NDG Introduction to Linux I - Chapter 5: File Manipulation

Objectives

This chapter will cover the following exam objectives:

103.3: Perform basic file management v2

Weight: 4

Candidates should be able to use the basic Linux commands to manage files and directories.

Key Knowledge Areas:

> Copy, move and remove files and directories individually.
> Section 5.1 | Section 5.5 | Section 5.6 | Section 5.7 | Section 5.8 | Section 5.9
>
> Copy multiple files and directories recursively.
> Section 5.5
>
> Remove files and directories recursively.
> Section 5.7 | Section 5.8 | Section 5.9

Key Terms

cp
Command used to copy files and directories. cp will copy a SOURCE to a DEST, or multiple SOURCES to a DIRECTORY.
Section 5.5

file
Command used to determine the type of file. file tests each argument in an attempt to classify it. There are three sets of tests, preformed in this order: filesystem test, magic tests, and language tests.
Section 5.3

ls
Command that will list information about files. The current directory is listed by default.
Section 5.2

mkdir
Command used to create directories, if they do not already exist.
Section 5.8

rm
Command used to remove files or directories. By default the rm command will not remove directories.
Section 5.7

rmdir
Command that is used to remove empty directories in the filesystem.
Section 5.9

touch
Command used to change the file timestamps. touch will allow a user to update the access and modification times of each FILE to the current time.
Section 5.4

Content

5.1 Introduction

Everything is considered a file in Linux. Therefore, file management is an important topic. Not only are your documents considered files, but hardware devices such as hard drives and memory are considered files as well. Even directories are considered files since they are special files that are used to contain other files.

The essential commands needed for file management are often named by short abbreviations of their functions. You can use the `ls` command to list files, the `cp` command to copy files, the `mv` command to move or rename files, and the `rm` command to remove files.

For working with directories, use the `mkdir` command to make directories, the `rmdir` command to remove directories (if they are empty), and the `rm -r` command to recursively delete directories containing files.

Note:

While permissions will be covered in greater detail later in the course, they can have an impact on file management commands.

Permissions determine the level of access that a user has on a file. When a user runs a program and the program accesses a file, then the permissions are checked to determine if the user has the correct access rights to the file.

There are three types of permissions: read, write, and execute. Each has a different meaning depending on whether they are applied to a file or a directory.

Read:

- On a file, this allows processes to read the contents of the file, meaning the contents can be viewed and copied.
- On a directory, file names in the directory can be listed, but other details are not available.

Write:

- A file can be written to by the process, so changes to a file can be saved. Note that the write permission usually requires the read permission on the file to work correctly with most applications.
- On a directory, files can be added to or removed from the directory. Note that the write permission requires the execute permission on the directory to work correctly.

Execute:

- A file can be executed or run as a process.
- On a directory, the user can use the `cd` command to "get into" the directory and use the directory in a pathname to access files and, potentially, subdirectories under this directory.

Typically, there are two places where you should always have the write and execute permissions on the directory: your home directory (for permanent files) and the /tmp directory (for temporary files).

Permissions will be covered in greater detail later in the course.

5.2 Listing Files

While technically not a file management command, the ability to list files using the ls command is critical to file management. By default, the ls command will list the files in the current directory:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

For arguments, the ls command will accept an arbitrary number of different path names to attempt to list:

```
ls [OPTION]... [FILE]...
```

Note the use of the period . character in the following example. When used where a directory is expected, it represents the current directory.

```
sysadmin@localhost:~$ ls . Documents  /usr/local
.:
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

/usr/local:
bin  etc  games  include  lib  man  sbin  share  src

Documents:
School             alpha-third.txt  hidden.txt   numbers.txt  spelling.tx
Work               alpha.txt        letters.txt   os.csv       words
adjectives.txt     animals.txt      linux.txt     people.csv
alpha-first.txt    food.txt         longfile.txt  profile.txt
alpha-second.txt   hello.sh         newhome.txt   red.txt
```

However, when the period . character is used at the beginning of a file or directory name, it makes the file hidden by default. These hidden files are not normally displayed when using the ls command, though many can be commonly found in user home directories. Hidden files and directories are typically used for individual specific data or settings. Using the -a option will display all files, including hidden files:

```
sysadmin@localhost:~$ ls -a
.              .bashrc    .selected_editor  Downloads  Public
..             .cache     Desktop           Music      Templates
.bash_logout   .profile   Documents         Pictures   Videos
```

