6.Working with files and directories

6.1 Introduction

When working in a Linux Operating System, you will need to know how to manipulate files and directories. Some Linux distributions have GUI-based applications that allow you to manage files, but it is important to know how to perform these operations via the command line.

The command line features a rich collection of commands that allow you to manage files. In this chapter you will learn how to list files in a directory as well as how to copy, move and delete files.

The core concepts taught in this chapter will be important for later chapters as more file manipulate commands are covered, such as how to view files, compress files and set file permissions.

6.3 Understanding Files and Directories

Files are used to store data such as text, graphics and programs. Directories (AKA, "folders") are used to provide a hierarchical organization structure. This structure is somewhat different than what you might be used to if you have previously worked on Microsoft Windows systems.

On a Windows system, the *top level* of the directory structure is called *My Computer*. Each physical device (hard drive, DVD drive, USB thumb drive, network drive, etc.) shows up under My Computer, each assigned a drive letter, such as *C:* or *D:*. A visual representation of this structure:
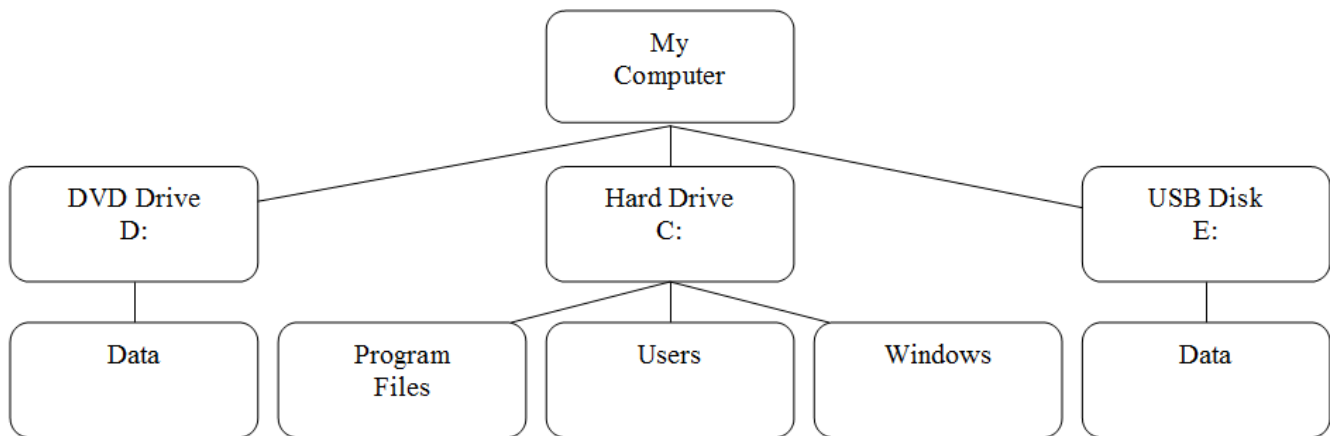
Like Windows, a Linux directory structure has a top level, however it is not called My Computer, but rather the *root directory* and it is symbolized by the / character. There are also no drives in Linux; each physical device is accessible under a directory, not a drive letter. A visual representation of a typical Linux directory structure:
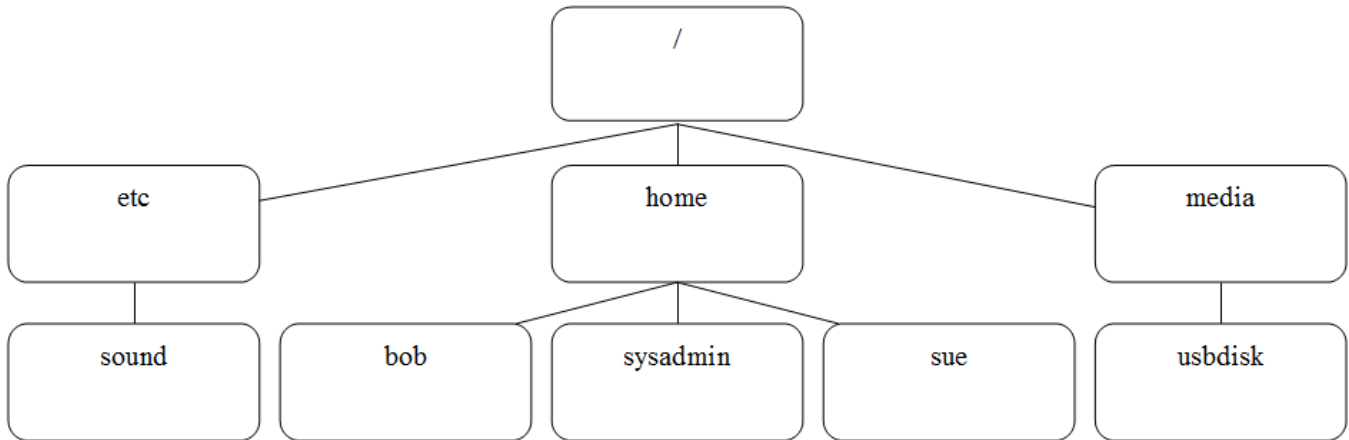
This directory structure is called the *filesystem* by most Linux users.

6.3.1 Directory Path

Using the previous graphic as a point of reference, you will see that there is a directory named sound under a directory named etc, which is under the / directory. An easier way to say this, is to refer to the *path*.

A path allows you to specify the exact location of a directory. For the sound directory, the path would be /etc/sound. The first / character represents the root directory while each other / character is used to separate the directory names.

This sort of path is called an *absolute path*. With an absolute path, you always provide directions to a directory (or a file) starting from the top of the directory structure, the root directory. Later in this chapter, we will cover a different sort of path called a *relative path*.

The following graphic demonstrates three additional absolute paths:
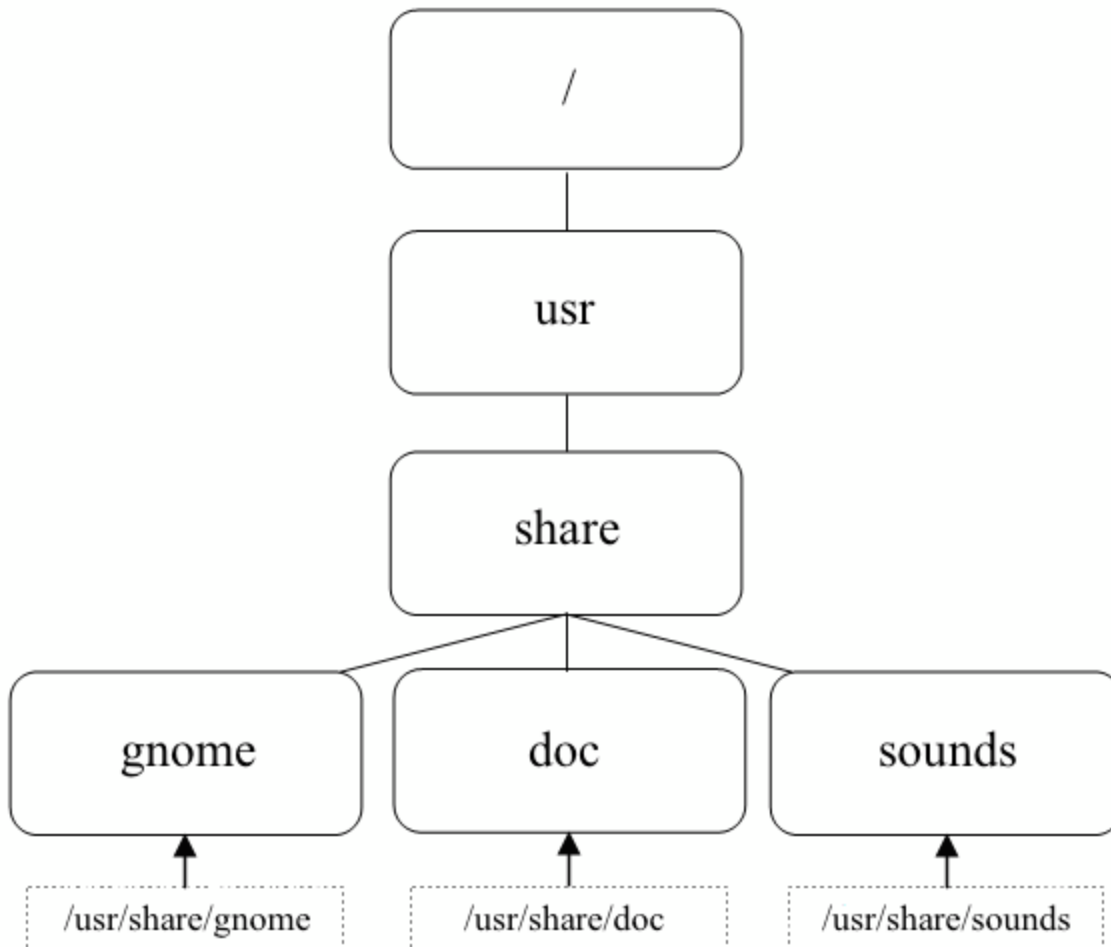
6.3.2 Home Directory

The term *home directory* often causes confusion to beginning Linux users. To begin with, on most Linux distributions there is a directory called home under the root directory: /home.

