**Linux Unhatched**

Arguments

command [options…] **[arguments…]**

An argument can be used to specify something for the command to act upon. The ls command can be given the name of a directory as an argument, and it will list the contents of that directory. In the next example, the Documents directory will be used as an argument:

**sysadmin@localhost:~$** ls Documents

School alpha-second.txt food.txt linux.txt os.csv

Work alpha-third.txt hello.sh longfile.txt people.csv

adjectives.txt alpha.txt hidden.txt newhome.txt profile.txt

alpha-first.txt animals.txt letters.txt numbers.txt red.txt

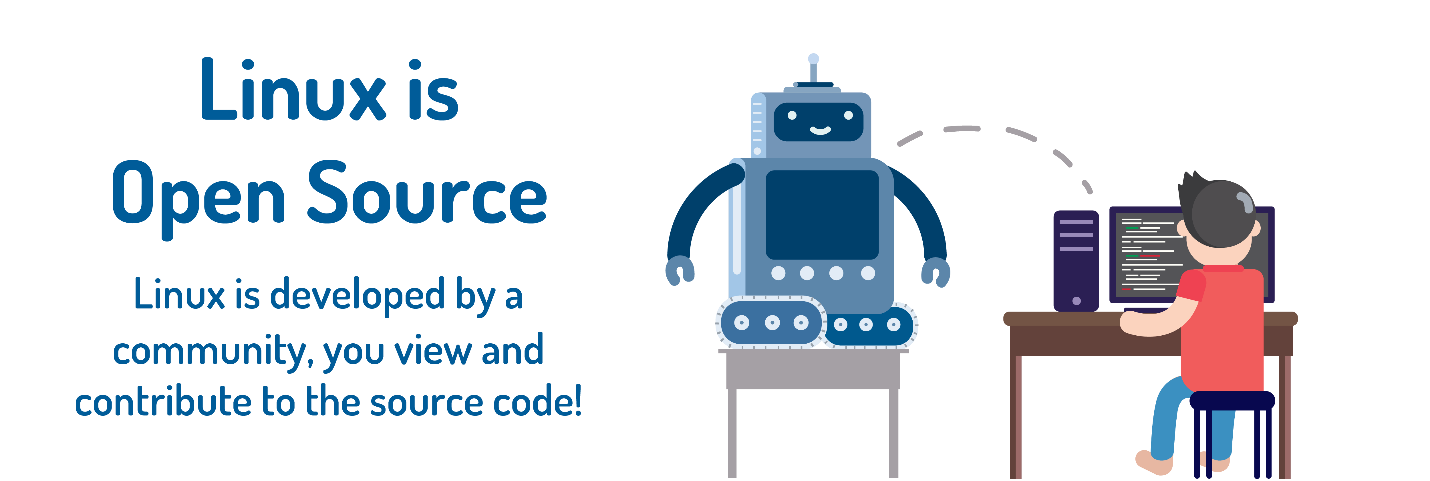
The resulting output is a list of files contained with the Documents directory.

Because Linux is open source, there are some interesting secrets that have been added by developers. For example, the aptitude command is a package management tool available on some Linux distributions. This command will accept moo as an argument:

**sysadmin@localhost:~$** aptitude moo

There are no Easter Eggs in this program.

There is more to this trick than meets the eye, keep reading!



Options

command **[options…]** [arguments…]

Options can be used to alter the behavior of a command. On the previous page, the ls command was used to list the contents of a directory. In the following example, the -l option is provided to the ls command, which results in a "long display" output, meaning the output gives more information about each of the files listed:

**sysadmin@localhost:~$** ls -l

total 32

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Desktop

drwxr-xr-x 4 sysadmin sysadmin 4096 Aug 4 20:58 Documents

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Downloads

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Music

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Pictures

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Public

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Templates

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Videos

Often the character is chosen to be mnemonic for its purpose, like choosing the letter *l* for *long* or *r* for *reverse*. By default, the ls command prints the results in alphabetical order, so adding the -r option will print the results in reverse alphabetical order.

**sysadmin@localhost:~$** ls -r

Videos Templates Public Pictures Music Downloads Documents Desktop

Multiple options can be used at once, either given as separate options as in -l -r or combined like -lr. The output of all of these examples would be the same:

ls -l -r

ls -rl

ls -lr

As explained above, -l gives a long listing format while -r reverses the listing. The result of using both options is a long listing given in reverse order:

**sysadmin@localhost:~$** ls -l -r

total 32

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Videos

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Templates

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Public

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Pictures

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Music

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Downloads

drwxr-xr-x 4 sysadmin sysadmin 4096 Aug 4 20:58 Documents

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Desktop

**sysadmin@localhost:~$** ls -rl

total 32

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Videos

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Templates

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Public

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Pictures

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Music

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Downloads

drwxr-xr-x 4 sysadmin sysadmin 4096 Aug 4 20:58 Documents

drwxr-xr-x 2 sysadmin sysadmin 4096 Aug 4 20:58 Desktop

Ultimately, commands can use many combinations of options and arguments. The possibilities for each command will be unique. Remember the aptitude easter egg?

**sysadmin@localhost:~$** aptitude moo

There are no Easter Eggs in this program.

It is possible to alter the behavior of this command using options. See what happens when the -v (verbose) option is added:

**sysadmin@localhost:~$** aptitude -v moo

There really are no Easter Eggs in this program.

By combining multiple -v options, we can get a variety of responses:

**sysadmin@localhost:~$** aptitude -vv moo

Didn't I already tell you that there are no Easter Eggs in this program?

**sysadmin@localhost:~$** aptitude -vvv moo

Stop it!

Remember multiple options can be denoted separately or combined:

aptitude -v -v moo

aptitude -vv moo

Keep adding -v options to see how many unique responses you can get!



## Printing Working Directory

In order to discover where you are currently located within the filesystem, the pwd command can be used. The pwd command prints the working directory, your current location within the filesystem:

pwd [OPTIONS]

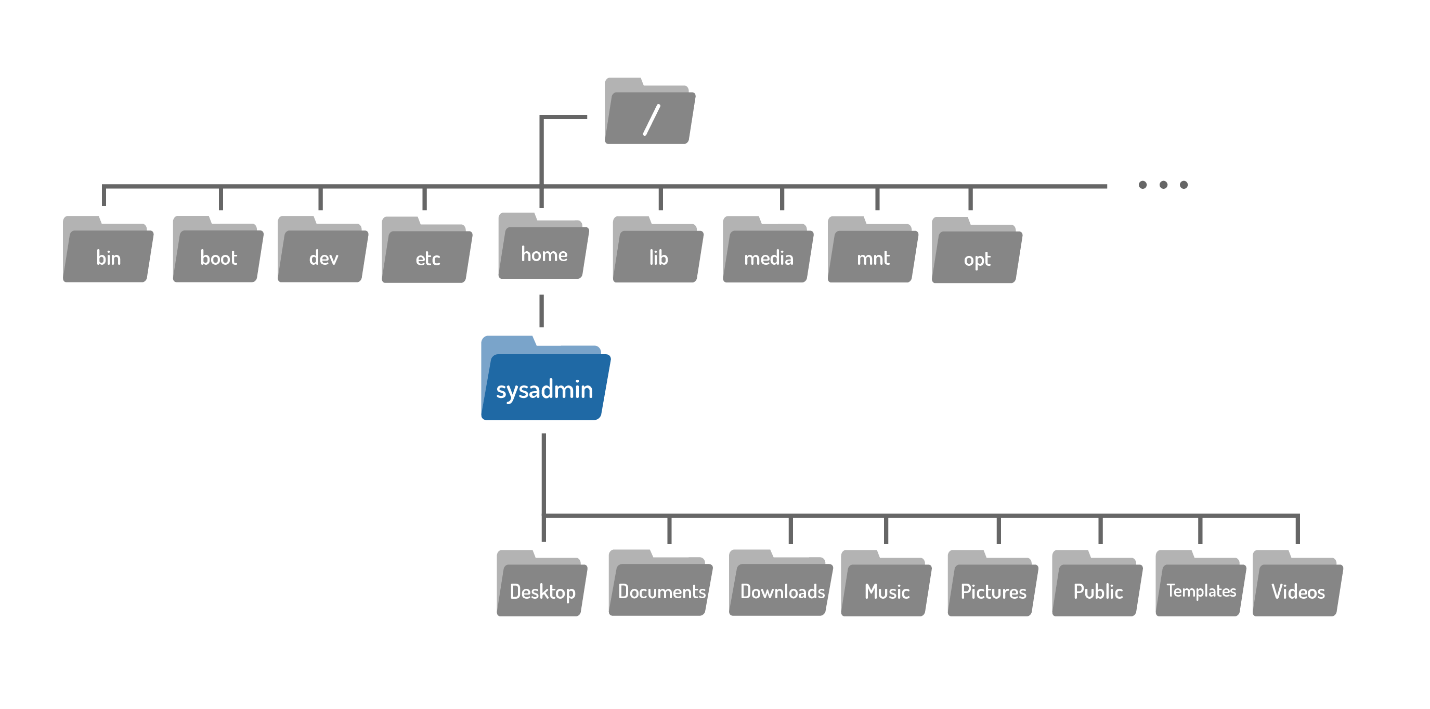
**Consider This**

Don't turn on your printer just yet! In the early days of computing the command line output would be sent to physical printers. This method was replaced by video displays which could display information more quickly. We still use the word print even though the output is just being displayed on your screen.

**sysadmin@localhost:~$** pwd

/home/sysadmin

The output of the above command indicates that the user is currently in their home folder, shown in the filesystem below.



**Consider This**

Notice our virtual machines employ a prompt that displays the current working directory, emphasized with the color blue. In the first prompt above, the blue ~ is equivalent to /home/sysadmin, representing the user's home directory.

**sysadmin@localhost:~$**

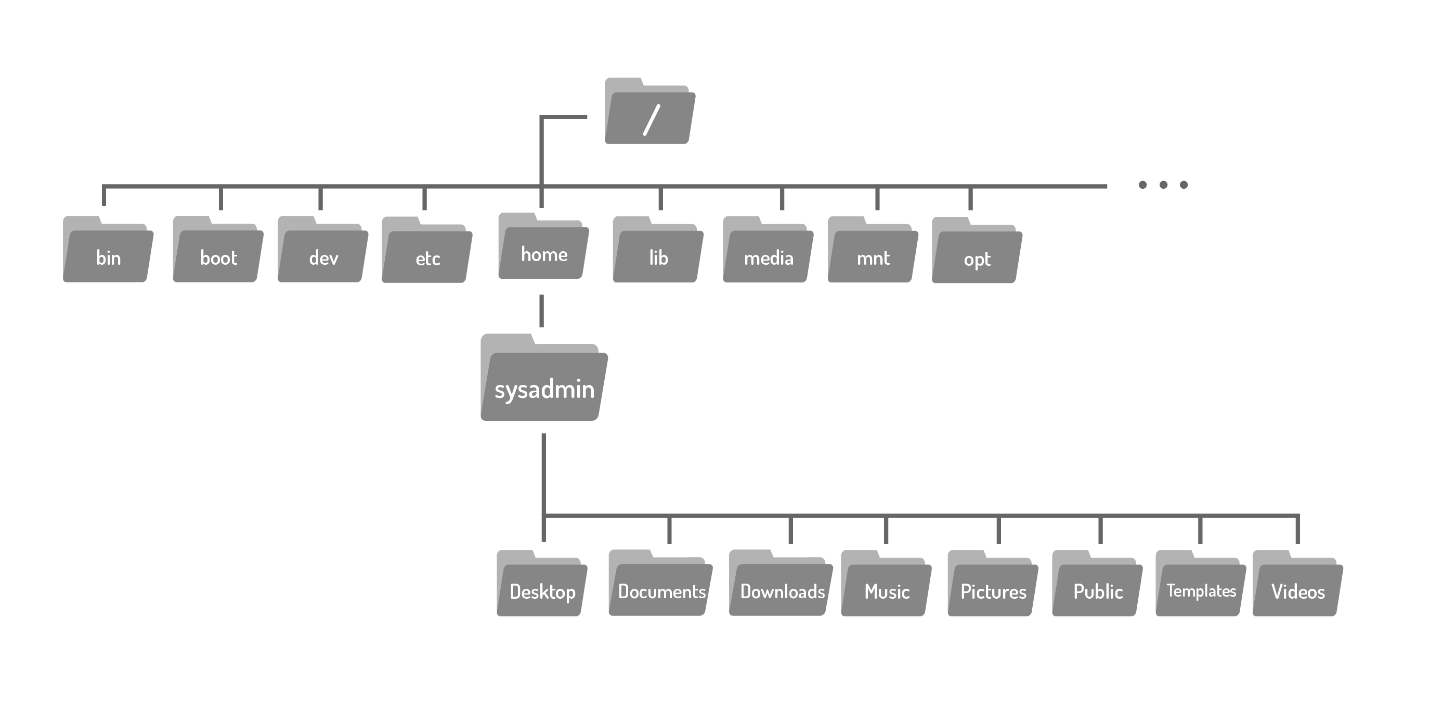
After changing directories (we will learn how to do this in the next section), the new location can also be confirmed in the new prompt, again shown in blue.

**sysadmin@localhost:/etc/calendar$**

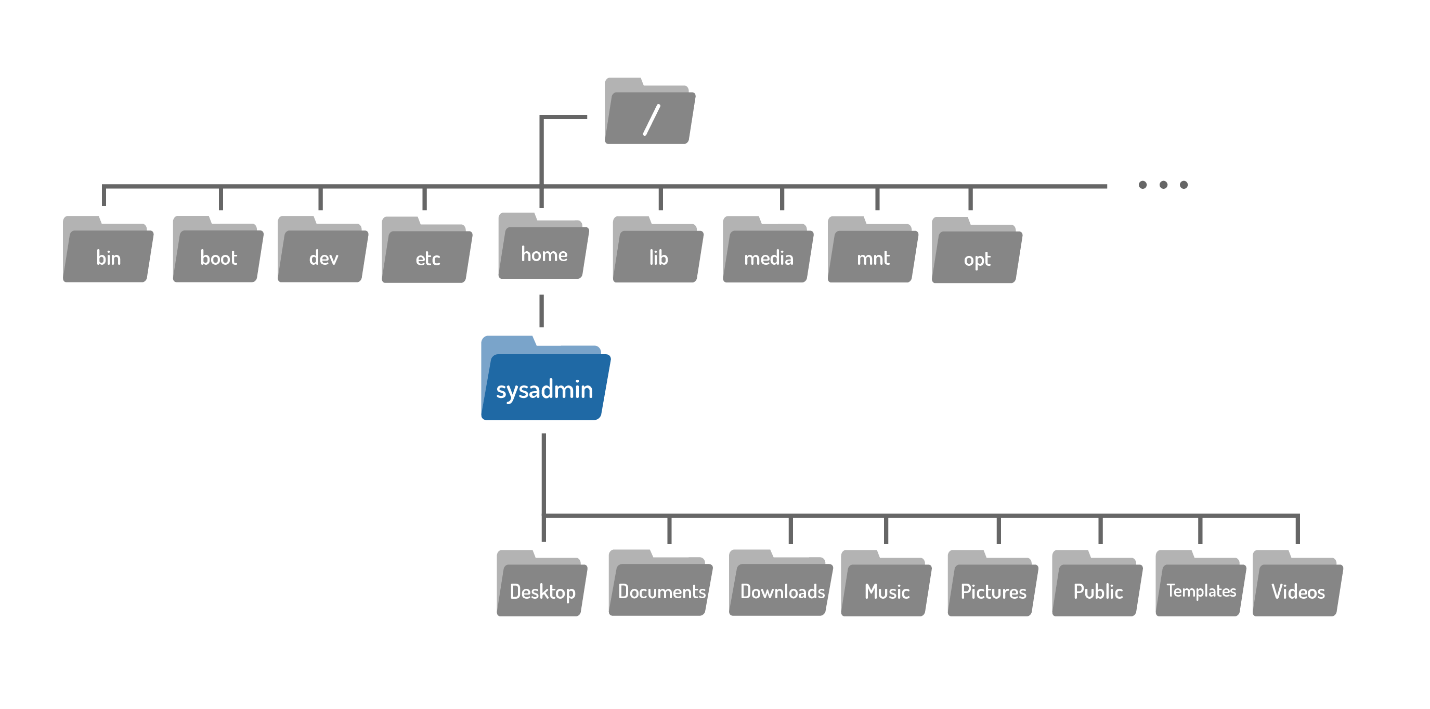


## Changing Directories

Files are used to store data such as text, graphics and programs. Directories are a type of file used to store other files–they provide a hierarchical organizational structure. The image below shows an abbreviated version of the filesystem structure on the virtual machines.



