



Chapter 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)	Q&A	Self Test

Linux installation is easy, at least for anyone serious about Red Hat certification.

However, most administrators have to maintain existing systems. Critical skills related to filesystems include adding new partitions, creating logical volumes, mounting filesystems, 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 that are not accessed frequently, should be mounted only on a temporary basis; that is the job of the automounter.

INSIDE THE EXAM

Inside the Exam

Some of the RHCSA objectives listed in this chapter overlap and may be 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 **fdisk** or **parted** to create an MBR-style partition. However, you should know that the Red Hat implementation of **fdisk** does not support GPT partitions, whereas **gdisk** and **parted** do. Make sure that appropriate partitions meet the requirements of the exam.

The current RHCSA objectives include the following related requirements:

- Add new partitions, logical volumes and swap to a system non-destructively
- List, create, delete partitions on MBR and GPT disks

Logical Volumes

Partitions and disks are 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 physical volumes:

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

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 **vfat**, **ext4**, and **xfs** 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. Although 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, three tools still predominate for administrators who need to create and manage partitions: **fdisk**, **gdisk**, and **parted**. While these tools are primarily applied to local hard disks, they can also be used for other media such as drives attached over a network.

Current System State

Before using the **fdisk**, **gdisk**, or **parted** utilities to create or modify a partition, do check currently available free space along with currently mounted filesystems. The following commands make it easy: **df** and **fdisk -l**. The following example in Figure 6-1 illustrates how the **df** command displays the total, used, and available free space on all currently mounted filesystems.

 **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 filesystems, tmpfs, and devtmpfs), they add up to about 11.5GB of allocated space. If the hard drive is larger, unallocated space may be used for another partition. Partitions can be

FIGURE 6-1

Disk space usage shown by the df command

```
[root@server1 ~]# df
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/mapper/rhel_server1-root	10229760	3387916	6841844	34%	/
devtmpfs	499652	0	499652	0%	/dev
tmpfs	508936	92	508844	1%	/dev/shm
tmpfs	508936	7120	501816	2%	/run
tmpfs	508936	0	508936	0%	/sys/fs/cgroup
/dev/mapper/rhel_server1-home	1020588	70580	950008	7%	/home
/dev/vda1	508588	121244	387344	24%	/boot

FIGURE 6-2

Output of the
df command in
human-readable
format

```
[root@server1 ~]# df -h
Filesystem                Size      Used Avail Use% Mounted on
/dev/mapper/rhel_server1-root 9.8G    3.3G    6.6G   34% /
devtmpfs                   488M          0   488M    0% /dev
tmpfs                       498M    148K    497M    1% /dev/shm
tmpfs                       498M    7.0M    491M    2% /run
tmpfs                       498M          0   498M    0% /sys/fs/cgroup
/dev/mapper/rhel_server1-home 997M     74M    924M    8% /home
/dev/vdal                  497M    119M    379M   24% /boot
[root@server1 ~]#
```

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 filesystems, such as /home, /tmp, and /var.

To print the partition sizes in a more readable format, you can use the **-h** command option shown in Figure 6-2.

The second command, **mount**, lists the format and mount options for each filesystem. From the figure, examine the partition represented by device /dev/mapper/rhel_server1-home. Note how it is mounted on the /home directory with the xfs file type. It separates the home directories of regular users in a dedicated partition:

```
[root@server1 ~]# mount | grep home
/dev/mapper/rhel_server1-home on /home type xfs ↵
(rw,relatime,seclabel,attr2,inode64,noquota)
```

If the output of the **mount** command confuses you, consider the **findmnt** command, which prints all mounted filesystems in a tree-like format, as shown in Figure 6-3.

In the output, note the presence of “special filesystems” such as proc and sysfs, which we cover later in this chapter.

The fdisk Utility

The **fdisk** utility is common to many operating systems. The Mac OS has a fully featured version of **fdisk**. Older versions of Microsoft Windows have a simplified version of **fdisk**.

Although the Linux implementation of **fdisk** includes a wide variety of commands, you need to know only the few discussed here.

fdisk works with partitions created using the traditional Master Boot Record (MBR) partitioning scheme. On newer systems that run UEFI firmware rather than a traditional BIOS, you may see a different partitioning standard: the GUID Partition Table (GPT). The **fdisk** support of GPT is considered experimental. The preferred tools to manage GPT partitions are **gdisk** and **parted**.

FIGURE 6-3 Output of the findmnt command

```
[root@server1 ~]# findmnt
TARGET                                     SOURCE      FSTYPE     OPTIONS
/                                          /dev/mapper/rhel_server1-root
                                          xfs        rw,relatime,seclabel,attr2,inode64,n
/proc                                     proc        rw,nosuid,nodev,noexec,relatime
├─/proc/sys/fs/binfmt_misc               systemd-1   autofs     rw,relatime,fd=33,pgrp=1,timeout=300
├─/proc/fs/nfsd                          sunrpc     nfsd       rw,relatime
├─/sys                                   sysfs      sysfs     rw,nosuid,nodev,noexec,relatime,secl
├─/sys/kernel/security                  securityfs securityf  rw,nosuid,nodev,noexec,relatime
├─/sys/fs/cgroup                       tmpfs      tmpfs     rw,nosuid,nodev,noexec,seclabel,mode
├─├─/sys/fs/cgroup/systemd              cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,xatt
├─├─/sys/fs/cgroup/cpuset               cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,cpus
├─├─/sys/fs/cgroup/cpu,cpuacct          cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,cpua
├─├─/sys/fs/cgroup/memory               cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,memo
├─├─/sys/fs/cgroup/devices              cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,devi
├─├─/sys/fs/cgroup/freezer              cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,free
├─├─/sys/fs/cgroup/net_cls              cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,net_
├─├─/sys/fs/cgroup/blkio                cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,blkio
├─├─/sys/fs/cgroup/perf_event            cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,perf
├─├─/sys/fs/cgroup/hugetlb              cgroup     cgroup    rw,nosuid,nodev,noexec,relatime,huge
├─/sys/fs/pstore                       pstore     pstore    rw,nosuid,nodev,noexec,relatime
├─/sys/kernel/config                   configfs   configfs  rw,relatime
├─/sys/fs/selinux                       selinuxfs  selinuxfs rw,relatime
├─/sys/kernel/debug                     debugfs    debugfs   rw,relatime
├─/sys/fs/fuse/connections              fusectl    fusectl   rw,relatime
├─/dev                                   devtmpfs   devtmpfs  rw,nosuid,seclabel,size=499652k,nr_i
├─├─/dev/shm                           tmpfs      tmpfs     rw,nosuid,nodev,seclabel
├─├─/dev/pts                           devpts     devpts    rw,nosuid,noexec,relatime,seclabel,g
├─├─/dev/hugepages                      hugetlbfs  hugetlbfs rw,relatime,seclabel
├─├─/dev/mqueue                        mqueue     mqueue    rw,relatime,seclabel
├─/run                                  tmpfs      tmpfs     rw,nosuid,nodev,seclabel,mode=755
├─├─/run/user/1000/gvfs                 gvfsd-fuse fuse.gvfs  rw,nosuid,nodev,relatime,user_id=100
└─/var/lib/nfs/rpc_pipefs              sunrpc     rpc_pipef rw,relatime
```

Start fdisk: Help and More

The following screen output lists commands that show how to start **fdisk**, 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 **fdisk -l** commands for clues.

When you start **fdisk**, type **m** to list basic **fdisk** commands:

```
# fdisk /dev/vda
Welcome to fdisk (util-linux 2.23.2).
```

Changes will remain in memory only, until you decide to write them. Be careful before using the write command.