Under this /home directory there will be a directory for each user on the system. The directory name will be the same as the name of the user, so a user named bob would have a *home directory* called /home/bob.

Your home directory is a very important directory. To begin with, when you open a shell, you should automatically be placed in your home directory, as this is where you will do most of your work.

Additionally, your home directory is one of the few directories where you have the full control to create and delete additional files and directories. Most other directories in a Linux filesystem are protected with *file permissions*, a topic that will be covered in detail in a later chapter.

On most Linux distributions, the only users who can access any files in your home directory are you and the administrator on the system (the *root* user). This can be changed by using file permissions.
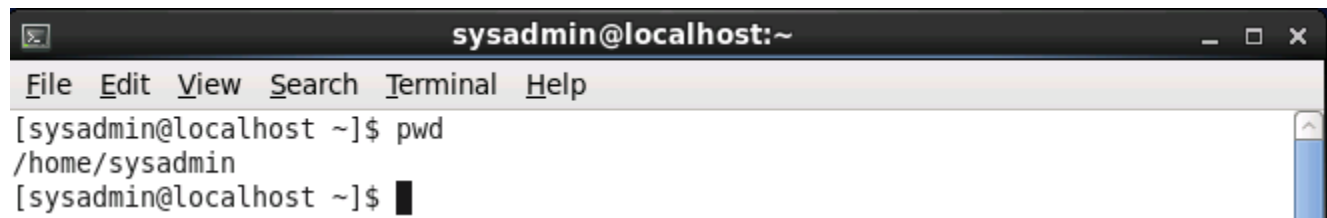
Your home directory even has a special symbol that you can use to represent it: ~. If your home directory is /home/sysadmin, you can just type ~ on the command line in place of /home/sysadmin. You can also refer to another

user's home directory by using the notation~*user*, where *user* is the name of the user account whose home directory you want to refer to. For example, ~bob would be the same as /home/bob.

6.3.3 Current Directory

Your *current directory* is the directory where you are currently working in a terminal. When you first open a terminal, the current directory should be your home directory, but this can change as you explore the filesystem and change to other directories.

While you are in a command line environment, you can determine your current directory by using the pwd command:

Additionally, most systems have the default user prompt display the current directory:
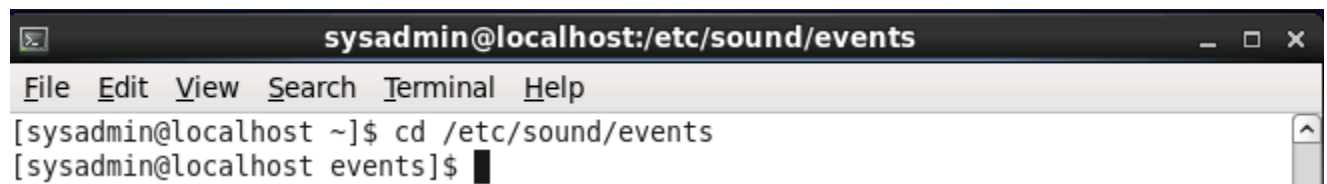
[sysadmin@localhost ~]$

In the graphic above, the ~ character indicates your current directory. As mentioned previously, the ~ character represents your home directory.

Normally the prompt only displays the name of the current directory, not the full path from the root directory down. In other words, if you were in the /usr/share/doc directory, your prompt will likely just provide you with the name doc for the current directory. If you want the full path, use the pwd command.

6.3.4 Changing Directories

If you want to change to a different directory, use the cd (change directory) command. For example, the following command will change the current directory to a directory called/etc/sound/events:

Note that there is no output if the cd command is successful. This is one of those "no news is good news" type of things. If you try to change to a directory that does not exist, you will receive an error message:

```
                  sysadmin@localhost:/etc/sound/events          _  □  ×

 File  Edit  View  Search  Terminal  Help

[sysadmin@localhost events]$ cd /etc/junk
bash: cd: /etc/junk: No such file or directory
[sysadmin@localhost events]$ ▮
```

If you want to return to your home directory, you can either type the cd command with no arguments or use
the cd command with the ~ character as an argument:

```
                       sysadmin@localhost:~                     _  □  ×

 File  Edit  View  Search  Terminal  Help

[sysadmin@localhost events]$ cd
[sysadmin@localhost ~]$ pwd
/home/sysadmin
[sysadmin@localhost ~]$ cd /etc
[sysadmin@localhost etc]$ cd ~
[sysadmin@localhost ~]$ pwd
/home/sysadmin
[sysadmin@localhost ~]$ ▮
```

6.3.5 Absolute vs. Relative Pathnames

Recall that a pathname is essentially a description of where a file or directory is located in the filesystem. You can also
consider a pathname to be directions that tell the system where to find a file or directory. For example, the cd
/etc/sound/events command means "change to theevents directory that you will find under the sound directory that
you will find under the etcdirectory that you will find under the / directory".

When you give a pathname that starts from the root directory, it is called an *absolute path*. In many cases, providing an
absolute path makes sense. For example, if you are in your home directory and you want to go to
the /etc/sound/events directory, then providing an absolute path to the cd command makes sense:

```
                        joe@localhost:~                          _  □  ×

 File  Edit  View  Search  Terminal  Help

[joe@localhost ~]$ ls -l /etc/grub.conf
lrwxrwxrwx. 1 root root 22 Nov  6  2012 /etc/grub.conf -> ../boot/grub/grub.conf
[joe@localhost ~]$ ▮
```

However, what if you were in the /etc/sound directory and you wanted to go to the/etc/sound/events directory? It
would be tedious to type the complete path to get to a directory that is only one level below your current location. In a
situation like this, you want to use a*relative path*:

```
joe@localhost:~                              _ □ ✕

File  Edit  View  Search  Terminal  Help
[joe@localhost ~]$ grub-md5-crypt
Password:
Retype password:
$1$eRsrbl$v1p5us79Harrku4vbJgKF0
[joe@localhost ~]$ █
```

A relative path provides directions using your **current location** as a point of reference. Recall that this is different from absolute paths, which always require you to use the **root directory** as a point of reference.

There is a handy relative path technique that you can use to move up one level in the directory structure: the .. directory. Regardless of which directory you are in, .. always represents one directory higher than your current directory (with the exception of when you are in the / directory):

```
sysadmin@localhost:/etc/sound                    _ □ ✕

File  Edit  View  Search  Terminal  Help
[sysadmin@localhost events]$ pwd
/etc/sound/events
[sysadmin@localhost events]$ cd ..
[sysadmin@localhost sound]$ pwd
/etc/sound
[sysadmin@localhost sound]$ █
```

Sometimes using relative pathnames are a better choice then absolute pathnames, however this is not always the case. Consider if you were in the /etc/sound/events directory and then you wanted to go to the /usr/share/doc directory. Using an absolute pathname, you would execute the cd /usr/share/doc command. Using relative pathnames, you would execute the cd ../../../usr/share/doc command:

```
sysadmin@localhost:/usr/share/doc                _ □ ✕

File  Edit  View  Search  Terminal  Help
[sysadmin@localhost events]$ pwd
/etc/sound/events
[sysadmin@localhost events]$ cd ../../../usr/share/doc
[sysadmin@localhost doc]$ pwd
/usr/share/doc
[sysadmin@localhost doc]$ █
```
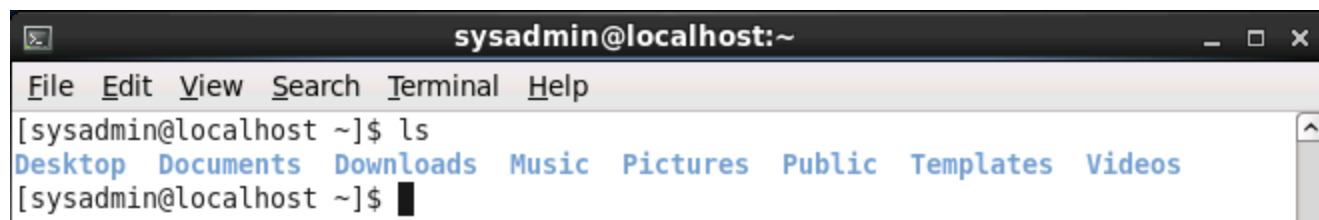
Relative and absolute paths are not just for the cd command. Any time you specify a file or a directory you can use either relative or absolute paths.