When you start a fresh virtual machine, either by opening the course or after using the reset button, you are logged in as the sysadmin user in your home directory, highlighted below:



To navigate the filesystem structure, use the cd (change directory) command to change directories.

cd [options] [path]

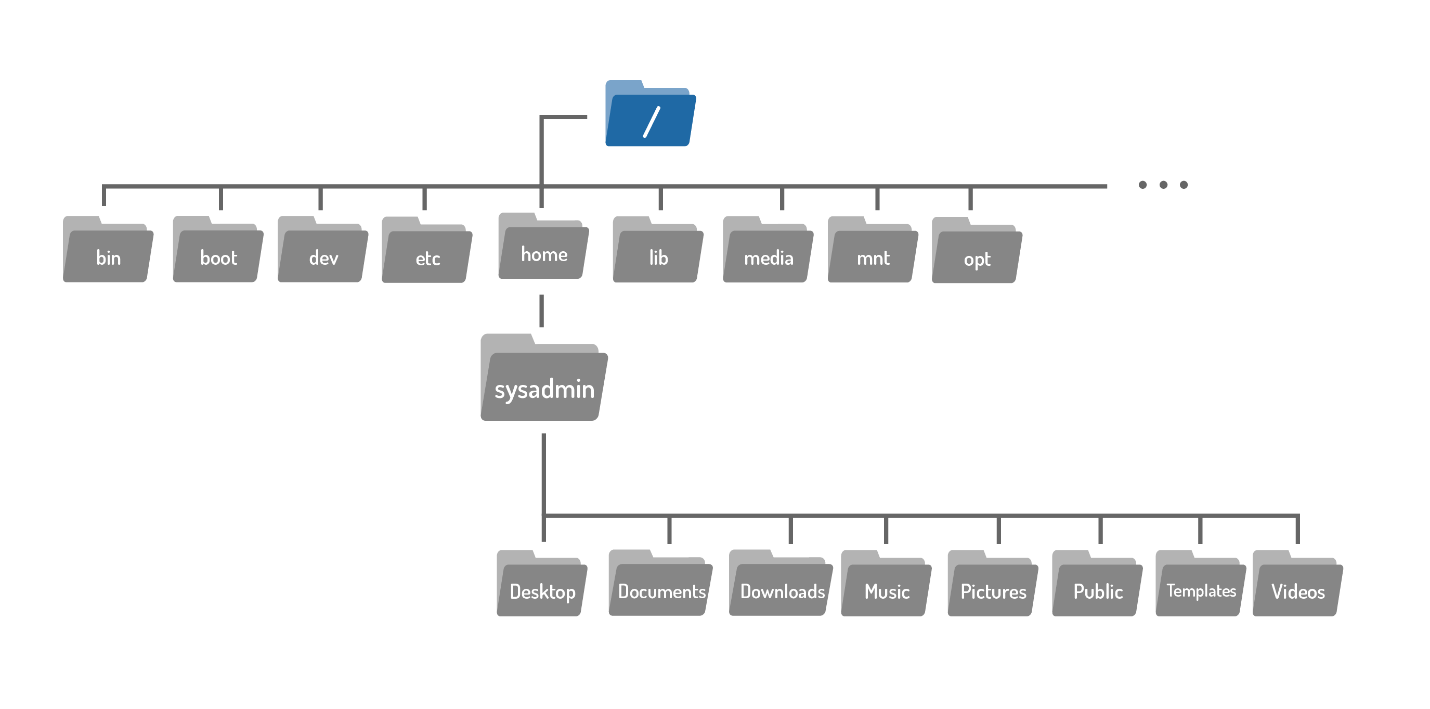
If you look back at the graphic above, you will see the Documents directory is located within the home directory, where you are currently located. To move to the Documents directory, use it as argument to the cd command:

**sysadmin@localhost:~$** cd Documents

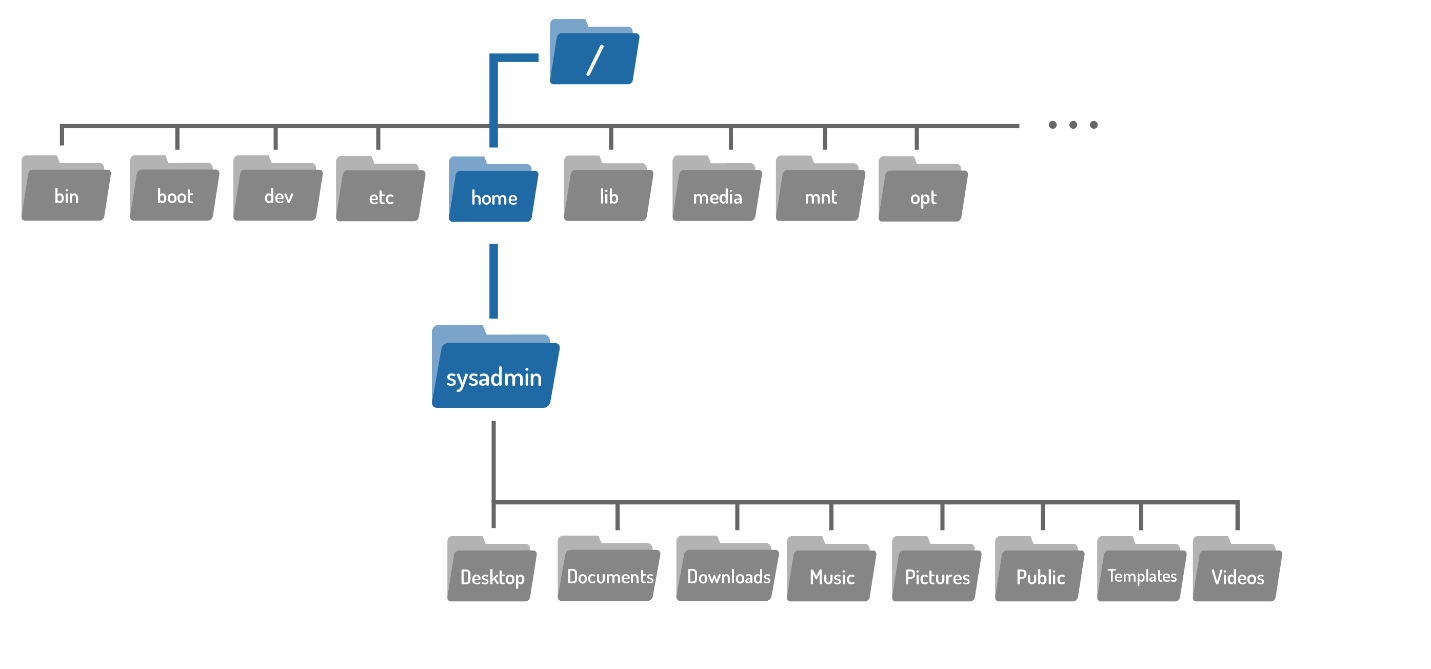
**sysadmin@localhost:~/Documents$**

Directories are equivalent to folders on Windows and Mac OS. Like these more popular operating systems, a Linux directory structure has a top level. It is not called "My Computer", but rather the root directory and it is represented by the / character. To move to the root directory, use the / character as the argument to the cd command.

**sysadmin@localhost:~$** cd /



The argument to the cd command is more than just the name of a directory, it is actually a path. A path is a list of directories separated by the / character. For example, /home/sysadmin is the path to your home directory:



If you think of the filesystem as a map, paths are the step-by-step directions; they can be used to indicate the location of any file within the filesystem. There are two types of paths: absolute and relative. Absolute paths start at the root of the filesystem, relative paths start from your current location.

### **Absolute Paths**

An absolute path allows you to specify the exact location of a directory. It always starts at the root directory, therefore it always begins with the / character. The path to the home directory /home/sysadminis an absolute path. The path begins at the root / directory, moves into the home directory, and then into the sysadmin directory. Following this path on a graphical user interface (GUI) like your home computer would look something like this:



Use this path as an argument to the cd command to move back into the home directory for the sysadmin user.

**sysadmin@localhost:/$** cd /home/sysadmin

**sysadmin@localhost:~$**

No output means the command succeeded. Go ahead and confirm this using the pwd command:

**sysadmin@localhost:~$** pwd

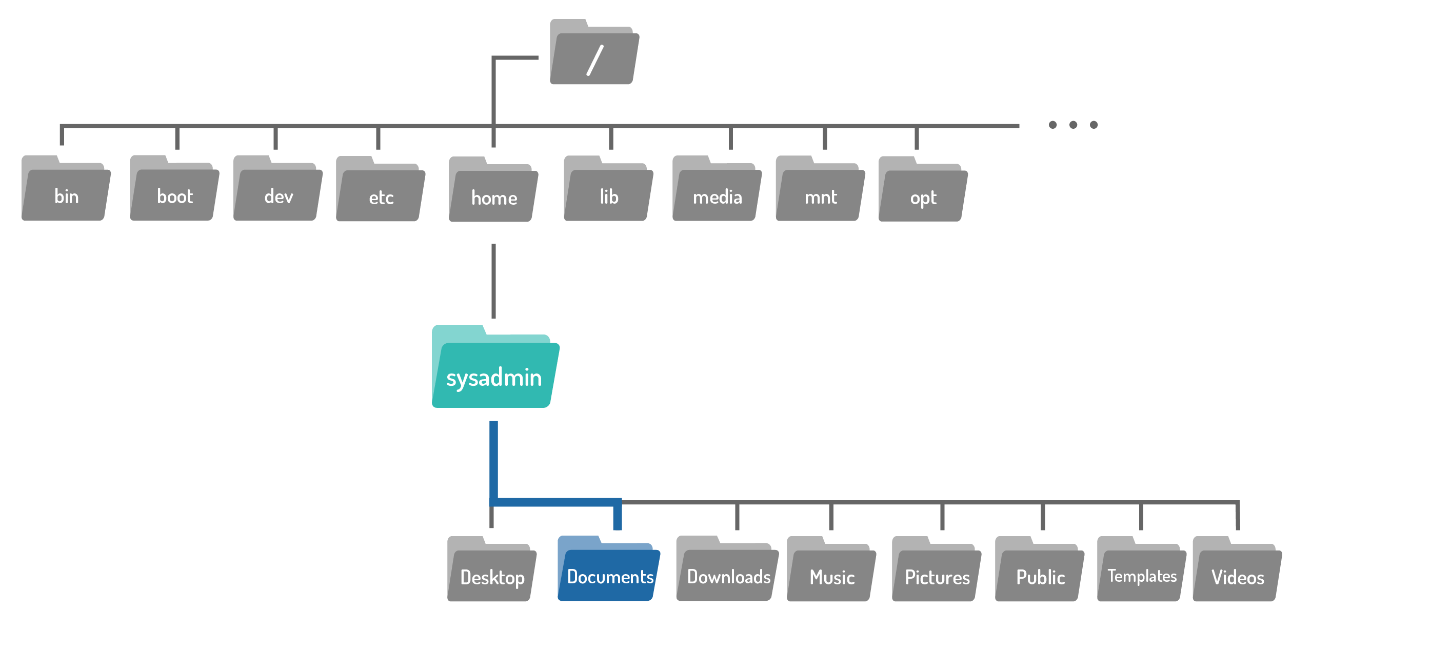
/home/sysadmin

### **Relative Paths**

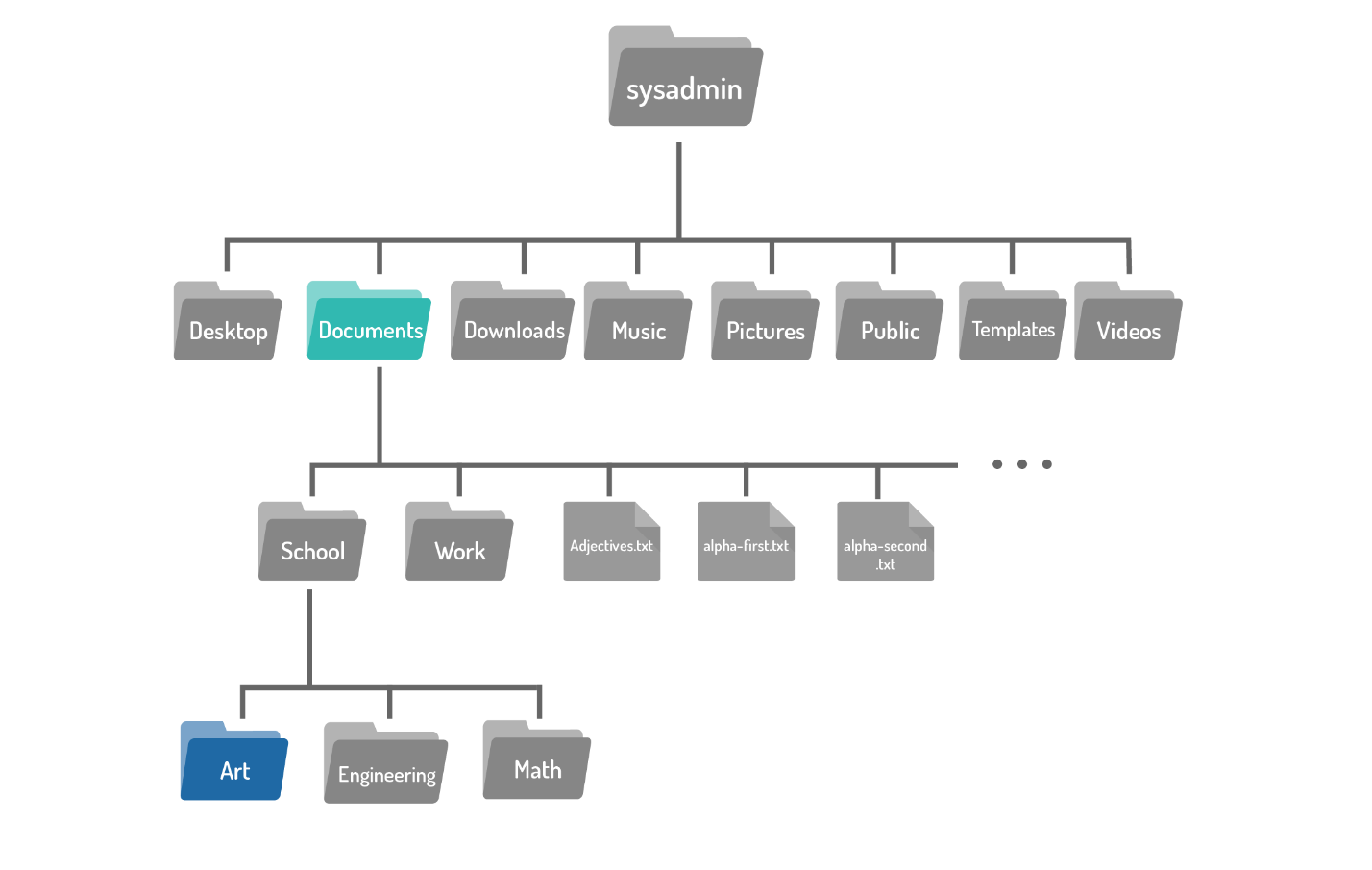
A relative path gives directions to a file relative to your current location in the filesystem. Relative paths do not start with the / character, they start with the name of a directory. Take another look at the first cd command example. The argument is an example of the simplest relative path: the name of a directory in your current location.

