```
# group: acctg
user::rw-
user:bob:r--
group::rw-
mask::rw-
other::---
[root@server1 ~]#_
```

After running the `getfacl` command, you will notice an extra node in the output: the mask. The mask is compared to all additional user and group permissions in the ACL. If the mask is more restrictive, it takes precedence when it comes to permissions. For example, if the mask is set to `r--` and the user bob has `rw-`, then the user bob actually gets `r--` to the file. When you run the `setfacl` command, the mask is always made equal to the least restrictive permission assigned so that it does not affect additional ACL entries. The mask was created as a mechanism that could easily revoke permissions on a file that had several additional users and groups added to the ACL.

To remove all extra ACL assignments on the doc1 file, use the `–b` option to the `setfacl` command:

```
[root@server1 ~]# setfacl -b doc1
[root@server1 ~]# ls -l doc1
-rw-rw----   1 user1     acctg            0 May  2 22:01 doc1
[root@server1 ~]#_
```

Managing Filesystem Attributes

As with the Windows operating system, Linux has file attributes that can be set, if necessary. These attributes work outside Linux permissions and are filesystem-specific. This section examines attributes for the ext4 filesystem that you configured for your Fedora Linux system during Hands-On Project 2-1. Filesystem types will be discussed in more depth in Chapter 5.

To see the filesystem attributes that are currently assigned to a file, you can use the **`lsattr` (list attributes) command**, as shown here for the doc1 file:

```
[root@server1 ~]# lsattr doc1
--------------e---- doc1
[root@server1 ~]#_
```

By default, all files have the `e` attribute, which writes to the file in "extent" blocks (rather than immediately in a byte-by-byte fashion). If you would like to add or remove attributes, you can use the **`chattr` (change attributes) command**. The following example assigns the immutable attribute (`i`) to the doc1 file and displays the results:

```
[root@server1 ~]# chattr +i doc1
[root@server1 ~]# lsattr doc1
----i---------e---- doc1
[root@server1 ~]#_
```

The immutable attribute is the most commonly used filesystem attribute and prevents the file from being modified in any way. Because attributes are applied at a filesystem level, not even the root user can modify a file that has the immutable attribute set.

Note 21

Most filesystem attributes are rarely set, as they provide for low-level filesystem functionality. To view a full listing of filesystem attributes, visit the manual page for the `chattr` command.

Similarly, to remove an attribute, use the chattr command with the – option, as shown here with the doc1 file:

```
[root@server1 ~]# chattr -i doc1
[root@server1 ~]# lsattr doc1
--------------e---- doc1
[root@server1 ~]#_
```

Summary

- The Linux directory tree obeys the Filesystem Hierarchy Standard, which allows Linux users and developers to locate system files in standard directories.
- Many file management commands are designed to create, change the location of, or remove files and directories. The most common of these include `cp`, `mv`, `rm`, `rmdir`, and `mkdir`.
- You can find files on the filesystem using an indexed database (the `locate` command) or by searching the directories listed in the PATH variable (the `which` command). However, the most versatile command used to find files is the `find` command, which searches for files based on a wide range of criteria.
- Files can be linked two ways. In a symbolic link, one file serves as a pointer to another file. In a hard link, one file is a linked duplicate of another file.
- Each file and directory has an owner and a group owner. In the absence of system restrictions, the owner of the file or directory can change permissions and give ownership to others.
- File and directory permissions can be set for the owner (user), group owner members (group), as well as everyone else on the system (other).
- There are three regular file and directory permissions (read, write, execute) and three special file and directory permissions (SUID, SGID, sticky bit). The definitions of these permissions are different for files and directories.
- Permissions can be changed using the `chmod` command by specifying symbols or numbers.
- To ensure security, new files and directories receive default permissions from the system, less the value of the umask variable.
- The root user has all permissions to all files and directories on the Linux filesystem. Similarly, the root user can change the ownership of any file or directory on the Linux filesystem.
- The default ACL (user, group, other) on a file or directory can be modified to include additional users or groups.
- Filesystem attributes can be set on Linux files to provide low-level functionality such as immutability.

Key Terms

access control list (ACL)
chattr (change attributes) command
chgrp (change group) command
chmod (change mode) command
chown (change owner) command
cp (copy) command
data blocks
Filesystem Hierarchy Standard (FHS)
find command
getfacl (get file ACL) command
group
hard link
inode
inode table
interactive mode
ln (link) command
locate command
lsattr (list attributes) command
mkdir (make directory) command
mode
mv (move) command
other
owner
passwd command
PATH variable
permission
primary group
pruning
recursive
rm (remove) command

`rmdir` (remove directory) command
`setfacl` (set file ACL) command
source file/directory
superblock
symbolic link
target file/directory
`touch` command
`type` command
umask
`umask` command
`unlink` command
user
`whereis` command
`which` command

Review Questions

1. A symbolic link is depicted by an @ symbol appearing at the beginning of the filename when viewed using the `ls -l` command.
   a. True
   b. False
2. What was created to define a standard directory structure and common file location for UNIX and Linux systems?
   a. POSIX
   b. X.500
   c. FHS
   d. OOBLA
3. There is no real difference between the “S” and “s” special permissions when displayed using the `ls -l` command. One just means it is on a file, and the other means that it is on a directory.
   a. True
   b. False
4. The default permissions given by the system prior to analyzing the umask are ________________ for newly created directories and ________________ for newly created files.
   a. rw-rw-rw- and rw-rw-rw-
   b. rw-rw-rw- and r--r--r--
   c. rw-rw-rw- and rwxrwxrwx
   d. rwxrwxrwx and rw-rw-rw-
5. What must a Fedora Linux user do to run `cp` or `mv` interactively and be asked whether to overwrite an existing file?
   a. Just run `cp` or `mv` because they run in interactive mode by default.
   b. Run `interactive cp` or `interactive mv`.
   c. Run `cp -i` or `mv -i`.
   d. Run `cp -interactive` or `mv -interactive`.
6. The root user utilizes the `chown` command to give ownership of a file to another user. What must the root user do to regain ownership of the file?
   a. Have the new owner run `chgrp` and list the root user as the new owner.
   b. Run `chgrp` again listing the root user as the new owner.
   c. Nothing, because this is a one-way, one-time action.
   d. Run `chown` and list the root user as the new owner.
7. After typing the `ls -F` command, you see the following line in the output:
   `-rw-r-xr-- 1 user1 root 0 Apr 29 15:40 file1`
   How do you interpret the mode of file1?
   a. User1 has read and write, members of the root group have read and execute, and all others have read permissions to the file.
   b. Members of the root group have read and write, user1 has read and execute, and all others have read permissions to the file.
   c. All users have read and write, members of the root group have read and execute, and user1 has read permissions to the file.
   d. User1 has read and write, all others have read and execute, and members of the root group have read permissions to the file.
8. After typing the command `umask 731`, the permissions on all subsequently created files and directories will be affected. In this case, what will be the permissions on all new files?
   a. rw-rw-rw-
   b. rwxrw-r--
   c. ---r--rw-
   d. ----wx--x
9. You noticed a file in your home directory that has a + symbol appended to the mode. What does this indicate?
   a. Special permissions have been set on the file.
   b. The file has one or more files on the filesystem that are hard linked to it.
   c. The sticky bit directory permission has been set on the file and will remain inactive as a result.
   d. Additional entries exist within the ACL of the file that can be viewed using the `getfacl` command.
10. When you change the data in a file that is hard-linked to three others, ________________.
    a. the data in the file you modified and the data in all hard-linked files are modified because they have different inodes

b. the data in the file you modified as well as the data in all hard-linked files are modified because they share the same data and inode
c. only the data in the file you modified is affected
d. only the data in the file you modified and any hard-linked files in the same directory are affected

11. The command `chmod 317 file1` would produce which of the following lines in the `ls` command?

a. -w-r--rwx 1 user1 root 0 Apr 29 15:40 file1
b. -wx--xrwx 1 user1 root 0 Apr 29 15:40 file1
c. -rwxrw-r-x 1 user1 root 0 Apr 29 15:40 file1
d. --w-rw-r-e 1 user1 root 0 Apr 29 15:40 file1

12. Which of the following commands will change the user ownership and group ownership of *file1* to user1 and root, respectively?

a. `chown user1:root file1`
b. `chown user1 : root file1`
c. `chown root:user1 file1`
d. `chown root : user1 file1`

13. What does the /var directory contain?

a. various additional programs
b. log files and spool directories
c. temporary files
d. files that are architecture-independent

14. What does the `mv` command do? (Choose all that apply.)

a. It renames a file.
b. It renames a directory.
c. It moves a directory to another location on the filesystem.
d. It moves a file to another location on the filesystem.

15. A file has the following permissions: *r----x-w-*. The command `chmod 143 file1` would have the same effect as the command ________________. (Choose all that apply.)

a. `chmod u+x-r,g+r-x,o+x file1`
b. `chmod u=w,g=rw,o=rx file1`
c. `chmod u-r-w,g+r-w,o+r-x file1`
d. `chmod u=x,g=r,o=wx file1`
e. `chmod u+w,g+r-w,o+r-x file1`

16. The `which` command ________________.

a. can only be used to search for aliases
b. searches for a file in all directories, starting from the root
c. is not a valid Linux command
d. searches for a file only in directories that are in the PATH variable

17. Hard links need to reside on the same filesystem, whereas symbolic links need not be on the same filesystem as their target.

a. True
b. False

18. When applied to a directory, the SGID special permission ________________.

a. allows users the ability to use more than two groups for files that they create within the directory
b. causes all new files created in the directory to have the same group membership as the directory, and not the entity that created them
c. causes users to have their permissions checked before they are allowed to access files in the directory
d. cannot be used because it is applied only to files

19. Given the following output from the `ls` command, how many other files are hard linked with file3?

```
drwxr-xr-x  3  root   root  4096 Apr 8   07:12  Desktop
-rw-r--r--  3  root   root  282  Apr 29  22:06  file1
-rw-r--r--  1  root   root  282  Apr 29  22:06  file2
-rw-r--r--  4  root   root  282  Apr 29  22:06  file3
-rw-r--r--  2  root   root  282  Apr 29  22:06  file4
-rw-r--r--  1  root   root  282  Apr 29  22:06  file5
-rw-r--r--  1  user1  sys   282  Apr 29  22:06  file6
```

a. one
b. two
c. three
d. four

20. Only the root user can modify a file that has the immutable attribute set.

a. True
b. False

Hands-On Projects

These projects should be completed in the order given. The hands-on projects presented in this chapter should take a total of three hours to complete. The requirements for this lab include:

- A computer with Fedora Linux installed according to Hands-On Project 2-1
- Completion of all hands-on projects in Chapter 3

Project 4-1

Estimated Time: 10 minutes
Objective: Create directories.
Description: In this hands-on project, you log in to the computer and create new directories.

1. Boot your Fedora Linux virtual machine. After your Linux system has loaded, switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5**. Log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
2. At the command prompt, type **`ls -F`** and press **Enter**. Note the contents of your home folder.
3. At the command prompt, type **`mkdir mysamples`** and press **Enter**. Next type **`ls -F`** at the command prompt, and press **Enter** to verify the creation of the subdirectory.
4. At the command prompt, type **`cd mysamples`** and press **Enter**. Next, type **`ls -F`** at the command prompt and press **Enter**. What are the contents of the subdirectory mysamples?
5. At the command prompt, type **`mkdir undermysamples`** and press **Enter**. Next, type **`ls -F`** at the command prompt and press **Enter**. What are the contents of the subdirectory mysamples?
6. At the command prompt, type **`mkdir todelete`** and press **Enter**. Next, type **`ls -F`** at the command prompt and press **Enter**. Does the subdirectory todelete you just created appear listed in the display?
7. At the command prompt, type **`cd ..`** and press **Enter**. Next, type **`ls -R`** and press **Enter**. Notice that the subdirectory mysamples and its subdirectory undermysamples are both displayed.
8. At the command prompt, type **`cd ..`** and press **Enter**. At the command prompt, type **`pwd`** and press **Enter**. What is your current directory?
9. At the command prompt, type **`mkdir foruser1`** and press **Enter**. At the command prompt, type **`ls -F`** and press **Enter**. Does the subdirectory you just created appear listed in the display?
10. Type **`exit`** and press **Enter** to log out of your shell.

Project 4-2

Estimated Time: 15 minutes
Objective: Copy files and directories.
Description: In this hands-on project, you copy files and directories using the `cp` command.

1. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
2. Next, type **`ls -F`** at the command prompt and press **Enter**. Note the contents of your home folder.
3. At the command prompt, type **`cp sample1`** and press **Enter**. What error message was displayed and why?
4. At the command prompt, type **`cp sample1 sample1A`** and press **Enter**. Next, type **`ls -F sample*`** at the command prompt and press **Enter**. How many files are there and what are their names?

5. At the command prompt, type **`cp sample1 mysamples/sample1B`** and press **Enter**. Next, type **`ls -F sample*`** at the command prompt and press **Enter**. How many files are there and what are their names?
6. At the command prompt, type **`cd mysamples`** and press **Enter**. Next, type **`ls -F`** at the command prompt and press **Enter**. Was sample1B copied successfully?
7. At the command prompt, type **`cp /root/sample2 .`** and press **Enter**. Next, type **`ls -F`** at the command prompt and press **Enter**. How many files are there and what are their names?
8. At the command prompt, type **`cp sample1B ..`** and press **Enter**. Next, type **`cd ..`** at the command prompt and press **Enter**. At the command prompt, type **`ls -F sample*`** and press **Enter**. Was the sample1B file copied successfully?
9. At the command prompt, type **`cp sample1 sample2 sample3 mysamples`** and press **Enter**. What message do you get and why? Choose **y** and press **Enter**. Next, type **`cd mysamples`** at the command prompt and press **Enter**. At the command prompt, type **`ls -F`** and press **Enter**. How many files are there and what are their names?
10. At the command prompt, type **`cd ..`** and press **Enter**. Next, type **`cp mysamples mysamples2`** at the command prompt and press **Enter**. What error message did you receive? Why?
11. At the command prompt, type **`cp -R mysamples mysamples2`** and press **Enter**. Next, type **`ls -F`** at the command prompt, and press **Enter**. Was the directory copied successfully? Type **`ls -F mysamples2`** at the command prompt and press **Enter**. Were the contents of mysamples successfully copied to mysamples2?
12. Type **`exit`** and press **Enter** to log out of your shell.

Project 4-3

Estimated Time: 20 minutes
Objective: Organize files and directories.
Description: In this hands-on project, you use the `mv` command to rename and move files and directories.

1. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
2. Next, type **`ls -F`** at the command prompt and press **Enter**. Note the contents of your home folder.
3. At the command prompt, type **`mv sample1`** and press **Enter**. What error message was displayed and why?
4. At the command prompt, type **`mv sample1 sample4`** and press **Enter**. Next, type **`ls -F sample*`** at the command prompt and press **Enter**. How many files are listed and what are their names? What happened to sample1?
5. At the command prompt, type **`mv sample4 mysamples`** and press **Enter**. Next, type **`ls -F sample*`** at the command prompt and press **Enter**. How many files are there and what are their names? Where did sample4 go?
6. At the command prompt, type **`cd mysamples`** and press **Enter**. Next, type **`ls -F sample*`** at the command prompt and press **Enter**. Notice that the sample4 file you moved in Step 5 was moved here.
7. At the command prompt, type **`mv sample4 ..`** and press **Enter**. Next, type **`ls -F sample*`** at the command prompt and press **Enter**. How many files are there and what are their names? Where did the sample4 file go?
8. At the command prompt, type **`cd ..`** and press **Enter**. Next, type **`ls -F`** at the command prompt and press **Enter** to view the new location of sample4.
9. At the command prompt, type **`mv sample4 mysamples/sample2`** and press **Enter**. What message appeared on the screen and why?

10. Type `y` and press **Enter** to confirm you want to overwrite the file in the destination folder.
11. At the command prompt, type `mv sample? mysamples` and press **Enter**. Type `y` and press **Enter** to confirm you want to overwrite the file sample3 in the destination folder.
12. At the command prompt, type `ls -F sample*` and press **Enter**. How many files are there and why?
13. At the command prompt, type `mv sample1* mysamples` and press **Enter**. Type `y` and press **Enter** to confirm you want to overwrite the file sample1B in the destination directory.
14. At the command prompt, type `ls -F sample*` and press **Enter**. Notice that there are no sample files in the /root directory.
15. At the command prompt, type `cd mysamples` and press **Enter**. Next, type `ls -F` at the command prompt and press **Enter**. Notice that all files originally in /root have been moved to this directory.
16. At the command prompt, type `cd ..` and press **Enter**. Next, type `ls -F` at the command prompt and press **Enter**. Type `mv mysamples samples` and press **Enter**. Next, type `ls -F` at the command prompt and press **Enter**. Why did you not need to specify the recursive option to the `mv` command to rename the mysamples directory to samples?
17. Type `exit` and press **Enter** to log out of your shell.

Project 4-4

Estimated Time: 20 minutes
Objective: Create hard and symbolic links.
Description: In this hands-on project, you make and view links to files and directories.

1. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
2. At the command prompt, type `cd samples` and press **Enter**. Next, type `ls -F` at the command prompt and press **Enter**. What files do you see? Next, type `ls -l` at the command prompt and press **Enter**. What is the link count for the sample1 file?
3. At the command prompt, type `ln sample1 hardlinksample` and press **Enter**. Next, type `ls -F` at the command prompt and press **Enter**. Does anything in the terminal output indicate that sample1 and hardlinksample are hard-linked? Next, type `ls -l` at the command prompt and press **Enter**. Does anything in the terminal output indicate that sample1 and hardlinksample are hard-linked? What is the link count for sample1 and hardlinksample? Next, type `ls -li` at the command prompt and press **Enter** to view the inode numbers of each file. Do the two hard-linked files have the same inode number?
4. At the command prompt, type `ln sample1 hardlinksample2` and press **Enter**. Next, type `ls -l` at the command prompt and press **Enter**. What is the link count for the files sample1, hardlinksample, and hardlinksample2? Why?
5. At the command prompt, type `vi sample1` and press **Enter**. Enter a sentence of your choice into the vi editor, and then save your document and quit the vi editor.
6. At the command prompt, type `cat sample1` and press **Enter**. Next, type `cat hardlinksample` at the command prompt and press **Enter**. Next, type `cat hardlinksample2` at the command prompt and press **Enter**. Are the contents of each file the same? Why?
7. At the command prompt, type `ln –s sample2 symlinksample` and press **Enter**. Next, type `ls -F` at the command prompt and press **Enter**. Does anything in the terminal output indicate that sample2 and symlinksample are symbolically linked? Next, type `ls -l` at the command prompt and press **Enter**. Does anything in the terminal output indicate that sample2 and symlinksample are symbolically linked? Next, type `ls -li` at the command prompt and press **Enter** to view the inode numbers of each file. Do the two symbolically linked files have the same inode number?

8. At the command prompt, type **vi symlinksample** and press **Enter**. Enter a sentence of your choice into the vi editor, and then save your document and quit the vi editor.
9. At the command prompt, type **ls -l** and press **Enter**. What is the size of the symlinksample file compared to sample2? Why? Next, type **cat sample2** at the command prompt and press **Enter**. What are the contents and why?
10. At the command prompt, type **ln -s /etc/samba samba** and press **Enter**. Next, type **ls -F** at the command prompt and press **Enter**. What file type is indicated for samba? Next, type **cd samba** at the command prompt and press **Enter**. Type **pwd** at the command prompt and press **Enter** to view your current directory. What is your current directory? Next, type **ls -F** at the command prompt and press **Enter**. What files are listed? Next, type **ls -F /etc/samba** at the command prompt and press **Enter**.

