEL directive. As you'll see in the next section, UUIDs can represent a partition, a logical volume, or a RAID array. In all cases, that volume should be formatted to the filesystem noted on each line. That volume is mounted on the directory in the second column. The advantage of an UUID, relative to the LABEL directive, is that the UUID is automatically created when the volume is formatted with a command like mkfs.ext4.

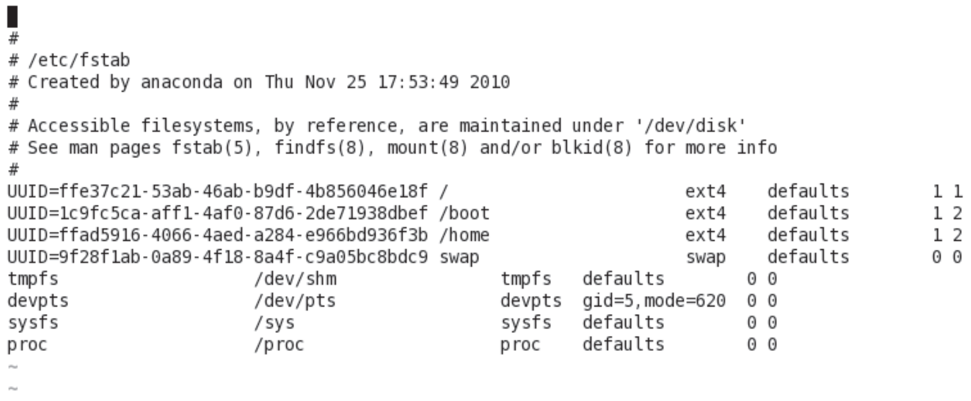
on the job

***Line wrapping in the /etc/fstab file is now allowed on current versions of Linux, including RHEL 6. It is necessary to accommodate the length of the UUID.***

But to some extent, UUIDs are beside the point. As shown in Figure 6-13, there are six fields associated with each filesystem, described from left to right in Table 6-8. You can verify how partitions are actually mounted in the /etc/mtab file, as shown in Figure 6-14. Note the differences, especially the use of the device file associated with the partition, LV, or RAID array.

**FIGURE 6-13**

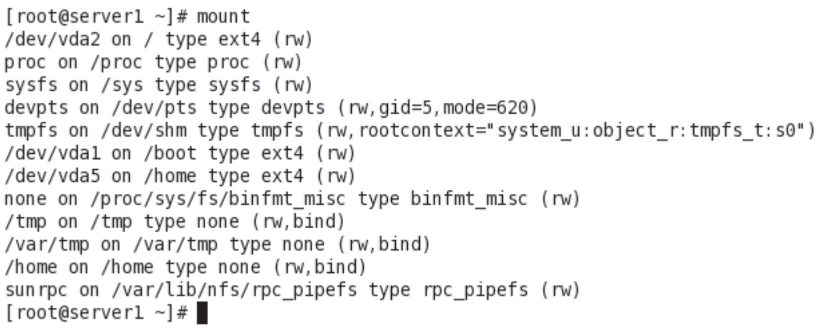
Sample /etc/fstab



```
#
# /etc/fstab
# Created by anaconda on Thu Nov 25 17:53:49 2010
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
UUID=ffe37c21-53ab-46ab-b9df-4b856046e18f /                       ext4    defaults        1 1
UUID=1c9fc5ca-aff1-4af0-87d6-2de71938dbef /boot                   ext4    defaults        1 2
UUID=ffad5916-4066-4aed-a284-e966bd936f3b /home                   ext4    defaults        1 2
UUID=9f28f1ab-0a89-4f18-8a4f-c9a05bc8bdc9 swap                    swap    defaults        0 0
tmpfs                   /dev/shm                tmpfs   defaults        0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
~
~
```

**FIGURE 6-14**

Sample /etc/mtab



```
[root@server1 ~]# mount
/dev/vda2 on / type ext4 (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,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/vda1 on /boot type ext4 (rw)
/dev/vda5 on /home type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/tmp on /tmp type none (rw,bind)
/var/tmp on /var/tmp type none (rw,bind)
/home on /home type none (rw,bind)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
[root@server1 ~]# █
```

When adding a new partition, you could just add the device file associated with the partition, LV, or LUKS volume to the first column. The last four lines include virtual filesystems, and are discussed later in this chapter.

**TABLE 6-8**

Description of /etc/fstab by Column, Left to Right

| Field Name | Description |
|---|---|
| Device | Lists the device to be mounted; you may substitute the UUID or LABEL. |
| Mount Point | Notes the directory where the filesystem will be mounted. |
| Filesystem Format | Describes the filesystem type. Valid filesystem types include ext, ext2, ext3, ext4, msdos, vfat, devpts, proc, tmpfs, udf, iso9660, nfs, smb, and swap. |
| Mount Options | Covered in the following section. |
| Dump Value | Either 0 or 1. A value of 1 means that data is automatically saved to disk by the **dump** command when you exit Linux. |
| Filesystem Check Order | Determines the order that filesystems are checked by the **fsck** command during the boot process. The root directory (/) filesystem should be set to 1, and other local filesystems should be set to 2. Removable filesystems such as those associated with CD/DVD drives should be set to 0, which means that they are not checked during the Linux boot process. |

## Universally Unique Identifiers in /etc/fstab

In /etc/fstab, note the focus on UUIDs, short for Universally Unique Identifiers. Every formatted volume has an UUID, a unique 128-bit number. Each UUID represents either a partition, a logical volume, or a RAID array.

