

**Top 50+ Linux Commands**

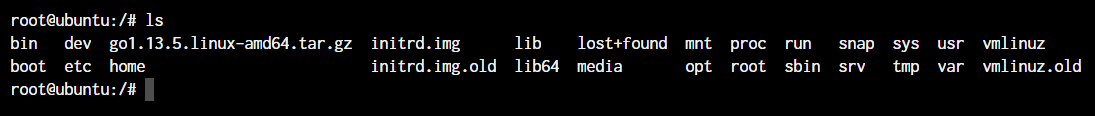


1. **ls** - The most frequently used command in Linux to list directories
2. **pwd** - Print working directory command in Linux
3. **cd** - Linux command to navigate through directories
4. **mkdir** - Command used to create directories in Linux
5. **mv** - Move or rename files in Linux
6. **cp** - Similar usage as mv but for copying files in Linux
7. **rm** - Delete files or directories
8. **touch** - Create blank/empty files
9. **ln** - Create symbolic links (shortcuts) to other files
10. **cat** - Display file contents on the terminal
11. **clear** - Clear the terminal display
12. **echo** - Print any text that follows the command
13. **less** - Linux command to display paged outputs in the terminal
14. **man** - Access manual pages for all Linux commands
15. **uname** - Linux command to get basic information about the OS
16. **whoami** - Get the active username
17. **tar** - Command to extract and compress files in Linux
18. **grep** - Search for a string within an output
19. **head** - Return the specified number of lines from the top
20. **tail** - Return the specified number of lines from the bottom
21. **diff** - Find the difference between two files
22. **cmp** - Allows you to check if two files are identical
23. **comm** - Combines the functionality of diff and cmp
24. **sort** - Linux command to sort the content of a file while outputting
25. **export** - Export environment variables in Linux
26. **zip** - Zip files in Linux
27. **unzip** - Unzip files in Linux
28. **ssh** - Secure Shell command in Linux
29. **service** - Linux command to start and stop services
30. **ps** - Display active processes
31. **kill and killall** - Kill active processes by process ID or name
32. **df** - Display disk filesystem information
33. **mount** - Mount file systems in Linux
34. **chmod** - Command to change file permissions
35. **chown** - Command for granting ownership of files or folders
36. **ifconfig** - Display network interfaces and IP addresses
37. **traceroute** - Trace all the network hops to reach the destination
38. **wget** - Direct download files from the internet
39. **ufw** - Firewall command
40. **iptables** - Base firewall for all other firewall utilities to interface with
41. **apt, pacman, yum, rpm** - Package managers depending on the distro
42. **sudo** - Command to escalate privileges in Linux
43. **cal** - View a command-line calendar
44. **alias -** Create custom shortcuts for your regularly used commands
45. **dd** - Majorly used for creating bootable USB sticks
46. **whereis** - Locate the binary, source, and manual pages for a command
47. **whatis** - Find what a command is used for
48. **top** - View active processes live with their system usage
49. **useradd and usermod** - Add new user or change existing users data
50. **passwd** - Create or update passwords for existing users

Now let’s dive a little deeper into each of these commands and understand them in more detail. We already have a lot of existing articles for each of those individual commands. For your convenience, we’ll add links to all the existing articles, and continue to update the article as new topics are covered.

**The ls command in Linux**

The ls command is used to list files and directories in the current working directory. This is going to be one of the most frequently used Linux commands you must know of.



As you can see in the above image, using the command by itself without any arguments will give us an output with all the files and directories in the directory. The command offers a lot of flexibility in terms of displaying the data in the output.

Learn more about the [ls command (link to full article)](https://www.digitalocean.com/community/tutorials/ls-command-in-linux-unix)

**The pwd command in Linux**

The pwd command allows you to print the current working directory on your terminal. It’s a very basic command and solves its purpose very well.

Pwd Default Output

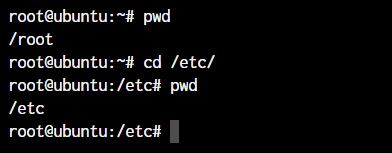
Now, your terminal prompt should usually have the complete directory anyway. But in case it doesn’t, this can be a quick command to see the directory that you’re in. Another application of this command is when creating scripts where this command can allow us to find the directory where the script has been saved.