    Note that your samba directory is merely a pointer to the /etc/samba directory. How can this type of linking be useful?
11. Type **exit** and press **Enter** to log out of your shell.

Project 4-5

Estimated Time: 15 minutes
Objective: Find system files.
Description: In this hands-on project, you find files on the filesystem using the find, locate, which, type, and whereis commands.

1. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
2. At the command prompt, type **touch newfile** and press **Enter**. Next, type **locate newfile** at the command prompt and press **Enter**. Did the locate command find the file you just created? Why?
3. At the command prompt, type **updatedb** and press **Enter**. When the command is finished, type **locate newfile** at the command prompt and press **Enter**. Did the locate command find the file? If so, how quickly did it find it? Why?
4. At the command prompt, type **find / -name "newfile"** and press **Enter**. Did the find command find the file? If so, how quickly did it find it? Why?
5. At the command prompt, type **find /root -name "newfile"** and press **Enter**. Did the find command find the file? How quickly did it find it? Why?
6. At the command prompt, type **which newfile** and press **Enter**. Did the which command find the file? Type **echo $PATH** at the command prompt and press **Enter**. Is the /root directory listed in the PATH variable? Is the /usr/bin directory listed in the PATH variable?
7. At the command prompt, type **which grep** and press **Enter**. Did the which command find the file? Why?
8. At the command prompt, type **type grep** and press **Enter**. Next, type **whereis grep** and press **Enter**. Do these commands return less or more than the which command did in the previous step? Why?
9. At the command prompt, type **find /root -name "sample*"** and press **Enter**. What files are listed?
10. At the command prompt, type **find /root -type l** and press **Enter**. What files are listed?
11. At the command prompt, type **find /root -size 0** and press **Enter**. What types of files are listed? Type **find / -size 0 | less** to see all of the files of this type on the system.
12. Type **exit** and press **Enter** to log out of your shell.

Project 4-6

Estimated Time: 10 minutes
Objective: Remove files and directories.
Description: In this hands-on project, you delete files and directories using the `rmdir` and `rm` commands.

1. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
2. At the command prompt, type **`cd samples`** and press **Enter**. At the command prompt, type **`ls -R`** and press **Enter**. Note the two empty directories todelete and undermysamples.
3. At the command prompt, type **`rmdir undermysamples todelete`** and press **Enter**. Next, type **`ls -F`** at the command prompt and press **Enter**. Were both directories deleted successfully? Why?
4. At the command prompt, type **`rm sample1*`** and press **Enter**. What message is displayed? Answer **n** to all three questions.
5. At the command prompt, type **`rm -f sample1*`** and press **Enter**. Why were you not prompted to continue? Next, type **`ls -F`** at the command prompt and press **Enter**. Were all three files deleted successfully? What other command may be used to delete a file within Linux?
6. At the command prompt, type **`cd ..`** and press **Enter**. Next, type **`rmdir samples`** at the command prompt and press **Enter**. What error message do you receive and why?
7. At the command prompt, type **`rm -rf samples`** and press **Enter**. Next, type **`ls -F`** at the command prompt and press **Enter**. Was the samples directory and all files within it deleted successfully?
8. Type **`exit`** and press **Enter** to log out of your shell.

Project 4-7

Estimated Time: 30 minutes
Objective: Set access permissions.
Description: In this hands-on project, you apply and modify access permissions on files and directories and test their effects.

1. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
2. At the command prompt, type **`touch permsample`** and press **Enter**. Next, type **`chmod 777 permsample`** at the command prompt and press **Enter**.
3. At the command prompt, type **`ls -l`** and press **Enter**. Who has read, write, and execute permissions to this file?
4. At the command prompt, type **`chmod 000 permsample`** and press **Enter**. Next, type **`ls -l`** at the command prompt and press **Enter**. Who has read, write, and execute permissions to this file?
5. At the command prompt, type **`rm -f permsample`** and press **Enter**. Were you able to delete this file? Why?
6. At the command prompt, type **`cd /`** and press **Enter**. Next, type **`pwd`** at the command prompt and press **Enter**. Type **`ls -F`** at the command prompt and press **Enter**. Note the directories you see under the /.
7. At the command prompt, type **`ls -l`** and press **Enter** to view the owner, group owner, and permissions on the foruser1 directory created in Hands-On Project 4-1. Who is the owner and group owner? If you were logged in as the user user1, in which category would you be placed (user, group, other)? What permissions do you have as this category (read, write, execute)?
8. At the command prompt, type **`cd /foruser1`** and press **Enter** to enter the foruser1 directory. Next, type **`ls -F`** at the command prompt and press **Enter**. Are there any files in this directory? Type **`cp /etc/hosts .`**

at the command prompt and press **Enter**. Next, type `ls -F` at the command prompt and press **Enter** to ensure that a copy of the hosts file was made in your current directory.

9. Switch to a different command-line terminal (tty3) by pressing **Ctrl+Alt+F3** and log in to the terminal using the user name of **user1** and the password of **LINUXrocks!**.
10. At the command prompt, type `cd /foruser1` and press **Enter**. Were you successful? Why? Next, type `ls -F` at the command prompt and press **Enter**. Were you able to see the contents of the directory? Why? Next, type `rm -f hosts` at the command prompt and press **Enter**. Why did you receive an error message?
11. Switch back to your previous command-line terminal (tty5) by pressing **Ctrl+Alt+F5**. Note that you are logged in as the root user on this terminal and within the /foruser1 directory.
12. At the command prompt, type `chmod o+w /foruser1` and press **Enter**. Were you able to change the permissions on the /foruser1 directory successfully? Why?
13. Switch back to your previous command-line terminal (tty3) by pressing **Ctrl+Alt+F3**. Note that you are logged in as the user1 user on this terminal and within the /foruser1 directory.
14. At the command prompt, type `rm -f hosts` at the command prompt and press **Enter**. Were you successful now? Why?
15. Switch back to your previous command-line terminal (tty5) by pressing **Ctrl+Alt+F5**. Note that you are logged in as the root user on this terminal.
16. At the command prompt, type `cp /etc/hosts .` at the command prompt and press **Enter** to place another copy of the hosts file in your current directory (/foruser1).
17. At the command prompt, type `ls -l` and press **Enter**. Who is the owner and group owner of this file? If you were logged in as the user user1, in which category would you be placed (user, group, other)? What permissions does user1 have as this category (read, write, execute)?
18. Switch back to your previous command-line terminal (tty3) by pressing **Ctrl+Alt+F3**. Note that you are logged in as the user1 user on this terminal and in the /foruser1 directory.
19. At the command prompt, type `cat hosts` at the command prompt and press **Enter**. Were you successful? Why? Next, type `vi hosts` at the command prompt to open the hosts file in the vi editor. Delete the first line of this file and save your changes. Were you successful? Why? Exit the vi editor and discard your changes.
20. Switch back to your previous command-line terminal (tty5) by pressing **Ctrl+Alt+F5**. Note that you are logged in as the root user on this terminal and in the /foruser1 directory.
21. At the command prompt, type `chmod o+w hosts` and press **Enter**.
22. Switch back to your previous command-line terminal (tty3) by pressing **Ctrl+Alt+F3**. Note that you are logged in as the user1 user on this terminal and in the /foruser1 directory.
23. At the command prompt, type `vi hosts` at the command prompt to open the hosts file in the vi editor. Delete the first line of this file and save your changes. Why were you successful this time? Exit the vi editor.
24. At the command prompt, type `ls -l` and press **Enter**. Do you have permission to execute the hosts file? Should you make this file executable? Why? Next, type `ls -l /usr/bin` at the command prompt and press **Enter**. Note how many of these files to which you have execute permission. Type `file /usr/bin/* | more` at the command prompt and press **Enter** to view the file types of the files in the /bin directory. Should these files have the execute permission?
25. Type `exit` and press **Enter** to log out of your shell.
26. Switch back to your previous command-line terminal (tty5) by pressing **Ctrl+Alt+F5**. Note that you are logged in as the root user on this terminal.
27. Type `exit` and press **Enter** to log out of your shell.

Project 4-8

Estimated Time: 15 minutes
Objective: Set default permissions.
Description: In this hands-on project, you view and manipulate the default file and directory permissions using the umask variable.

1. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **user1** and the password of **LINUXrocks!**.
2. At the command prompt, type **`umask`** and press **Enter**. What is the default umask variable?
3. At the command prompt, type **`touch utest1`** and press **Enter**. Next, type **`ls -l`** at the command prompt and press **Enter**. What are the permissions on the utest1 file? Do these agree with the calculation in Figure 4-4? Create a new directory by typing the command **`mkdir udir1`** at the command prompt and pressing **Enter**. Next, type **`ls -l`** at the command prompt and press **Enter**. What are the permissions on the udir1 directory? Do these agree with the calculation in Figure 4-4?
4. At the command prompt, type **`umask 007`** and press **Enter**. Next, type **`umask`** at the command prompt and press **Enter** to verify that your umask variable has been changed to 007.
5. At the command prompt, type **`touch utest2`** and press **Enter**. Next, type **`ls -l`** at the command prompt and press **Enter**. What are the permissions on the utest2 file? Do these agree with the calculation in Figure 4-5? Create a new directory by typing the command **`mkdir udir2`** at the command prompt and pressing **Enter**. Next, type **`ls -l`** at the command prompt and press **Enter**. What are the permissions on the udir2 directory? Do these agree with the calculation in Figure 4-5?
6. Type **`exit`** and press **Enter** to log out of your shell.

Project 4-9

Estimated Time: 15 minutes
Objective: Set file and directory ownership.
Description: In this hands-on project, you view and change file and directory ownership using the `chown` and `chgrp` commands.

1. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
2. At the command prompt, type **`touch ownersample`** and press **Enter**. Next, type **`mkdir ownerdir`** at the command prompt and press **Enter**. Next, type **`ls -l`** at the command prompt and press **Enter** to verify that the file ownersample and directory ownerdir were created and that root is the owner and group owner of each.
3. At the command prompt, type **`chgrp sys owner*`** and press **Enter** to change the group ownership to the sys group for both ownersample and ownerdir. Why were you successful?
4. At the command prompt, type **`chown user1 owner*`** and press **Enter** to change the ownership to the user1 user for both ownersample and ownerdir. Why were you successful?
5. At the command prompt, type **`chown root.root owner*`** and press **Enter** to change the ownership and group ownership back to the root user for both ownersample and ownerdir. Although you are not the current owner of these files, why did you not receive an error message?
6. At the command prompt, type **`mv ownersample ownerdir`** and press **Enter**. Next, type **`ls -lR`** at the command prompt and press **Enter** to note that the ownersample file now exists within the ownerdir directory and that both are owned by root.
7. At the command prompt, type **`chown -R user1 ownerdir`** and press **Enter**. Next, type **`ls -lR`** at the command prompt and press **Enter**. Who owns the ownerdir directory and ownersample file?
8. At the command prompt, type **`rm -Rf ownerdir`** and press **Enter**. Why were you able to delete this directory without being the owner of it?
9. Type **`exit`** and press **Enter** to log out of your shell.

Project 4-10

Estimated Time: 15 minutes
Objective: Set special permissions and ACL entries.
Description: In this hands-on project, you view and set special permissions on files and directories as well as modify the default ACL on a file.

1. Switch to a command-line terminal (tty3) by pressing **Ctrl+Alt+F3** and log in to the terminal using the user name of **user1** and the password of **LINUXrocks!**.
2. At the command prompt, type `touch specialfile` and press **Enter**. Next, type `ls –l` at the command prompt and press **Enter** to verify that specialfile was created successfully. Who is the owner and group owner of specialfile?
3. At the command prompt, type `chmod 4777 specialfile` and press **Enter**. Next, type `ls -l` at the command prompt and press **Enter**. Which special permission is set on this file? If this file were executed by another user, who would that user be during execution?
4. At the command prompt, type `chmod 6777 specialfile` and press **Enter**. Next, type `ls -l` at the command prompt and press **Enter**. Which special permissions are set on this file? If this file were executed by another user, who would that user be during execution and which group would that user be a member of?
5. At the command prompt, type `chmod 6444 specialfile` and press **Enter**. Next, type `ls -l` at the command prompt and press **Enter**. Can you tell if execute is not given underneath the special permission listings? Would the special permissions retain their meaning in this case?
6. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
7. At the command prompt, type `mkdir /public` and press **Enter**. Next, type `chmod 1777 /public` at the command prompt and press **Enter**. Which special permission is set on this directory? Who can add or remove files to and from this directory?
8. At the command prompt, type `touch /public/rootfile` and press **Enter**.
9. Type `exit` and press **Enter** to log out of your shell.
10. Switch back to your previous command-line terminal (tty3) by pressing **Ctrl+Alt+F3**. Note that you are logged in as the user1 user on this terminal.
11. At the command prompt, type `touch /public/user1file` and press **Enter**. Next, type `ls -l /public` at the command prompt and press **Enter**. What files exist in this directory and who are the owners?
12. At the command prompt, type `rm /public/user1file` and press **Enter**. Were you prompted to confirm the deletion of the file?
13. At the command prompt, type `rm /public/rootfile` and press **Enter**. Note the error message that you receive because of the sticky bit.
14. Type `exit` and press **Enter** to log out of your shell.
15. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
16. At the command prompt, type `touch aclfile` and press **Enter**. Next, type `getfacl aclfile` at the command prompt and press **Enter**. Are there any additional entries beyond user, group, and other?
17. At the command prompt, type `setfacl –m u:user1:r-- aclfile` and press **Enter**. Next, type `ls -l aclfile` at the command prompt and press **Enter**. Is there a + symbol following the mode? Next, type `getfacl aclfile` at the command prompt and press **Enter**. Explain the permissions. When would this permission set be useful?
18. At the command prompt, type `setfacl –b aclfile` and press **Enter**. Next, type `ls -l aclfile` at the command prompt and press **Enter**. Is there a + symbol following the mode?
19. Type `exit` and press **Enter** to log out of your shell.

Project 4-11

Estimated Time: 15 minutes
Objective: Set filesystem attributes.
Description: In this hands-on project, you configure and research filesystem attributes.

1. Switch to a command-line terminal (tty5) by pressing **Ctrl+Alt+F5** and log in to the terminal using the user name of **root** and the password of **LINUXrocks!**.
2. At the command prompt, type `touch toughfile` and press **Enter**. Next, type `lsattr toughfile` at the command prompt and press **Enter**. What filesystem attributes are set on toughfile by default?
3. At the command prompt, type `chattr +i toughfile` and press **Enter**. Next, type `lsattr toughfile` at the command prompt and press **Enter**. Is the immutable attribute set on toughfile?
4. At the command prompt, type `vi toughfile` and press **Enter**. Does the vi editor warn you that the file is read-only? Add a line of text of your choice in the vi editor and attempt to save your changes using `w!` at the `:` prompt. Were you successful? Quit the vi editor using `q!` at the `:` prompt.
5. At the command prompt, type `rm -f toughfile` and press **Enter**. Were you successful? Why?
6. At the command prompt, type `chattr -i toughfile` and press **Enter**. Next, type `rm -f toughfile` and press **Enter**. Were you successful this time? Why?
7. At the command prompt, type `man chattr` and press **Enter**. Search the manual page for other filesystem attributes. Which attribute tells the Linux kernel to automatically compress/decompress the file as it is written and read from the filesystem? Which attribute causes the data blocks of a file to be immediately overwritten once the file has been deleted? Type `q` and press **Enter** to quit out of the manual page when finished.
8. Type `exit` and press **Enter** to log out of your shell.

Discovery Exercises

Discovery Exercise 4-1

Estimated Time: 30 minutes
Objective: Explore FHS directories.
Description: Use the `ls` command with the `-F` option to explore directories described in the Filesystem Hierarchy Standard starting with /bin. Do you recognize any of the commands in /bin? Explore several other FHS directories and note their contents. Refer to Table 4-1 for a list of directories to explore. Further, visit www.pathname.com/fhs and read about the Filesystem Hierarchy Standard. What benefits does it offer Linux?

Discovery Exercise 4-2

Estimated Time: 35 minutes
Objective: Compose file management commands.
Description: Write the commands required for the following tasks. Try out each command on your system to ensure that it is correct:

a. Make a hierarchical directory structure under /root that consists of one directory containing three subdirectories.
b. Copy two files into each of the subdirectories.
c. Create one more directory with three subdirectories beneath it and move files from the subdirectories containing them to the counterparts you just created.
d. Hard-link three of the files. Examine their inodes.
e. Symbolically link two of the files and examine their link count and inode information.
f. Make symbolic links from your home directory to two directories in this structure and examine the results.
g. Delete the symbolic links in your home directory and the directory structure you created under /root.

Discovery Exercise 4-3

Estimated Time: 35 minutes
Objective: Compose file search commands.
Description: Write the command that can be used to answer the following questions. (*Hint:* Try each on the system to check your results.)

a. Find all files on the system that have the word "test" as part of their filename.
b. Search the PATH variable for the pathname to the `awk` command.
c. Find all files in the /usr directory and subdirectories that are larger than 50 kilobytes in size.
d. Find all files in the /usr directory and subdirectories that are less than 70 kilobytes in size.
e. Find all files in the / directory and subdirectories that are symbolic links.
f. Find all files in the /var directory and subdirectories that were accessed less than 60 minutes ago.
g. Find all files in the /var directory and subdirectories that were accessed less than six days ago.
h. Find all files in the /home directory and subdirectories that are empty.
i. Find all files in the /etc directory and subdirectories that are owned by the group bin.

Discovery Exercise 4-4

Estimated Time: 10 minutes
Objective: Express permission sets numerically.
Description: For each of the following modes, write the numeric equivalent (e.g., 777):

a. rw-r--r--
b. r--r--r--
c. ---rwxrw-
d. -wxr-xrw-
e. rw-rw-rwx
f. -w-r-----

Discovery Exercise 4-5

Estimated Time: 15 minutes
Objective: Explain permissions.
Description: Fill in the permissions in Table 4-7 with checkmarks, assuming that all four files are in the directory /public, which has a mode of rwxr-xr-x.

Table 4-7 Permissions table for Discovery Exercise 4-5

Filename	Mode		Read	Edit	Execute	List	Delete
sample1	`rw-rw-rw-`	User Group Other					
sample2	`r--r-----`	User Group Other					
sample3	`rwxr-x---`	User Group Other					
sample4	`r-x------`	User Group Other					

Discovery Exercise 4-6

Estimated Time: 15 minutes
Objective: Explain permissions.
Description: Fill in the permissions in Table 4-8 with checkmarks, assuming that all four files are in the directory /public, which has a mode of rwx--x---.

Table 4-8 Permissions table for Discovery Exercise 4-6

Filename	Mode		Read	Edit	Execute	List	Delete
sample1	`rwxr--r--`	User					
		Group					
		Other					
sample2	`r-xr--rw-`	User					
		Group					
		Other					
sample3	`--xr-x---`	User					
		Group					
		Other					
sample4	`r-xr--r--`	User					
		Group					
		Other					

Discovery Exercise 4-7

Estimated Time: 15 minutes
Objective: Calculate default permissions.
Description: For each of the following umasks, calculate the default permissions given to new files and new directories:

a. 017
b. 272
c. 777
d. 000
e. 077
f. 027

Discovery Exercise 4-8

Estimated Time: 10 minutes
Objective: Outline secure umasks.
Description: For each of the umasks in Discovery Exercise 4-7, list the umasks that are reasonable to use to increase security on your Linux system and explain why.

