Top 100 Linux Commands

With this article ‘**Top 100 Linux Commands**’, get to explore the world of **Linux** Commands, that every **Linux** user should know. From file management and networking to system administration, these commands cover a wide range of functionalities. Whether you are a beginner or an experienced user, this curated collection will give you the necessary skills to navigate the entire area of **Linux Command-Line tools**.

Table of Contents

List of Top 100 LINUX Commands

Following, I will give a list of 100 top Linux commands with their description, syntax & useful options. You can also learn more about these commands with necessary examples & practical usages by visiting the corresponding attached link that is appended after each command.

1. ls

The **ls** command **'Lists'** the contents, both files and subdirectories of the current directory by default. It is one of the most used commands, as one can view the contents of a directory without exiting the terminal and perform their desired tasks on the specific contents.

Syntax

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

Useful Options

- **-a** → Doesn't ignore the hidden files (files named with **.(dot)** at the beginning).
- **-h** → Prints sizes in human-readable forms.
- **-l** → Lists in a long form.
- **-S** → Sorts according to file size, largest first.
- **-d** → Only lists the current directory, not its contents.

To learn more read, [The "ls" Command in Linux [7+ Practical Examples]]

2. cd

The word **cd** stands for **c**hange **d**irectory. This command is used for changing the current directory of the user. It will take the user from the current directory (current location) to a specified directory.

Syntax

cd [OPTION]... [DIRECTORY]

Useful Options

- **cd ~[username]** → Changes the directory to the home directory of the specified user.
- **cd ..** → Changes directory one directory up the current directory.
- **cd -** → Changes the directory to the previously changed directory.

To learn more read, [The "cd" Command in Linux [6 Practical Examples]]

3. pwd

The **pwd** command stands for **p**rint **w**orking **d**irectory. It displays the **Absolute path** of the **current** directory, in a simple context, prints the name of the **current/working** directory all the way beginning from the root(/) directory. So it shows where the **Terminal** currently is in detail and will help you when you are lost inside some unknown directory.

Syntax

pwd [OPTION]...

Useful Options

- **-L, --logical** → Even as it carries symlinks, PWD utilizes from the environment.
- **-P, --physical** → Avoids the symlinks.

When no option is mentioned, it is assumed that **option -P** is being used.
To learn more read, [The "pwd" Command in Linux [4 Practical Examples]]

4. mkdir

The command **mkdir** is the abbreviation for **make dir**ectory. As the name suggests, the **mkdir** command is used to create one or more directories.

Syntax

mkdir [OPTION]... DIRECTORY...

Useful Options

- **-p - -parents** → Creates the necessary parent directories if required.
- **-v, --verbose** → Prints message of what is being performed.

To learn more read, [The "mkdir" Command in Linux [6+ Practical Examples]]

5. rm

The **rm** command is the abbreviation for **rem**ove. As the name suggests, it removes files and the removal is **permanent**, so be cautious while using it. The command can also be used to remove directories and their contents permanently.

Syntax

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

Useful Options

- **-i** → Displays interactive prompt before completing the deletion each time.
- **-I** → Only shows prompt while deleting 3 or more files or deleting recursively.
- **-d, --dir** → Removes the empty directories.
- **-R, -r, --recursive** → Removes any directory as well as its contents recursively.
- **-v, --verbose** → Prints message of what is being performed.

To learn more read, [The "rm" Command in Linux [7 Practical Examples]]()

6. rmdir

The **rmdir** command in **Linux** is used to **remove** only **"empty directories."** In addition, if you try to **remove** a **non-empty** directory using the **rmdir** command, it will prompt the **"Directory not empty"** error message by **preventing** accidentally removing the **non-empty** directories. And this decreases the **risk** of losing necessary data.

Syntax

rmdir [OPTION]... DTRECTORY_NAME...

Useful Options

- **--ignore-fail-on-non-empty** → Ignore each failure that is only because a directory is non-empty.
- **-p, --parents** → Remove the directory along with its empty content Directories.
- **-v, --verbose** → Generate a diagnostic report for each file that is processed.

To learn more read, [The "rmdir" Command in Linux [7 Practical Examples]]()

7. touch

The **touch** command allows us to update a file's access or modification time. However, if the file doesn't exist we can **create** that file. This ability to create files makes the **touch** command one of the most useful commands.

Syntax

touch [OPTION]... FILE...

Useful Options

- **-a** → changes only the access time.
- **-m** → Changes only the modification time.

To learn more read, [The "touch" Command in Linux [8 Practical Examples]]()

8. cp

The **cp** command resembles the word '**c**o**p**y'. As the name suggests, it copies things from one place to another place. The command can copy one or multiple files to the specified destination directory. If the directory doesn't exist it just renames the files. It can also be used to copy directories and their contents.

Syntax

The syntax for Copying Files
cp [OPTION]... [-T] SOURCE DESTINATION
The syntax for Copying Files to a Directory
cp [OPTION]... SOURCE… DIRECTORY
The syntax for Copying Directory
cp [OPTION]... SOURCE DIRECTORY DESTINATION DIRECTORY

Useful Options

- **-i, --interactive** → Displays interactive prompt before completing the modification.
- **-R, -r, --recursive** → Copies the directory as well as its contents recursively.
- **-v, --verbose** → Prints message of what is being performed.

To learn more read, The "cp" Command in Linux [6 Practical Examples]

9. mv

The **mv** command is a widely used **file/folder** management command that allows changing the location of a file or folder. Moreover, when changing the location you can rename the file as well just by changing the file name.

Syntax

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

Useful Options

- **-n, --no-cobber** → Does not overwrite an existing file.
- **-i, --interactive** → Prompts before overwriting.
- **-f, --force** → Does not prompt before overwriting.
- **-v, --verbose** → Explains what is being done.
- **-u, --update** → Moves only when the source file is newer than the Destination file.

To learn more read, The "mv" Command in Linux [8 Practical Examples]

10. dd

The **dd** command converts and copies a file to another directory. This command can be used to create a backup inside the hard drive or an external hard drive.

Syntax

dd [OPERAND]...
dd OPTION

Useful Operands

- **bs=BYTES** → Reads/Writes up to a given number of **BYTES** at a time i.e. block size.
- **cbs=BYTES** → Converts a given number of **BYTES** at a time.
- **conv=CONVS** → Converts file as per the following symbols-
  ascii: from **EBCDIC** to **ASCII**.
  ebcdic: from **ASCII** to **EBCDIC**.
  lcase: changes upper case to lower case.
  ucase: changes lower case to upper case.
- **sync** → Synchronizes I/O for both data and metadata.
- **count** → Indicates the number of blocks

- **ibs=BYTES** → Reads up to a given number of BYTES at a time.
- **of=FILE** → Writes to FILE.
- **if =FILE** → Reads from FILE.
- **seek=N** → Skips **N** obs-sized blocks at the start of the output.
- **skip=N** → Skips **N** obs-sized blocks at the start of input.

The mentioned **N** and **BYTES** can have values from the following multiplicative suffixes:

- **c**=1,
- **w**=2,
- **b**=512,
- **kB**=1000,
- **K**=1024,
- **MB**=1000*1000,
- **M**=1024*1024,
- **xM**=M,
- **GB**=1000*1000*1000,
- **G**=1024*1024*1024, and so on for T, P, E, Z, Y.
- Binary prefixes can be used, too:
- **KiB**=K,
- **MiB**=M, and so on.

Useful Options

- **if** → read the file instead of standard input.
- **of** → write the file instead of the standard output.

To learn more read, [The "dd" Command in Linux [7+ Practical Examples]]

11. cat

The **cat** command prints the contents of the file specified. Generally, **cat** (con**cat**enates) reads the contents of the files fed to its arguments and prints them serially on the **terminal**.

Syntax

cat [OPTION]... [FILE]...

Useful Options

- **-E, --show-ends** → Display **$** at end of each line.
- **-n, --number** → Displays line numbers when utilized.
- **-s, --squeeze-blank** → suppress repeated empty output lines.

To learn more read, [The "cat" Command in Linux [10 Practical Examples]]

12. less

The less command is a handy command in **Linux** that shows any file's contents one page at a time. This command is more useful when viewing a large file with many lines because it doesn't load the entire file, so it gives a fast loading speed. You can navigate through the lines of text using the **ARROW keys** and navigate page to page using the **SPACE bar**.

Syntax

less [OPTIONS]... (FILE_PATH or FILE_NAME)

Useful Options

- **-n --line-numbers** → when enabled it stops showing line numbers.
- **-N --LINE-NUMBERS** → displays line numbers at starting points of each line.

To learn more read, [The “less” Command in Linux [10 Practical Examples]]

13. head

The **head** command prints the **first** (by default 10 lines) few lines of a file. You can also print the first specific lines or bytes from a file using the **head** command in **Linux**.

Syntax

head [OPTION]... [FILE]...

Useful Options

- **-n** → Prints the first n lines.
- **-v, --verbose** → Prints message of what is being performed.

To learn more read, [The “head” Command in Linux [7 Practical Examples]]

14. tail

The **tail** command outputs the **last** (by default 10 lines) few lines of a file. But it can also be used to view only data instead of lines easily. Moreover, this command can be used to show specific file lines too.

Syntax

tail [OPTION]... [FILE]...

Useful Options

- **-n NUM** → Shows NUM number of lines in the command line output.
- **-n +NUM** → Shows all the lines after NUM number of lines.
- **-f, --follow** → Follows any addition to the file and updates the output in the command line.
- **-q, --quiet, --silent** → Doesn’t output the header line with the output.
- **-v, --verbose** → Outputs header file names.
- **-s, --sleep-interval** → Sets the timer interval of update for the **-f** option.

To learn more read, [The “tail” Command in Linux [7 Practical Examples]]

15. grep

The **grep** command is very useful for searching files and directories containing matching words or characters lines. Then it prints the entire line containing the match. It is the shortcut for “**Global Regular Expression Print**”.

Syntax

grep [OPTION]... PATTERNS [FILE…]

Useful Options

- **-c, --count** → Does not print any matches but rather the total occurring number.
- **-i, --ignore-case** → Ignores the sensitivity of cases.
- **-v, --invert-match** → Invert the sense of matching, to select non-matching lines.
- **-w, --word-regexp** → search for a whole word.

To learn more read, [The “grep” Command in Linux [10+ Practical Examples]]

16. locate

The **locate** command performs the search operation from an **existing database** and prints the results with the exact directory path.