6.4 Listing Files in a Directory

Now that you are able to move from one directory to another, you will want to start displaying the contents of these directories. The ls command (ls is short for list) can be used to display the contents of a directory as well as detailed information about the files that are within a directory.

By itself, the ls command will list the files in the current directory:

```
sysadmin@localhost:~
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
[sysadmin@localhost ~]$ 
```
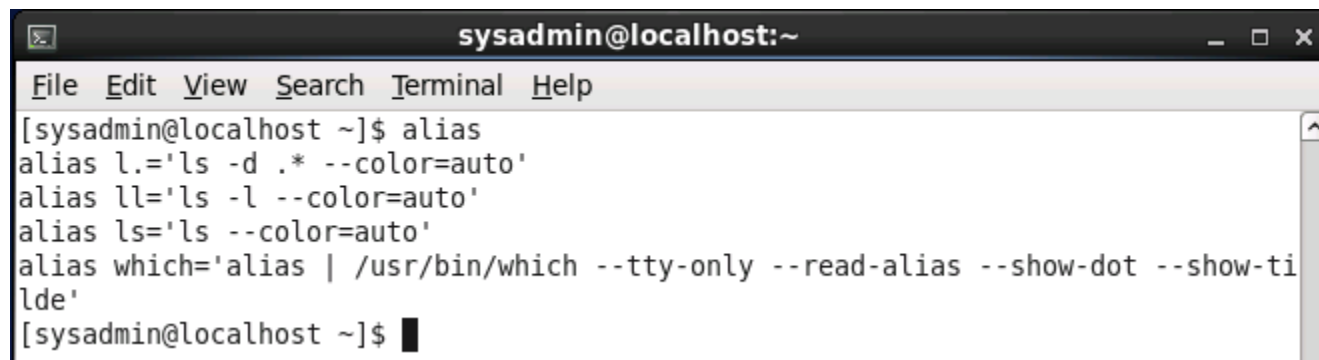
6.4.1 Listing Colors

There are many different types of files in Linux. As you learn more about Linux, you will discover many of these types. The following is a brief summary of some of the more common file types:

Type	Description
plain file	A file that isn't a special file type; also called a normal file
directory	A directory file (contains other files)
executable	A file that can be run like a program
symbolic link	A file that points to another file

On many Linux distributions, regular user accounts are modified so that the ls command displays filenames, color-coded by file type. For example, directories may be displayed in blue, executable files may be displayed in green, and symbolic links may be displayed in cyan (light blue).
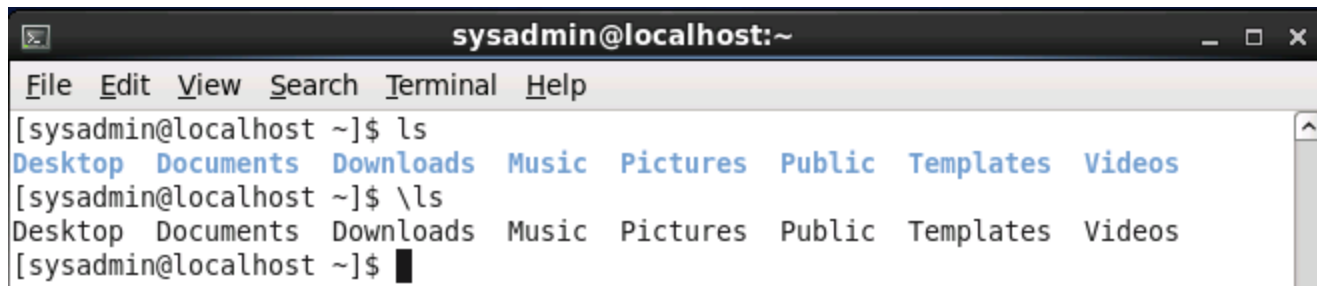
This is not a normal behavior for the ls command, but rather something that happens when you use the --color option to the ls command. The reason why ls seems to automatically perform this coloring, is that there is an *alias* for the ls command so it runs with the --coloroption:

```
sysadmin@localhost:~
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ alias
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-ti
lde'
[sysadmin@localhost ~]$ 
```

As you can see from the output above, when the ls command is executed, it really runs the command ls --color=auto.

In some cases, you might not want to see all of the colors (they can be a bit distracting sometimes). To avoid using the alias, place a backslash character (\ ) in front of your command:
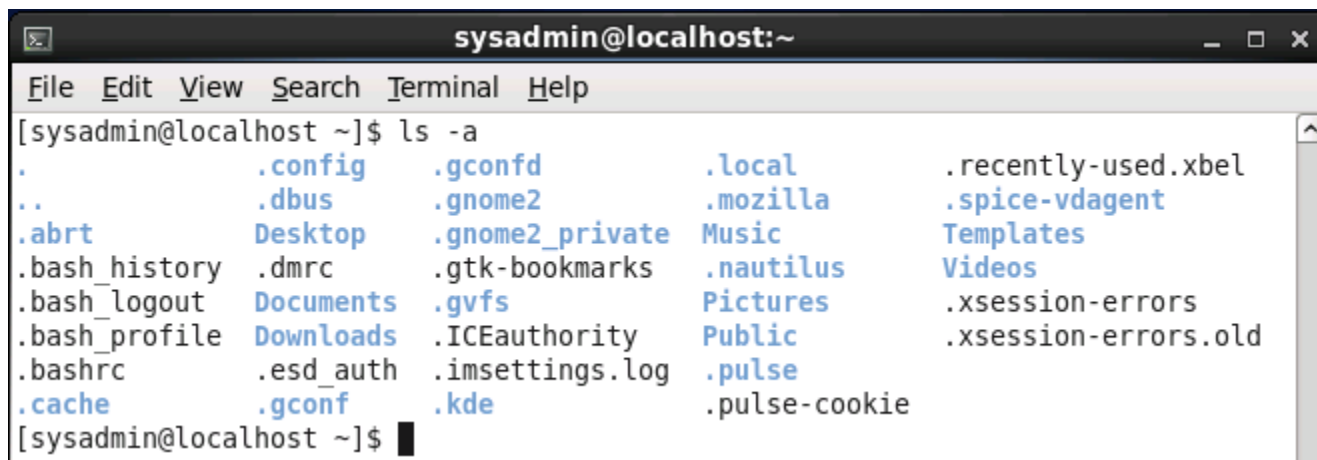

6.4.2 Listing Hidden Files

When you use the ls command to display the contents of a directory, not all files are shown automatically.
The ls command doesn't display *hidden files* by default. A hidden file is any file (or directory) that begins with a dot (.) character.

To display all files, including hidden files, use the -a option to the ls command:
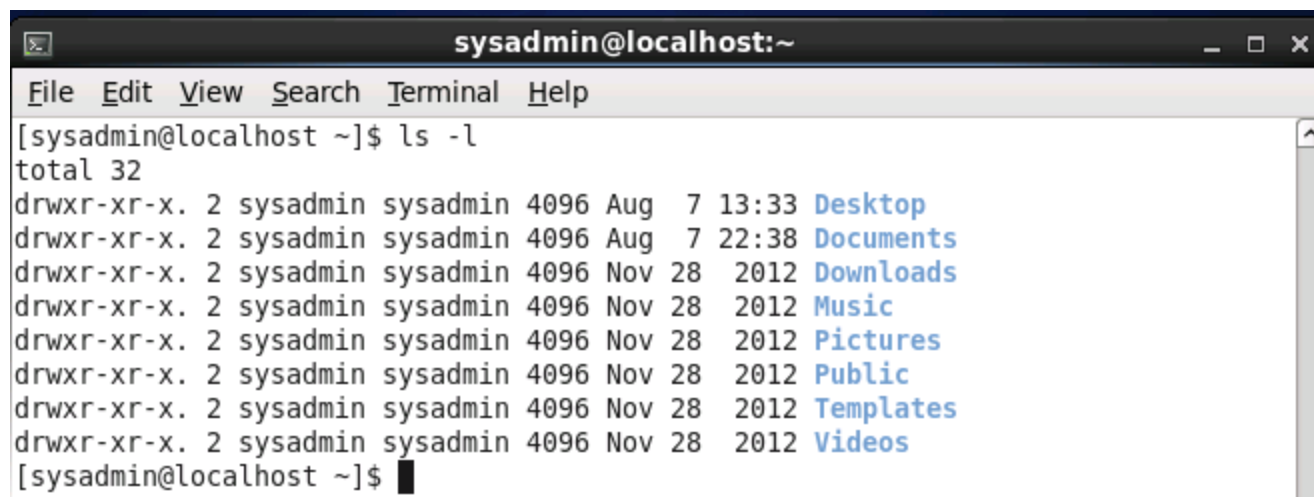
Why are files hidden in the first place? Most of the hidden files are *customization files*, designed to customize how Linux, your shell or programs work. For example, the .bashrc file in your home directory customizes features of the shell, such as creating or modifying variables and aliases.