Discovery Exercise 4-9

Estimated Time: 30 minutes
Objective: Configure umasks.
Description: Starting from the Linux default permissions for file and directories, what umask would you use to ensure that for all new_______________?

a. directories, the owner would have read, write, and execute; members of the group would have read and execute; and others would have read
b. files, the owner would have read and execute; the group would have read, write, and execute; and others would have execute
c. files, the owner would have write; the group would have read, write, and execute; and others would have read and write
d. directories, the owner would have read, write, and execute; the group would have read, write, and execute; and others would have read, write, and execute
e. directories, the owner would have execute; the group would have read, write, and execute; and others would have no permissions
f. files, the owner would have read and write; the group would have no permissions; and others would have write
g. directories, the owner would have read, write, and execute; the group would have read; and others would have read and execute
h. directories, the owner would have write; the group would have read, write, and execute; and others would have read, write, and execute
i. files, the owner would have no permissions; the group would have no permissions; and others would have no permissions

Discovery Exercise 4-10

Estimated Time: 30 minutes
Objective: Configure permissions.
Description: What permissions argument to the `chmod` command would you use to impose the following permissions?

a. on a directory such that: the owner would have read, write, and execute; the group would have read and execute; and others would have read
b. on a file such that: the owner would have read and write; the group would have no permissions; and others would have write
c. on a file such that: the owner would have write; the group would have read, write, and execute; and others would have read and write
d. on a file such that: the owner would have read and execute; the group would have read, write, and execute; and others would have execute
e. on a directory such that: the owner would have execute; the group would have read, write, and execute; and others would have no permissions
f. on a directory such that: the owner would have write; the group would have read, write, and execute; and others would have read, write, and execute
g. on a directory such that: the owner would have read, write, and execute; the group would have read; and others would have read and execute
h. on a directory such that: the owner would have read, write, and execute; the group would have read, write, and execute; and others would have read, write, and execute
i. on a file such that: the owner would have no permissions; the group would have no permissions; and others would have no permissions

Chapter 5

Linux Filesystem Administration

Chapter Objectives

1. Identify the structure and types of device files in the /dev directory.
2. Identify common filesystem types and their features.
3. Mount and unmount filesystems to and from the Linux directory tree.
4. Create and manage filesystems on hard disk drives, SSDs, and removable media storage devices.
5. Create and use ISO images.
6. Use the LVM to create and manage logical volumes.
7. Monitor free space on mounted filesystems.
8. Check filesystems for errors.
9. Use disk quotas to limit user space usage.

Navigating the Linux directory tree and manipulating files are common tasks that are performed daily by all users. However, administrators must provide this directory tree for users, as well as manage and fix the storage devices that support it. In this chapter, you learn about the various device files that represent storage devices and the different filesystems that can be placed on those devices. Next, you learn how to create and manage filesystems on a wide variety of different storage devices, as well as learn standard disk partitioning, LVM configuration, and filesystem management. The chapter concludes with a discussion of disk usage, filesystem errors, and restricting the ability of users to store files.

The /dev Directory

Fundamental to administering the disks used to store information is an understanding of how these disks are specified by the Linux operating system. Most standard devices on a Linux system (such as hard disk drives, SSDs, terminals, and serial ports) are represented by a file on the filesystem called a **device file**. There is one file per device, and these files are typically found in the **/dev directory**. This allows you to specify devices on the system by using the pathname to the file that represents it in the /dev directory.

Recall from Chapter 2 that the first partition on the first SATA/SCSI/SAS hard disk drive or SSD is identified by the installation program as sda1. When working with Linux utilities, you can specify the pathname to the file /dev/sda1 to refer to this partition.

Furthermore, each device file specifies how data should be transferred to and from the device. There are two methods for transferring data to and from a device. The first method involves transferring information character-by-character to and from the device. Devices that transfer data in this fashion are referred to as **character devices**. The second method transfers chunks or blocks of information at a time by using physical memory to buffer the transfer. Devices that use this method of transfer are called **block devices**; they can transfer information much faster than character devices. Device files that represent storage, such as CDs, DVDs, USB flash drives, hard disk drives, and SSDs, are typically block device files because they are formatted with a filesystem that organizes the available storage into discrete blocks that can be written to. Tape drives and most other devices, however, are typically represented by character device files.

To see whether a particular device transfers data character-by-character or block-by-block, recall that the `ls -l` command displays a c or b character in the type column indicating the type of device file. To view the type of the file /dev/sda1, you can use the following command:

```
[root@server1 ~]# ls -l /dev/sda1
brw-rw----    1 root     disk     8,    1 Feb 23 16:02 /dev/sda1
[root@server1 ~]#_
```

From the leftmost character in the preceding output, you can see that the /dev/sda1 file is a block device file. Table 5-1 lists common device files that you may find on your Linux system and their types.

Table 5-1 Common device files

Device File	Description	Block or Character
/dev/hda1	First partition on the first PATA hard disk drive (primary master)	Block
/dev/hdb1	First partition on the second PATA hard disk drive (primary slave)	Block
/dev/hdc1	First partition on the third PATA hard disk drive (secondary master)	Block
/dev/hdd1	First partition on the fourth PATA hard disk drive (secondary slave)	Block
/dev/sda1	First partition on the first SATA/SCSI/SAS hard disk drive or SSD	Block
/dev/sdb1	First partition on the second SATA/SCSI/SAS hard disk drive or SSD	Block
/dev/vda1	First partition on the first QEMU virtual disk	Block
/dev/vdb1	First partition on the second QEMU virtual disk	Block
/dev/nvme0n1p1	First partition in the first namespace on the first NVMe SSD	Block
/dev/nvme1n1p1	First partition in the first namespace on the second NVMe SSD	Block
/dev/zram0	First RAM disk used by zswap for virtual memory	Block
/dev/zram1	Second RAM disk used by zswap for virtual memory	Block
/dev/sr0	First writeable SATA CD or DVD device in the system	Block
/dev/loop0	First loopback interface	Block
/dev/tty1	First local terminal on the system (Ctrl+Alt+F1)	Character
/dev/tty2	Second local terminal on the system (Ctrl+Alt+F2)	Character
/dev/ttyS0	First serial port on the system (COM1)	Character
/dev/ttyS1	Second serial port on the system (COM2)	Character
/dev/psaux	PS/2 mouse port	Character
/dev/lp0	First parallel port on the system (LPT1)	Character
/dev/null	Device file that represents nothing; any data sent to this device is discarded	Character

(continues)

Table 5-1 Common device files (continued)

Device File	Description	Block or Character
/dev/zero	Device file that produces NULL (empty) characters; it can be used within commands to generate sample input	Character
/dev/random /dev/urandom	Device file that produces pseudorandom numbers; it can be used to provide random numbers for use within commands	Character
/dev/st0	First SCSI tape device in the system	Character
/dev/bus/usb/*	USB device files	Block or Character

Note 1

If a device file is not present on your system, the underlying hardware was not detected by the **udev daemon**, which is responsible for automatically creating device files as necessary.

After a typical Fedora Linux installation, you will find several hundred different device files in the /dev directory that represent devices on the system. This large number of device files on a Linux system does not require much disk space because all device files consist of inodes and no data blocks; as a result, the entire contents of the /dev directory is 0 KB in size unless other regular files are stored within it. When using the `ls -l` command to view device files, the portion of the listing describing the file size is replaced by two numbers: the major number and the minor number. The **major number** of a device file points to the device driver for the device in the Linux kernel; several devices can share the same major number if they are of the same general type (i.e., two different SATA devices might share the same major number as they use the same driver in the Linux kernel). The **minor number** indicates the particular device itself; in the case of storage devices, different minor numbers are used to represent different partitions as shown in the following output:

```
[root@server1 ~]# ls -l /dev/sda*
brw-rw----. 1 root disk 8, 0 Sep  7 09:41 /dev/sda
brw-rw----. 1 root disk 8, 1 Sep  7 09:41 /dev/sda1
brw-rw----. 1 root disk 8, 2 Sep  7 09:41 /dev/sda2
brw-rw----. 1 root disk 8, 3 Sep  7 09:41 /dev/sda3
[root@server1 ~]# ls -l /dev/sdb*
brw-rw----. 1 root disk 8, 16 Sep  7 10:18 /dev/sdb
brw-rw----. 1 root disk 8, 17 Sep  7 10:18 /dev/sdb1
brw-rw----. 1 root disk 8, 18 Sep  7 10:18 /dev/sdb2
brw-rw----. 1 root disk 8, 19 Sep  7 10:18 /dev/sdb3
[root@server1 ~]#_
```

Note 2

In the previous output, note that /dev/sdb* shares the same major number as /dev/sda* because they use the same driver in the Linux kernel. Because it is rare to create more than 15 partitions on a single device, Linux starts minor numbers for additional devices of the same type in increments of 16; thus, the minor number for /dev/sdb is 16, and the minor number for /dev/sdc is 32, and so on. If you create more than 15 partitions on a single device, this minor numbering scheme is automatically adjusted by the udev daemon.

Together, the device file type (block or character), the major number (device driver), and the minor number (specific device) make up the unique characteristics of each device file. To create a device file, you need to know these three pieces of information.

If a device file becomes corrupted, it is usually listed as a regular file instead of a block or character special file. Recall from Chapter 4 that you can use the `find /dev -type f` command to search for regular files under the /dev directory to identify whether corruption has taken place. If you find a corrupted device file or accidentally delete a device file, the **`mknod` command** can be used to re-create the device file if you know the type and major and minor numbers. An example of re-creating the /dev/sda1 block device file used earlier with a major number of 8 and a minor number of 1 is shown in the following example:

```
[root@server1 ~]# mknod /dev/sda1 b 8 1
[root@server1 ~]# ls -l /dev/sda1
brw-rw----. 1 root disk 8, 1 Oct  9 15:02 /dev/sda1
[root@server1 ~]#_
```

To see a list of block and character devices that are currently used on the system and their major numbers, you can view the contents of the **/proc/devices** file. To view the block devices on the system, you can view the contents of the /sys/block directory or run the **`lsblk` command**. These methods are shown below:

```
[root@server1 ~]# cat /proc/devices
Character devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  7 vcs
 10 misc
 13 input
 14 sound
 21 sg
 29 fb
116 alsa
128 ptm
136 pts
162 raw
180 usb
188 ttyUSB
189 usb_device
202 cpu/msr
203 cpu/cpuid
240 usbmon

Block devices:
  8 sd
  9 md
 11 sr
 65 sd
128 sd
252 zram
253 device-mapper
254 mdp
259 blkext
```

```
[root@server1 ~]#_
[root@server1 ~]# ls /sys/block
sda  sdb  sr0
[root@server1 ~]# lsblk
NAME    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda       8:0    0   50G  0 disk
├─sda1    8:1    0  600M  0 part /boot/efi
├─sda2    8:2    0    1G  0 part /boot
└─sda3    8:3    0   35G  0 part /
sdb       8:16   0   40G  0 disk
├─sdb1    8:17   0    5G  0 part /var
├─sdb2    8:18   0   20G  0 part /var/spool
└─sdb3    8:19   0   15G  0 part /var/log
sr0      11:0    1 1024M  0 rom
zram0   252:0    0  3.8G  0 disk [SWAP]
[root@server1 ~]#_
```

Note 3

You can also use the **udevadm command** to view detailed information for a particular device file. For example, `udevadm info /dev/sda` will display details for the first SATA/SCSI/SAS storage device on the system, and `udevadm info /dev/sda1` will display details for the first partition on that storage device.

Filesystems

Recall from Chapter 2 that files must be stored within a partition on the hard disk drive or SSD in a defined format called a filesystem so that the operating system can work with them. The type of filesystem used determines how files are managed on the physical storage device. Each filesystem can have different methods for storing files and features that make the filesystem robust against errors. Although many types of filesystems are available, all filesystems share three common components, as discussed in Chapter 4: the superblock, the inode table, and the data blocks. On a structural level, these three components work together to organize files and allow rapid access to, and retrieval of, data. As discussed in Chapter 2, journaling filesystems contain a fourth component called a journal that keeps track of changes that are to be written to the filesystem. In the event of a power outage, the filesystem can check the journal to complete any changes that were not performed to prevent filesystem errors related to the power outage. All storage media, such as hard disk drives, SSDs, USB flash drives, and DVDs, need to contain a filesystem before they can be used by the Linux operating system.

Note 4

Creating a filesystem on a device is commonly referred to as **formatting**.

Filesystem Types

As mentioned, many filesystems are available for use in the Linux operating system. Each has its own strengths and weaknesses; thus, some are better suited to some tasks and not as well suited to others. One benefit of Linux is that you need not use only one type of filesystem on the system; you can use several partitions formatted with different filesystems under the same directory tree. In addition, files and directories appear the same throughout the directory tree regardless of whether there is one filesystem or 20 filesystems in use by the Linux system. Table 5-2 lists some common filesystems available for use in Linux.

Table 5-2 Common Linux filesystems

Filesystem	Description
btrfs	B-tree File System—A new filesystem for Linux systems that includes many features that are geared toward large-scale storage, including compression, subvolumes, quotas, snapshots, and the ability to span multiple devices. While relatively new and still in development, it is envisioned to be a replacement for the ext4 filesystem in the long term. Its name is commonly pronounced as "Butter F S." You will learn how to configure btrfs in Chapter 6.
exFAT	Extended FAT filesystem—An improved version of the FAT filesystem with large file support. It is the most common filesystem used on removeable storage devices such as USB flash drives and portable USB hard drives, as it has full support on modern Linux, macOS, and Windows operating systems.
ext2	Second extended filesystem—The traditional filesystem used on Linux, it supports access control lists (individual user permissions). In addition, it retains its name from being the new version of the original extended filesystem, based on the Minix filesystem.
ext3	Third extended filesystem—A variation on ext2 that allows for journaling and, thus, has a faster startup and recovery time.
ext4	Fourth extended filesystem—A variation on ext3 that has larger filesystem support and speed enhancements.
iso9660	ISO 9660 filesystem—A filesystem that originated from the International Standards Organization recommendation 9660 that is used by ISO images, CDs, and DVDs.
msdos, fat	File Allocation Table (FAT) filesystem. It can use a 12, 16, or 32-bit table to store file locations.
ntfs	New Technology File System (NTFS)—A Microsoft proprietary filesystem developed for its Windows operating systems. Due to license restrictions, Linux systems can read from, but not write to, NTFS filesystems.
udf	Universal Disk Format (UDF) filesystem—A DVD filesystem originally intended as a modern replacement for the ISO 9660 filesystem.
vfat	FAT filesystem with long filename support.
xfs	X File System (XFS)—A very high-performance filesystem created by Silicon Graphics for use on their IRIX UNIX systems. Many Linux administrators prefer to use xfs on systems that need to quickly write large numbers of files to the filesystem.
zfs	Zettabyte File System (ZFS)—A very high-performance filesystem and volume manager originally created by Sun Microsystems that protects against data corruption and has features that support very large distributed storage systems. Many large-scale Linux server systems in industry use the zfs filesystem to store and manage large amounts of data. You will learn how to configure zfs in Chapter 6.

Note 5

Filesystem support is typically built into the Linux kernel or added as a package on most distributions.

Mounting

The term **mounting** originated in the 1960s when information was stored on large tape reels that had to be mounted on computers to make the data available. Today, mounting still refers to making data available. More specifically, it refers to the process whereby a device is made accessible to users via the logical directory tree. This device is attached to a certain directory on the directory tree called a **mount point**. Users can then create files and subdirectories in this mount point directory, which are then stored on the filesystem that was mounted to that particular directory.

Remember that directories are merely files that do not contain data; instead, they contain a list of files and subdirectories organized within them. Thus, it is easy for the Linux system to cover up directories to prevent user access to that data. This is essentially what happens when a device is mounted to a certain directory; the mount point directory is temporarily covered up by that device while the device remains mounted. Any file contents that were present in the mount point directory prior to mounting are not lost; when the device is unmounted, the mount point directory is uncovered, and the previous file contents are revealed. Suppose, for example, that you mount a USB flash drive that contains a filesystem to the /mnt directory. The /mnt directory is an empty directory that is commonly used as a temporary mount point for mounting removable media devices. Before mounting, the directory structure would resemble that depicted in Figure 5-1. After the USB flash drive is mounted to the /mnt directory, the contents of the /mnt directory would be covered up by the filesystem on the USB flash drive, as illustrated in Figure 5-2.

Figure 5-1 The directory structure prior to mounting

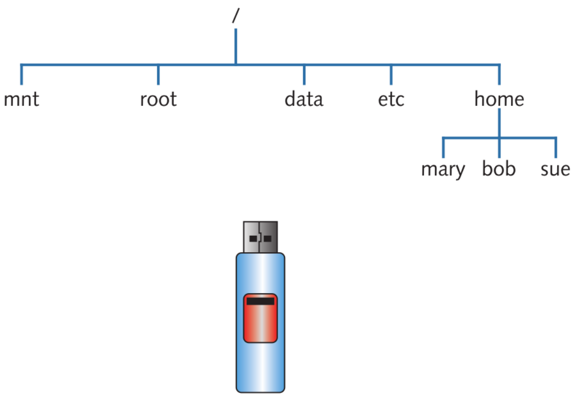
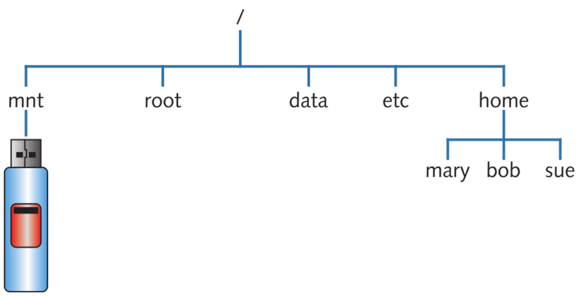


Figure 5-2 The directory structure after mounting a USB flash drive



If a user then stores a file in the /mnt directory, as shown in Figure 5-2, that file will be stored in the filesystem on the USB flash drive. Similarly, if a user creates a subdirectory under the /mnt directory depicted in Figure 5-2, that subdirectory will be made in the filesystem on the USB flash drive.

Any existing directory can be used as a mount point. If a user mounts a USB flash drive to the /usr/bin directory, all files in the /usr/bin directory are covered up during the time the drive is mounted, including the command used to unmount the drive. Thus, it is safe practice to create empty directories used specifically for mounting devices to avoid making existing files inaccessible to users.

Note 6

Most systems today have several removable media devices such as CDs, DVDs, USB flash drives, and USB hard drives that may be connected to the computer for long periods of time. As a result, it is considered good form to create subdirectories under the /media directory on your Linux system to mount these removable media devices and only use the /mnt directory to temporarily mount devices. For example, you could mount your USB flash drive to the /media/USBdrive directory and your DVD to the /media/DVD directory. You can then access the files on your USB flash drive by navigating to the /media/USBdrive directory, as well as access the files on your DVD by navigating to the /media/DVD directory.