```

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
  g   create a new empty GPT partition table
  G   create an IRIX (SGI) partition table
  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): q
#

```

A wide variety of commands are 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 a new drive is installed 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/vdb`, and `/dev/vdc`.



SATA, PATA, and SAS SCSI drives are all represented by device files such as `/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 does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0xcb0a51f1.

```

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 if you save your changes.

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 typically be large enough to fill the rest of the disk; 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 examine 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 in a drive formatted with the MBR scheme, you can have up to four primary partitions, which would correspond to numbers 1 through 4. One of the primary partitions can be configured as an extended partition. The remaining partitions are logical partitions, numbered 5 and above. With an extended partition, you can create a maximum of 12 logical 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 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.)

```
# fdisk /dev/vdb

Command (m for help): n
Command action
   p   primary partition (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default
2097151): +500M
Partition 1 of type Linux and of size 500 MiB is set
```

```

Command (m for help): a
Selected partition 1

Command (m for help): p
Disk /dev/vdb: 1073 MB, 1073741824 bytes, 2097152 sectors
...
  Device Boot      Start         End      Blocks   Id  System
/dev/vdb1    *        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 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 run **partprobe /dev/vdb**, the kernel will read the new partition table and you'd be able to use the newly created partition.

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
Selected partition 1
Hex code (type L to list all codes): L

```

The list of available partition identifiers, as shown in Figure 6-4, is impressive. Note how it's not limited to Linux partitions. However, as this book covers Linux, Table 6-1 lists associated partition types.

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

FIGURE 6-4 Linux partition types in fdisk

```

0 Empty                24 NEC DOS             81 Minix / old Lin bf Solaris
1 FAT12                27 Hidden NTFS Win 82 Linux swap / So c1 DRDOS/sec (FAT-
2 XENIX root           39 Plan 9              83 Linux           c4 DRDOS/sec (FAT-
3 XENIX usr            3c PartitionMagic     84 OS/2 hidden C: c6 DRDOS/sec (FAT-
4 FAT16 <32M          40 Venix 80286        85 Linux extended c7 Syrix
5 Extended             41 PPC PReP Boot     86 NTFS volume set da Non-FS data
6 FAT16               42 SFS                87 NTFS volume set db CP/M / CTOS / .
7 HPFS/NTFS/exFAT     4d QNX4.x             88 Linux plaintext de Dell Utility
8 AIX                 4e QNX4.x 2nd part 8e Linux LVM       df BootIt
9 AIX bootable        4f QNX4.x 3rd part 93 Amoeba          e1 DOS access
a OS/2 Boot Manag    50 OnTrack DM        94 Amoeba BBT      e3 DOS R/O
b W95 FAT32          51 OnTrack DM6 Aux  9f BSD/OS          e4 SpeedStor
c W95 FAT32 (LBA)    52 CP/M              a0 IBM Thinkpad hi eb BeOS fs
e W95 FAT16 (LBA)    53 OnTrack DM6 Aux  a5 FreeBSD        ee GPT
f W95 Ext'd (LBA)    54 OnTrackDM6        a6 OpenBSD         ef EFI (FAT-12/16/
10 OPUS              55 EZ-Drive          a7 NeXTSTEP        f0 Linux/PA-RISC b
11 Hidden FAT12       56 Golden Bow        a8 Darwin UFS       f1 SpeedStor
12 Compaq diagnost  5c Priam Edisk       a9 NetBSD           f4 SpeedStor
14 Hidden FAT16 <3    61 SpeedStor         ab Darwin boot      f2 DOS secondary
16 Hidden FAT16       63 GNU HURD or Sys  af HFS / HFS+       fb VMware VMFS
17 Hidden HPFS/NTF    64 Novell Netware    b7 BSDI fs          fc VMware VMKCORE
18 AST SmartSleep     65 Novell Netware    b8 BSDI swap         fd Linux raid auto
1b Hidden W95 FAT3    70 DiskSecure Mult  bb Boot Wizard hid fe LANstep
1c Hidden W95 FAT3    75 PC/IX             be Solaris boot     ff BBT
1e Hidden W95 FAT1    80 Old Minix
Hex code (type L to list all codes): █

```

TABLE 6-1 Linux Partition Types in fdisk

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 LVM for partitions used as physical volumes.
fd	Linux RAID; for partitions used as components of a RAID array.

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*.

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

```
# fdisk /dev/vdb
Command (m for help): p

Disk /dev/vdb: 1073 MB, 1073741824 bytes, 2097152 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 label type: dos
Disk identifier: 0x2e3c116d

Device      Boot      Start          End      Blocks      Id  System
/dev/vdb1                2048      1026047       512000      83   Linux
Command (m for help): d
Selected partition 1
Partition 1 is deleted
```

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.



If you remove a partition, the partition table on disk is modified to reflect the change, but the actual data on the partition isn't removed. This means that if you re-create the partition using the same layout (same start/end sectors), your data will still be there. It's worth trying this procedure in case you accidentally delete a partition by mistake.

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 6-4.

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, you could run the following sequence of commands to set up a new swap partition on the second hard drive. The details of what you see depend on the partitions you may have created. It'll be a 900MB swap space on the first primary partition (/dev/vdb1).

```
Command (m for help): n
Command action
    e   extended
    p   primary partition (1-4)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048): 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default
2097151): +900M
Partition 1 of type Linux and of size 900 MiB is set
```

Command (m for help): p

```
Disk /dev/vdb: 1073 MB, 1073741824 bytes, 2097152 sectors
...
```

Device	Boot	Start	End	Blocks	Id	System
/dev/vdb1		2048	1845247	921600	83	Linux

```
Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 82
Changed system type of partition 'Linux' to '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 configure RHEL to use that newly created swap partition.

The **gdisk** Utility

As we have illustrated in the previous section, the MBR partitioning scheme supports a maximum of 15 partitions for data (3 primary and 12 logical), plus an extended partition, which is simply a “container” for logical partitions. In contrast, the GPT partitioning scheme can hold up to 128 partitions.

Another limitation of MBR is the disk size. The MBR scheme uses 32-bit logical addresses, which support disk drives up to 2TB. On the other hand, the GPT format relies on 64-bit addresses, which support drives of up to 8 million terabytes!

Although you could use **fdisk** and select the **g** command to switch to a GPT partition table format, the **gdisk** utility is preferred for GPT partitions. If you’re familiar with **fdisk**, **gdisk** should seem familiar too. When you start **gdisk** on a disk with an MBR partition table, you will see the following warning:

```
[root@server1 ~]# gdisk /dev/vdb
...
*****
Found invalid GPT and valid MBR; converting MBR to GPT format.
THIS OPERATION IS POTENTIALLY DESTRUCTIVE! Exit by typing 'q' if
you don't want to convert your MBR partitions to GPT format!
*****
```

As suggested by the message, type **q** to exit; otherwise, you may lose any data on your drive. Once started, **gdisk** works in a similar way to **fdisk**. Type the question mark (?) to get a list of commands:

```
Command (? for help): ?
b          back up GPT data to a file
c          change a partition's name
d          delete a partition
i          show detailed information on a partition
l          list known partition types
n          add a new partition
o          create a new empty GUID partition table (GPT)
q          quit without saving changes
r          recovery and transformation options (experts only)
s          sort partitions
t          change a partition's type code
v          verify disk
w          write table to disk and exit
x          extra functionality (experts only)
?          print this menu
```