Syntax
ls [OPTION]... PATTERN...
Useful Options
- **-c, --count** → Does not print any matches but rather the total occurring number.
- **-e, --existing** → Only prints the existing match.
- **-i, --ignore-case** → Ignores the sensitivity of cases.
- **-p, --nofollow** → Ignores punctuation and spaces when matching patterns.
- **-l, N, --limit=N** → Stop searching after limit matches have been found.

To learn more read, [The “locate” Command in Linux [7 Practical Examples]]

17. find
The **find** command in **Linux** performs the task of searching & listing files and directories in a directory hierarchy in real time, not like the **locate** command from an existing file.
Syntax
find [OPTIONS] [PATH] [EXPRESSION]
Useful Options
- **-type d/f** → Where, **d** (limits the search to only directories), **f** (limits the search to only files).
- **-size +n**,**n**,**-n** → Finds for a specific size **n**.
- **-name** pattern → Searches for the given pattern.
- **-exec** → Performs our customized tasks on the matches.

To learn more read, [The “find” Command in Linux [10+ Practical Examples]]

18. chmod
The **chmod** command is the abbreviation for **ch**ange **mod**e. The **chmod** command can be used to alter the permission attributes of system contents.
Syntax
- **Syntax 1**
  chmod [OPTION]... MODE[,MODE]... FILE…
- **Syntax 2**
  chmod [OPTION]... OCTAL-MODE FILE…
- **Syntax 3**
  chmod [OPTION]... - -reference=RFILE FILE…

Here, **MODE** can be specified alphabetically or in **OCTAL** number format.
Useful Options
- **-R, --recursive** → Changes files and directories recursively.
- **-v, --verbose** → Outputs a Diagnostic when a file is processed.
- **-c, --change** → Reports only when a change is made.
- **-f, --silent, --quiet** → Suppresses most error massages.
- **--version** → Outputs version information and exit.
- **--help** → Displays this help and exit.

To learn more read, [The “chmod” Command in Linux [6 Practical Examples]]

19. chown

The word **chown** stands for **C**hange **Own**er. As the name says, this command is used for changing owners. When I say **C**hange **O**wner I mean changing both file owner and group owner. The **chown** command is very useful when it comes to accessing the permission of a file or directory.

Syntax

chown [OPTION]... [OWNER] [:[GROUP]] FILE…
chown [OPTION]... --reference=RFILE FILE…

Useful Options

- **-R, --recursive** → Changes files and directories recursively.
- **-v, --verbose** → Outputs a Diagnostic when a file is processed.
- **-c, --change** → Reports only when a change is made.
- **-f, --silent, --quiet** → Suppresses most error massages.
- **--version** → Outputs version information and exit.
- **--help** → Displays this help and exit.

To learn more read, **The "chown" Command in Linux [8 Practical Examples]**

20. chgrp

The **chgrp** command, short for "**ch**ange **gr**ou**p**", is a useful tool in **Linux** for managing file **permissions** and **access control**. It alters the group name that a file or directory belongs to, which in turn affects the permissions that users have to access and modify the file.

Syntax

chgrp [OPTION]... [GROUP_NAME] [DIRECTORY/FILE_NAME]...

Useful Options

- **-c, --changes** → Works like verbose but reports only when a change is made.
- **-f, --silent, --quiet** → Can be used to suppress error messages.
- **-v, --verbose** → Generates a diagnostic report for each file that is processed.
- **--dereference** → Modifies the referent of each symbolic link (the default behavior) instead of the symbolic link itself.
- **--no-preserve-root** → Does not treat root directory '/' specially (the default).
- **--preserve-root** → Fails to operate recursively on root directory '/'.
- **--reference=RFILE** → Employs the group of RFILE instead of specifying a GROUP value.
- **-R, --recursive** → Performs actions on files and directories recursively.

To learn more read, **The “chgrp” Command in Linux [7 Practical Examples]**

21. tar

The **tar** command in **Linux** is used to archive files into a tar archive. Using the command options, you can also compress file size with **bzip2** & **gzip**. While creating a tar archive using the **tar** command, it always keeps the original files. This command is also used to extract files from the **tar archive** in different ways.

Syntax

tar [OPTION]... [FILE]

Useful Options

- **-c** → Creates uncompressed tar archive.
- **-x** → Extracts files from the tar archive.
- **-v** → Displays archived files of the tar archive.
- **-f** → Sets the name of a tar archive**.**
- **-t** → Displays the list of the tar archive.
- **-j** → Creates tar archive with bzip2.
- **-z** → Creates tar archive with gzip.
- **-r** → Updates files/directories of the existing tar archive.
- **--wildcards** → Specifies patterns of archived files of a tar archive.
- **--delete** → Deletes file/directory from the tar archive.

To learn more read, **The "tar" Command in Linux [12 Practical Examples]**

22. gzip

gzip command in **Linux** is used to **compress** & **decompress** files. The file size is decreased without losing data using this command in **Linux**. This command can also be used to test the integrity of the compressed files. Like some other compression commands, it replaces the original file while compression.

Syntax

gzip [OPTION]... [FILE]...

Useful Options

- **-f** → Forces compression of the file & removes the original file.
- **-k** → Compresses file but does not remove the original file.
- **-r** → Compresses all files of a specified directory.
- **-v** → Displays the name & percentage of reduction of file.
- **-d** → Decompresses file that is compressed.

To learn more read, **The "gzip" Command in Linux [9 Practical Examples]**

23. gunzip

The **gunzip** command in **Linux** is used to **decompress** the compressed files. It is used to check the **integrity** of the compressed files as well as the **percentage reduction** of the files in **Linux**. This command can also be used to display the **contents** of the compressed files. Like some other decompression commands, it **replaces** the **original file** while decompression.

Syntax

gunzip [OPTION]... [FILE]...

Useful Options

- **-k** → Compresses file but does not remove the original file.
- **-v** → Displays the name & percentage of reduction of file.
- **-l** → Displays information about the compressed files.
- **-c** → Displays contents of the compressed files.

To learn more read, **The "gunzip" Command in Linux [9 Practical Examples]**

24. zip

The **zip** command is a simple command that helps compress the file size to save disk space. Compressed files are easier to transfer and less time-consuming.

Syntax

zip [OPTIONS] File.zip File…

Useful Options

- **-r** → Zips a directory.
- **-d** → Removes specified files without unzipping.
- **-m** → Deletes the original file after zipping.
- **-x** → Excludes specified files from zipping.
- **-e** → Creates a password-protected zip archive.
- **-s** → Splits a large zip file into parts.

To learn more read, **The "zip " Command in Linux [9 Practical Examples]**

25. unzip

The **unzip** command extracts all the files & directories from the zip file in the **current** directory. This command also displays the list of files & directories it is extracting from the zip file in the **terminal**.

Syntax

unzip [OPTION]... [FILE]

Useful Options

- **-d** → Extracts files to the specified directory.
- **-l** → Lists contents of zip(compressed) file.
- **-t** → Checks if the zip file is a valid zip archive or not.
- **-v** → Displays detailed info of each file inside a zip archive.
- **-x** → Extracts only mentioned files.

To learn more read, **The "unzip" Command in Linux [7 Practical Examples]**

26. ssh

The "**ssh**" of the **ssh** command in **Linux** stands for "**Secure Shell**".This command in **Linux** is used to securely connect two computers (**client** and **server)** following the **SSH** protocol. You can execute commands from a remote location and transfer data using the **ssh** command. The data transfer is encrypted which prevents **hackers** from accessing them.

Syntax

ssh [OPTION]... username@ip_address

Useful Options

- **-4** → Only uses **IPv4 addresses**.
- **-6** → Only uses **IPv6 addresses.**
- **-C** → Compresses data.
- **-i** → Used to log in with a **private key**.
- **-l** → Specifies the **user**.
- **-p** → Used to specify a **port**.

To learn more read, **The "ssh" Command in Linux [18 Practical Examples]**

27. scp

The “**scp**” of the **scp** command in **Linux** stands for “**Secure Copy**”. It is used to copy files between two computers (**client** and **server**). It uses the **SSH** (**Secure Shell**) protocol which provides **secure** communication over an **insecure network**.

Syntax

The **scp** command in **Linux** has **two** syntaxes, one for **upload** and the other for **download**.
For **uploading**:
scp [OPTION]... SOURCE username@IP_address:DESTINATION
For **downloading**:
scp [OPTION]... username@IP_address:SOURCE DESTINATION
Here,

A. **SOURCE** means the path to the file **which** will be copied.
B. **DESTINATION** means the path **where** will the file be copied.
C. **username** denotes the **username** of the **server.**
D. **IP_address** denotes the **IP address** of the **server.**

Useful Options

- **-4** → Only uses **IPv4**.
- **-6** → Only uses **IPv6**.
- **-C** → **Compresses** files.
- **-P** → Used for **logging** in with a **specific port**.
- **-v** → Enables **verbose** mode.
- **-r** → Copies **recursively**.

To learn more read, [The “scp” Command in Linux [4 Practical Examples]]

28. sync

The **sync** command allows putting your data in sync both in permanent storage & volatile memory. It makes sure all buffered modifications made to file metadata and data are written to the underlying file systems. Every **Linux** user should use the **sync** command before halting the processor in an unusual manner to save all the cache data.

Syntax

sync [option]... [file]...

Useful Options

- **-d, --data** → Syncs only the data of the filesystem, and essential metadata.
- **-f, --file-system** → Syncs the filesystem containing the specified files.

To learn more read, [The “sync” Command in Linux [8 Practical Examples]]

29. rsync

The **rsync** command in **Linux** is used to **synchronize files** and **directories** between two computers (**local host** and **remote host**). At first, it uses **SSH(Secure Shell)** to connect to a **remote host** and decides which part of **local files** (or **directories**) is needed to be copied. The **rsync** command is an efficient way to transfer files as the manual method is tedious.

Syntax

rsync [OPTION]... SOURCE DESTINATION

Useful Options

- **-a** → Copies **attributes** of data.

- **-h** → Shows data in a **human-readable** format.
- **-v** → Enables **verbose mode**.
- **-z** → **Compresses** data.
- **--progress** → Shows **progress report**.
- **--ignore-existing** → Ignores already existing files in the **server**.
- **--delete** → Removes files from the **server** when they are deleted in the **client**.
- **--remove-source-files** → Automatically deletes files after syncing.
- **--chmod** → Sends file **permissions**.
- **--dry-run** → Performs a **dry run**.
- **--max-size** → Sets maximum file **size**.
- **--bwlimit** → Sets maximum **bandwidth**.