These customization files are not ones that you work with on a regular basis. There are also many of them, as you can see, and having them displayed will make it more difficult to find the files that you do regularly work with. So, the fact that they are hidden is to your benefit.

6.4.3 Long Display Listing

There is information about each file, called *metadata* that is sometimes helpful to display. This may include who owns a file, the size of a file and the last time the contents of a file were modified. You can display this information by using the -l option to the ls command:

```
sysadmin@localhost:~                               _ □ ×

File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls -l
total 32
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 22:38 Documents
drwxr-xr-x. 2 sysadmin sysadmin 4096 Nov 28  2012 Downloads
drwxr-xr-x. 2 sysadmin sysadmin 4096 Nov 28  2012 Music
drwxr-xr-x. 2 sysadmin sysadmin 4096 Nov 28  2012 Pictures
drwxr-xr-x. 2 sysadmin sysadmin 4096 Nov 28  2012 Public
drwxr-xr-x. 2 sysadmin sysadmin 4096 Nov 28  2012 Templates
drwxr-xr-x. 2 sysadmin sysadmin 4096 Nov 28  2012 Videos
[sysadmin@localhost ~]$ ▊
```

In the output above, each line describes metadata about a single file. The following describes each of the fields of data that you will see in the output of the ls -l command:

File type

drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug 7 13:33 Desktop

The first character of each output line indicated the type of file. Common file types include:

    d = directory
    - = plain file
    l = symbolic link

Permissions

drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug 7 13:33 Desktop

The next ten characters will demonstrate the *permissions* of the file. The permissions are used to determine who has access to the file. They will be covered in detail in a later chapter.

Hard link count

drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug 7 13:33 Desktop

The hard link count of a file is used to demonstrate how many hard links there are to this file. Links are more of an administrator topic and not covered in this course.

User owner

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

Every file is owned by a user account. This is important because the owner has the rights to set permissions on a file and the owner has his/her own permissions on the file. Permissions will be covered in detail in a later chapter.

Group owner

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

Every file is owned by a group account. This is important because any member of this group will have special access to this file based on the group permissions on the file. Permissions will be covered in detail in a later chapter.

File size

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

This field describes the size of the file in bytes. Note: For directories, this value does not describe the total size of the directory, but rather how many bytes are reserved to keep track of the filenames in the directory (in other words, ignore this field for directories).

Modification timestamp

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

This field indicates the last time the file contents were modified. For directories this timestamp indicates the last time a file as added or deleted from the directory.
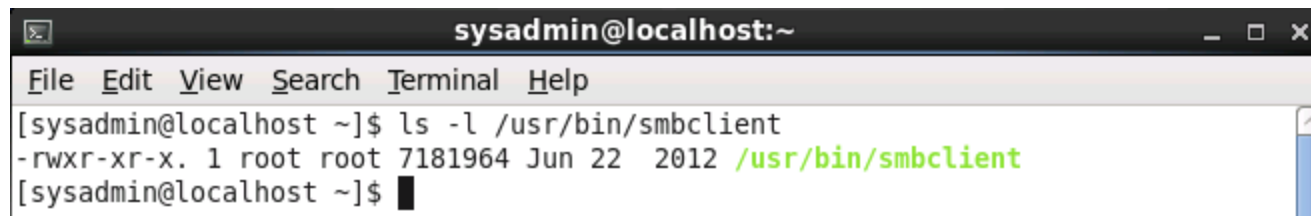
Name

```
drwxr-xr-x. 2 sysadmin sysadmin 4096 Aug  7 13:33 Desktop
```

The last field is the name of the file or directory.

6.4.3.1 Human Readable Sizes

When you display file sizes with the -l option to the ls command, you end up with file sizes in bytes. For text files, a byte is 1 character.

For smaller files, byte sizes are fine. However, for larger files it is hard to comprehend how large the file is. For example, consider the output of the following command:
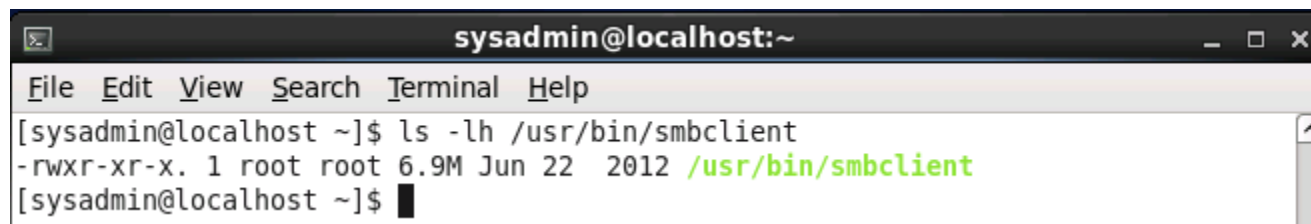
```
sysadmin@localhost:~
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls -l /usr/bin/smbclient
-rwxr-xr-x. 1 root root 7181964 Jun 22  2012 /usr/bin/smbclient
[sysadmin@localhost ~]$
```

As you can see, the file size is hard to determine in bytes. Is 7181964 a large file or small? It seems fairly large, but it is hard to determine using bytes.

Think of it this way: if someone were to give you the distance between Boston and New York using inches, that value would essentially be meaningless because for a distance like that, you think in terms of miles.

It would be better if the file size was presented in a more human readable size, like megabytes or gigabytes. To accomplish this, use the -h option to the ls command:

```
sysadmin@localhost:~
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls -lh /usr/bin/smbclient
-rwxr-xr-x. 1 root root 6.9M Jun 22  2012 /usr/bin/smbclient
[sysadmin@localhost ~]$
```

The -l option must be used with the -h option.

6.4.4 Recursive Listing

There will be times when you want to display all of the files in a directory as well as all of the files in all subdirectories under a directory. This is called a *recursive listing*.

To perform a recursive listing, use the -R option to the ls command:

```
sysadmin@localhost:~
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls -R /etc/ppp
/etc/ppp:
chap-secrets  ip-down.ipv6to4  ip-up.ipv6to4  ipv6-up   pap-secrets
ip-down       ip-up            ipv6-down      options   peers

/etc/ppp/peers:
[sysadmin@localhost ~]$
```

Note that in the previous example, the files in the /etc/ppp directory were listed first. After that, the files in the /etc/ppp/peers directory were listed (there were no files in this case, but if any file had been in this directory, they would have been displayed).

Be careful with this option; for example, running the command ls -R / would list every file on the file system, including all files on any attached USB device and DVD in the system. Limit the use of the -R option to smaller directory structures.

6.4.5 Sort a Listing

By default, the ls command sorts files alphabetically by file name. Sometimes, It may be useful to sort files using different criteria.

To sort files by size, we can use the -S option. Note the difference in the output of the following two commands:

```
[sysadmin@localhost ~]$ ls /etc/ppp
chap-secrets   ip-down.ipv6to4   ip-up.ipv6to4   ipv6-up   pap-secrets
ip-down        ip-up             ipv6-down       options   peers
[sysadmin@localhost ~]$ ls -S /etc/ppp
ip-up.ipv6to4  ip-down.ipv6to4   ipv6-down   ip-down        pap-secrets
peers          ipv6-up           ip-up       chap-secrets   options
[sysadmin@localhost ~]$
```

The same files and directories are listed, but in a different order. While the -S option works by itself, you can't really tell that the output is sorted by size, so it is most useful when used with the -l option. The following command will list files from largest to smallest and display the actual size of the file.

```
[sysadmin@localhost ~]$ ls -lS /etc/ppp
total 44
-rwxr-xr-x. 1 root root 6517 Apr 27  2012 ip-up.ipv6to4
drwxr-xr-x. 2 root root 4096 Jun 22  2012 peers
-rwxr-xr-x. 1 root root 3262 Apr 27  2012 ip-down.ipv6to4
-rwxr-xr-x. 1 root root 3196 Apr 27  2012 ipv6-up
-rwxr-xr-x. 1 root root 1687 Apr 27  2012 ipv6-down
-rwxr-xr-x. 1 root root  430 Apr 27  2012 ip-up
-rwxr-xr-x. 1 root root  386 Apr 27  2012 ip-down
-rw-------. 1 root root   78 Aug 22  2010 chap-secrets
-rw-------. 1 root root   77 Aug 22  2010 pap-secrets
-rw-r--r--. 1 root root    5 Aug 22  2010 options
[sysadmin@localhost ~]$
```