When the Linux system is first turned on, a filesystem on the hard drive is mounted to the / directory. This is referred to as the **root filesystem** and contains most of the operating system files. Other filesystems on storage devices inside the computer can also be mounted to various mount point directories under the / directory at boot time, as well as via entries in the filesystem table (**/etc/fstab**) discussed in the following sections.

The **`mount` command** is used to mount devices to mount point directories, and the **`umount` command** is used to unmount devices from mount point directories; both of these commands are discussed throughout the remainder of this chapter.

Working with USB Flash Drives

The most common type of removable media used to store files that need to be transferred from computer to computer are USB flash drives. USB flash drives are recognized as SCSI drives by operating systems, and often ship with a single partition formatted with the DOS FAT or exFAT filesystem. However, you can reformat the filesystem on this partition with a different filesystem of your choice after determining the appropriate device file. If your system has a single SATA hard disk drive (/dev/sda), the first USB flash drive inserted into your system will be recognized as /dev/sdb, and the partition on it can be represented by /dev/sdb1. To verify that your USB flash drive model (e.g., Kingston DataTraveler) was recognized by the system after inserting it, you can use the **`lsusb` command**:

```
[root@server1 ~]# lsusb
Bus 001 Device 003: ID 0930:6545 Toshiba Corp. Kingston DataTraveler
Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
[root@server1 ~]#_
```

To verify the device file used to represent the partition on your USB flash drive, you can use the **`lsblk`** command; a number `1` in the `RM` column indicates that the device is removable storage. The **`lsblk`** output shown below indicates that the first partition on the first removeable storage device is /dev/sdb1, and that it is not currently mounted to a directory:

```
[root@server1 ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   50G  0 disk
├─sda1   8:1    0  600M  0 part /boot/efi
├─sda2   8:2    0    1G  0 part /boot
└─sda3   8:3    0   35G  0 part /
sdb      8:16   1    8G  0 disk
└─sdb1   8:17   1    8G  0 part
sr0     11:0    1 1024M  0 rom
zram0  252:0    0  3.8G  0 disk [SWAP]
[root@server1 ~]#_
```

To create a new filesystem on a USB flash drive, you can use the **mkfs (make filesystem) command** and specify the filesystem type using the `-t` switch and the device file representing the partition. Thus, to format /dev/sdb1 with the ext2 filesystem you can type the following command:

```
[root@server1 ~]# mkfs -t ext2 /dev/sdb1
mke2fs 1.46.5 (30-Dec-2023)
Creating filesystem with 1952512 4k blocks and 488640 inodes
Filesystem UUID: 95b7fdcd-8baa-4e3f-ad6b-1ff5b5ba1f2f
Superblock backups stored on blocks:
       32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
[root@server1 ~]#_
```

Note 7

Note from the previous output that each newly formatted Linux filesystem is given a **Universally Unique Identifier (UUID)** that can be used to identify the filesystem. This UUID can be used to identify filesystems that need to be mounted at boot time, as discussed later in this chapter.

Alternatively, you can specify a different filesystem after the `-t` option, such as the FAT filesystem with long filename support (vfat). This results in output different from the `mkfs` command, as shown in the following example:

```
[root@server1 ~]# mkfs -t vfat /dev/sdb1
mkfs.fat 4.2 (2023-01-31)
[root@server1 ~]#_
```

Because the FAT filesystem is universally supported by Windows, macOS, UNIX, and Linux systems, it is often used on removable media. However, the maximum size of a FAT filesystem is 32 GB. For USB flash drives and USB hard disk drives or SSDs larger than 32 GB, you should instead use the exFAT filesystem, which has a maximum filesystem size of 128 PB (petabytes) and is universally supported by modern versions of Windows, macOS, UNIX, and Linux. To format /dev/sdb1 with the exFAT filesystem, you can run the following command:

```
[root@server1 ~]# mkfs -t exfat /dev/sdb1
Creating exFAT filesystem(/dev/sdb1, cluster size=32768)

Writing volume boot record: done
Writing backup volume boot record: done
Fat table creation: done
Allocation bitmap creation: done
Upcase table creation: done
Writing root directory entry: done
Synchronizing...

exFAT format complete!
[root@server1 ~]#_
```

If you do not specify the filesystem using the `mkfs` command, the default filesystem assumed is the ext2 filesystem as shown below:

```
[root@server1 ~]# mkfs /dev/sdb1
mke2fs 1.46.5 (30-Dec-2023)
Creating filesystem with 1952512 4k blocks and 488640 inodes
```

```
Filesystem UUID: 95b7fdcd-8baa-4e3f-ad6b-1ff5b5ba1f2f
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
[root@server1 ~]#_
```

Although the most common command to create filesystems is the `mkfs` command, you can use other variants and shortcuts to the `mkfs` command. For example, to create an ext2 filesystem, you could type `mke2fs /dev/sdb1` on the command line. Other alternatives to the `mkfs` command are listed in Table 5-3.

Table 5-3 Commands used to create filesystems

Command	Filesystem It Creates
`mkfs`	Filesystems of most types

`mkdosfs` `mkfs.msdos`
`mkfs.fat`
`mkfs.vfat`
	FAT (12, 16, or 32-bit, depending on the size of the filesystem)

`mkfs.ext2` `mke2fs`	ext2
`mkfs.ext3` `mke2fs -t ext3`	ext3
`mkfs.ext4` `mke2fs -t ext4`	ext4
`mkisofs`	ISO 9660
`mkfs.xfs`	XFS
`mkudffs` `mkfs.udf`	UDF
`mkntfs` `mkfs.ntfs`	NTFS
`mkexfatfs` `mkfs.exfat`	exFAT

After a USB flash drive has been formatted with a filesystem, it must be mounted to the directory tree before it can be used. A list of currently mounted filesystems can be obtained by using the `mount` command with no options or arguments, which reads the information listed in the **/etc/mtab** (mount table) file. On modern systems, the output of this command is quite lengthy as it contains many special filesystems and filesystem parameters. Consequently, it is much easier to see a list of currently mounted filesystems using the `lsblk` command or the **`df` (disk free space) command**. The `-T` option to the `df` command will also print the filesystem type, and the `-h` option displays friendly (human readable) size formats, as shown in the following sample output:

```
[root@server1 ~]# df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  4.0M     0  4.0M   0% /dev
```

```
tmpfs          tmpfs      2.0G     0  2.0G   0% /dev/shm
tmpfs          tmpfs      784M  1.2M  783M   1% /run
/dev/sda3      ext4        35G  8.9G   24G  28% /
tmpfs          tmpfs      2.0G  8.0K  2.0G   1% /tmp
/dev/sda2      ext4       974M  150M  757M  17% /boot
/dev/sda1      vfat       599M   14M  585M   3% /boot/efi
tmpfs          tmpfs      392M   92K  392M   1% /run/user/42
tmpfs          tmpfs      392M   76K  392M   1% /run/user/0
tmpfs          tmpfs      392M   76K  392M   1% /run/user/1000
[root@server1 ~]#_
```

From the preceding example output, you can see that the ext4 filesystem on /dev/sda3 is mounted to the / directory, the ext4 filesystem on /dev/sda2 is mounted to the /boot directory, and the vfat filesystem on /dev/sda1 is mounted to the /boot/efi directory. The other filesystems listed are special filesystems that are used by the system; these filesystems are called **virtual filesystems** or **pseudo filesystems** and are discussed later in this book.

To mount a device on the directory tree, you can use the `mount` command with options and arguments to specify the filesystem type, the device to mount, and the directory on which to mount the device (mount point). It is important to ensure that no user is currently using the mount point directory; otherwise, the system gives you an error message and the device is not mounted. To check whether the /media/USBdrive directory is being used by any users, you can use the **`fuser` command** with the `–u` option, as shown in the following output:

```
[root@server1 ~]# fuser -u /media/USBdrive
[root@server1 ~]#_
```

The preceding output indicates the /media/USBdrive directory is not being used by any user processes; to mount a USB flash drive (/dev/sdb1) formatted with the ext2 filesystem to this directory, you could run the following command:

```
[root@server1 ~]# mount –t ext2 /dev/sdb1 /media/USBdrive
[root@server1 ~]# df -hT
Filesystem     Type       Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs   4.0M     0  4.0M   0% /dev
tmpfs          tmpfs      2.0G     0  2.0G   0% /dev/shm
tmpfs          tmpfs      784M  1.2M  783M   1% /run
/dev/sda3      ext4        35G  8.9G   24G  28% /
tmpfs          tmpfs      2.0G  8.0K  2.0G   1% /tmp
/dev/sda2      ext4       974M  150M  757M  17% /boot
/dev/sda1      vfat       599M   14M  585M   3% /boot/efi
tmpfs          tmpfs      392M   92K  392M   1% /run/user/42
tmpfs          tmpfs      392M   76K  392M   1% /run/user/0
tmpfs          tmpfs      392M   76K  392M   1% /run/user/1000
/dev/sdb1      ext2       7.9G   24K  7.5G   1% /media/USBdrive
[root@server1 ~]#_
```

Note 8

If you omit the `–t` option to the `mount` command, it attempts to automatically detect the filesystem on the device. Thus, the command `mount /dev/sdb1 /media/USBdrive` will perform the same action as the `mount` command shown in the preceding output.

Notice that /dev/sdb1 is mounted to the /media/USBdrive directory in the preceding output of the `mount` command. To access and store files on the USB flash drive, you can now treat the /media/USBdrive directory as the root of the USB flash drive. When an ext2, ext3, or ext4 filesystem is created on a device, one directory called lost+found is created by default and used by the `fsck` command discussed later in this chapter. To explore the recently mounted filesystem, you can use the following commands:

```
[root@server1 ~]# cd /media/USBdrive
[root@server1 USBdrive]# pwd
/media/USBdrive
[root@server1 USBdrive]# ls -F
lost+found/
[root@server1 USBdrive]#_
```

To copy files to the USB flash drive, specify the /media/USBdrive directory as the target for the `cp` command, as follows:

```
[root@server1 USBdrive]# cd /etc
[root@server1 etc]# cat issue
\S
Kernel \r on an \m (\l)
[root@server1 etc]# cp issue /media/USBdrive
[root@server1 etc]# cd /media/USBdrive
[root@server1 USBdrive]# ls -F
issue  lost+found/
[root@server1 USBdrive]# cat issue
\S
Kernel \r on an \m (\l)
[root@server1 USBdrive]#_
```

Similarly, you can also create subdirectories under the USB flash drive to store files; these subdirectories are referenced under the mount point directory. To make a directory called workfiles on the USB flash drive mounted in the previous example and copy the /etc/inittab file to it, you can use the following commands:

```
[root@server1 USBdrive]# pwd
/media/USBdrive
[root@server1 USBdrive]# ls -F
issue  lost+found/
[root@server1 USBdrive]# mkdir workfiles
[root@server1 USBdrive]# ls -F
issue  lost+found/  workfiles
[root@server1 USBdrive]# cd workfiles
[root@server1 workfiles]# pwd
/media/USBdrive/workfiles
[root@server1 workfiles]# cp /etc/inittab .
[root@server1 workfiles]# ls
inittab
[root@server1 workfiles]#_
```

Even though you can remove a USB flash drive without permission from the system, doing so is likely to cause error messages to appear on the terminal screen. Before a USB flash drive is removed, it must be properly unmounted using the `umount` command. The `umount` command can take the name of the device to unmount or the mount point directory as an argument. Similar to mounting a device, unmounting a device requires that the mount point directory has no users using it. If you try to unmount the USB flash drive mounted to the /media/USBdrive directory while it is being used, you receive an error message similar to the one in the following example:

```
[root@server1 USBdrive]# pwd
/media/USBdrive
[root@server1 USBdrive]# umount /media/USBdrive
umount: /media/USBdrive: target is busy.
[root@server1 USBdrive]# fuser -u /media/USBdrive
/media/USBdrive:         17368c(root)
[root@server1 USBdrive]# cd /root
[root@server1 ~]# umount /media/USBdrive
[root@server1 ~]# df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs          tmpfs     2.0G     0  2.0G   0% /dev/shm
tmpfs          tmpfs     784M  1.2M  783M   1% /run
/dev/sda3      ext4       35G  8.9G   24G  28% /
tmpfs          tmpfs     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2      ext4      974M  150M  757M  17% /boot
/dev/sda1      vfat      599M   14M  585M   3% /boot/efi
tmpfs          tmpfs     392M   92K  392M   1% /run/user/42
tmpfs          tmpfs     392M   76K  392M   1% /run/user/0
tmpfs          tmpfs     392M   76K  392M   1% /run/user/1000
[root@server1 ~]#_
```

Notice from the preceding output that you were still using the /media/USBdrive directory because it was the current working directory. The `fuser` command also indicated that the root user had a process using the directory. After the current working directory was changed, the `umount` command was able to unmount the USB flash drive from the /media/USBdrive directory, and the output of the `mount` command indicated that it was no longer mounted.

Recall that mounting simply attaches a disk device to the Linux directory tree so that you can treat the device like a directory full of files and subdirectories. A device can be mounted to any existing directory. However, if the directory contains files, those files are inaccessible until the device is unmounted. Suppose, for example, that you create a directory called /USBdrive for mounting USB flash drives and a file inside called samplefile, as shown in the following output:

```
[root@server1 ~]# mkdir /USBdrive
[root@server1 ~]# touch /USBdrive/samplefile
[root@server1 ~]# ls /USBdrive
samplefile
[root@server1 ~]#_
```

If the USB flash drive used earlier is mounted to the /USBdrive directory, a user who uses the /USBdrive directory will be using the filesystem on the USB flash drive; however, when nothing is mounted to the /USBdrive directory, the previous contents are available for use:

```
[root@server1 ~]# mount /dev/sdb1 /USBdrive
[root@server1 ~]# df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs          tmpfs     2.0G     0  2.0G   0% /dev/shm
tmpfs          tmpfs     784M  1.2M  783M   1% /run
/dev/sda3      ext4       35G  8.9G   24G  28% /
tmpfs          tmpfs     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2      ext4      974M  150M  757M  17% /boot
```

```
/dev/sda1       vfat      599M   14M  585M   3% /boot/efi
tmpfs           tmpfs     392M   92K  392M   1% /run/user/42
tmpfs           tmpfs     392M   76K  392M   1% /run/user/0
tmpfs           tmpfs     392M   76K  392M   1% /run/user/1000
/dev/sdb1       ext2      7.9G   24K  7.5G   1% /USBdrive
[root@server1 ~]# ls -F /USBdrive
issue  lost+found/  workfiles
[root@server1 ~]# umount /USBdrive
[root@server1 ~]# ls /USBdrive
samplefile
[root@server1 ~]#_
```

The mount command used in the preceding output specifies the filesystem type, the device to mount, and the mount point directory. To save time typing on the command line, you can alternatively specify one argument and allow the system to look up the remaining information in the /etc/fstab (filesystem table) file. The /etc/fstab file has a dual purpose; it is used to mount devices at boot time and is consulted when a user does not specify enough arguments on the command line when using the mount command.

The /etc/fstab file has six fields:

<device to mount> <mount point> <type> <mount options> <dump#> <fsck#>

The device to mount can be the path to a device file (e.g., /dev/sda1), the filesystem UUID (e.g., UUID=db545f1d-c1ee-4b70-acbe-3dc61b41db20), the GPT partition UUID (e.g., PARTUUID=ed835873-01), or the filesystem label (e.g., LABEL=BackupDisk). The mount point specifies the directory to which the device should be mounted. The type can be a specific value (such as ext4) or can be automatically detected. The mount options are additional options that the mount command accepts when mounting the volume (such as read only, or "ro"). Any filesystems with the mount option "noauto" are not automatically mounted at boot time; a complete list of options that the mount command accepts can be found by viewing the manual page for the mount command.

The dump# is used by the dump command (discussed in Chapter 11) when backing up filesystems; a 1 in this field indicates that the filesystem should be backed up, whereas a 0 indicates that no backup is necessary. The fsck# is used by the fsck command discussed later in this chapter when checking filesystems at boot time for errors; any filesystems with a 1 in this field are checked before any filesystems with a number 2, and filesystems with a number 0 are not checked.

Note 9

You can use the **`blkid` command** to display the filesystem and partition UUIDs on your system. You can also use the `lsblk --fs` command to display filesystem UUIDs and labels.

Note 10

Filesystem labels are optional; to set a label on a filesystem, you must use a command for the filesystem type. For example, the **`e2label` command** can be used to set a label on an ext2/ext3/ext4 filesystem, the **`fatlabel` command** can be used to set a label on a FAT filesystem, the **`exfatlabel` command** can be used to set a label on an exFAT filesystem, and the **`xfs_admin` command** can be used to set a label on an XFS filesystem.

Note 11

To easily associate filesystem UUIDs and labels with their device files, the udev daemon creates symbolic links for each UUID and label in subdirectories under /dev/disk that point to the correct device file (e.g., /dev/sdb1); /dev/disk/by-uuid stores symbolic links by filesystem UUID, /dev/disk/by-partuuid stores symbolic links by GPT partition UUID, and /dev/disk/by-label stores symbolic links by filesystem label. You can also find symbolic links for each disk by Linux kernel identifier under /dev/disk/by-id and by PCI bus identifier under /dev/disk/by-path.

Note 12

To mount all filesystems in the /etc/fstab file that are intended to mount at boot time, you can type the `mount -a` command.

The following output displays the contents of a sample /etc/fstab file:

```
[root@server1 ~]# cat /etc/fstab
# 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.
#
# After editing this file, run 'systemctl daemon-reload' to update
# systemd units generated from this file.
#
UUID=e8ffcefc-8a11-474e-b879-d7adb1fd1e94 /         ext4 defaults  1 1
UUID=c203af16-e9b2-4577-aab5-5f353f0f2074 /boot     ext4 defaults  1 2
UUID=F4D6-9E57                            /boot/efi vfat umask=077 0 2
/dev/sdb1                                 /USBdrive auto noauto    0 0
[root@server1 ~]#_
```

Thus, to mount the first USB flash drive (/dev/sdb1) to the /USBdrive directory and automatically detect the type of filesystem on the device, specify enough information for the `mount` command to find the appropriate line in the /etc/fstab file:

```
[root@server1 ~]# mount /dev/sdb1
[root@server1 ~]# df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs          tmpfs     2.0G     0  2.0G   0% /dev/shm
tmpfs          tmpfs     784M  1.2M  783M   1% /run
/dev/sda3      ext4       35G  8.9G   24G  28% /
tmpfs          tmpfs     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2      ext4      974M  150M  757M  17% /boot
/dev/sda1      vfat      599M   14M  585M   3% /boot/efi
tmpfs          tmpfs     392M   92K  392M   1% /run/user/42
tmpfs          tmpfs     392M   76K  392M   1% /run/user/0
tmpfs          tmpfs     392M   76K  392M   1% /run/user/1000
/dev/sdb1      ext2      7.9G   24K  7.5G   1% /USBdrive
[root@server1 ~]# umount /dev/sdb1
[root@server1 ~]#_
```

The `mount` command in the preceding output succeeded because a line in /etc/fstab described the mounting of the /dev/sdb1 device. Alternatively, you could specify the mount point as an argument to the `mount` command to mount the same device via the correct entry in /etc/fstab:

```
[root@server1 ~]# mount /USBdrive
[root@server1 ~]# df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs          tmpfs     2.0G     0  2.0G   0% /dev/shm
tmpfs          tmpfs     784M  1.2M  783M   1% /run
/dev/sda3      ext4       35G  8.9G   24G  28% /
tmpfs          tmpfs     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2      ext4      974M  150M  757M  17% /boot
```

```
/dev/sda1      vfat     599M  14M  585M  3% /boot/efi
tmpfs          tmpfs    392M  92K  392M  1% /run/user/42
tmpfs          tmpfs    392M  76K  392M  1% /run/user/0
tmpfs          tmpfs    392M  76K  392M  1% /run/user/1000
/dev/sdb1      ext2     7.9G  24K  7.5G  1% /USBdrive
[root@server1 ~]# umount /USBdrive
[root@server1 ~]#_
```

Table 5-4 lists commands that are useful when mounting and unmounting USB flash drives.

