1. pwd command

Use the **pwd** command to find out the path of the current working directory (folder) you're in. The command will return an absolute (full) path, which is basically a path of all the directories that starts with a forward slash **(/)**. An example of an absolute path is **/home/username**.

2. cd command

To navigate through the Linux files and directories, use the **cd** command. It requires either the full path or the name of the directory, depending on the current working directory that you're in.

Let's say you're in **/home/username/Documents** and you want to go to **Photos**, a subdirectory of **Documents**. To do so, simply type the following command: **cd Photos**.

Another scenario is if you want to switch to a completely new directory, for example,**/home/username/Movies**. In this case, you have to type **cd** followed by the directory's absolute path: **cd /home/username/Movies**.

There are some shortcuts to help you navigate quickly:

- **cd ..** (with two dots) to move one directory up
- **cd** to go straight to the home folder
- **cd-** (with a hyphen) to move to your previous directory

3. ls command

The **ls** command is used to view the contents of a directory. By default, this command will display the contents of your current working directory.

If you want to see the content of other directories, type **ls** and then the directory's path. For example, enter **ls /home/username/Documents** to view the content of **Documents**.

There are variations you can use with the **ls** command:

- **ls -R** will list all the files in the sub-directories as well
- **ls -a** will show the hidden files
- **ls -al** will list the files and directories with detailed information like the permissions, size, owner, etc.

4. cat command

cat (short for concatenate) is one of the most frequently used commands in Linux. It is used to list the contents of a file on the standard output (sdout). To run this command, type **cat** followed by the file's name and its extension. For instance: **cat file.txt**.

Here are other ways to use the **cat** command:

- **cat > filename** creates a new file

- **cat filename1 filename2>filename3** joins two files (1 and 2) and stores the output of them in a new file (3)
- to convert a file to upper or lower case use, **cat filename | tr a-z A-Z >output.txt**

5. cp command

Use the **cp** command to copy files from the current directory to a different directory. For instance, the command **cp scenery.jpg /home/username/Pictures** would create a copy of **scenery.jpg** (from your current directory) into the **Pictures** directory.

6. mv command

The primary use of the **mv** command is to move files, although it can also be used to rename files.

The arguments in mv are similar to the cp command. You need to type **mv**, the file's name, and the destination's directory. For example: **mv file.txt /home/username/Documents**.

To rename files, the Linux command is **mv oldname.ext newname.ext**

7. mkdir command

Use **mkdir** command to make a new directory — if you type **mkdir Music** it will create a directory called **Music**.

There are extra **mkdir** commands as well:

- To generate a new directory inside another directory, use this Linux basic command **mkdir Music/Newfile**
- use the **p** (parents) option to create a directory in between two existing directories. For example, **mkdir -p Music/2022/Newfile** will create the new "2022" file.

8. rmdir command

If you need to delete a directory, use the **rmdir** command. However, rmdir only allows you to delete empty directories.

9. rm command

The **rm** command is used to delete directories and the contents within them. If you only want to delete the directory — as an alternative to rmdir — use **rm -r**.

Note: Be very careful with this command and double-check which directory you are in. This will delete everything and there is no undo.

10. touch command

The **touch** command allows you to create a blank new file through the Linux command line. As an example, enter touch **/home/username/Documents/Web.html** to create an HTML file entitled **Web** under the **Documents** directory.

11. locate command

You can use this command to **locate** a file, just like the search command in Windows. What's more, using the **-i** argument along with this command will make it case-insensitive, so you can search for a file even if you don't remember its exact name.

To search for a file that contains two or more words, use an asterisk **(*)**. For example, **locate -i school*note** command will search for any file that contains the word "school" and "note", whether it is uppercase or lowercase.

12. find command

Similar to the **locate** command, using **find** also searches for files and directories. The difference is, you use the **find** command to locate files within a given directory.

As an example, find **/home/ -name notes.txt** command will search for a file called **notes.txt** within the home directory and its subdirectories.

Other variations when using the **find** are:

- To find files in the current directory use, **find . -name notes.txt**
- To look for directories use, **/ -type d -name notes. txt**

13. grep command

Another basic Linux command that is undoubtedly helpful for everyday use is **grep**. It lets you search through all the text in a given file.

To illustrate, **grep blue notepad.txt** will search for the word blue in the notepad file. Lines that contain the searched word will be displayed fully.

14. sudo command

Short for "**SuperUser Do**", this command enables you to perform tasks that require administrative or root permissions. However, it is not advisable to use this command for daily use because it might be easy for an error to occur if you did something wrong.