To identify the UUID for available volumes, run the **blkid** command. The output should sort of match volumes from partitions, LVs, RAID arrays, and LUKS-encrypted filesystems to their UUIDs, as defined in /etc/fstab. To that end, both LVs and LUKS filesystems use /dev/mapper devices. For one of the LVs and the VGs created in this chapter, that device file would be /dev/mapper/vg_00-lv_00. To verify, run the **ls -l** command on both device files (/dev/mapper/vg_00-lv_00 and /dev/vg_00/lv_00). You should see both device files linked to the same /dev/dm-0 (the dm-* number can vary) device file. In the same way, you can verify that the /dev/mapper file for the LUKS encrypted volume is linked to another /dev/dm-* device file.

Alternatively, you could use the **dumpe2fs** command on available volumes; for example, the following command identifies the UUID associated with the noted LV:

```
# dumpe2fs /dev/vg_00/lv_00 | grep UUID
```

As UUIDs are not limited to LVs, you should be able to get equivalent information for a partition from a command like the following:

```
# dumpe2fs /dev/vda1 | grep UUID
```

Of course, the same is true with a properly configured and formatted LUKS volume (in this case, the UUID starts with a 7), with a command like the following:

```
# dumpe2fs /dev/mapper/7* | grep UUID
```

But the **dumpe2fs** command does not work with swap volumes, whether they be partitions or logical volumes. To identify the UUID associated with swap space, you just have to rely on the output of the **blkid** command.

## The mount Command

The **mount** command can be used to attach local and network partitions to specified directories. Mount points are not fixed; you can mount a CD drive or even a shared network directory to any empty directory if appropriate ownership and permissions are set. Closely related is the **umount** (not unmount) command, which unmounts selected volumes from associated directories.

First, try the **mount** command by itself. It'll display all currently mounted filesystems, along with important mount options. For example, the following output suggests that the top-level root directory is mounted in read-write mode, on the /dev/vda2 partition, formatted to the ext4 filesystem:

```
/dev/vda2 on / type ext4 (rw)
```

As suggested earlier, the **mount** command is closely related to the /etc/fstab file. If you've unmounted a directory, and have made changes to the /etc/fstab file, the easiest way to implement the currently configured /etc/fstab file is with the following command:

```
# mount -a
```

However, if a filesystem is already mounted, this command doesn't change its status, no matter what has been done to the /etc/fstab file. But if the system is subsequently rebooted, the options configured in /etc/fstab are used automatically.

If you're not sure about a possible change to the /etc/fstab file, it's possible to test it out with the **mount** command. For example, the following command remounts the volume associated with the /home directory, in read-only mode:

```
# mount -o remount,ro /home
```

You can confirm the result by rerunning the **mount** command. The following output should reflect the result on the /home directory:

```
/dev/vda5 on /home type ext4 (ro)
```

If you've read this book from the beginning, you've already seen the **mount** command at work with access control lists (ACLs), and even the ISO files associated with downloaded CD/DVDs. To review, the following command remounts the noted /home directory with ACLs:

```
# mount -o remount,acl /dev/vda5 /home
```

And for ISO files, the following command mounts the noted RHEL 6 ISO file on the /mnt directory:

```
# mount -o loop rhel-server-6.0-x86_64-dvd.iso /mnt
```

## More Filesystem Mount Options

Many **mount** command options are appropriate for the /etc/fstab file. One option most commonly seen in that file is **defaults**. While that is the appropriate mount option for most /etc/fstab filesystems, there are other options, such as those listed in Table 6-9. If you want to use multiple options, separate them by commas. Don't use spaces between options. The list in Table 6-9 is not comprehensive, and these options can also be used with the **mount** command. You can find out more from the mount man page, available with the **man mount** command.

**TABLE 6-9**

Options for the mount command and /etc/fstab

| Mount Option | Description |
|---|---|
| async | Data is read and written asynchronously. |
| atime | The inode associated with each file is updated each time the file is accessed. |
| auto | Searches through /etc/filesystems for the appropriate format for the partition; normally associated with a floppy or removable drive. |
| defaults | Uses default mount options rw, suid, dev, exec, auto, nouser, and async. |
| dev | Permits access to character devices such as terminals or consoles and block devices such as drives. |
| exec | Allows binaries (compiled programs) to be run on this filesystem. |
| noatime | The inode associated with each file is not updated when accessed. |
| noauto | Requires explicit mounting. Common option for CD and floppy drives. |
| nodev | Devices on this filesystem are not read or interpreted. |
| noexec | Binaries (compiled programs) cannot be run on this filesystem. |
| nosuid | Disallows setuid or setgid permissions on this filesystem. |
| nouser | Only root users are allowed to mount the specified filesystem. |
| remount | Remounts a currently mounted filesystem. Also an option for the mount command. |
| ro | Mounts the filesystem as read-only. |
| rw | Mounts the filesystem as read/write. |
| suid | Allows setuid or setgid permissions on programs on this filesystem. |
| sync | Reads and writes are done at the same speed (synchronously) on this filesystem. |
| user | Allows nonroot users to mount this filesystem. By default, this also sets the noexec, nosuid, and nodev options. |

There are more options available, including **noatime**, **noauto**, **nodev**, **noexec**, **nosuid**, and **nouser**, which are the opposites of **atime**, **auto**, **dev**, **exec**, **suid**, and **user**, respectively.

## Virtual Filesystems

This section describes the virtual filesystems configured in the /etc/fstab configuration file. The four standard virtual filesystem configuration lines are:

```
tmpfs      /dev/shm      tmpfs    defaults         0 0
devpts     /dev/pts      devpts   gid=5,mode=620   0 0
sysfs      /sys          sysfs    defaults         0 0
proc       /proc         proc     defaults         0 0
```

- The **tmpfs** filesystem is a virtual memory filesystem that uses both RAM and swap space.
- The **devpts** filesystem relates to pseudo-terminal devices.
- The **sysfs** filesystem provides dynamic information about system devices. Explore the associated /sys directory. You'll find a wide variety of information related to the devices and drivers attached to the local system.
- The **proc** filesystem is especially useful, as it provides dynamically configurable options for changing the behavior of the kernel. As an RHCE skill, you may learn more about options in the proc filesystem in Chapter 12.