To learn more read, The "rsync" Command in Linux [10 Practical Examples]

30. man

The **man** command in **Linux** stands for **manual**. Upon execution, it will display a **manual** page or **documentation** of a specified **Linux** command. It displays information like the **synopsis**, **description**, **options**, **exit status**, **authors**, **copyright,** etc.

Syntax

man [OPTION]... [Command_NAME]...

Useful Options

- **SECTION COMMAND** → Shows the specific SECTION of a COMMAND.
- **-k KEYWORD** → Search for the Keyword in the whole manual page and shows all the matches.
- **-f KEYWORD** → Looks for a short description of any Keyword or Command.
- **-d, --default** → Resets the man command behavior to default.
- **-i, --ignore-case** → Ignore case sensitivity of the command.
- **-I, --match-case** → Looking inside the man page with case sensitivity.
- **-a, --all** → Shows all manual pages that match the specific keyword or command.

To learn more read, The "man" Command in Linux [6 Practical Examples]

31. alias

The **alias** command in **Linux** is used as a **shortcut** that takes a **string** that can **execute** a command and **replaces** it with **another string** of **text**. This command can **replace** a large complex command with a relatively **simple string** and reuse that command many times. Professionals use this command quite a lot to increase their functionality.

Syntax

alias [option] [name]='[value]'

Note: You must keep the **command/value** inside a **quotation**. You can use either a **single quotation** or a **double quotation**.

Useful Options

- **-p** → **Lists** all available aliases in the command line.
- **--help** → Shows the **help page** of the alias command.

To learn more read, The "alias" Command in Linux [3 Practical Examples]

32. unalias

The **unalias** command is used to eliminate items from the list of aliases for the current user, specifically those added during the current login session. Permanent aliases may also be temporarily suppressed; however, they will be restored once the user logs in again.

Syntax

unalias [OPTION] [ALIAS_NAME]...

Useful Options

- **-a** → Removes all **alias** for the current user for the current shell for the current session.

To learn more read, The "unalias" Command in Linux [3 Practical Examples]

33. history

The **history** command displays the previously executed commands and a number to imply a serial is also shown with this command. This command will help you recall any command you've used before but forgot about it.

Syntax

The **history** command has a really simple syntax. Just type **history** and that's it. You can also add a number after the command to show how many previously executed commands you want to see in the command line.
But on the **man page**, there is **no syntax** available for the **history** command.
Note: Generally the more you use **history**, the larger the **history** command will become. So, I recommend you **pipe** this command with the **less command** for an interactive view.

Useful Options

- **-d EVENT_NUM** → Deletes the instance with the event number of EVENT_NUM.
- **-c** → Deletes all the history.

Note: Deleting only one instance won't remove the event number, the next commands in the list will just take its place.
To learn more read, The "history" Command in Linux [6 Practical Examples]

34. sudo

The **sudo** command allows a permitted user to execute a command as the **superuser** or another user, as specified by the security policy. It is considered analogous to the "**run as administrator**" process of Windows.

Syntax

sudo [OPTION]...[COMMAND]...

Useful Options

- **-D directory, --chdir=directory** → Executes the command in the specific directory.
- **-e** → Edits one or multiple files instead of executing commands.
- **-h** → Displays a short help message and exits.
- **-l** → Runs specific commands as the root user.
- **-k** → Kills the user's timestamps.
- **-u user, --user=user** → Executes the command as a user other than the specific default user.

To learn more read, [The “sudo” Command in Linux [8 Practical Examples]]

35. su

The **su** command is an excellent tool to **secure multiple-user systems**, enabling you to **switch** from one user to another.

Syntax

su [options] [-] [user [argument...]]

Useful Options

- **-c, --command=command** → Passes the command to the shell with the **-c** option.
- **-f, --fast** → Passes **-f** to the shell, which may or may not be helpful, depending on the shell.
- **-g,--group=group** → Specifies the primary group. This option is available to the root user only.
- **-G, --supp-group=group:** Specifes a supplementary group. This option is available to the root user only. The first specified supplementary group is also used as a primary group if the **--group** option is not specified.
- **–h, –help** → Shows the help file for the su command.
- **-, -l, --login** → Starts as a login shell with an environment similar to a real login.
- **–p, –preserve–environment** → Preserves the shell environment.
- **–s, –shell** → Allows you to specify a different shell environment to run in.

To learn more read, [The “su” Command in Linux [6 Practical Examples]]

36. ip

The **ip** command in **Linux** allows users to display as well as manipulate routing tables, network interfaces, and devices. You can extract various network information using this command. In addition to other applications, you can also assign new IP addresses using this command.

Syntax

ip [OPTION]... OBJECT COMMAND

Note: As a parameter, if no specific **COMMAND** is mentioned then the default commands **list** or, **help** are assumed. Moreover, the following actions can be passed as the possible **COMMAND**:

add
delete/del
show

Useful Options

- **-br, --brief** → Displays only basic information.
- **-h, --human** → Displays human-readable values.
- **-4** → Filters only the IPv4 addresses.
- **-6** → Filters only the IPv6 addresses.

To learn more read, [The “ip” Command in Linux [9+ Practical Examples]]

37. iptables

The **iptables** command in **Linux** is used to set up, maintain and inspect the **tables** of IP packet filters. These tables contain built-in **chains** which are basically a set of **rules** to

match packets going through the network. Each rule defines the action performed on a set of packets.
The **Linux-based Firewall** is maintained by the **IPv4/IPv6 packet** filtering command **iptables**. The **Firewall** rules within this table specify **targets** for each packet. **Targets** can be any of the following:

- ➔ **ACCEPT:** Lets a packet come through the network.
- ➔ **DROP:** Rejects the packet from entering the network.
- ➔ **QUEUE:** Receives the packet and passes it to the queue.
- ➔ **RETURN:** Stops the current chain and resumes the next rule of the previous chain.

Syntax

iptables [-t table] [OPTION]...
The present **Linux** distribution comes with **five** independent **tables**. A description of each **table** is given below.

A. **filter:** It is the default table displayed when the **-t** option is not passed. This table has the following built-in chains: **INPUT**, **OUTPUT,** and **FORWARD**.
B. **nat:** This table is used when packets with a new connection arrive. Includes **INPUT**, **OUTPUT**, **PREROUTING**, and **POSTROUTING** built-in chains.
C. **mangle:** It is consulted for specialized packet alteration. The two built-in chains within this table are **OUTPUT** and **PREROUTING**.
D. **raw:** Configures exemptions from connection tracking. Built-in chains include **OUTPUT** and **PREROUTING**.
E. **security:** Used for **Mandatory Access Control (MAC)** networking rules implemented by **Linux Security Modules**. It contains **INPUT**, **OUTPUT,** and **FORWARD** built-in chains.

Useful Options

The iptables command options are divided into three main groups: **COMMAND**, **PARAMETERS,** and **OTHER OPTIONS**.

COMMANDS

These options indicate a certain action to be performed. Only one of the following commands can be applied at a time.

- **-A, --append** → Appends rule/s to the end of a chain.
- **-C, --check** → Checks if a specified rule exists.
- **-D, --delete** → Deletes rule/s from a specified chain.
- **-I, --insert** → Inserts rule/s to a specified chain.
- **-L, --list** → Display all rules from a chain.

PARAMETERS

The parameters are used to specify a rule.

- **-d, --destination** → Specifies destination.
- **-p, --protocol** → Specifies the protocol of a rule or, packet.
- **-j, --jump** → Specifies the target of a rule upon packet matching.
- **-s, --source** → Specifies source.

OTHER OPTIONS

These are the general options for the **iptables** command in **Linux**.

- **--line-numbers** → Adds a line number to the beginning of each rule.
- **-n, --numeric** → Displays numeric IP addresses and port numbers.
- **-v, --verbose** → Displays verbose output.

To learn more read, **The “iptables” Command in Linux [6 Practical Examples]**

38. ping

The **ping** command shows the connection latency between the host and servers. The **ping** command takes a URL, and IP address as arguments and shows the latency between the local machine and the server.

Syntax

ping [OPTION]... <Destination>

Useful Options

- **-c COUNT** → **Limits Ping requests** for packets after **COUNT** number.
- **-T** → **Timestamp** option.
- **-w** → **Limits** the total time.
- **-D** → Prints **timestamps** before each line.
- **-f** → **Floods** the network with packets.
- **-i** → Fixes the **interval** between each packet sent.
- **-q, Quite an output** → Nothing is shown in the command line.
- **-s** → Fixes the **packet size** sent.
- **-v** → Provides **verbose** information.
- **-4** → Uses **IPv4** only.
- **-6** → Uses **IPv6** only.

Note: To exit from the **ping** command, press **CTRL+C**.

To learn more read, **The "ping" Command in Linux [9 Practical Examples]**

39. traceroute

The **traceroute** in **Linux** is used to track the path a **data packet** takes. A data packet passes multiple **hops** to reach its **destination**. If there are any issues in any network, the **traceroute** command can be used to find the problem and solve it. That's why the **traceroute** command in **Linux** is a useful **network troubleshooting tool**.

Syntax

traceroute [OPTION]... [HOST]

Note: Here, **HOST** denotes the name of a **website** or **IP address** (for instance-linuxsimply.com).

Useful Options

- **-n** → Hides **hostname.**
- **-f**, **--first_ttl** → Specifies **initial hop** number.
- **-m**, **--max_ttl** → Sets **maximum hops** number.
- **-q, --nprobes** → Limit the **number of tests.**
- **-w**, **--waittime** → Limits **response time.**
- **-4** → Uses only **IPv4.**
- **-6** → Uses only **IPv6.**
- **-F** → Enables "**Don't Fragment**" mode.
- **-g**, **--gateway** → Routes through a certain **IP address.**
- **-d** → Disables resolution of the **IP address.**
- **-t**, **--tos** → Specifies the maximum number of **hops** for a **packet.**
- **-p**, **--port** → Tests a **port.**

To learn more read, [The “traceroute” Command in Linux [13 Practical Examples]]

40. wget

The **wget** which stands for **“w**eb **get”,** allows users to download files from networks using protocols like **HTTP**, **HTTPS**, **FTP**, and **FTPS**. The non-interactive downloading occurs in the background while a user is either disconnected or busy with another process. It utilizes recursiveness by continuously attempting until a file is entirely retrieved. The **wget** Command in **Linux** also downloads complete **HTML**/ **XHTML** pages and replicates the content structure to browse the websites locally.