```
Command (? for help):
```

The next screen output shows the steps used to create a new 500MB partition (**n**) on the disk device `/dev/vdc`:

```
[root@server1 ~]# gdisk /dev/vdc
GPT fdisk (gdisk) version 0.8.6

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries

Command (? for help): n
Partition number (1-128, default 1): 1
First sector (34-2097118, default = 2048) or {+-}size{KMGT}: 2048
Last sector (2048-2097118, default = 2097118) or {+-}size{KMGT}:
+500M
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help): w
```

As with **fdisk**, the **gdisk** tool doesn't write the changes to disk until you type the write (**w**) command. At any time you can quit the utility without saving any changes with the quit (**q**) command.

The parted Utility

In its different forms, the **parted** utility is becoming increasingly popular. It's an excellent tool developed by the Free Software 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 more information, see www.gnu.org/software/parted.



In some ways, the parted utility may be more risky. For example, one of the authors of this book 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, there was a backup of this virtual system and it could be restored with little trouble.

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/vdb` drive is associated with the second virtual drive on a virtual machine. Your computer may have a different hard drive; you can check the output from the **df** and **fdisk -l** commands for clues.

As you can see in Figure 6-5, when **parted** is run, it opens its own command-line prompt. Enter **help** for a list of available commands.

A wide variety of commands are available at the **parted** interface. When compared to **fdisk** and **gdisk**, **parted** can do more in some ways, as you will see in the next sections.

Using parted: In a Nutshell

At the **parted** command-line prompt, start with the **print** command to list the current partition table. Assuming sufficient unallocated space is available, you can then make a

FIGURE 6-5 The parted command options

```
(parted) help
  align-check TYPE N           check partition N for TYPE(min|opt)
  alignment
  help [COMMAND]
  COMMAND
  mklabel,mktable LABEL-TYPE  create a new disklabel (partition
  table)
  mkpart PART-TYPE [FS-TYPE] START END  make a partition
  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
  and END                      rescue a lost partition near START
  rm NUMBER                    delete partition NUMBER
  select DEVICE               choose the device to edit
  disk_set FLAG STATE        change the FLAG on selected device
  disk_toggle [FLAG]
  device                       toggle the state of FLAG on selected
  set NUMBER FLAG STATE      change the FLAG on partition NUMBER
  toggle [NUMBER [FLAG]]
  NUMBER                      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) █
```

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 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: btrfs, nilfs2, ext4, ext3, ext2, fat32, fat16, hfsx,
hfs+, hfs, jfs, swsusp, linux-swap(v1), linux-swap(v0), 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, linux-swap, linux-swap(new), linux-swap(old)
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.

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

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

```
Error: /dev/vdb: unrecognised disk label
```

Before you can do anything else with this drive, you need to create a label. As shown from the list of available commands, you can do so with the **mklabel** command. If you type **msdos**, the MBR-style partition scheme will be used. To use the GTP format, type **gpt** at the command prompt:

```
(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 you have selected the MBR partition scheme, you need to specify the partition type:

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

For **parted**, we used 1MB to start the partition at sector 2,048. Although we could have used “0MB” as the starting point, that would have generated a warning because the partition would have not been properly aligned at the 1MB boundary for best performance. Now review the results with the **print** command:

```
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

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

If this is the first partition you’ve created, the file system type column will be empty. However, for the purposes of this chapter, don’t exit from **parted** just yet.



The GUI parted tools (GParted and 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 do the following:

- 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/vdb10` 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** filesystem 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: Virtio Block Device (virtblk)
Disk /dev/vdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:
```

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

Now exit from **parted**. To use these partitions, you need to run commands such as **mkswap**, **swapon**, and **mkfs.xfs**. We cover these commands later in this chapter.

```
(parted) quit

# mkswap /dev/vdb2
# swapon /dev/vdb2
```

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

```
# mkfs.xfs /dev/vdb1
```

Using parted: Set Up a Different Partition Type

When a partition is created in **parted**, you can change its type with the **set** command. If you have a hard drive with unused partitions, open it with the **parted** command. For example, the following command opens the second virtual hard drive:

```
# parted /dev/vdb
```

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: Virtio Block Device (virtblk)
Disk /dev/vdb: 1074MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
  1      1049kB  500MB   499MB   primary  xfs          lvm
  2      501MB   1000MB  499MB   primary  linux-swap(v1)
```

You can use similar steps 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 7 system, first confirm the result. Exit from **parted** and then run the following commands:

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

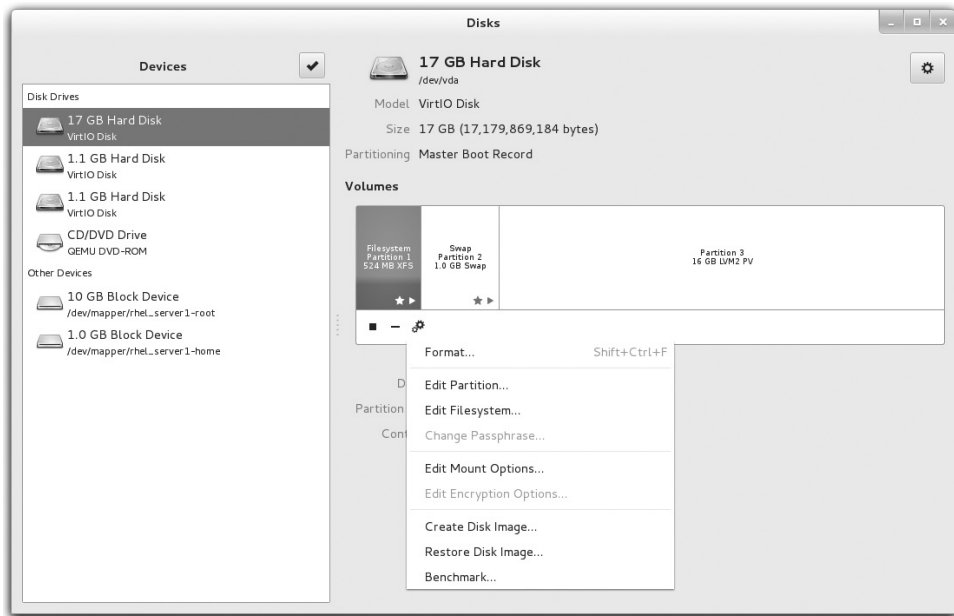
You'll see the **lvm** flag as shown previously from the **parted** command; you'll see the following confirmation in the output to the **fdisk** command:

```
Device Boot Start End Blocks Id System
/dev/vdb1 2048 976895 487424 8e 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**, **gdisk**, or **parted** for this purpose. You can choose to use all the 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

FIGURE 6-6 The Disk Utility

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 7 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 **`gnome-disks`** command.

The Disk Utility screen shown in Figure 6-6 depicts the baseline virtual machine created in Chapter 2; it lists the virtual hard drive, device `/dev/vda`, as well as its root and home partitions, the two additional drives, and the DVD drive.

The functionality includes the following clickable options from the settings menu (“gear” icon):

- **Format** On a drive, sets the MBR or GPT-style partition formats. On a partition, formats a partition for a number of filesystem formats.
- **Edit Partition** Sets the partition type, such as Linux swap or Linux LVM.
- **Edit Filesystem** Sets the filesystem label; labels were commonly used on RHEL 5.
- **Edit Mount Option** Configures the filesystem mount options, such as mount points and filesystem type.