The -A option to the ls command will display almost all files of a directory. The current directory file . and the parent directory file .. are omitted.

```
sysadmin@localhost:~$ ls -A
.bash_logout  .profile          Documents  Pictures   Videos
.bashrc       .selected_editor  Downloads  Public
.cache         Desktop          Music      Templates
```

The -d option to the ls command is also important. When listing a directory, the ls command normally will show the contents of that directory, but when the -d option is added, then it will display the directory itself:

```
sysadmin@localhost:~$ ls -ld /var/log
drwxrwxr-x 1 root syslog 4096 Mar  8 02:11 /var/log
```

Listing the contents of a directory using the ls command only requires the read permission on the directory.

Command	File Type	Permission
ls	Directory	Read

Long Listing

To learn the details about a file, such as the type of file, permissions, ownership, or the timestamp, perform a long listing, using the -l option to the ls command. Because it provides a variety of output, the /var/log directory is listed as an example below:

```
sysadmin@localhost:~$ ls -l /var/log/
total 900
-rw-r--r-- 1 root   root   15322 Feb 22 16:32 alternatives.log
drwxr-xr-x 1 root   root    4096 Jul 19  2018 apt
-rw-r----- 1 syslog adm      560 Mar  8 02:17 auth.log
-rw-r--r-- 1 root   root   35330 May 26  2018 bootstrap.log
-rw-rw---- 1 root   utmp       0 May 26  2018 btmp
-rw-r----- 1 syslog adm      293 Mar  8 02:17 cron.log
-rw-r--r-- 1 root   adm    85083 Feb 22 16:32 dmesg
-rw-r--r-- 1 root   root 351960 Jul 19  2018 dpkg.log
-rw-r--r-- 1 root   root   32064 Feb 22 16:32 faillog
drwxr-xr-x 2 root   root    4096 Jul 19  2018 journal
-rw-rw-r-- 1 root   utmp 292584 Mar  8 02:11 lastlog
-rw-r----- 1 syslog adm    14289 Mar  8 02:17 syslog
-rw------- 1 root   root   64128 Feb 22 16:32 tallylog
-rw-rw-r-- 1 root   utmp.    384 Mar  8 02:11 wtmp
```

Viewing the above output as fields that are separated by spaces, they indicate:

File Type

```
-rw-r--r-- 1 root root  15322 Feb 22 16:32 alternatives.log
drwxr-xr-x 1 root root   4096 Jul 19  2018 apt
```

The first character of each line indicates the type of file. The file types are:

Symbol	File Type	Description
d	directory	A file used to store other files.
-	regular file	Includes readable files, image files, binary files, and compressed files.
l	symbolic link	Points to another file.
s	socket	Allows for communication between processes.
p	pipe	Allows for communication between processes.
b	block file	Used to communicate with hardware.
c	character file	Used to communicate with hardware.

Note that alternatives.log is a regular file -, while the apt is a directory d.

Permissions

```
drwxr-xr-x 1 root root  4096 Jul 19 2018 journal
```

The next nine characters demonstrate the permissions of the file. Permissions indicate how certain users can access a file.

The permissions are read r, write w, and execute x. Breaking this down a bit:

User Owner			Group Owner			Other		
Read	Write	Execute	Read	Write	Execute	Read	Write	Execute
r	w	x	r	w	x	r	w	x

Permissions will be covered in greater detail later in the course.

Hard Link Count

```
-rw-r----- 1 syslog adm 560 Mar 8 02:17 auth.log
```

There is only one directory name that links to this file.

Links will be covered in greater detail later in the course.

User Owner

```
-rw-r----- 1 syslog adm 293 Mar 8 02:17 cron.log
```

The syslog user owns this file. Every time a file is created, the ownership is automatically assigned to the user who created it. This is important because the owner has the rights to set permissions on a file.

File ownership will be covered in greater detail later in the course.

Group Owner

```
-rw-rw-r-- 1 root utmp 292584 Mar 8 02:11 lastlog
```

This indicates which group owns this file. This is important because any member of this group has a set of permissions on the file. Although root created this file, any member of the utmp group has read and write access to it (as indicated by the group permissions).

File ownership will be covered in greater detail later in the course.