Syntax

wget [Option] … [URL] …

Useful Options

- **- b** → Downloads file in the background.
- **- c** → Continues interrupted downloads,
- **- i** → Downloads multiple files.
- **- t, --tries = [number]** → Fixes the number of retry attempts for interrupted downloads.
- **- m, -- mirror** → Mirrors a website locally for offline browsing.
- **- r** → Turns on Recursive retrieval of files, the default value = 5.
- **- A, -- accept “*.<FileType>”** → Downloads only specific file type.
- **- R, -- reject “*.<FileType>”** → Rejects downloads for specific file type.
- **- O, Upper case** → Downloads file with a different name.
- **- p** → Downloads in a specific directory.
- **- - limit-rate = [amount]** → Limits bandwidth/speed while downloading.
- **- - no-check-certificate** → Ignores **SSL** certification check.

To learn more read, [The “wget” Command in Linux [14 Practical Examples]]

41. curl

The **curl** command in **Linux** stands for “**C**lient **URL**”. It is a file transfer **command-line** tool powered by **libcurl,** a free client-side **URL** transfer library. This means it allows you to send or receive data over the network. It supports most of the network protocols (**HTTP**, **FTP**, **SFTP**, **SMTP**, **TFTP**, **POP3**, **TELNET**, **FILE**, etc). You can use the **curl** command in **Linux** to transfer single or multiple files at once.

Syntax

curl [options / URLs]

Note: In the above syntax “ **/** ” indicates that you can write **URL**s with or without **options** using the **curl** command in **Linux**.

Useful Options

- **-C** → Continues interrupted downloads.
- **-I** → Outputs the HTTP header only.
- **--limit-rate <amount>** → Limits bandwidth/speed while downloading.
- **-libcurl <filename>** → Output C source code of the URL.
- **-L** → Redirects if the requested page is moved.
- **-o** → Downloads file with default different name.
- **-O, Upper case** → Downloads file with a different name.

To learn more read, [The “curl” Command in Linux [9 Practical Examples]]

42. df

The "**df**" comes from "**disk free**", which displays the size, used & available space, and mounted on the information of the filesystem. As you know, no machine has unlimited disk space, and checking available space is necessary sometimes.

Syntax

df [OPTION]... [FILE]...

Useful Options

- **-a, --all** → Displays **all** file systems including pseudo, duplicate, and inaccessible.
- **-h, --human-readable** → Shows sizes in powers of **1024**.
- **-H, --si** → Shows sizes in powers of **1000**.
- **--output** → Prints output in a **customized** format. See the manual page for details.
- **--total** → Ignores insignificant entries to available space and shows a **grand total**.
- **-t, --type** → Only prints a **specific type** of file system. See the manual page for details.
- **-x, --exclude-type** → Prints file system **excluding** specific file system type. See the manual page for details.
- **-T, --print-type** → Prints file system type.
- **-i, --inodes** → Llists **inodes** information.

To learn more read, The "df" Command in Linux [11 Practical Examples]

43. du

du command means **d**isk **u**sage. Using this command the total usage of the disk and the disk usage of the different files are shown in the terminal.

Syntax

du [OPTION]... [FILE]...

Useful Options

- **-a** → Shows the total disk usage.
- **-ah** → Shows all files disk usage.
- **--time** → Shows the last modification time.
- **-B, --block-size** → Defines a new scale.
- **-c, --total** → Shows **grand total** at the end.
- **-S, --separate-dirs** → Doesn't show **subdirectories**.
- **--si** → Uses **1000** as the scale.
- **-s, --summarize** → Shows only **total disk usage.**

To learn more read, The "du" Command in Linux [14 Practical Examples]

44. ps

The **ps** (**P**rocess **S**tatus) command shows the process status and information about that process.

Syntax

ps [OPTIONS]

Useful Options

- **-a, --all** → Shows all the processes including hidden processes.
- **-r** → Shows all the running processes.
- **--pid PID** → Shows the specific process according to PID.

To learn more read, [The “ps” Command in Linux [9+ Practical Examples]]()

45. kill

The **kill** command in **Linux** is one of the handiest tools which can be used to terminate one or multiple processes from the **Terminal**. It sends a signal to the process, which ultimately closes or terminates, or kills a particular process or group of processes. If the user doesn’t specify any signal to be sent along with the **kill** command, then the default **TERM** signal is sent that terminates the process. It is a very useful tool for **multitaskers** who handle multiple programs simultaneously.

Syntax

kill command is a built-in command that takes **options** and **PID** (process ID) or process name. The syntax of the **kill** command is as follows.

```
kill [options]  [PID]...
```

Useful Options

- **-l** → Shows the available option of the **kill** command.
- **-- SIGHUP -1** → Reloads the process.
- **-- SIGKILL, -KILL, -9** → Terminates the process.
- **-- SIGTERM, -TERM -15** → Terminates the process.

To learn more read, [The “kill” Command in Linux [4+ Practical Examples]]()

46. shutdown

The **shutdown** command in **Linux** is useful for **safely shutting down** the **system**. The machine can be shut off **immediately** or on a **schedule** with a 24-hour format. And When executing the command, all currently active **users** and processes are informed that the **system** is **shutting down**. New **login** attempts are not permitted when it is initiated.

Syntax

```
shutdown [OPTIONS]... [TIME] [MESSAGE]
```

Useful Options

- **-r** → Asks for the system to be restarted following a shutdown.
- **-h** → Equivalent to halt unless power is off depending on the specified option.
- **-H** → Stops the operating system.
- **-P** → Powers off the machine.
- **-c** → Cancels a shutdown that is currently in effect. This can be used to invoke a shutdown with a time argument other than "+0" or "now."
- **-k** → Does not halt, power off or reboot, only send out the warning messages.

To learn more read, [The “shutdown” Command in Linux [7 Practical Examples]]()

47. reboot

Using the **reboot** command in **Linux**, any user can restart the whole system using the command line interface.

Syntax

```
reboot [OPTIONS...]
```

Useful Options

- **-d/--no-wtmp** → Does not write **wtmp** record during reboot.
- **--help** → Prints a short help text and then exits.
- **-f/--force** → Forcefully reboots the system.
- **--no-wall** → Does not send any wall message during reboot.
- **-p/--poweroff** → Power off the machine.

To learn more read, The "reboot" Command in Linux [3 Practical Examples]

48. mount

The **mount** command in **Linux** is used to attach a removable storage device or filesystem to an existing **directory** making them accessible. All files on the device are placed in a big tree-like hierarchy, starting from the **root directory** (noted as **/**). Just like the root holds the tree, all the child filesystems emerge like the branches from the **root directory**. And the **mount** command helps to mount the device filesystem to the **Linux** filesystem (large tree structure).

Syntax

mount [OPTION]... <device> <target>

Note: Only a **superuser** can mount filesystems. So to run the command use **'sudo'** (**S**ubstitute **U**ser **DO**) before the command. Temporarily it allows you to appoint your current user to have **root privileges**.

Useful Options

- **-a --all** → Mounts all files listed in /etc/fstab.
- **-o --options** → Limits the file system set that **-a** applies to.
- **-t --type** → Indicates the file system type.
- **-T** → Specifies an alternative /etc/fstab file.
- **-l** → Lists all the files mounted and added labels to each device.
- **-L --label** → Mounts the partition with the specified label.
- **-M** → Moves the mounted file to another location.
- **-r** → Mounts the filesystem in read-only mode.
- **-R** → Remounts a filesystem to a different location, making its contents available in both places.
- **-v** → Mounts verbosely, describing each operation.

To learn more read, The "mount" Command in Linux [15 Practical Examples]

49. date

The **date** command is one of the most frequently used commands in **Linux**. It is used to get or set the date and time of the system. This command can also be used to find out the modification time of files and can calculate the past and future dates as well. By default, the date command displays the **Central Standard Time (CST)** Time Zone. But you can adjust the date view on your terminal as well.

Syntax

date [OPTION]... [+FORMAT]

Useful Options

- **-d/--date=STRING** → Converts time from the string such as today, tomorrow, 1 year ago, etc.
- **-f/--file=DATEFILE** → Runs date command for each line of the file.
- **-s/--set=STRING** → Sets time described by the string.

- **-r** → Returns the last modification time of a file.
- **TZ** → Prints out time in a given time zone.
- **-u** → Returns the time in the UTC time zone.

To learn more read, **The “date” Command in Linux [8+ Practical Examples]**

50. ncal

The **ncal** command in **Linux** is a **Unix-based** tool to view **calendars**. However, it doesn’t come by default in most systems and may require to install manually. It is the successor of **cal** command. It provides advanced formatting compared to the **cal** command. In addition to that, it has some extra features like showing **holidays**, the **lunar calendar,** etc.

Syntax

cal [OPTION]... [ARGUMENT]...

Useful Options

- **-h** → Removes the **highlight** on the current date.
- **-y** → Used to view the **calendar** of a **specific year**.
- **-m,** → Shows the calendar of a **specific month** of the **current year**.
- **-3,** → Prints the calendar of **previous**, **current,** and **following** months.
- **-w,** → Shows the **week number**.
- **-B,** → Displays a specific number of months **before** the **current month**.
- **-A,** → Prints a specific number of months **after** the **current month**.

To learn more read, **The “ncal” Command in Linux [8 Practical Examples]**

51. whoami

The **whoami** command simply displays the currently logged-in user. It can be handy while working on a machine with multiple users. It displays the **username** of the effective user in the current shell. Moreover, It is useful in **bash scripting** to show who is running the script.

Syntax

whoami [OPTION]...

Useful Options

- **--help** → Displays the **man** page and exit.
- **--version** → Output version information and exit.

To learn more read, **The “whoami” Command in Linux [5 Practical Examples]**

52. id

The **id** command in **Linux** displays current user and group information according to the application of the command. You can modify the usage by passing a specific **username** or **options**.

Syntax

id [OPTION]… [USER]

Useful Options

- **-g/--group** → Displays only the effective group ID.
- **-G/--groups** → Displays all the group IDs of the user.
- **-r/--real** → Displays real IDs.
- **-n/--name** → Displays names.

- **-u/--user** → Display only the effective user ID.
- **-z/--zero** → Delimits entries with a NULL character.
- **-Z/--context** → Displays security context of the process.

To learn more read, [The “id” Command in Linux [7+ Practical Examples]]()

53. uname

The **uname** command, short for **UNIX name**, is a useful tool in **Linux** for getting information about the hardware and software of the current system including the **operating system**, **kernel**, **processor architecture**, and network node **hostname**.

Syntax

uname [OPTION]...