- **Create Disk Image** Creates an image file with the content of a drive or partition.
- **Restore Disk Image** Restores the content of a drive from a disk image.
- **Benchmark** Allows measurements of read and write performance.
- **Unmount the Filesystem** Unmounts a filesystem. (This option appears as the “stop” icon.)
- **Delete Partition** Deletes a partition. (This option appears as the “minus” icon.)
- **Create Partition** Creates a new partition. (This option appears as the “plus” icon.)

Not all of these options appear in Figure 6-6; 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.

EXERCISE 6-1

Work with **fdisk** and **parted**

In this exercise, you’ll work with both the **fdisk** and **parted** utilities. It assumes you have a new empty drive, such as a drive on a virtual machine. For the purpose of this exercise, **fdisk** and **parted** will be used on drives `/dev/vdb` and `/dev/vdc`, respectively. Feel free to substitute accordingly. Do save the results of this work. You’ll use it for exercises that follow later in this chapter.

1. Run the **fdisk -l /dev/vdb** command to review the current status of the `/dev/vdb` drive (if your first disk drive has a different device name, substitute with the correct name, such as `/dev/sdb`.)
2. Open disk `/dev/vdb` with the **fdisk /dev/vdb** command.
3. Run the **p** command to display any previously configured partitions.
4. Create a new partition with the **n** command. If primary partitions are available, create one with the **p** command. If options for primary partition numbers are presented, select the first available.
5. 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, specify 10,000 here:

```
First sector (1845248-2097151, default 1845248): 10000
```

6. 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, specify 1,950,000 here:

```
Last sector, +sectors or +size{K,M,G} (1845248-2097151, default 2097151):
1950000
```

7. Run the **p** command again to review the result. Run the **w** command to write the result to disk.
 8. Review the result on the /dev/vdb disk with the **parted /dev/vdb print** command.
 9. Open the other available free disk (/dev/vdc) with the **parted /dev/vdc** command.
 10. 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.
 11. 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). Enter **xfs** as 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.
 12. Run the **quit** command to exit from **parted**.
 13. Run the **fdisk -l /dev/vdc** command to review the result.
 14. 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 XFS. Although many users enable other filesystems such as Btrfs, Red Hat may not support them.

Linux supports a rich variety of filesystems. Except for some old filesystems such as ext2, most filesystems incorporate features such as journal-based transactions, large storage support, delayed allocation, and complex algorithms to optimize read and write performance. In the following sections we split filesystems into two broad categories: “standard formatting” and journaling. While this is an oversimplification, it is sufficient to categorize the filesystem important to Linux.

The filesystems described in this book are just a subset of those that can be configured on a RHEL system. The Linux kernel makes it possible to set up more.

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 versions of the Linux operating systems used the Extended Filesystem (ext). In the twentieth century, Red Hat formatted its partitions to the Second Extended Filesystem (ext2). Starting with RHEL 5, Red Hat moved to the Third Extended Filesystem (ext3). For RHEL 6, Red Hat progressed to the Fourth Extended Filesystem (ext4). Both ext3 and ext4 are journaling filesystems.

The size of current filesystems has increased the importance of journaling because such filesystems are more resilient to failure. So in general, the nonjournaling filesystems described in Table 6-2 are legacy filesystems. Of course, filesystems such as ISO 9660 and swap are still in common use.

TABLE 6-2 Some Standard Filesystems

Filesystem Type	Description
ext	The first Linux filesystem, used only on early versions of the 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 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 filesystem.
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 and sys	Two Linux virtual filesystems. <i>Virtual</i> means that the filesystem doesn't occupy real disk space. Instead, files are created as needed. Used to provide information on kernel configuration and device status.
devpts	The Linux implementation of the Open Group's Unix98 PTY support.
tmpfs	A filesystem stored in memory. Used on RHEL 7 for the /run partition.

Journaling Filesystems

Journaling filesystems have two main advantages. First, they're 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 7, the default filesystem is XFS, a highly scalable, journal-based filesystem.

This isn't the only journaling filesystem options available, however. We list a few of the options commonly used for RHEL in Table 6-3. From this list, Red Hat officially supports only ext3, ext4, and XFS. At the time of writing, Btrfs is considered to be a "technology preview" and is not fully supported by Red Hat.

The Red Hat move to XFS is a testament to its use as a server operating system. For example, volumes formatted to XFS can theoretically be as large as 8 exabytes (EB). That's a serious increase over the maximum size of an ext4 volume: 16 terabytes (TB).

XFS supports a large number of concurrent operations, guarantees space for files, guarantees faster checks, and more. As XFS has been a part of the Linux kernel since 2004, it is proven technology.

Filesystem Format Commands

Several commands 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.ext3**, **mkfs.ext4**, and **mkfs.xfs**.

TABLE 6-3 Some Journaling Filesystems

Filesystem Type	Description
ext3	The default filesystem for RHEL 5.
ext4	The default filesystem for RHEL 6.
XFS	Developed by Silicon Graphics as a journaling filesystem, it supports very large files and features such as B-tree indexing and dynamic allocation inodes.
JFS	IBM's journaled filesystem, commonly used on IBM enterprise servers.
Btrfs	The B-tree filesystem was developed to offer a set of features comparable with Oracle ZFS. It offers some advanced features such as snapshots, storage pools, and compression.
NTFS	The current Microsoft Windows filesystem.

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 format 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 XFS filesystem using one of the following commands:

```
# mkfs -t xfs /dev/sdb5
# mkfs.xfs /dev/sdb5
```

You can format partitions, logical volumes, and RAID arrays to other filesystems. The options available in RHEL 7 include the following:

- **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.fat** (or **mkfs.vfat**, **mkfs.msdos**, **mkdosfs**) formats a partition to the Microsoft-compatible FAT filesystem; it does not create bootable filesystems. (All these commands are the same because they are symbolic links to **mkfs.fat**.)
- **mkfs.xfs** formats a volume to the RHEL 7 default XFS filesystem.
- **mkswap** sets up a Linux swap area.

These commands assume 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.

Swap Volumes

Although Linux can use swap files, the swap space is generally configured in properly formatted partitions or logical volumes. 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, swap volumes 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 rescue mode 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 xfs /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 -li** command to all four files, which are part of the `/sbin` directory.

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 then 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/vdb print** and **fdisk -l /dev/vdc** commands.
2. Format the partition created by the first drive with the **mkfs.ext2 /dev/vdb1** command. Review the current status of the volume with the **dumpe2fs -h /dev/vdb1 | grep features** command. What features do you see in the output? Save the output temporarily. One way to do so is to open a new command-line console. Check the system with the **fsck.ext2 /dev/vdb1** command.
3. Mount the newly formatted partition with the **mount /dev/vdb1 /mnt** command. Review the output with the **mount** command by itself. If the mount and format worked, you'll see the following output:

```
/dev/vdb1 on /mnt type ext2 (rw,relatime,seclabel)
```

4. Unmount the formatted partition with the **umount /mnt** command.

5. Run the **mkfs.ext4 /dev/vdb1** 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. Now on the other partition created in Exercise 6-1, apply the **mkfs.xfs /dev/vdc1** command.
 8. Mount the newly formatted partition on the **/mnt** directory and then run the **mount** command by itself. Can you confirm the filesystem of the **/dev/vdc1** 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/cdrom**. Detected devices are documented as files in the **/sys** 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 <http://refspecs.linuxfoundation.org/fhs.shtml>.

Separate Linux Filesystems

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. The way these components are organized on storage media is known as a filesystem. The FHS makes it easier for Linux distributions to adhere to a common directory structure.

