1. ls

The ls command - the list command - functions in the Linux terminal to show all of the major directories filed under a given file system. For example, the command:

ls /applications

...will show the user all of the folders stored in the overall applications folder.

The ls command is used for viewing files, folders and directories.

2. cd

The cd command - change directory - will allow the user to change between file directories. As the name command name suggest, you would use the cd command to circulate between two different directories. For example, if you wanted to change from the home directory to the Arora directory, you would input the following command:

cd/arora/applications

As you might have noted, the path name listed lists in reverse order. Logically **cd/arora/applications** reads change to the arora directory which is stored in the applications directory. All Linux commands follow a logical path.

3. mv

The mv command - move - allows a user to move a file to another folder or directory. Just like dragging a file located on a PC desktop to a folder stored within the "Documents" folder, the mv command functions in the same manner. An example of the mv command is:

mv/arora/applications/majorapps /arora/applications/minorapps

The first part of the command **mv/arora/applications/majorapps** lists the application to be moved. In this case, arora. The second part of the command **/arora/applications/minorapps** lists where arora will be moved to - from majorapps to minorapps.

4. man

The man command - the manual command - is used to show the manual of the inputted command. Just like a film on the nature of film, the man command is the meta command of the Linux CLI. Inputting the man command will show you all information about the command you are using. An example:

man cd

The inputting command will show the manual or all relevant information for the change directory command.

5. mkdir

The mkdir - make directory - command allows the user to make a new directory. Just like making a new directory within a PC or Mac desktop environment, the mkdir command makes new directories in a Linux environment. An example of the mkdir command

mkdir testdirectory

The example command made the directory "testdirectory".

6. rmdir

The rmdir - remove directory - command allows the user to remove an existing command using the Linux CLI.

An example of the rmdir command:

rmdir testdirectory

7. touch

The touch command - a.k.a. the make file command - allows users to make files using the Linux CLI. Just as the mkdir command makes directories, the touch command makes files. Just as you would make a .doc or a .txt using a PC desktop, the touch command makes empty files. An example of the touch command:

touch testfile.txt

The example touch command effectively created the file testfile.txt. As noted by the extension, the file created is a .txt or text file. To equate, a .txt file in Linux is akin to a .txt notebook file within a Windows or Mac OS.

8. rm

The rm command - remove - like the rmdir command is meant to remove files from your Linux OS. Whereas the rmdir command will remove directories and files held within, the rm command will delete created files. An example of the rm command:

rm testfile.txt

The aforementioned command removed testfile.txt. Interestingly, whereas the rmdir command will only delete an empty directory, the rm command will remove both files and directories with files in it. This said, the rm command carries more weight than the rmdir command and should be used with more specificity.

9. locate

The locate - a.k.a. find - command is meant to find a file within the Linux OS. If you don't know the name of a certain file or you aren't sure where the file is saved and stored, the locate command comes in handy. A locate command example:

locate -i *red*housecity***

The stated command will locate an file with the a file name containing "Red", "House" and "City". A note on the input: the use of "-i" tells the system to search for a file unspecific of capitalization (Linux functions in lower case). The use of the asterik "*" signifies searching for a wildcard. A wildcard tells the system to pull any and all files containing the search criteria.

By specifying -i with wildcards, the locate CLI command will pull back all files containing your search criteria effectivley casting the widest search net the system will allow.

10. clear

The clear command does exactly what it says. When your Linux CLI gets all mucked up with various readouts and information, the clear command clears the screen and wipes the board clean. Using the clear command will take the user back to the start prompt of whatever directory you are currently operating in. To use the clear command simply type **clear**.

---

1. alias

The alias command lets you give your own name to a command or sequence of commands. You can then type your short name, and the shell will execute the command or sequence of commands for you.

```
alias cls=clear
```

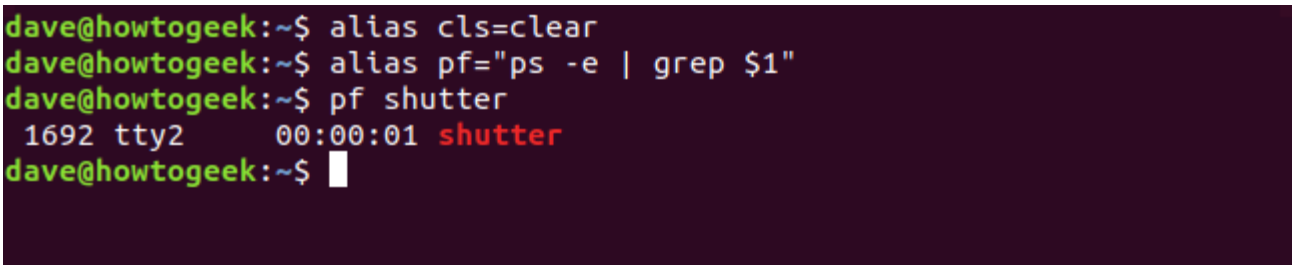
This sets up an alias called `cls` . It will be another name for `clear` . When you type `cls`, it will clear the screen just as though you had typed `clear` . Your alias saves a few keystrokes, sure. But, if you frequently move between Windows and Linux command line, you can find yourself typing the Windows `cls` command on a Linux machine that doesn't know what you mean. Now it will know.

Aliases can be much more intricate than that simple example. Here's an alias called `pf` (for process find) that is just a little more complex. Note the use of quotation marks around the command sequence. This is required if the command sequence has spaces in it. This alias uses the `ps` command to list the running processes and then pipes them through the `grep` command. The `grep` command looks for entries in the output from `ps` that match the command line parameter `$1` .

```
alias pf="ps -e | grep $1"
```

If you wanted to discover the process ID (PID) of the `shutter` process—or to find out if `shutter` was even running—you could use the alias like this. Type `pf`, a space, and the name of the process you are interested in:

```
pf shutter
```



Aliases defined on the command line will die with the terminal window. When you close it, they are gone. To make your aliases always be available to you, add them to the `.bash_aliases` file in your home directory.

2. cat

The `cat` command (short for “concatenate”) lists the contents of files to the terminal window. This is faster than opening the file in an editor, and there’s no chance you can accidentally alter the file. To read the contents of your `.bash_log_out` file, type the following command while the home directory is your current working directory, as it is by default:

```
cat .bash_logout
```

```
dave@howtogeek:~$ cat .bash_logout
# ~/.bash_logout: executed by bash(1) when login shell exits.

# when leaving the console clear the screen to increase privacy

if [ "$SHLVL" = 1 ]; then
    [ -x /usr/bin/clear_console ] && /usr/bin/clear_console -q
fi
dave@howtogeek:~$
```

With files longer than the number of lines in your terminal window, the text will whip past too fast for you to read. You can pipe the output from `cat` through `less` to make the process more manageable. With `less` you can scroll forward and backward through the file using the Up and Down Arrow keys, the PgUp and PgDn keys, and the Home and End keys. Type `q` to quit from less.

```
cat .bashrc | less
```

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-
doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the histor
y.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
:
```

3. cd

The `cd` command changes your current directory. In other words, it moves you to a new place in the filesystem.

If you are changing to a directory that is within your current directory, you can simply type `cd` and the name of the other directory.

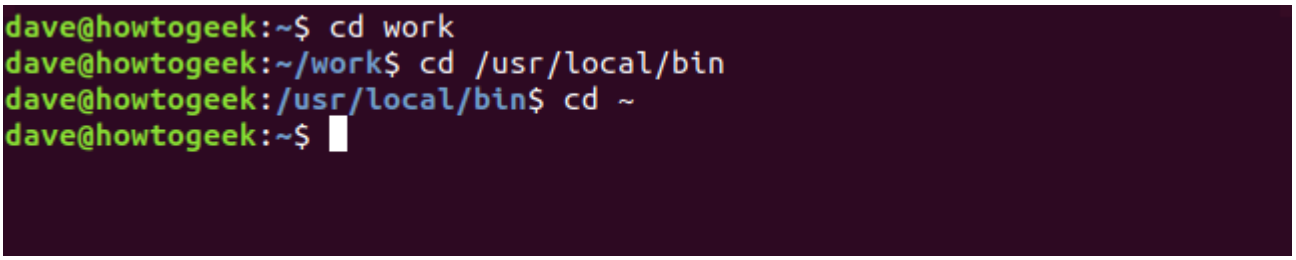
```
cd work
```

If you are changing to a directory elsewhere within the filesystem directory tree, provide the path to the directory with a leading /.

```
cd /usr/local/bin
```

To quickly return to your home directory, use the ~ (tilde) character as the directory name.

```
cd ~
```

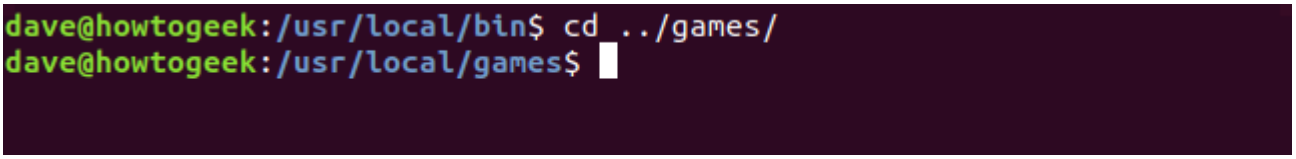


Here's another trick: You can use the double dot symbol `..` to represent the parent of the current directory. You can type the following command to go up a directory:

```
cd ..
```

Imagine you are in a directory. The parent directory has other directories in it, as well as the directory you're currently in. To change into one of those other directories, you can use the `..` symbol to shorten what you have to type.

```
cd ../games
```



4. chmod

The `chmod` command sets the file permissions flags on a file or folder. The flags define who can read, write to or execute the file. When you list files with the `-l` (long format) option you'll see a string of characters that look like

```
-rwxrwxrwx
```

If the first character is a `-` the item is a file, if it is a `d` the item is a directory. The rest of the string is three sets of three characters. From the left, the first three represent the file permissions of the *owner*, the middle three represent the file permissions of the *group* and the rightmost three characters represent the permissions for *others*. In each set, an `r` stands for read, a `w` stands for write, and an `x` stands for execute.

If the `r`, `w`, or `x` character is present that file permission is granted. If the letter is not present and a `-` appears instead, that file permission is not granted.

One way to use `chmod` is to provide the permissions you wish to give to the owner, group, and others as a 3 digit number. The leftmost digit represents the owner. The middle digit represents the group. The rightmost digit represents the others. The digits you can use and what they represent are listed here:

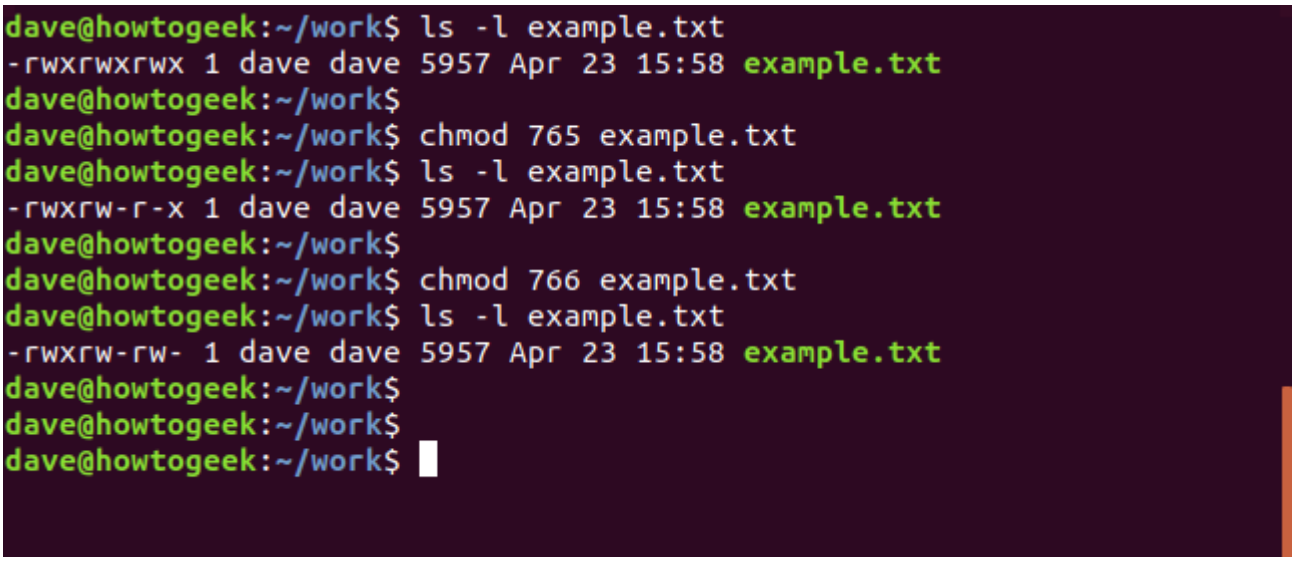
- **0:** No permission
- **1:** Execute permission
- **2:** Write permission
- **3:** Write and execute permissions
- **4:** Read permission
- **5:** Read and execute permissions
- **6:** Read and write permissions

- **7:** Read, write and execute permissions

Looking at our example.txt file, we can see that all three sets of characters are `rwx`. That means everyone has read, write and execute rights with the file.

To set the permission to be read, write, and execute (7 from our list) for the *owner;* read and write (6 from our list) for the *group;* and read and execute (5 from our list) for the *others* we'd need to use the digits 765 with the `chmod` command:

```
chmod -R 765 example.txt
```



To set the permission to be read, write and execute (7 from our list) for the *owner*, and read and write (6 from our list) for the *group* and for the *others* we'd need to use the digits 766 with the `chmod` command:

```
chmod 766 example.txt
```

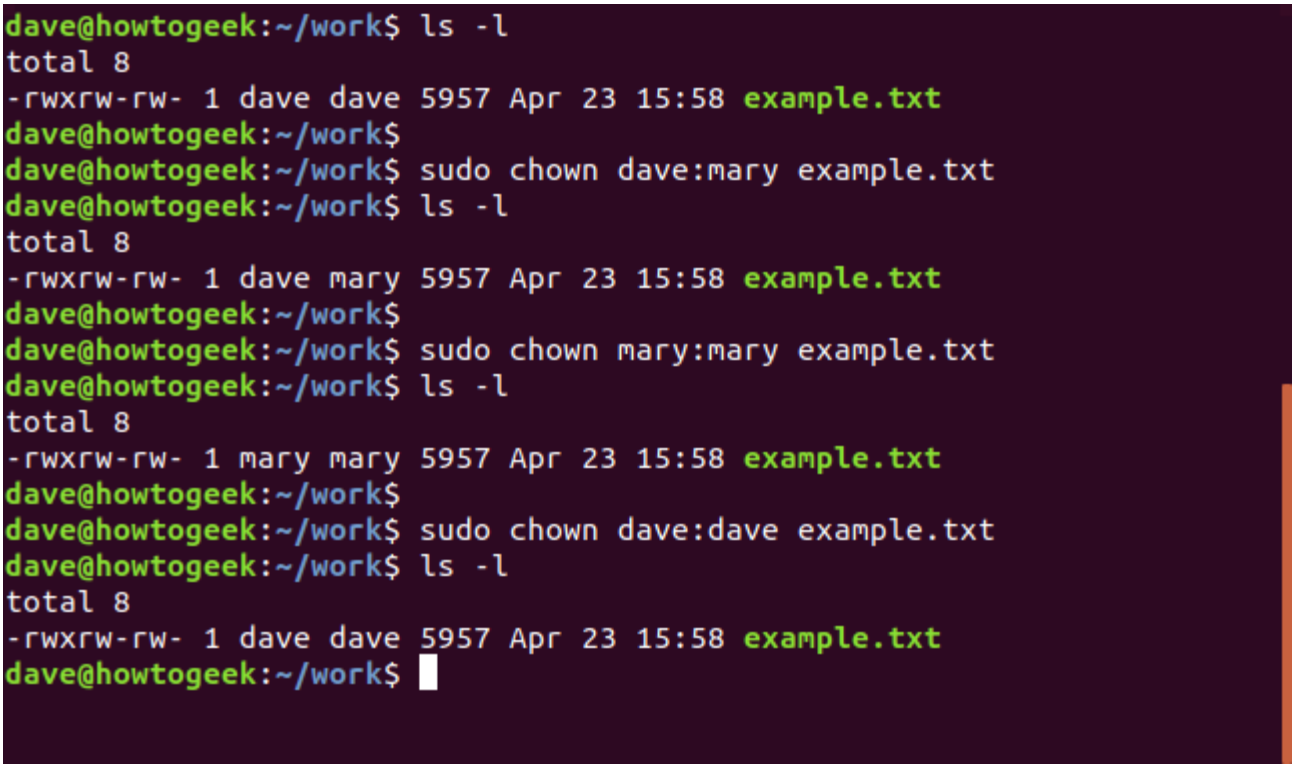
5. chown

The `chown` command allows you to change the owner and group owner of a file. Listing our example.txt file with `ls -l` we can see `dave dave` in the file description. The first of these indicates the name of the file owner, which in this case is the user `dave`. The second entry shows that the name of the group owner is also `dave`. Each user has a

default group created when the user is created. That user is the only member of that group. This shows that the file is not shared with any other groups of users.

You can use `chown` to change the owner or group, or both of a file. You must provide the name of the owner and the group, separated by a `:` character. You will need to use `sudo`. To retain dave as the owner of the file but to set mary as the group owner, use this command:

```
sudo chown dave:mary example.txt
```



To change both the owner and the group owner to mary, you would use the following command;

```
sudo chown mary:mary example.txt
```

To change the file so that dave is once more the file owner and the group owner, use this command:

```
sudo chown dave:dave example.txt
```

6. curl

The `curl` command is a tool to retrieve information and files from Uniform Resource Locators (URLs) or internet addresses.

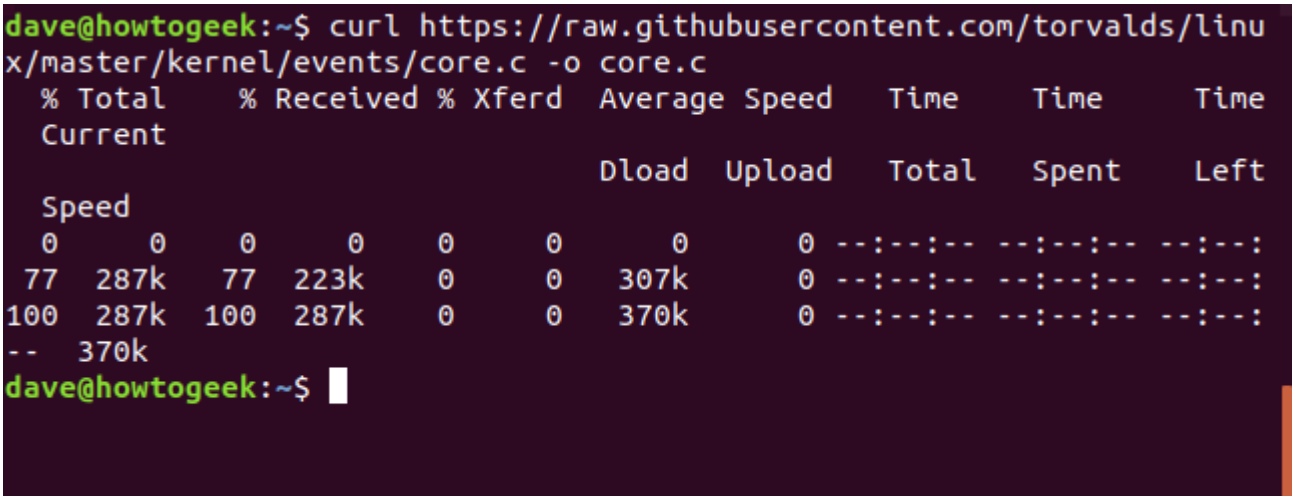
The `curl` command may not be provided as a standard part of your Linux distribution. Use `apt-get` to install this package onto your system if you're using Ubuntu or another Debian-based distribution. On other Linux distributions, use your Linux distribution's package management tool instead.

```
sudo apt-get install curl
```

Suppose you want to retrieve a single file from a GitHub repository. There is no officially supported way to this. You're forced to clone the entire repository. With `curl` however, we can retrieve the file we want on its own.

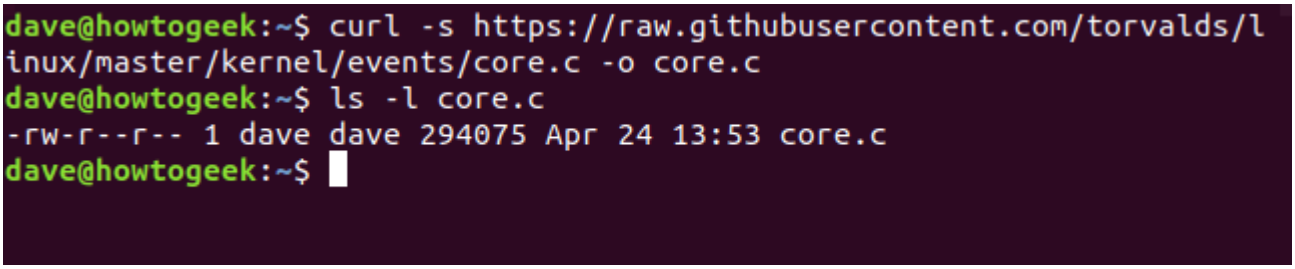
This command retrieves the file for us. Note that you need to specify the name of the file to save it in, using the `-o` (output) option. If you do not do this, the contents of the file are scrolled rapidly in the terminal window but not saved to your computer.

```
curl
https://raw.githubusercontent.com/torvalds/linux/master/kernel/events/core.c
-o core.c
```



If you don't want to see the download progress information use the `-s` (silent) option.

```
curl -s
https://raw.githubusercontent.com/torvalds/linux/master/kernel/events/core.c
-o core.c
```

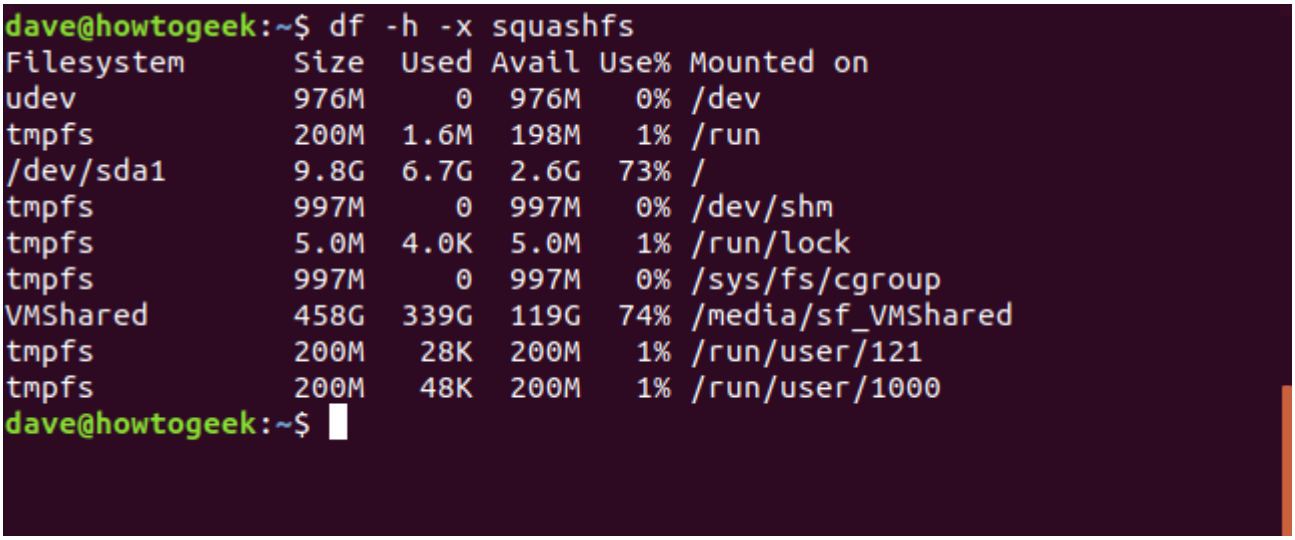


7. df

The `df` command shows the size, used space, and available space on the mounted filesystems of your computer.

Two of the most useful options are the `-h` (human readable) and `-x` (exclude) options. The human-readable option displays the sizes in Mb or Gb instead of in bytes. The exclude option allows you to tell `df` to discount filesystems you are not interested in. For example, the `squashfs` pseudo-filesystems that are created when you install an application with the `snap` command.