Useful Options

- **-a, --all** → Prints all available information about the system, including the kernel name, network node hostname, kernel release, kernel version, machine hardware name, and processor type (excluding the **-p** and **-i** options if they are unknown).
- **-s, --kernel-name** → Displays the kernel name.
- **-n, --nodename** → Displays the network node hostname.
- **-r, --kernel-release** → Displays the kernel release.
- **-v, --kernel-version** → Displays the version of the kernel.
- **-m, --machine** → Displays the machine hardware name.
- **-p, --processor** → Displays the type of processor.
- **-i, --hardware-platform** → Displays the hardware platform (non-portable).
- **-o, --operating-system** → Displays the operating system name.

To learn more read, [The “uname” Command in Linux [11 Practical Examples]]()

54. Bash

The **bash** command in **Linux** is a popular tool for starting a shell from the beginning, creating a new login shell, running a bash script from the terminal, etc.

Syntax

bash [OPTIONS] [COMMAND_STRING / FILE]

Useful Options

- **-c** → Creates a new shell different from the current shell and performs the later task on the new shell.
- **-i** → Makes the shell interactive.
- **-l** → Makes bash act as if it had been invoked as a login shell.
- **-r** → Makes the shell restricted)
- **-v** → Makes shell print input lines as they are read.
- **-x** → Prints commands and their arguments as they are executed.

To learn more read, [The “bash” Command in Linux [5 Practical Examples]]()

55. exit

The **exit** command is used to **end or close the current login session** in the **Linux** ecosystem. **Shell** or **bash** jobs as well as the **Linux CLI** can be closed using this simple command.

Syntax

exit [INTEGER_VALUE] USER_NAME

Useful Options

There are no OPTIONS available for the exit command in Linux.
To learn more read, [The “exit” Command in Linux [5 Practical Examples]]

56. clear

The **clear** command in **Linux** clears your **terminal** screen if possible. It also **scrolls down** the **terminal** screen to clear it. It observes the **terminal** type in the **environment** given by the environment variable **TERM**, and then in the **terminfo** database to decide how the screen can be cleared. **Standard output** is also written by the **clear** command in **Linux**.

Syntax

clear [-OPTION]

Useful Options

- **-T** → Indicates the type of terminal.
- **-V** → Reports the version of ncurses.
- **-x** → Scrolls down the terminal, does not remove history.

To learn more read, [The “clear” Command in Linux [3 Practical Examples]]

57. echo

The **echo** command in **Linux** takes a string of text and **displays** the output in the command line. Even though the command may look like a straightforward command, you can also **pipe** or **redirect** the echo command with another command to do certain things like create/edit files if you want.

Syntax

Just add your text after the **echo** command and the command will **print** everything after the command in the command line.
echo [SHORT-OPTION]... [STRING]...
echo LONG-OPTION
Note: You can also keep the text inside **double** or **single** quotes. By default, the system takes **no quotes** as **double quotes**. But in case you use **single** quotes, the command line will print **exact text** inside without considering any **variables** or **commands** inside.

Useful Options

- **-n** → Does not **output** the **trailing** new line.
- **-e** → Enables **interpretation** of the **backslash** escapes.
- **-E** → Disables **interpretation** of the **backslash** escapes.

To learn more read, [The “echo” Command in Linux [7 Practical Examples]]

58. sed

The **sed** command is used to work in a **stream editor** called **Sed** in **Linux. Sed** editor can make basic **text transformations** in a file. With the help of the **sed** command, you can edit text files without opening files.

Syntax

sed [OPTION]... [COMMAND] [FILE]

Useful Options

- **-i** → Modifies and saves the original file.
- **-v** → Displays version information, and exit.

To learn more read, [The “sed” Command in Linux [7 Practical Examples]]

59. Make

The **make** command in **Linux** can build any **program** from the terminal by **compiling** or **recompiling** the pieces of a large **program** where necessary.

Syntax

make [OPTION]... [TARGET]...

Useful Options

- **-b, -m** → Options are used to ignore for compatibility with other versions of the make.
- **-B, --always-make** → Forces a rebuild of all targets, ignoring any existing dependencies and any files already builds.
- **-f=file, --file=file, --makefile=FILE** → Ensures the usage of the file as a makefile.
- **-e, --environment-overrides** → Gives variables taken from the environment precedence over variables from makefiles.
- **-i, --ignore-errors** → Ignores all errors in commands executed to remake files.
- **-k, --keep-going** → Used to continue as much as possible after an error. While the target that failed, and those that depend on it, cannot be remade, the other dependencies of these targets can be processed as well.
- **-R, --no-builtin-variables** → Doesn’t define any built-in variables.
- **-s, --silent, --quiet** → Does not print the commands as they are executed.
- **--warn-undefined-variables** → Warns when an undefined variable is referenced.
- **--trace** → Prints the information about the disposition of each target.

To learn more read, [The “make” Command in Linux [5 Practical Examples]]

60. xargs

The **xargs** command in **Linux** is a helpful tool for **processing** large lists of inputs and **executing** commands for each item in the list. It allows you to efficiently process inputs from other commands and automate repetitive tasks.

Syntax

xargs [OPTION]... [COMMAND [INITIAL-ARGUMENTS]]

Useful Options

- **-a, --arg-file=FILE** → Reads input from a file instead of standard input.
- **-I, --replace[=STRING]** → Replaces occurrences of a string with the arguments passed to **xargs**.

- **-L, --max-lines=NUM** → Passes no more than NUM arguments to each invocation of the command.
- **-n, --max-args=NUM** → Passes no more than NUM arguments to each invocation of the command.
- **-p, --interactive** → Prompts the user for confirmation before executing each command.
- **-r, --no-run-if-empty** → Does not run the command if the input is empty.
- **-s, --max-chars=NUM** → Passes no more than NUM characters to each invocation of the command.
- **-t, --verbose** → Prints the command line before executing it.

To learn more read, [The “xargs” Command in Linux [5 Practical Examples]]

61. exec

The **exec** command replaces the current terminal process with a new command that takes over the **memory**, **process ID**, and some other resources. This command in **Linux** is often used to **execute** specific **programs** or **commands** without creating a new **process**. If the **exec** command is executed without any other command, it terminates the terminal. So, the **bash** command needs to be executed before that.

Syntax

exec [OPTION]... [ARGUMENT]...

Useful Options

- **-c** → Executes command with an empty environment.

To learn more read, [The “exec” Command in Linux [8 Practical Examples]]

62. awk

The **awk** command in **Linux** is a **scripting** language. It is used for **text processing**, **data manipulation,** and **report generation** while working in the command line. The **awk** allows the user to use **variables**, **numerical functions**, **string functions,** and **logical operators** without further **compilation**. It enables a programmer to develop a **precise** but **impactful script** that can do a specific **task** for the matched line of every record inside a **file**.

Syntax

awk '{action}' [file_name.txt]

Note: In **action**, you have to specify the **action** you want, and **file_name.txt** is inside a squared bracket, meaning the file is not mandatory.

Useful Options

- **-F value** → Sets the field separator, FS, to value.
- **-f file** → Program text is read from the file instead of from the command line. Multiple -f options are allowed.
- **-v var=value** → Assigns a value to program variable var.

To learn more read, [The “awk” Command in Linux [11+ Practical Examples]]

63. sort

The **sort** command in **Linux** is used to sort lines of text files. It is capable of sorting alphabetically and numerically, in ascending or descending order. It considers all contents as **ASCII** and rearranges them based on **ASCII** value.

Syntax

sort [OPTION]... [FILE]...

Useful Options

- **-f, --ignore-case** → Makes **sort** case insensitive.
- **-M, --month-sort** → Sorts by month.
- **-n, --numeric-sort** → Compares based on the numerical value.
- **-R, --random-sort** → Shuffles the contents of the file.
- **-r, --reverse** → Reverses the comparison.
- **-c, --check** → Checks whether a file is sorted or not.
- **-k, --key** → Sorts by key.
- **-o, --output** → Used to print output in another file.
- **-u, --unique** → Removes duplicates excluding unique.

To learn more read, **The "sort" Command in Linux [16 Practical Examples]**

64. cut

The **cut** command in **Linux** is a utility tool for extracting a range of information from a file. Using the command you can slice texts based on **byte/s**, **character/s**, **field/s,** or **delimiters**. You can apply these parameters with the help of the **options**. You must type at least one of the available **options** after the **cut** command. Otherwise, the system will show error messages.

Syntax

cut OPTION... [FILE]...

Useful Options

- **-b, --bytes** → Extracts only these bytes from the file.
- **-c, --characters** → Extracts only these characters from a file.
- **-d, --delimiter** → Extracts contents between assigned delimiters from a file.
- **-f, --fields** → Extracts only these fields from a file.

The options above mentioned accept only one type of range at a time. Types of ranges are defined as follows:

- **N-** → Extracts only the Nth integer, counting starts from 1.
- **N-M** → Extracts from the Nth integer to the Mth integer.
- **M-** → Extracts from the Mth integer to the end of the file.

To learn more read, **The “cut” Command in Linux [8 Practical Examples]**

65. paste

The **paste** command in **Linux** is a helpful tool for merging lines from multiple files. It takes input from multiple files and pastes the corresponding line together, separated by a delimiter. This command is very useful for **merging** columns of data from multiple files to a single file for further processing.

Syntax

paste [OPTION]... [FILE]...

Useful Options

- **-d, --delimiters=LIST** → Reuses characters from **LIST** instead of TABs.
- **-s,--serial** → Pastes one file at a time instead of in a parallel manner.
- **-z, --zero-terminated** → Merges files separated by a **NULL** character instead of a newline character.

To learn more read, [The “paste” Command in Linux [6 Practical Examples]]

66. nano

Nano is **ano**ther text editor. It is a simple and intuitive text editor that has many different shortcuts and is very light. It comes with the basic Ubuntu install.

Syntax

Simply open any file in the nano editor using,

nano [OPTION]... [FILE]...

Or, you can put the cursor on a **specific line(or column)** by adding the **line number (or column)** with a **plus (+)** sign. You can also make the string **case insensitive** by adding “**c**” or “**r**” followed by a **plus(+)** sign. For that, use the syntaxes,

nano [options] [[+line[,column]] file]...

nano [options] [[+[crCR](/|?)string] file]...

Useful Options

- **-l, --linenumbers** → Shows line numbers on the left.
- **-m, --mouse** → Enables mouse support.
- **-v, --view** → Open a file in **read-only** made.
- **-i, --autoindent** → Indents a new line automatically.
- **-e, --emptyline** → Keeps line below title bar blank.
- **-g, --showcursor** → Makes the cursor visible.