## Add Your Own Filesystems to /etc/fstab

If you need to set up a special directory, it sometimes makes sense to set it up on a separate volume. Different volumes for different directories means that files in that volume can't overload critical directories such as /boot. While it's nice to follow the standard format of the /etc/fstab file, it is an extra effort. If required on a Red Hat exam, it'll be in the instructions that you see.

So in most cases, it's sufficient to set up a new volume in /etc/fstab with the associated device file, such as a /dev/vda6 partition, a /dev/mapper/* LUKS encrypted volume, or a /dev/NewVol/NewLV LV. Make sure the device file reflects the new volume that you've created, the intended mount directory (such as /special), and the filesystem format that you've applied (such as ext4).

## Removable Media and /etc/fstab

In general, removable media should not be mounted automatically during the boot process. That's possible in the /etc/fstab configuration file with an option like **noauto**. But in general, it's not standard in RHEL to set up removable media in /etc/fstab.

To read removable media such as smart cards and CD/DVDs, RHEL somewhat automates the mounting of such media in the GNOME or KDE Desktop Environments. While the details of this process are not part of the Red Hat Exam Prep guide, the process is based on configuration files in the /etc/udev/rules.d directory. If RHEL detects your hardware, click Places; in the menu that appears, select the entry for the removable media. If multiple removable media options are loaded, you can select the media to mount in the Removable Media submenu.

If that doesn't work for some reason, you can use the **mount** command directly. For example, the following command mounts a CD/DVD in a drive.

```
# mount -t iso9660 /dev/sr0 /mnt
```

The **-t** switch specifies the type of filesystem (iso9660). The device file /dev/sr0 represents the first CD/DVD drive; /mnt/ is the directory through which you can access the files from the CD/DVD after mounting. But /dev/sr0? How is anyone supposed to remember that?

Linux addresses that in a couple of ways. First, it sets up links from more sensibly named files such as /dev/cdrom, which you can confirm with the **ls -l /dev/cdrom** command. Second, try the **blkid** command. If removable media (other than a CD/DVD) are connected, you'll see it in the output to the command, including the associated device file.

Just remember that it is important to unmount removable media such as USB keys before removing them. Otherwise, the data that you thought was written to the disk might still be in the unwritten RAM cache. In that case, you would lose that data.

Given these examples of how removable media can be mounted, you should have a better idea on how such media can be configured in the /etc/fstab configuration file. The standard **defaults** option is inappropriate in most cases, as it mounts a system in read-write mode (even for read-only DVDs), attempts to mount automatically during the boot process, and limits access to the root administrative user. But that can be changed with the right options. For example, to configure a CD drive that can be mounted by regular users, you could add the following line to /etc/fstab:

```
/dev/sr0 /cdrom auto ro,noauto,users 0 0
```

This line sets up a mount in read-only mode, does not try to mount it automatically during the boot process, and supports access by regular users.

As desired, similar options are possible for removable media such as USB keys. But that can be more problematic with multiple USB keys; one may be detected as /dev/sdc once, and then later detected as /dev/sdd, if there's a second USB key installed. However, if properly configured, each USB key should have unique UUIDs. As described earlier, UUIDs are the RHEL 6 way to identify different volumes to be mounted in /etc/fstab.

## Networked Filesystems

The /etc/fstab file can be used to automate mounts from shared directories. The two major sharing services of interest are NFS and Samba. This section provides only a brief overview to how such shared directories can be configured in the /etc/fstab file; for more information, see Chapters 15 and 16.

In general, shares from networked directories should be assumed to be unreliable. People step on power lines, on Ethernet cables, and so on. If your system uses a wireless network, that adds an additional level of unreliability. In other words, the settings in the /etc/fstab file should account for that. So if there's a problem either in the network connection, or perhaps a problem like a power failure on the remote NFS server, that problem should not lead to a "hang" of a client, where the terminal (or more) is frozen due to the lost connection. Similar issues do not apply to Samba.

A connection to a shared NFS directory is based on its hostname or IP address, along with the full path to the directory on the server. So to connect to a remote NFS server on system *server1* that shares the /var/ftp/pub directory, you could mount that share with the following command (assuming the /share directory exists):

```
# mount -t nfs server1.example.com:/pub /share
```

But that mount has some risks. The Red Hat Storage Administration Guide suggests the following entry in /etc/fstab:

```
server1:/pub  /share  nfs rsize=8192,wsize=8192,timeo=14,intr,udp 0 0
```

The **rsize** and **wsize** variables specify the size of the blocks of data to be read and written in bytes, respectively. The **timeo=14** directive specifies that the client will wait 1.4 seconds for the connection request to be completed. The **intr** allows the client to interrupt a connection; that's useful when the NFS server is not responding. The **udp** specifies a connection using the User Datagram Protocol (UDP). If the connection is to a NFS version 4 server, substitute **nfs4** for **nfs** in the third column.

In contrast, such variables are not required for shared Samba directories. The following line is generally all that's needed for a share of the same directory and server:

```
//server/pub  /share  cifs rw,username=user,password=pass, 0 0
```

Older Samba servers may require a substitution of **smbfs** for **cifs**, which are acronyms for the Samba File System and its successor, the Common Internet File System. If you're disturbed by the open display of a username and password in the /etc/fstab file, which is world-readable, try the following option:

```
//server/pub  /share  cifs rw,credentials=/etc/secret 0 0
```

You can then set up the /etc/secret file as accessible only to the root administrative user, with the username and password in the following format:

```
username=user
password=password
```

## CERTIFICATION OBJECTIVE 6.07

## The Automounter

With network mounts and portable media, problems may come up if connections are lost or media are removed. During the server configuration process, you could be mounting directories from a number of remote systems. You may also want temporary access to removable media such as USB keys or Zip drives. The automount daemon, also known as the automounter or autofs, can help. It can automatically mount specific directories as needed. It can unmount a directory automatically after a fixed period of time.