```
df -h -x squashfs
```



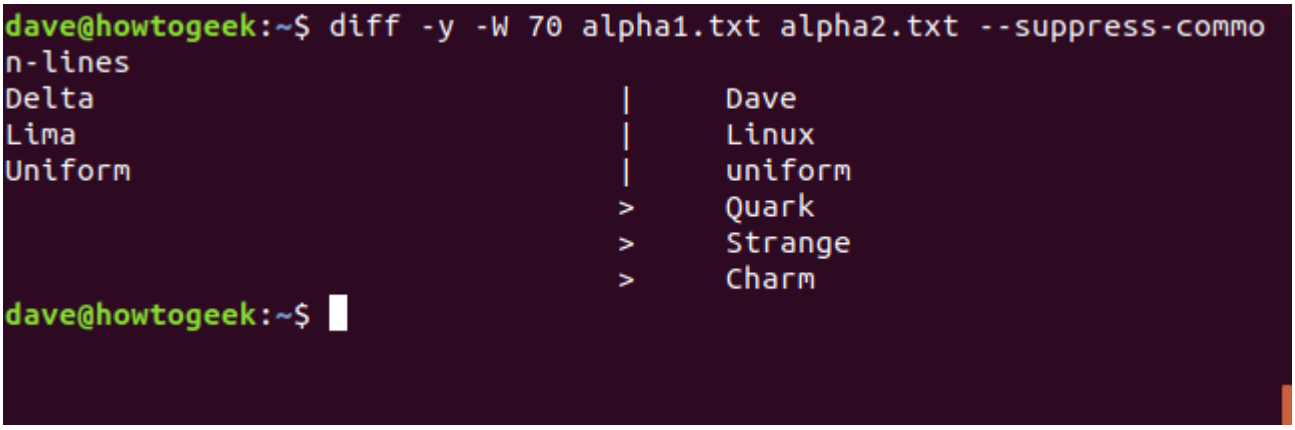
RELATED: ***How to View Free Disk Space and Disk Usage From the Linux Terminal***

8. diff

The `diff` command compares two text files and shows the differences between them. There are many options to tailor the display to your requirements.

The `-y` (side by side) option shows the line differences side by side. The `-w` (width) option lets you specify the maximum line width to use to avoid wraparound lines. The two files are called alpha1.txt and alpha2.txt in this example. The `--suppress-common-lines` prevents `diff` from listing the matching lines, letting you focus on the lines which have differences.

```
diff -y -W 70 alpha1.txt alpha2.txt --suppress-common-lines
```



RELATED: ***How to Compare Two Text Files in the Linux Terminal***

9. echo

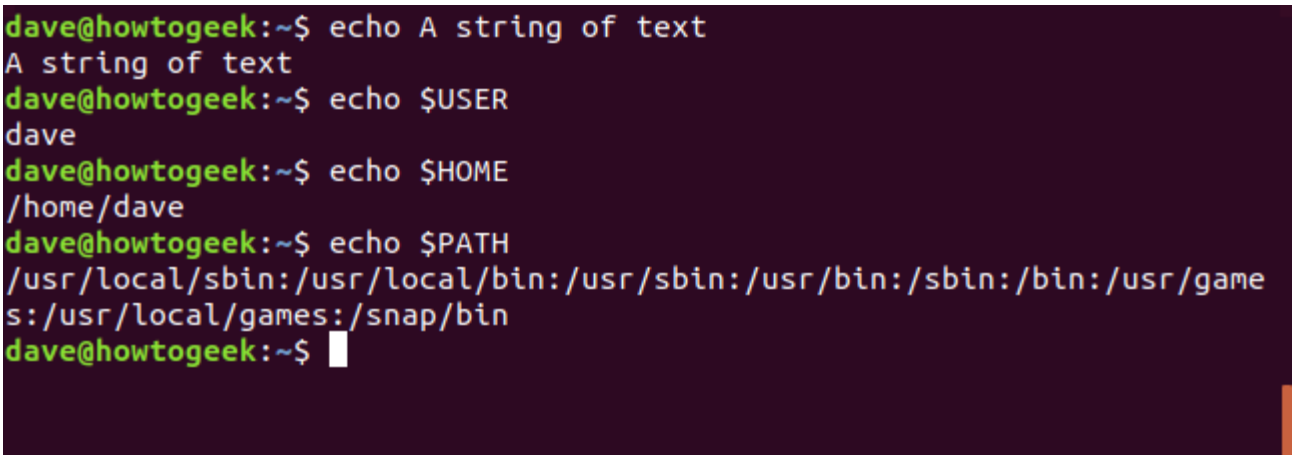
The `echo` command prints (echoes) a string of text to the terminal window.

The command below will print the words “A string of text” on the terminal window.

```
echo A string of text
```

The `echo` command can show the value of environment variables, for example, the `$USER`, `$HOME`, and `$PATH` environment variables. These hold the values of the name of the user, the user’s home directory, and the path searched for matching commands when the user types something on the command line.

```
echo $USER

echo $HOME

echo $PATH
```



The following command will cause a bleep to be issued. The `-e` (escape code) option interprets the escaped a character as a 'bell' character.

```
echo -e "\a"
```

The `echo` command is also invaluable in shell scripts. A script can use this command to generate visible output to indicate the progress or results of the script as it is executed.

10. exit

The exit command will close a terminal window, end the execution of a shell script, or log you out of an SSH remote access session.

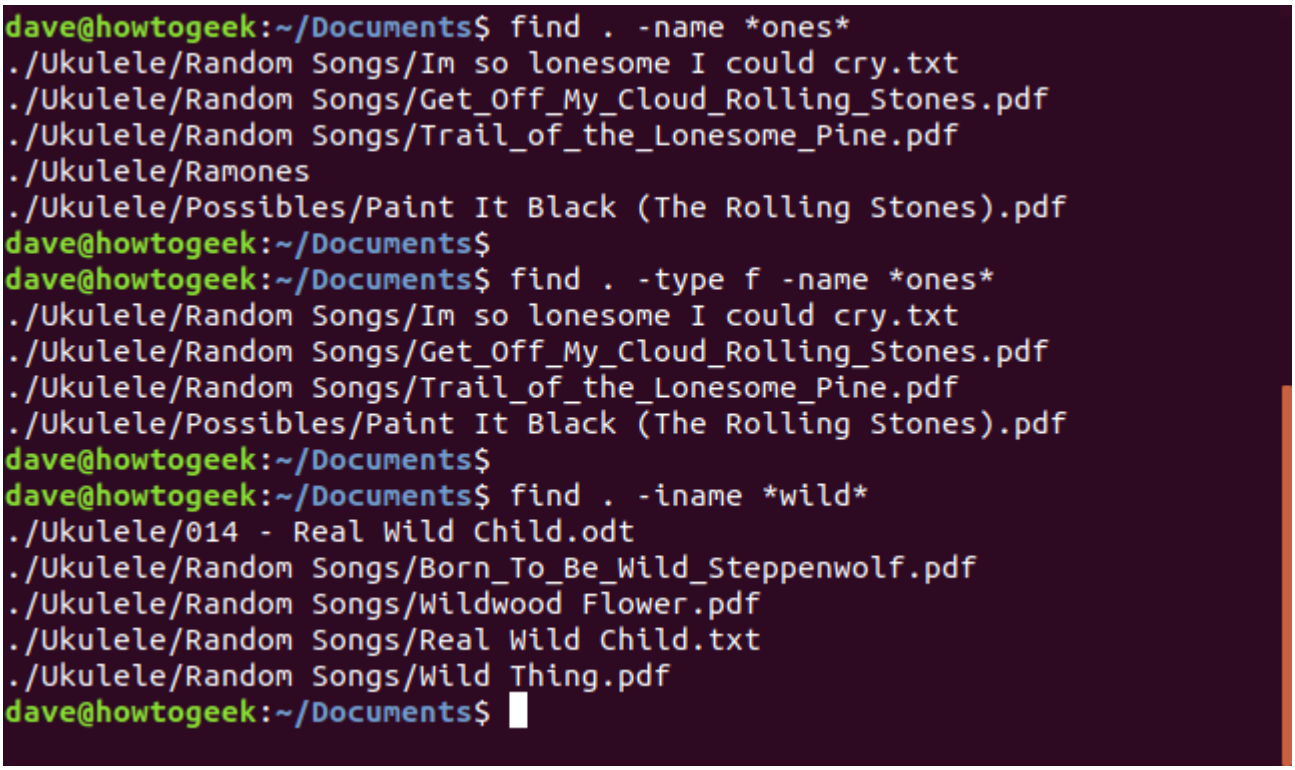
```
exit
```

11. find

Use the `find` command to track down files that you know exist if you can't remember where you put them. You must tell `find` where to start searching from and what it is looking for. In this example, the `.` matches the current folder and the `-name` option tells `find` to look for files with a name that matches the search pattern.

You can use wildcards, where `*` represents any sequence of characters and `?` represents any single character. We're using `*ones*` to match any file name containing the sequence "ones." This would match words like bones, stones, and lonesome.

```
find . -name *ones*
```



As we can see, `find` has returned a list of matches. One of them is a directory called Ramones. We can tell `find` to restrict the search to files only. We do this using the `-type` option with the `f` parameter. The `f` parameter stands for files.

```
find . -type f -name *ones*
```

If you want the search to be case insensitive use the `-iname` (insensitive name) option.

```
find . -iname *wild*
```

12. finger

The `finger` command gives you a short dump of information about a user, including the time of the user's last login, the user's home directory, and the user account's full name.

```
dave@howtogeek:~$ finger mary
Login: mary                             Name: Mary Quinn
Directory: /home/mary                   Shell: /bin/bash
Last login Wed Apr 24 16:00 (EDT) on pts/2 from 192.168.4.27
No mail.
No Plan.
dave@howtogeek:~$
```

13. free

The `free` command gives you a summary of the memory usage with your computer. It does this for both the main Random Access Memory (RAM) and swap memory. The `-h` (human) option is used to provide human-friendly numbers and units. Without this option, the figures are presented in bytes.