File Size

```
-rw-r----- 1 syslog adm 14289 Mar 8 2018 syslog
```

This displays the size of the file in bytes.

Timestamp

```
-rw-rw---- 1 root   utmp      0 May 26  2018 btmp
-rw-r--r-- 1 root   adm   85083 Feb 22 16:32 dmesg
```

This indicates the time that the file's contents were last modified. This time would be listed as just the date and year if the file was last modified more than six months from the current date. Otherwise, the month, day, and time is displayed. Using the `ls` command with the `--full-time` option will display timestamps in full detail.

File Name

```
-rw-r--r-- 1 root root 35330 May 26 2018 bootstrap.log
```

The final field contains the name of the file or directory. In the case of symbolic links, the link name will be shown along with an arrow and the path name of the file that is linked is shown.

```
lrwxrwxrwx. 1 root root 22 Nov 6 2012 /etc/grub.conf -> ../boot/grub/grub.conf
```

Links will be covered in greater detail later in the course.

While listing the contents of a directory using the `ls` command only requires the read permission on the directory, viewing file details with the `-l` option also requires the execute permission on the directory.

Command	**File Type**	**Permission**
`ls -l`	Directory	Read, Execute

Sorting

The output of the `ls` command is sorted alphabetically by file name. It can sort by other methods as well:

Option	**Function**
`-S`	Sorts files by size

Option	Function
-t	Sorts by timestamp
-r	Reverses any type of sort

With both time and size sorts, add the -l option or the --full-time option, to be able to view those details:

```
sysadmin@localhost:~$ ls -t --full-time
total 32
drwxr-xr-x 2 sysadmin sysadmin 4096 2019-02-22 16:32:34.000000000 +0000 Desktop
drwxr-xr-x 4 sysadmin sysadmin 4096 2019-02-22 16:32:34.000000000 +0000 Documents
drwxr-xr-x 2 sysadmin sysadmin 4096 2019-02-22 16:32:34.000000000 +0000 Downloads
drwxr-xr-x 2 sysadmin sysadmin 4096 2019-02-22 16:32:34.000000000 +0000 Music
drwxr-xr-x 2 sysadmin sysadmin 4096 2019-02-22 16:32:34.000000000 +0000 Pictures
drwxr-xr-x 2 sysadmin sysadmin 4096 2019-02-22 16:32:34.000000000 +0000 Public
drwxr-xr-x 2 sysadmin sysadmin 4096 2019-02-22 16:32:34.000000000 +0000 Templates
drwxr-xr-x 2 sysadmin sysadmin 4096 2019-02-22 16:32:34.000000000 +0000 Videos
```

Recursion

When managing files, it is important to understand what the term recursive option means. When the recursive option is used with file management commands, it means to apply that command to not only the specified directory, but also to all subdirectories and all of the files within all subdirectories. To perform a recursive listing with the ls command, add the -R option.

```
sysadmin@localhost:~$ ls -lR /var/log
/var/log:
total 900

-rw-r--r-- 1 root   root  15322 Feb 22 16:32 alternatives.log
drwxr-xr-x 1 root   root   4096 Jul 19  2018 apt
-rw-r----- 1 syslog adm     560 Mar  8 02:17 auth.log
-rw-r--r-- 1 root   root  35330 May 26  2018 bootstrap.log
-rw-rw---- 1 root   utmp      0 May 26  2018 btmp
-rw-r----- 1 syslog adm     293 Mar  8 02:17 cron.log
-rw-r--r-- 1 root   adm   85083 Feb 22 16:32 dmesg
```

```
-rw-r--r-- 1 root   root 351960 Jul 19  2018 dpkg.log
-rw-r--r-- 1 root   root  32064 Feb 22 16:32 faillog
drwxr-xr-x 2 root   root   4096 Jul 19  2018 journal
-rw-rw-r-- 1 root   utmp 292584 Mar  8 02:11 lastlog
-rw-r----- 1 syslog adm   14289 Mar  8 02:17 syslog
-rw------- 1 root   root  64128 Feb 22 16:32 tallylog
-rw-rw-r-- 1 root   utmp.   384 Mar  8 02:11 wtmp

/var/log/apt:
total 144
-rw-r--r-- 1 root root  13232 Jul 19  2018 eipp.log.xz
-rw-r--r-- 1 root root  18509 Jul 19  2018 history.log
-rw-r----- 1 root adm  108711 Jul 19  2018 term.log

/var/log/journal:
total 0
```

Warning

Use the recursive option carefully because its impact can be huge! With the ls command, using the -R option can result in a large amount of output in your terminal. For the other file management commands, the recursive option can impact a large number of files and a large amount of disk space.