**The cd command in Linux**

While working within the terminal, moving around within directories is pretty much a necessity. The cd command is one of the important Linux commands you must know and it will help you to navigate through directories. Just type **cd** followed by directory as shown below.

root@ubuntu:~# cd <directory path>

Copy



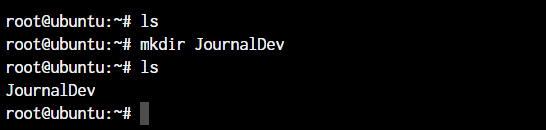
As you can see in the above command, I simply typed **cd /etc/** to get into the /etc directory. We used the **pwd command** to print the current working directory.

**The mkdir command in Linux**

The mkdir command allows you to create directories from within the terminal. The default syntax is **mkdir** followed by the directory name.

root@ubuntu:~# mkdir <folder name>

Copy



As you can see in the above screenshot, we created the JournalDev directory with just this simple command.

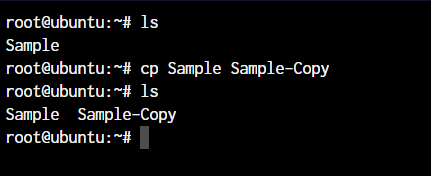
Learn more about the [mkdir command (Link to article)](https://www.digitalocean.com/community/tutorials/mkdir-command-linux-unix)

**The cp and mv commands**

The cp and mv commands are equivalent to the copy-paste and cut-paste in Windows. But since Linux doesn’t really have a command for renaming files, we also make use of the mv command to rename files and folders.

root@ubuntu:~# cp <source> <destination>

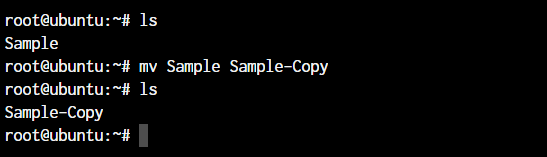
Copy



In the above command, we created a copy of the file named Sample. Let’s see how what happens if we use the mv command in the same manner. For this demonstration, I’ll delete the Sample-Copy file.

root@ubuntu:~# mv <source> <destination

Copy



In the above case, since we were moving the file within the same directory, it acted as rename. The file name is now changed.

Learn more about the [cp command (Link to article)](https://www.digitalocean.com/community/tutorials/cp-command-in-linux-unix) and [mv command (Link to article)](https://www.digitalocean.com/community/tutorials/linux-mv-command-usage-with-examples).

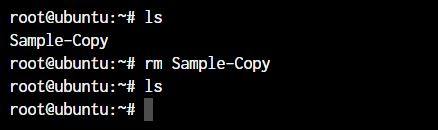
**The rm command in Linux**

In the previous section, we deleted the Sample-Copy file. The rm command is used to delete files and folders and is one of the important Linux commands you must know.

root@ubuntu:~# rm <file name>

root@ubuntu:~# rm -r <folder/directory name>

Copy



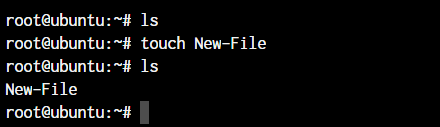
To delete a directory, you have to add the **-r** argument to it. Without the -r argument, rm command won’t delete directories.

**The touch command in Linux**

To create a new file, the touch command will be used. The **touch** keyword followed by the file name will create a file in the current directory.

root@ubuntu:~# touch <file name>

Copy

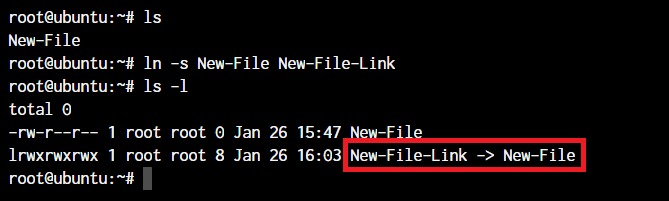


**The ln command in Linux**

To create a link to another file, we use the ln command. This is one of the important Linux commands that you should know if you’re planning to work as a Linux administrator.

root@ubuntu:~# ln -s <source path> <link name>

Copy



The basic syntax involves using the **-s** parameter so we can create a symbolic link or soft link.

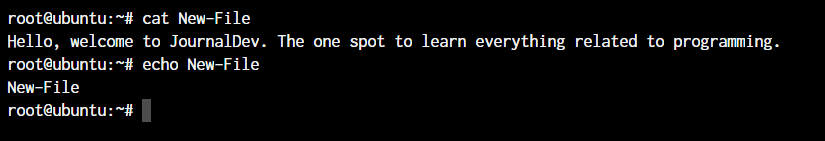
**The cat, echo, and less commands**

When you want to output the contents of a file, or print anything to the terminal output, we make use of the cat or echo commands. Let’s see their basic usage. I’ve added some text to our New-File that we created earlier.

root@ubuntu:~# cat <file name>

root@ubuntu:~# echo <Text to print on terminal>

Copy



As you can see in the above example, the **cat command** when used on our “New-File”, prints the contents of the file. At the same time, when we use **echo command**, it simply prints whatever follows after the command.