```
free -h
```

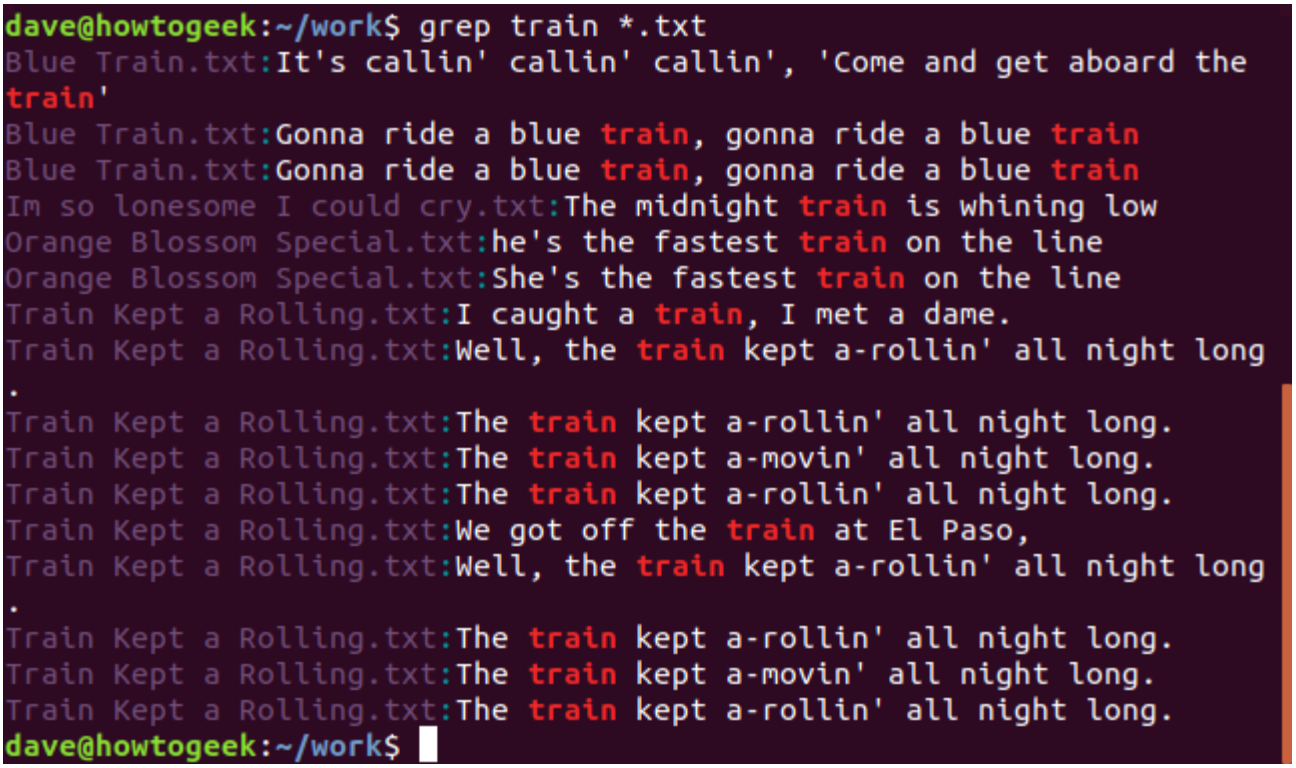
```
dave@howtogeek:~/Documents$ free -h
              total        used        free      shared  buff/cache
available
Mem:           1.9G        1.1G         74M        321M        771M
     369M
Swap:          472M        6.5M        465M
dave@howtogeek:~/Documents$
```

14. grep

The `grep` utility searches for lines which contain a search pattern. When we looked at the alias command, we used `grep` to search through the output of another program, `ps` . The `grep` command can also search the contents of files. Here we're searching for the word "train" in all text files in the current directory.

```
grep train *.txt
```

The output lists the name of the file and shows the lines that match. The matching text is highlighted.



The functionality and sheer usefulness of `grep` definitely warrants you checking out its man page.

15. groups

The `groups` command tells you which groups a user is a member of.

```
groups dave

groups mary
```

```
dave@howtogeek:~/work$ groups dave
dave : dave adm cdrom sudo dip plugdev lpadmin sambashare vboxsf geek
dave@howtogeek:~/work$
dave@howtogeek:~/work$ groups mary
mary : mary sudo
dave@howtogeek:~/work$
```

16. gzip

The `gzip` command compresses files. By default, it removes the original file and leaves you with the compressed version. To retain both the original and the compressed version, use the `-k` (keep) option.

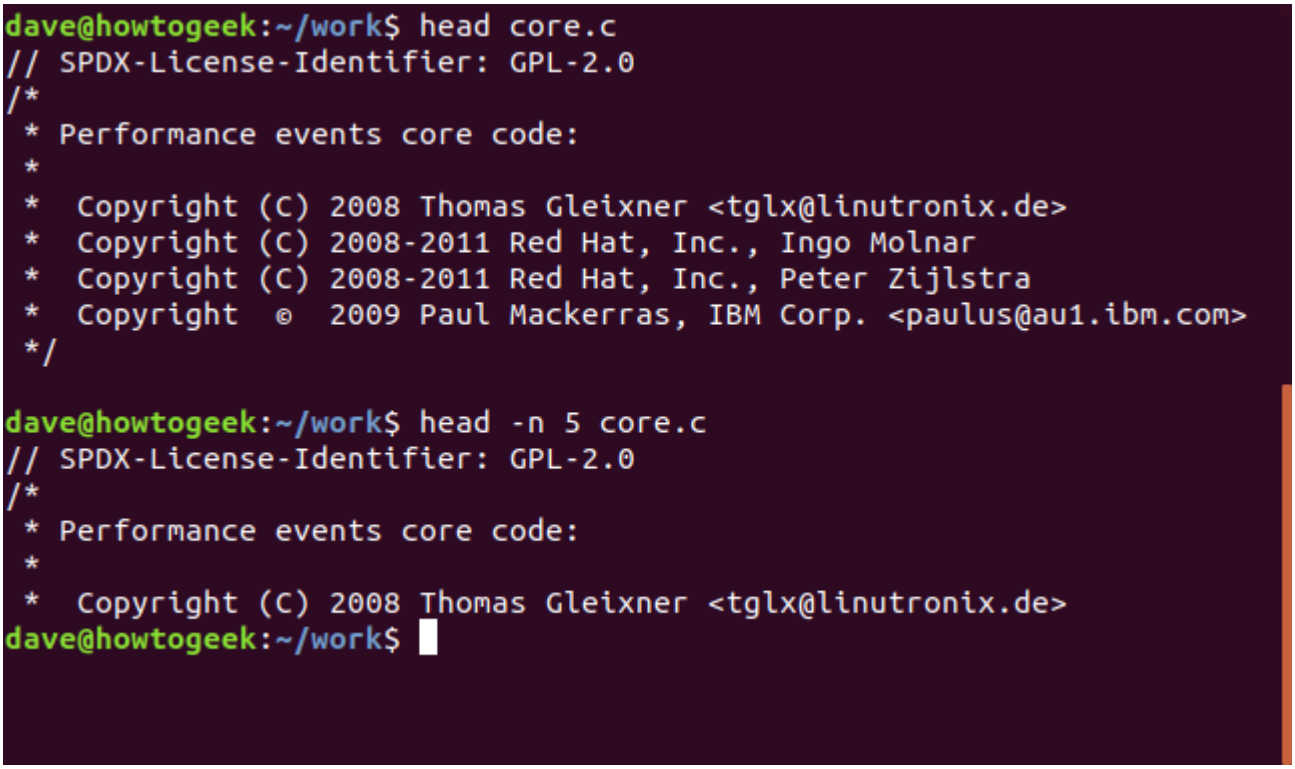
```
gzip -k core.c
```

```
dave@howtogeek:~/work$ gzip -k core.c
dave@howtogeek:~/work$ ls -l core*
-rw-r--r-- 1 dave dave 294075 Apr 24 14:33 core.c
-rw-r--r-- 1 dave dave  73416 Apr 24 14:33 core.c.gz
dave@howtogeek:~/work$
dave@howtogeek:~/work$
```

17. head

The `head` command gives you a listing of the first 10 lines of a file. If you want to see fewer or more lines, use the `-n` (number) option. In this example, we use `head` with its default of 10 lines. We then repeat the command asking for only five lines.

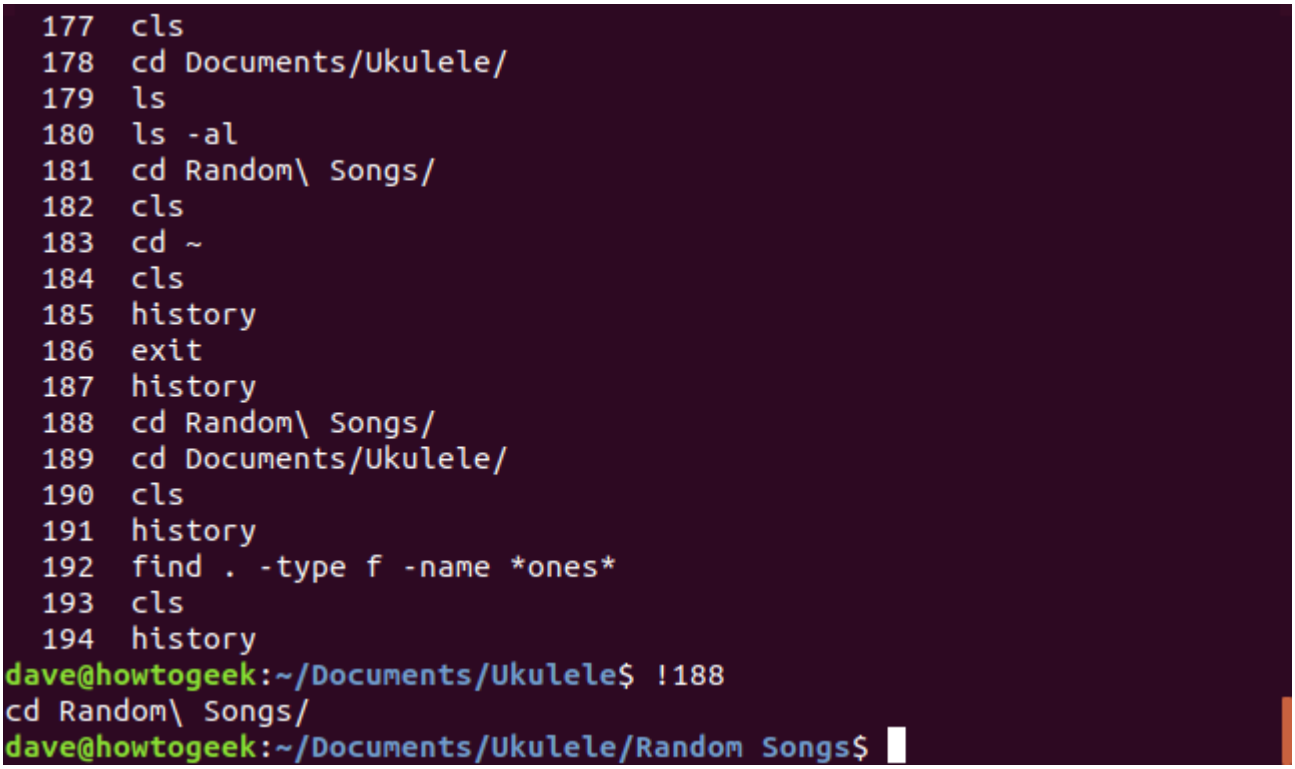
```
head -core.c

head -n 5 core.c
```



18. history

The history command lists the commands you have previously issued on the command line. You can repeat any of the commands from your history by typing an exclamation point ! and the number of the command from the history list.

```
!188
```



Typing two exclamation points repeats your previous command.