**sysadmin@localhost:~$** cd Documents

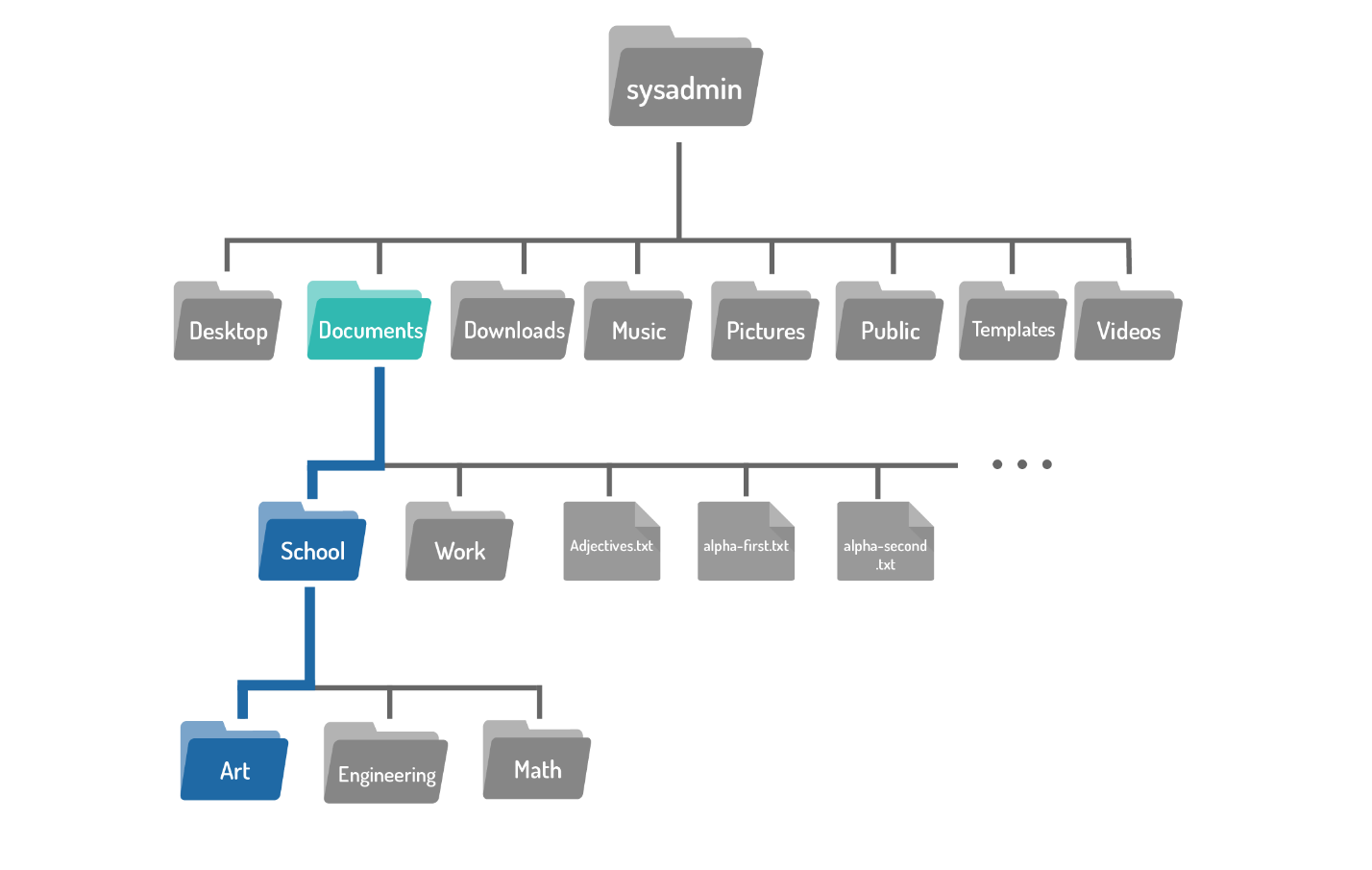
**sysadmin@localhost:~/Documents$**



The image below shows a map of the files contained within the sysadmin directory. You are currently in the Documents directory and want to move to the Art directory:



A relative path begins in from with the current directory, however you don't include it in the path. The first step would be to move into the School directory, and then move into the Art directory. Use the / character to separate the directory names and the result School/Art is a relative path from the Documents directory to the Art directory:



Use the relative path as an argument to the cd command to move into the Art directory.

**sysadmin@localhost:~/Documents/$** cd School/Art

**sysadmin@localhost:~/Documents/School/Art$**

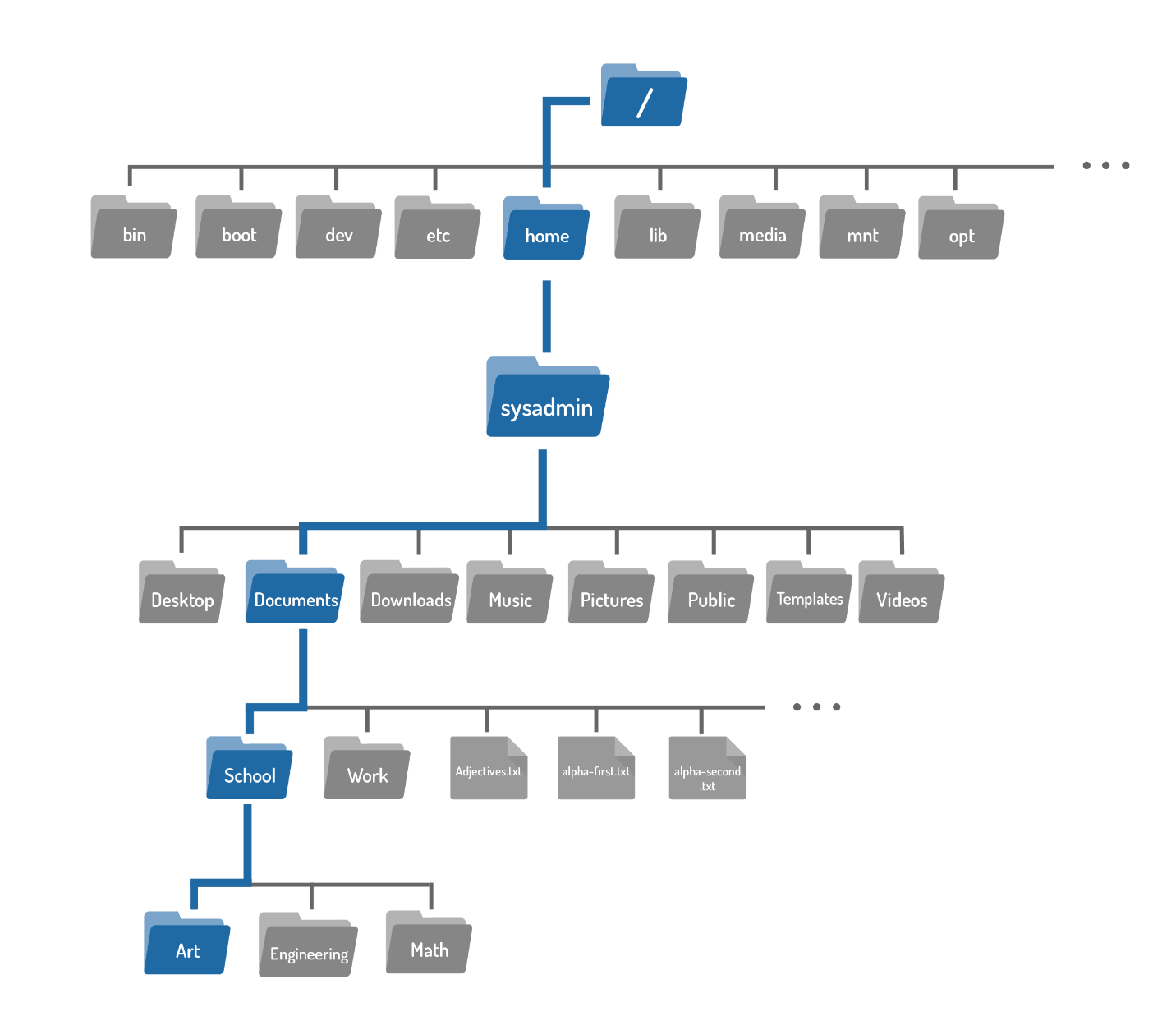
Use the pwd command to confirm the change:

**sysadmin@localhost:~/Documents/School/Art$** pwd

/home/sysadmin/Documents/School/Art

**Consider This**

The output of the pwd command is the absolute path to the Art directory.



**Consider This**

In the example above the cd command followed the School/Art path:

cd School/Art

A path can also be broken down into multiple cd commands. The following set of commands would achieve the same results:

cd School

cd Art

### **Shortcuts**

**The .. Characters**

Regardless of which directory you are in, .. always represents one directory higher relative to the current directory, sometimes referred to as the parent directory. To move from the Art directory back to the School directory:

**sysadmin@localhost:~/Documents/School/Art$** cd ..

**sysadmin@localhost:~/Documents/School$**

**The . Character**

Regardless of which directory you are in, the . character always represents your current directory. For the cd this shortcut is not very useful, but it will come in handy for commands covered in subsequent sections.

**The ~ Character**

The home directory of the current user is represented by the ~ character. As stated above, you always begin as the sysadmin user, whose home is located at /home/sysadmin. To return to your home directory at any time execute the following command:

**sysadmin@localhost:~/Documents/School$** cd ~



## Listing Files

The ls command is used to list the contents of a directory. You've already seen it used a few times before in examples, but this page will help ensure you are comfortable with its use.

ls [OPTIONS] [FILE]

By default, when the ls command is used with no options or arguments, it will list the files in the current directory:

**sysadmin@localhost:~$** ls

Desktop Documents Downloads Music Pictures Public Templates Videos

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

**sysadmin@localhost:~$** ls -l /var/log/

total 832

-rw-r--r-- 1 root root 17869 Mar 14 17:48 alternatives.log

drwxr-x--- 2 root adm 4096 Mar 14 17:48 apache2

drwxr-xr-x 2 root root 4096 Mar 14 17:45 apt

-rw-r----- 1 syslog adm 380 Jul 28 03:45 auth.log

-rw-r--r-- 5 root root 47816 Mar 2 23:10 bootstrap.log

-rw-rw---- 5 root utmp 0 Mar 2 23:10 btmp

-rw-r----- 1 syslog adm 324 Jul 28 03:45 cron.log

-rw-r----- 1 root adm 85083 Mar 14 17:48 dmesg

-rw-r--r-- 1 root root 315196 Mar 14 17:48 dpkg.log

-rw-r--r-- 1 root root 32064 Mar 14 17:48 faillog

drwxr-xr-x 2 root root 4096 Jun 30 06:53 fsck

-rw-r----- 1 syslog adm 106 Jul 28 03:45 kern.log

-rw-rw-r-- 1 root utmp 292584 Jul 28 03:45 lastlog

-rw-r----- 1 syslog adm 18703 Jul 28 03:46 syslog

drwxr-xr-x 2 root root 4096 Apr 11 2014 upstart

-rw-rw-r-- 1 root utmp 384 Jul 28 03:45 wtmp

Each line corresponds to a file contained within the directory. The information can be broken down into fields separated by spaces. The fields are as follows:

* **File Type**
* -rw-r--r-- 1 root root 17869 Mar 14 17:48 alternatives.log
* drwxr-x--- 2 root adm 4096 Mar 14 17:48 apache2

The first field actually contains ten characters, where the first character indicates the type of file and the next nine specify permissions. The file types are:

| **Symbol** | **File Type** | **Description** |
| --- | --- | --- |
| d | directory | A file used to store other files. |
| - | regular file | Includes readable files, images 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. |

The first file alternatives.log is a regular file -, while the second file apache2 is a directory d.

**Permissions**

drwxr-xr-x 1 root root 0 Apr 11 21:58 upstart

Permissions indicate how certain users can access a file. Keep reading to learn more about permissions.

* **Hard Link Count**

-rw-r----- 1 syslog adm 23621 Aug 23 15:17 auth.log

This number indicates how many hard links point to this file. Hard links are beyond the scope of this module, but are covered in the [NDG Linux Essentials](https://www.netacad.com/courses/linux-essentials/) course.

* **User Owner**

-rw-r----- 1 syslog adm 416 Aug 22 15:43 kern.log

User syslog owns this file. Every time a file is created, the ownership is automatically assigned to the user who created it.

* **Group Owner**

-rw-rw-r-- 1 root utmp 292584 Aug 20 18:44 lastlog

Indicates which group owns this file

* **File Size**

-rw-r----- 1 syslog adm 1087150 Aug 23 15:17 syslog.1

Directories and larger files may be shown in kilobytes since displaying their size in bytes would present a very large number. Therefore, in the case of a directory, it might actually be a multiple of the block size used for the file system. Block size is the size of a series of data stored in the filesystem.

* **Timestamp**

drwxr-xr-x 1 root root 32 Jul 17 03:36 fsck

This indicates the time that the file's contents were last modified.

* **Filename**

-rw-r--r-- 1 root root 47816 Jul 17 03:36 bootstrap.log

The final field contains the name of the file or directory.

**Consider This**

In the case of symbolic links, a file that points to another file, the link name will be displayed along with an arrow and the pathname of the original file.

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

Symbolic links are beyond the scope of this module, but are covered in the NDC LINUX ESSENTIAL  course.

### **Sorting**

By default the output of the ls command is sorted alphabetically by filename. It can sort by other methods as well.

**Follow Along**

The options in examples below will be combined with the -l option so the relevant details of the files are displayed. Notice fields corresponding to the search option.

The -t option will sort the files by timestamp:

**sysadmin@localhost:~$** ls -lt /var/log

total 840

-rw-r----- 1 syslog adm 27014 Jul 28 00:10 syslog

-rw-r----- 1 syslog adm 380 Jul 27 23:10 auth.log

-rw-rw-r-- 1 root utmp 292584 Jul 27 23:10 lastlog

-rw-rw-r-- 1 root utmp 384 Jul 27 23:10 wtmp

-rw-r----- 1 syslog adm 324 Jul 27 23:10 cron.log

-rw-r----- 1 syslog adm 106 Jul 27 23:10 kern.log

drwxr-xr-x 2 root root 4096 Jun 30 06:56 fsck

-rw-r--r-- 1 root root 17869 Mar 14 17:48 alternatives.log

-rw-r----- 1 root adm 85083 Mar 14 17:48 dmesg

-rw-r--r-- 1 root root 32064 Mar 14 17:48 faillog

-rw-r--r-- 1 root root 315196 Mar 14 17:48 dpkg.log

drwxr-x--- 2 root adm 4096 Mar 14 17:48 apache2

drwxr-xr-x 2 root root 4096 Mar 14 17:45 apt

-rw-r--r-- 5 root root 47816 Mar 2 23:10 bootstrap.log

-rw-rw---- 5 root utmp 0 Mar 2 23:10 btmp

drwxr-xr-x 2 root root 4096 Apr 11 2014 upstart

The -S option will sort the files by file size:

**sysadmin@localhost:~$** ls -l -S /var/log

total 840

-rw-r--r-- 1 root root 315196 Mar 14 17:48 dpkg.log

-rw-rw-r-- 1 root utmp 292584 Jul 27 23:10 lastlog

-rw-r----- 1 root adm 85083 Mar 14 17:48 dmesg

-rw-r--r-- 5 root root 47816 Mar 2 23:10 bootstrap.log

-rw-r--r-- 1 root root 32064 Mar 14 17:48 faillog

-rw-r----- 1 syslog adm 27014 Jul 28 00:10 syslog

-rw-r--r-- 1 root root 17869 Mar 14 17:48 alternatives.log

drwxr-x--- 2 root adm 4096 Mar 14 17:48 apache2

drwxr-xr-x 2 root root 4096 Mar 14 17:45 apt

drwxr-xr-x 2 root root 4096 Jun 30 06:56 fsck

drwxr-xr-x 2 root root 4096 Apr 11 2014 upstart

-rw-rw-r-- 1 root utmp 384 Jul 27 23:10 wtmp

-rw-r----- 1 syslog adm 380 Jul 27 23:10 auth.log

-rw-r----- 1 syslog adm 324 Jul 27 23:10 cron.log

-rw-r----- 1 syslog adm 106 Jul 27 23:10 kern.log

-rw-rw---- 5 root utmp 0 Mar 2 23:10 btmp

The -r option will reverse the order of any type of sort. Notice the difference when it is added to the previous example:

**sysadmin@localhost:~$** ls -lSr /var/log

total 840

-rw-rw---- 5 root utmp 0 Mar 2 23:10 btmp

-rw-r----- 1 syslog adm 106 Jul 27 23:10 kern.log

-rw-r----- 1 syslog adm 324 Jul 27 23:10 cron.log

-rw-r----- 1 syslog adm 380 Jul 27 23:10 auth.log

-rw-rw-r-- 1 root utmp 384 Jul 27 23:10 wtmp

drwxr-xr-x 2 root root 4096 Apr 11 2014 upstart

drwxr-xr-x 2 root root 4096 Jun 30 06:56 fsck

drwxr-xr-x 2 root root 4096 Mar 14 17:45 apt

drwxr-x--- 2 root adm 4096 Mar 14 17:48 apache2

-rw-r--r-- 1 root root 17869 Mar 14 17:48 alternatives.log

-rw-r----- 1 syslog adm 27014 Jul 28 00:10 syslog

-rw-r--r-- 1 root root 32064 Mar 14 17:48 faillog

-rw-r--r-- 5 root root 47816 Mar 2 23:10 bootstrap.log

-rw-r----- 1 root adm 85083 Mar 14 17:48 dmesg

-rw-rw-r-- 1 root utmp 292584 Jul 27 23:10 lastlog

-rw-r--r-- 1 root root 315196 Mar 14 17:48 dpkg.log

The numbers in file size field switch from descending to ascending.