The **less command** is used when the output printed by any command is larger than the screen space and needs scrolling. The less command allows use to break down the output and scroll through it with the use of the enter or space keys.

The simple way to do this is with the use of the pipe operator (**|**).

root@ubuntu:~# cat /boot/grub/grub.cfg | less

Copy

Learn more about the [echo command(Link to article)](https://www.digitalocean.com/community/tutorials/linux-echo-command-examples) and [cat command(Link to article)](https://www.digitalocean.com/community/tutorials/linux-cat-command-examples).

**The man command in Linux**

The man command is a very useful Linux command you must know. When working with Linux, the packages that we download can have a lot of functionality. Knowing it all is impossible.

The man pages offer a really efficient way to know the functionality of pretty much all the packages that you can download using the package managers in your Linux distro.

root@ubuntu:~# man <command>

Copy

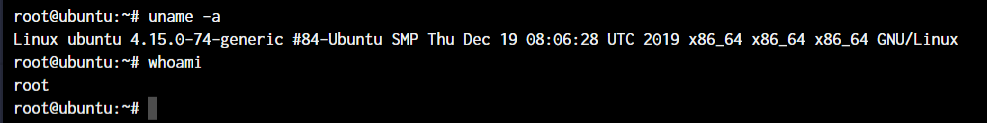
**The uname and whoami commands**

The uname and whoami commands allow you to know some basic information which comes really handy when you work on multiple systems. In general, if you’re working with a single computer, you won’t really need it as often as someone who is a network administrator.

Let’s see the output of both the commands and the way we can use these.

root@ubuntu:~# uname -a

Copy



The parameter **-a** which I’ve supplied to uname, stands for “all”. This prints out the complete information. If the parameter is not added, all you will get as the output is “Linux”.

**The tar, zip, and unzip commands**

The tar command in Linux is used to create and extract archived files in Linux. We can extract multiple different archive files using the tar command.

To create an archive, we use the -c parameter and to extract an archive, we use the -x parameter. Let’s see it working.

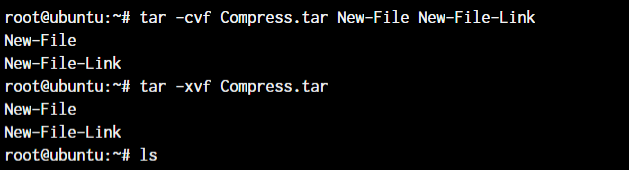
#Compress

root@ubuntu:~# tar -cvf <archive name> <files seperated by space>

#Extract

root@ubuntu:~# tar -xvf <archive name>

Copy



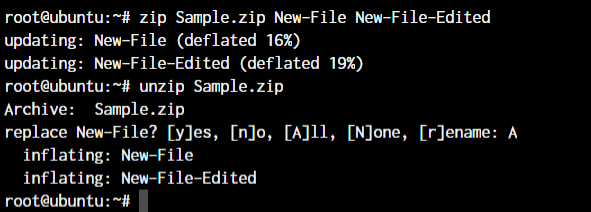
In the first line, we created an archive named **Compress.tar** with the New-File and New-File-Link. In the next command, we have extracted those files from the archive.

Now coming to the zip and unzip commands. Both these commands are very straight forward. You can use them without any parameters and they’ll work as intended. Let’s see an example below.

root@ubuntu:~# zip <archive name> <file names separated by space>

root@ubuntu:~# unzip <archive name>

Copy



Since we already have those files in the same directory, the unzip command prompts us before overwriting those files.