Table 5-4 Useful commands when mounting and unmounting filesystems

Command	Description
mount df -hT lsblk --fs	Displays mounted filesystems and their type
mount -t <type> <device> <mount point>	Mounts a <device> of a certain <type> to a <mount point> directory
fuser -u <directory>	Displays the users using a particular directory
umount <mount point> *or* umount <device>	Unmounts a <device> from its <mount point> directory

Working with CDs, DVDs, and ISO Images

CDs and DVDs are another form of removeable media used by some systems today. Like USB flash drives, CDs and DVDs can be mounted with the `mount` command and unmounted with the `umount` command, as shown in Table 5-4; however, the device file used with these commands is different. The device files used by CD and DVD drives depend on the technology used by the drive itself. To make the identification of your CD or DVD drive easier, Linux creates a symbolic link to the correct device file for your first CD or DVD drive called /dev/cdrom. For example, if your system contains a writable SATA CD/DVD drive, a long listing of /dev/cdrom may show the following:

```
[root@server1 ~]# ls -l /dev/cdrom
lrwxrwxrwx. 1 root root 3 Jul 19 07:51 /dev/cdrom -> sr0
[root@server1 ~]#_
```

In this case, you could use /dev/cdrom or /dev/sr0 to mount CDs or DVDs. To write to a CD or DVD in this same drive, however, you must use disc burning software that knows how to write data to /dev/cdrom or /dev/sr0.

Note 13

Nearly all DVD drives can also work with CDs.

Note 14

Many OSS disc burning software applications are available for Linux. One example is the graphical Brasero Disc Burner program that you can install by running the command `dnf install brasero` as the root user.

In addition, CDs and DVDs typically use either the ISO 9660 or UDF filesystem type and are read-only when accessed using Linux (recall that you must use disc burning software to record to a CD or DVD). Thus, to mount a CD or DVD to a directory, you should use the `-r` (read-only) option to the `mount` command to avoid warnings. To mount a sample CD to the /media/CD directory and view its contents, you could use the following commands:

```
[root@server1 ~]# mount -r /dev/cdrom /media/CD
[root@server1 ~]# df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs          tmpfs     2.0G     0  2.0G   0% /dev/shm
tmpfs          tmpfs     784M  1.2M  783M   1% /run
/dev/sda3      ext4       35G  8.9G   24G  28% /
tmpfs          tmpfs     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2      ext4      974M  150M  757M  17% /boot
/dev/sda1      vfat      599M   14M  585M   3% /boot/efi
tmpfs          tmpfs     392M   92K  392M   1% /run/user/42
tmpfs          tmpfs     392M   76K  392M   1% /run/user/0
tmpfs          tmpfs     392M   76K  392M   1% /run/user/1000
/dev/sr0       iso9660   430M  430M    0M 100% /media/CD
[root@server1 ~]# ls -F /media/CD
autorun.inf*   install*   graphics/   jungle/   jungle.txt*    joystick/
[root@server1 ~]# umount /media/CD
[root@server1 ~]#_
```

As with USB flash drives, you can modify the /etc/fstab file such that you can specify only a single argument to the `mount` command to mount a CD or DVD. Also remember that the mount point directory must not be in use to successfully mount or unmount CDs and DVDs; the `fuser` command can be used to verify this.

Unlike USB flash drives, CDs and DVDs cannot be ejected from the drive until they are properly unmounted, because the `mount` command locks the CD/DVD drive as a precaution. Alternatively, you can use the **`eject` command**, which unmounts the filesystem and forces the CD or DVD drive to physically eject the disc.

The ISO 9660 filesystem type is not limited to CDs and DVDs. ISO images, like the one you installed a Fedora Linux virtual machine with Hands-On Project 2-1, also contain an ISO 9660 filesystem. These files can be easily written to a CD or DVD using disc burning software or mounted and accessed by your Linux system. If you download an ISO image called sample.iso, you can mount it to the /mnt directory as a read-only loopback device. This allows your system to access the contents of the sample.iso file, as shown here:

```
[root@server1 ~]# mount -o loop -r -t iso9660 sample.iso /mnt
[root@server1 ~]# df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs          tmpfs     2.0G     0  2.0G   0% /dev/shm
tmpfs          tmpfs     784M  1.2M  783M   1% /run
/dev/sda3      ext4       35G  8.9G   24G  28% /
tmpfs          tmpfs     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2      ext4      974M  150M  757M  17% /boot
/dev/sda1      vfat      599M   14M  585M   3% /boot/efi
tmpfs          tmpfs     392M   92K  392M   1% /run/user/42
tmpfs          tmpfs     392M   76K  392M   1% /run/user/0
tmpfs          tmpfs     392M   76K  392M   1% /run/user/1000
/dev/loop0     iso9660   364M  364M     0 100% /mnt
```

```
[root@server1 ~]# ls /mnt
setup.exe      tools     binaries
[root@server1 ~]#_
```

You can then view or execute files within the /mnt directory or copy files from the /mnt directory to another directory to extract the contents of the ISO image.

To create a new ISO image from a directory of files, you can use the **mkisofs command**. The following command creates an ISO image called newimage.iso that contains all of the files and subdirectories under the /data directory with additional support for the Rock Ridge (-R) and Joliet (-J) standards:

```
[root@server1 ~]# mkisofs -RJ -o newimage.iso /data
I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 256
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
182 extents written (0 MB)
[root@server1 ~]#_
```

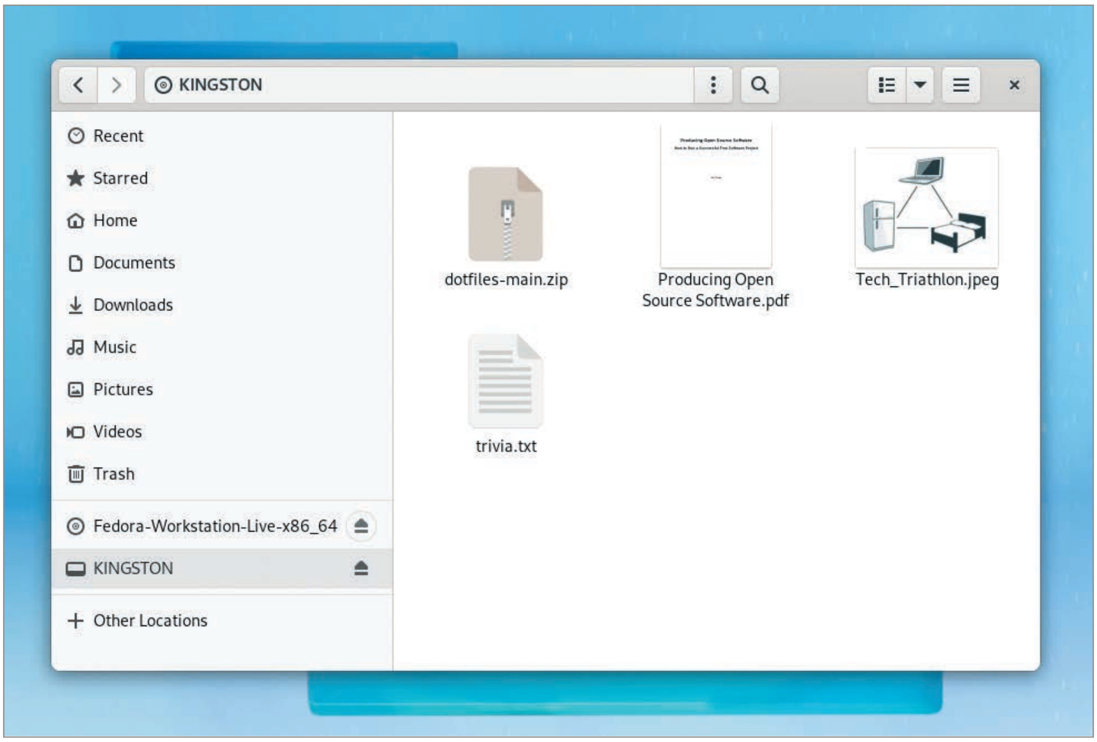
Working with Removeable Media within a Desktop Environment

Because of their large capacity and portability, removable storage devices often store copies of data, pictures, music, movies, programs, and documents that users regularly use within a desktop environment. When working within a desktop environment, a process automatically mounts removeable media to a directory so that you can work with it immediately, much like on a Windows or macOS computer. When you insert a USB flash drive, CD, or DVD while in a desktop environment, it is automatically mounted by the system to the /run/media/*username*/*label* directory, where *username* is the name of the user logged into the desktop environment and *label* is the filesystem label on the removeable media. For example, if you insert a DVD with a filesystem label of "Fedora-Workstation-Live-x86_64" into your system while logged into a desktop environment as user1, the system will automatically create a /run/media/user1/Fedora-Workstation-Live-x86_64 directory and mount the DVD to it. Similarly, if you insert a USB flash drive with a filesystem label of "KINGSTON" into your system while logged into a desktop environment as user1, the system will automatically create a /run/media/user1/KINGSTON directory and mount the USB flash drive to it. This is shown in the following output:

```
[root@server1 ~]# df -hT
Filesystem    Type      Size  Used Avail Use% Mounted on
devtmpfs      devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs         tmpfs     2.0G     0  2.0G   0% /dev/shm
tmpfs         tmpfs     784M  1.2M  783M   1% /run
/dev/sda3     ext4       35G  8.9G   24G  28% /
tmpfs         tmpfs     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2     ext4      974M  150M  757M  17% /boot
/dev/sda1     vfat      599M   14M  585M   3% /boot/efi
tmpfs         tmpfs     392M   92K  392M   1% /run/user/42
tmpfs         tmpfs     392M   76K  392M   1% /run/user/0
tmpfs         tmpfs     392M   76K  392M   1% /run/user/1000
/dev/sr0      iso9660   3.8G  3.8G    0M 100% /run/media/user1/
Fedora-Workstation-Live-x86_64
/dev/sdb1     exfat      64G  506M 63.5G   1% /run/media/user1/KINGSTON
[root@server1 ~]#_
```

In addition, the system will also display filesystem label shortcuts to the /run/media/user1/Fedora-Workstation-Live-x86_64 and /run/media/user1/KINGSTON directories in the Files application within the desktop environment so that you can easily access the contents of your removeable media, as shown in Figure 5-3.

Figure 5-3 Accessing removeable media within the GNOME desktop



When you are finished accessing the files on the removeable media, you can click the eject icon next to the filesystem label shortcuts shown in Figure 5-3 to unmount the removeable media device, and, in the case of a CD or DVD, eject the physical disk as well.

Removeable media is not limited to USB flash drives, CDs, and DVDs; you can use many other types of removeable media devices on Linux, including SD memory cards, smartphones, external hard disk drives, and external SSDs. Most removable storage device manufacturers emulate the SCSI protocol in the firmware of the device itself, much like a USB flash drive.

When working with removable media devices within a desktop environment, understanding the device names and mount point directories used by the devices themselves is often irrelevant. You can work with the files on removable media devices by accessing the appropriate icons in the Files application that represent the devices in the desktop environment. However, if you are working in a command-line terminal on a Linux server that does not have a desktop environment, you must manually mount and manage removeable storage devices.

Working with Hard Disk Drives and SSDs

Hard disk drives typically come in three flavors: PATA, SATA, and SCSI/SAS. PATA hard disk drives must be set to one of four configurations, each of which has a different device file:

- Primary master (/dev/hda)
- Primary slave (/dev/hdb)
- Secondary master (/dev/hdc)
- Secondary slave (/dev/hdd)

SATA and SCSI/SAS hard disk drives typically have faster data transfer speeds than PATA hard disk drives, and most systems allow for the connection of more than four SATA or SCSI/SAS hard disk drives. As a result of these benefits, both SATA and SCSI/SAS hard disk drives are well suited to Linux servers that require a great deal of storage space for programs and user files. However, SATA and SCSI/SAS hard disk drives have different device files associated with them:

- First SATA/SCSI/SAS hard disk drive (/dev/sda)
- Second SATA/SCSI/SAS hard disk drive (/dev/sdb)
- Third SATA/SCSI/SAS hard disk drive (/dev/sdc)
- And so on

SSDs are a newer technology that use much faster NAND flash storage. PATA, SATA, and SCSI/SAS SSDs provide a hard disk compatible interface so that they can function as a drop-in replacement for hard disk drives. Most SSDs of this type on the market are SATA or SAS; as a result, /dev/sda could refer to the first hard disk drive or the first SSD, /dev/sdb could refer to the second hard disk drive or second SSD, and so on. NVMe SSDs are often faster than SATA and SAS SSDs as they provide an SSD-only architecture that directly connects to the PCIe bus on a computer. As a result, NVMe SSDs use different device files; /dev/nvme0 is the first NVMe SSD, /dev/nvme1 is the second NVMe SSD, and so on. Unlike other SSDs or hard disk drives, NVMe SSDs can be divided into namespaces, and each namespace can then be further subdivided into partitions that contain a filesystem. The device files for NVMe SSDs reflect this; the first partition on the first namespace on the first NVMe SSD is /dev/nvme0n1p1, the second partition on the first namespace on the first NVMe SSD is /dev/nvme0n1p2, and so on.

Note 15

For simplicity, this book will refer primarily to hard disk drives when discussing permanent storage from now on. However, all of the concepts related to hard disk drives apply equally to SSDs.

Standard Hard Disk Drive Partitioning

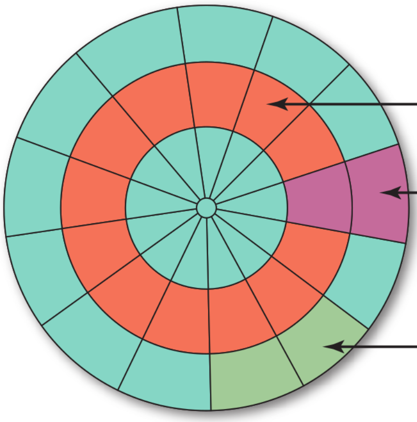
Recall that hard disk drives have the largest storage capacity of any device that you use to store information on a regular basis. As helpful as this storage capacity can be, it also poses some problems; as the size of a disk increases, organization becomes more difficult and the chance of error increases. To solve these problems, Linux administrators divide a hard disk drive into smaller partitions. Each partition can contain a separate filesystem and can be mounted to different mount point directories. Recall from Chapter 2 that Linux requires two partitions at minimum: a partition that is mounted to the / directory (the root partition) and a partition that is mounted to the /boot directory. You can optionally create a swap partition, but the system will automatically create a swap file or compressed zswap RAM disk (/dev/zram0) for virtual memory if you do not.

It is a good practice to use more than just two partitions on a Linux system. This division allows you to do the following:

- Segregate different types of data—for example, home directory data is stored on a separate partition mounted to /home.
- Allow for the use of more than one type of filesystem on one hard disk drive—for example, some filesystems are tuned for database use.
- Reduce the chance that filesystem corruption will render a system unusable; if the partition that is mounted to the /home directory becomes corrupted, it does not affect the system because operating system files are stored on a separate partition mounted to the / directory.
- Speed up access to stored data by keeping filesystems as small as possible.
- Allow for certain operating system features—for example, a /boot/efi partition is required to boot systems that use a UEFI BIOS.

On a physical level, hard disk drives contain circular metal platters that spin at a fast speed. Data is read off these disks in concentric circles called **tracks**; each track is divided into **sectors** of information, and sectors are combined into more usable **blocks** of data, as shown in Figure 5-4. Most hard disk drives contain several platters organized on top of each other such that they can be written to simultaneously to speed up data transfer. A series consisting of the same concentric track on all of the metal platters inside a hard disk drive is known as a **cylinder**.

Figure 5-4 The physical areas of a hard disk drive



Note 16

SSDs use circuitry within the drive itself to map data to logical tracks, sectors, blocks, and cylinders to ensure that the OS can work with SSDs like any hard disk drive. This allows you to create partitions and filesystems on an SSD in the same way that you would a hard disk drive. Because SSDs are much faster at reading and writing data compared to hard disk drives, they are typically used on production Linux servers today.

Partition definitions are stored in the first readable sector of the hard disk drive known as the Master Boot Record (MBR). Large hard disk drives (>2 TB) and newer hard disk drives use a GUID Partition Table (GPT) in place of an MBR to allow for the additional addressing of sectors. If the MBR or GPT area of the hard disk drive becomes corrupted, the entire contents of the hard disk drive might be lost.

Note 17

It is common for Linux servers to have several hard disk drives. In these situations, it is also common to configure one partition on each hard disk drive and mount each partition to different directories on the directory tree. Thus, if one partition fails, an entire hard disk drive can be replaced with a new one and the data retrieved from a back-up source.

Recall from Chapter 2 that hard disk drives with an MBR normally contain up to four primary partitions; to overcome this limitation, you can use an extended partition in place of one of these primary partitions. An extended partition can then contain many more subpartitions called logical drives; each logical drive can be formatted with a filesystem. Partition device files start with the name of the hard disk drive (e.g., /dev/sda) and append a number indicating the partition on that hard disk drive. The first primary partition is given the number 1, the second primary partition is given the number 2, the third primary partition is given the number 3, and the fourth primary partition is given the number 4. If any one of these primary partitions is labeled as an extended partition, the logical drives within are named starting with number 5. Unlike hard disk drives that use an MBR, GPT hard disk drives don't need to adhere to the limitation of four primary partitions. Instead, you can create as many as 128 partitions (e.g., /dev/sda1 to /dev/sda128). Consequently, a hard disk drive that uses a GPT has no need for extended partitions or logical drives. Table 5-5 lists some common hard disk drive partition names.