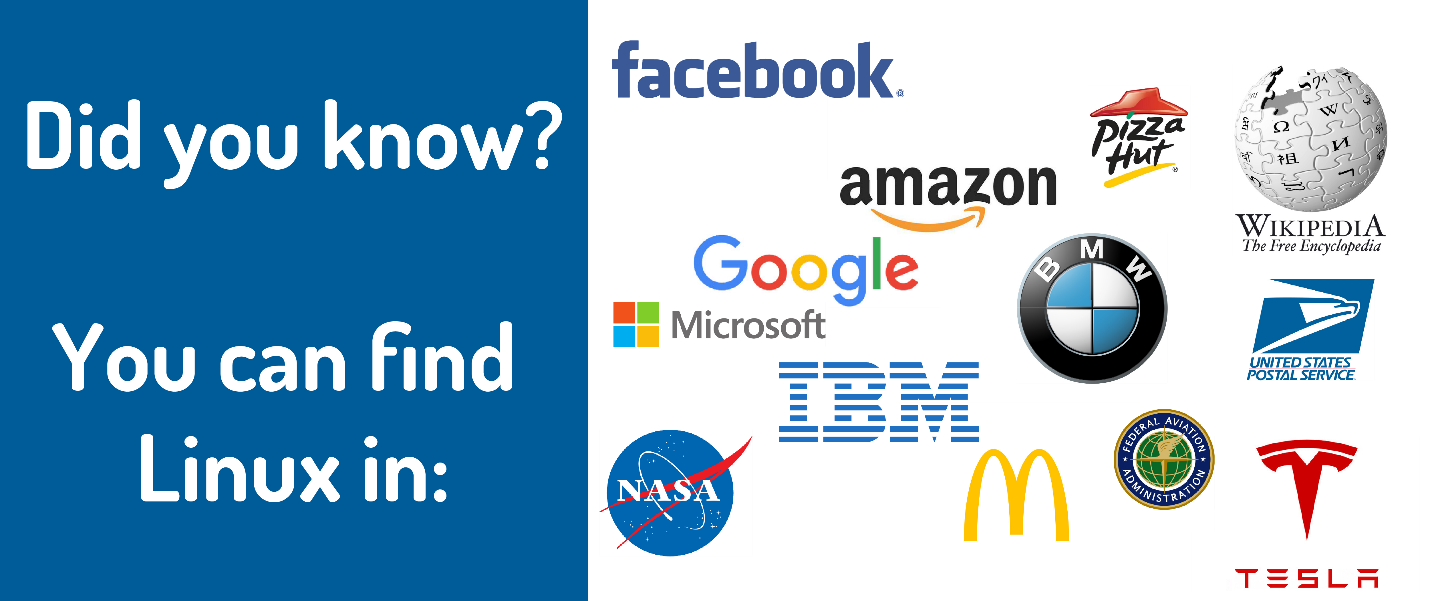
Used alone the -r option with list the files in reverse alphabetical order:

**sysadmin@localhost:~$** ls -r /var/log

wtmp lastlog faillog cron.log auth.log alternatives.log

upstart kern.log dpkg.log btmp apt

syslog fsck dmesg bootstrap.log apache2





## Administrative Access

There are many Linux commands which deal with sensitive information like passwords, system hardware, or otherwise operate under other exceptional circumstances. Preventing regular users from executing these commands helps to protect the system. Logging in as the root user provides administrative access, allowing for the execution of some of the privileged commands.

### **The su Command**

su OPTIONS USERNAME

The su command allows you to temporarily act as a different user. It does this by creating a new shell. The shell is simply a text input console that lets you type in commands. By default, if a user account is not specified, the su command will open a new shell as the root user, which provides administrative privileges.

**Follow Along**

Utilizing the login shell option is recommended, as the login shell fully configures the new shell with the settings of the new user. This option can be specified one of three ways:

su -

su -l

su --login

After executing the su command, a password is required. On our virtual machines, the password for both the root and sysadmin accounts is netlab123. If you ever forget, it is displayed every time a new virtual machine is started. As a security measure, the password will not be visible as it is typed.

**sysadmin@localhost:~$** su -

Password:

**root@localhost:~#**

Note the command prompt has changed to reflect that you are now logged in as the root user. To logout and return to the sysadmin account, use the exit command. Note the prompt changes back:

**root@localhost:~#** exit

exit

**sysadmin@localhost:~$**

To avoid executing any sensitive commands, we’ve configured the Steam Locomotive command, the sl command, to require administrative access. If the command is executed as sysadmin, it fails:

**sysadmin@localhost:~$** sl

sl: Permission denied

Use the su command to switch to the root account and execute the sl command with administrative access:

**sysadmin@localhost:~$** su -

Password:

**root@localhost:~#** sl

(@@) ( ) (@) ( ) @@ () @ O @

( )

(@@@@)

( )

(@@@)

==== \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

\_D \_| |\_\_\_\_\_\_\_/ \\_\_I\_I\_\_\_\_\_===\_\_|\_\_\_\_\_\_\_\_\_|

|(\_)--- | H\\_\_\_\_\_\_\_\_/ | | =|\_\_\_ \_\_\_| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/ | | H | | | | ||\_| |\_|| \_|

| | | H |\_\_--------------------| [\_\_\_] | =|

| \_\_\_\_\_\_\_\_|\_\_\_H\_\_/\_\_|\_\_\_\_\_/[][]~\\_\_\_\_\_\_\_| | -|

|/ | |-----------I\_\_\_\_\_I [][] [] D |=======|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_/ =| o |=-~~\ /~~\ /~~\ /~~\ \_\_\_\_Y\_\_\_\_\_\_\_\_\_\_\_|\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|/-=|\_\_\_|= || || || |\_\_\_\_\_/~\\_\_\_/ |\_D\_\_D\_\_D\_| |\_D\_\_

\\_/ \\_O=====O=====O=====O/ \\_/ \\_/ \\_/ \\_/

Use the exit command again to return to the sysadmin account.

**root@localhost:~#** exit

exit

**sysadmin@localhost:~$**

### **The sudo Command**

sudo [OPTIONS] COMMAND

The sudo command allows a user to execute a command as another user without creating a new shell. Instead, to execute a command with administrative privileges, use it as an argument to the sudo command. Like the su command, the sudo command assumes by default the root user account should be used to execute commands.

**Consider This**

The sudo command can be used to switch to other user accounts as well. To specify a different user account use the -u option.

Execute the sl command as the root user by putting sudo in front of it:

**sysadmin@localhost:~$** sudo sl

Password:

**Note**

Remember the password is netlab123. The prompt for the password will not appear again as long as the user continues to execute sudo commands less than five minutes apart.

Once the command has completed, notice the prompt has not changed, you are still logged in as sysadmin. The sudo command only provides administrative access for the execution of the specified command. This is an advantage as it reduces the risk that a user accidentally executes a command as root. The intention to execute a command is clear; the command is executed as root if prefixed with the sudo command. Otherwise, the command is executed as a regular user.



## Permissions

Permissions determine the ways different users can interact with a file or directory. When listing a file with the ls -l command, the output includes permission information. For the example we will use a script called hello.sh located in the Documents directory:

**Follow Along**

Use the following command to switch to the Documents directory:

**sysadmin@localhost:~$** cd ~/Documents

**sysadmin@localhost:~/Documents$** ls -l hello.sh

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

Below is a review of the fields relevant to permissions.

### **File Type Field**

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

The first character of this output indicates the type of a file. Recall that if the first character is a -, this is a regular file. If the character was a d, it would be a directory.

### **Permissions Field**

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

After the file type character, the permissions are displayed. The permissions are broken into three sets of three characters:

### **Owner**

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

The first set is for the user who owns the file. If your current account is the user owner of the file, then the first set of the three permissions will apply and the other permissions have no effect.

The user who owns the file, and who these permissions apply to, can be determined by the user owner field:

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

### **Group**

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

The second set is for the group that owns the file. If your current account is not the user owner of the file and you are a member of the group that owns the file, then the group permissions will apply and the other permissions have no effect.

The group for this file can be determined by the group owner field:

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

### **Other**

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

The last set is for everyone else, any one who that first two sets of permissions do not apply to. If you are not the user who owns the file or a member of the group that owns the file, the third set of permissions applies to you.

### **Permission Types**

There are three different permissions that can be placed on a file or directory: read, write, and execute. The manner in which these permissions apply differs for files and directories, as shown in the chart below:

| **Permission** | **Effects on File** | **Effects on Directory** |
| --- | --- | --- |
| read (r) | Allows for file contents to be read or copied. | Without execute permission on the directory, allows for a non-detailed listing of files. With execute permission, ls -l can provide a detailed listing. |
| write (w) | Allows for contents to be modified or overwritten. Allows for files to be added or removed from a directory. | For this permission to work, the directory must also have execute permission. |
| execute (x) | Allows for a file to be run as a process, although script files require read permission, as well. | Allows a user to change to the directory if parent directories have execute permission as well. |

**Consider This**

Understanding which permissions apply is an important skill set in Linux. For example, consider the following set of permissions:

-r--rw-rwx. 1 sysadmin staff 999 Apr 10 2013 /home/sysadmin/test

In this scenario, the user sysadmin ends up having less access to this file than members of the staff group or everyone else. The user sysadmin only has the permissions of r--. It doesn't matter if sysadmin is a member of the staff group; once user ownership has been established, only the user owner's permissions apply.



## Changing File Permissions

The chmod command is used to change the permissions of a file or directory. Only the root user or the user who owns the file is able to change the permissions of a file.

**Consider This**

Why is the command named chmod instead of chperm? Permissions used to be referred to as modes of access, so the command chmod really means change the modes of access.

There are two techniques for changing permissions with the chmod command: symbolic and octal. The symbolic method is good for changing one set of permissions at a time. The octal or numeric method requires knowledge of the octal value of each of the permissions and requires all three sets of permissions (user, group, other) to be specified every time. For the sake of simplicity, only the symbolic method will be covered. To learn more about the octal method check out [NDG Linux Essentials](https://www.netdevgroup.com/online/courses/ndg_linux_essentials.html)!

**Follow Along**

Use the following command to switch to the Documents directory:

**sysadmin@localhost:~$** cd ~/Documents

### **The Symbolic Method**

chmod [<SET><ACTION><PERMISSIONS>]... FILE

To use the symbolic method of chmod first indicate which set of permissions is being changed:

chmod [<SET><ACTION><PERMISSIONS>]... FILE

| **Symbol** | **Meaning** |
| --- | --- |
| u | User: The user who owns the file. |
| g | Group: The group who owns the file. |
| o | Others: Anyone other than the user owner or member of the group owner. |
| a | All: Refers to the user, group and others. |

Next, specify an action symbol:

chmod [<SET><ACTION><PERMISSIONS>]... FILE

| **Symbol** | **Meaning** |
| --- | --- |
| + | Add the permission, if necessary |
| = | Specify the exact permission |
| - | Remove the permission, if necessary |

After an action symbol, specify one or more permissions to be acted upon.

chmod [<SET><ACTION><PERMISSIONS>]... FILE

| **Symbol** | **Meaning** |
| --- | --- |
| r | read |
| w | write |
| x | execute |

Finally, a space and the pathnames for the files to assign those permissions.

chmod [<SET><ACTION><PERMISSIONS>]... FILE

The file hello.sh used in the examples on the previous page is a script. A script is a file that can be executed, similar to a command:

**sysadmin@localhost:~/Documents$** ls -l hello.sh

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

However currently, the execute permission is not set for any of the permission groups:

-rw-rw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

Attempting to execute this script using the following syntax fails:

**sysadmin@localhost:~/Documents$** ./hello.sh

-bash: ./hello.sh: Permission denied

Since the system is currently logged in as the sysadmin user, and sysadmin is the owner of this file, giving the user owner the execute permission should allow you to execute this script. Using the chmod command with the u character to represent the user owner permission set, the + character to indicate a permission is being added, and the x character to represent the execute permission, the command should be executed as follows:

**sysadmin@localhost:~/Documents$** chmod u+x hello.sh

No output indicates the command succeeded. Confirm by checking the permissions using the ls -l command:

**sysadmin@localhost:~/Documents$** ls -l hello.sh

-rwxrw-r-- 1 root sysadmin 112 Aug 1 03:48 hello.sh

The user owner now has the execute permission listed:

-rwxrw-r-- 1 sysadmin sysadmin 21 Aug 1 02:35 hello.sh

Finally, attempt to execute the script again. Use the command syntax shown below:

./hello.sh

**sysadmin@localhost:~/Documents$** ./hello.sh

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( Hello World! )

--------------

\

\

<(^)

( )

**Consider This**

Notice that to execute the script in the previous example, a ./ character combination was placed before the script name.

./hello.sh

This indicates the "command" should be run from the current directory.



## Changing File Ownership

Initially, the owner of a file is the user who creates it. The chown command is used to change the ownership of files and directories. Changing the user owner requires administrative access. A regular user cannot use this command to change the user owner of a file, even to give the ownership of one of their own files to another user. However, the chown command also permits changing group ownership, which can be accomplished by either root or the owner of the file.

To change the user owner of a file, the following syntax can be used. The first argument, [OWNER], specifies which user is to be the new owner. The second argument, FILE, specifies which file's ownership is changing.

chown [OPTIONS] [OWNER] FILE

**Follow Along**

Use the following command to switch to the Documents directory:

**sysadmin@localhost:~$** cd ~/Documents

Currently all the files in the Documents directory are owned by the sysadmin user. This can be verified by using the ls -l command. Recall that the third column indicates the user owner.

**sysadmin@localhost:~/Documents$** ls -l

total 148

drwxrwxr-x 2 sysadmin sysadmin 4096 Aug 1 03:40 School

drwxrwxr-x 2 sysadmin sysadmin 4096 Aug 1 03:40 Work

-rw-r--r-- 1 sysadmin sysadmin 39 Mar 14 17:48 adjectives.txt

-rw-r--r-- 1 sysadmin sysadmin 90 Mar 14 17:48 alpha-first.txt

-rw-r--r-- 1 sysadmin sysadmin 89 Mar 14 17:48 alpha-first.txt.original

-rw-r--r-- 1 sysadmin sysadmin 106 Mar 14 17:48 alpha-second.txt

-rw-r--r-- 1 sysadmin sysadmin 195 Mar 14 17:48 alpha-third.txt

-rw-r--r-- 1 sysadmin sysadmin 390 Mar 14 17:48 alpha.txt

-rw-r--r-- 1 sysadmin sysadmin 42 Mar 14 17:48 animals.txt

-rw-r--r-- 1 sysadmin sysadmin 14 Mar 14 17:48 food.txt

-rwxrw-r-- 1 sysadmin sysadmin 112 Aug 1 03:48 hello.sh

-rw-r--r-- 1 sysadmin sysadmin 67 Mar 14 17:48 hidden.txt

-rw-r--r-- 1 sysadmin sysadmin 10 Mar 14 17:48 letters.txt

-rw-r--r-- 1 sysadmin sysadmin 83 Mar 14 17:48 linux.txt

-rw-r--r-- 1 sysadmin sysadmin 66540 Mar 14 17:48 longfile.txt

-rw-r--r-- 1 sysadmin sysadmin 235 Mar 14 17:48 newhome.txt

-rw-r--r-- 1 sysadmin sysadmin 10 Mar 14 17:48 numbers.txt

-rw-r--r-- 1 sysadmin sysadmin 77 Mar 14 17:48 os.csv

-rw-r--r-- 1 sysadmin sysadmin 59 Mar 14 17:48 people.csv

-rw-r--r-- 1 sysadmin sysadmin 110 Mar 14 17:48 profile.txt

-rw-r--r-- 1 sysadmin sysadmin 51 Mar 14 17:48 red.txt

To switch the owner of the hello.sh script to the root user, use root as the first argument and hello.sh as the second argument. Don't forget to use the sudo command in order to gain the necessary administrative privileges. Use password netlab123 when prompted:

**sysadmin@localhost:~/Documents$** sudo chown root hello.sh

[sudo] password for sysadmin:

Confirm the user owner has changed by executing the ls -l command. Use the filename as an argument to limit the output:

**sysadmin@localhost:~/Documents$** ls -l hello.sh

-rwxrw-r-- 1 root sysadmin 112 Aug 1 03:48 hello.sh

The user owner field is now root indicating the change was successful.