To learn more read, [The “nano” Command in Linux [13 Practical Examples]]

67. diff

The **diff** command in **Linux** compares files line by line**.** It can also differentiate between the contents within directories.

Syntax

diff [OPTION]... FILES

Useful Options

- **-B** → Ignores blank lines.
- **-c** → Displays copied context.
- **-i** → Ignores case differences.
- **-r** → Recursively compares subdirectories.
- **-s** → Reports if files are identical.
- **-q** → Reports if files are different.
- **-u** → Displays unified context.
- **-w** → Ignores all white spaces.
- **-y** → Displays results in two columns.

To learn more read, [The “diff” Command in Linux [11 Practical Examples]]

68. patch

The **patch** command in **Linux** updates any source code of any program by identifying the data that needs to be changed/updated from a new file.

Syntax

patch [OPTIONS] [ORIGINAL_FILE] [PATCHFILE]

Useful Options

- **-b, --backup** → Creates backup files. When patching a file, rename or copy the original instead of removing it.
- **-c, --context** → Specifies the number of context lines to be included in the patch file.
- **-d dir, --directory=dir** → Changes to the directory "**dir"** immediately before doing anything else.
- **-D define, --ifdef=define** → Uses the **#ifdef ... #endif** constructs to mark changes, with define as the differentiating symbol.
- **--dry-run** → Prints the results of applying the patches without changing any files.

To learn more read, **The "patch" Command in Linux [4 Practical Examples]**

69. wc

The "wc" of the **wc** command comes from "**Word Count**".It shows the word count, characters count, and the number of lines of file/files provided to it as an argument.

Syntax

wc [OPTION]... [FILE]...

Useful Options

- **-w**, **--words** → Displays the word counts.
- **-l**, **--lines** → Shows the line counts.
- **-m**, **--chars** → Displays the character counts.
- **-c**, **--bytes** → Shows the byte counts.
- **-L**, **--max-line-length** → Prints the length of the longest line.
- **--files0-from** → Takes input from the files specified by Nul-terminated names in a file.

To learn more read, **The "wc" Command in Linux [15 Practical Examples]**

70. tee

The **tee** command in **Linux** is used to read standard input and write to another file apart from standard output. This command is very useful to store the intermediate output of multiple commands.

Syntax

tee [OPTION]... [FILE]...

Useful Options

- **-a** → Appends output to the given files.
- **-i** → Ignores Interruptions.
- **-p** → Diagnoses errors in writing to non-pipe output.

To learn more read, **The "tee" Command in Linux [4 Practical Examples]**

71. ln

The **ln** command in **Linux** is used to **create links (shortcuts)** to the source file/directory. There are **two types** of links and these are **hard links** & **symbolic links.** When a file is linked with a **hard** link, if you make any changes to any of the files this will reflect on both files whereas when a file is linked with a **symbolic** link if any of the files/directories are moved or deleted, this will break the link between the files.

Syntax

ln [OPTION]... [SOURCE_FILE] [LINKED_FILE]

Here, **SOURCE_FILE** represents the name of the source file, and **LINKED_FILE** denotes the name of the linked file created with the **ln** command.

Useful Options

No option is needed to create a hard link to the file.

- **-s** → Creates symbolic links to files/directories.
- **-b** → Creates a backup of the file.

To learn more read, The “ln” Command in Linux [6 Practical Examples]

72. which

The **which** command in **Linux** can locate a command if it is passed as an **argument**. It takes the **argument** and searches for the corresponding name in the **$PATH environment variable** of executable files.

Syntax

which [OPTION] filename ...

Useful Options

-a → Prints all the locations of each command.

To learn more read, The “which” Command in Linux [3 Practical Examples]

73. uptime

The **uptime** command in **Linux** provides helpful information about the amount of time the system has been running, the number of users currently logged in, and the system load averages for a specified period of time.

Syntax

uptime [OPTION]...

Useful Options

- **-p, --pretty** → Shows uptime information in a human-readable format.
- **-h, --help** → Displays this help page
- **-s, --since** → Prints the date and time since the system has been up, in **yyyy-mm-dd HH:MM:SS** format.

To learn more read, The “uptime” Command in Linux [5 Practical Examples]

74. file

The **file** command in **Linux** tests each file passed as an argument to determine its classification. During the process, the system runs three types of tests and the succeeding test determines the file type.
The descriptions of these tests are given below.

I. **filesystem test:** Checks whether the file is empty or if it is of a special type. The test examines the return from a **stat(2)** system call.
II. **magic test:** Checks for files with fixed data formats. It uses the concept of "magic number" that indicates extensions of data files.
III. **language test:** Determines in what language the file is written. It looks for a particular string that can appear anywhere in the first few blocks of the file.

Syntax

file [OPTION]... FILE …

Useful Options

- **-b** → Does not print filenames with output.
- **-c** → Prints out the parsed form of the file.
- **-d** → Shows debugging information of a file.
- **-f** → Reads file names from a name file.
- **-i** → Gives mime-type strings as output.
- **-s** → Determines the type of special files.
- **-z** → Determines the type of compressed files.

To learn more read, The "file" Command in Linux [9+ Practical Examples]

75. finger

In **Linux,** the major objective of the **finger** command is to display the **users' login information** on a system**.** This command helps to display the **idle status** of the user. It can also be used to display the **user's plan** and **project.** The information provided by the **finger** command may vary depending on the **system configuration.**

Syntax

finger [OPTION]... [USERNAME]...

Useful Options

- **-p** → Shows information except for the user's plan or project.
- **-s** → Displays idle status with login information.
- **-m** → Prevents matching of user names.

To learn more read, The "finger" Command in Linux [6 Practical Examples]

76. users

The **user** command is a utility tool to find the name of all **users** who are currently **logged in** to the system.

Syntax

users [OPTION]... [FILE]

Useful Options

- **--help** → Displays the help section of the **users** command.
- **--version** → Displays the version information of the **users** command.

To learn more read, The "users" Command in Linux [4 Practical Examples]

77. groups

A **group** is a collection of users in **Linux** that helps to manage those multiple users with the same **security** and **privileges**. Moreover, One **user** can be part of multiple **groups**. The **groups** command in **Linux** allows users to see which groups they and other users belong to.

Syntax

groups [OPTION]... [USERNAME]...

Useful Options

- **--help** → Offers a brief description of the **groups command** and links to resources that are related.
- **--version** → Provides the **version** details of the **groups command.**

To learn more read, [The “groups” Command in Linux [6 Practical Examples]]

78. passwd

The **passwd** command in **Linux** is used for changing user **password**. The word **passwd** is used as a short form of Password.

Syntax

passwd [OPTIONS] [USER]

Useful Options

- **-a, --all** → Shows the status of all users.
- **-d, --delete** → Deletes user password. The user account becomes unprotected.
- **-h, --help** → Display help message and exit.
- **-e, --expire** → Immediately makes the user **password** expire.
- **-i, --interactive** → Make an account inactive after password expiration.

To learn more read, [The "passwd" Command in Linux [7 Practical Examples]]

79. useradd

The **useradd** command is a **Linux** utility that allows you to create **new user accounts** on the **system**.

Syntax

useradd [OPTION]... user_name

Useful Options

- **-c** → Specify a comment or description for the user account.
- **-d** → Sets the user's home directory.
- **-D** → Sets the user default value.
- **-e,--expiredate** → Sets the date on which the user account will be disabled.
- **-f,--inactive** → Sets password inactivity period of the new account.
- **-g,--gid** → Sets the user's primary group.
- **-G,--groups** → Sets the user's secondary groups.
- **-h,--help** → Displays the help message.
- **-m** → Creates the user's home directory if it does not already exist.
- **-M** → Does not create the user's home directory.

- **-p,--password** → Encrypted password of the new account.
- **-r,--system** → Creates a system account.
- **-s** → Sets the user's login shell.

To learn more read, The "useradd" Command in Linux [12 Practical Examples]

80. userdel

The **userdel** command in **Linux** is used for deleting user accounts on a system. It helps to manage user accounts and maintain the **security** of the system. The userdel command can help to remove an old or unused user account or to clean up your system by removing unnecessary accounts.

Syntax

userdel [OPTION]... USER_NAME

Useful Options

- **-f, --force** → Force removal of files, even if not owned by the user.
- **-h, --help** → Displays the help message.
- **-r, --remove** → Removes the home directory and all traces of the user from the system.
- **-R, --root CHROOT_DIR** → Allows you to specify a directory to chroot into before performing the user deletion, which can be useful for separate partition or filesystem cases.
- **-Z, --selinux-user** → Removes any **SELinux** user mapping for the user. **SELinux** is a security module that provides fine-grained security policies for Linux systems.

To learn more read, The "userdel" Command in Linux [4 Practical Examples]

81. usermod

The **usermod** command is used to modify the attributes of an existing **user**. This command allows the **root user** or **superuser** to modify the user name, user ID, groups, home directory, password, user shell, expiry date, and other user details of an existing **user**. Generally, the **usermod** command provides the opportunity to modify the files like **/etc/group, /etc/shadow, /etc/gshadow, /etc/login.def & /etc/passwd**.

Syntax

usermod [OPTION]... USER

Useful Options

- **-a, --append** → Adds the user to one or more supplementary groups. You can use it only with the option **-G.**
- **-b, --badnames** → Permits non-compliant names.
- **-c, --comment** → Adds comment field for the user account.
- **-d, --home** → Modifies the login directory for any existing user account.
- **-e, --expiredate** → Sets account expiry date. The date is specified in the format YYYY-MM-DD.
- **-f, --inactive** → Sets the number of days after a password will expire until the account is permanently disabled.

- **-g, --gid** → Changes the primary group for a User.
- **-G, --groups** → Adds supplementary groups.
- **-l, --login** → Changes the login name from user_name to new_user_name.
- **-L, --lock** → Locks the user’s password.
- **-m, --move-home** → Moves the contents of the home directory from the existing home directory to the new directory.
- **-o, --non-unique** → Allows to change the user ID to a non-unique value.
- **-p, --password** → Allows to specify the new unencrypted password.
- **-R, --root CHROOT_DIR** → Applies changes in the CHROOT_DIR directory and use the configuration files from the CHROOT_DIR directory.
- **-s, --shell** → Specifies the user's new login shell.
- **-u, --uid** → Specifies the new numerical value of the user's ID.
- **-U, --unlock** → Unlocks a user's password.