### Mounting via the Automounter

Once a partition is mounted, it stays mounted until you unmount it or shut down the system. The permanence of the mount can cause problems. For example, if you've mounted a USB key and then physically remove the key, Linux may not have had a chance to write the file to the disk. Data would be lost. The same issue applies to secure digital cards or other hotswappable removable drives.

Another issue: mounted NFS directories may cause problems if the remote computer fails or the connection is lost. Systems may slow down or even hang as it looks for the mounted directory.

This is where the automounter can help. It relies on the **autofs** daemon to mount configured directories as needed, on a temporary basis. In RHEL, the relevant configuration files are auto.master, auto.misc, auto.net, and auto.smb, all in the /etc directory. If you use the automounter, keep the /misc and /net directories free. Red Hat configures automounts on these directories by default, and they won't work if local files or directories are stored there. Subsections will cover each of these files, and a hypothetical auto.home file for automatically mounting directories from a remote server on the local /home directory.

on the job

***You won't even see the /misc and/or /net directories unless you properly configure /etc/auto.master and the autofs daemon is running.***

Default automounter settings are configured in /etc/sysconfig/autofs. The default settings include a timeout of 300 seconds; in other words, if nothing happens on an automount within that time, the share is automatically unmounted:

```
TIMEOUT=300
```

The **BROWSE_MODE** can allow you to search from available mounts. The following directive disables it by default:

```
BROWSE_MODE="no"
```

There are a wide variety of additional settings available, as commented in the /etc/sysconfig/autofs file, which also assumes that the **autofs** daemon is active.

## /etc/auto.master

The standard /etc/auto.master file includes a series of comments, with three default commands. The first refers to the /etc/auto.misc file as the configuration file for this directory. The **/net -hosts** command allows you to specify the host to automount a network directory, as specified in /etc/auto.net.

```
/misc /etc/auto.misc
/net  -hosts
+auto.master
```

In any case, these commands point to configuration files for each service. Shared directories from each service are automatically mounted, on demand, on the given directory (/misc and /net).

You can set up the automounter on other directories. One popular option is to set up the automounter on the /home directory. In this way, you can configure user home directories on remote servers, mounted on demand. Users are given access to their home directories upon login, and based on the **TIMEOUT** directive in the /etc/sysconfig/autofs file, all mounted directories are automatically unmounted 300 seconds after that user logs off the system.

```
# /home /etc/auto.home
```

This works only if a /home directory doesn't already exist on the local system. As the Red Hat exam requires the configuration of a number of regular users, your systems should include a /home directory for regular users. In that case, you could substitute a different directory, leading to a line like the following:

```
/shared /etc/auto.home
```

One method that can be used to create the auto.home directory is discussed shortly. Just remember, for any system accessed over a network, you'll need to be sure that the firewall allows traffic associated with the given service.

## /etc/auto.misc

Red Hat conveniently provides standard automount commands in comments in the /etc/auto.misc file. It's helpful to analyze this file in detail. I use the default RHEL version of this file. The first four lines are comments, which I skip. The first directive is:

```
cd     -fstype=iso9660,ro,nosuid,nodev    :/dev/cdrom
```

In RHEL, this directive is active by default, assuming you've activated the **autofs** service. In other words, if you have a CD in the /dev/cdrom drive, you can access its files through the automounter with the **ls /misc/cd** command, even as a regular user. The automounter accesses it using the ISO9660 filesystem. It's mounted read-only (**ro**); set user ID permissions are not allowed (**nosuid**), and devices on this filesystem are not used (**nodev**).

With the **TIMEOUT** interval defined in /etc/sysconfig/autofs, the CD is unmounted 300 seconds after the last time it's accessed. There are a number of other sample commands, commented out, ready for use. Of course, you would have to delete the comment character (#) before using any of these commands. And you'd have to adjust names and device files accordingly; for example, /dev/hda1 is no longer used as a device file on the latest Linux systems, even for PATA hard drives.

As suggested by one of the comments, "the following entries are samples to pique your imagination." The first of these commented commands allows you to set up a /misc/linux mount point from a shared NFS directory, /pub/linux, from the ftp.example.org computer:

```
#linux   -ro,soft,intr      ftp.example.org:/pub/linux
```

The next command assumes that the /boot directory is stored on the /dev/hda1 partition (even though /dev/hda1 is no longer a standard device file on RHEL 6). With this command, you don't need to mount /boot when you start Linux. Instead, this command allows you to automount it with the **mount /misc/boot** command.

```
#boot    -fstype=ext2      :/dev/hda1
```

The following three commands apply to a floppy disk drive. Don't laugh; floppies are fairly easy to create and configure on most virtual machine systems. The first command, set to an "auto" filesystem type, searches through /etc/filesystems to try to match what's on your floppy. The next two commands assume that the floppy is formatted to the ext2 filesystem.

```
#floppy      -fstype=auto     :/dev/fd0
#floppy      -fstype=ext2     :/dev/fd0
#e2floppy    -fstype=ext2     :/dev/fd0
```

The next command points to the first partition on the third drive. The **jaz** at the beginning suggests this is suitable for an Iomega-type Jaz drive.

```
#jaz       -fstype=ext2       :/dev/sdc1
```

Finally, the last command is based on an older system where the automounter is applied to a legacy PATA drive. Of course, the /dev/hdd device file is no longer used, so substitute accordingly. But the **removable** at the beginning suggests this is also suitable for removable hard drives. Of course, you'd likely have to change the filesystem format to something like ext4. As suggested earlier in this chapter, the **blkid** command can help identify available device files from removable systems like USB keys and portable drives.

```
#removable   -fstype=ext2       :/dev/hdd
```

In general, you'll need to modify these lines for available hardware.