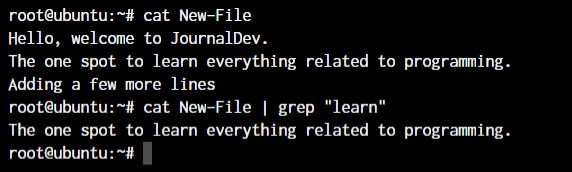
Learn more about the [tar command (Link to article)](https://www.digitalocean.com/community/tutorials/linux-tar-command) and [zip and unzip commands (Link to article)](https://www.digitalocean.com/community/tutorials/zip-unzip-linux)

**The grep command in Linux**

If you wish to search for a specific string within an output, the grep command comes into the picture. We can pipe (**|**) the output to the grep command and extract the required string.

root@ubuntu:~# <Any command with output> | grep "<string to find>"

Copy



This was a simple demonstration of the command. Learn more about the [grep command (Link to article)](https://www.digitalocean.com/community/tutorials/grep-command-in-linux-unix)

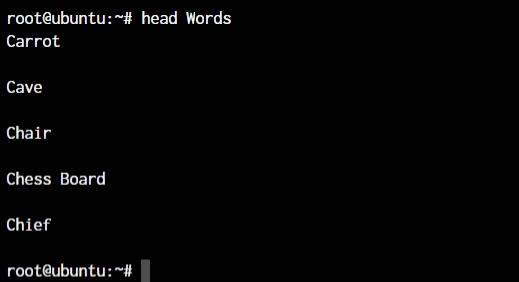
**The head and tail commands**

When outputting large files, the head and the tail commands come in handy. I’ve created a file named “Words” with a lot of words arranged alphabetically in it. The head command will output the first 10 lines from the file, while the tail command will output the last 10. This also includes any blank lines and not just lines with text.

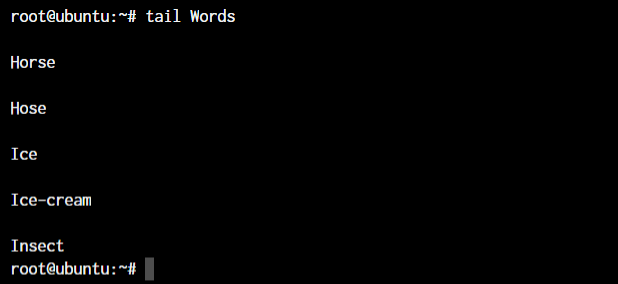
root@ubuntu:~# head <file name>

root@ubuntu:~# tail <file name>

Copy



As you can see, the head command showed 10 lines from the top of the file.



The tail command outputted the bottom 10 lines from the file.

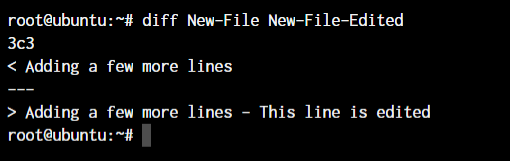
Learn more about the [tail command(Link to article)](https://www.digitalocean.com/community/tutorials/linux-tail-command-examples)

**The diff, comm, and cmp commands**

Linux offers multiple commands to compare files. The diff, comm, and cmp commands compare differences and are some of the most useful Linux commands you must know. Let’s see the default outputs for all the three commands.

root@ubuntu:~# diff <file 1> <file 2>

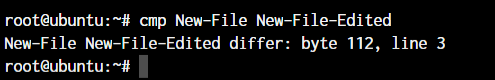
Copy



As you can see above, I’ve added a small piece of text saying “This line is edited” to the New-File-Edited file.

root@ubuntu:~# cmp <file 1> <file 2>

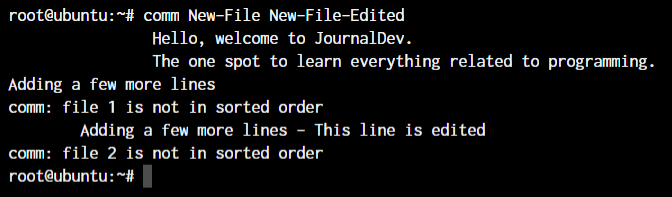
Copy



The cmp command only tells use the line number which is different. Not the actual text. Let’s see what the comm command does.

root@ubuntu:~# comm <file 1> <file2>

Copy



The text that’s aligned to the left is the text that’s only present in file 1. The center-aligned text is present only in file 2. And the right-aligned text is present in both the files.

By the looks of it, comm command makes the most sense when we’re trying to compare larger files and would like to see everything arranged together.

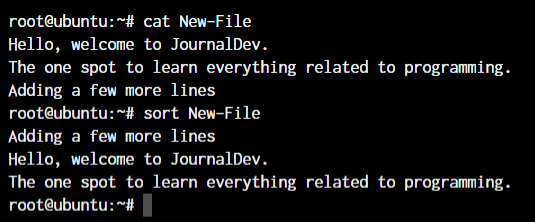
**The sort command in Linux**

The sort command will provide a sorted output of the contents of a file. Let’s use the sort command without any parameters and see the output.