**Consider This**

Try executing the hello.sh script again. It fails! Why?

**sysadmin@localhost:~/Documents$** ./hello.sh

-bash: ./hello.sh: Permission denied

Only the user owner has the execute permission, and now the root user is the user owner. This file now requires administrative access to execute. Use the sudo command to execute the script as the root user.

**sysadmin@localhost:~/Documents$** sudo ./hello.sh

[sudo] password for sysadmin:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( Hello World! )

--------------

\

\

<(^)

( )



## Moving Files

The mv command is used to move a file from one location in the filesystem to another.

mv SOURCE DESTINATION

The mv command requires at least two arguments. The first argument is the source, a path to the file to be moved. The second argument is the destination, a path to where the file will be moved to. The files to be moved are sometimes referred to as the source, and the place where the files are to be placed is called the destination.

**Follow Along**

Use the following command to switch to the Documents directory:

**sysadmin@localhost:~$** cd ~/Documents

To move the people.csv file into the Work directory, use the filename as the source, and the directory name as the destination:

**sysadmin@localhost:~/Documents$** mv people.csv Work

If a file is moved from one directory to another without specifying a new name for the file, it will retain its original name. The move above can be confirmed using the ls command on the Work directory:

**sysadmin@localhost:~/Documents$** ls Work

people.csv

The mv command is able to move multiple files, as long as the final argument provided to the command is the destination. For example, to move three files into the School directory:

**sysadmin@localhost:~/Documents$** mv numbers.txt food.txt alpha.txt School

**sysadmin@localhost:~/Documents$** ls School

Art Engineering Math alpha.txt food.txt numbers.txt

Moving a file within the same directory is an effective way to rename it. For example, in the following example the animals.txt file is given a new name of zoo.txt:

mv animals.txt zoo.txt

**sysadmin@localhost:~/Documents$** ls

School alpha-second.txt hidden.txt newhome.txt

Work alpha-third.txt letters.txt os.csv

adjectives.txt animals.txt linux.txt profile.txt

alpha-first.txt hello.sh longfile.txt red.txt

**sysadmin@localhost:~/Documents$** mv animals.txt zoo.txt

**sysadmin@localhost:~/Documents$** ls

School alpha-second.txt letters.txt os.csv

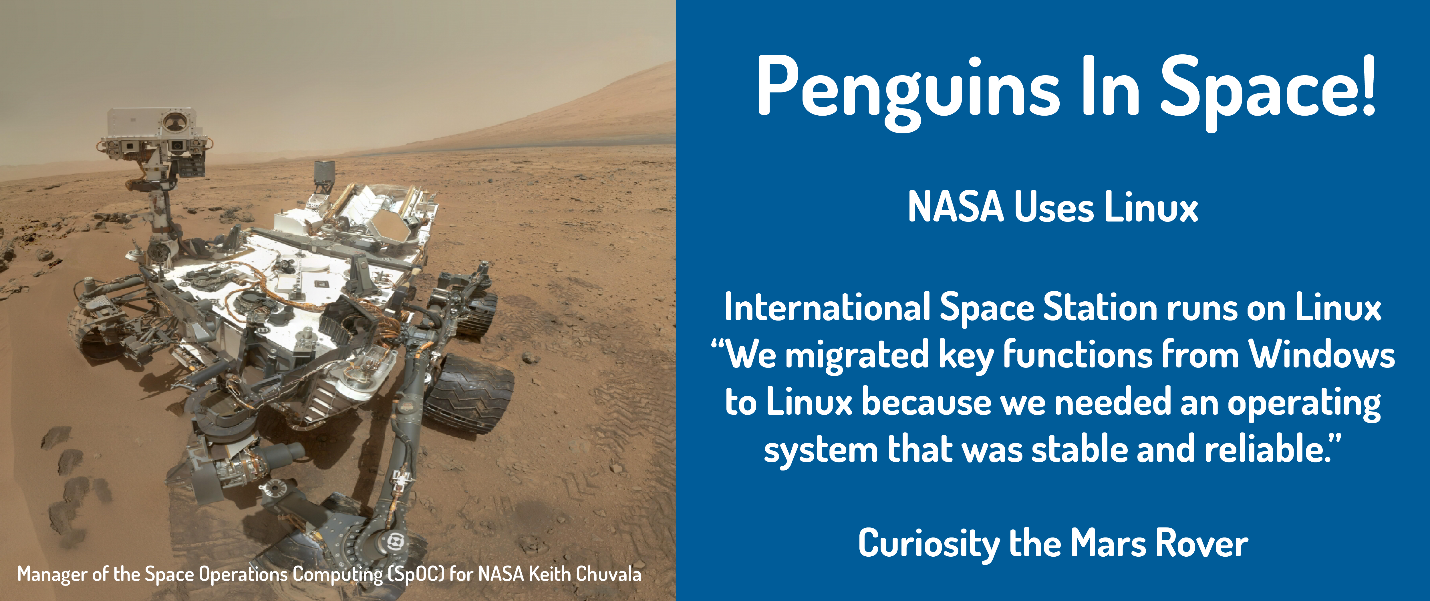
Work alpha-third.txt linux.txt profile.txt

adjectives.txt hello.sh longfile.txt red.txt

alpha-first.txt hidden.txt newhome.txt zoo.txt

**Consider This**

Permissions can have an impact on file management commands, such as the mv command. Moving a file requires write and execute permissions on both the origin and destination directories.





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.
* A copy of a file can be used to transfer a file to removable media devices.
* A copy of an existing document can be used as a template for a new document.

cp [*OPTIONS*] *SOURCE* *DESTINATION*

**Follow Along**

Use the following command to switch to the Documents directory:

**sysadmin@localhost:~$** cd ~/Documents

The cp command is used to copy files. Similar to the mv command, it requires at least two arguments: a source and a destination. For example, to copy the /etc/passwd file to the current directory, use the following command:

**sysadmin@localhost:~/Documents$** cp /etc/passwd .

**Note**

The second argument is the . character. Recall from the *Changing Directories* section that is a shortcut which represents the current directory.

The result of executing the previous command would create a copy of the contents of the /etc/passwd file in the Documents directory, since that is our current directory. This can be confirmed using the ls command:

**sysadmin@localhost:~/Documents$** ls

School alpha-second.txt letters.txt os.csv zoo.txt

Work alpha-third.txt linux.txt passwd

adjectives.txt hello.sh longfile.txt profile.txt

alpha-first.txt hidden.txt newhome.txt red.txt

**Consider This**

Permissions can have an impact on file management commands, such as the cp command. In order to copy a file, it is necessary to have execute permission to access the directory where the file is located and the read permission for the file being copied.

It is also necessary to have write and execute permission on the directory the file is being copied to. Typically, there are two places where you should always have write and execute permission on the directory: your home directory and the /tmp directory.



Copying Files

The dd command is a utility for copying files or entire partitions at the bit level.

dd [*OPTIONS*] OPERAND

This command has several useful features, including:

* It can be used to clone or delete (wipe) entire disks or partitions.
* It can be used to copy raw data to removable devices, such as USB drives and CDROMs.
* It can backup and restore the MBR (Master Boot Record).
* It can be used to create a file of a specific size that is filled with binary zeros, which can then be used as a swap file (virtual memory).

Let's examine the following example. The dd command creates a file named /tmp/swapex with 50 blocks of zeros that are one megabyte in size:

**sysadmin@localhost:~$** dd if=/dev/zero of=/tmp/swapex bs=1M count=50

50+0 records in

50+0 records out

52428800 bytes (52 MB) copied, 0.825745 s, 635 MB/s

The dd command uses special arguments to specify how it will work. The following illustrates some of the more commonly used arguments:

| **Argument** | **Description** |
| --- | --- |
| if | Input File: The input file to be read from.  dd if=/dev/zero of=/tmp/swapex bs=1M count=50  The example reads from the /dev/zerofile, a special file containing an unlimited number of zeros. |
| of | Output File: The output file to be written.  dd if=/dev/zero of=/tmp/swapex bs=1M count=50 |
| bs | Block Size: The block size to be used. By default, the value is considered to be in bytes. Use the following suffixes to specify other units: K, M, G, and T for kilobytes, megabytes, gigabytes and terabytes respectively.  dd if=/dev/zero of=/tmp/swapex bs=1M count=50  The example uses a block size of one megabyte. |
| count | Count: The number of blocks to be read from the input file.  dd if=/dev/zero of=/tmp/swapex bs=1M count=50  The example command reads 50 blocks. |

**Consider This**

No block size or count needs to be specified when copying over entire devices. For example, to clone from one hard drive (/dev/sda) to another (/dev/sdb) execute the following command:

dd if=/dev/sda of=/dev/sdb



## Removing Files

The rm command is used to delete files and directories. 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 almost always permanently gone.

rm [OPTIONS] FILE

**Follow Along**

Use the following command to switch to the Documents directory:

**sysadmin@localhost:~$** cd ~/Documents

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

**sysadmin@localhost:~/Documents$** rm linux.txt

**sysadmin@localhost:~/Documents$** ls linux.txt

ls: cannot access linux.txt: No such file or directory

The rm command will ignore directories that it's asked to remove; to delete a directory, use a recursive option, either the -r or -R options. Just be careful since these options are "recursive", this will delete all files and all subdirectories:

**sysadmin@localhost:~/Documents$** rm Work

rm: cannot remove 'Work': Is a directory

**sysadmin@localhost:~/Documents$** rm -r Work

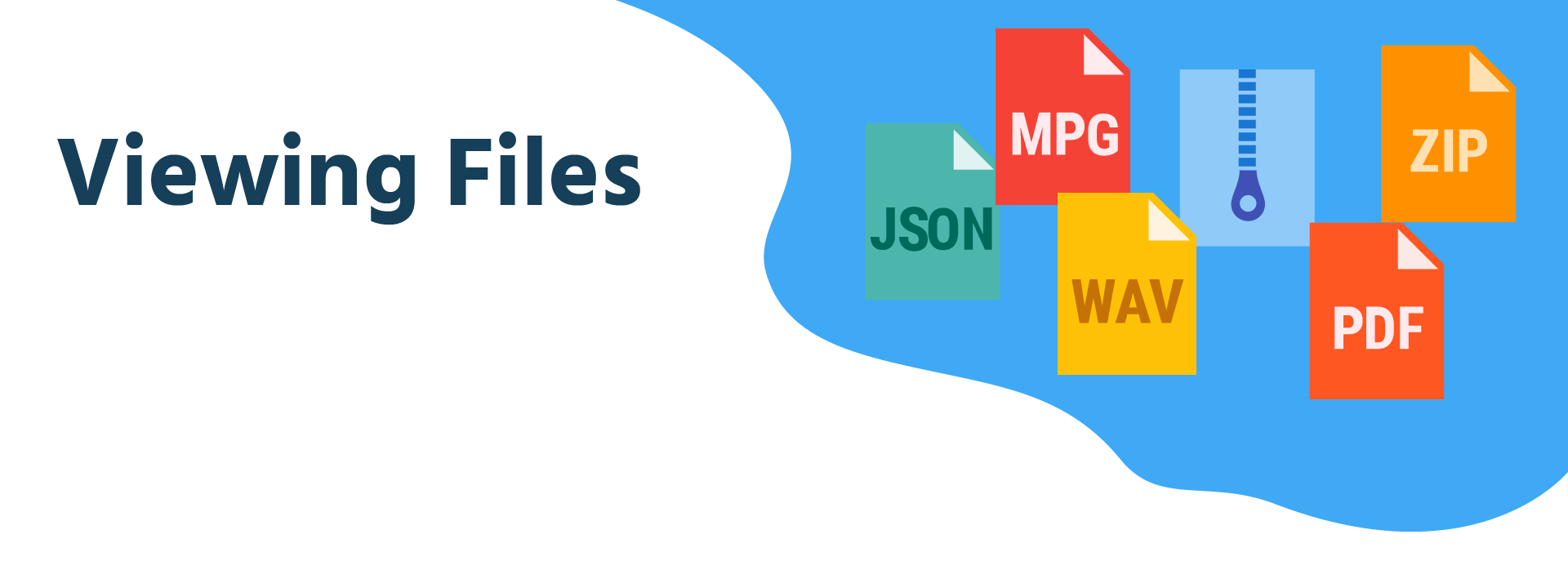
**Warning**

The rm command removes files permanently. To repeat the examples above, reset the terminal using the reset button.

**Consider This**

Permissions can have an impact on file management commands, such as the rm command.

To delete a file within a directory, a user must have write and execute permission on a directory. Regular users typically only have this type of permission in their home directory and its subdirectories.



## Viewing Files

There are a few Linux commands available to view the content of files. The cat command, which stands for “concatenate”, is often used to quickly view the contents of small files.

The cat command will display the entire contents of the file, hence why it is mainly recommended for smaller files where the output is limited and does not require scrolling. To view the contents of a file using the cat command, simply type the command and use the name of the file you wish to view as the argument:

cat [OPTIONS] [FILE]

Our VM has a few small text files that you can view with the cat command. One such file is the animals.txt file:

**Follow Along**

Use the following command to switch to the Documents directory:

**sysadmin@localhost:~$** cd ~/Documents

**sysadmin@localhost:~/Documents$** cat animals.txt

1 retriever

2 badger

3 bat

4 wolf

5 eagle

The cat command displays all five lines of the file above. When viewing larger files, the cat command can result in very lengthy output that cannot be paused to scroll through. A better method of viewing long text files, is with a pager command which has a functionality that can pause and scroll through the output of the file.

**Note**

Examples of pager commands include the more and less commands. These and additional commands used for viewing files in Linux are covered in [NDG Linux Essentials](https://www.netdevgroup.com/online/courses/ndg_linux_essentials.html).

Another way to view the content of files is by using the head and tail commands. These commands are used to view a select number of lines from the top or bottom of a file. Taking a look at a few lines of a file can sometimes be helpful to ensure that the file is the one you want to use.

Another reason to preview only the first or last few lines, is because some files, such as system log files are frequently updated with new entries. Similar to the cat command, the head and tail commands use the name of the file you want to view as the argument to the command:

head [OPTIONS] [FILE]

tail [OPTIONS] [FILE]

To compare the output of the head and tail commands with that of the cat command, use the cat command to view the entire alpha.txt file:

**sysadmin@localhost:~/Documents$** cat alpha.txt

A is for Apple

B is for Bear

C is for Cat

D is for Dog

E is for Elephant

F is for Flower

G is for Grapes

H is for Happy

I is for Ink

J is for Juice

K is for Kangaroo

L is for Lol

M is for Monkey

N is for Nickel

O is for Oval

P is for Pickle

Q is for Quark

R is for Rat

S is for Sloth

T is for Turnip

U is for Up

V is for Velvet

W is for Walrus

X is for Xenon

Y is for Yellow

Z is for Zebra

**sysadmin@localhost:~/Documents$**

In the example above, all twenty-six lines of the file are displayed.

To filter the output and view lines from the top of the alpha.txt file, use the head command:

**sysadmin@localhost:~/Documents$** head alpha.txt

A is for Apple

B is for Bear

C is for Cat

D is for Dog

E is for Elephant

F is for Flower

G is for Grapes

H is for Happy

I is for Ink

J is for Juice

Then, to view lines at the bottom of the alpha.txt file, you use the tail command:

**sysadmin@localhost:~/Documents$** tail alpha.txt

Q is for Quark

R is for Rat

S is for Sloth

T is for Turnip

U is for Up

V is for Velvet

W is for Walrus

X is for Xenon

Y is for Yellow

Z is for Zebra

By examining the output of the head and tail commands above, you can see that the default behavior of the head and tail commands in this shell is to display ten lines.