### /etc/auto.net

With the /etc/auto.net configuration script, you can review and read shared NFS directories. It now works with the hostnames or IP addresses of NFS servers. By default, executable permissions are enabled on this file.

Assuming the automounter is active, and can connect to an NFS server with an IP address of 192.168.122.1, you can review shared NFS directories on that system with the following command:

```
# /etc/auto.net 192.168.122.1 -fstype=nfs,hard,intr,nodev,nosuid \

        /srv/ftp 192.168.122.1:/srv/ftp
```

This output tells me that the /srv/ftp directory on the 192.168.122.1 system is shared via NFS. Based on the directives in /etc/auto.master, you could access this share (assuming appropriate firewall and SELinux settings) with the following command:

```
# ls /net/192.168.122.1/srv/ftp
```

### /etc/auto.smb

One of the problems associated with the configuration of a shared Samba or CIFS directory is that it works, at least in its standard configuration, only with public directories. In other words, if you activate the /etc/auto.smb file, it'll only work with directories shared without a username or a password.

If you accept these unsecure conditions, it's possible to set up the /etc/auto.smb file in the same way as the /etc/auto.net file. First, you'd have to add it to the /etc/auto.master file in a similar fashion, with the following directive:

```
/smb  /etc/auto.smb
```

You'd then need to specifically stop and restart the automounter service with the following commands:

```
# /etc/init.d/autofs stop
# /etc/init.d/autofs start
```

You'll then be able to review shared directories with the following command; substitute a hostname or IP address if desired. Of course, this won't work unless the Samba server is activated on the noted server1.example.com system, and the firewall is configured to allow access through associated TCP/IP ports.

```
# /etc/auto.smb server1.example.com
```

### /etc/auto.home

Yes, it's possible to set up shared home directories from a central server using the automounter. However, that requires a central authentication database, such as the Network Information Service (NIS) or the Lightweight Directory Access Protocol (LDAP). Of course, it also requires a shared /home directory from that remote system. However, NIS is no longer part of the Red Hat exams, and LDAP client configuration is only part of the RHCSA exam requirements.

Therefore, if you're asked to configure a shared /home directory filesystem with the automounter, it would be only with the help of an existing LDAP server. LDAP and other authentication options are covered in Chapter 8.

To that end, automounter configuration requires the activation of five directives in the /etc/sysconfig/autofs file:

```
DEFAULT_MAP_OBJECT_CLASS="automountMap"
DEFAULT_ENTRY_OBJECT_CLASS="automount"
DEFAULT_MAP_ATTRIBUTE="automountMapName"
DEFAULT_ENTRY_ATTRIBUTE="automountKey"
DEFAULT_VALUE_ATTRIBUTE="automountInformation"
```

You'd then configure the shared home directory, per exam requirements, in the /etc/auto.home file. It may be something simple, similar to the following directive:

```
* -rw,soft,intr server1.example.com:/home/&
```

In this case, the shared directory is mounted with read-write permissions, soft interrupts that assume the network connection is completely reliable, and permission for signals to interrupt file operations.

## Activate the Automounter

Once appropriate files have been configured, you can start, restart, or reload the automounter. As it is governed by the **autofs** daemon, you can stop, start, restart, or reload that service with one of the following commands:

```
# service autofs stop
# service autofs start
# service autofs restart
# service autofs reload
```

With the default command in the /etc/auto.misc file, you should now be able to mount a CD on the /misc/cd directory, automatically, just by accessing the configured directory. Once you have a CD in the drive, the following command should work:

```
# ls /misc/cd
```

If you navigate to the /misc/cd directory, the automounter would ignore any timeouts. Otherwise, /misc/cd is automatically unmounted according to the timeout, which according to the **TIMEOUT** directive in /etc/sysconfig/autofs is 300 seconds.

## EXERCISE 6-3

### Configure the Automounter

In this exercise, you'll test the automounter. You'll need at least a CD. Ideally, you should also have a USB key, or a secure digital (SD) card. First, however, you need to make sure that the **autofs** daemon is in operation, modify the appropriate configuration files, and then restart **autofs**. You can then test the automounter in this lab.

1. From the command line interface, run the following command to make sure the **autofs** daemon is running:

   ```
   # service autofs start
   ```

2. Review the /etc/auto.master configuration file in a text editor. The defaults are sufficient to activate the configuration options in /etc/auto.misc and /etc/auto.net.
3. Check the /etc/auto.misc configuration file in a text editor. Make sure it includes the following line (which should already be there by default). Save and exit from /etc/auto.misc.

   ```
   cd     -fstype=iso9660,ro,nosuid,nodev    :/dev/cdrom
   ```

4. Now reload the **autofs** daemon. Since it's already running, all you need to do is make sure it rereads associated configuration files.

   ```
   # service autofs reload
   ```

5. The automounter service is now active. Insert a CD or DVD into an appropriate drive and run the following command. If successful, it should display the contents of the CD or DVD:

```
# ls /misc/cd
```

6. Run the **ls /misc** command immediately. You should see the CD directory in the output.
7. Wait at least five minutes, and repeat the previous command. What do you see?