The basic syntax of the sort command is:

root@ubuntu:~# sort <filename>

Copy



**The export command in Linux**

The export command is specially used when exporting environment variables in runtime. For example, if I wanted to update the bash prompt, I’ll update the PS1 environment variable. The bash prompt will be updated with immediate effect.

root@ubuntu:~# export <variable name>=<value>

Copy

Export Environment Variables

If for some reason, your bash prompt doesn’t update, just type in **bash** and you should see the updated terminal prompt.

Learn more about the [export command(Link to article)](https://www.digitalocean.com/community/tutorials/export-command-linux)

**The ssh command in Linux**

The ssh command allows us to connect to an external machine on the network with the use of the ssh protocol. The basic syntax of the ssh command is:

root@ubuntu:~ -->> ssh username@hostname

Copy

Learn more about [ssh command(Link to article)](https://www.digitalocean.com/community/tutorials/ssh-command-linux-unix)

**The service command in Linux**

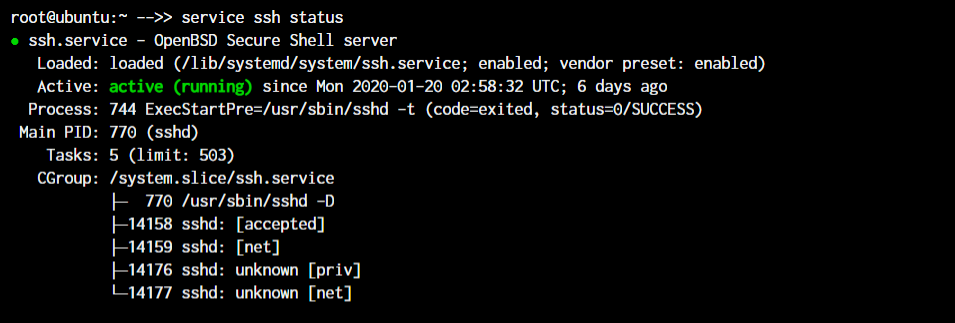
The service command in Linux is used for starting and stopping different services within the operating system. The basic syntax of the command is as below.

root@ubuntu:~ -->> service ssh status

root@ubuntu:~ -->> service ssh stop

root@ubuntu:~ -->> service ssh start

Copy



As you can see in the image, the ssh server is running on our system.

**The ps, kill, and killall commands**

While we’re on the topic of processes, let’s see how we can find active processes and kill them. To find the running processes, we can simply type **ps** in the terminal prompt and get the list of running processes.

root@ubuntu:~ -->> ps

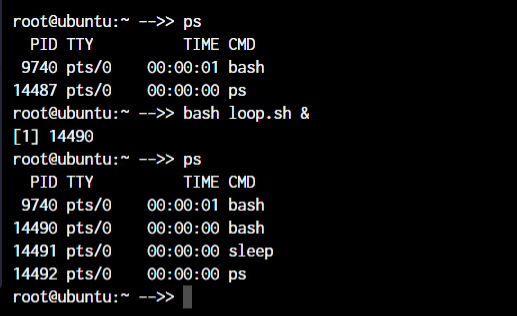
root@ubuntu:~ -->> kill <process ID>

root@ubuntu:~ -->> killall <process name>

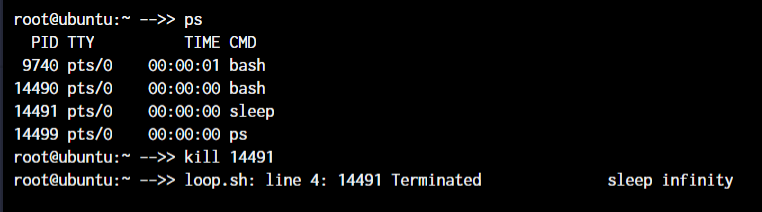
Copy

For demonstration purposes, I’m creating a shell script with an infinite loop and will run it in the background.

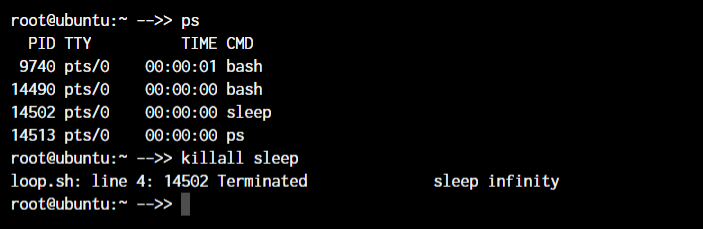
With the use of the **&** symbol, I can pass a process into the background. As you can see, a new bash process with PID 14490 is created.