It may also be useful to use the -h option to display human-readable file sizes:
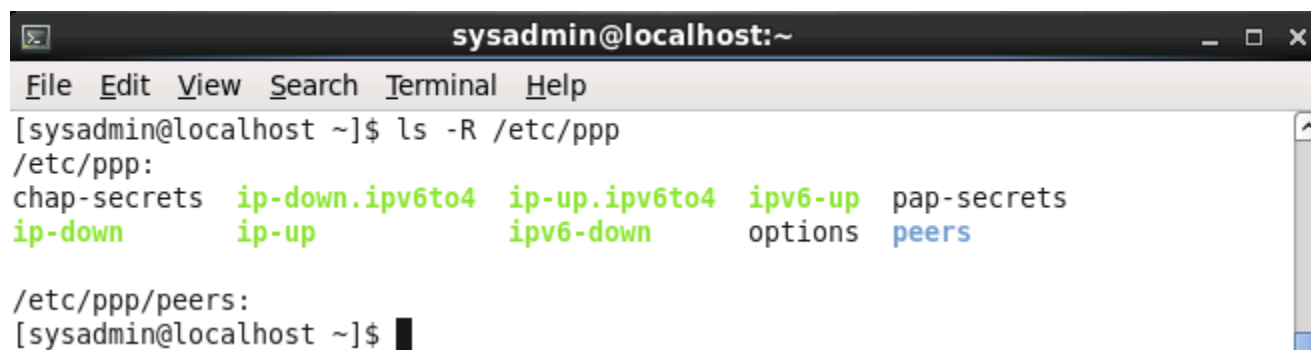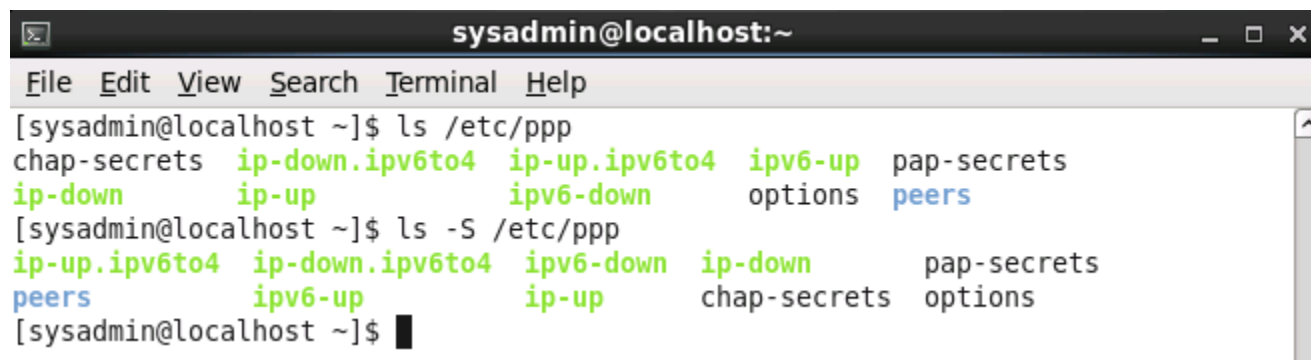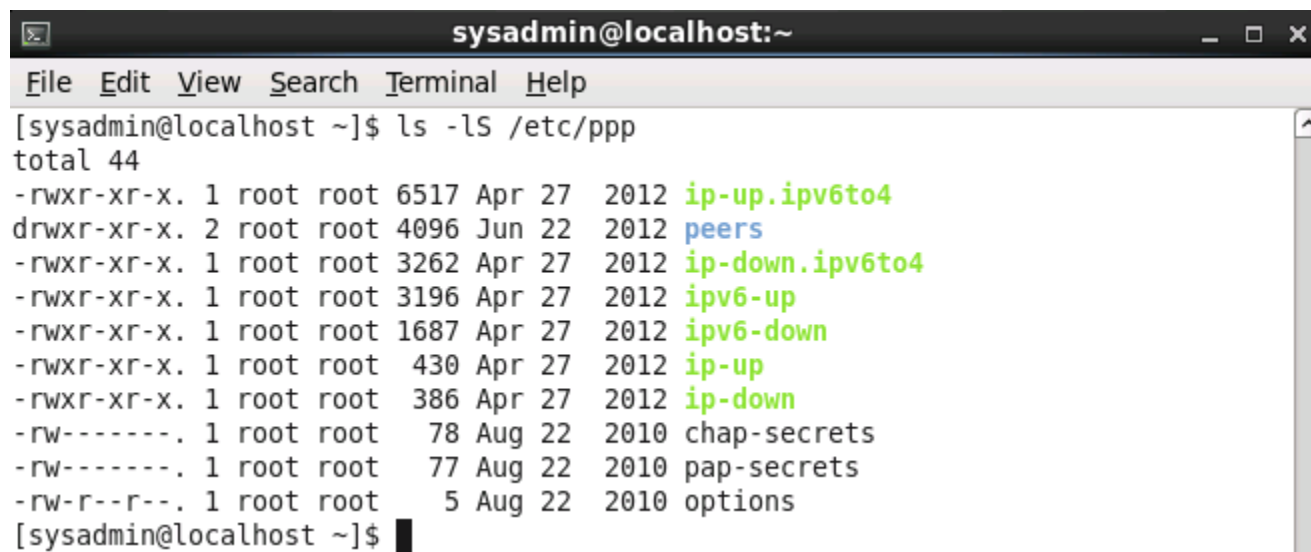
```
                              sysadmin@localhost:~                    _  ☐  ✕

File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls -lSh /etc/ppp
total 44K
-rwxr-xr-x. 1 root root 6.4K Apr 27  2012 ip-up.ipv6to4
drwxr-xr-x. 2 root root 4.0K Jun 22  2012 peers
-rwxr-xr-x. 1 root root 3.2K Apr 27  2012 ip-down.ipv6to4
-rwxr-xr-x. 1 root root 3.2K Apr 27  2012 ipv6-up
-rwxr-xr-x. 1 root root 1.7K Apr 27  2012 ipv6-down
-rwxr-xr-x. 1 root root  430 Apr 27  2012 ip-up
-rwxr-xr-x. 1 root root  386 Apr 27  2012 ip-down
-rw-------. 1 root root   78 Aug 22  2010 chap-secrets
-rw-------. 1 root root   77 Aug 22  2010 pap-secrets
-rw-r--r--. 1 root root    5 Aug 22  2010 options
[sysadmin@localhost ~]$ ▮
```

It is also possible to sort files based on the time they were modified. You can do this by using the -t option.

The -t option will list the most recently modified files first. This option can be used alone, but again, is usually more helpful when paired with the -l option:

```
                              sysadmin@localhost:~                    _  ☐  ✕

File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls -tl /etc/ppp
total 44
drwxr-xr-x. 2 root root 4096 Jun 22  2012 peers
-rwxr-xr-x. 1 root root  386 Apr 27  2012 ip-down
-rwxr-xr-x. 1 root root 3262 Apr 27  2012 ip-down.ipv6to4
-rwxr-xr-x. 1 root root  430 Apr 27  2012 ip-up
-rwxr-xr-x. 1 root root 6517 Apr 27  2012 ip-up.ipv6to4
-rwxr-xr-x. 1 root root 1687 Apr 27  2012 ipv6-down
-rwxr-xr-x. 1 root root 3196 Apr 27  2012 ipv6-up
-rw-------. 1 root root   78 Aug 22  2010 chap-secrets
-rw-r--r--. 1 root root    5 Aug 22  2010 options
-rw-------. 1 root root   77 Aug 22  2010 pap-secrets
[sysadmin@localhost ~]$ ▮
```

It is important to remember that the modified date on directories represents the last time a file was added to or removed from the directory.