## SCENARIO & SOLUTION

| | |
|---|---|
| You need to configure several new partitions for a standard Linux partition, for swap space, and for a logical volume | Use the **fdisk** or **parted** utility to create partitions, and then modify their partition types with the **t** or **set** commands. |
| You want to set up a mount during the boot process based on the UUID | Identify the UUID of the volume with the **dumpe2fs** or **blkid** commands, and use that UUID in the /etc/fstab file. |
| You need to configure a volume to the ext2, ext3, or ext4 filesystem type. | Format the target volume with a command like **mkfs.ext2**, **mkfs.ext3**, or **mkfs.ext4**. |
| You want to set up a logical volume. | Use the **pvcreate** command to create PVs; use the **vgcreate** command to combine PVs in VGs; use the **lvcreate** command to create an LV; format that LV for use. |
| You want to add new filesystems without destroying others | Use the free space available on existing or newly installed hard drives. |
| You need to encrypt a volume with sensitive data | Configure and format a special volume with the **cryptsetup** command; be prepared with an appropriate passphrase. |
| You want to expand the space available to an LV | Use the **lvextend** command to increase the space available to an LV, and then use the **resize2fs** command to expand the formatted filesystem accordingly. |
| You need to configure automated mounts to a shared network filesystem | Configure the filesystem either in /etc/fstab or through the automounter. |

## CERTIFICATION SUMMARY

As a Linux administrator, you should know how to create and manage new filesystem volumes. To create a new filesystem, you need to know how to create, manage, and format partitions, as well as how to set up those partitions for logical volumes. With the increasing focus on security, you also need to know how to encrypt those volumes with LUKS.

RHEL 6 also supports the configuration of logical volumes. The process is a bit intricate, as it requires the configuration of a partition as a PV. Multiple PVs can then be configured as a VG. Logical volumes can then be configured from desired portions of a VG. Associated commands are **pv***, **vg***, and **lv***; those and others can be accessed from the **lvm>** prompt. The GUI Logical Volume Management tool is also useful for this purpose.

Linux supports the format of partitions, RAID arrays, and logical volumes to a wide variety of filesystems. While the default is ext4, Linux supports formats and checks associated with regular and journaling filesystems associated with Linux, Microsoft, and other operating systems.

Once configured, partitions and logical volumes, whether they be encrypted or not, can be configured in the /etc/fstab file. That configuration is read during the boot process and can also be used by the **mount** command. If desired, removable filesystems and shared network directories can also be configured in /etc/fstab.

The /etc/fstab file is not the only option to set up mounts. You can automate this process for regular users with the automounter. Properly configured, it allows users to access shared network directories, removable media, and more through paths defined in /etc/auto.master.

## ✓ TWO-MINUTE DRILL

Here are some of the key points from the certification objectives in Chapter 6.

### Storage Management and Partitions

- ❑ The **fdisk** and **parted** utilities can help you create and delete partitions.
- ❑ Both **fdisk** and **parted** can be used to configure partitions for logical volumes and RAID arrays.
- ❑ The RHEL 6 Disk Utility, which can be started with the **palimpsest** command, can serve as a front end not only for partition management, but also for mounting, formatting, and filesystem checks.

### Filesystem Formats

- ❑ Linux tools can be used to configure and format volumes to a number of different filesystems.
- ❑ Examples of standard filesystems include MS-DOS and ext2.
- ❑ Journaling filesystems, which include logs that can restore metadata, are more resilient; the default RHEL 6 filesystem is ext4.
- ❑ RHEL 6 supports a variety of **mkfs.*** filesystem format and **fsck.*** filesystem check commands.

### Basic Linux Filesystems and Directories

- ❑ Linux files and filesystems are organized into directories based on the FHS.
- ❑ Some Linux directories are well suited to configuration on separate filesystems.

### Logical Volume Management (LVM)

- ❑ LVM is based on physical volumes, logical volumes, and volume groups.
- ❑ You can create and add LVM systems with a wide variety of commands starting with **pv***, **lv***, and **vg***.
- ❑ The space from new partitions configured as PVs can be allocated to existing volume groups with the **vgextend** command; they can be added to LVs with the **lvextend** command.
- ❑ The extra space can be used to extend an existing filesystem with the **resize2fs** command.

- ❑ The GUI LVM tool is an alternative for those who don't remember all of the commands required to manage logical volumes.

## Volume Encryption with the Linux Unified Key Setup

- ❑ The passphrases that can be used with LUKS can be considerably more secure than regular passwords.
- ❑ Volumes can be LUKS-encrypted during the installation process.
- ❑ LUKS encryption after installation depends on the dm_crypt module and commands in the cryptsetup-luks package.
- ❑ The safety of encrypted volumes can be enhanced with random data, added with either the **badblocks** command from the /dev/urandom random number generator.
- ❑ The **cryptsetup luksFormat /dev/sda1** command sets up a passphrase on the given partition. Once configured, you can format the /dev/mapper device that's created.

## Filesystem Management

- ❑ Standard filesystems are mounted as defined in /etc/fstab.
- ❑ Filesystem volumes are now identified by their UUIDs; for a list, run the **blkid** command.
- ❑ The **mount** command can either use the settings in /etc/fstab or mount filesystem volumes directly.
- ❑ It's also possible to configure mounts of shared network directories from NFS and Samba servers in /etc/fstab.
- ❑ With the right commands, you can convert an ext2 formatted filesystem to ext3; and then convert an ext3 formatted filesystem to ext4.

## The Automounter

- ❑ With the automounter, you can configure automatic mounts of removable media and shared network drives.
- ❑ Key automounter configuration files are auto.master, auto.misc, and auto.net, in the /etc directory.
- ❑ It's possible to configure an /etc/auto.home file to make home directories available from a remote shared network server.

## SELF TEST

The following questions will help measure your understanding of the material presented in this chapter. As no multiple-choice questions appear on the Red Hat exams, no multiple-choice questions appear in this book. These questions exclusively test your understanding of the chapter. Getting results, not memorizing trivia, is what counts on the Red Hat exams. There may be more than one answer to many of these questions.