Consider This

Some commands use the -r option for recursion while others use the -R option.
The -R option is used recursively with the ls command because the -r option is used for reversing how the files are sorted.

5.3 Viewing File Types

When viewing a binary file in a CLI, the terminal may become corrupted and unable to display the output of other subsequent commands correctly. To avoid viewing binary files, use the file command. The file command "looks at" the contents of a file to report what kind of file it is; it does this by matching the content to known types stored in a magic file.

Many commands that you will use in Linux expect that the file name provided as an argument is a text file (rather than some sort of binary file). As a result, it is a good idea to use the `file` command before attempting to access a file to make sure that it does contain text. For example, the Documents/newhome.txt file is in ASCII English text format and can be viewed without any problem:

```
sysadmin@localhost:~$ file Documents/newhome.txt
Documents/newhome.txt: ASCII text
```

In the next example, the file /bin/ls is an Executable Link Format (ELF); this file shouldn't be displayed with the `cat` command. While the `cat` command is able to output the contents of binary files, the terminal application may not handle displaying the content of binary files correctly.

```
sysadmin@localhost:~$ file /bin/ls
/bin/ls: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically lin
ked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]
=9567f9a28e66f4d7ec4baf31cfbf68d0410f0ae6, stripped
```

Note

The `cat` command is used to display the content of text files.

This command will be covered in greater detail later in the course.

Important

If you use a command that is expecting a text file argument, you may get strange results if you provide a non-text file. Your terminal may become disrupted by attempting to output binary content; as a result, the terminal may not be able to function properly anymore.

The result will be strange characters being displayed in the terminal. If this occurs, use the `exit` or `logout` commands to leave the terminal you were using and then reconnect or log in again. It may also be possible to execute the `reset` command. If this does not solve the problem, closing the terminal and opening a new one will solve the problem.

5.4 Creating and Modifying Files

The `touch` command performs two functions. It can create an empty file or update the modification timestamp on an existing file.

```
touch [OPTION]... FILE...
```

If the argument provided is an existing file, then the touch command will update the file's timestamp:

```
sysadmin@localhost:~$ ls -l Documents/red.txt
-rw-r--r-- 1 sysadmin sysadmin 51 Feb 22 16:32 Documents/red.txt
sysadmin@localhost:~$ touch Documents/red.txt
sysadmin@localhost:~$ ls -l Documents/red.txt
-rw-r--r-- 1 sysadmin sysadmin 51 Mar  8 02:59 Documents/red.txt
```

If the argument is a file that does not exist, a new file is created with the current time for the timestamp. The contents of this new file will be empty:

```
sysadmin@localhost:~$ touch newfile
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  newfile
Documents  Music      Public    Videos
sysadmin@localhost:~$ ls -l newfile
-rw-rw-r-- 1 sysadmin sysadmin 0 Mar  8 03:00 newfile
```

Each file has three timestamps:

- The last time the file's contents were modified. This is the timestamp provided by the ls -l command by default. The touch command modified this timestamp by default.
- The last time the file was accessed. To modify this timestamp, use the -a option with the touch command.
- The last time the file attributes were modified. These file attributes, which include permissions and file ownership, are also called the file's metadata. To modify this timestamp, use the -c option with the touch command.

The touch command will normally update the specified time to the current time, but the -t option with a timestamp value can be used instead.

```
sysadmin@localhost:~$ touch -t 201912251200 newfile
sysadmin@localhost:~$ ls -l newfile
-rw-rw-r-- 1 sysadmin sysadmin 0 Dec 25  2019 newfile
```

In the above command, the date/time syntax for the touch command is CCYYMMDDHHMM (optionally add SS for the seconds). The table below breaks down the date/time structure used in the example above in greater detail:

Syntax	Example	Meaning
CC	20	The century
YY	19	The year
MM	12	The month
DD	25	The day
HH	12	The hour
MM	00	The minutes

Because the example above uses the precise time of 12:00 AM, we don't need to put 00 in the seconds (SS) field.