If the files in a directory were modified many days or months ago, it may be harder to tell exactly when they were modified, as only the date is provided for older files. For more detailed modification time information you can use the --full-time option to display the complete timestamp (including hours, seconds, minutes...):

```
                        sysadmin@localhost:~              _ □ ✕
 File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls -t --full-time /etc/ppp
total 44
drwxr-xr-x. 2 root root 4096 2012-06-22 14:51:40.000000000 -0500 peers
-rwxr-xr-x. 1 root root  386 2012-04-27 04:48:34.000000000 -0500 ip-down
-rwxr-xr-x. 1 root root 3262 2012-04-27 04:48:34.000000000 -0500 ip-down.ipv6to4
-rwxr-xr-x. 1 root root  430 2012-04-27 04:48:34.000000000 -0500 ip-up
-rwxr-xr-x. 1 root root 6517 2012-04-27 04:48:34.000000000 -0500 ip-up.ipv6to4
-rwxr-xr-x. 1 root root 1687 2012-04-27 04:48:34.000000000 -0500 ipv6-down
-rwxr-xr-x. 1 root root 3196 2012-04-27 04:48:34.000000000 -0500 ipv6-up
-rw-------. 1 root root   78 2010-08-22 18:21:35.000000000 -0500 chap-secrets
-rw-r--r--. 1 root root    5 2010-08-22 18:21:35.000000000 -0500 options
-rw-------. 1 root root   77 2010-08-22 18:21:35.000000000 -0500 pap-secrets
[sysadmin@localhost ~]$ █
```

The --full-time option will assume the -l option automatically.

It is possible to perform a reverse sort with either the -S or -t options by using the -r option. The following command will sort files by size, smallest to largest:

```
                        sysadmin@localhost:~              _ □ ✕
 File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls -lrS /etc/ppp
total 44
-rw-r--r--. 1 root root    5 Aug 22  2010 options
-rw-------. 1 root root   77 Aug 22  2010 pap-secrets
-rw-------. 1 root root   78 Aug 22  2010 chap-secrets
-rwxr-xr-x. 1 root root  386 Apr 27  2012 ip-down
-rwxr-xr-x. 1 root root  430 Apr 27  2012 ip-up
-rwxr-xr-x. 1 root root 1687 Apr 27  2012 ipv6-down
-rwxr-xr-x. 1 root root 3196 Apr 27  2012 ipv6-up
-rwxr-xr-x. 1 root root 3262 Apr 27  2012 ip-down.ipv6to4
drwxr-xr-x. 2 root root 4096 Jun 22  2012 peers
-rwxr-xr-x. 1 root root 6517 Apr 27  2012 ip-up.ipv6to4
[sysadmin@localhost ~]$ █
```

The following command will list files by modification date, oldest to newest:

```
[sysadmin@localhost ~]$ ls -lrt /etc/ppp
total 44
-rw-------. 1 root root   77 Aug 22  2010 pap-secrets
-rw-r--r--. 1 root root    5 Aug 22  2010 options
-rw-------. 1 root root   78 Aug 22  2010 chap-secrets
-rwxr-xr-x. 1 root root 3196 Apr 27  2012 ipv6-up
-rwxr-xr-x. 1 root root 1687 Apr 27  2012 ipv6-down
-rwxr-xr-x. 1 root root 6517 Apr 27  2012 ip-up.ipv6to4
-rwxr-xr-x. 1 root root  430 Apr 27  2012 ip-up
-rwxr-xr-x. 1 root root 3262 Apr 27  2012 ip-down.ipv6to4
-rwxr-xr-x. 1 root root  386 Apr 27  2012 ip-down
drwxr-xr-x. 2 root root 4096 Jun 22  2012 peers
[sysadmin@localhost ~]$
```

6.4.6 Listing With Globs

In a previous chapter, we discussed the use of file globs to match filenames using wildcard characters. For example, we demonstrated that you can list all of the files in the /etc directory that begin with the letter "e" with the following command:


```
[sysadmin@localhost ~]$ echo /etc/e*
/etc/enscript.cfg /etc/environment /etc/ethers /etc/event.d /etc/exports
[sysadmin@localhost ~]$
```

Now that you know that the ls command is normally used to list files in a directory, using the echo command may seem to have been a strange choice. However, there is something about the ls command that might have caused confusion while we were discussing globs. This "feature" might also cause problems when you try to list files using glob patterns.

Keep in mind that it is the shell, not the echo or ls command, that expands the glob pattern into corresponding file names. In other words, when you typed the echo /etc/e* command, what the shell did **before** executing the echo command was replace e* with all of the files and directories within the /etc directory that match the pattern.

So, if you were to run the ls /etc/e* command, what the shell would really run would be this:

ls /etc/encript.cfg /etc/environment /etc/ethers /etc/event.d /etc/exports

When the ls command sees multiple arguments, it performs a list operation on each item separately. In other words, the command ls /etc/encript.cfg /etc/environment is essentially the same as ls /etc/encript.cfg; ls /etc/environment.

Now consider what happens when you run the ls command on a file, such as encript.cfg:

```
┌─┐                    sysadmin@localhost:~              _  □  ✕
│▣│
 File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls /etc/enscript.cfg                       ▲
/etc/enscript.cfg
[sysadmin@localhost ~]$ █
```

As you can see, running the ls command on a single file results in the name of the file being printed. Typically this is useful if you want to see details about a specific file by using the -loption to the ls command:

```
┌─┐                    sysadmin@localhost:~              _  □  ✕
│▣│
 File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls -l /etc/enscript.cfg                    ▲
-rw-r--r--. 1 root root 4843 Nov 11  2010 /etc/enscript.cfg
[sysadmin@localhost ~]$ █
```

However, what if the ls command is given a directory name as an argument? In this case, the output of the command is different than if the argument was a file name:

```
┌─┐                    sysadmin@localhost:~              _  □  ✕
│▣│
 File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls /etc/event.d                            ▲
ck-log-system-restart  ck-log-system-start   ck-log-system-stop
[sysadmin@localhost ~]$ █
```

If you give a directory name as an argument to the ls command, the command will display the**contents** of the directory (the names of the files in the directory), not just provide the directory name. The filenames you see in the example above are the names of the files in the/etc/event.d directory.

Why is this a problem when using globs? Consider the following output:

```
┌─┐                    sysadmin@localhost:~              _  □  ✕
│▣│
 File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls /etc/e*                                 ▲
/etc/enscript.cfg  /etc/environment  /etc/ethers  /etc/exports

/etc/event.d:
ck-log-system-restart  ck-log-system-start   ck-log-system-stop
[sysadmin@localhost ~]$ █
```

As you can see, when the ls command sees a filename as an argument, it just displays the filename. However, for any directory, it will display the contents of the directory, not just the directory name.

This becomes even more confusing in a situation like the following:

```
sysadmin@localhost:~                          _ □ ✕

File  Edit  View  Search  Terminal  Help

[sysadmin@localhost ~]$ ls /etc/ev*
ck-log-system-restart  ck-log-system-start  ck-log-system-stop
[sysadmin@localhost ~]$ ▮
```

In the previous example, it seems like the ls command is just plain wrong. But what really happened is that the only thing that matches the glob /etc/ev* is the /etc/event.d directory. So, the ls command only displayed the files in that directory!

There is a simple solution to this problem: when you use glob arguments with the ls command, always use the -d option. When you use the -d option, then the ls command won't display the contents of a directory, but rather the name of the directory:

```
sysadmin@localhost:~                          _ □ ✕

File  Edit  View  Search  Terminal  Help

[sysadmin@localhost ~]$ ls -d /etc/e*
/etc/enscript.cfg  /etc/environment  /etc/ethers  /etc/event.d  /etc/exports
[sysadmin@localhost ~]$ ▮
```

6.5 Copying Files

The cp command is used to copy files. It requires that you specify a source and a destination. The structure of the command is as follows:

cp [source] [destination]

The *source* is the file you wish to copy. The *destination* is where you want the copy to be located. When successful, the cp command will not have any output (no news is good news). The following command will copy the /etc/hosts file to your home directory:

```
sysadmin@localhost:~                          _ □ ✕

File  Edit  View  Search  Terminal  Help

[sysadmin@localhost ~]$ cp /etc/hosts ~
[sysadmin@localhost ~]$ ls
Desktop    Downloads  Music      Public     Videos
Documents  hosts      Pictures   Templates
[sysadmin@localhost ~]$ ▮
```

Remember; the ~ character represents your home directory.