Now, to kill a process with the **kill** command, you can type **kill** followed b the PID of the process.



But if you do not know the process ID and just want to kill the process with the name, you can make use of the killall command.



You will notice that PID 14490 stayed active. That is because both the times, I killed the sleep process.

Learn more about [ps command (Link to article)](https://www.digitalocean.com/community/tutorials/linux-ps-command).

**The df and mount commands**

When working with Linux, the df and mount commands are very efficient utilities to mount filesystems and get details of the file system.

When I say mount, it means that we’ll connect the device to a folder so we can access the files from our filesystem. The default syntax to mount a filesystem is below:

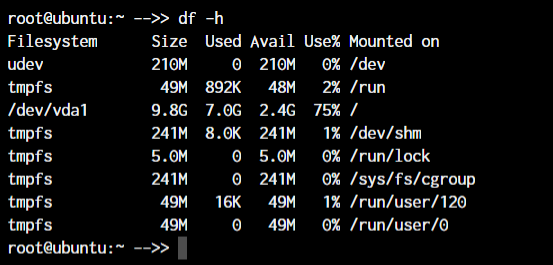
root@ubuntu:~ -->> mount /dev/cdrom /mnt

root@ubuntu:~ -->> df -h

Copy

In the above case, **/dev/cdrom** is the device that needs to be mounted. Usually, a mountable device is found inside the /dev folder. **/mnt** is the destination folder to mount the device to. You can change it to any folder you want but I’ve used /mnt as it’s pretty much a system default folder for mounting devices.

To see the mounted devices and get more information about them, we make use of the df command. Just typing **df** will give us the data in bytes which is not readable. So we’ll use the **-h** parameter to make the data human-readable.



Learn more about the [df command(Link to article)](https://www.digitalocean.com/community/tutorials/linux-df-command-check-disk-space)

**The chmod and chown commands**

The chmod and chown commands give us the functionality to change the file permissions and file ownership are the most important Linux commands you should know.

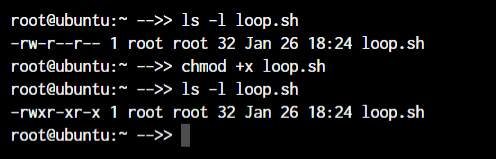
The main difference between the functions of the two commands is that the chmod command allows changing file permissions, while chown allows us to change the file owners.

The default syntax for both the commands is **chmod <parameter> filename** and **chown <user:group> filename**

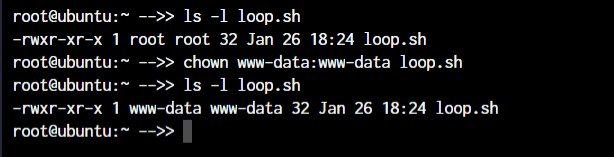
root@ubuntu:~ -->> chmod +x loop.sh

root@ubuntu:~ -->> chmod root:root loop.sh

Copy



In the above example, we’re adding executable permissions to the [loop.sh](http://loop.sh/) file with the **chmod command**. Apart from that, with the **chown command**, we’ve made it accessible only by root user and users within the root group.



As you will notice, the **root root** part is now changed to **www-data** which is the new user who has full file ownership.

Learn more about the [chmod command(Link to article)](https://www.digitalocean.com/community/tutorials/linux-chmod-command-examples) and [chown command (Link to article)](https://www.digitalocean.com/community/tutorials/linux-chown-command-examples)

**The ifconfig and traceroute commands**

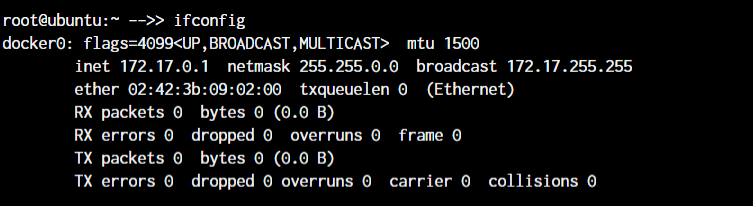
Moving on to the networking section in Linux, we come across the ifconfig and traceroute commands which will be frequently used if you manage a network.

The ifconfig command will give you the list of all the network interfaces along with the IP addresses, MAC addresses and other information about the interface.

root@ubuntu:~ -->> ifconfig

Copy

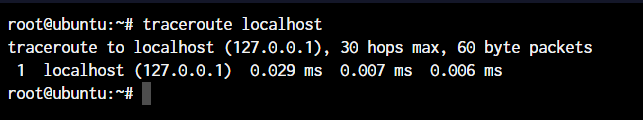
There are multiple parameters that can be used but we’ll work with the basic command here.