In order to view all three timestamps that are kept for a file, use the stat command with the path to the file as an argument:

```
sysadmin@localhost:~$ stat Documents/alpha.txt
  File: Documents/alpha.txt
  Size: 390             Blocks: 8          IO Block: 4096   regular file
Device: 802h/2050d      Inode: 5898795     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1001/sysadmin)   Gid: ( 1001/sysadmin)
Access: 2019-02-22 16:32:34.000000000 +0000
Modify: 2019-02-22 16:32:34.000000000 +0000
Change: 2019-02-22 16:37:01.149507126 +0000
 Birth: -
```

To create a file with the touch command, you must have the write and execute permission on the parent directory. If the file already exists, modifying it with the touch command only requires the write permission on the file itself.

Command	File Type	Permission
touch NEW_FILE	Parent Directory	Write, Execute
touch EXISTING_FILE	File	Write

5.5 Copying Files

Creating copies of files can be useful for numerous reasons:

- If a copy of a file is created before changes are made, then it is possible to revert back to the original.
- It can be used to transfer a file to removable media devices.
- A copy of an existing document can be made; in this way, it is possible to take advantage of the existing layout and content to get started more quickly than from scratch.

The cp command is used to copy files. It takes at least two arguments: the first argument is the path to the file to be copied and the second argument is the path to where the copy will be placed. The files to be copied are sometimes referred to as the source, and the location where the copies are placed is called the destination.

```
cp [OPTION]... SOURCE DESTINATION
```

Recall that the tilde ~ character represents your home directory; it is commonly used as a source or destination. To view the current contents of the home directory, use the ls command:

```
sysadmin@localhost:~$ ls
Desktop Downloads  Pictures  Templates  newfile
Documents  Music        Public  Videos
```

To demonstrate how the cp command works, the /etc/services file can be copied to the home directory using the following command:

```
sysadmin@localhost:~$ cp /etc/services ~
```

The result of executing the previous command would create a copy of the contents of the /etc/services file in your home directory. The new file in your home directory would also be named services. The success of the cp command can be verified using the ls command:

```
sysadmin@localhost:~$ ls
Desktop Downloads  Pictures  Templates  newfile
Documents  Music        Public  Videos  services
```

To copy a file and rename the file in one step you can execute a command like:

```
sysadmin@localhost:~$ cp /etc/services ~/ports
```

The previous command would copy the contents of the /etc/services file into a new file named ports in your home directory:

```
sysadmin@localhost:~$ ls

Desktop Downloads  Pictures  Templates  newfile  services

Documents  Music         Public  Videos  ports
```

To copy multiple files into a directory, additional file names to copy can be included as long as the last argument is a destination directory:

```
cp [OPTION]... SOURCE... DIRECTORY
```

Although they may only represent one argument, a wildcard pattern can be expanded to match multiple path names. The following command copies all files in the Documents directory that start with an n to the home directory; as a result, newhome.txt and numbers.txt are copied.

```
sysadmin@localhost:~$ cp Documents/n* ~

sysadmin@localhost:~$ ls

Desktop Downloads  Pictures  Templates  newfile        numbers.txt  services

Documents  Music         Public  Videos  newhome.txt  ports
```

An issue that frequently comes up when using wildcard patterns with the cp command is that sometimes they match the names of directories as well as regular files. If an attempt is made to copy a directory, the cp command will print an error message.

Using the -v option makes the cp command print verbose output, so you can always tell what copies succeed as well as those that fail. In the following wildcard example, we've marked in red the lines which indicate that the directories were matched, but not copied.

```
sysadmin@localhost:~$ cp -v /etc/b* ~

'/etc/bash.bashrc' -> '/home/sysadmin/bash.bashrc'

cp: -r not specified; omitting directory '/etc/bind'

'/etc/bindresvport.blacklist' -> '/home/sysadmin/bindresvport.blacklist'

cp: -r not specified; omitting directory '/etc/binfmt.d'
```

In order to copy a directory, its contents must be copied as well, including all files within the directories and all of its subdirectories. This can be done by using the -r or -R recursive options.

```
sysadmin@localhost:~$ cp -R -v /etc/perl ~

'/etc/perl' -> '/home/sysadmin/perl'

'/etc/perl/CPAN' -> '/home/sysadmin/perl/CPAN'

'/etc/perl/Net' -> '/home/sysadmin/perl/Net'

'/etc/perl/Net/libnet.cfg' -> '/home/sysadmin/perl/Net/libnet.cfg'
```

Copying a file using the `cp` command requires the execute permission to access the directory where the file is located and the read permission for the file you are trying to copy.