Every FHS starts with the top-level root directory, also known by its symbol, the single forward slash (**/**). All the other directories shown in Table 6-4 are subdirectories of the root directory. Unless they are 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 importantly, not all listed directories can or should be mounted separately.

Mounted directories are often known as *volumes*, which can span multiple partitions if used with the Logical Volume Manager (LVM). While the root directory (**/**) is the top-level directory in the FHS, the root user's home directory (**/root**) is just a subdirectory.

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. On RHEL 7, it is a symbolic link to /usr/bin.
/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. On RHEL 7, this is a symbolic link to /usr/lib.
/lib64	Same as /lib, but includes 64-bit libraries. On RHEL 7, this is a symbolic link to /usr/lib64.
/media	The mount point for removable media, including DVDs and USB disk drives.
/misc	The standard mount point for local directories mounted via the automounter.
/mnt	A mount point for temporarily mounted filesystems.
/net	The standard mount point for network directories mounted via the automounter.
/opt	Common location for third-party application files.
/proc	A virtual filesystem listing information for 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.
/run	A tmpfs filesystem for files that should not persist after a reboot. On RHEL 7, this filesystem replaces /var/run, which is a symbolic link to /run.
/sbin	System administration commands. Don't mount this directory separately. On RHEL 7, this is a symbolic link to /usr/bin.
/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.

(Continued)

TABLE 6-4 Basic Filesystem Hierarchy Standard Directories (*continued*)

Directory	Description
/sys	Similar to the /proc filesystem. Used to expose information about devices, drivers, and some kernel features.
/tmp	Temporary files. By default, Red Hat Enterprise Linux deletes all files in this directory periodically.
/usr	Programs and read-only data. Includes many system administration commands, utilities, and libraries.
/var	Variable data, including log files and printer spools.



In Linux, the word *filesystem* has different meanings. It can refer to the FHS or to a format such as ext3. A filesystem mount point such as /var represents the directory on which a filesystem 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, /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 /dev, /etc, and /root. Files within these directories are essential to the smooth operation of Linux as an operating system. Although 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.

Files in the /proc and /sys directories are filled only during the boot process and disappear when a system is shut down, and as such they are stored in a special in-memory virtual filesystem.

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.

CERTIFICATION OBJECTIVE 6.04

Logical Volume Management (LVM)

Logical Volume Management (LVM, also known as the Logical Volume Manager) creates an abstraction layer between physical devices, such as disks and partitions, and volumes that are formatted with a filesystem.

LVM can simplify disk management. As an example, assume that the /home filesystem is configured on its own logical volume. If extra space is available on the volume group associated with /home, you can easily resize the filesystem. If no space is available, you can make more room by adding a new physical disk and allocate its storage capacity to the volume group. On LVM, volume groups are like storage pools and they aggregate together the capacity of multiple storage devices. Logical volumes reside on volume groups and can span multiple physical disks.



LVM is an important tool to manage the space available to different volumes.

Definitions in LVM

To work with LVM, you need to understand how partitions configured for that purpose are used. First, with the **fdisk**, **gdisk** or **parted** utilities, you need to create partitions configured to the LVM partition type. You can also use an entire disk device.

Once those partitions or disk devices are available, they need to be set up as physical volumes (PVs). That process initializes a disk or partition for use by LVM. Then, you create volume groups (VGs) from one or more physical volumes. Volume groups organize the physical storage in a collection of manageable disk chunks known as physical extents (PEs). With the right commands, you can then organize those PEs into logical volumes (LVs). Logical volumes are made of logical extents (LEs), which map to the underlining PEs. You can then format and mount the LVs. For those who are new to LVM, it may be important to break out each definition:

- **Physical volume (PV)** A PV is a partition or a disk drive initialized to be used by LVM.
- **Physical extent (PE)** A PE is a small uniform segment of disk space. PVs are split into PEs.
- **Volume group (VG)** A VG is a storage pool, made of one or more PVs.

- **Logical extent (LE)** Every PE is associated with an LE, and these PEs can be combined into a logical volume.
- **Logical volume (LV)** An LV is a part of a VG and is made of LEs. An LV can be formatted with a filesystem and then mounted on the directory of your choice.

You'll see this broken down in the following sections. 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 logical volume, 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.

Create a Physical Volume

The first step is to start with a physical partition or a hard disk drive. Based on the discussion earlier in this chapter, you should be able to set up partitions 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 *volume group*:

```
# vgcreate volume group /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 *volume group* VG with the following command:

```
# vgextend volume group /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 that new logical volume on a directory.

But this isn't useful if you don't know how much space is associated with each PE. You can use the **vgdisplay** command to display the size of the PEs, or specify the size with the **-s** option of the **vgcreate** command when you initialize the volume group. Alternatively, you can use the **-L** switch to set a size in MB, GB, or another unit of measure. 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 logical volume gets formatted and mounted 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

A wide variety of LVM commands related to PVs, LVs, and VGs are available. 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.

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 6-5 Physical Volume Management Commands

Physical Volume Command	Description
pvchange	Changes attributes of a PV: the pvchange -x n /dev/sda10 command disables the allocation of PEs from the <code>/dev/sda10</code> partition.
pvck	Checks the consistency of a physical volume's metadata.
pvcreate	Initializes a disk or partition as a PV; the partition should be flagged with the LVM file type.

(Continued)

TABLE 6-5 Physical Volume Management Commands (*continued*)

Physical Volume Command	Description
pvdisplay	Displays currently configured PVs.
pvmove	Moves PEs in a VG from the specified PV to free locations on other PVs; prerequisite to disabling a PV. One example: pvmove /dev/sda10 .
pvremove	Removes a given PV from a list of recognized volumes: 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	Scans disks for physical volumes.

TABLE 6-6 Volume Group Commands

Volume Group Command	Description
vgcfgbackup vgcfgrestore	Used to back up and restore the metadata associated with LVM; by default, the backup files are in the /etc/lvm directory.
vgchange	Similar to pvchange , this command allows you to change the configuration settings of a VG. For example, vgchange -a y enables all local VGs.
vgck	Checks the consistency of a volume group metadata.
vgconvert	Supports conversions from LVM1 systems to LVM2: vgconvert -M2 VolGroup00 converts VolGroup00 to the LVM2 metadata format.
vgcreate	Creates a VG, from one 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 vgimport	Used to export and import unused VGs from those available; the vgexport -a command exports all inactive VGs.
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.

TABLE 6-6 Volume Group Commands (*continued*)

Volume Group Command	Description
vgreduce	The vgreduce vgroup00 /dev/sda11 command removes the /dev/sda11 PV from vgroup00, assuming /dev/sda11 is unused.
vgremove	The vgremove vgroup00 command removes vgroup00, assuming it has no LVs assigned to it.
vgrename	Allows the renaming of LVs.
vgs	Displays basic information on configured VGs.
vgscan	Scans all devices for VGs.
vgsplit	Splits a volume group.

TABLE 6-7 Logical Volume Commands

Logical Volume Command	Description
lvchange	Similar to pvchange , this command changes the attributes of an LV: for example, the lvchange -a n vgroup00/lvol00 command disables the use of the LV labeled lvol00.
lvconvert	Converts a logical volume between different types, such as linear, mirror, or snapshot.
lvcreate	Creates a new LV in an existing VG. For example, lvcreate -l 200 volume01 -n lvol01 creates lvol01 using 200 extents in the VG named volume01.
lvdisplay	Displays currently configured LVs.
lvextend	Adds space to an LV: the lvextend -L 4G /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 the LV lvol01 from VG volume01.
lvrename	Renames an LV.
lvresize	Resizes an LV; can be done by -L for size. For example, lvresize -L +4GB volume01 /lvol01 adds 4GB to the size of lvol01.
lvs	Lists all configured LVs.
lvscan	Scans for all LVs.