When working with traceroute, you can simply specify the IP address, the hostname or the domain name of the endpoint.

root@ubuntu:~ -->> traceroute <destination address>

Copy



Now obviously, localhost is just one hop (which is the network interface itself). You can try this same command with any other domain name or IP address to see all the routers that your data packets pass through to reach the destination.

Learn more about the [ifconfig command(Link to article)](https://www.digitalocean.com/community/tutorials/linux-ifconfig-command-examples)

**The wget command in Linux**

If you want to download a file from within the terminal, the wget command is one of the handiest command-line utilities available. This will be one of the important Linux commands you should know when working with source files.

When you specify the link for download, it has to directly be a link to the file. If the file cannot be accessed by the wget command, it will simply download the webpage in HTML format instead of the actual file that you wanted.

Let’s try an example. The basic syntax of the wget command is :

root@ubuntu:~ -->> wget <link to file>

OR

root@ubuntu:~ -->> wget -c <link to file>

Copy

The **-c** argument allows us to resume an interrupted download.

**The ufw and iptables commands**

UFW and IPTables are firewall interfaces for the Linux Kernel’s netfilter firewall. IPTables directly passes firewall rules to netfilter while UFW configures the rules in IPTables which then sends those rules to netfilter.

Why do we need UFW when we have IPTables? Because IPTables is pretty difficult for a newbie. UFW makes things extremely easy. See the below example where we are trying to allow the port 80 for our webserver.

root@ubuntu:~# iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT

root@ubuntu:~# ufw allow 80

Copy

I’m sure you now know why UFW was created! Look at how easy the syntax becomes. Both these firewalls are very comprehensive and can allow you to create any kind of configuration required for your network. Learn at least the basics of UFW or IPTables firewall as these are the Linux commands you must know.

Learn more [opening ports on Linux(Link to article)](https://www.digitalocean.com/community/tutorials/opening-a-port-on-linux)

**Package Managers in Linux**

Different distros of Linux make use of different package managers. Since we’re working on a Ubuntu server, we have the **apt package manager**. But for someone working on a Fedora, Red Hat, Arch, or Centos machine, the package manager will be different.

* **Debian and Debian-based distros** - apt install <package name>
* **Arch and Arch-based distros** - pacman -S <package name>
* **Red Hat and Red Hat-based distros** - yum install <package name>
* **Fedora and CentOS** - yum install <package>

Getting yourself well versed with the package manager of your distribution will make things much easier for you in the long run. So even if you have a GUI based package management tool installed, try an make use of the CLI based tool before you move on to the GUI utility. Add these to your list of Linux commands you must know.

**The sudo command in Linux**

*“With great power, comes great responsibility”*

This is the quote that’s displayed when a sudo enabled user(sudoer) first makes use of the sudo command to escalate privileges. This command is equivalent to having logged in as root (based on what permissions you have as a sudoer).

non-root-user@ubuntu:~# sudo <command you want to run>

Password:

Copy

Just add the word **sudo** before any command that you need to run with escalated privileges and that’s it. It’s very simple to use, but can also be an added security risk if a malicious user gains access to a sudoer.

Learn more about the [sudo command (Link to article)](https://www.digitalocean.com/community/tutorials/sudo-command-in-linux)

**The cal command in Linux**

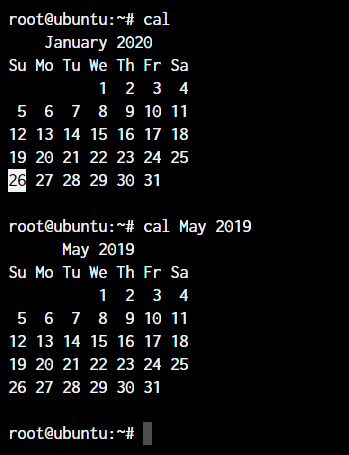
Ever wanted to view the calendar in the terminal? Me neither! But there apparently are people who wanted it to happen and well here it is.

The **cal** command displays a well-presented calendar on the terminal. Just enter the word cal on your terminal prompt.

root@ubuntu:~# cal

root@ubuntu:~# cal May 2019

Copy



Even though I don’t need it, it’s a really cool addition! I’m sure there are people who are terminal fans and this is a really amazing option for them.

**The alias command**

Do you have some commands that you run very frequently while using the terminal? It could be **rm -r** or **ls -l**, or it could be something longer like **tar -xvzf**. This is one of the productivity-boosting Linux commands you must know.