You will also need to have the write and execute permissions on the directory where you want to put the copied file.

Command	File Type	Permission
`cp`	Source Directory	Execute
`cp`	Source File	Read
`cp`	Destination Directory	Write, Execute

Consider This

The archive `-a` option of the `cp` command copies the contents of the file and also attempts to maintain the original timestamps and file ownership. For regular users, only original timestamps can be maintained, since regular users can't create files owned by other users. If the root user uses the `-a` option, then the `cp` command will create a new file owned by the original file owner and also use the original file's timestamps.

The `-a` option also implies that recursion will be done. Therefore, it can also be used to copy a directory.

5.6 Moving Files and Directories

The `mv` command is used to move a file from one path name in the filesystem to another. When you move a file, it's like creating a copy of the original file with a new path name and then deleting the original file.

```
mv [OPTION]... SOURCE DESTINATION
```

If you move a file from one directory to another, and you don't specify a new name for the file, then it will retain its original name. For example, the following will move the ~/Documents/red.txt file into the home directory and the resulting file name will be red.txt:

```
sysadmin@localhost:~$ cd Documents
sysadmin@localhost:~/Documents$ mv red.txt ~
sysadmin@localhost:~/Documents$ ls ~
Desktop Downloads  Pictures  Templates  red.txt
Documents  Music    Public    Videos
```

Like the cp command, the mv command will allow you to specify multiple files to move, as long as the final argument provided to the command is a directory.

```
mv [OPTION]... SOURCE... DIRECTORY
```

For example:

```
sysadmin@localhost:~/Documents$ mv animals.txt food.txt alpha.txt /tmp
sysadmin@localhost:~/Documents$ ls /tmp
alpha.txt  animals.txt  food.txt
```

There is no need for a recursive option with the mv command. When a directory is moved, everything it contains is automatically moved as well.

Moving a file within the same directory is an effective way to rename it. For example, in the following command, the people.csv file is moved from the current directory to the current directory and given a new name of founders.csv. In other words, people.csv is renamed founders.csv:

```
sysadmin@localhost:~/Documents$ mv people.csv founders.csv
```

Use the mv command to move a directory from one location to another. Just like moving files, the original directory will be deleted from its previous location. In the following example, the mv command is used to move the Engineering directory from the School directory to the Work directory:

```
sysadmin@localhost:~/Documents$ cd
sysadmin@localhost:~$ ls Documents/School
Art  Engineering  Math
sysadmin@localhost:~$ mv Documents/School/Engineering Documents/Work
sysadmin@localhost:~$ ls Documents/Work
Engineering
```

Moving a file using the mv command requires the write and execute permissions on both the origin and destination directories.

5.7 Deleting Files

The rm command is used to remove files and directories.

```
rm [OPTION]... [FILE]...
```

Warning

It is important to keep in mind that deleted files and directories do not go into a "trash can" as with desktop-oriented operating systems. When a file is deleted with the rm command, it is permanently gone.

Without any options, the rm command is typically used to remove regular files:

```
sysadmin@localhost:~$ cd Documents
sysadmin@localhost:~/Documents$ rm alpha.txt
sysadmin@localhost:~/Documents$ ls alpha.txt
ls: cannot access alpha.txt: No such file or directory
```

Extra care should be applied when using wildcards to specify which files to remove, as the extent to which the pattern might match files may be beyond what was anticipated. To avoid accidentally deleting files when using globbing characters, use the -i option. This option makes the rm command confirm interactively every file that you delete:

```
sysadmin@localhost:~/Documents$ rm -i a*
rm: remove regular file 'adjectives.txt'? y
rm: remove regular file 'alpha-first.txt'? y
rm: remove regular file 'alpha-first.txt.original'? y
rm: remove regular file 'alpha-second.txt'? y
rm: remove regular file 'alpha-third.txt'? y
rm: remove regular file 'animals.txt'? Y
sysadmin@localhost:~/Documents$ cd
sysadmin@localhost:~$
```

Some distributions make the -i option a default option by making an alias for the rm command.