The -n option with the head and tail commands can be used to specify the amount of lines to display. To use the -n option, specify the amount of lines from the file you want to display after the option and use the filename as an argument:

head -n number\_of\_lines filename

For example, to change the output of the head command to view the first five lines of the alpha.txt file:

**sysadmin@localhost:~/Documents$** head -n 5 alpha.txt

A is for Apple

B is for Bear

C is for Cat

D is for Dog

E is for Elephant

View the last five lines of the alpha.txt file:

**sysadmin@localhost:~/Documents$** tail -n 5 alpha.txt

V is for Velvet

W is for Walrus

X is for Xenon

Y is for Yellow

Z is for Zebra

**sysadmin@localhost:~/Documents$**



## Filtering Input

The grep command is a text filter that will search input and return lines which contain a match to a given pattern.

grep [OPTIONS] PATTERN [FILE]

**Follow Along**

Use the following command to switch to the Documents directory:

**sysadmin@localhost:~$** cd ~/Documents

If the example below fails, repeat the example from Section 11:

**sysadmin@localhost:~/Documents$** cp /etc/passwd .

For example, the passwd file we previously copied into the Documents directory contains the details of special system accounts and user accounts on the system. This file can be very large, however the grep command can be used filter out information about a specific user, such as the sysadmin user. Use sysadmin as the pattern argument and passwd as the file argument:

**sysadmin@localhost:~/Documents$** grep sysadmin passwd

sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin

The command above returned the line from the passwd which contains the pattern sysadmin.

**Note**

This line is the /etc/passwd entry pertaining to the user sysadmin and provides information that is beyond the scope of this course. To learn more about this file, check out [NDG Linux Essentials](https://www.netdevgroup.com/online/courses/ndg_linux_essentials.html).

The example above uses a simple search term as the pattern, however grep is able to interpret much more complex search patterns.



## Regular Expressions

Regular expressions have two common forms: basic and extended. Most commands that use regular expressions can interpret basic regular expressions. However, extended regular expressions are not available for all commands and a command option is typically required for them to work correctly.

The following table summarizes basic regular expression characters:

| **Basic Regex Character(s)** | **Meaning** |
| --- | --- |
| . | Any one single character |
| [ ] | Any one specified character |
| [^ ] | Not the one specified character |
| \* | Zero or more of the previous character |
| ^ | If first character in the pattern, then pattern must be at beginning of the line to match, otherwise just a literal ^ |
| $ | If last character in the pattern, then pattern must be at the end of the line to match, otherwise just a literal $ |

The following table summarizes the extended regular expressions, which must be used with either the egrep command or the -E option with the grep command:

| **Extended Regex Character(s)** | **Meaning** |
| --- | --- |
| + | One or more of the previous pattern |
| ? | The preceding pattern is optional |
| { } | Specify minimum, maximum or exact matches of the previous pattern |
| | | Alternation - a logical "or" |
| ( ) | Used to create groups |

Only basic regular expressions have been covered here. For more information concerning extended regular expressions, check out the [NDG Linux Essentials](https://www.netdevgroup.com/online/courses/ndg_linux_essentials.html) and [NDG Introduction to Linux](https://www.netdevgroup.com/online/courses/ndg_introduction_to_linux_1.html) courses.



## Basic Patterns

Regular expressions are patterns that only certain commands are able to interpret. Regular expressions can be expanded to match certain sequences of characters in text. The examples displayed on this page will make use of regular expressions to demonstrate their power when used with the grep command. In addition, these examples provide a very visual demonstration of how regular expressions work, the text that matches will be displayed in a red color.

**Follow Along**

Use the following cd command to change to the Documents directory.

**sysadmin@localhost:~$** cd ~/Documents

The simplest of all regular expressions use only literal characters, like the example from the previous page:

**sysadmin@localhost:~/Documents$** grep sysadmin passwd

sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin

### **Anchor Characters**

Anchor characters are one of the ways regular expressions can be used to narrow down search results. For example, the pattern root appears many times in the /etc/passwd file:

**sysadmin@localhost:~/Documents$** grep 'root' passwd

root:x:0:0:root:/root:bin/bash

operator:x:11:0:operator:/root:/sbin/nologin

To prevent the shell from misinterpreting them as special shell characters, these patterns should be protected by strong quotes, which simply means placing them between single quotes.

The first anchor character ^ is used to ensure that a pattern appears at the beginning of the line. For example, to find all lines in /etc/passwd that start with root use the pattern ^root. Note that ^ must be the first character in the pattern to be effective.

**sysadmin@localhost:~/Documents$** grep '^root' /etc/passwd

root:x:0:0:root:/root:bin/bash

For the next example, first examine the alpha-first.txt file. The cat command can be used to print the contents of a file:

**sysadmin@localhost:~/Documents$** cat alpha-first.txt

A is for Animal

B is for Bear

C is for Cat

D is for Dog

E is for Elephant

F is for Flower

The second anchor character $ can be used to ensure a pattern appears at the end of the line, thereby effectively reducing the search results. To find the lines that end with an r in the alpha-first.txt file, use the pattern r$:

**sysadmin@localhost:~/Documents$** grep 'r$' alpha-first.txt

B is for Bear

F is for Flower

Again, the position of this character is important, the $ must be the last character in the pattern in order to be effective as an anchor.

### **Match a Single Character With .**

The following examples will use the red.txt file:

**sysadmin@localhost:~/Documents$** cat red.txt

red

reef

rot

reeed

rd

rod

roof

reed

root

reel

read

One of the most useful expressions is . It will match any character except for the new line character. The pattern r..f would find any line that contained the letter r followed by exactly two characters (which can be any character except a newline) and then the letter f:

**sysadmin@localhost:~/Documents$** grep 'r..f' red.txt

reef

roof

The same concept can be repeated using other combinations. The following will find four letter words that start with r and with d:

**sysadmin@localhost:~/Documents$** grep 'r..d' red.txt

reed

read

This character can be used any number of times. To find all words that have at least four characters the following pattern can be used:

**sysadmin@localhost:~/Documents$** grep '....' red.txt

reef

reeed

roof

reed

root

reel

read

The line does not have to be an exact match, it simply must contain the pattern, as seen here when r..t is searched for in the /etc/passwd file:

**sysadmin@localhost:~/Documents$** grep 'r..t' /etc/passwd

root:x:0:0:root:/root:/bin/bash

operator:x:1000:37::/root:

### **Match a Single Character With [ ]**

The square brackets [ ] match a single character from the list or range of possible characters contained within the brackets.

For example, given the profile.txt file:

**sysadmin@localhost:~/Documents$** cat profile.txt

Hello my name is Joe.

I am 37 years old.

3121991

My favorite food is avocados.

I have 2 dogs.

123456789101112

To find all the lines in the profile.txt which have a number in them, use the pattern [0123456789] or [0-9]:

**sysadmin@localhost:~/Documents$** grep '[0-9]' profile.txt

I am 37 years old.

3121991

I have 2 dogs.

123456789101112

On the other hand, to find all the lines which contain any non-numeric characters, insert a ^ as the first character inside the brackets. This character negates the characters listed:

**sysadmin@localhost:~/Documents$** grep '[^0-9]' profile.txt

Hello my name is Joe.

I am 37 years old.

My favorite food is avocados.

I have 2 dogs.

**Note**

Do not mistake [^0-9] to match lines which do not contain numbers. It actually matches lines which contain non-numbers. Look at the original file to see the difference. The third and sixth lines only contain numbers, they do not contain non-numbers so those lines do not match.

When other regular expression characters are placed inside of square brackets, they are treated as literal characters. For example, the . normally matches any one character, but placed inside the square brackets, then it will just match itself. In the next example, only lines which contain the . character are matched.

**sysadmin@localhost:~/Documents$** grep '[.]' profile.txt

Hello my name is Joe.

I am 37 years old.

My favorite food is avocados.

I have 2 dogs.

### **Match a Repeated Character Or Patterns With \***

The regular expression character \* is used to match zero or more occurrences of a character or pattern preceding it. For example e\* would match zero or more occurrences of the letter e:

**sysadmin@localhost:~/Documents$** cat red.txt

red

reef

rot

reeed

rd

rod

roof

reed

root

reel

read

**sysadmin@localhost:~/Documents$** grep 're\*d' red.txt

red

reeed

rd

reed

It is also possible to match zero or more occurrences of a list of characters by utilizing the square brackets. The pattern [oe]\* used in the following example will match zero or more occurrences of the o character or the e character:

**sysadmin@localhost:~/Documents$** grep 'r[oe]\*d' red.txt

red

reeed

rd

rod

reed

When used with only one other character, \* isn't very helpful. Any of the following patterns would match every string or line in the file: .\* e\* b\* z\*.

**sysadmin@localhost:~/Documents$** grep 'z\*' red.txt

red

reef

rot

reeed

rd

rod

roof

reed

root

reel

read

**sysadmin@localhost:~/Documents$** grep 'e\*' red.txt

red

reef

rot

reeed

rd

rod

roof

reed

root

reel

read

This is because \* can match zero occurrences of a pattern. In order to make the \* useful, it is necessary to create a pattern which includes more than just the one character preceding \*. For example, the results above can be refined by adding another e to make the pattern ee\* effectively matching every line which contains at least one e.

**sysadmin@localhost:~/Documents$** grep 'ee\*' red.txt

red

reef

reeed

reed

reel

read

### **Standard Input**

If a file name is not given, the grep command will read from standard input, which normally comes from the keyboard with input provided by the user who runs the command. This provides an interactive experience with grep where the user types in the input and grep filters as it goes. Feel free to try it out, just press **Ctrl-D** when you're ready to return to the prompt.

**Follow Along**

Use the following cd command to return to the home directory:

**sysadmin@localhost:~/Documents$** cd ~





## Shutting Down

The shutdown command arranges for the system to be brought down in a safe way. All logged-in users are notified that the system is going down and within the last five minutes leading up to the shutdown, new logins are prevented.

shutdown [OPTIONS] TIME [MESSAGE]

**Follow Along**

The shutdown command requires administrative access, switch to the root account for this section using the following command. Use netlab123 as the password.

**sysadmin@localhost:~$** su -

Password:

**root@localhost:~#**

Our virtual machines won't actually shutdown! Feel free to use the command, but be aware it will just revert to the prompt after the countdown instead of bringing the system down. You may have to press **Enter** to bring back the command prompt.

**root@localhost:~#** shutdown now

Broadcast message from sysadmin@localhost

(/dev/console) at 2:05 ...

The system is going down for maintenance NOW!

Unlike other commands used to bring the system down, the shutdown command requires a time argument specifying when the shutdown should begin. Formats of this time argument can be the word now, a time of day in the format hh:mm or the number of minutes to delay in the format +minutes.

**Consider This**

The clock on our system may be set to a different timezone than the one in which you are located. To check the time in the terminal, use the date command. On our machines, the default format of the date command output is as follows:

weekday month day hour:minute:second UTC year

The letters UTC present in the output indicates that the time is being displayed as Universal Coordinated Time.

**root@localhost:~#** date

Fri Jan 22 01:50:07 UTC 2016

**root@localhost:~#** shutdown 01:51

Broadcast message from sysadmin@localhost

(/dev/console) at 1:50 ...

The system is going down for maintenance in 1 minute!

Broadcast message from sysadmin@localhost

(/dev/console) at 1:51 ...

The system is going down for maintenance NOW!

The shutdown command also has an optional message argument, indicating a message that will appear in the terminals of all users. For example:

**root@localhost:~#** shutdown +1 "Goodbye World!"

Broadcast message from sysadmin@localhost

(/dev/console) at 3:07 ...

The system is going down for maintenance in 1 minute!

Goodbye World!

shutdown: Unable to shutdown system

**root@localhost:~#**

Broadcast message from sysadmin@localhost

(/dev/console) at 3:08 ...

The system is going down for maintenance NOW!

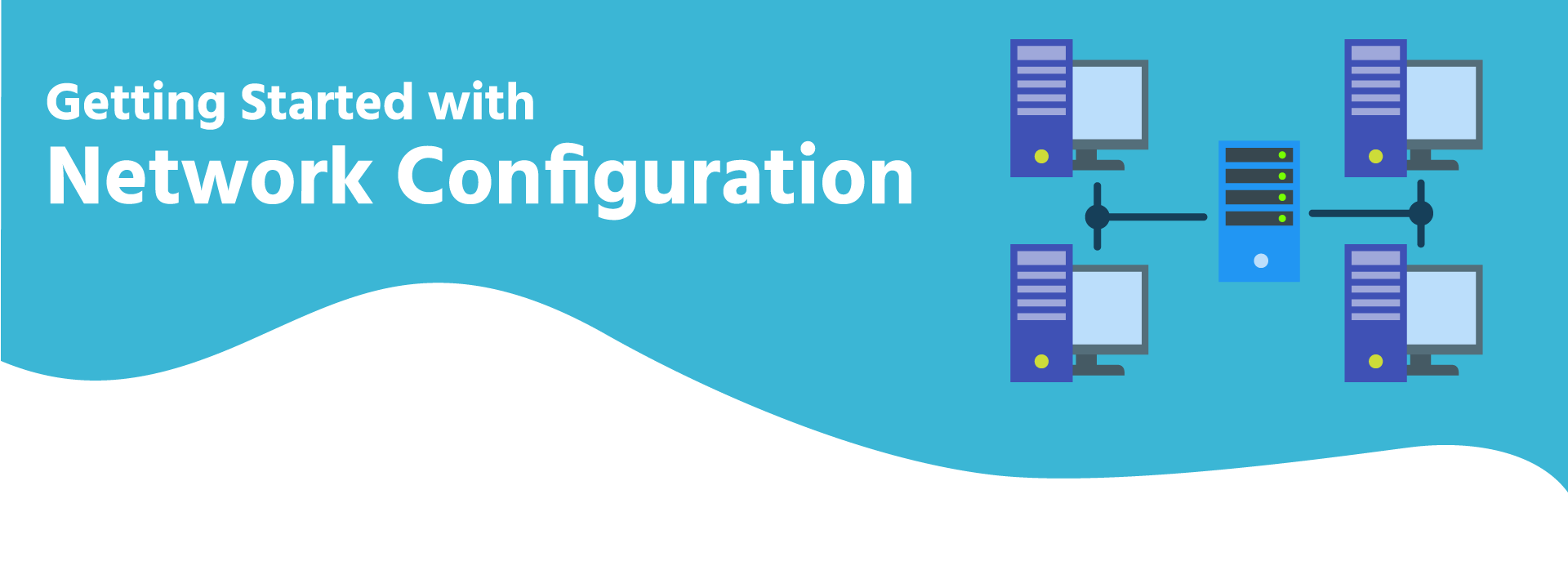
Goodbye World!

**Follow Along**

Exit the root account using the exit command:

**root@localhost:~#** exit

logout



## Network Configuration

The ifconfig command stands for "interface configuration" and is used to display network configuration information.

ifconfig [OPTIONS]

**Note**

The iwconfig command is similar to the ifconfig command, but it is dedicated to wireless network interfaces.

Not all network settings are important for this module, but it is important to note in the following example that the IPv4 address of the primary network device eth0 is 192.168.1.2 and that the device is currently active (UP):

**sysadmin@localhost:~$** ifconfig

eth0 Link encap:Ethernet HWaddr b6:84:ab:e9:8f:0a

inet addr:192.168.1.2 Bcast:0.0.0.0 Mask:255.255.255.0

inet6 addr: fe80::b484:abff:fee9:8f0a/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:95 errors:0 dropped:4 overruns:0 frame:0

TX packets:9 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:25306 (25.3 KB) TX bytes:690 (690.0 B)

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

inet6 addr: ::1/128 Scope:Host

UP LOOPBACK RUNNING MTU:65536 Metric:1

RX packets:6 errors:0 dropped:0 overruns:0 frame:0

TX packets:6 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:0

RX bytes:460 (460.0 B) TX bytes:460 (460.0 B)

**Consider This**

The lo device is referred to as the loopback device. It is a special network device used by the system when sending network-based data to itself.

The ifconfig command can also be used to temporarily modify network settings. Typically these changes should be permanent, so using the ifconfig command to make such changes is fairly rare.

The ping command is used to verify connectivity between two computers. It does this by sending packets to another machine on a network. If the sender receives a response it should be possible to connect to that machine.

Information is sent using “packets”; the encapsulated unit of data sent over a network. In order for the packets to find the other computer, they will need an address. The ping command uses IP addresses to identify a computer on the network that it wants to connect to.