15. df command

Use **df** command to get a report on the system's disk space usage, shown in percentage and KBs. If you want to see the report in megabytes, type **df -m**.

16. du command

If you want to check how much space a file or a directory takes, the **du** (Disk Usage) command is the answer. However, the disk usage summary will show disk block numbers instead of the usual size format. If you want to see it in bytes, kilobytes, and megabytes, add the **-h** argument to the command line.

17. head command

The **head** command is used to view the first lines of any text file. By default, it will show the first ten lines, but you can change this number to your liking. For example, if you only want to show the first five lines, type **head -n 5 filename.ext**.

18. tail command

This one has a similar function to the head command, but instead of showing the first lines, the **tail** command will display the last ten lines of a text file. For example, **tail -n filename.ext.**

19. diff command

Short for difference, the **diff** command compares the contents of two files line by line. After analyzing the files, it will output the lines that do not match. Programmers often use this command when they need to make program alterations instead of rewriting the entire source code.

The simplest form of this command is **diff file1.ext file2.ext**

20. tar command

The **tar** command is the most used command to archive multiple files into a **tarball** — a common Linux file format that is similar to zip format, with compression being optional.

This command is quite complex with a long list of functions such as adding new files into an existing archive, listing the content of an archive, extracting the content from an archive, and many more. Check out some **practical examples** to know more about other functions.

21. chmod command

chmod is another Linux command, used to change the read, write, and execute permissions of files and directories. As this command is rather complicated, you can read **the full tutorial** in order to execute it properly.

22. chown command

In Linux, all files are owned by a specific user. The **chown** command enables you to change or transfer the ownership of a file to the specified username. For instance, **chown linuxuser2 file.ext** will make **linuxuser2** as the owner of the **file.ext**.

23. jobs command

jobs command will display all current jobs along with their statuses. A job is basically a process that is started by the shell.

24. kill command

If you have an unresponsive program, you can terminate it manually by using the **kill** command. It will send a certain signal to the misbehaving app and instructs the app to terminate itself.

There is a total of sixty-four signals that you can use, but people usually only use two signals:

- **SIGTERM (15)** — requests a program to stop running and gives it some time to save all of its progress. If you don't specify the signal when entering the kill command, this signal will be used.
- **SIGKILL (9)** — forces programs to stop immediately. Unsaved progress will be lost.

Besides knowing the signals, you also need to know the process identification number (PID) of the program you want to **kill**. If you don't know the PID, simply run the command **ps ux**.

After knowing what signal you want to use and the PID of the program, enter the following syntax:

kill [signal option] PID.

25. ping command

Use the **ping** command to check your connectivity status to a server. For example, by simply entering **ping google.com**, the command will check whether you're able to connect to Google and also measure the response time.

26. wget command

The Linux command line is super useful — you can even download files from the internet with the help of the **wget** command. To do so, simply type **wget** followed by the download link.

27. uname command

The **uname** command, short for Unix Name, will print detailed information about your Linux system like the machine name, operating system, kernel, and so on.

28. top command

As a terminal equivalent to Task Manager in Windows, the **top** command will display a list of running processes and how much CPU each process uses. It's very useful to monitor system resource usage, especially knowing which process needs to be terminated because it consumes too many resources.

29. history command

When you've been using Linux for a certain period of time, you'll quickly notice that you can run hundreds of commands every day. As such, running **history** command is particularly useful if you want to review the commands you've entered before.

30. man command

Confused about the function of certain Linux commands? Don't worry, you can easily learn how to use them right from Linux's shell by using the **man** command. For instance, entering **man tail** will show the manual instruction of the tail command.

31. echo command

This command is used to move some data into a file. For example, if you want to add the text, “Hello, my name is John” into a file called name.txt, you would type **echo Hello, my name is John >> name.txt**

32. zip, unzip command

Use the **zip** command to compress your files into a zip archive, and use the **unzip** command to extract the zipped files from a zip archive.

33. hostname command

If you want to know the name of your host/network simply type **hostname**. Adding a **-I** to the end will display the IP address of your network.

34. useradd, userdel command

Since Linux is a multi-user system, this means more than one person can interact with the same system at the same time. **useradd** is used to create a new user, while **passwd** is adding a password to that user’s account. To add a new person named John type, **useradd John** and then to add his password type, **passwd 123456789.**

To remove a user is very similar to adding a new user. To delete the users account type, **userdel UserName**