6.5.1 Verbose Mode

The -v option will cause the cp command to produce output if successful. The -v option stands for *verbose*:

```
sysadmin@localhost:~                    _ □ ✕

File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ cp -v /etc/hosts ~
`/etc/hosts' -> `/home/sysadmin/hosts'
[sysadmin@localhost ~]$ █
```

When the destination is a directory, the resulting new file will have the same name as the original file. If you want the new file to have a different name, you must provide the new name as part of the destination

```
sysadmin@localhost:~                    _ □ ✕

File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ cp /etc/hosts ~/hosts.copy
[sysadmin@localhost ~]$ ls
Desktop    Downloads  hosts.copy  Pictures  Templates
Documents  hosts      Music       Public    Videos
[sysadmin@localhost ~]$ █
```

6.5.2 Avoid Overwriting Data

The cp command can be destructive to existing data if the destination file already exists. In the case where the destination file exists , the cp command will overwrite the existing file's contents with the contents of the source file. To illustrate this potential problem, first a new file is created in the sysadmin home directory by copying an existing file:

```
sysadmin@localhost:~                    _ □ ✕

File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ cp /etc/skel/.bash_logout ~/example.txt
[sysadmin@localhost ~]$ █
```

View the output of the ls command to see the file and view the contents of the file using the more command:

```
root@localhost:~

File  Edit  View  Search  Terminal  Help
[root@localhost ~]# ls -l /etc/rc.d/rc5.d/K15httpd
lrwxrwxrwx. 1 root root 15 Mar  1 09:43 /etc/rc.d/rc5.d/K15httpd -> ../init
tpd
[root@localhost ~]# █
```

In the next example, you will see that the cp command destroys the original contents of the example.txt file. Notice that after the cp command is complete, the size of the file is different from the original (158 bytes rather then 18) and the contents are different as well:

```
sysadmin@localhost:~                          _ □ ✕
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ cp /etc/hosts ~/example.txt
[sysadmin@localhost ~]$ ls -l example.txt
-rw-rw-r--. 1 sysadmin sysadmin 158 Sep 21 14:11 example.txt
[sysadmin@localhost ~]$ cat example.txt
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6
[sysadmin@localhost ~]$ ▊
```

There are two options that can be used to safeguard against accidental overwrites. With the –I (interactive) option, the cp will prompt before overwriting a file. The following example will demonstrate this option, first restoring the content of the original file:


```
sysadmin@localhost:~                          _ □ ✕
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ cp /etc/skel/.bash_logout ~/example.txt
[sysadmin@localhost ~]$ ls -l example.txt
-rw-rw-r--. 1 sysadmin sysadmin 18 Sep 21 15:56 example.txt
[sysadmin@localhost ~]$ more example.txt
# ~/.bash_logout

[sysadmin@localhost ~]$ cp -i /etc/hosts ~/example.txt
cp: overwrite `/home/sysadmin/example.txt'? n
[sysadmin@localhost ~]$ ls -l example.txt
-rw-rw-r--. 1 sysadmin sysadmin 18 Sep 21 15:56 example.txt
[sysadmin@localhost ~]$ more example.txt
# ~/.bash_logout

[sysadmin@localhost ~]$ ▊
```

Notice that since the value of **n** (no) was given when prompted to overwrite the file, no changes were made to the file. If a value of **y** (yes) was given, then the copy process would have taken place.

The -i option requires you to answer **y** or **n** for every copy that could end up overwriting an existing file's contents. This can be tedious when a bunch of overwrites could occur, such as the example demonstrated below:
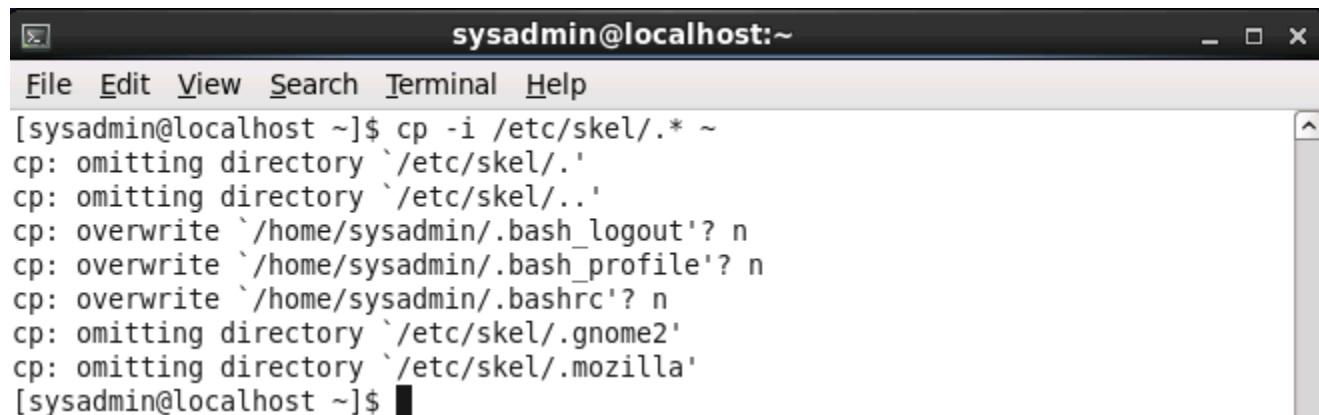


```
sysadmin@localhost:~                          _ □ ✕
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ cp -i /etc/skel/.* ~
cp: omitting directory `/etc/skel/.'
cp: omitting directory `/etc/skel/..'
cp: overwrite `/home/sysadmin/.bash_logout'? n
cp: overwrite `/home/sysadmin/.bash_profile'? n
cp: overwrite `/home/sysadmin/.bashrc'? n
cp: omitting directory `/etc/skel/.gnome2'
cp: omitting directory `/etc/skel/.mozilla'
[sysadmin@localhost ~]$ ▊
```

As you can see from the example above, the cp command tried to overwrite three existing files, forcing the user to answer three prompts. If this situation happened for 100 files, it could become very annoying, very quickly.

If you want to automatically answer **n** to each prompt, use the -n option. It essentially stands for "no rewrite".

6.5.3 Copying Directories

In a previous example, error messages were given when the cp command attempted to copy directories:

```
[sysadmin@localhost ~]$ cp -i /etc/skel/.* ~
cp: omitting directory `/etc/skel/.'
cp: omitting directory `/etc/skel/..'
cp: overwrite `/home/sysadmin/.bash_logout'? n
cp: overwrite `/home/sysadmin/.bash_profile'? n
cp: overwrite `/home/sysadmin/.bashrc'? n
cp: omitting directory `/etc/skel/.gnome2'
cp: omitting directory `/etc/skel/.mozilla'
[sysadmin@localhost ~]$
```

Where the output says "...omitting directory...", the cp command is saying that it cannot copy this item because the command does not copy directories by default. However, the -r option to thecp command will have it copy both files and directories.

Be careful with this option: the entire directory structure will be copied. This could result in copying a lot of files and directories!

6.6 Moving Files

To move a file, use the mv command. The syntax for the mv command is much like the cpcommand:

mv [source] [destination]

In the following example, the hosts file that was generated earlier is moved from the current directory to the Videos directory:

```
                          sysadmin@localhost:~                    _ □ ×
 File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls
Desktop     Downloads    hosts   Pictures   Templates
Documents   example.txt  Music   Public     Videos
[sysadmin@localhost ~]$ mv hosts Videos
[sysadmin@localhost ~]$ ls
Desktop     Downloads    Music       Public      Videos
Documents   example.txt  Pictures    Templates
[sysadmin@localhost ~]$ ls Videos
hosts
[sysadmin@localhost ~]$ ▮
```

When a file is moved, the file is removed from the original location and placed in a new location. This can be somewhat tricky in Linux because users need specific permissions to remove files from a directory. If you don't have the right permissions, you will receive a "permission denied" error message:

```
                          sysadmin@localhost:~                    _ □ ×
 File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ mv /etc/hosts .
mv: cannot move `/etc/hosts' to `./hosts': Permission denied
[sysadmin@localhost ~]$ ▮
```

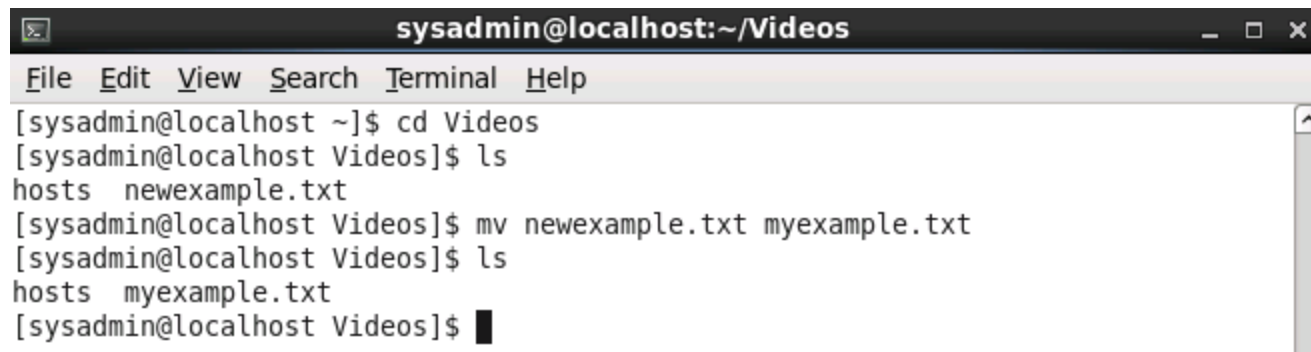A detailed discussion of permissions is provided in a later chapter.