By default, the ping command will continue sending packets until the break command (**CTL** + **C**) is entered at the console. To limit how many pings are sent, use the -c option followed by the number of pings to be sent. The example below shows ping being limited to 4 iterations with -c 4.

If the ping command is successful, you will see output like the following:

**root@localhost:~#** ping -c 4 192.168.1.2

PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.

64 bytes from 192.168.1.2: icmp\_req=1 ttl=64 time=0.051 ms

64 bytes from 192.168.1.2: icmp\_req=2 ttl=64 time=0.064 ms

64 bytes from 192.168.1.2: icmp\_req=3 ttl=64 time=0.050 ms

64 bytes from 192.168.1.2: icmp\_req=4 ttl=64 time=0.043 ms

--- 192.168.1.2 ping statistics ---

4 packets transmitted, 4 received, 0% packet loss, time 2999ms

rtt min/avg/max/mdev = 0.043/0.052/0.064/0.007 ms

**root@localhost:~#**

If the ping command fails, you will receive a message stating, Destination Host Unreachable:

**root@localhost:~#** ping -c 4 192.168.1.1

PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.

From 192.168.1.2 icmp\_seq=1 Destination Host Unreachable

From 192.168.1.2 icmp\_seq=2 Destination Host Unreachable

From 192.168.1.2 icmp\_seq=3 Destination Host Unreachable

From 192.168.1.2 icmp\_seq=4 Destination Host Unreachable

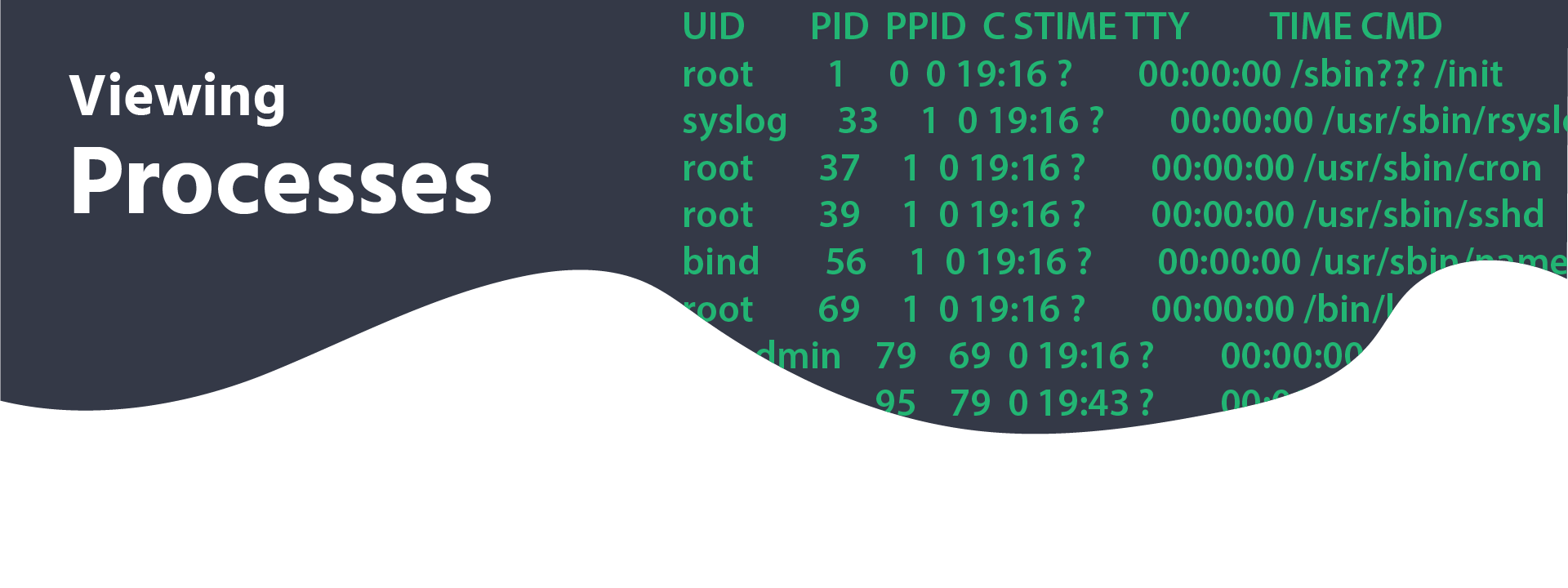
--- 192.168.1.1 ping statistics ---

4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 2999ms

pipe 4

**root@localhost:~#**

The ping command may fail even though the remote machine is connecting. This is because some administrators configure their machines, or even entire networks, not to respond to ping requests as a security measure. The ping command also works with a hostname, or domain name like yahoo.com. Using this first saves time, if that ping command is successful, there is proper name resolution AND the IP address is functioning properly as well.



Viewing Processes

Running a command results in something called a *process*. In the Linux operating system, processes are executed with the privileges of the user who executes the command. This allows for processes to be limited to certain capabilities based upon the user identity.

Although there are exceptions, generally the operating system will differentiate users based upon whether they are the administrator. Typically regular users, like the sysadmin user, cannot control another user's processes. Users who have administrative privileges, like the root account, can control any user processes, including stopping any user process.

The ps command can be used to list processes.

ps [*OPTIONS*]

**sysadmin@localhost:~$** ps

PID TTY TIME CMD

80 ? 00:00:00 bash

94 ? 00:00:00 ps

The ps command will display the processes that are running in the current terminal by default. In the example above, the bottom line is the process created by the execution of the ps command. The output includes the following columns of information:

* PID: The process identifier, which is unique to the process. This information is useful for controlling the process by its ID number.
* TTY: The name of the terminal where the process is running. This information is useful for distinguishing between different processes that have the same name.
* TIME: The total amount of processor time used by the process. Typically, this information isn't used by regular users.
* CMD: The command that started the process.

Instead of viewing just the processes running in the current terminal, users may want to view every process running on the system. The -e option will display every process:

**sysadmin@localhost:~$** ps -e

PID TTY TIME CMD

1 ? 00:00:00 init

33 ? 00:00:00 rsyslogd

37 ? 00:00:00 cron

39 ? 00:00:00 sshd

56 ? 00:00:00 named

69 ? 00:00:00 login

79 ? 00:00:00 bash

94 ? 00:00:00 ps

Typically, the -f option is also used as it provides more detail in the output of the command, including options and arguments. Look for the ps command on the last line, the CMD column now includes the options used:

**sysadmin@localhost:~$** ps -ef

UID PID PPID C STIME TTY TIME CMD

root 1 0 0 19:16 ? 00:00:00 /sbin??? /init

syslog 33 1 0 19:16 ? 00:00:00 /usr/sbin/rsyslogd

root 37 1 0 19:16 ? 00:00:00 /usr/sbin/cron

root 39 1 0 19:16 ? 00:00:00 /usr/sbin/sshd

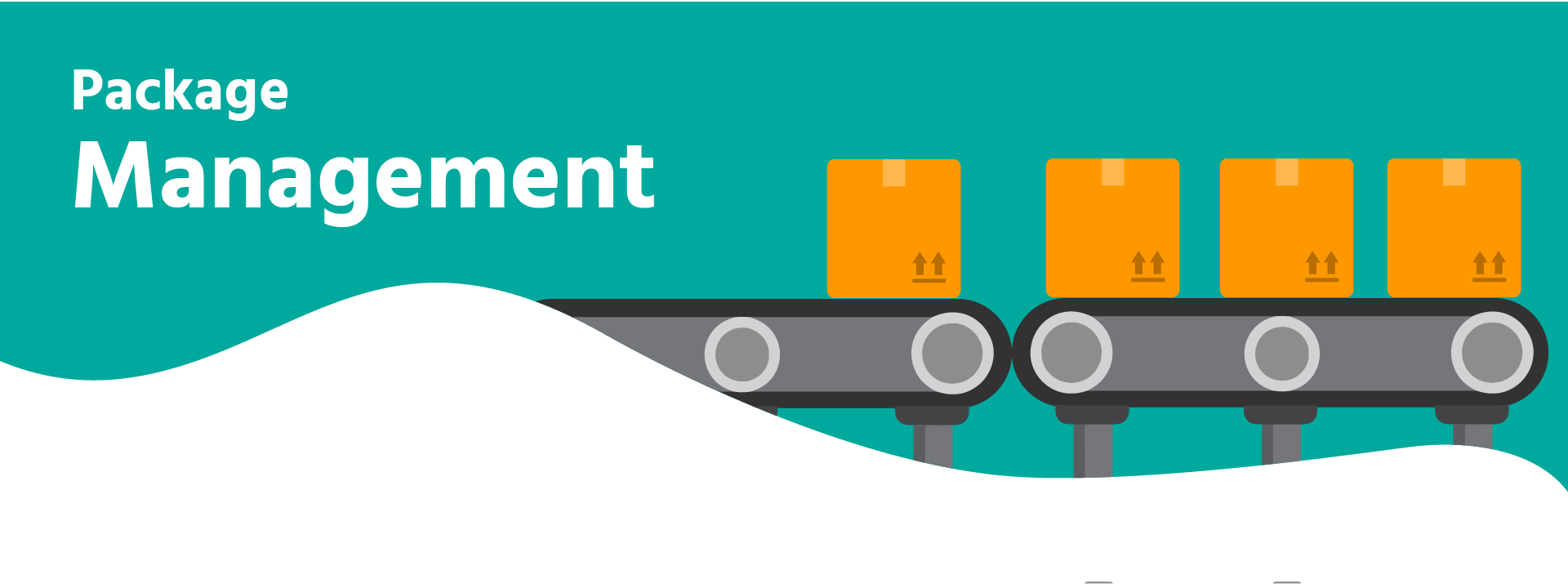
bind 56 1 0 19:16 ? 00:00:00 /usr/sbin/named -u bind

root 69 1 0 19:16 ? 00:00:00 /bin/login -f

sysadmin 79 69 0 19:16 ? 00:00:00 -bash

sysadmin 95 79 0 19:43 ? 00:00:00 ps -ef





## Package Management

Package management is a system by which software can be installed, updated, queried or removed from a filesystem. In Linux, there are many different software package management systems, but the two most popular are those from Debian and Red Hat. The virtual machines for this course use Ubuntu, a derivative of Debian.

At the lowest level of the Debian package management system is the dpkg command. This command can be tricky for novice Linux users, so the Advanced Package Tool, apt-get, a front-end program to the dpkg tool, makes management of packages even easier.

**Note**

A front-end program is a program that users can see and interact with.

**Follow Along**

Many of the package management commands require administrative access, so they will be prefaced with the sudo command. Use netlab123 as the password when prompted.

### **Installing Packages**

Package files are commonly installed by downloading them directly from repositories located on Internet servers. The Debian repositories contain more than 65,000 different packages of software. Before installing a package, it is good practice to use the refresh the list of available packages using the apt-get update command.

sudo apt-get update

**sysadmin@localhost:~$** sudo apt-get update

[sudo] password for sysadmin:

Ign file: amd64/ InRelease

Ign file: amd64/ Release.gpg

Ign file: amd64/ Release

Reading package lists... Done

To search for keywords within these packages, you can use the apt-cache search command.

apt-cache search [keyword]

The keyword that is used should match part of the name or description of the package that is to be located. Multiple keywords can be used to further clarify the search; for example, the search term web server would provide better results than web or server.

To find packages associated with the cow keyword:

**sysadmin@localhost:~$** apt-cache search cow

cowsay - configurable talking cow

Once you've found the package that you want to install, you can install it with the apt-get install command:

sudo apt-get install [package]

**sysadmin@localhost:~$** sudo apt-get install cowsay

[sudo] password for sysadmin:

Reading package lists... Done

Building dependency tree

Reading state information... Done

Suggested packages:

filters

The following NEW packages will be installed:

cowsay

0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.

Need to get 0 B/18.5 kB of archives.

After this operation, 90.1 kB of additional disk space will be used.

Selecting previously unselected package cowsay.

(Reading database ... 24313 files and directories currently installed.)

Preparing to unpack .../cowsay\_3.03+dfsg1-6\_all.deb ...

Unpacking cowsay (3.03+dfsg1-6) ...

Processing triggers for man-db (2.6.7.1-1ubuntu1) ...

Setting up cowsay (3.03+dfsg1-6) ...

**Consider This**

The cowsay command is a configurable talking cow! Use a word or phrase as an argument:

**sysadmin@localhost:~$** cowsay 'NDG Linux Unhatched'

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

< NDG Linux Unhatched >

---------------------

\ ^\_\_^

\ (oo)\\_\_\_\_\_\_\_

(\_\_)\ )\/\

||----w |

|| ||

We recommend enclosing the argument in single quotes to prevent the shell from interpreting special characters.

### **Updating Packages**

The apt-get install command can also update a package, if that package is installed and a newer version is available. If the package is not already on the system, it would be installed; if it is on the system, it would be updated.

Updating all packages of the system should be done in two steps. First, update the cache of all packages available with apt-get update. Second, execute the apt-get upgrade command and all packages and dependencies will be updated.

apt-get update

apt-get upgrade

**sysadmin@localhost:~$** sudo apt-get update

[sudo] password for sysadmin:

Ign file: amd64/ InRelease

Ign file: amd64/ Release.gpg

Ign file: amd64/ Release

Reading package lists... Done

**sysadmin@localhost:~$** sudo apt-get upgrade

Reading package lists... Done

Building dependency tree

Reading state information... Done

Calculating upgrade... Done

0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

### **Removing Packages**

The apt-get command is able to either remove or purge a package. The difference between the two is that purging deletes all package files, while removing deletes all but the configuration files for the package.

An administrator can execute the apt-get remove command to remove a package or the apt-get purge command to purge a package completely from the system.

apt-get remove [package]

apt-get purge [package]

For example, to purge cowsay completely, execute the following command. Enter **Y** when prompted:

**sysadmin@localhost:~$** sudo apt-get purge cowsay

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following packages will be REMOVED:

cowsay\*

0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.

After this operation, 90.1 kB disk space will be freed.

Do you want to continue? [Y/n] y

(Reading database ... 24377 files and directories currently installed.)

Removing cowsay (3.03+dfsg1-6) ...

Processing triggers for man-db (2.6.7.1-1ubuntu1) ...



## Updating User Passwords

The passwd command is used to update a user’s password. Users can only change their own passwords, whereas the root user can update the password for any user.

passwd [OPTIONS] [USER]

For example, since we are logged in as the sysadmin user we can change the password for that account. Execute the passwd command. You will be prompted to enter the existing password once and the new password twice. For security reasons, no output is displayed while the password is being typed. The output is shown as follows:

**sysadmin@localhost:~$** passwd

Changing password for sysadmin.

(current) UNIX password: netlab123

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

If the user wants to view status information about their password, they can use the -S option:

**sysadmin@localhost:~$** passwd -S sysadmin

sysadmin P 03/01/2015 0 99999 7 -1

The output fields are explained below:

| **Field** | **Example** | **Meaning** |
| --- | --- | --- |
| User Name | sysadmin | The name of the user. |
| Password Status | P | P indicates a usable password.  L indicates a locked password.  NP indicates no password. |
| Change Date | 03/01/2015 | The date when the password was last changed. |
| Minimum | 0 | The minimum number of days that must pass before the current password can be changed by the user. |
| Maximum | 99999 | The maximum number of days remaining for the password to expire. |
| Warn | 7 | The number of days prior to password expiry that the user is warned. |
| Inactive | -1 | The number of days after password expiry that the user account remains active. |

**Follow Along**

Switch the root account using the following command:

**sysadmin@localhost:~$** su root

Password:

**root@localhost:~#**

Use netlab123 as the password.

The root user can change the password of any user. If the root user wants to change the password for sysadmin, they would execute the following command:

**root@localhost:~#** passwd sysadmin

Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

**Follow Along**

Exit the root account using the exit command:

**root@localhost:~#** exit

logout



## Redirection

Adding content to files in Linux can be done in a variety of ways. Linux has a few text editors that can be used to add content to a file. However, this method requires some familiarity with Linux text editor commands.

**Note**

Linux text editors are covered in the next section of this course.

There is a way in Linux to quickly add content to a file using a command line feature called input/output (I/O) redirection. I/O redirection allows for information in the command line to be sent to files, devices, and other commands. The input or output of a command is redirected from its default destination to a different location. I/O redirection is like a series of train tracks, where a switch can be enabled to direct the output of a command on a different track so it goes somewhere else in the shell. In this section, we are writing to files by redirecting the output of a command to a file.