FIGURE 6-7

Configuration of a
volume group (VG)

```
[root@server1 ~]# vgdisplay
--- Volume group ---
VG Name                rhel_server1
System ID
Format                 lvm2
Metadata Areas         1
Metadata Sequence No   3
VG Access               read/write
VG Status               resizable
MAX LV                 0
Cur LV                 2
Open LV                 2
Max PV                 0
Cur PV                 1
Act PV                 1
VG Size                 14.53 GiB
PE Size                 4.00 MiB
Total PE                3720
Alloc PE / Size         2750 / 10.74 GiB
Free PE / Size           970 / 3.79 GiB
VG UUID                oYuR2x-uaUH-AZsZ-McNz-92Jh-qfYk-Ma0FDT

[root@server1 ~]# █
```

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 6-7 illustrates the configuration of VG `rhel_server1`.

Although a number of **lvm*** commands are installed, just four of them are active: **lvm**, **lvmconf**, **lvmdiskscan**, and **lvmdump**. Other **lvm*** commands are obsolete or not implemented yet. 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 physical volumes. Finally, the **lvmdump** command sets up a configuration report in the root administrative user's home directory (`/root`).

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 the filesystem associated with the LV. As an example, you can use a command similar to the following:

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

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

```
# 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 an 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 the Red Hat exams. But do you really want to risk user data in practice?)
2. Extend the VG to include new partitions configured to the appropriate type. For example, to add /dev/sdd1 to the vg_00 VG, run the following command:

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

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

```
# vgdisplay vg_00
```

4. 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
```

5. The **lvextend** command can increase the space allocated to an LV in KB, MB, GB, or even TB. For example, you could specify a 2GB LV with the following command:

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

If you prefer to specify the extra space to be added rather than the total space, you can use the syntax in the following example, which adds 1GB to the logical volume:

```
# lvextend -L +1G /dev/vg_00/lv_00
```

6. Resize the formatted volume with the **xfs_growfs** command (or with **resize2fs**, if it is an ext2/ext3/ext4 filesystem). If you're using the entire extended LV, the command is simple:

```
# xfs_growfs /dev/vg_00/lv_00
```

7. 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.xfs -f /dev/vg_00/lv_00
```

8. In either case, you’d finish the process by checking the new filesystem size with the **df** command:

```
# df -h
```

CERTIFICATION OBJECTIVE 6.05

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 to customize that file for local, remote, and removable filesystems.

The `/etc/fstab` File

To look at the contents of the `/etc/fstab` file, run the **cat `/etc/fstab`** command. From the example shown in Figure 6-8, different filesystems are configured on each line.

In RHEL 7 the default is to use UUIDs to mount non-LVM filesystems. As you’ll see in the next section, UUIDs can represent a partition, a logical volume, or a RAID array. In all cases, volumes should be formatted to the filesystem noted on each line and are mounted on the directory listed in the second column. The advantage of UUIDs and logical volume devices is that they are unique, whereas device names such as `/dev/sdb2` may change after a reboot, depending on the order in which the disks are initialized.

But to some extent, UUIDs are beside the point. As shown in Figure 6-8, six fields are 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 6-9.

FIGURE 6-8 Sample /etc/fstab

```
[root@server1 ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Mon Feb  2 17:41:03 2015
#
# 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
#
/dev/mapper/rhel_server1-root /          xfs     defaults        1 1
UUID=c89968bc-acc5-4d60-8deb-97542cb766c6 /boot          xfs     defaults        1 2
/dev/mapper/rhel_server1-home /home          xfs     defaults        1 2
UUID=9d37eaf0-2c0b-4e57-b05f-87c2e21d3a95 swap          swap     defaults        0 0
[root@server1 ~]#
```

FIGURE 6-9 Sample /etc/mtab

```
[root@server1 ~]# cat /etc/mtab
rootfs / rootfs rw 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
sysfs /sys sysfs rw,seclabel,nosuid,nodev,noexec,relatime 0 0
devtmpfs /dev devtmpfs rw,seclabel,nosuid,size=499652k,nr_inodes=124913,mode=755 0 0
securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev/shm tmpfs rw,seclabel,nosuid,nodev 0 0
devpts /dev/pts devpts rw,seclabel,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,seclabel,nosuid,nodev,mode=755 0 0
tmpfs /sys/fs/cgroup tmpfs rw,seclabel,nosuid,nodev,noexec,mode=755 0 0
cgroup /sys/fs/cgroup/systemd cgroup rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/usr/lib/systemd/systemd-cgroups-agent,name=systemd 0 0
pstore /sys/fs/pstore pstore rw,nosuid,nodev,noexec,relatime 0 0
cgroup /sys/fs/cgroup/cpuset cgroup rw,nosuid,nodev,noexec,relatime,cpuset 0 0
cgroup /sys/fs/cgroup/cpu,cpuacct cgroup rw,nosuid,nodev,noexec,relatime,cpuacct,cpu 0 0
cgroup /sys/fs/cgroup/memory cgroup rw,nosuid,nodev,noexec,relatime,memory 0 0
cgroup /sys/fs/cgroup/devices cgroup rw,nosuid,nodev,noexec,relatime,devices 0 0
cgroup /sys/fs/cgroup/freezer cgroup rw,nosuid,nodev,noexec,relatime,freezer 0 0
cgroup /sys/fs/cgroup/net_cls cgroup rw,nosuid,nodev,noexec,relatime,net_cls 0 0
cgroup /sys/fs/cgroup/blkio cgroup rw,nosuid,nodev,noexec,relatime,blkio 0 0
cgroup /sys/fs/cgroup/perf_event cgroup rw,nosuid,nodev,noexec,relatime,perf_event 0 0
cgroup /sys/fs/cgroup/hugetlb cgroup rw,nosuid,nodev,noexec,relatime,hugetlb 0 0
configfs /sys/kernel/config configfs rw,relatime 0 0
/dev/mapper/rhel_server1-root / xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
selinuxfs /sys/fs/selinux selinuxfs rw,relatime 0 0
systemd-1 /proc/sys/fs/binfmt_misc autofs rw,relatime,fd=33,prgrp=1,timeout=300,minproto=5,maxp
roto=5,direct 0 0
hugetlbfs /dev/hugepages hugetlbfs rw,seclabel,relatime 0 0
debugfs /sys/kernel/debug debugfs rw,relatime 0 0
mqueue /dev/mqueue mqueue rw,seclabel,relatime 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw,relatime 0 0
sunrpc /proc/fs/nfsd nfsd rw,relatime 0 0
/dev/mapper/rhel_server1-home /home xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
/dev/vda1 /boot xfs rw,seclabel,relatime,attr2,inode64,noquota 0 0
fusectl /sys/fs/fuse/connections fusectl rw,relatime 0 0
gvfsd-fuse /run/user/1000/gvfs fuse.gvfsd-fuse rw,nosuid,nodev,relatime,user_id=1000,group_id=
1000 0 0
/dev/sr0 /run/media/alex/RHEL-7.0\040Server.x86_64 iso9660 ro,nosuid,nodev,relatime,uid=1000,g
id=1000,icharset=utf8,mode=0400,dmode=0500 0 0
[root@server1 ~]#
```

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 the device path.
Mount Point	Notes the directory where the filesystem will be mounted.
Filesystem Format	Describes the filesystem type. Valid filesystem types are xfs, ext2, ext3, ext4, msdos, vfat, iso9660, nfs, smb, swap, and many others.
Mount Options	Covered in the following section.
Dump Value	Either 0 or 1. If you use the dump command to back up filesystems, this field controls which filesystems need to be dumped.
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.

Note the differences, especially the use of the device file in place of UUIDs, and the presence of virtual filesystems such as tmpfs and sysfs, which are discussed later in this chapter.

When adding a new partition, you could just add the device file associated with the partition or logical volume to the first column.

Universally Unique Identifiers in /etc/fstab

In /etc/fstab, note the focus on UUIDs, short for Universally Unique Identifiers. Every formatted volume has a 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 with the name of the device as an argument. The output will give the UUID of the device. As an example, to retrieve the UUID of the “root” logical volume in the “rhel_server1” volume group, run the following command:

```
# blkid /dev/rhel_server1/root
/dev/rhel_server1/root: UUID="2142e97a-dbec-495c-b7d9-1369270089ff" ↵
TYPE="xfs"
```

Alternatively, you could use the **xfs_admin** and **dumpe2fs** commands for the XFS and ext2/ext3/ext4 filesystems, respectively; for example, the following command identifies the UUID associated with the noted LV:

```
# xfs_admin -u /dev/rhel_server1/root
```

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

```
# xfs_admin -u /dev/vda1
```

Of course, the same is true if you have a configured and formatted ext volume, with a command such as the following:

```
# dumpe2fs /dev/mapper/rhel_server1-test | grep UUID
```

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 `/dev/mapper/rhel_server1-root` volume is mounted on the top-level root directory in read-write mode and formatted to the xfs filesystem:

```
/dev/mapper/rhel_server1-root on / type xfs ↵
(rw,relatime,seclabel,attr2,inode64,noquota)
```

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 mount all filesystems currently configured in the `/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 `/boot` directory in read-only mode:

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

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

```
/dev/vda1 on /boot type xfs (ro,relatime,seclabel,attr2,inode64,noquota)
```

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 7 ISO file on the /mnt directory:

```
# mount -o loop rhel-server-7.0-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**. Although 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. 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	All I/O is done asynchronously on this filesystem.
atime	Updates the inode access time every time the file is accessed.
auto	Can be mounted with the mount -a command.
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	Does not update the inode access time every time the file is accessed.
noauto	Requires explicit mounting. This is a common option for CD drives and removable media.
nodev	Device files on this filesystem are not read or interpreted.
noexec	Binaries (compiled programs) cannot be run on this filesystem.
nosuid	Disallows setuid and setgid permissions on this filesystem.
nouser	Only root users are allowed to mount the specified filesystem.

TABLE 6-9 Options for the mount Command and /etc/fstab (*continued*)

Mount Option	Description
remount	Remounts a currently mounted filesystem.
ro	Mounts the filesystem as read-only.
rw	Mounts the filesystem as read/write.
suid	Allows setuid and setgid permissions on programs on this filesystem.
sync	All I/O is done synchronously on this filesystem.
user	Allows non-root users to mount this filesystem. By default, this also sets the noexec , nosuid , and nodev options.

Virtual Filesystems

This section describes some of the virtual filesystems used by RHEL 7 and listed in /etc/mtab. Here are the most common:

- **tmpfs** This virtual memory filesystem uses both RAM and swap space.
- **devpts** This filesystem relates to pseudo-terminal devices.
- **sysfs** This 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.
- **proc** This filesystem is especially useful because 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.
- **cgroups** This filesystem is associated with the control group feature of the Linux kernel, which allows you to set limits on system resource usage for a process or a group of processes.

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 UUID, or a LVM device such as

`/dev/mapper/NewVol-NewLV` or `/dev/NewVol/NewLV`. Make sure the device file reflects the new volume you've created, the intended mount directory (such as `/special`), and the filesystem format you've applied (such as `xfs`).

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 such as **noauto**, but in general it's not standard in RHEL to set up removable media in `/etc/fstab`.

To read removable media such as smartcards and CD/DVDs, RHEL partially automates the mounting of such media in the GNOME Desktop Environment. Although the details of this process are not part of the Red Hat Exam Prep guide, the process is based on configuration files in the `/usr/lib/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?

Fortunately, Linux addresses this 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, it provides the **blkid** command. Try it. 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 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 because 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; for example, 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 a unique UUID. There's another option: rather than using static mounts for removable devices, you can rely on the automounter, as we will illustrate later in this chapter.

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 another 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 such as a power failure on the remote NFS server, you should specify in the mount options how you want the client to behave.

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 `/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 does not specify any options. You can try the following entry in `/etc/fstab`:

```
server1:/pub /share nfs rsize=65536,wsiz=65536,hard,udp 0 0
```

The **rsize** and **wsiz** variables determine the maximum size (in bytes) of the data to be read and written in each request. The **hard** directive specifies that the client will retry failed NFS requests indefinitely, blocking client requests potentially until the NFS server becomes available. Conversely, the **soft** option will cause the client to fail after a predefined amount of retransmissions, but at the cost of risking the integrity of the data. 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. Note that NFS version 4 requires TCP. In contrast, shared Samba directories use a different set of options. The following line is generally all that's needed for a share of the same directory and server:

```
//server1/pub /share cifs rw,username=user,password=pass, 0 0
```

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:

```
//server1/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 6.06

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. The automount daemon, also known as the automounter or **autofs**, can help. It can automatically mount a specific filesystem as needed. It can unmount a filesystem automatically after a fixed period of time.

Mounting via the Automounter

Once a partition is mounted manually with the **mount** command or via `/etc/fstab`, 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 hot-swappable removable drives.

Another issue: mounted NFS filesystems may cause problems if the remote computer fails or the connection is lost. Systems may slow down or even hang as the local system 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.



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
```

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

```
BROWSE_MODE="no"
```

A wide variety of additional settings are available, as commented in `/etc/sysconfig/autofs`.

/etc/auto.master

The standard `/etc/auto.master` file includes a series of directives, with four uncommented lines by default. The first refers to the `/etc/auto.misc` file as the configuration file for the `/misc` directory. The **/net -hosts** directive allows you to specify the host to automount a network directory, as specified in `/etc/auto.net`.

```
/misc /etc/auto.misc
/net -hosts
+dir:/etc/auto.master.d
+auto.master
```

In any case, these directives 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 users are logged off from 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 such as the following:

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

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 directives in comments in the `/etc/auto.misc` file. It's helpful to analyze this file in detail. We use the default RHEL version of this file. The first four lines are comments, which we 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 owner user ID permissions are not allowed (**nosuid**), and device files on this filesystem are not used (**nodev**).

A number of other directives are commented out, ready for use. Of course, you would have to delete the comment character (**#**) before using any of these configuration lines, 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 lines allows you to set up a `/misc/linux` mount point from a shared NFS directory, `/pub/linux`, from the `ftp.example.org` host:

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

The next line assumes that a filesystem is stored on the `/dev/hda1` partition. With this directive, you can automount the filesystem in `/misc/boot`.

