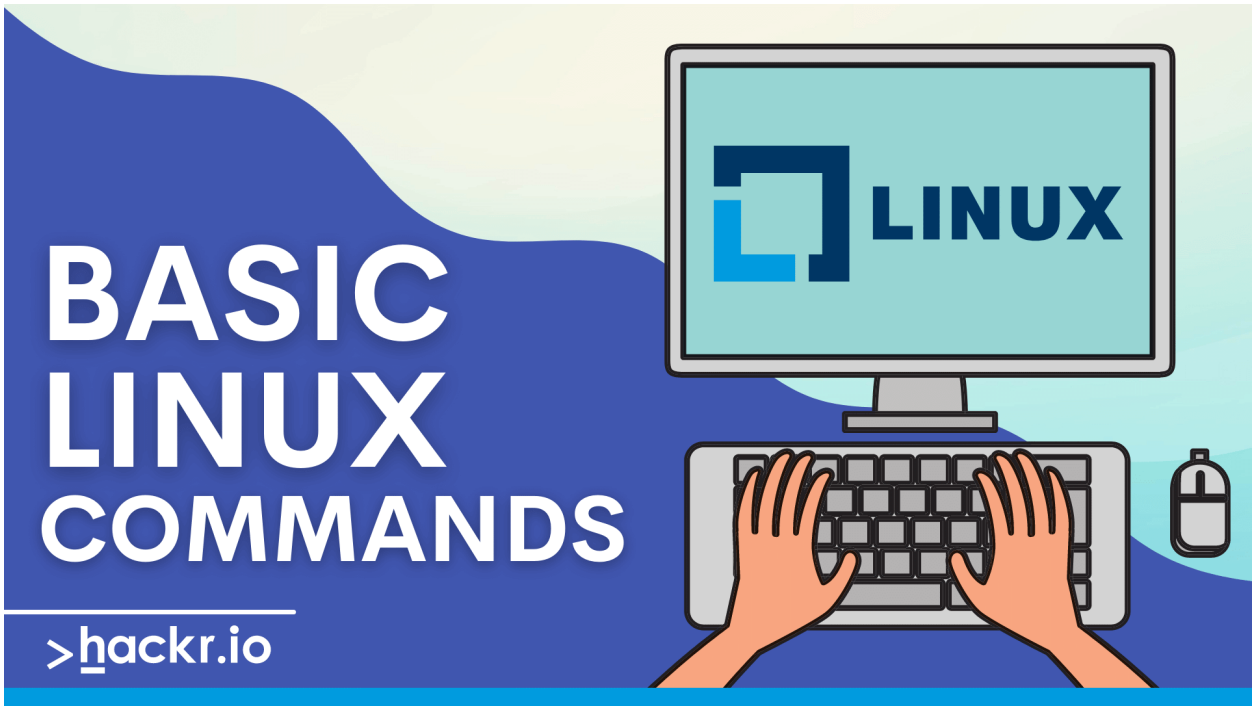

Basic Linux Commands

We've listed 25 of the most common Linux commands here. They're not all you need to know, but they are some of the most common. And remember, **Linux commands are case-sensitive**.

1. ls

This command lists directory contents. If you've used the Windows command prompt, then you should know that the command dir is used to list the contents in a directory. This is what the *ls* command does in Linux - it lists files and directories. Some versions may support color-coding. The names in blue represent the names of directories.

The command *ls -l | more* – helps paginate the output so you can view page by page. Otherwise the listing scrolls down rapidly. You can always use ctrl + c to return to the command line.

```
$ ls -l filename
```

2. cd /var/log

This changes the current directory. Note that it uses a forward slash. The example used here changes the location to a Linux directory that is present in all versions of Linux.

When you use *ls –l* you will be able to see more details of the contents in the directory. It lists the following:

- Permissions associated with the file
- The owner of the file
- The group associated with the file
- The size of the file
- The timestamp
- The name of the file

```
$ cd /var/log
```

3. grep

This finds text in a file. The *grep* command searches through many files at a time to find a piece of text you are looking for.

grep PATTERN [FILE]

grep failed transaction.log

The above command will find all of the words in the files that matched the word ‘failed’.

```
$ grep ‘failed’ transaction.log
```

4. su / sudo command

There are some commands that need elevated rights to run on a Linux system. You must run these as a System Administrator.

The *su* command changes the shell so that it is used as a super user. Until you use the exit command, you can continue to be the super user.

The *sudo* command is used when you just need to run something as a super user, you can use the *sudo* command. This will allow you to run the command in elevated rights

and once the command is executed you will be back to your normal rights and permissions.

An example is the shutdown command, which turns off the computer system.