### Storage Management and Partitions

1. What **fdisk** command lists configured partitions from all attached hard drives?

   ________________________________________________

2. After creating a swap partition, what command activates it?

   ________________________________________________

### Filesystem Formats

3. What is the primary advantage of a journaling filesystem such as ext4?

   ________________________________________________

4. What command formats /dev/sdb3 to the default Red Hat filesystem format?

   ________________________________________________

### Basic Linux Filesystems and Directories

5. What directory is mounted on a partition separate from the top-level root directory in the default RHEL 6 installation?

   ________________________________________________

6. Name three directories just below /, not suitable for mounting separately from the volume with the top-level root directory.

   ________________________________________________

## Logical Volume Management (LVM)

**7.** Once you've created a new partition and set it to the <A>Logical Volume Management filetype, what command adds it as a PV?

____________________________________________________________

**8.** Once you've added more space to an LV, what command would expand the formatted filesystem to fill the new space?

____________________________________________________________

## Volume Encryption with the Linux Unified Key Setup

**9.** What command configures the /dev/sdc1 partition with a LUKS passphrase?

____________________________________________________________

## Filesystem Management

**10.** To change the mount options for a local filesystem, what file would you edit?

____________________________________________________________

**11.** What would you add to the /etc/fstab file to set up access control lists on the /home directory, mounted on partition /dev/vda6, with other default options? Assume you can't find the UUID of /dev/vda6. Assume a dump value of 1 and a filesystem check order of 2.

____________________________________________________________

## The Automounter

**12.** If you've started the **autofs** daemon and want to read the list of shared NFS directories from the server1.example.com computer, what automounter-related command would you use?

____________________________________________________________

**13.** Name three configuration files associated with the default installation of the automounter on RHEL 6.

____________________________________________________________

## LAB QUESTIONS

Several of these labs involve format exercises. You should do these exercises on test machines only. The instructions in these labs delete all of the data on a system. The second Lab sets up KVM for this purpose. However, some readers may not have hardware that supports KVM. Options to KVM include virtual machine solutions such as VMware, available from www.vmware.com or Virtualbox, open-source edition, available from www.virtualbox.org.

Red Hat presents its exams electronically. For that reason, most of the labs in this and future chapters are available from the CD that accompanies the book, in the Chapter5/ subdirectory. It's available in .doc, .html, and .txt formats, in the filename starting with 56505-labs. In case you haven't yet set up RHEL 6 on a system, refer to the first lab of Chapter 2 for installation instructions. However, the answers for each lab follows the self test answers for the fill-in-the-blank questions.

The answers for each lab follows the self test answers for the fill-in-the-blank questions.

## SELF TEST ANSWERS

### Storage Management and Partitions

1. The **fdisk** command that lists configured partitions from all attached hard drives is **fdisk -l**.
2. After creating a swap partition, the **swapon *devicename*** command activates it; just substitute the device file associated with the volume (such as /dev/sda1 or /dev/VolGroup00/LogVol03) for ***devicename*.**

### Filesystem Formats

3. The primary advantage of a journaling filesystem such as ext4 is faster data recovery if power is suddenly cut.
4. The command that formats /dev/sdb3 to the default Red Hat filesystem format is **mkfs.ext4 /dev/sdb3**. The **mkfs -t ext4 /dev/sdb3** and **mke2fs -t ext4 /dev/sdb3** commands are also acceptable answers.

### Basic Linux Filesystems and Directories

5. The /boot directory is mounted separately from /.
6. There are many correct answers to this question; some directories not suitable for mounting separately from / include /bin, /dev, /etc, /lib, /root, /sbin, and /selinux. (In contrast, several directories are essentially shown as placeholders for mounting, including /media and /mnt. Other directories such as /proc and /sys are virtual filesystems; in other words, as they don't include files kept on storage media, they should not be mounted separately from / for other reasons.)

### Logical Volume Management (LVM)

7. Once you've created a new partition and set it to the Logical Volume Management filetype, the command that adds it as a PV is **pvcreate**. For example, if the new partition is /dev/sdb2, the command is **pvcreate /dev/sdb2**.
8. Once you've added more space to an LV, the command that would expand the formatted filesystem to fill the new space is **resize2fs**.

### Volume Encryption with the Linux Unified Key Setup

9. The command that encrypts the /dev/sdc1 partition with a passphrase is **cryptsetup luksFormat /dev/sdc1**.

### Filesystem Management

10. To change the mount options for a local filesystem, edit /etc/fstab.
11. Since the UUID is unknown, you'll need to use the device file for the volume, in this case, /dev/vda6. Thus, the line to be added to /etc/fstab is:

```
/dev/vda6  /home    ext4     defaults,acl,usrquota        1 2
```

### The Automounter

12. If you've started the **autofs** daemon and want to read the list of shared NFS directories from the first.example.com computer, the automounter-related command you'd use to list those directories is **/etc/auto.net server1.example.com**.
13. The configuration files associated with the default installation of the automounter include auto.master, auto.misc, auto.net, and auto.smb, all in the /etc directory, as well as /etc/sysconfig/autofs. The /etc/auto.home directory is not a part of the default installation.

## LAB ANSWERS

One of the assumptions with these labs is that where a directory such as /test1 is specified, that you create it before mounting a volume device file on it, or including it in a key configuration file such as /etc/fstab. Otherwise, you'll encounter possibly unexpected errors.

### Lab 1

1. It shouldn't matter whether partitions are created in the **fdisk** or the **parted** utility. As long as the partition types were correctly configured, you should see one Linux partition and one Linux swap partition in the configured hard drives, in the output to the **fdisk -l** command.
2. If you're confused about what UUID to use in /etc/fstab, run the **blkid** command. If the given partitions have been properly formatted (with the **mkfs.ext4** and **mkswap** commands), you'll see the UUID for the new partitions in the output to **blkid**.