Deleting a file with the rm command requires the write and execute permissions on its parent directory. Regular users typically only have this type of permission in their home directory and its subdirectories.

Command	File Type	Permission
rm	Parent Directory	Write, Execute

Consider This

The /tmp and /var/tmp directories do have the special permission called sticky bit set on them so that files in these directories can only be deleted by the user that owns them (with the exception of the root user who can delete any file in any directory). So, this means if you copy a file to the /tmp directory, then other users of the system will not be able to delete your file.

5.8 Creating Directories

The mkdir command allows you to create (make) a directory. Creating directories is an essential file management skill, since you will want to maintain some functional organization with your files and not have them all placed in a single directory.

```
mkdir [OPTION]... DIRECTORY...
```

Typically, you have a handful of directories in your home directory by default. Exactly what directories you have will vary due to the distribution of Linux, what software has been installed on your system, and actions that may have been taken by the administrator.

For example, upon successful graphical login on a default installation of Ubuntu, the following directories have already been created automatically in the user's home directory:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

The bin directory is a common directory for users to create in their home directory. It is a useful directory to place scripts that the user has created. In the following example, the user creates a bin directory within their home directory:

```
sysadmin@localhost:~$ mkdir bin
sysadmin@localhost:~$ ls
Desktop Downloads  Pictures  Templates  bin
Documents  Music   Public    Videos
```

The mkdir command can accept a list of space-separated path names for new directories to create:

```
sysadmin@localhost:~$ mkdir one two three
sysadmin@localhost:~$ ls
Desktop Downloads  Pictures  Templates  bin     two
Documents  Music   Public    Videos     one     three
```

Directories are often described in terms of their relationship to each other. If one directory contains another directory, then it is referred to as the parent directory. The subdirectory is referred to as a child directory. In the following example, /home is the parent directory and /sysadmin is the child directory:

```
sysadmin@localhost:~$ pwd
/home/sysadmin
```

When creating a child directory, its parent directory must first exist. If an administrator attempted to create a directory named blue inside of the non-existent /home/sysadmin/red directory, there would be an error:

```
sysadmin@localhost:~$ mkdir /home/sysadmin/red/blue
mkdir: cannot create directory '/home/sysadmin/red/blue': No such file or
directory
```

By adding the `-p` option, the `mkdir` command automatically creates the parent directories for any child directories about to be created. This is especially useful for making deep path names:

```
sysadmin@localhost:~$ mkdir -p /home/sysadmin/red/blue/yellow/green
sysadmin@localhost:~$ ls -R red
red:
blue

red/blue:
yellow

red/blue/yellow:
green

red/blue/yellow/green:
```

To create a directory with the `mkdir` command, you must have the write and execute permissions on the parent of the proposed directory. For example, to create the /etc/test directory requires the write and execute permissions in the /etc directory.

Command	File Type	Permission
mkdir	Parent Directory	Write, Execute

5.9 Removing Directories

The rmdir command is used to remove empty directories:

```
rmdir [OPTION]... DIRECTORY...
```

```
sysadmin@localhost:~$ rmdir bin
```

Using the -p option with the rmdir command will remove directory paths, but only if all of the directories contain other empty directories.

```
sysadmin@localhost:~$ rmdir red
rmdir: failed to remove 'red': Directory not empty
sysadmin@localhost:~$ rmdir -p red/blue/yellow/green
```

Otherwise, if a directory contains anything except other directories, you'll need to use the rm command with a recursive option. The rm command alone will ignore directories that it's asked to remove; to delete a directory, use either the -r or -R recursive options. Just be careful as this will delete all files and all subdirectories:

```
sysadmin@localhost:~$ mkdir test
sysadmin@localhost:~$ touch test/file1.txt
sysadmin@localhost:~$ rmdir test
rmdir: failed to remove 'test': Directory not empty
sysadmin@localhost:~$ rm test
rm: cannot remove 'test': Is a directory

sysadmin@localhost:~$ rm -r test
sysadmin@localhost:~$
```

To delete a directory with the `rmdir` command, you must have
the write and execute permissions on the parent directory. For example, to remove
the /etc/test directory requires the write and execute permissions in the /etc directory.

Command	**File Type**	**Permission**
`rmdir`	Parent Directory	Write, Execute