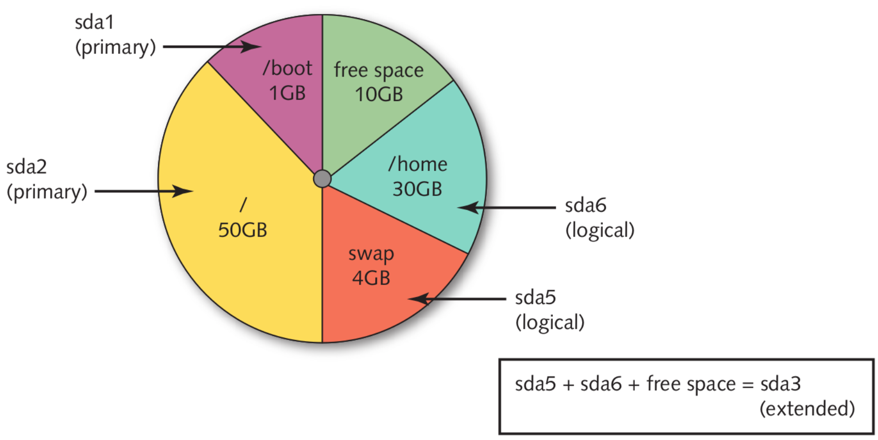
Table 5-5 Common MBR partition device files for /dev/hda and /dev/sda

MBR Partition	GPT Partition	PATA Device Name (assuming /dev/hda)	SATA/SCSI/SAS Device Name (assuming /dev/sda)	NVMe Device Name (assuming /dev/nvme0)
1st primary partition	1st partition	/dev/hda1	/dev/sda1	/dev/nvme0p1
2nd primary partition	2nd partition	/dev/hda2	/dev/sda2	/dev/nvme0p2
3rd primary partition	3rd partition	/dev/hda3	/dev/sda3	/dev/nvme0p3
4th primary partition	4th partition	/dev/hda4	/dev/sda4	/dev/nvme0p4
1st logical drive	5th partition	/dev/hda5	/dev/sda5	/dev/nvme0p5
2nd logical drive	6th partition	/dev/hda6	/dev/sda6	/dev/nvme0p6
3rd logical drive	7th partition	/dev/hda7	/dev/sda7	/dev/nvme0p7
nth logical drive	***n***th partition	/dev/hda***n***	/dev/sda***n***	/dev/nvme0p***n***

Note from Table 5-5 that any one of the MBR primary partitions can be labeled as the extended partition and contain the logical drives. Also, for hard disk drives other than those listed in Table 5-5 (e.g., /dev/sdc), the partition numbers remain the same (e.g., /dev/sdc1, /dev/sdc2, and so on).

An example Linux MBR hard disk drive structure for the first SATA/SCSI/SAS hard disk drive (/dev/sda) can contain a partition for the /boot filesystem (/dev/sda1), a partition for the root filesystem (/dev/sda2), as well as an extended partition (/dev/sda3) that further contains a swap partition (/dev/sda5), a /home filesystem partition (/dev/sda6), and some free space, as shown in Figure 5-5.

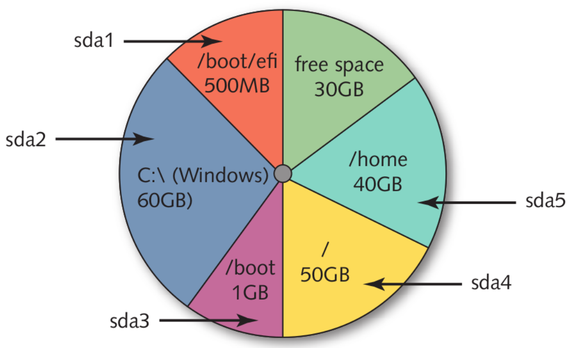
Figure 5-5 A sample MBR partitioning strategy



A more complicated example Linux GPT hard disk drive structure for the first SATA/SCSI/SAS hard disk drive might involve preserving the Windows operating system partition, allowing a user to boot into and use the Linux operating system or boot into and use the Windows operating system. This is known as dual booting and is discussed in Chapter 8. Recall from Chapter 2 that systems that have a UEFI BIOS create a small UEFI System Partition; this partition is formatted with the FAT filesystem and used to store boot-related information for one or more operating systems. If Windows has created this partition, Linux will modify and use it to store the information needed to dual boot both operating systems.

In Figure 5-6, a UEFI System Partition was created as /dev/sda1 because the system had a UEFI BIOS, and the Windows partition was created as /dev/sda2. Linux mounted the UEFI System Partition to the /boot/efi directory and created a separate partition for the /boot filesystem (/dev/sda3), the root filesystem (/dev/sda4), and the /home filesystem (/dev/sda5).

Figure 5-6 A sample dual-boot GPT partitioning strategy



Working with Standard Hard Disk Drive Partitions

Recall that you can create partitions at installation using the graphical installation program. To create partitions after installation, you can use the **`fdisk` command** to create partitions that will be stored in the MBR or GPT on the hard disk drive. To use the `fdisk` command, specify the hard disk drive to partition as an argument. An example of using `fdisk` to work with the first SATA hard disk drive (/dev/sda) is shown in the following output:

```
[root@server1 ~]# fdisk /dev/sda
Welcome to fdisk (util-linux 2.38).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help):
```

Note from the preceding output that the `fdisk` command displays a prompt for the user to accept commands; a list of possible `fdisk` commands can be seen if the user types `m` at this prompt, as shown in the following example:

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

  DOS (MBR)
   a   toggle a bootable flag
   b   edit nested BSD disklabel
   c   toggle the dos compatibility flag
```

```
  Generic
   d   delete a partition
   F   list free unpartitioned space
   l   list known partition types
   n   add a new partition
   p   print the partition table
   t   change a partition type
   v   verify the partition table
   i   print information about a partition

  Misc
   m   print this menu
   u   change display/entry units
   x   extra functionality (experts only)

  Script
   I   load disk layout from sfdisk script file
   O   dump disk layout to sfdisk script file

  Save & Exit
   w   write table to disk and exit
   q   quit without saving changes

  Create a new label
   g   create a new empty GPT partition table
   G   create a new empty SGI (IRIX) partition table
   o   create a new empty DOS partition table
   s   create a new empty Sun partition table
Command (m for help):_
```

To print a list of the partitions currently set on /dev/sda, you could type `p` at the prompt:

```
Command (m for help): p
Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Disk model: Virtual Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: 6DE393B9-5DBA-466F-AF72-1D79894BBFEB

Device       Start       End  Sectors  Size Type
/dev/sda1     2048   1230847  1228800  600M EFI System
/dev/sda2  1230848   3327999  2097152    1G Linux filesystem
/dev/sda3  3328000  76728319 73400320   35G Linux filesystem
/dev/sda4 76728320 104855551 28127232 13.4G Linux filesystem
Command (m for help):_
```

Notice the Disklabel type of gpt, indicating that /dev/sda uses a GPT. For MBR storage devices, the Disklabel type will list msdos. The device names for each partition appear on the left side of the preceding output, including the number of sectors used by each partition (including the start and end sectors), partition size and type (/dev/sda1 is the UEFI System Partition while the remaining partitions contain Linux filesystems). To remove the /dev/sda4 partition and all the data contained on the filesystem within, you could type d at the prompt:

```
Command (m for help): d
Partition number (1-4, default 4): 4
Partition 4 has been deleted.

Command (m for help): p
Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Disk model: Virtual Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: 6DE393B9-5DBA-466F-AF72-1D79894BBFEB

Device        Start       End  Sectors  Size Type
/dev/sda1      2048   1230847  1228800  600M EFI System
/dev/sda2   1230848   3327999  2097152    1G Linux filesystem
/dev/sda3   3328000  76728319 73400320   35G Linux filesystem
Command (m for help):_
```

To create two additional partitions (/dev/sda4 and /dev/sda5), you could type n at the prompt and specify the partition to create, the starting sector on the hard disk drive, and the size in blocks (+5G makes a 5 GB partition):

```
Command (m for help): n
Partition number (4-128, default 4): 4
First sector (76728320-104857566, default 76728320): 76728320
Last sector, +/-sectors or +/-size{K,M,G,T,P} (76728320-104857566,
default 104855551): +5G

Created a new partition 4 of type 'Linux filesystem' and of size 5 GiB.

Command (m for help): n
Partition number (5-128, default 5): 5
First sector (87214080-104857566, default 87214080): 87214080
Last sector, +/-sectors or +/-size{K,M,G,T,P} (87214080-104857566,
default 104855551): 104855551

Created a new partition 5 of type 'Linux filesystem' and of size 8.4 GiB.

Command (m for help): p
Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Disk model: Virtual Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: 6DE393B9-5DBA-466F-AF72-1D79894BBFEB

Device         Start       End  Sectors  Size Type
/dev/sda1       2048   1230847  1228800  600M EFI System
/dev/sda2    1230848   3327999  2097152    1G Linux filesystem
```

```
/dev/sda3   3328000  76728319 73400320   35G Linux filesystem
/dev/sda4  76728320  87214079 10485760    5G Linux filesystem
/dev/sda5  87214080 104855551 17641472  8.4G Linux filesystem
Command (m for help):_
```

Note 18

Instead of entering the first sector for a new partition in the examples above, you can press Enter to accept the default value of the next available sector. Similarly, you can press Enter to accept the default value of the last sector (the last available sector on the hard disk drive) to create a partition that uses the remaining available space.

Note 19

When you create a new partition on an MBR storage device, `fdisk` will first prompt you to choose the partition type (primary, extended, or logical drive).

Notice from the preceding output that the default type for new partitions created with `fdisk` is "Linux filesystem." The partition type describes the use of the partition; while it doesn't restrict partition functionality in any way, you should choose a type that allows others to easily identify its usage. To change a partition type, you can type `t` at the prompt, and then choose the partition number and type code. Typing `L` at the prompt will list all 199 available type codes using the `less` command. For example, to change the /dev/sda4 partition to type 19 (Linux swap) and the /dev/sda5 partition to type 21 (Linux server data), you can type the following at the prompt:

```
Command (m for help): t
Partition number (1-5, default 5): 4
Partition type or alias (type L to list all): L
  1 EFI System                     C12A7328-F81F-11D2-BA4B-00A0C93EC93B
  2 MBR partition scheme           024DEE41-33E7-11D3-9D69-0008C781F39F
  3 Intel Fast Flash               D3BFE2DE-3DAF-11DF-BA40-E3A556D89593
  4 BIOS boot                      21686148-6449-6E6F-744E-656564454649
  5 Sony boot partition            F4019732-066E-4E12-8273-346C5641494F
  6 Lenovo boot partition          BFBFAFE7-A34F-448A-9A5B-6213EB736C22
  7 PowerPC PReP boot              9E1A2D38-C612-4316-AA26-8B49521E5A8B
  8 ONIE boot                      7412F7D5-A156-4B13-81DC-867174929325
  9 ONIE config                    D4E6E2CD-4469-46F3-B5CB-1BFF57AFC149
 10 Microsoft reserved             E3C9E316-0B5C-4DB8-817D-F92DF00215AE
 11 Microsoft basic data           EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
 12 Microsoft LDM metadata         5808C8AA-7E8F-42E0-85D2-E1E90434CFB3
 13 Microsoft LDM data             AF9B60A0-1431-4F62-BC68-3311714A69AD
 14 Windows recovery environment   DE94BBA4-06D1-4D40-A16A-BFD50179D6AC
 15 IBM General Parallel Fs        37AFFC90-EF7D-4E96-91C3-2D7AE055B174
 16 Microsoft Storage Spaces       E75CAF8F-F680-4CEE-AFA3-B001E56EFC2D
 17 HP-UX data                     75894C1E-3AEB-11D3-B7C1-7B03A0000000
 18 HP-UX service                  E2A1E728-32E3-11D6-A682-7B03A0000000
 19 Linux swap                     0657FD6D-A4AB-43C4-84E5-0933C84B4F4F
 20 Linux filesystem               0FC63DAF-8483-4772-8E79-3D69D8477DE4
 21 Linux server data              3B8F8425-20E0-4F3B-907F-1A25A76F98E8
:q
Partition type or alias (type L to list all): 19
Changed type of partition 'Linux filesystem' to 'Linux swap'.
```

```
Command (m for help): t
Partition number (1-5, default 5): 5
Partition type or alias (type L to list all): 21
Changed type of partition 'Linux filesystem' to 'Linux server data'.

Command (m for help): p
Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Disk model: Virtual Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: 6DE393B9-5DBA-466F-AF72-1D79894BBFEB

Device         Start       End   Sectors  Size Type
/dev/sda1       2048   1230847   1228800  600M EFI System
/dev/sda2    1230848   3327999   2097152    1G Linux filesystem
/dev/sda3    3328000  76728319  73400320   35G Linux filesystem
/dev/sda4   76728320  87214079  10485760    5G Linux swap
/dev/sda5   87214080 104855551  17641472  8.4G Linux server data
Command (m for help):_
```

Finally, to save partition changes to the hard disk drive and exit `fdisk`, you can type `w` at the prompt:

```
Command (m for help): w
The partition table has been altered.
Syncing disks. [root@server1 ~]#_
```

If you modify the hard disk drive that also hosts the Linux operating system (as in the preceding example), you can either run the **`partprobe` command** or reboot your system to ensure that your partition changes are seen by the Linux kernel. Following this, you can use the `mkfs`, `mount`, and `umount` commands discussed earlier, specifying the partition device file as an argument. To create an ext4 filesystem on the /dev/sda5 partition created earlier, you can use the following command:

```
[root@server1 ~]# mkfs -t ext4 /dev/sda5
mke2fs 1.46.5 (30-Dec-2023)
Discarding device blocks: done
Creating filesystem with 2205184 4k blocks and 551616 inodes
Filesystem UUID: eab39f1b-8e60-4d2a-9baf-fdfede1fe189
Superblock backups stored on blocks:
       32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
[root@server1 ~]#_
```

To mount this ext4 filesystem to a new mount point directory called /data and view the contents, you can use the following commands:

```
[root@server1 ~]# mkdir /data
[root@server1 ~]# mount /dev/sda5 /data
[root@server1 ~]# df -hT
Filesystem     Type      Size  Used Avail Use% Mounted on
devtmpfs       devtmpfs  4.0M     0  4.0M   0% /dev
```

```
tmpfs           tmpfs     2.0G     0  2.0G   0% /dev/shm
tmpfs           tmpfs     784M  1.2M  783M   1% /run
/dev/sda3       ext4       35G  8.9G   24G  28% /
tmpfs           tmpfs     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2       ext4      974M  150M  757M  17% /boot
/dev/sda1       vfat      599M   14M  585M   3% /boot/efi
tmpfs           tmpfs     392M   92K  392M   1% /run/user/42
tmpfs           tmpfs     392M   76K  392M   1% /run/user/0
tmpfs           tmpfs     392M   76K  392M   1% /run/user/1000
/dev/sda5       ext4      8.2G   24K  7.8G   1% /data
[root@server1 ~]# ls -F /data
lost+found/
[root@server1 ~]#_
```

To allow the system to mount this filesystem automatically at every boot, you can edit the /etc/fstab file such that it has the following entry for /dev/sda5:

```
[root@server1 ~]# cat /etc/fstab
# 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.
#
# After editing this file, run 'systemctl daemon-reload' to update
# systemd units generated from this file.
#
UUID=e8ffcefc-8a11-474e-b879-d7adb1fd1e94 /         ext4 defaults   1 1
UUID=c203af16-e9b2-4577-aab5-5f353f0f2074 /boot     ext4 defaults   1 2
UUID=F4D6-9E57                            /boot/efi vfat umask=077 0 2
/dev/sdb1                                 /USBdrive auto noauto     0 0
/dev/sda5                                 /data     ext4 defaults   0 0
[root@server1 ~]#_
```

Although swap partitions do not contain a filesystem, you must still prepare swap partitions and activate them for use on the Linux system. To do this, you can use the **mkswap command** to prepare the swap partition and the **swapon command** to activate it. To prepare and activate the /dev/sda4 partition created earlier as virtual memory, you can use the following commands:

```
[root@server1 ~]# mkswap /dev/sda4
Setting up swapspace version 1, size = 5 GiB (5368705024 bytes)
no label, UUID=1932ddbf-8c5b-4936-9588-3689a9b25e74
[root@server1 ~]# swapon /dev/sda4
[root@server1 ~]#_
```

Note 20

You can also use the **swapoff command** to deactivate a swap partition.

Next, you can edit the /etc/fstab file to ensure that the new /dev/sda4 partition is activated as virtual memory at boot time, as shown here:

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

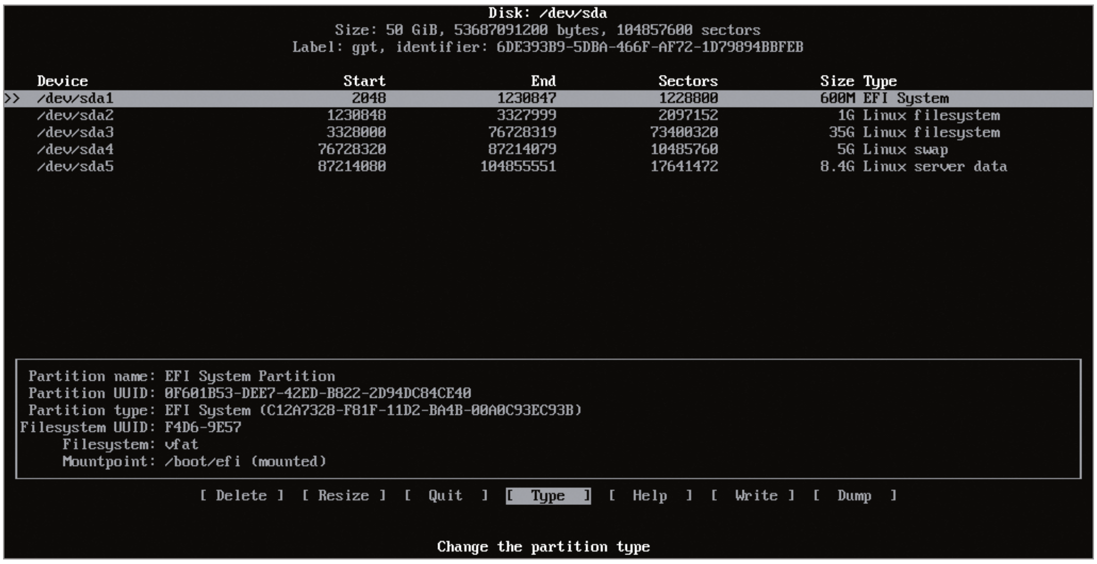
```
#
# After editing this file, run 'systemctl daemon-reload' to update
# systemd units generated from this file.
#
UUID=e8ffcefc-8a11-474e-b879-d7adb1fd1e94 /         ext4 defaults  1 1
UUID=c203af16-e9b2-4577-aab5-5f353f0f2074 /boot     ext4 defaults  1 2
UUID=F4D6-9E57                            /boot/efi vfat umask=077 0 2
/dev/sdb1                                 /USBdrive auto noauto    0 0
/dev/sda5                                 /data     ext4 defaults  0 0
/dev/sda4                                 swap      swap defaults  0 0
[root@server1 ~]#_
```

Note 21

You can create and activate multiple swap partitions, even if your system already uses a swap file or compressed zswap RAM disk (e.g., /dev/zram0) for virtual memory. In this case, the sum total of all swap partitions, swap files, and zswap RAM disks will comprise the virtual memory on the system.

An easier alternative to `fdisk` is the **`cfdisk` command**. If you run the `cfdisk /dev/sda` command following the creation of the partitions you examined earlier, you will see the interactive graphical utility shown in Figure 5-7. You can use this utility to quickly create, manipulate, and delete partitions using choices at the bottom of the screen that you can navigate using your cursor keys.

Figure 5-7 The cfdisk utility



While `fdisk` and `cfdisk` can be used to create or modify both MBR and GPT partitions, there are many other partitioning commands that you can use. For example, the **`gdisk` (GPT fdisk) command** can create and work with GPT partitions using an interface that is nearly identical to `fdisk`. If your hard disk drive has an MBR, `gdisk` will first prompt you to convert the MBR to a GPT, which will destroy all existing MBR partitions on the disk.

Many Linux distributions also contain the **parted (GNU Parted) command**, which can be used to create and modify partitions on both MBR and GPT hard disk drives. Once in the GNU Parted utility, you can type help to obtain a list of valid commands, or type print to print the existing partitions, as shown here:

```
[root@server1 ~]# parted /dev/sdb
GNU Parted 3.4
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Msft Virtual Disk (scsi)
Disk /dev/sdb: 8590MB
Sector size (logical/physical): 512B/4096B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  8590MB  8589MB  primary  xfs
(parted)_
```

Note from the previous output that /dev/sdb uses an MBR (msdos) partition table and contains a single 8589 MB partition (/dev/sdb1) that contains an xfs filesystem type. In the following example, this partition is removed and two additional primary partitions are created: a 5 GB partition with an ext4 filesystem type (/dev/sdb1) and a 3 GB partition with an xfs filesystem type (/dev/sdb1).