```
!!
```

19. kill

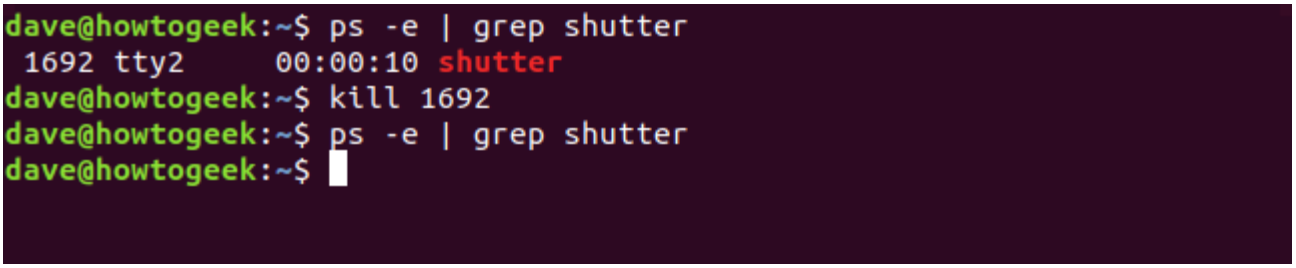
The `kill` command allows you to terminate a process from the command line. You do this by providing the process ID (PID) of the process to `kill`. Don't kill processes willy-nilly. You need to have a good reason to do so. In this example, we'll pretend the `shutter` program has locked up.

To find the PID of `shutter` we'll use our `ps` and `grep` trick from the section about the `alias` command, above. We can search for the `shutter` process and obtain its PID as follows:

```
ps -e | grep shutter.
```

Once we have determined the PID—1692 in this case—we can kill it as follows:

```
kill 1692
```



20. less

The `less` command allows you to view files without opening an editor. It's faster to use, and there's no chance of you inadvertently modifying the file. With `less` you can scroll forward and backward through the file using the Up and Down Arrow keys, the PgUp and PgDn keys and the Home and End keys. Press the Q key to `quit` from `less`.

To view a file provide its name to `less` as follows:

```
less core.c
```

You can also pipe the output from other commands into `less`. To see the output from `ls` for a listing of your entire hard drive, use the following command:

```
ls -R / | less
```

Use `/` to search forward in the file and use `?` to search backward.

21. ls

This might be the first command the majority of Linux users meet. It lists the files and folders in the directory you specify. By default, `ls` looks in the current directory. There are a great many options you can use with `ls` , and we strongly advise reviewing its the man page. Some common examples are presented here.

To list the files and folders in the current directory:

```
ls
```

To list the files and folders in the current directory with a detailed listing use the `-l` (long) option:

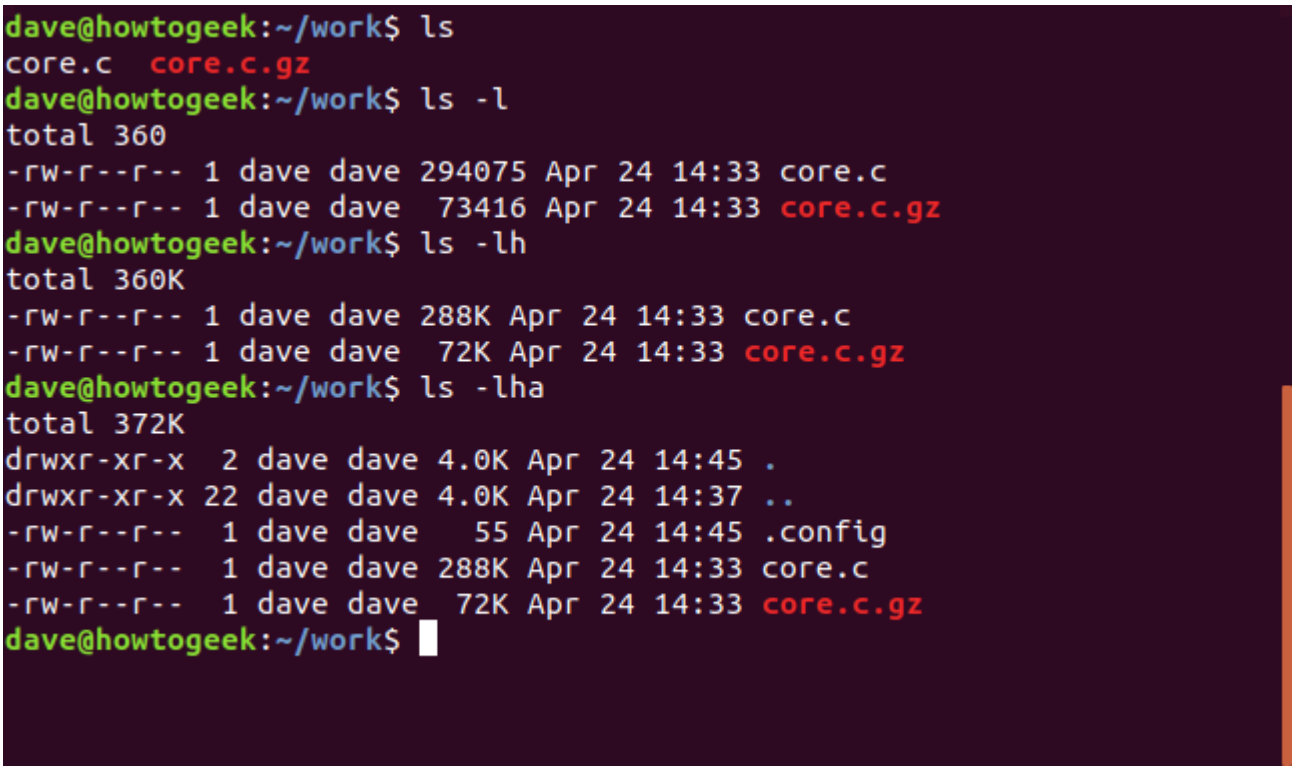
```
ls -l
```

To use human-friendly file sizes include the `-h` (human) option:

```
ls -lh
```

To include hidden files use the `-a` (all files) option:

```
ls -lha
```



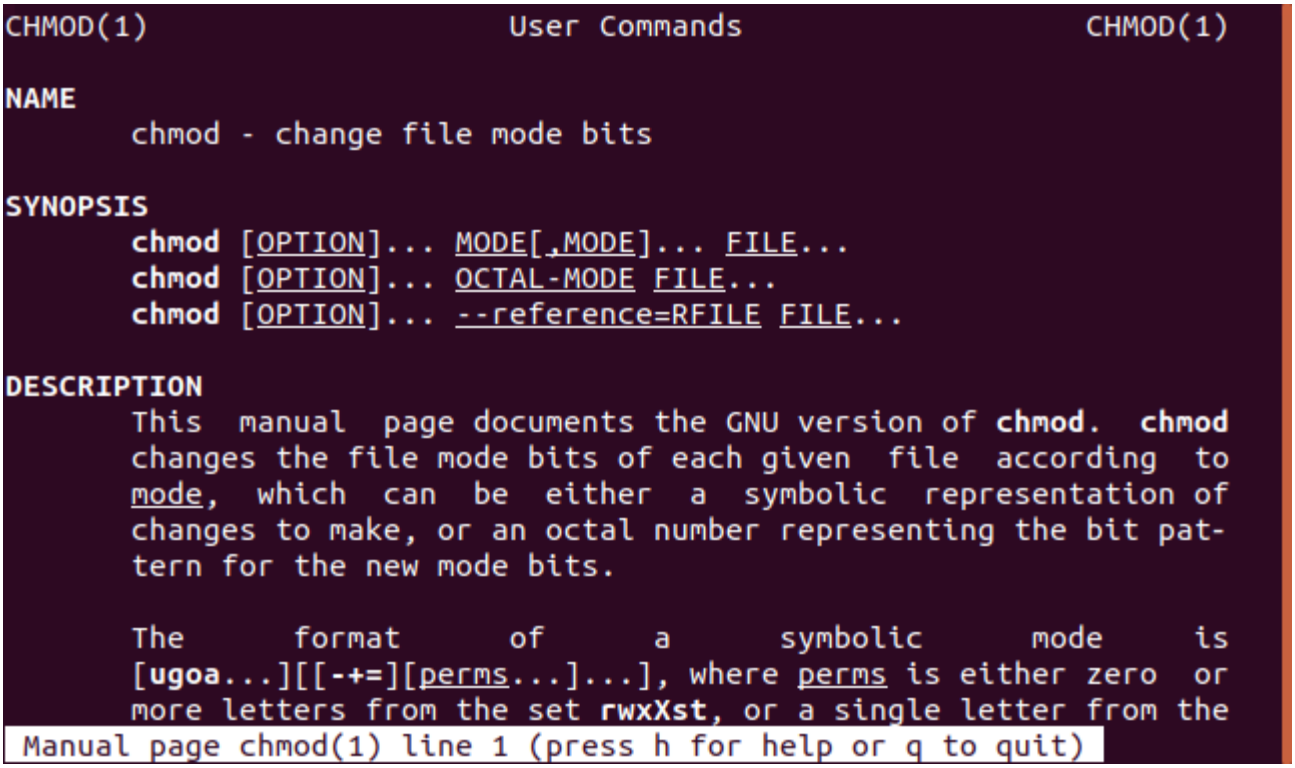
22. man

The man command displays the “man pages” for a command in `less` . The man pages are the user manual for that command. Because `man` uses `less` to display the man pages, you can use the search capabilities of `less`.

For example, to see the man pages for `chown`, use the following command:

```
man chown
```

Use the Up and Down arrow or PgUp and PgDn keys to scroll through the document. Press `q` to quit the man page or press`h` for help.

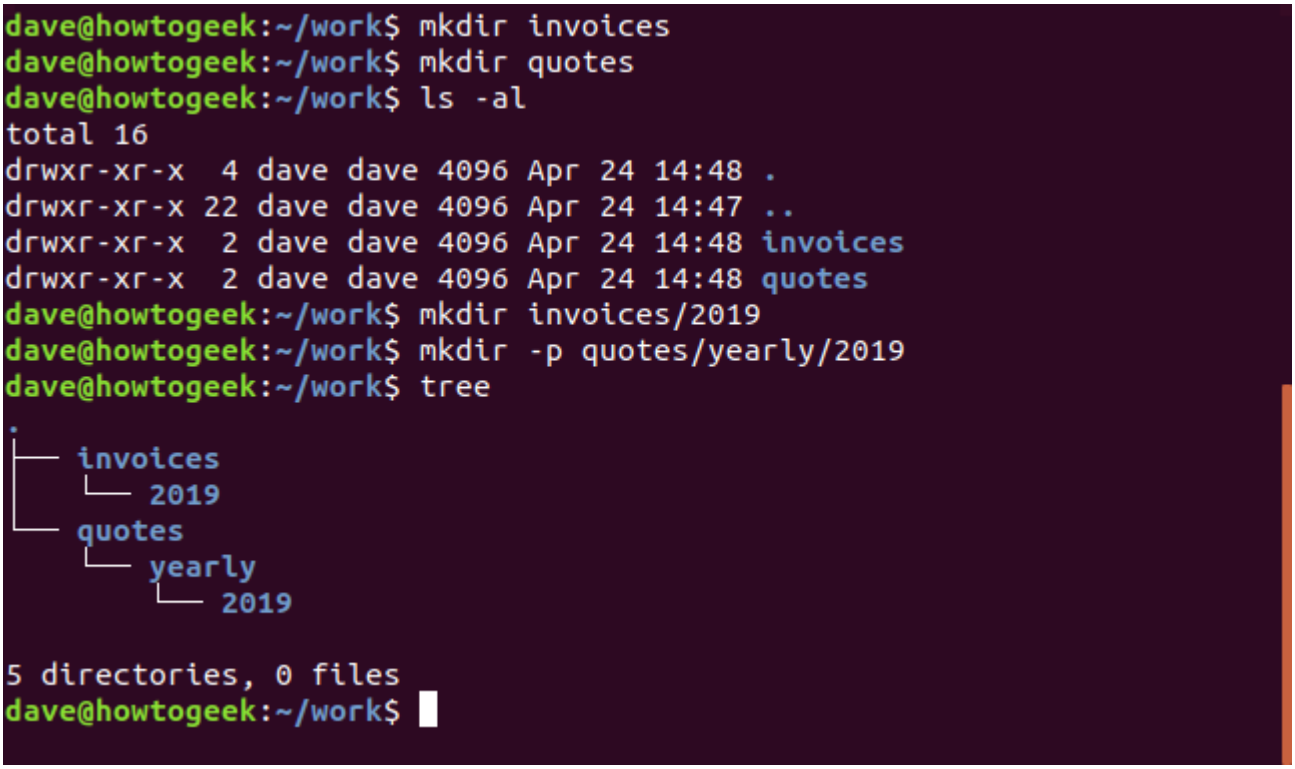


23. mkdir

The `mkdir` command allows you to create new directories in the filesystem. You must provide the name of the new directory to `mkdir`. If the new directory is not going to be within the current directory, you must provide the path to the new directory.

To create two new directories in the current directory called “invoices” and “quotes,” use these two commands:

```
mkdir invoices

mkdir quotes
```



To create a new directory called “2019” inside the “invoices” directory, use this command:

```
mkdir invoices/2109
```

If you are going to create a directory, but its parent directory does not exist, you can use the `-p` (parents) option to have `mkdir` create all of the required parent directories too. In the following command, we are creating the “2019” directory inside the “yearly” directory inside the “quotes” directory. The “yearly” directory does not exist, but we can have `mkdir` create all the specified directories at once:

```
mkdir -p quotes/yearly/2019
```

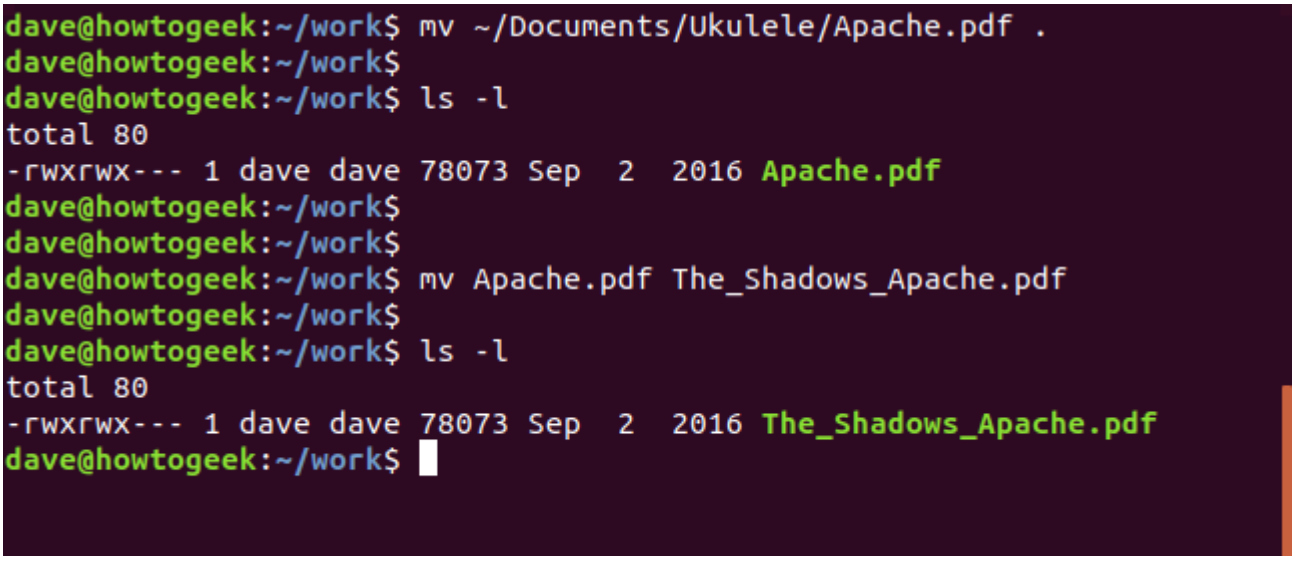
The “yearly” directory is also created.

24. mv

The `mv` command allows you to move files and directories from directory to directory. It also allows you to rename files.

To move a file you must tell `mv` where the file is and where you want it to be moved to. In this example, we're moving a file called `apache.pdf` from the "~/Document/Ukulele" directory and placing it in the current directory, represented by the single `.` character.

```
mv ~/Documents/Ukulele/Apache.pdf .
```



To rename the file, you "move" it into a new file with the new name.

```
mv Apache.pdf The_Shadows_Apache.pdf
```

The file move and rename action could have been achieved in one step:

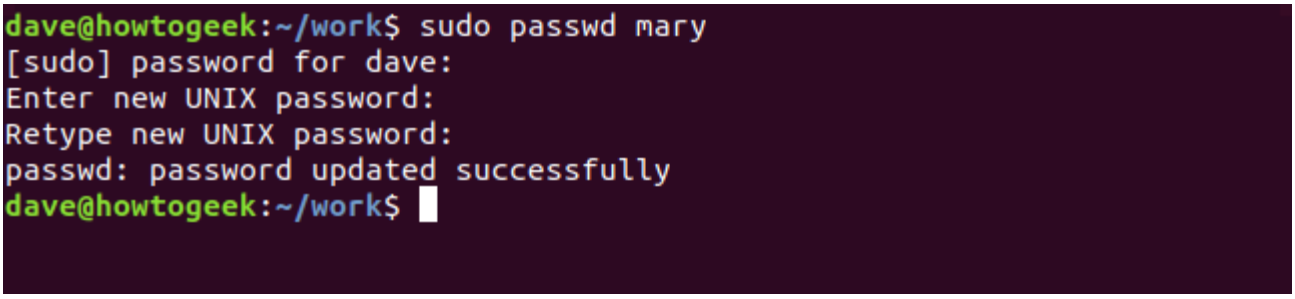
```
mv ~/Documents/Ukulele/Apache.pdf ./The_Shadows_Apache.pdf
```

25. passwd

The `passwd` command lets you change the password for a user. Just type `passwd` to change your own password.

You can also change the password of another user account, but you must use `sudo`. You will be asked to enter the new password twice.

```
sudo passwd mary
```

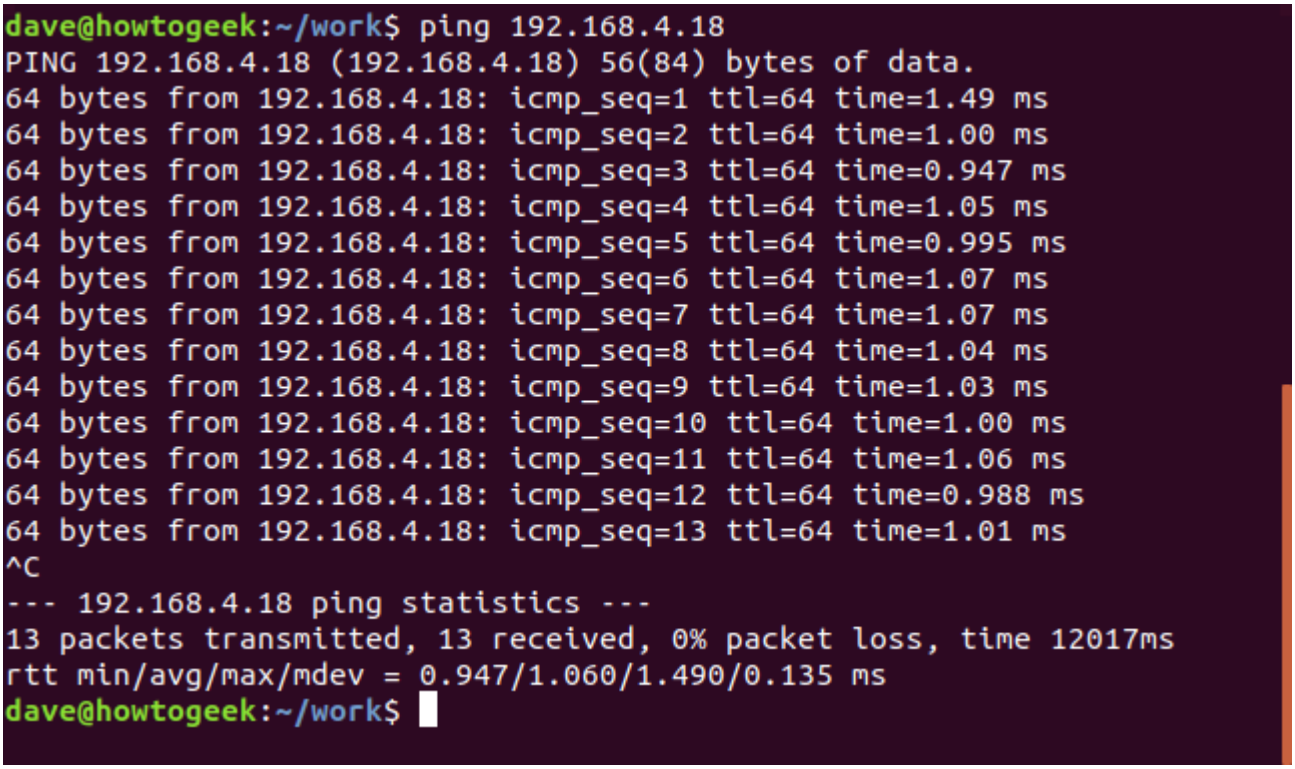


26. ping

The `ping` command lets you verify that you have network connectivity with another network device. It is commonly used to help troubleshoot networking issues. To use `ping`, provide the IP address or machine name of the other device.

```
ping 192.168.4.18
```

The `ping` command will run until you stop it with Ctrl+C.



Here's what's going on here:

- The device at IP address 192.168.4.18 is responding to our ping requests and is sending back packets of 64 bytes.
- The Internet Control Messaging Protocol (ICMP) sequence numbering allows us to check for missed responses (dropped packets).
- The TTL figure is the "time to live" for a packet. Each time the packet goes through a router, it is (supposed to be) decremented by one. If it reaches zero the packet is thrown away. The aim of this is to prevent network loopback problems from flooding the network.
- The time value is the duration of the round trip from your computer to the device and back. Simply put, the lower this time, the better.

To ask `ping` to run for a specific number of ping attempts, use the `-c` (count) option.

```
ping -c 5 192.168.4.18
```

To hear a ping, use the `-a` (audible) option.

```
ping -a 192.168.4.18
```

27. ps

The `ps` command lists running processes. Using `ps` without any options causes it to list the processes running in the current shell.

```
ps
```

To see all the processes related to a particular user, use the `-u` (user) option. This is likely to be a long list, so for convenience pipe it through `less`.

```
ps -u dave | less
```

```
  PID TTY          TIME CMD
 1123 ?        00:00:00 systemd
 1124 ?        00:00:00 (sd-pam)
 1137 ?        00:00:00 gnome-keyring-d
 1141 tty2     00:00:00 gdm-x-session
 1143 tty2     00:00:57 Xorg
 1150 ?        00:00:00 dbus-daemon
 1154 tty2     00:00:00 gnome-session-b
 1254 ?        00:00:00 VBoxClient
 1255 ?        00:00:00 VBoxClient
 1264 ?        00:00:00 VBoxClient
 1265 ?        00:00:00 VBoxClient
 1269 ?        00:00:00 VBoxClient
 1270 ?        00:00:00 VBoxClient
 1274 ?        00:00:00 VBoxClient
 1275 ?        00:00:41 VBoxClient
 1286 ?        00:00:00 ssh-agent
 1288 ?        00:00:00 at-spi-bus-laun
 1293 ?        00:00:00 dbus-daemon
 1296 ?        00:00:00 at-spi2-registr
:
```

To see every process that is running, use the `-e` (every process) option:

```
ps -e | less
```

28. pwd

Nice and simple, the `pwd` command prints the working directory (the current directory) from the root / directory.

```
pwd
```

```
dave@howtogeek:~/work$ pwd
/home/dave/work
dave@howtogeek:~/work$
```

29. shutdown

The shutdown command lets you shut down or reboot your Linux system.

Using `shutdown` with no parameters will shut down your computer in one minute.

shutdown

```
dave@howtogeek:~/work$ shutdown
```

To shut down immediately, use the `now` parameter.