- ***sudo shutdown 2*:** shutdown and turns of the computer after 2 minutes.
- ***sudo shutdown –r 2*:** shuts down and reboots in 2 minutes.
- **Using *ctrl C* or *shutdown –c:*** helps in stopping the shutdown process.

```
$ sudo shutdown 2
$ sudo shutdown -r 2
```

5. pwd

One way to identify the directory you are working in is the *pwd* command. It displays the current working directory path and is useful when directory changes are made frequently.

```
$ pwd
```

6. passwd

Though it looks similar to the *pwd* command, this command is very different. This command is used to change the user account password.

You could change your password or the password of other users. Note that the normal system users may only change their own password, while *root* may modify the password for any account.

passwd [username] - changes the password for the user.

```
$ passwd admin
```

7. mv

The mv command moves a file or renames it. Here the file name gets changed from *first.txt* to *second.txt.*

```
$ mv first.txt second.txt
```

Type *ls* to view the change.

8. cp

This command copies a file. The *cp* command issues a copy of the file *second.txt* in the same directory.

```
$ cp second.txt third.txt
```

You can use *ls – l* to see the new file created.

9. rm

This command is used to remove files in a directory or the directory itself. A directory cannot be removed if it is not empty.

rm [name of the file]

rm –r removes all the contents in a directory and the directory as well.

```
$ rm file1
$ rm -r myproject
```

10. mkdir

The *mkdir* command makes a directory. The command is written as follows: *mkdir [directory name]*

```
$ mkdir myproject
```

11. chmod

This command changes the mode of a file system object. Files can have read, write, and execute permissions.

For example:

- *chmod mode FILE*

- *chmod 744 script.sh*
- The first number stands for the user who is associated with the file
- The second number is for the group associated with the file
- The third number is associated with everyone else who is not a part of the user or group

```
$ chmod 744 script.sh
```

12. chown

This command is used to change the ownership of a file/folder or even multiple files/folders for a specified user/group.

chown owner_name file_name

```
$ chown user1 script.sh
```

Assume that if you are a user named *user1* and you want to change ownership to *root* use “sudo”.

```
$ sudo chown root script.sh
```

13. cat

The *cat* command (short for “concatenate”) is one of the most frequently used commands in Linux. *cat* command allows you to create single or multiple files, view contents of files, concatenate files (combining files), and redirect output in terminal or files.

```
$ cat file.txt
$ cat file1.txt file2.txt
```

The output will be the entire contents of the file(s).

14. echo

This command is used to display a text or a string to the standard output or a file.

$ echo “This is an article on basic linux commands”

The output would be “This is an article on basic linux commands”, without the quotes.

The *echo –e* option acts as an interpretation of escape characters that are back-slashed. \n the newline character is interpreted by the echo –e command.

```
$ echo –e “This is an article is for beginners. \nIt is on basic linux commands
```

The output of the command above would be:

```
This is an article is for beginners.
It is on basic linux commands
```

15. wc

The *wc* (word count) command is used to find out the number of new lines, word count, byte, and characters count in a file specified by the file arguments.

wc [options] filenames.

```
$ wc –l readme.txt
```

Shows the output as - 120 readme.txt

- **wc -l** : Prints the number of lines in a file.
- **wc -w** : Prints the number of words in a file.
- **wc -c** : Displays the count of bytes in a file.
- **wc -m** : Prints the count of characters from a file.
- **wc -L** : Prints only the length of the longest line in a file.

16. man

This command is used to view the online reference manual pages for commands/programs.

```
$ man grep
```

```
$ man mkdir
```

17. history

This command is used to show previously used commands or obtain information about the commands executed by a user.

```
$ history
```

18. clear

This command clears the terminal screen.

```
$ clear
```

19. apt –get

apt -get is a powerful and free front-end package manager for Debian/Ubuntu systems. It is used to install new software packages, remove available software packages, upgrade existing software packages as well as upgrade the entire operating system. apt – stands for advanced packaging tool.

```
$ sudo apt-get update
```

20. reboot

This command halts, powers off, or reboots a system.

```
$ reboot
```

21. locate

The *locate* command is used to find a file and runs in the background, unlike the *find* command.

```
$ locate file1.txt
```

22. diff

The *diff* command compares two files line by line to find differences. The output will be the lines that are different.

```
$ diff file1.txt file2.txt
```

23. useradd

The *useradd* command creates a new user. The username is added after the useradd command, as follows:

```
$ useradd John
```

24. exit

The *exit* command exits the current shell. When you hit enter, you'll be taken out of the terminal.

```
$ exit
```

25. kill

The *kill* command is used to end a process, usually an unresponsive one. The *kill* command also includes the process ID or the program name, as shown here:

```
$ kill 522551
```