If you know a command that you run very often, it’s time to create an alias. What’s an alias? In simple terms, it’s another name for a command that you’ve defined.

root@ubuntu:~# alias lsl="ls -l"

OR

root@ubuntu:~# alias rmd="rm -r"

Copy

Now every time you enter **lsl** or **rmd** in the terminal, you’ll receive the output that you’d have received if you had used the full commands.

The examples here are for really small commands that you can still type by hand every time. But in some situations where a command has too many arguments that you need to type, it’s best to create a shorthand version of the same.

Learn more about [alias command (LInk to article)](https://www.digitalocean.com/community/tutorials/linux-alias-command)

**The dd command in Linux**

This command was created to convert and copy files from multiple file system formats. In the current day, the command is simply used to create bootable USB for Linux but there still are some things important you can do with the command.

For example, if I wanted to back up the entire hard drive as is to another drive, I’ll make use of the dd command.

root@ubuntu:~# dd if = /dev/sdb of = /dev/sda

Copy

The **if** and **of** arguments stand for **input file** and **output file**.

**The whereis and whatis commands**

The names of the commands make it very clear as to their functionality. But let’s demonstrate their functionality to make things more clear.

The **whereis** command will output the exact location of any command that you type in after the **whereis** command.

root@ubuntu:~# whereis sudo

sudo: /usr/bin/sudo /usr/lib/sudo /usr/share/man/man8/sudo.8.gz

Copy

The **whatis** command gives us an explanation of what a command actually is. Similar to the whereis command, you’ll receive the information for any command that you type after the **whatis** command.

root@ubuntu:~# whatis sudo

sudo (8) - execute a command as another user

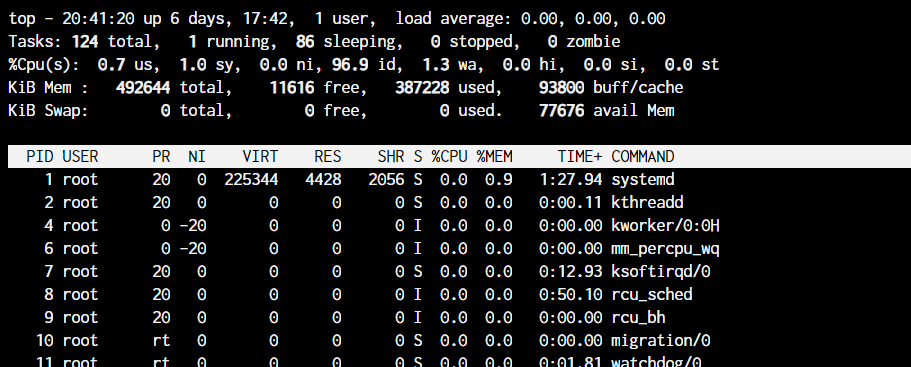
Copy

**The top command in Linux**

A few sections earlier, we talked about the ps command. You observed that the ps command will output the active processes and end itself.

The top command is like a CLI version of the task manager in Windows. You get a live view of the processes and all the information accompanying those processes like memory usage, CPU usage, etc.

To get the top command, all you need to do is type the word **top** in your terminal.



**The useradd and usermod commands**

The **useradd** or **adduser** commands are the exact same commands where adduser is just a symbolic link to the useradd command. This command allows us to create a new user in Linux.

root@ubuntu:~# useradd JournalDev -d /home/JD

Copy

The above command will create a new user named JournalDev with the home directory as **/home/JD**.

The **usermod** command, on the other hand, is used to modify existing users. You can modify any value of the user including the groups, the permissions, etc.

For example, if you want to add more groups to the user, you can type in:

root@ubuntu:~# usermod JournalDev -a -G sudo, audio, mysql

Copy

Learn more on [how to create and manage users on Linux (Link to article)](https://www.digitalocean.com/community/tutorials/create-manage-new-users-linux)

**The passwd command in Linux**

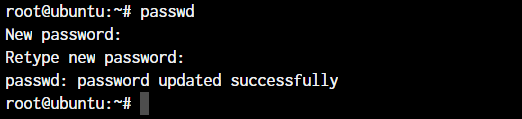
Now that you know how to create new users, let’s also set the password for them. The **passwd** command lets you set the password for your own account, or if you have the permissions, set the password for other accounts.

The command usage is pretty simple:

root@ubuntu:~# passwd

New password:

Copy



If you add the username after **passwd**, you can set passwords for other users. Enter the new password twice and you’re done. That’s it! You will have a new password set for the user!