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

The following three lines apply to a floppy disk drive. Don't laugh; virtual floppies are fairly easy to create and configure on most virtual machine systems. The first directive, set to an "auto" filesystem type, searches through `/etc/filesystems` to try to match what's on your floppy. The next two directives 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 line points to the first partition on the third SCSI drive. The **jaz** at the beginning suggests this is suitable for an old 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 XFS. As suggested earlier in this chapter, the **blkid** command can help identify available device files from removable systems such as 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 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 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 restart the automounter service with the following command:

```
# systemctl restart autofs
```

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
```

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:

```
# systemctl stop autofs
# systemctl start autofs
# systemctl restart autofs
# systemctl reload autofs
```

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:
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
```


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

```
# systemctl reload autofs
```

- 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
```

- Run the **ls /misc** command immediately. You should see the CD directory in the output.
- Wait at least five minutes and then 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 , gdisk or parted utility to create partitions, and then modify their partition types with the t or set command.
You want to set up a mount during the boot process based on the UUID.	Identify the UUID of the volume with the blkid command, and use that UUID in the <code>/etc/fstab</code> file.
You need to format a volume to the XFS filesystem type.	Format the target volume with the command mkfs.xfs .
You need to format a volume to the ext2, ext3, or ext4 filesystem type.	Format the target volume with a command such as 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 want to expand the space available to an LV formatted with the XFS filesystem.	Use the lvextend command to increase the space available to an LV, and then use the xfs_growfs command to expand the formatted filesystem accordingly.
You need to configure automated mounts to a shared network filesystem.	Configure the filesystem either in <code>/etc/fstab</code> 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.

RHEL 7 also supports the configuration of logical volumes. The process is a bit intricate, as it requires the configuration of a partition as a PV. One or more 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.

Linux supports the format of partitions, RAID arrays, and logical volumes to a wide variety of filesystems. Although the default is XFS, 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 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**, **gdisk** and **parted** utilities can help you create and delete partitions.
- ☐ **fdisk**, **gdisk**, and **parted** can be used to configure partitions for logical volumes and RAID arrays.
- ☐ Disks can use the traditional MBR-style partitioning scheme, which supports primary, extended, and logical partitions, or the GPT scheme, which supports up to 128 partitions.

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 7 filesystem is XFS.
- ❑ RHEL 7 supports a variety of **mkfs.*** filesystem format-check 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 **lvcreate** and **lvextend** commands.
- ❑ The extra space can be used to extend an existing XFS filesystem with the **xfs_growfs** command.

Filesystem Management

- ❑ Standard filesystems are mounted as defined in `/etc/fstab`.
- ❑ Filesystem volumes are usually 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`.

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.



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 correct answer to many of these questions.

Storage Management and Partitions

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

2. After a swap partition has been created, what command activates it?

Filesystem Formats

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

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

Basic Linux Filesystems and Directories

5. What filesystem is mounted on a directory separate from the top-level root directory in the default RHEL 7 installation?

6. Name three directories just below / that are 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 Logical Volume Management type, what command adds it as a PV?

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

Filesystem Management

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

10. What would you add to the `/etc/fstab` file to set up access to the partition `/dev/vda6`, mounted on the `/usr` directory as read-only with other default options? Assume you can't find the UUID of `/dev/vda6`. Also assume a dump value of 1 and a filesystem check order of 2.

The Automounter

11. 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?

12. Name three configuration files associated with the default installation of the automounter on RHEL 7.

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.

Red Hat presents its exams electronically. For that reason, the labs in this and future chapters are available from the DVD that accompanies the book; look for this chapter's labs in the `Chapter6/` subdirectory. The labs are available in `.doc`, `.html`, and `.txt` formats. In case you haven't yet set up RHEL 7 on a system, refer to the first lab of Chapter 2 for installation instructions. However, the answers for each lab follow the Self Test answers for the fill-in-the-blank questions.



SELF TEST ANSWERS

Storage Management and Partitions

1. The **fdisk** command option that lists configured partitions from all attached hard drives is **fdisk -l**.
2. After creating a swap partition, you can use the **mkswap *devicename*** and **swapon *devicename*** commands to initialize and activate the volume; 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 XFS is faster data recovery.
4. The command that formats `/dev/sdb3` to the default Red Hat filesystem format is **mkfs.xfs /dev/sdb3**. The **mkfs -t xfs /dev/sdb3** command is also an acceptable answer.

Basic Linux Filesystems and Directories

5. The `/boot` filesystem is mounted separately from `/`.
6. There are many correct answers to this question; some of the directories not suitable for mounting separately from `/` include `/dev`, `/etc`, and `/root`. (In contrast, several directories are essentially shown as placeholders for mounting, including `/media` and `/mnt`.)

Logical Volume Management (LVM)

7. Once you've created a new partition and set it to the Logical Volume Management file type, 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 underlining XFS filesystem to fill the new space is **xfs_growfs**.

Filesystem Management

9. To change the mount options for a local filesystem, edit `/etc/fstab`.
10. 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 /usr xfs defaults,ro 1 2
```

The Automounter

11. 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**.
12. 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`.

LAB ANSWERS

One of the assumptions with these labs is that where a directory such as `/test1` is specified, 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 possibly encounter unexpected errors.

Lab 1

1. It shouldn't matter whether partitions are created in the **fdisk** or **parted** utility. If 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.xfs** and **mkswap** commands), you'll see the UUID for the new partitions in the output to **blkid**. 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`.
3. You should be able to confirm the configuration of a new swap partition in the output to the **cat /proc/swaps** command. You should also be able to verify the result in the Swap line associated with the **top** command, as well as in the output of the **free -h** command.
4. 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 900MB. Some of that variance comes from the differences between units of measure that rely on base 2 or base 10 numbers. Don't panic; that variance is normal. 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 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 the partitions prepared for logical volumes, run the **fdisk -l** command. Appropriate partitions should appear with the “Linux LVM” label. To verify the configuration of PVs, run the **pvs** command. The output should report the devices and space allocated to PVs.
2. To verify the configuration of a VG, run the **vgs** command. The output should list the VG created during the lab from the PVs, including the space available.
3. To verify the configuration of an 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.
4. To verify the UUID of the newly formatted volume, run the following command:

```
# blkid <device_path>
```

5. If the /etc/fstab file is properly configured, you should be able to run the **mount -a** command. Then you should see the logical volume mounted on the /test2 directory.
6. 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 commands in this lab are **lvextend** and **xfs_growfs**. While there are a number of excellent command options available, all you really need with that command is the device file for the LV. As with Lab 2, the result can be confirmed after an appropriate **mount** and **df** command. However, to ensure that no data have been lost during the process, you could create some test files before resizing the LV and the filesystem.

Lab 4

There are several steps associated with this lab:

1. Ensure that all the partitions and volumes created in the previous labs have been unmounted, eliminated from /etc/fstab, and removed (with **{lv,vg,pv}remove**).
2. You don’t need to partition the /dev/vdb and /dev/vdc drives. It is sufficient to initialize the entire drives as PVs, using the **pvccreate /dev/vdb** and **pvccreate /dev/vdc** commands.
3. Run the **vgcreate -s 2M vg01 /dev/vdb /dev/vdc** command to create the VG.
4. Create the LV with the **lvcreate -l 800 -n lv01 vg01** command.

5. Format the filesystem with the **mkfs.ext4 /dev/vg01/lv01** command.
6. Add the correct entry to `/etc/fstab`.
7. Create the mount point (**mkdir /test4**).

If the `/etc/fstab` file is properly configured, you should be able to run the **mount -a** command. Then you should see `/dev/mapper/vg01-lv01` mounted on the `/test4` directory.

Lab 5

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 remote_ipaddr** command, where *remote_ipaddr* 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
```

Alternatively, 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 **systemctl reload autofs** 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.