When it comes to command input and output there are three paths, or “tracks”. These paths are called file descriptors. The first file descriptor is standard input, abbreviated as STDIN. Standard input is the information the command receives and processes when it is executed, essentially what a user types on the keyboard. The second file descriptor is standard output, abbreviated as STDOUT. Standard output is the information that the command displays, the output of the command. The last file descriptor is standard error, abbreviated as STDERR. STDERR, are the error messages generated by commands that are not correctly executed. The following are examples of how file descriptors will appear in the terminal:

**Standard Input (STDIN)**

sysadmin@localhost:~$ ls ~/Documents

**Standard Output (STDOUT)**

sysadmin@localhost:~$ ls

Desktop Documents Downloads Music Pictures Public Templates Videos

**Standard Error (STDERR)**

sysadmin@localhost:~$ ls fakefile

ls: cannot access fakefile: No such file or directory

This section will cover one of the three file descriptors, STDOUT, and how to redirect STDOUT from where you normally see it, in the terminal, to a file in the filesystem. To use redirection, simply use a greater-than symbol > along with a filename:

[COMMAND] > [FILE]

To demonstrate redirection, we will use the output of the cat command. Without redirection, the output of the cat command will be displayed in the terminal:

**Follow Along**

Use the following command to switch to the Documents directory:

**sysadmin@localhost:~$** cd ~/Documents

**sysadmin@localhost:~/Documents$** cat food.txt

Food is good.

Now use the > character to redirect the STDOUT of the cat food.txt command above to a new file called newfile1.txt:

**sysadmin@localhost:~/Documents$** cat food.txt > newfile1.txt

**sysadmin@localhost:~/Documents$**

As you can see, there is no output displayed since the STDOUT has been redirected to the newfile1.txt file. Verify that the STDOUT of the cat food.txt command is in newfile1.txt:

**sysadmin@localhost:~/Documents$** cat newfile1.txt

Food is good.

This is useful if you need to copy content from an important file to another file in order to edit the contents without modifying the original file. However, what if you want to add a comment or note to a file? To do this, you can use the echo command. The echo command is used to print output in the terminal:

**sysadmin@localhost:~/Documents$** echo "Hello"

Hello

**sysadmin@localhost:~/Documents$**

Printing comments to the screen is a fun feature but the echo command can be made more useful by using redirection. Using the echo command, content can be added to the newfile1.txt file:

**sysadmin@localhost:~/Documents$** cat newfile1.txt

Food is good.

**sysadmin@localhost:~/Documents$** echo "I like food." > newfile1.txt

**sysadmin@localhost:~/Documents$** cat newfile1.txt

I like food.

**sysadmin@localhost:~/Documents$**

Notice that the STDOUT of the echo command has replaced the original content of the file. This is because the single > character will overwrite any contents in an existing file. To append rather than overwrite content to a file, use a double greater-than symbol >>:

**sysadmin@localhost:~/Documents$** echo "This food is good." >> newfile1.txt

**sysadmin@localhost:~/Documents$** cat newfile1.txt

I like food.

This food is good.

**sysadmin@localhost:~/Documents$**

**Important**

To redirect information to an existing file, the user must have write permissions on that file.



## Text Editor

The premier text editor for Linux and UNIX is a program called vi. While there are numerous editors available for Linux that range from the tiny editor nano to the massive emacs editor, there are several advantages to the vi editor:

* The vi editor is available on every Linux distribution in the world. This is not true of any other editor.
* The vi editor can be executed both in a CLI (command line interface) and a GUI (graphical user interface).
* While new features have been added to the vi editor, the core functions have been around for decades. This means that if someone learned the vi editor in the 1970s, they could use a modern version without any problem. While that seems trivial, it may not seem so trivial twenty years from now.

**Consider This**

The correct way to pronounce the vi editor is the vee-eye editor. The letters vi stand for visual, but it was never pronounced this way by the developers, but rather the letter v followed by the letter i.

In reality, most Linux systems don't include the original vi, but an improved version of it known as vim, for vi improved. This fact may be hidden by most Linux distributions. For the most part, vim works just like vi, but has additional features. For the topics that will be covered in this course, either vi or vim will work.

To get started using vi, simply type the command followed by the pathname to the file to edit or create:

**sysadmin@localhost:~$** vi newfile.txt

There are three modes used in vi: command mode, insert mode, and ex mode.

## Command Mode Movement

Initially, the program starts in command mode. Command mode is used to type commands, such as those used to move around a document, manipulate text, and access the other two modes. To return to command mode at any time, press the **Esc** key.

Once some text has been added into a document, to perform actions like moving the cursor, the **Esc** key needs to be pressed first to return to command mode. This seems like a lot of work, but remember that vi works in a terminal environment where a mouse is useless.

Movement commands in vi have two aspects, a motion and an optional number prefix, which indicates how many times to repeat that motion. The general format is as follows:

[count] motion

The following table summarizes the motion keys available:

| **Motion** | **Result** |
| --- | --- |
| h | Left one character |
| j | Down one line |
| k | Up one line |
| l | Right one character |
| w | One word forward |
| b | One word back |
| ^ | Beginning of line |
| $ | End of the line |

**Note**

Since the upgrade to vim it is also possible to use the arrow keys ←↓↑→ instead of hjkl respectively.

These motions can be prefixed with a number to indicate how many times to perform the movement. For example, 5h would move the cursor five characters to the left and 3w would move the cursor three words to the right.

To move the cursor to a specific line number, type that line number followed by the G character. For example, to get to the fifth line of the file type 5G. 1G or gg can be used to go to the first line of the file, while a lone G will take you to the last line. To find out which line the cursor is currently on, use **CTRL-G**.

## Command Mode Actions

The standard convention for editing content with word processors is to use copy, cut, and paste. The vi program has none of these. Instead, vi uses the following three commands:

| **Standard** | **Vi** | **Meaning** |
| --- | --- | --- |
| cut | d | delete |
| copy | y | yank |
| paste | P | p | put |

The motions learned from the previous page are used to specify where the action is to take place, always beginning with the present cursor location. Either of the following general formats for action commands is acceptable:

action [count] motion

[count] action motion

### **Delete**

Delete removes the indicated text from the page and saves it into the buffer, the buffer being the equivalent of the "clipboard" used in Windows or Mac OSX. The following table provides some common usage examples:

| **Action** | **Result** |
| --- | --- |
| dd | Delete current line |
| 3dd | Delete the next three lines |
| dw | Delete the current word |
| d3w | Delete the next three words |
| d4h | Delete four characters to the left |

### **Change**

Change is very similar to delete; the text is removed and saved into the buffer, however, the program is switched to insert mode to allow immediate changes to the text. The following table provides some common usage examples:

| **Action** | **Result** |
| --- | --- |
| cc | Change current line |
| cw | Change current word |
| c3w | Change the next three words |
| c5h | Change five characters to the left |

### **Yank**

Yank places content into the buffer without deleting it. The following table provides some common usage examples:

| **Action** | **Result** |
| --- | --- |
| yy | Yank current line |
| 3yy | Yank the next three lines |
| yw | Yank the current word |
| y$ | Yank to the end of the line |

### **Put**

Put places the text saved in the buffer either before or after the cursor position. Notice that these are the only two options, put does not use the motions like the previous action commands.

| **Action** | **Result** |
| --- | --- |
| p | Put (paste) after cursor |
| P | Put before cursor |

### **Searching in vi**

Another standard function that word processors offer is find. Often, people use **CTRL+F** or look under the edit menu. The vi program uses search. Search is more powerful than find because it supports both literal text patterns and regular expressions.

To search forward from the current position of the cursor, use the / to start the search, type a search term, and then press the **Enter** key to begin the search. The cursor will move to the first match that is found.

To proceed to the next match using the same pattern, press the n key. To go back to a previous match, press the N key. If the end or the beginning of the document is reached, the search will automatically wrap around to the other side of the document.

To start searching backwards from the cursor position, start by typing ?, then type the pattern to search for matches and press the **Enter** key.

## Insert Mode

Insert mode is used to add text to the document. There a few ways to enter insert mode from command mode, each differentiated by where the text insertion will begin. The following table covers the most common:

| **Input** | **Purpose** |
| --- | --- |
| a | Enter insert mode right after the cursor |
| A | Enter insert mode at the end of the line |
| i | Enter insert mode right before the cursor |
| I | Enter insert mode at the beginning of the line |
| o | Enter insert mode on a blank line after the cursor |
| O | Enter insert mode on a blank line before the cursor |

## Ex Mode

Originally, the vi editor was called the ex editor. The name vi was the abbreviation of the **visual** command in the ex editor which switched the editor to "visual" mode.

In the original normal mode, the ex editor only allowed users to see and modify one line at a time. In the visual mode, users could see as much of the document that will fit on the screen. Since most users preferred the visual mode to the line editing mode, the ex program file was linked to a vi file, so that users could start ex directly in visual mode when they ran the vi link.

Eventually, the actual program file was renamed vi and the ex editor became a link that pointed the vi editor.

When the ex mode of the vi editor is being used, it is possible to view or change settings, as well as carry out file-related commands like opening, saving or aborting changes to a file. In order to get to the ex mode, type a : character in command mode. The following table lists some common actions performed in ex mode:

| **Input** | **Purpose** |
| --- | --- |
| :w | Write the current file to the filesystem |
| :w filename | Save a copy of the current file as filename |
| :w! | Force writing to the current file |
| :1 | Go to line number 1 or whatever number is given |
| :e filename | Open filename |
| :q | Quit if no changes made to file |
| :q! | Quit without saving changes to file |

A quick analysis of the table above reveals that if an exclamation mark, !, is added to a command, it then attempts to force the operation. For example, imagine you make changes to a file in the vi editor and then try to quit with :q, only to discover that the command fails. The vi editor doesn't want to quit without saving the changes you made to a file, but you can force it to quit with the ex command :q!.

**Consider This**

Although the ex mode offers several ways to save and quit, there's also ZZ that is available in command mode; this is the equivalent of :wq. There are many more overlapping functions between ex mode and command mode. For example, ex mode can be used to navigate to any line in the document by typing : followed by the line number, while the G can be used in command mode as previously demonstrated.

**Follow Along**

If you have a text file open, exit it by executing the :q! command. This will quit without saving changes.

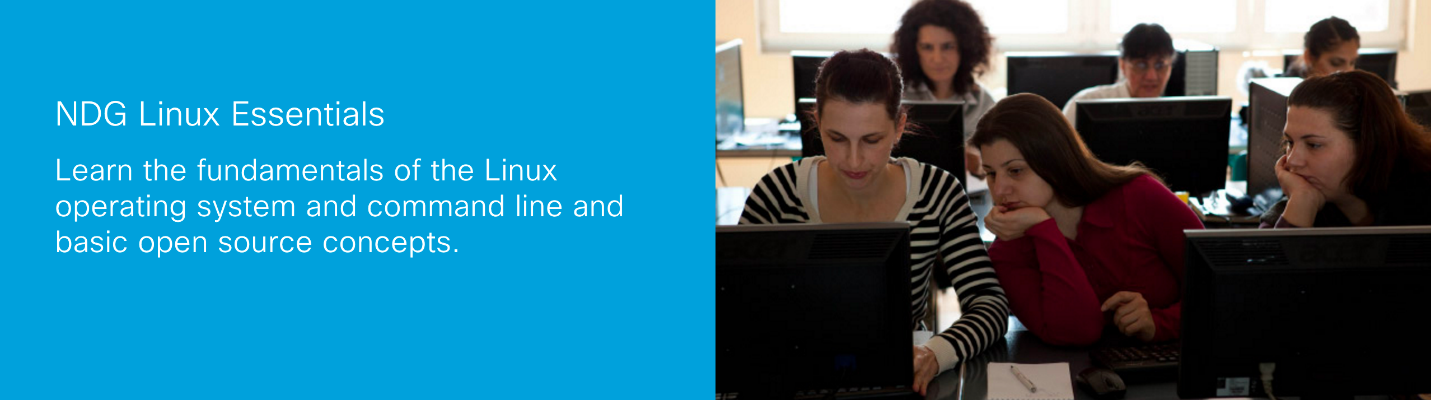


## Moving Forward

We hope you've enjoyed this brief introduction into the world of Linux. The content contained here aligns to the Linux knowledge covered under LPI Linux Essentials exam objectives, but there is so much more! Get certification level Linux skills to advance your career!

### **[NDG Linux Essentials](https://www.netacad.com/courses/os-it/ndg-linux-essentials" \t "_blank)**

If you're interested in broadening your Linux knowledge, Cisco Networking Academy also offers three Linux courses. [NDG Linux Essentials](https://www.netacad.com/courses/os-it/ndg-linux-essentials) is perfect for beginners looking to grasp the basics, while the [NDG Linux Series](https://www.netacad.com/courses/os-it/ndg-linux-I) offers beginners more rigorous in-depth coverage.

[[](https://www.netacad.com/courses/linux-essentials/)](https://www.netacad.com/courses/linux-essentials/" \t "_blank)

The NDG Linux Essentials course is designed to prepare you for the Linux Professional Institute Linux Essentials Professional Development

Certificate, which validates a demonstrated understanding of:

* FOSS, the various communities, and licenses
* Knowledge of open source applications in the workplace as they relate to closed source equivalents
* Basic concepts of hardware, processes, programs and the components of the Linux Operating System
* How to work on the command line and with files
* How to create and restore compressed backups and archives
* System security, users/groups and file permissions for public and private directories
* How to create and run simple scripts

To obtain the Linux Essentials Professional Development Certificate you must pass Linux Essentials (LPI-010) which covers:

* The Linux community and a career in open source
* Finding your way on a Linux system
* The power of the command line
* The Linux operating system
* Security and file permissions

The Linux Essentials Professional Development Certificate is the beginning of your path to becoming a Linux certified professional. Information about the Linux Professional Institute certifications can be found by going to [http://www.lpi.org](http://www.lpi.org/).

Do not be concerned if you have little or no Linux experience. This course is the perfect starting place designed to teach all of the concepts from square one. However, if you do not find this material challenging enough, consider starting with NDG Introduction to Linux I, a more rigorous introductory course.

### **[NDG Linux Series](https://www.netacad.com/courses/ndg-intro-linux/" \t "_blank)**

[[](https://www.netacad.com/courses/ndg-intro-linux/)](https://www.netacad.com/courses/ndg-intro-linux/" \t "_blank)

The NDG Linux Series is designed to prepare you for the Linux Professional Institute Level 1 Certification. LPIC-1 is a Linux Server Professional certification for Linux administrators that confirms your ability to do the following tasks:

* Work at the Linux Command line
* Perform easy maintenance tasks: help users, add users to a larger system, backup and restore, shutdown and reboot.
* Install and configure a workstation (including X) and connect it to a LAN, or a standalone PC to the internet.

To obtain the LPIC-1 certification you must pass Exam 101 and 102. NDG Introduction to Linux I is designed to prepare you for Exam 101, which covers:

* System Architecture
* Linux Installation and Package Management
* GNU and Unix Commands
* Devices, Linux Filesystems, Filesystem Hierarchy Standard

NDG Introduction to Linux II aligns to the Exam 102 objectives, which cover:

* Shells, Scripting and Data Management
* Interfaces and Desktops
* Administrative Tasks
* Essential System Services
* Networking Fundamentals
* Security

The LPIC-1 certification is the first of three LPI professional certifications. Information about all Linux Professional Institute certifications is available at: [http://www.lpi.org](http://www.lpi.org/).

### **[IT Essentials](https://www.netacad.com/courses/it-essentials/" \t "_blank)**

For more on fundamental computer and career skills for entry-level IT jobs, check out the rest of [IT Essentials](https://www.netacad.com/courses/it-essentials/).

[[](https://www.netacad.com/courses/it-essentials/)](https://www.netacad.com/courses/it-essentials/" \t "_blank)

IT Essentials covers fundamental computer and career skills for entry-level IT jobs. The IT Essentials curriculum includes hands-on labs that provide practical experience. Virtual tools help you hone your troubleshooting skills and practice what you learn.

* Develop working knowledge of how computers operate
* Develop critical thinking and complex problem-solving skills using hands-on labs and virtual learning tools
* Apply skills and procedures to install and upgrade hardware and software and troubleshoot systems
* Practice what you learn on real equipment and using the Cisco Packet Tracer simulation tool
* Get immediate feedback on your work through built-in quizzes and tests
* Get connected to the global Cisco Networking Academy community