```
(parted) rm 1
(parted) mkpart
Partition type?  primary/extended? primary
File system type?  [ext2]? ext4
Start? 0GB
End? 5GB
(parted) mkpart
Partition type?  primary/extended? primary
File system type?  [ext2]? xfs
Start? 5GB
End? 8GB
(parted) print
Model: Msft Virtual Disk (scsi)
Disk /dev/sdb: 8590MB
Sector size (logical/physical): 512B/4096B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  5000MB  4999MB  primary  ext4         lba
 2      5000MB  8000MB  3000MB  primary  xfs          lba

(parted) quit
Information: You may need to update /etc/fstab.
[root@server1 ~]#_
```

As with the partition types shown in other tools, such as `fdisk`, the ext4 and xfs filesystem types specified in the previous example provide descriptive information only. After creating a partition in the `parted` command, you must still format it with a filesystem using the `mkfs` command and mount it to the directory tree using the `mount` command, like you did earlier after creating partitions with `fdisk`. Alternatively, you can prepare the partition for swap using the `mkswap` command and activate it using the `swapon` command. Finally, you can update the /etc/fstab file to mount new filesystems and activate new swap partitions automatically at boot time.

Note 22

As with *`fdisk`*, after making changes to the hard disk drive that hosts the Linux operating system using *`cfdisk`*, *`gdisk`*, or *`parted`*, you should run *`partprobe`* or reboot your system.

Working with the LVM

In the previous section, you learned how to create standard hard disk drive partitions on an MBR or GPT hard disk drive. You also learned how to create filesystems on those partitions and mount the filesystems to a directory within the Linux filesystem hierarchy.

Instead of creating and mounting filesystems that reside on standard partitions, recall from Chapter 2 that you can use the Logical Volume Manager (LVM) to create logical volumes that can be mounted to directories within the Linux filesystem hierarchy. Using volumes to host filesystems is far more flexible than using standard partitions because it allows you to select free space from unused partitions across multiple hard disk drives in your computer. This free space is then pooled together into a single group from which volumes can be created. These volumes can be formatted with a filesystem and mounted to a directory on the Linux filesystem hierarchy. Furthermore, additional hard disk drives can easily be added to the LVM, where existing volumes can be extended to take advantage of the additional storage space.

The LVM consists of several different components:

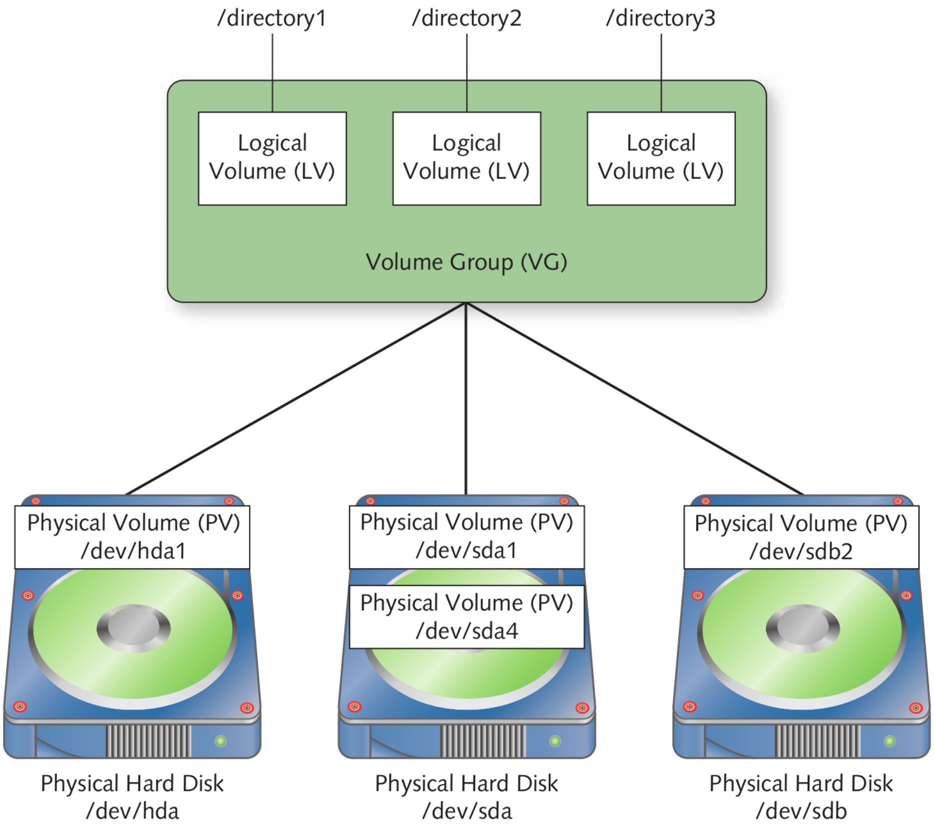
- **Physical Volumes (PVs)** are unused partitions on hard disk drives that the LVM can use to store information.
- **Volume Groups (VGs)** contain one or more PVs. They represent the pools of storage space that are available to the LVM for creating logical volumes. Additional PVs can easily be added to a VG after creation.
- **Logical Volumes (LVs)** are the usable volumes that are created by the LVM from the available storage space within a VG. LVs contain a filesystem and are mounted to a directory in the Linux filesystem hierarchy. In addition, LVs can be resized easily by the LVM to use more or less storage space.

The LVM subsystem in Linux manages the storage of all data that is saved to LVs. The physical location of the data is transparent to the user. Furthermore, the LVM has error correction abilities that minimize the chance that data will become corrupted or lost. Figure 5-8 illustrates the relationships among LVM components in a sample LVM configuration that creates four PVs from the standard partitions on three hard disk drives. These PVs are added to a VG divided into three LVs that are each mounted to a directory on the Linux filesystem hierarchy (/directory1, /directory2, and /directory3).

To configure the LVM, you must first create one or more PVs that reference an unused partition on a hard disk drive in your computer. Say, for example, that you recently created a new partition called /dev/sda4. Rather than placing a filesystem on /dev/sda4, you could instead allow the LVM to use the /dev/sda4 partition using the **`pvcreate` command**, as shown here:

```
[root@server1 ~]# pvcreate /dev/sda4
  Physical volume "/dev/sda4" successfully created
[root@server1 ~]#_
```

Figure 5-8 A sample LVM configuration



Note 23

You should use the `pvcreate` command to create PVs for each unused partition that you want the LVM to use. For simplicity, this section begins with how to create a single PV to demonstrate the configuration of LVM.

The **`pvdisplay` command** can be used to display detailed information about each PV. The following `pvdisplay` command indicates that /dev/sda4 has 13.41 GB of available space:

```
[root@server1 ~]# pvdisplay
  "/dev/sda4" is a new physical volume of "13.41 GiB"
  --- NEW Physical volume ---
  PV Name               /dev/sda4
  VG Name
  PV Size               13.41 GiB
  Allocatable           NO
  PE Size               0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               R6Q0ei-urbT-p6nB-HpXj-CB5u-sLeG-eNVlwm
[root@server1 ~]#_
```

After you have created PVs, you can create a VG that uses the space in the PVs by using the **vgcreate command**. For example, to create a VG called vg00 that uses the /dev/sda4 PV, you could use the following vgcreate command:

```
[root@server1 ~]# vgcreate vg00 /dev/sda4
  Volume group "vg00" successfully created
[root@server1 ~]#_
```

To create a VG that uses multiple PVs, add multiple device arguments to the vgcreate command. For example, the vgcreate vg00 /dev/sda5 /dev/sdb1 /dev/sdc3 command would create a VG called vg00 that uses three PVs (/dev/sda5, /dev/sdb1, and /dev/sdc3).

When creating a VG, it is important to choose the block size for saving data as it cannot be safely changed later. This is called the **physical extent (PE) size** of the VG. A large PE size results in larger write operations and a larger maximum filesystem size for LVs. For example, a PE size of 32 MB will allow for a maximum LV size of 2TB.

By default, the vgcreate command chooses an appropriate PE size according to the current sizes of the PVs that are associated with the VG, but you can use the –s or --physicalextent-size options with the vgcreate command to select a different PE size during VG creation. This is important if you plan on adding a large amount of additional storage to the VG later.

The **vgdisplay command** can be used to display detailed information about each VG. The following vgdisplay command indicates that vg00 has access to 13.41 GB of storage using a PE size of 4 MB:

```
[root@server1 ~]# vgdisplay
  --- Volume group ---
  VG Name               vg00
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               13.41 GiB
  PE Size               4.00 MiB
  Total PE              3433
  Alloc PE / Size       0 / 0
  Free  PE / Size       3433 / 13.41 GiB
  VG UUID               Mfwl0f-nwbN-jW3s-9cNK-TFB6-M2Hw-xAF476
[root@server1 ~]#_
```

Next, you can create LVs from the available space in your VG using the **lvcreate command** and view your results using the **lvdisplay command**. The following commands create an LV called data1 that uses 10 GB of space from vg00 as well as an LV called data2 that uses 3.41 GB of space from vg00, displaying the results afterwards.

```
[root@server1 ~]# lvcreate -L 10GB -n data1 vg00
  Logical volume "data1" created
[root@server1 ~]# lvcreate -L 3.41GB -n data2 vg00
  Logical volume "data2" created
```

```
[root@server1 ~]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/vg00/data1
  LV Name                data1
  VG Name                vg00
  LV UUID                BbZUcM-Rqf7-1ic0-U1Ew-5DKN-CXjt-IskCLR
  LV Write Access        read/write
  LV Creation host, time server1, 2023-09-06 20:14:34 -0400
  LV Status              available
  # open                 0
  LV Size                10.00 GiB
  Current LE             2560
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0

  --- Logical volume ---
  LV Path                /dev/vg00/data2
  LV Name                data2
  VG Name                vg00
  LV UUID                CWjWz2-2qL8-z1Hf-qrzW-WU40-vE3j-PA3Uuf
  LV Write Access        read/write
  LV Creation host, time server1, 2023-09-06 20:14:45 -0400
  LV Status              available
  # open                 0
  LV Size                3.41 GiB
  Current LE             873
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:1
[root@server1 ~]#_
```

Notice from the preceding output that the new VGs can be accessed using the device files /dev/vg00/data1 and /dev/vg00/data2. You can also refer to your new VGs using the device files /dev/mapper/vg00-data1 and /dev/mapper/vg00-data2, which are used by the system when accessing the filesystems on your VGs. These device files are created by the device mapper framework within the Linux kernel, which maps physical block devices such as /dev/sda4 to logical devices within the LVM; as a result, these device files are merely shortcuts to device mapper device files (/dev/dm-*), as shown below:

```
[root@server1 ~]# ll /dev/vg00/data*
total 0
lrwxrwxrwx. 1 root root 7 Sep  6 20:14 data1 -> ../dm-0
lrwxrwxrwx. 1 root root 7 Sep  6 20:14 data2 -> ../dm-1
[root@server1 ~]# ll /dev/mapper/vg00*
lrwxrwxrwx. 1 root root 7 Sep  6 20:14 vg00-data1 -> ../dm-0
lrwxrwxrwx. 1 root root 7 Sep  6 20:14 vg00-data2 -> ../dm-1
[root@server1 ~]#_
```

Note 24

The device mapper files shown in the previous output reflect the identifiers that the Linux kernel uses when working with LVs, as the kernel stores configuration for each LV within the associated /sys/block/dm-*/ directories. However, Linux administrators need only refer to VG and LV names (such as vg00 and data1) when working with LVM commands, as these names are automatically mapped to the appropriate device mapper file.

You can work with these device files as you would normally work with any other hard disk drive partition device file. For example, to create an ext4 filesystem on these devices and mount them to the appropriate directories on the filesystem, you could use the following commands:

```
[root@server1 ~]# mkfs -t ext4 /dev/vg00/data1
mke2fs 1.46.5 (30-Dec-2023)
Discarding device blocks: done
Creating filesystem with 2621440 4k blocks and 655360 inodes
Filesystem UUID: 3e77d984-fdd6-461e-aab3-b2e83cd17f1b
Superblock backups stored on blocks:
     32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
[root@server1 ~]# mkfs -t ext4 /dev/vg00/data2
mke2fs 1.46.5 (30-Dec-2023)
Discarding device blocks: done
Creating filesystem with 893952 4k blocks and 223552 inodes
Filesystem UUID: 5afb6dc6-d283-4413-a2a5-c7d84333096a
Superblock backups stored on blocks:
     32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
[root@server1 ~]# mkdir /data1
[root@server1 ~]# mkdir /data2
[root@server1 ~]# mount /dev/vg00/data1 /data1
[root@server1 ~]# mount /dev/vg00/data2 /data2
[root@server1 ~]# df -hT
Filesystem              Type      Size  Used Avail Use% Mounted on
devtmpfs                devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs                   tmpfs     2.0G   12K  2.0G   1% /dev/shm
tmpfs                   tmpfs     784M  1.2M  783M   1% /run
/dev/sda3               ext4       35G  8.9G   24G  28% /
tmpfs                   tmpfs     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2               ext4      974M  150M  757M  17% /boot
/dev/sda1               vfat      599M   14M  585M   3% /boot/efi
tmpfs                   tmpfs     392M   92K  392M   1% /run/user/42
tmpfs                   tmpfs     392M   76K  392M   1% /run/user/0
tmpfs                   tmpfs     392M   76K  392M   1% /run/user/1000
```

```
/dev/mapper/vg00-data1 ext4      9.8G   24K  9.3G   1% /data1
/dev/mapper/vg00-data2 ext4      3.3G   24K  3.1G   1% /data2
[root@server1 ~]#_
```

Next, you can edit the /etc/fstab file to ensure that your new logical volumes are automatically mounted at system startup, as shown here:

```
[root@server1 ~]# cat /etc/fstab
# 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.
#
# After editing this file, run 'systemctl daemon-reload' to update
# systemd units generated from this file.
#
UUID=e8ffcefc-8a11-474e-b879-d7adb1fd1e94 /         ext4 defaults  1 1
UUID=c203af16-e9b2-4577-aab5-5f353f0f2074 /boot     ext4 defaults  1 2
UUID=F4D6-9E57                            /boot/efi vfat umask=077 0 2
/dev/vg00/data1                           /data1    ext4 defaults  0 0
/dev/vg00/data2                           /data2    ext4 defaults  0 0
[root@server1 ~]#_
```

As files are added over time, you may find that additional capacity is needed in your filesystems. With the LVM, you can easily add additional storage devices and extend the capacity of existing LVs and their associated filesystems. After adding an additional storage device, you create a new PV for the partition using the `pvcreate` command, add the new PV to your existing VG using the **`vgextend` command**, and then extend the size of your LV and filesystem to use the additional space from the VG using the **`lvextend` command**. Say, for example, that you wish to extend the size of the data2 LV and associated ext4 filesystem created in the previous example by 5 GB. To do this, you could add a new hard drive to your computer (e.g., /dev/sdb) and create a /dev/sdb1 partition that spans that entire hard drive using the `fdisk /dev/sdb` command. Next, you could run the following commands to extend the data2 LV and filesystem while it remains mounted and accessible by users:

```
[root@server1 ~]# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created.
[root@server1 ~]# vgextend vg00 /dev/sdb1
  Volume group "vg00" successfully extended
[root@server1 ~]# lvextend -L +5GB -r /dev/vg00/data2
  Size of logical volume vg00/data2 changed from 3.41 GiB (873 extents)
  to 8.41 GiB (2153 extents).

  Logical volume vg00/data2 successfully resized.
  resize2fs 1.46.5 (30-Dec-2023)
  Filesystem at /dev/mapper/vg00-data2 is mounted on /data2
  on-line resizing required; old_desc_blocks = 1, new_desc_blocks = 2
  The filesystem on /dev/mapper/vg00-data2 is now 2204672 (4k) blocks
  long.
[root@server1 ~]# df -hT
Filesystem             Type      Size  Used Avail Use% Mounted on
devtmpfs               devtmpfs  4.0M     0  4.0M   0% /dev
tmpfs                  tmpfs     2.0G   12K  2.0G   1% /dev/shm
tmpfs                  tmpfs     784M  1.2M  783M   1% /run
/dev/sda3              ext4       35G  8.9G   24G  28% /
tmpfs                  tmpfs     2.0G  8.0K  2.0G   1% /tmp
```

```
/dev/sda2                ext4      974M  150M  757M  17% /boot
/dev/sda1                vfat      599M   14M  585M   3% /boot/efi
tmpfs                    tmpfs     392M   92K  392M   1% /run/user/42
tmpfs                    tmpfs     392M   76K  392M   1% /run/user/0
tmpfs                    tmpfs     392M   76K  392M   1% /run/user/1000
/dev/mapper/vg00-data1 ext4        9.8G   24K  9.3G   1% /data1
/dev/mapper/vg00-data2 ext4        8.3G   24K  7.9G   1% /data2
[root@server1 ~]#_
```

Note 25

If you don't use the `-r` option to the `lvextend` command to resize the existing filesystem when extending an LV, your system will not be able to use the additional space. In this case, you can use a command to resize your filesystem after extending your LV. For example, to extend an ext2/ext3/ext4 filesystem, you can use the **`resize2fs` command**, and to extend an XFS filesystem, you can use either the **`xfs_info` command** or **`xfs_growfs` command**.

In addition to those discussed in this section, there are many other useful commands that can list and configure PVs, VGs, and LVs. Table 5-6 summarizes these commands.

Table 5-6 Common LVM Commands

Command	Description

`pvdisplay`
`pvscan`
`pvs`
	Displays PV configuration

`vgdisplay`
`vgscan`
`vgs`
	Displays VG configuration

`lvdisplay`
`lvscan`
`lvs`
	Displays LV configuration

`pvcreate`	Creates a PV
`vgcreate`	Creates a VG that includes one or more PVs
`lvcreate`	Creates a LV from available space within a VG
`pvremove`	Removes a PV
`vgremove`	Removes a VG
`lvremove`	Removes a LV
`vgextend`	Adds additional PVs to a VG
`vgreduce`	Removes PVs from a VG
`lvextend`	Expands the size of a LV using free storage in a VG
`lvreduce`	Reduces the size of a LV, returning freed space to the VG
`lvresize`	Performs the same functions as `lvextend` and `lvreduce`
`pvchange`	Modifies settings for an existing PV
`vgchange`	Modifies settings for an existing VG
`lvchange`	Modifies settings for an existing LV

Monitoring Filesystems

After filesystems are created on devices and those devices are mounted to the directory tree, they should be checked periodically for errors, disk space usage, and inode usage. This minimizes the problems that can occur as a result of a damaged filesystem and reduces the likelihood that a file cannot be saved due to insufficient disk space.

Disk Usage

Several filesystems can be mounted to the directory tree. As mentioned earlier, the more filesystems that are used, the less likely a corrupted filesystem will interfere with normal system operations. Conversely, more filesystems typically result in less space per filesystem and frequent monitoring is necessary to ensure that adequate space is always available to each filesystem. In addition to the /boot and root filesystems, many Linux administrators create /home, /usr, and /var filesystems during installation. The available space in the /home filesystem reduces as users store more data, and the available space in the /usr filesystem reduces as additional programs are added to the system. Moreover, log files and print queues in the /var filesystem grow in size continuously unless they are cleared periodically. The root filesystem, however, is the most vital to monitor; it should always contain a great deal of free space used as working space for the operating system. If free space on the root filesystem falls below 10 percent, the system might suffer from poorer performance or cease to operate.

The easiest method for monitoring free space by mounted filesystems is to use the `df` command discussed earlier alongside the `-h` option at minimum to list human readable size formats:

```
[root@server1 ~]# df -h
Filesystem                Size  Used Avail Use% Mounted on
devtmpfs                  4.0M     0  4.0M   0% /dev
tmpfs                     2.0G   12K  2.0G   1% /dev/shm
tmpfs                     784M  1.2M  783M   1% /run
/dev/sda3                  35G  8.9G   24G  28% /
tmpfs                     2.0G  8.0K  2.0G   1% /tmp
/dev/sda2                 974M  150M  757M  17% /boot
/dev/sda1                 599M   14M  585M   3% /boot/efi
tmpfs                     392M   92K  392M   1% /run/user/42
tmpfs                     392M   76K  392M   1% /run/user/0
tmpfs                     392M   76K  392M   1% /run/user/1000
/dev/mapper/vg00-data1    9.8G   24K  9.3G   1% /data1
/dev/mapper/vg00-data2    8.3G   24K  7.9G   1% /data2
[root@server1 ~]#_
```

From the preceding output, the only filesystems used alongside the root filesystem are the /boot, /boot/efi, /data1, and /data2 filesystems; the /home, /usr, and /var directories are simply directories on the root filesystem, which increases the importance of monitoring the root filesystem. Because the root filesystem is 28 percent used in the preceding output, there is no immediate concern. However, log files and software installed in the future will increase this number and might warrant the purchase of additional storage devices for data to reside on.

Note 26

The `df` command only views mounted filesystems; thus, to get disk free space statistics for a USB flash drive filesystem, you should mount it prior to running the `df` command.

If a filesystem is approaching full capacity, it might be useful to examine which directories on that filesystem are taking up the most disk space. You can then remove or move files from that directory to another filesystem that has sufficient space. To view the size of a directory and its contents, you can use the **du (directory usage) command**. As with the df command, the du command also accepts the –h option to make size formats more human readable. For directories that have a large number of files and subdirectories, you should use either the more or less command to view the output page-by-page, as shown with the following /usr directory:

```
[root@server1 ~]# du -h /usr | more
72K     /usr/lib/sysusers.d
8.0K    /usr/lib/sddm/sddm.conf.d
12K     /usr/lib/sddm
44K     /usr/lib/sysctl.d
4.0K    /usr/lib/kde3/plugins
8.0K    /usr/lib/kde3
244K    /usr/lib/tmpfiles.d
62M     /usr/lib/jvm/java-17-openjdk-17.0.2.0.8-7.fc36.x86_64/lib/server
76K     /usr/lib/jvm/java-17-openjdk-17.0.2.0.8-7.fc36.x86_64/lib/jfr
194M    /usr/lib/jvm/java-17-openjdk-17.0.2.0.8-7.fc36.x86_64/lib
194M    /usr/lib/jvm
4.0K    /usr/lib/games
48K     /usr/lib/kdump
64K     /usr/lib/abrt-java-connector
--More--
```

To view only a summary of the total size of a directory, add the –s switch to the du command, as shown in the following example with the /usr directory:

```
[root@server1 ~]# du –hs /usr
6.9G    /usr
[root@server1 ~]#_
```

Recall that every filesystem has an inode table that contains the inodes for the files and directories on the filesystem; this inode table is made during filesystem creation and is usually proportionate to the size of the filesystem. Each file and directory uses one inode; thus, a filesystem with several small files might use up all of the inodes in the inode table and prevent new files and directories from being created on the filesystem. To view the total number of inodes and free inodes for mounted filesystems on the system, you can add the -i option to the df command, as shown in the following output:

```
[root@server1 ~]# df -i
Filesystem                Inodes  IUsed   IFree IUse% Mounted on
devtmpfs                 1048576    466 1048110    1% /dev
tmpfs                     501506      4  501502    1% /dev/shm
tmpfs                     819200    875  818325    1% /run
/dev/sda3                2293760 238184 2055576   11% /
tmpfs                    1048576     34 1048542    1% /tmp
/dev/sda2                  65536     91   65445    1% /boot
/dev/sda1                      0      0       0     - /boot/efi
tmpfs                     100301     75  100226    1% /run/user/42
tmpfs                     100301     56  100245    1% /run/user/0
tmpfs                     100301     57  100244    1% /run/user/1000
/dev/mapper/vg00-data1    655360     11  655349    1% /data1
/dev/mapper/vg00-data2    542912     11  542901    1% /data2
[root@server1 ~]#_
```

The preceding output shows that the inode table for the root filesystem has 2293760 inodes and only 238184 (or 11%) of them are currently used.

Checking Filesystems for Errors

Filesystems themselves can accumulate errors over time. These errors are often referred to as **filesystem corruption** and are common on most filesystems. Those filesystems that are accessed frequently are more prone to corruption than those that are not. As a result, such filesystems should be checked regularly for errors.

The most common filesystem corruption occurs because a system was not shut down properly using the `shutdown`, `poweroff`, `halt`, or `reboot` commands. Data is stored in memory for a short period of time before it is written to a file on the filesystem. This process of saving data to the filesystem is called **syncing**. If the computer's power is turned off, data in memory might not be synced properly to the filesystem, causing corruption.

Filesystem corruption can also occur if the storage devices are used frequently for time-intensive tasks such as database access. As the usage of any system increases, so does the possibility for operating system errors when writing to storage devices. Along the same lines, physical hard disk drives and SSDs themselves can wear over time with heavy usage. Some parts of a hard disk drive platter may cease to hold a magnetic charge and some of the NAND flash memory cells in an SSD may cease to function properly; these areas are known as **bad blocks**. When the operating system finds a bad block, it puts a reference to the block in the bad blocks table on the filesystem. Any entries in the bad blocks table are not used for any future storage.

To check a filesystem for errors, you can use the **`fsck` (filesystem check) command**, which can check filesystems of many different types. The `fsck` command takes an option specifying the filesystem type and an argument specifying the device to check; if the filesystem type is not specified, the filesystem is automatically detected. The filesystem being checked must be unmounted beforehand for the `fsck` command to work properly, as shown next:

```
[root@server1 ~]# fsck /dev/vg00/data1
fsck from util-linux 2.38
e2fsck 1.46.5 (30-Dec-2023)
/dev/mapper/vg00-data1 is mounted.
e2fsck: Cannot continue, aborting.
[root@server1 ~]# umount /dev/vg00/data1
[root@server1 ~]# fsck /dev/vg00/data1
fsck from util-linux 2.38
e2fsck 1.46.5 (30-Dec-2023)
/dev/mapper/vg00-data1: clean, 11/655360 files, 66753/2621440 blocks
[root@server1 ~]#_
```

Note 27

Because the root filesystem cannot be unmounted, you should only run the `fsck` command on the root filesystem from single-user mode (discussed in Chapter 8) or from a system booted from live installation media (discussed in Chapter 6).

Notice from the preceding output that the `fsck` command does not display lengthy output when checking the filesystem; this is because the `fsck` command only performs a quick check for errors unless the `–f` option is used to perform a full check, as shown in the following example:

```
[root@server1 ~]# fsck -f /dev/vg00/data1
fsck from util-linux 2.38
e2fsck 1.46.5 (30-Dec-2023)
```

```
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/mapper/vg00-data1: 11/655360 files (0.0% non-contiguous), 66753/2621440 blocks
[root@server1 ~]#_
```

Table 5-7 displays a list of common options used with the `fsck` command.

Table 5-7 Common Options to the `fsck` Command

Option	Description
`-f`	Performs a full filesystem check
`-y`	Allows `fsck` to automatically repair any errors (if not run in interactive mode)
`-A`	Checks all filesystems in /etc/fstab that have a 1 or 2 in the sixth field
`-Cf`	Performs a full filesystem check and displays a progress line
`-AR`	Checks all filesystems in /etc/fstab that have a 1 or 2 in the sixth field but skips the root filesystem
`-V`	Displays verbose output

If the `fsck` command finds a corrupted file, it displays a message to the user asking whether to fix the error; to avoid these messages, you may use the `–y` option listed in Table 5-7 to specify that the `fsck` command should automatically repair any corruption. If the `fsck` command finds files it cannot repair, it places them in the lost+found directory on that filesystem and renames the file to the inode number.

To view the contents of the lost+found directory, mount the device and view the contents of the lost+found directory immediately under the mount point. Because it is difficult to identify lost files by their inode number, most users delete the contents of this directory periodically. Recall that the lost+found directory is automatically created when an ext2, ext3, or ext4 filesystem is created.

Just as you can use the `mke2fs` command to make an ext2, ext3, or ext4 filesystem, you can use the `e2fsck` command to check an ext2, ext3, or ext4 filesystem. The `e2fsck` command accepts more options and can check a filesystem more thoroughly than `fsck`. For example, by using the `–c` option to the `e2fsck` command, you can check for bad blocks on the underlying storage device and add them to a bad block table on the filesystem so that they are not used in the future, as shown in the following example:

```
[root@server1 ~]# e2fsck -c /dev/vg00/data1
e2fsck 1.46.5 (30-Dec-2023)
Checking for bad blocks (read-only test): done
/dev/vg00/data1: Updating bad block inode.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information

/dev/vg00/data1: ***** FILE SYSTEM WAS MODIFIED *****
/dev/vg00/data1: 11/655360 files (0.0% non-contiguous), 66753/2621440 blocks
[root@server1 ~]#_
```

Note 28

The `badblocks` command can be used to perform the same function as the `e2fsck` command with the `-c` option.

Note 29

You cannot use the *`fsck`* command to check and repair an XFS filesystem. Instead, you can use the **`xfs_db` command** to examine an XFS filesystem for corruption, the **`xfs_repair` command** to check and repair an XFS filesystem, as well as the **`xfs_fsr` command** to optimize an XFS filesystem and minimize the chance of future corruption.

Recall from earlier in this chapter that the `fsck` command is run at boot time when filesystems are mounted from entries in the /etc/fstab file. Any entries in /etc/fstab that have a 1 in the sixth field are checked first, followed by entries that have a 2 in the sixth field. However, on many Linux systems, a full filesystem check is forced periodically each time an ext2, ext3, or ext4 filesystem is mounted. This might delay booting for several minutes, depending on the size of the filesystems being checked. To change this interval to a longer interval, such as 20 days, you can use the `-i` option to the **`tune2fs` command**, as shown next:

```
[root@server1 ~]# tune2fs -i 20d /dev/vg00/data1
tune2fs 1.46.5 (30-Dec-2023)
Setting interval between checks to 1728000 seconds
[root@server1 ~]#_
```

The `tune2fs` command can be used to change or "tune" filesystem parameters after a filesystem has been created. Changing the interval between automatic filesystem checks to 0 disables filesystem checks altogether.

Disk Quotas

If there are several users on a Linux system, the system must have enough free space to support the files that each user expects to store on each filesystem. To prevent users from using unnecessary space, you can impose limits on filesystem usage. These restrictions, called **disk quotas**, can be applied to users or groups of users. Furthermore, **quotas** can restrict how many files and directories a user can create (i.e., restrict the number of inodes created) on a particular filesystem or the total size of all files that a user can own on a filesystem. Two types of quota limits are available: soft limits and hard limits. **Soft limits** are disk quotas that the user can exceed for a certain period of time with warnings (seven days by default), whereas **hard limits** are rigid quotas that the user cannot exceed. Quotas are typically enabled at boot time if there are quota entries in /etc/fstab, but they can also be turned on and off afterward by using the **`quotaon` command** and **`quotaoff` command**, respectively.

To set up quotas for the /data1 filesystem and restrict the user user1, you can perform the following steps:

1. Edit the /etc/fstab file to add the usrquota and grpquota mount options for the /data1 filesystem. The resulting line in /etc/fstab file should look like the following:

```
[root@server1 ~]# grep data1 /etc/fstab
/dev/vg00/data1     /data1   ext4    defaults,usrquota,grpquota   0 0
[root@server1 ~]#_
```

Note 30

You can also use journaled quotas on modern Linux kernels, which protects quota data during an unexpected shutdown. To use journaled quotas, replace the mount options of `defaults,usrquota,grpquota` in Step 1 with: `defaults,usrjquota=aquota.user,grpjquota=aquota.group,jqfmt=vfsv0`.

2. Remount the /data1 filesystem as read-write to update the system with the new options from /etc/fstab, as follows:

```
[root@server1 ~]# mount /data1 -o remount,rw
[root@server1 ~]#_
```

3. Run the `quotacheck -mavugf -F vfsv0` command, which looks on the system for file ownership and creates the quota database (`-f`) using the default quota format (`-F vfsv0`) for all filesystems with quota options listed in /etc/fstab (`-a`), giving verbose output (`-v`) for all users and groups (`-u` and `-g`) even if the filesystem is used by other processes (`-m`). Normally, this creates and places information in the /data/aquota.user and /data/aquota.group files. However, if your Linux kernel has been compiled using ext4 quota support, the quota information will be stored in a hidden inode on the ext4 filesystem itself. Sample output from the `quotacheck` command is shown here:

```
[root@server1 ~]# quotacheck -mavugf -F vfsv0
quotacheck: Scanning /dev/mapper/vg00-data1 [/data1] done
quotacheck: Checked 3 directories and 2 files
[root@server1 ~]#_
```

Note 31

If you receive any warnings at the beginning of the `quotacheck` output at this stage, you can safely ignore them because they are the result of newly created aquota.user and aquota.group files that have not been used yet, or the ext4 quota support feature of your Linux kernel.

4. Turn user and group quotas on for all filesystems that have quotas configured using the `quotaon -avug` command:

```
[root@server1 ~]# quotaon -avug
/dev/mapper/vg00-data1 [/data1]: group quotas turned on
/dev/mapper/vg00-data1 [/data1]: user quotas turned on
[root@server1 ~]#_
```

Note 32

You can also enable and disable quotas for individual filesystems. For example, you can use the `quotaon /data1` command to enable quotas for the /data1 filesystem and the `quotaoff /data1` command to disable them.

5. Edit the quotas for certain users by using the **`edquota` command** as follows: `edquota -u` *`username`*. This brings up the nano editor and allows you to set soft and hard quotas for the number of blocks a user can own on the filesystem (typically, 1 block = 1 kilobyte) and the total number of inodes (files and directories) that a user can own on the filesystem. A soft limit and hard limit of zero (0) indicates that there is no limit. To set a hard limit

of 20 MB (=20480KB) and 1000 inodes, as well as a soft limit of 18 MB (=18432KB) and 900 inodes, you can run the `edquota -u user1` command to open the quota table for user1 in the nano editor:

```
Disk quotas for user user1 (uid 1000):
Filesystem              blocks    soft    hard    inodes      soft     hard
/dev/mapper/vg001-data1   1188       0       0       326         0        0
```

Next, you can place the appropriate values in the columns provided and then save and quit the nano editor:

```
Disk quotas for user user1 (uid 1000):
Filesystem             blocks    soft    hard    inodes      soft     hard
/dev/mapper/vg00-data1   1188   18432   20480       326       900     1000
```

6. Edit the time limit for which users can go beyond soft quotas by using the `edquota -u -t` command, which opens the nano editor for you to change the default of seven days, as shown here:

```
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
Filesystem              Block grace period     Inode grace period
/dev/mapper/vg00-data1        7days                   7days
```

7. Ensure that quotas were updated properly by gathering a report for quotas by user on the /data1 filesystem using the **`repquota` command**, as shown in the following output:

```
[root@server1 ~]# repquota /data1
*** Report for user quotas on device /dev/mapper/vg00-data1
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft     hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --     573       0        0             20     0     0
user1     --    1188   18432    20480            326   900  1000
[root@server1 ~]#_
```

8. The aforementioned commands are only available to the root user; however, regular users can view their own quota using the **`quota` command**. The root user can use the `quota` command but can also use it to view quotas of other users:

```
[root@server1 ~]# quota
Disk quotas for user root (uid 0): none
[root@server1 ~]# quota -u user1
Disk quotas for user user1 (uid 500):
Filesystem        blocks   quota   limit   grace   files   quota   limit grace
/dev/mapper/vg00-data1
                    1188   18432   20480             326     900    1000
[root@server1 ~]#_
```

Note 33

To configure and manage quotas for an XFS filesystem, you must use the **`xfs_quota` command**.

Summary

- Disk devices are represented by device files that reside in the /dev directory. These device files specify the type of data transfer, the major number of the device driver in the Linux kernel, and the minor number of the specific device.
- Each disk device must contain a filesystem, which is then mounted to the Linux directory tree for usage with the mount command. The filesystem can later be unmounted using the umount command. The directory used to mount the device must not be in use by any logged-in users for mounting and unmounting to take place.
- Storage devices, such as hard disk drives and SSDs, must be partitioned into distinct sections before filesystems are created on those partitions. To partition a storage device, you can use a wide variety of tools including fdisk, cfdisk, gdisk, and parted.
- Many filesystems are available to Linux; each filesystem is specialized for a certain purpose, and several filesystems can be mounted to different mount points on the directory tree. You can create a filesystem on a device using the mkfs command and its variants.
- The LVM can be used to create logical volumes from the free space within multiple partitions on the various storage devices within your system. Like partitions, logical volumes can contain a filesystem and be mounted to the Linux directory tree. They allow for the easy expansion and reconfiguration of storage.
- Most removeable media devices are recognized as SCSI disks by the Linux system and are automounted by desktop environments.
- It is important to monitor disk usage using the df and du commands to avoid running out of storage space and inodes. Similarly, it is important to check disks for errors using the fsck command and its variants.
- You can use disk quotas to limit the space that each user has on filesystems.

Key Terms

/dev directory
/etc/fstab
/etc/mtab
/proc/devices
bad blocks
blkid command
block
block devices
cfdisk command
character devices
cylinder
device file
df (disk free space) command
disk quotas
du (directory usage) command
e2label command
edquota command
eject command
exfatlabel command
fatlabel command
fdisk command
filesystem corruption
formatting
fsck (filesystem check) command
fuser command
gdisk (GPT fdisk) command
hard limit
Logical Volume (LV)
lsblk command
lsusb command
lvcreate command
lvdisplay command
lvextend command
major number
minor number
mkfs (make filesystem) command
mkisofs command
mknod command
mkswap command
mount command
mount point
mounting
parted (GNU Parted) command
partprobe command
physical extent (PE) size
Physical Volume (PV)
pseudo filesystem
pvcreate command
pvdisplay command
quota command
quotaoff command
quotaon command
quotas
repquota command
resize2fs command
root filesystem
sector
soft limit
swapoff command
swapon command
syncing
track
tune2fs command
udev daemon
udevadm command
umount command
Universally Unique Identifier (UUID)
vgcreate command
vgdisplay command
vgextend command
virtual filesystem
Volume Group (VG)
xfs_admin command
xfs_db command
xfs_fsr command
xfs_growfs command
xfs_info command
xfs_quota command
xfs_repair command