```
shutdown now
```

```
dave@howtogeek:~$ shutdown now
```

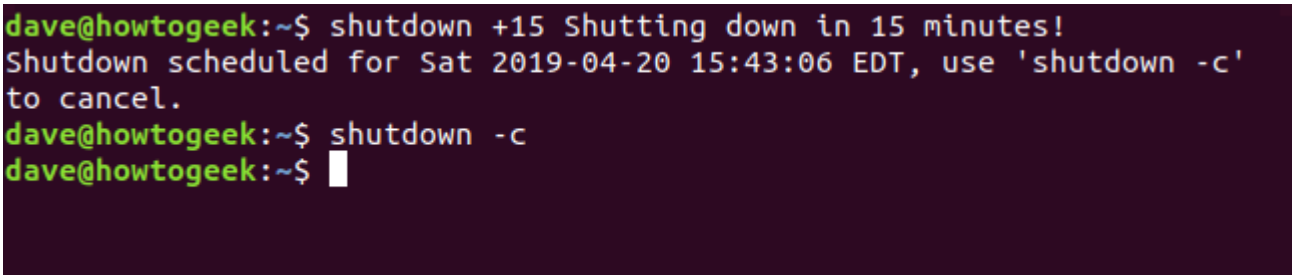
You can also schedule a shutdown and inform any logged in users of the pending shutdown. To let the `shutdown` command know when you want it to shut down, you provide it with a time. This can be a set number of minutes from now, such as `+90` or a precise time, like `23:00`. Any text message you provide is broadcast to logged in users.

```
shutdown 23:00 Shutdown tonight at 23:00, save your work and log out before
then!
```

```
dave@howtogeek:~$ shutdown 23:00 Shutdown tonight at 23:00, save your
work and log out before then!
Shutdown scheduled for Sat 2019-04-20 23:00:00 EDT, use 'shutdown -c'
to cancel.
dave@howtogeek:~$
```

To cancel a shutdown, use the `-c` (cancel) option. Here we have scheduled a shutdown for fifteen minutes time from now—and then changed our minds.

```
shutdown +15 Shutting down in 15 minutes!

shutdown -c
```

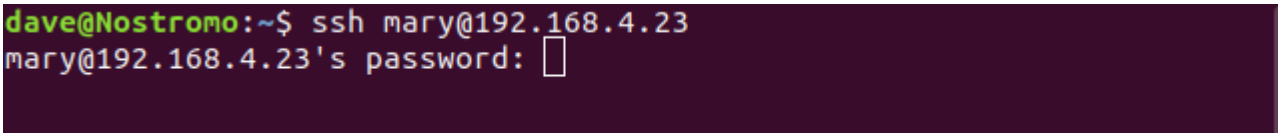


RELATED: ***How to Reboot or Shut Down Linux Using the Command Line***

30. SSH

Use the ssh command to make a connection to a remote Linux computer and log into your account. To make a connection, you must provide your user name and the IP address or domain name of the remote computer. In this example, the user mary is logging into the computer at 192.168.4.23. Once the connection is established, she is asked for her password.

```
ssh mary@192.168.4.23
```



Her user name and password are verified and accepted, and she is logged in. Notice that her prompt has changed from “Nostromo” to “howtogeek.”

Mary issues the `w` command to list the current users on “howtogeek” system. She is listed as being connected from pts/1, which is a pseudo-terminal slave. That is, it is not a terminal directly connected to the computer.

To close the session, mary types `exit` and is returned to the shell on the “Nostromo” computer.

```
w

exit
```

```
mary@howtogeek:~$ w
 15:02:34 up  4:24,  2 users,  load average: 0.17, 0.11, 0.09
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
dave     :0       :0               Tue15   ?xdm?   2:09   0.00s /usr/l
i
mary     pts/1    192.168.4.27     15:02    1.00s  0.05s  0.00s w
mary@howtogeek:~$ exit
logout
Connection to 192.168.4.23 closed.
dave@Nostromo:~$
```

31. sudo

The `sudo` command is required when performing actions that require root or superuser permissions, such as changing the password for another user.

```
sudo passwd mary
```

```
dave@howtogeek:~/work$ sudo passwd mary
[sudo] password for dave:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
dave@howtogeek:~/work$
```

32. tail

The `tail` command gives you a listing of the last 10 lines of a file. If you want to see fewer or more lines, use the `-n` (number) option. In this example, we use `tail` with its default of 10 lines. We then repeat the command asking for only five lines.

```
tail core.c

tail -n 5 core.c
```

```
dave@howtogeek:~/work$ tail core.c
        .attach         = perf_cgroup_attach,
        /*
         * Implicitly enable on dfl hierarchy so that perf events can
         * always be filtered by cgroup2 path as long as perf_event
         * controller is not mounted on a legacy hierarchy.
         */
        .implicit_on_dfl = true,
        .threaded       = true,
};
#endif /* CONFIG_CGROUP_PERF */
dave@howtogeek:~/work$ tail -n 5 core.c
         */
        .implicit_on_dfl = true,
        .threaded       = true,
};
#endif /* CONFIG_CGROUP_PERF */
dave@howtogeek:~/work$
```

33. tar

With the `tar` command, you can create an archive file (also called a tarball) that can contain many other files. This makes it much more convenient to distribute a collection of files. You can also use `tar` to extract the files from an archive file. It is common to ask `tar` to compress the archive. If you do not ask for compression, the archive file is created uncompressed.

To create an archive file, you need to tell `tar` which files to include in the archive file, and the name you wish the archive file to have.

In this example, the user is going to archive all of the files in the Ukulele directory, which is in the current directory.

```
dave@howtogeek:~/work$ ls
Ukulele
dave@howtogeek:~/work$
```

They have used the `-c` (create) option and the `-v` (verbose) option. The verbose option gives some visual feedback by listing the files to the terminal window as they are added to the archive. The `-f` (filename) option is followed by the desired name of the archive. In this case, it is `songs.tar`.

```
tar -cvf songs.tar Ukulele/
```

```
dave@howtogeek:~/work$ ls
Ukulele
dave@howtogeek:~/work$ tar -cvf songs.tar Ukulele/
```

The files are listed to the terminal window as they are added to the archive file.

There are two ways to tell `tar` that you want the archive file to be compressed. The first is with the `-z` (gzip) option. This tells tar to use the `gzip` utility to compress the archive once it has been created.

It is usual to add ".gz" as suffix to this type of archive. That allows anyone who is extracting files from it to know which commands to pass to `tar` to correctly retrieve the files.

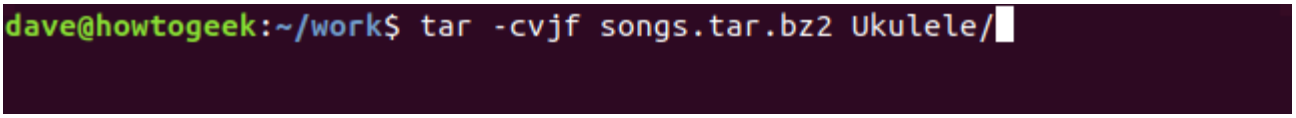
```
tar -cvzf songs.tar.gz Ukulele/
```

```
dave@howtogeek:~/work$ tar -cvzf songs.tar.gz Ukulele/
```

The files are listed to the terminal window as they are added to the archive file as before, but the creation of the archive will take a little longer because of the time required for the compression.

To create an archive file that is compressed using a superior compression algorithm giving a smaller archive file use the `-j` (bzip2) option.

```
tar -cvjf songs.tar.bz2 Ukulele/
```



Once again, the files are listed as the archive is created. The `-j` option is noticeably slower than the `-z` option.

If you are archiving a great many files, you must choose between the `-z` option for decent compression and reasonable speed, or the `-j` option for better compression and slower speed.

As can be seen in the screenshot below, the ".tar" file is the largest, the ".tar.gz" is smaller, and the ".tar.bz2" is the smallest of the archives.

To extract files from an archive file use the `-x` (extract) option. The `-v` (verbose) and `-f` (filename) options behave as they do when creating archives. Use `ls` to confirm which type of archive you are going to extract the files from, then issue the following command.

```
ls

tar -xvf songs.tar
```

```
dave@howtogeek:~/work$ ls
songs.tar  songs.tar.bz2  songs.tar.gz
dave@howtogeek:~/work$ tar -xvf songs.tar
```

The files are listed as they are extracted. Note that the Ukulele directory is also recreated for you.

To extract files from a ".tar.gz" archive, use the `-z` (gzip) option.

```
tar -xvzf songs.tar.gz
```

```
dave@howtogeek:~/work$ tar -xvzf songs.tar.gz
```

Finally, to extract files from a ".tar.bz2" archive use the `-j` option instead of the `-z` (gzip) option.

```
tar -xvjf songs.tar.bz2
```

```
dave@howtogeek:~/work$ tar -xvjf songs.tar.bz2
```

RELATED: ***How to Extract Files From a .tar.gz or .tar.bz2 File on Linux***

34. top

The `top` command shows you a real-time display of the data relating to your Linux machine. The top of the screen is a status summary.

The first line shows you the time and how long your computer has been running for, how many users are logged into it, and what the load average has been over the past one, five, and fifteen minutes.

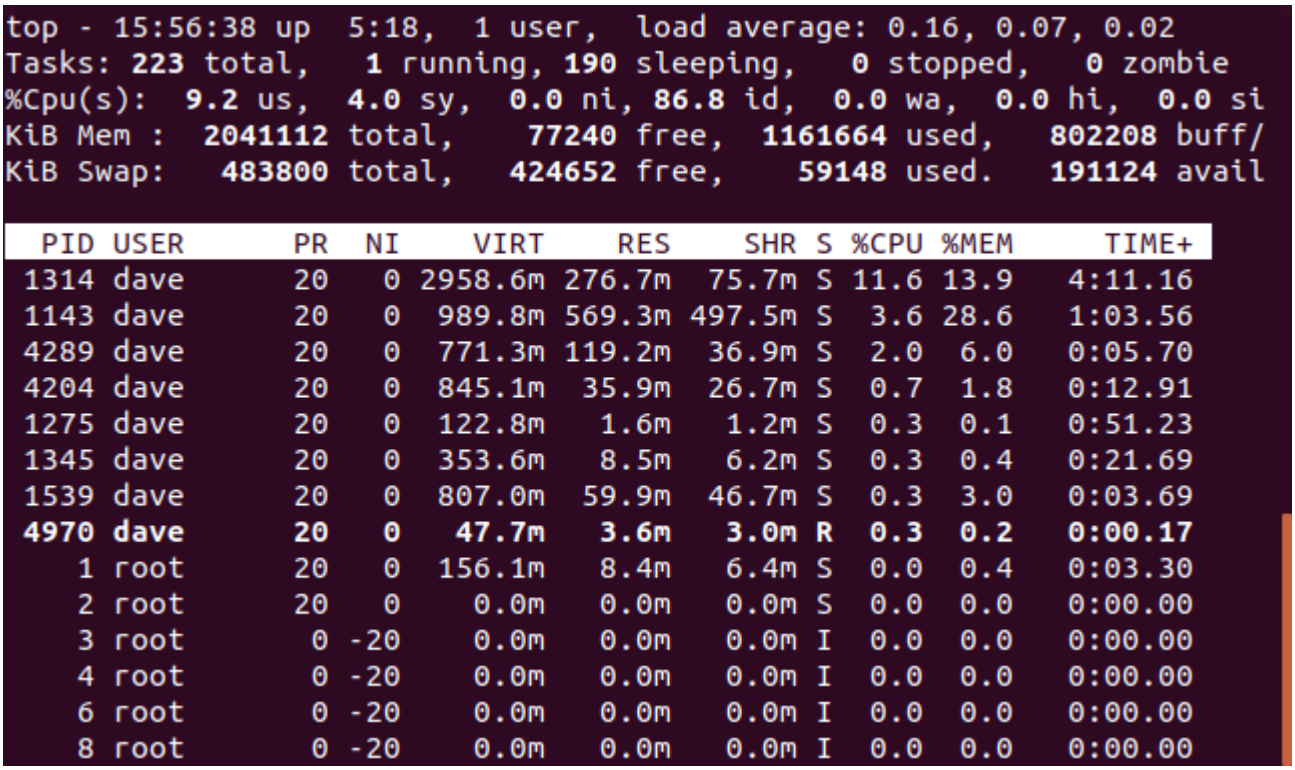
The second line shows the number of tasks and their states: running, stopped, sleeping and zombie.

The third line shows CPU information. Here's what the fields mean:

- us: value is the CPU time the CPU spends executing processes for users, in "user space"
- sy: value is the CPU time spent on running system "kernel space" processes
- ni: value is the CPU time spent on executing processes with a manually set nice value
- id: is the amount of CPU idle time
- wa: value is the time the CPU spends waiting for I/O to complete
- hi: The CPU time spent servicing hardware interrupts
- si: The CPU time spent servicing software interrupts
- st: The CPU time lost due to running virtual machines ("steal time")

The fourth line shows the total amount of physical memory, and how much is free, used and buffered or cached.

The fifth line shows the total amount of swap memory, and how much is free, used and available (taking into account memory that is expected to be recoverable from caches).



The user has pressed the E key to change the display into more humanly digestible figures instead of long integers representing bytes.

The columns in the main display are made up of:

- PID: Process ID
- USER: Name of the owner of the process
- PR: Process priority
- NI: The nice value of the process
- VIRT: Virtual memory used by the process
- RES: Resident memory used by the process
- SHR: Shared memory used by the process
- S: Status of the process. See the list below of the values this field can take
- %CPU: the share of CPU time used by the process since last update
- %MEM: share of physical memory used
- TIME+: total CPU time used by the task in hundredths of a second
- COMMAND: command name or command line (name + options)

(The command column didn't fit into the screenshot.)

The status of the process can be one of:

- D: Uninterruptible sleep
- R: Running
- S: Sleeping
- T: Traced (stopped)
- Z: Zombie

Press the Q key to exit from `top`.

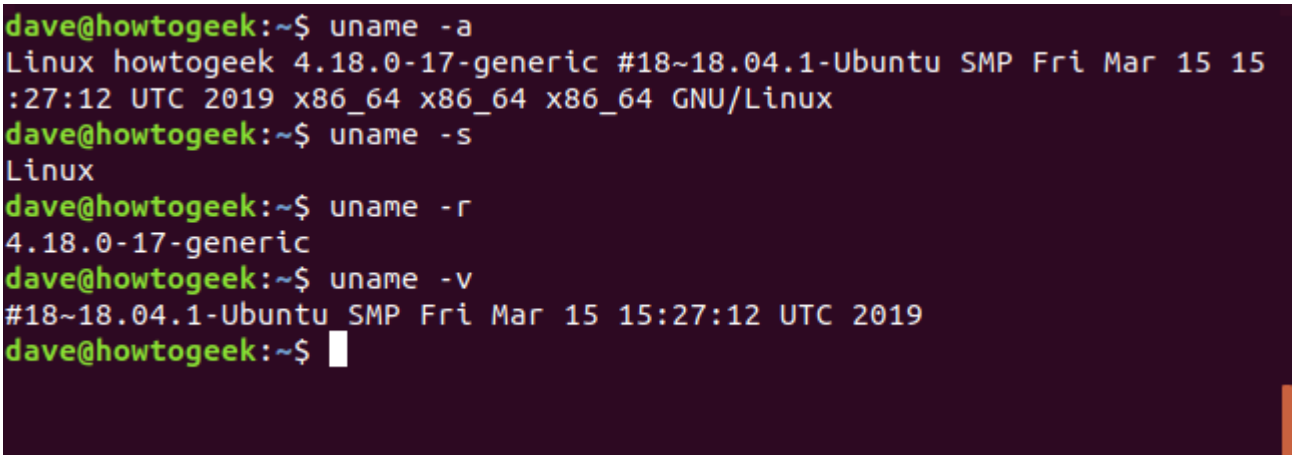
RELATED: ***How to Set Process Priorities With nice and renice on Linux***

35. uname

You can obtain some system information regarding the Linux computer you're working on with the `uname` command.

- Use the `-a` (all) option to see everything.
- Use the `-s` (kernel name) option to see the type of kernel.
- Use the `-r` (kernel release) option to see the kernel release.
- Use the `-v` (kernel version) option to see the kernel version.