To learn more read, **The “usermod” Command in Linux [14+ Practical Examples]**

82. groupadd

The **groupadd** command in **Linux** allows **administrators** to create new user **groups** in order to manage **privileges** and **permissions** for multiple users more easily. This command helps to avoid the time-consuming task of setting **permissions** and **privileges** for each user individually, which can also be prone to **errors**.

Syntax

groupadd [OPTION]... GROUP_NAME

Useful Options

- **-f, --force** → Exits with success status if the specified group already exists, and turns off -g if specified **GID already** exists.
- **-g, --gid GID** → Specifies the numerical value of the **group's ID**, which must be unique unless **-o** is used and must be non-negative.
- **-K, --key KEY=VALUE** → Overrides **/etc/login.defs** defaults with specified **KEY** and **VALUE**, multiple **-K** options can be used.
- **-o, --non-unique** → Allows adding a group with a non-unique **GID.**
- **-p, --password PASSWORD** → Sets an encrypted password for the group, the default is to disable the password.
- **-r, --system** → Creates a system group.
- **-R, --root CHROOT_DIR** → Specifies a different root directory than the default root directory
- **-P, --prefix PREFIX_DIR** → Specifies a different prefix than the default prefix to use when creating a new group.

To learn more read, **The “groupadd” Command in Linux [7 Practical Examples]**

83. addgroup

The **addgroup** command in **Linux** is used to create a new **group** on the current **machine**. It allows you to customize the **group’s settings** and manage **privileges** and **permissions.** It is similar to the **groupadd** command but more interactive than that.

Syntax

addgroup [OPTION]... GROUP_NAME

Useful Options

- **--debug** → Enables debugging mode.
- **--force-badname** → Allows you to create a group with a name that does not match the system's naming conventions.
- **--gid GID** → Specifies the numerical value of the **group's ID**, which must be unique and non-negative.
- **-h, --help** → Displays help message.
- **--system** → Creates a system group**.**

To learn more read, [The "addgroup" Command in Linux [7 Practical Examples]]

84. groupmod

The **groupmod** command is used to modify the attributes of an existing **group**. This command allows the **root user** or **superuser** to modify the **group name**, **GID**, **password,** and other important information of an existing **group**. Moreover, it provides the opportunity to modify files like **/etc/group, /etc/gshadow, /etc/login.def & /etc/passwd**.

Syntax

groupmod [OPTION]... GROUP_NAME

Useful Options

- **-g, --gid** → Changes the GID of a given group.
- **-n, --new-name** → Changes the name of an existing group.
- **-p, --password** → Enables password modification.
- **-o, --non-unique** → Allows changing the group GID to a **non-unique** value.
- **-R, --root CHROOT_DIR** → Applies changes in the CHROOT_DIR directory and use the configuration files from the CHROOT_DIR directory.

To learn more read, [The "groupmod" Command in Linux [5+ Practical Examples]]

85. screen

The **screen** command in **Linux** is used to run multiple **sessions** in a single **terminal**. For instance, you are working on a **server** remotely and you have only one **terminal**. But you want to run multiple processes simultaneously. The **screen** command makes it possible. Moreover, it minimizes the chances of abruptly closing a task midway that might be sensitive or takes a lot of time to complete.

Syntax

screen [OPTION]... [COMMAND [ARGUMENT]]...

Useful Options

- **-S** → Creates a new **session**.
- **-p** → Assigns a **password** to a **session**.
- **-X** → Executes commands inside a **session**.
- **-r** → Reattaches to a **session**.
- **-ls** → Lists out all the running **sessions**.

To learn more read, [The “screen” Command in Linux [13 Practical Examples]]()

86. apt

The **apt** command in **Linux** stands for “**A**dvanced **P**ackage **T**ool”. It is a **command-line interface** for managing the packages on a user’s system. You can perform several actions including installation, update, and removal of packages with this command. You must use the **sudo** command along with **apt** to get **root** permissions while managing packages on your machine.

Syntax

apt command pkg…

Useful Options

- **-d, --download-only** → Downloads a package but does not install.
- **--only-upgrade** → Upgrades a specific package.
- **--installed** → Returns only installed packages.
- **--upgradeable** → Returns only upgradeable packages.
- **-y** → Answers “yes” to prompt without interruption.

To learn more read, [The “apt” Command in Linux [13+ Practical Examples]]()

87. apt-get

The **Advanced Package Utility** (**APT**) library may be accessed via the command line tool **apt-get** (a package management system for **Linux** distributions), as it serves as the user's "back end" for other programs that utilize the **APT library**. Anyway, you may use the **apt-get** command in **Linux** to **find, install, manage, update, and uninstall** applications.

Syntax

apt-get [OPTION] [COMMAND]

Useful Options

- **--auto-remove** or **--autoremove** → When using apt-get with the install or remove command, this option acts like running the autoremove command.
- **-d** or **–download-only** → Specifies that **apt-get** should only retrieve the packages, and not unpack or install them.
- **-f** or **–fix-broken** → Specifies that **apt-get** should attempt to correct the system with broken dependencies in place.
- **--no-install-recommends** → Not to consider recommended packages as a dependency to install.

To learn more read, [The “apt-get” Command in Linux [10 Practical Examples]]()

88. getent

The **getent** command in **Linux** is a tool that enables the retrieval of data from **system databases** such as **passwd**, **group**, and **services**. It can be utilized to either list all entries in a **database** or obtain information about a **specific entry**.

Syntax

getent [OPTION]... database key...

Useful Options

- **-s, --service**: Specifies the service.

- **-i, --no-idn**: Disables IDN encoding in lookups.
- **-?, --help**: Show user help page.
- **--usage**: Display usage information.

To learn more read, [The “getent” Command in Linux [11 Practical Examples]]()

89. source

The **source** command in **Linux** executes commands from the file passed as an argument to it in the current shell environment. This command is beneficial for sourcing a script that sets variables, functions, or aliases in the current shell session.

Syntax

source filename [argument]

Useful Options

The **source** command in **Linux** is a **shell built-in** command. There is no option available for this command. To know more about this command, you can check its **help page** by executing the below command.

help source

To learn more read, [The “source” Command in Linux [3 Practical Examples]]()

90. service

The service command in **Linux** is mainly used for **starting**, **stopping**, and **restarting** services on our **operating system**. This command is very **versatile** in controlling our **system services**.

Syntax

service [OPTIONS] [Service_Name] [COMMAND]

Here, the **Service_Name** means the process running in the operating system, and **COMMAND** is the specific task you want to do.

Useful Options

- **--status-all** → Prints all available services on the terminal
- **--help/-h** → Prints the help section of the service command.
- **--version** → Shows the **version** of the service command.

To learn more read, [The “service” Command in Linux [6 Practical Examples]]()

91. jobs

The **jobs** command is used to display a set of jobs that are **currently running** in the background & the foreground in **Linux**. If the command prompt does not show any information that means no jobs are running at that time. By using the **jobs** command, you can see the **process ID, job number & status** of each Job.

Syntax

jobs [OPTION]

Useful Options

- **-l** → Displays the process ID, job number, and status of each job
- **-n** → Displays only jobs that have changed status since the last notice.

- **-p** → Displays only the process ID of each job.
- **-s** → Displays the jobs that are stopped.
- **-r** → Displays the jobs that are running.
- **-x** → Executes command in the background.

To learn more read, [The “jobs” Command in Linux [6 Practical Examples]]

92. htop

The **htop** command when used shows a list of all the running processes throughout the system. The **htop** command in **Linux** does the same function as the top command. But the difference is **htop** is more modern than the **top**. The **htop** command is more interactive than the **top** command.

Syntax

htop [-dCFhpustvH]

N.B: The traditional switches `-' and whitespace are optional. Anything bound by the square brackets is optional

Useful Options

- **-u, --user** → Only shows processes for a specific user.
- **-M, --no-mouse:** → Disables the interactive mouse operation.
- **--tree** → Shows system processes as a tree view.

To learn more read, The "htop" Command in Linux [7 Practical Examples]

93. at

The **at** command in **Linux** is a Command line utility tool used for scheduling one-time jobs. Users can specify a time for executing certain tasks. The command can also **list or delete jobs** that are saved for later execution. The **atq**, **atrm**, and **batch** commands are also part of the **at** command that enables it to queue, examine or remove scheduled jobs.

1. **at:** Execute commands at the specified time.
2. **atq:** List current users’ pending jobs.
3. **atrm:** Remove pending job specified by job ID.
4. **batch:** Runs commands when system load levels permit. By default, if the load average is below **1.5**. And by **1.5** means, the system is 100% busy running 1 process while there is another process in the queue waiting to use 50% of the system.

Syntax

at [OPTION]... runtime

Useful Options

- **-b** → Runs commands when system load levels permit, an alias for **batch**.
- **-c** → Displays specified job context on the terminal.
- **-f** → Reads jobs from a file.
- **-r/-d** → Removes pending job specified by job ID, an alias for **atrm**.
- **-l** → Lists current users’ pending jobs, an alias for **atq**.
- **-m** → Mails user upon completion of the job.
- **-M** → Does not mail the user upon completion of the job.

To learn more read, The “at” Command in Linux [7 Practical Examples]

94. cron

The cron command in **Linux** allows users to schedule commands to be repeated at a certain period of time. It recurrently matches the assigned time to the current time field. When there is a match the system executes given commands. **cron** is automatically started from **/etc/init.d** on entering multi-user run levels.

Syntax

cron [-f] [-l] [-L loglevel]

Note: In the above syntax the alphabets prefixed with "-" within the square brackets indicate **OPTIONs** for **cron**.

Useful Options

- **-f** → Process stays in the foreground, and does not daemonize.
- **-l** → Enables LSB-compliant names for /etc/cron.d files.
- **-n** → Includes the FQDN in the subject when sending emails.
- **-L loglevel** →

1 (Strats log of all cron jobs)
2 (Ends log of all cron jobs)
4 (Logs every failed job. Exit status is not equal to zero)
8 (Logs the process number of all cron jobs)

To learn more read, The "cron" Command in Linux [3 Practical Examples]

95. crontab

The **crontab** command in **Linux** is used for scheduling tasks at regular intervals. You can add your desired commands/scripts to **crontab** files by creating an editing session. The listed jobs in the **crontab** files are run by the **cron** daemon at predefined times.

Syntax

crontab [-u user] file
crontab [-u user] [Option]

Useful Options

- **-u** → Specify the user name whose crontab file is accessed.
- **-e** → Edit current crontab using the specified editor.
- **-l** → Display/List current crontab on the standard output.
- **-r** → Remove current crontab.
- **-i** → Ask before removing crontab using **-r**.