6.7 Moving Files While Renaming

If the destination for the mv command is a directory, then the new file will have the same name as the original file. However, if the destination is not a directory, then the file is renamed during the move process:

```
                          sysadmin@localhost:~                    _ □ ×
 File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls
Desktop     Downloads    Music       Public      Videos
Documents   example.txt  Pictures    Templates
[sysadmin@localhost ~]$ mv example.txt Videos/newexample.txt
[sysadmin@localhost ~]$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
[sysadmin@localhost ~]$ ls Videos
hosts   newexample.txt
[sysadmin@localhost ~]$ ▮
```

6.7.1 Renaming Files

The mv command is not just used to move a file, but also to rename a file. For example, the following commands will rename the newexample.txt file to myexample.txt:

Think of the previous mv example to mean "move the newexample.txt file from the current directory back into the current directory and give the new file the name myexample.txt".

6.7.2 Additional mv Options

Like the cp command, the mv command provides the following options:

Option	Meaning
-i	Interactive move: ask if a file is to be overwritten.
-n	Do not overwrite a destination files' contents
-v	Verbose: show the resulting move

There is no -r option as the mv command will by default move directories.

6.8 Creating Files

There are several ways of creating a new file, including using a program designed to edit a file (a text editor). In a later chapter, text editors will be covered.

There is also a way to simply create a file that can be populated with data at a later time. This is a useful feature since for some operating system features, the very existence of a file could alter how a command or service works. It is also useful to create a file as a "placeholder" to remind you to create the file contents at a later time.

To create an empty file, use the touch command as demonstrated below:

```
                        sysadmin@localhost:~                    _ □ ✕
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
[sysadmin@localhost ~]$ touch sample
[sysadmin@localhost ~]$ ls -l sample
-rw-rw-r--. 1 sysadmin sysadmin 0 Sep 21 18:38 sample
[sysadmin@localhost ~]$ ▮
```

Notice the size of the new file is 0 bytes. As previously mentioned, the touch command doesn't place any data within the new file.

6.9 Removing Files

To delete a file, use the rm command:

```
                        sysadmin@localhost:~                    _ □ ✕
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls
Desktop    Downloads  Pictures  sample     Videos
Documents  Music      Public    Templates
[sysadmin@localhost ~]$ rm sample
[sysadmin@localhost ~]$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
[sysadmin@localhost ~]$ ▮
```
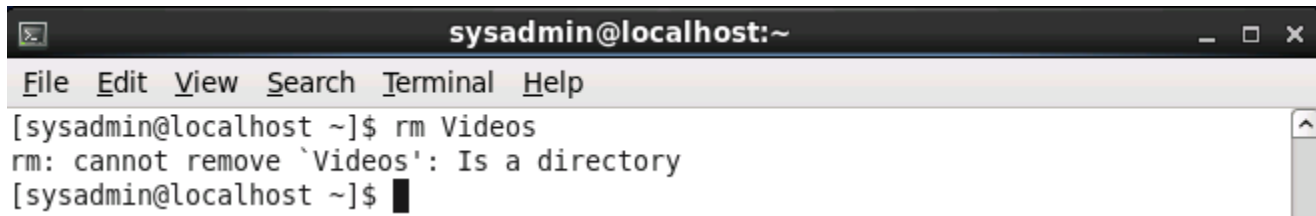
Note that the file was deleted with no questions asked. This could cause problems when deleting multiple files by using glob characters, for example: rm *.txt. Because these files are deleted without question, a user could end up deleting files that were not intended to be deleted.

Additionally, the files are permanently deleted. There is no command to undelete a file and no "trash can" from which to recover deleted files. As a precaution, users should use the -i option when deleting multiple files:

```
                        sysadmin@localhost:~                    _ □ ✕
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ touch sample.txt example.txt test.txt
[sysadmin@localhost ~]$ ls
Desktop    Downloads    Music     Public       Templates  Videos
Documents  example.txt  Pictures  sample.txt   test.txt
[sysadmin@localhost ~]$ rm -i *.txt
rm: remove regular empty file `example.txt'? y
rm: remove regular empty file `sample.txt'? n
rm: remove regular empty file `test.txt'? y
[sysadmin@localhost ~]$ ls
Desktop    Downloads  Pictures  sample.txt  Videos
Documents  Music      Public    Templates
[sysadmin@localhost ~]$ ▮
```
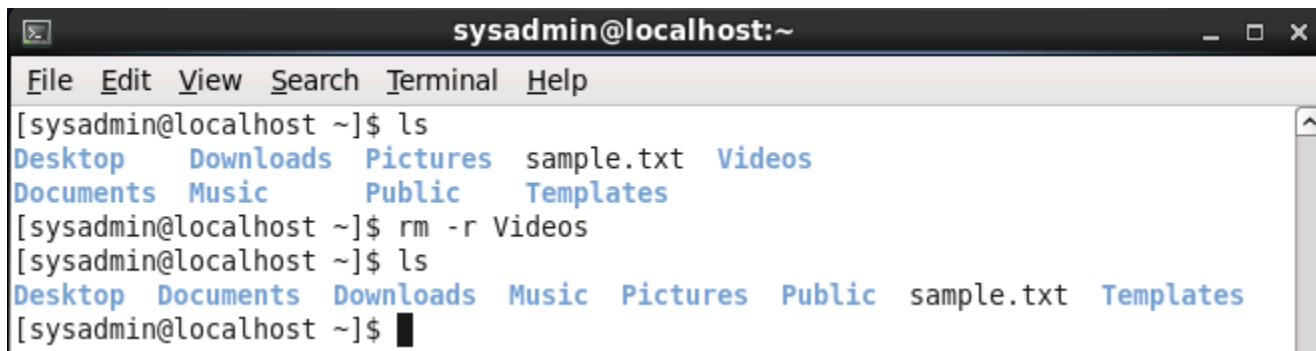
6.10 Removing Directories

You can delete directories using the rm command. However, the default usage (no options) of the rm command will fail to delete a directory:

```
sysadmin@localhost:~
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ rm Videos
rm: cannot remove `Videos': Is a directory
[sysadmin@localhost ~]$
```

If you want to delete a directory, use the -r option to the rm command:
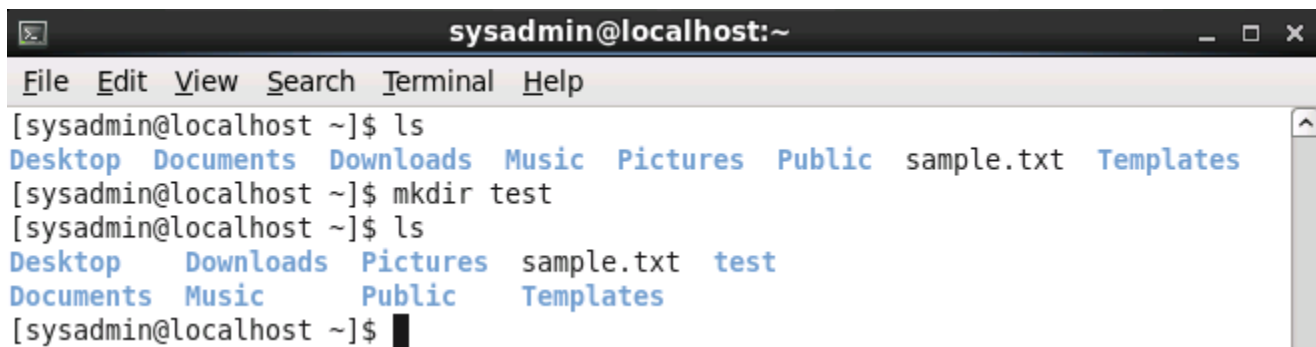
```
sysadmin@localhost:~
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls
Desktop     Downloads  Pictures   sample.txt  Videos
Documents   Music      Public     Templates
[sysadmin@localhost ~]$ rm -r Videos
[sysadmin@localhost ~]$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  sample.txt  Templates
[sysadmin@localhost ~]$
```

When a user deletes a directory, all of the files and subdirectories are deleted without any interactive question. It is best to use the -i option with the rm command.

You can also delete a directory with the rmdir command, but only if the directory is empty.

6.11 Making Directories

To create a directory, use the mkdir command:

```
sysadmin@localhost:~
File  Edit  View  Search  Terminal  Help
[sysadmin@localhost ~]$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  sample.txt  Templates
[sysadmin@localhost ~]$ mkdir test
[sysadmin@localhost ~]$ ls
Desktop     Downloads  Pictures   sample.txt  test
Documents   Music      Public     Templates
[sysadmin@localhost ~]$
```