```
uname -a

uname -s

uname -r

uname -v
```



36. w

The `w` command lists the currently logged in users.

```
w
```

```
dave@howtogeek:~/Documents$ w
 16:02:09 up  5:24,  4 users,  load average: 0.23, 0.12, 0.03
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
dave     :0       :0               Tue15   ?xdm?   2:34   0.00s /usr/l
i
tom      pts/1    192.168.4.25     15:59    2:28   0.03s  0.03s -bash
mary     pts/2    192.168.4.27     16:00    2:08   0.03s  0.03s -bash
dick     pts/3    192.168.4.27     16:00   27.00s  0.04s  0.04s -bash
dave@howtogeek:~/Documents$
```

37. whoami

Use `whoami` to find out who you are logged in as or who is logged into an unmanned Linux terminal.

```
whoami
```

```
dave@howtogeek:~/Documents$ whoami
dave
dave@howtogeek:~/Documents$
```

Vi Editor:

Starting vi

You may use vi to open an already existing file by typing

```
vi filename
```

where "filename" is the name of the existing file. If the file is not in your current directory, you must use the full pathname.

Or you may create a new file by typing

```
vi newname
```

where "newname" is the name you wish to give the new file.

To open a new file called "testvi," enter

```
vi testvi
```

On-screen, you will see blank lines, each with a tilde (~) at the left, and a line at the bottom giving the name and status of the new file:

```
~
"testvi" [New file]
```

vi Modes

vi has two modes:

- command mode
- insert mode

In command mode, the letters of the keyboard perform editing functions (like moving the cursor, deleting text, etc.). To enter command mode, press the escape <Esc> key.

In insert mode, the letters you type form words and sentences. Unlike many word processors, vi starts up in command mode.

Entering Text

In order to begin entering text in this empty file, you must change from command mode to insert mode. To do this, type

```
i
```

Nothing appears to change, but you are now in insert mode and can begin typing text. In general, vi's commands do not display on the screen and do not require the Return key to be pressed.

Type a few short lines and press <Return> at the end of each line. If you type a long line, you will notice the vi does not word wrap, it merely breaks the line unceremoniously at the edge of the screen.

If you make a mistake, pressing <Backspace> or <Delete> may remove the error, depending on your terminal type.

Moving the Cursor

To move the cursor to another position, you must be in command mode. If you have just finished typing text, you are still in insert mode. Go back to command mode by pressing <Esc>. If you are not sure which mode you are in, press <Esc> once or twice until you hear a beep. When you hear the beep, you are in command mode.

The cursor is controlled with four keys: h, j, k, l.

```
Key         Cursor Movement
---         ---------------
h         left one space
j         down one line
k         up one line
l         right one space
```

When you have gone as far as possible in one direction, the cursor stops moving and you hear a beep. For example, you cannot use l to move right and wrap around to the next line, you must use j to move down a line. See the section entitled "Moving Around in a File" for ways to move more quickly through a file.

Basic Editing

Editing commands require that you be command mode. Many of the editing commands have a different function depending on whether they are typed as upper- or lowercase. Often, editing commands can be preceded by a number to indicate a repetition of the command.

Deleting Characters

To delete a character from a file, move the cursor until it is on the incorrect letter, then type

```
x
```

The character under the cursor disappears. To remove four characters (the one under the cursor and the next three) type

```
4x
```

To delete the character before the cursor, type

```
X (uppercase)
```

Deleting Words

To delete a word, move the cursor to the first letter of the word, and type

```
dw
```

This command deletes the word and the space following it.

To delete three words type

```
3dw
```

Deleting Lines

To delete a whole line, type

```
dd
```

The cursor does not have to be at the beginning of the line. Typing dd deletes the entire line containing the cursor and places the cursor at the start of the next line. To delete two lines, type

```
2dd
```

To delete from the cursor position to the end of the line, type

```
D (uppercase)
```

Replacing Characters

To replace one character with another:

1. Move the cursor to the character to be replaced.
2. Type r
3. Type the replacement character.

The new character will appear, and you will still be in command mode.

Replacing Words

To replace one word with another, move to the start of the incorrect word and type

```
cw
```

The last letter of the word to be replaced will turn into a $. You are now in insert mode and may type the replacement. The new text does not need to be the same length as the original. Press <Esc> to get back to command mode.
To replace three words, type

```
3cw
```

Replacing Lines

To change text from the cursor position to the end of the line:

1. Type C (uppercase).
2. Type the replacement text.
3. Press <Esc>.

Inserting Text

To insert text in a line:

1. Position the cursor where the new text should go.
2. Type i
3. Enter the new text.

The text is inserted BEFORE the cursor.

4. Press <Esc> to get back to command mode.

Appending Text

To add text to the end of a line:

1. Position the cursor on the last letter of the line.
2. Type a
3. Enter the new text.

This adds text AFTER the cursor.

4. Press <Esc> to get back to command mode.

Opening a Blank Line

To insert a blank line below the current line, type

```
o (lowercase)
```

To insert a blank line above the current line, type

```
O (uppercase)
```

Joining Lines

To join two lines together:

1. Put the cursor on the first line to be joined.
2. Type J

To join three lines together:

1. Put the cursor on the first line to be joined.
2. Type 3J

Undoing

To undo your most recent edit, type

```
u
```

To undo all the edits on a single line, type

```
U (uppercase)
```

Undoing all edits on a single line only works as long as the cursor stays on that line. Once you move the cursor off a line, you cannot use U to restore the line.

Moving Around in a File

There are shortcuts to move more quickly though a file. All these work in command mode.

```
Key             Movement
```

```
---             --------
w               forward word by word
b               backward word by word
$               to end of line
0 (zero)        to beginning of line
H               to top line of screen
M               to middle line of screen
L               to last line of screen
G               to last line of file
1G              to first line of file
<Control>f      scroll forward one screen
<Control>b      scroll backward one screen
<Control>d      scroll down one-half screen
<Control>u      scroll up one-half screen
```

Moving by Searching

To move quickly by searching for text, while in command mode:

1. Type / (slash).
2. Enter the text to search for.
3. Press <Return>.

The cursor moves to the first occurrence of that text.

To repeat the search in a forward direction, type

```
n
```

To repeat the search in a backward direction, type

```
N
```

Closing and Saving a File

With vi, you edit a copy of the file, rather than the original file. Changes are made to the original only when you save your edits.

To save the file and quit vi, type

```
ZZ
```

The vi editor editor is built on an earler Unix text editor called ex. ex commands can be used within vi. ex commands begin with a : (colon) and end with a <Return>. The command is displayed on the status line as you type. Some ex commands are useful when saving and closing files.

To save the edits you have made, but leave vi running and your file open:

1. Press <Esc>.
2. Type :w
3. Press <Return>.

To quit vi, and discard any changes your have made since last saving:

1. Press <Esc>.
2. Type :q!
3. Press <Return>.

VI EDITOR:

```
    vi filename    edit a file named "filename"
    vi newfile     create a new file named "newfile"
```

ENTERING TEXT

```
    i            insert text left of cursor
    a            append text right of cursor
```

MOVING THE CURSOR

```
    h            left one space
    j            down one line
    k            up one line
    l            right one space
```

BASIC EDITING

```
    x         delete character
    nx        delete n characters
    X         delete character before cursor
    dw        delete word
    ndw       delete n words
    dd        delete line
    ndd       delete n lines
    D         delete characters from cursor to end of line
    r         replace character under cursor
    cw        replace a word
    ncw       replace n words
    C         change text from cursor to end of line
    o         insert blank line below cursor
                 (ready for insertion)
    O         insert blank line above cursor
                 (ready for insertion)
    J         join succeeding line to current cursor line
    nJ        join n succeeding lines to current cursor line
    u         undo last change
    U         restore current line
```

MOVING AROUND IN A FILE

```
    w            forward word by word
    b            backward word by word
    $            to end of line
    0 (zero)     to beginning of line
    H            to top line of screen
    M            to middle line of screen
```

```
L             to last line of screen
G             to last line of file
1G            to first line of file
<Control>f    scroll forward one screen
<Control>b    scroll backward one screen
<Control>d    scroll down one-half screen
<Control>u    scroll up one-half screen
n             repeat last search in same direction
N             repeat last search in opposite direction
```

CLOSING AND SAVING A FILE

```
ZZ             save file and then quit
:w             save file
:q!             discard changes and quit file
```