To learn more read, The "crontab" Command in Linux [10 Practical Examples]

96. uniq

The **uniq** command helps to detect any duplicate entity inside a text file to avoid redundancy.

Syntax

uniq [OPTION]... [INPUT [OUTPUT]]

Useful Options

- **-c, --count** → Sets a prefix at the start of the lines by the number of occurrences.
- **-d, --repeated** → Prints the duplicate lines, one for each group.
- **-D** → It prints all the duplicate lines.
- **-f, --skip-fields=N** → Avoids comparing the first N fields.

- **-i, --ignore-case** → Ignores the differences in the case when comparing.
- **-u, --unique** → Prints the unique lines.

To learn more read, [The “uniq” Command in Linux [6 Practical Examples]]

97. dig

The **dig** command in **Linux** is a network administration **command-line tool** that stands for **D**omain **I**nformation **G**roper. It is used to gather **DNS** (**D**omain **N**ame **S**ystem) information. The command performs tasks related to **DNS** lookups to query **DNS** name servers. Mainly it is used by network administrators to verify and troubleshoot **DNS** problems.

Syntax

dig [server] [name] [type]

Here,

1. **Server** ⟶ The name or IP address of the name server whom we want to query. If you don't specify a server, the **dig** command will use your machine’s pre-configured **DNS**.
2. **Name** ⟶ It is the name of the resource record that is to be looked up using the **DNS** name server.
3. **Type** ⟶ The type of query or **DNS** record you are looking for. The **type** argument must be valid. If no type argument is applied, the **dig** command performs a lookup on the **A** record by default.

Some Common DNS record types:

1. **A** ⟶ **Address** records are one of the basic and most commonly used records, which are used to directly map a hostname to an IP Address. They translate domain names and store them as IP addresses. Address records can only hold IPv4 addresses.
2. **MIX**⟶ Stands for **Mail Exchange**. Which maps message transfer agents for the domain. And it stores instructions to direct emails to mail servers.
3. **SIG**⟶ SIG for the **SIGnature** record, used for **encryption** protocol.
4. **TXT** ⟶**TeXT** records are used to store definitive text.
5. **CNAME** ⟶ (**Canonical NAME)** record type is used instead of an **A** (Adress) record if a domain is an alias for another domain.
6. **SOA** ⟶(**Start Of Authority**) record type holds important information about a domain or zone.
   NS ⟶ (**Name Server**) record contains the authoritative name servers.

Useful Options

- **-4** → Used for IPv4 only.
- **-6** → Used for IPv6 only.
- **-b address** → Sets the source IP address of the query.
- **-c class** → Sets the query class.
- **-f file** → Used for Batch mode. The dig command gets lookup requests from the specified files and processes each line systematically as they are organized in the file.
- **-i** → Used for reverse IPv6 lookups.
- **-m** → Used to enable memory usage debugging.
- **-u** → DIsplays the query time in microseconds.
- **-v** → Displays the version information.
- **-x** → Reverses lookups for mapping the addresses to names.

- **-t type** → Specifies the resource record type to query.

To learn more read, The " dig" Command in Linux [10 Practical Examples]

98. nslookup

The **nslookup** is a **network-administrator** command tool that stands for **N**ame **S**erver **Lookup**. It performs queries on **DNS** (**D**omain **N**ame **S**ystem) to obtain domain names, **IP** address mapping, or any other specific **DNS** record. We mainly use the **nslookup** command to troubleshoot **DNS-related** problems.

Syntax

The syntax for **interactive** mode is,

nslookup [OPTION]... [Name | -] [Server]

And the syntax for **non-interactive** mode is,

nslookup [OPTION]... <Domain_Name>

Here,

- **OPTION**→ **OPTION**s are used to modify the **nslookup** command.
- **Domain Name** → The name or **IP** address of the name server whom we want to query. If you don't specify a server, the **nslookup** command will use your machine's pre-configured **DNS**.

Useful Options

Interactive Options:

- **host** [server] → Looks up information for the host using the default or specified server.
- **server** domain → Looks up information about the domain using the default server.
- **lserver** domain → Looks up information about the domain using the initial server.
- **set** keyword [=value], → Changes state information that affects the lookups (**=all**, to display the current values of the frequently used options to set)
- **domain**=name → Sets the search list to name.
- **class** =value → Changes the query class to one of these values, such as (**IN** (**IN**ternet), **CH**(**CH**aos), **HS**(**H**e**S**iod), **ANY**(wildcard)). The default is **IN**.
- **port**=value → Changes the default **TCP/UDP** name server port to value, default is **53**.
- **type**=value → (**a, any, cname, gid, hinfo, mb, mg, minfo, mr, mx, ns, ptr, soa, txt**)
- **retry**=number → Sets the number of retries to a number.
- **timeout**=number → Sets the initial timeout interval for waiting for a reply, in seconds.
- **[no]debug,** to turn on or off the display of the full and intermediate response packets while searching. The default in **nobug**.
- **[no]d2** → Turns on or off the debugging mode.
- **[no]recurse** → Tells the name server to query other servers if it does not have the information.
- **[no]fail** → Tries the next nameserver if a name server responds with **SERVFAIL**. The default is nofail.
- **exit,** to exit the program.

Non-Interactive Options

- **-domain=[domain-name]** → Changes the default DNS name.
- **-debug,** show debugging info.
- **-port=[port-number],** → Changes the standard **port** number.

- **-timeout=[seconds]** → Specifies the time for the server to respond.
- **-type=a** → Displays all the info about **DNS A** records.
- **-type=any** → Displays info about all types of **DNS** records.
- **-type=hinfo** → Prints hardware-related information about the host.
- **-tpe=ns** → Displays **Name Server** records.
- **-type=mx** → Displays all the info about **DNS Mail Exchange** records.
- **-type=ptr** → Displays **Pointer Records**.
- **-type=soa** → Displays **Start Of Authority** records.

Some Common DNS record types:

- **A** ⟶ **Address** records are one of the basic and most commonly used records used to map a hostname to an IP Address directly. They translate domain names and store them as **IP** addresses. Address records can only hold **IPv4** addresses.
- **MX**⟶ Stands for **Mail Exchange**. Which maps message transfer agents for the domain. And it stores instructions to direct emails to mail servers.
- **SIG**⟶ SIG for the **SIGnature** record, used for **encryption** protocol.
- **TXT** ⟶**TeXT** records are used to store definitive text.
- **CNAME** ⟶ (**Canonical NAME)** record type is used instead of an **A** (Adress) record if a domain is an alias for another domain.
- **SOA** ⟶(**Start Of Authority**) record type holds important information about a domain or zone.
- **NS** ⟶ (**Name Server**) record contains the authoritative name servers.

To learn more read, The "nslookup" Command in Linux [12 Practical Examples]

99. netstat

The **netstate** command in **Linux** is a **network display command tool** that stands for **Net**work **Stat**istics. It displays all the network-associated information such as the list of all the network connections on the system, routing tables, interface statistics, multicast memberships, masquerade connections, etc. The command can also display the current state of a network interface, like its **IP** address, **netmask**, and **status**.

Syntax

netstat [OPTION]...

Useful Options

- **-a**, **--all** → Displays all active connections.
- **-at** → Displays all active **TCP** connections.
- **-au** → Displays all active **UDP** connections.
- **-l**, **--listening** → Displays only listening ports.
- **-M**, **--masquerade** → Displays all masquerade connections.
- **-r**, **--route** → Prints the kernel routing tables.
- **-g**, **--groups** → Displays multicast group membership information.
- **-i**, **--interfaces** → Displays all network interfaces.
- **-ie** → Displays the statistics for a specific network interface.
- **-s**, **--statistics** → Displays the summary statistics of each protocol.
- **-st** → Displays the statistics of **TCP** protocol.
- **-su** → Displays the statistics of **UDP** protocol.
- **-c**, **--continuous** → Prints **netstat** information continuously.

- **-n**, **--numeric** → Displays numeric addresses defining symbolic hosts, ports, or usernames.
- **-v**, **--verbose** → Displays the detailed output.
- **-A**, **--protocol=family** → Specifies the address families for which connections are to be displayed.
- **-e**, **--extend** → Displays extended output.
- **-o**, **--timers** → Includes networking timers-related information.
- **-p**, **--program** → Displays the **PID** and name of the process of the corresponding sockets.
- **-C** → Displays the routing information from the route cache.

To learn more read, The "netstat" Command in Linux [22 Practical Examples]

100. neofetch

The **neofetch** command is an easy and super fast command line tool to fetch system information such as the **hostname**, **operating system**, **kernel version**, and **desktop environment** in an artistic and visually charming way within seconds.

Syntax

neofetch <func_name> [OPTION]... "value"

Here,

1. **func_name** → It specifies a function name to quickly display that function's information. Which is the second part of the information on the display configuration.
2. **Options** → Options are used to modify and increase the usage of the command in a user-friendly way.
3. **Value** → These are the values that we assign to our options to modify the command**.**

Useful Options

Info

- **--disable infoname** → Disables an info line from appearing in the output. Where infoname is the function name that can be memory, disk, hostname, shell, or any info you want to disable.
- **--speed_type type** → Changes the type of CPU speed to display, these types could be current, min, max, bios, etc.
- **--cpu_brand on/off** → Enables or disables CPU brand in the display.
- **--cpu_speed on/off** → Shows/hides the CPU speed.
- **--cpu_temp C/F/off** → Shows/hides CPU temperature.
- **--cpu_brand on/off** → Shows/hides CPU bran.

Text formatting

- **--colors x x x x x x** → Changes the text colors in this order.
- **--underline on/off** → Enables/disables the underline.
- **--bold on/off** → Enables/disables bold text.

Color Blocks

- **--color_blocks on/off** → Enables/disables the color blocks.
- **--col_offset auto/num** → For eft-padding of color blocks.
- **--block_width num** → Sets the width of color blocks in spaces.
- **--block_height num** → Sets the height of color blocks in lines.

BARS

- **--bar_char 'elapsed char' 'total char'** → Characters to use when drawing bars.
- **--bar_border on/off** → Surrounds the bar with [ ] or not.

To learn more read, **The “neofetch” Command in Linux [12+ Practical Examples]**

Conclusion

Exhale! You have reached the end of this long list of commands. After completing this huge article, you will cover almost 90% of the commands you will need to master for the **CLI** of **Linux**. Hope this article helps! Happy exploring and command-line empowerment!

Copyright ©2023 linuxsimply.com | All rights reserved