3. You should be able to test the configuration of a new partition and directory in /etc/fstab with the **mount -a** command. Then a **mount** command by itself should be able to confirm appropriate configuration in /etc/fstab.
4. You should be able to confirm the configuration of a new swap partition in the output to the **cat /proc/swaps** command. If you add the space allocated to each swap partition, you should also be able to verify the result in the **Swap** line associated with the **top** command.
5. Remember, all changes should survive a reboot. For the purpose of this lab, you may want to reboot this system to confirm this. However, reboots take time; if you have multiple tasks during an exam, you may want to wait until completing as much as possible before rebooting a system.

## Lab 2

This discussion is focused on how you can verify the results of this lab. Even if you've configured the exact spare partitions described in this lab and followed exact instructions, it's quite possible that your LV won't be exactly 900M. Some of that variance comes from the differences between cylinders and sectors; others come from the difference between base 2 and base 10 numbers. Don't panic; the people who grade Red Hat exams have made allowances for that variance. The same proviso applies to Lab 3 as well.

Keep in mind that logical volumes are based on appropriately configured partitions, set up as PVs, collected into a VG, and then subdivided into an LV. That LV is then formatted and then mounted on an appropriate directory; for the purpose of this lab, that directory is /test2. The UUID of that formatted volume can then be used to set up that LV as a mount in the /etc/fstab file.

1. To verify appropriate partitions prepared for logical volumes, run the **fdisk -l** command. Appropriate partitions should appear with the "Linux LVM" label.
2. To verify the configuration of appropriate PVs, run the **pvs** command. The output should report the devices and space allocated to PVs.
3. To verify the configuration of an appropriate VG, run the **vgs** command. The output should list the VG created during the lab from available PVs, including the space available.
4. To verify the configuration of an appropriate LV, run the **lvs** command. The output should list the LV, the VG from where it was created, and the amount of space allocated to that LV.
5. To verify the UUID of the newly formatted volume, run either of the following commands:

```
# dumpe2fs /dev/volgroup1/logvol1 | grep UUID
# blkid
```

6. If the /etc/fstab file is properly configured, you should be able to run the **mount -a** command. Then you should see the /dev/mapper/volgroup1-logvol1 (or whatever the name of the LV device file is) mounted on the /test2 directory.
7. As with Lab 1, all changes should survive a reboot. At some point, you'll want to reboot the local system to check for success or failure of this and other labs.

## Lab 3

Based on the information from Lab 2, you should already know what the size of the current LV is. The associated **df** command should confirm the result; the **df -m** command, with its output in MB, could help.

The key command in this lab is **resize2fs**. While there are a number of excellent switches available, all you really need with that command is the device file for the LV along with the desired size. As with Lab 2, the result can be confirmed after an appropriate **mount** and **df** command. However, full success can't be confirmed until the /etc/fstab file has been revised, and the **mount -a** command run to confirm that everything in /etc/fstab, including the new line, actually works. Of course, based on the instructions in the lab, you could focus with the **mount /test3** command. Since the remaining information is in /etc/fstab, the device file for the mount should not be required.

## Lab 4

There are several steps associated with confirming the successful creation of a LUKS-encrypted filesystem. The **blkid** command should specify crypto_LUKS in the output associated with those volumes that have been encrypted. There should also be a /dev/mapper device associated with that volume. That /dev/mapper device file should be mountable; if so, it confirms an appropriate format for the encrypted filesystem.

If you've encountered problems with many of these commands, you may need to load associated modules with a command like **modprobe dm_crypt**.

## Lab 5

The configuration of a LUKS-encrypted filesystem in the /etc/fstab file can be a bit tricky. A number of LUKS-related commands won't work unless the dm_crypt module is loaded. If you made certain kinds of mistakes in Lab 4, the dm_crypt module won't be reloaded on the next reboot.

If you're uncertain about the process, first try to use the associated /dev/mapper device in the /etc/fstab file. Afterwards, find the appropriate UUID, as shown in the output to the **blkid** command. And if you forget to configure the /etc/crypttab file, the system won't ask for the passphrase during the boot process, which would make the encryption of the protected filesystem somewhat worthless. Just be sure to include the **none** directive after the name and UUID of the unencrypted device.

However, if everything works, the system should prompt for the passphrase during the boot process, and then mount the encrypted volume automatically.

## Lab 6

The configuration of the automounter on a shared NFS directory is easier than it looks. Before you begin, make sure the shared NFS directory is available from the remote computer with the **showmount -e remoteNFSipaddr** command, where remoteNFSipaddr is the IP address of the remote NFS server. If that doesn't work, you may have skipped a step described in the lab. For more information on NFS servers, refer to Chapter 16.

Of course, there's the CD/DVD. If the automounter is running and a CD/DVD drive is in the appropriate location, you should be able to read the contents of that drive with the **ls /misc/cd** command. That matches the default configuration of the /etc/auto.master and /etc/auto.misc files.

As for the shared NFS directory, there are two approaches. You could modify the following commented sample NFS configuration directive. Of course, you'd have to at least change ftp.example.org to the name or IP address of the NFS server, and /pub/linux to /tmp (or whatever is the name of the directory being shared.

```
linux  -ro,soft,intr  ftp.example.org:/pub/linux
```

Or you could just directly take advantage of the **/etc/auto.net** script. For example, if the remote NFS server is on IP address 192.168.122.50, run the following command:

```
# /etc/auto.net 192.168.122.50
```

You should see the /tmp directory shared in the output. If so, you'll be able to access it more directly with the following command:

```
# ls /net/192.168.122.50/tmp
```

If you really want to learn the automounter, try modifying the aforementioned directive in the /etc/auto.misc configuration file. Assuming the automounter is already running, you can make sure the automounter rereads the applicable configuration files with the **/etc/init.d/autofs reload** command.

If you use the same first directive in the aforementioned line, you'll be able to use the automounter to access the same directory with the **ls /misc/linux** command.