**Linux Commands**

🡪Linux is an Operating system, like MacOS or Windows.

🡪It is also the most popular Open source and free.

**Instance:**

🡪An instance in AWS refers to s virtual server running in Amazon EC2(Elastic Compute Cloud).

🡪It acts like a remote computer where you can run applications, host websites, or perform computing tasks.

**Linux Commands:**

1. **ls**

🡪List files in a directory.

1. **pwd**

🡪Shows current directory

1. **cd <dir>**

🡪Change directory

1. **mkdir <dir>**

🡪Create a new directory

1. **rm <file>**

🡪Delete a file

1. **touch filename.txt**

🡪Creates a empty file

1. **echo ”Hello AWS” > file.txt**

🡪Creates a file with content

1. **cat file.text**

🡪To read the information from the file.

1. **rm -r <dir>**

🡪Delete a directory

1. **cp <src> <dest>**

🡪Copy files or directories

1. **mv <src> <dest>**

🡪Move or rename files

1. **less <file>**

🡪View file content (scrollable) one page at a time without opening an editor.

1. **nano <file>**

🡪Edit file using Nano editor

1. **vim <file>**

🡪Edit file using Vim editor

1. **ps aux**

🡪List running process

1. **uname -a**

🡪Show system details

1. **df -h**

🡪Check disk usage

1. **free -m**

🡪Show memory usage

1. **cd ..**

🡪Back to the home folder.

1. **cd ../<dir>**

🡪The cd ../<dir> command is used to move **up one directory** (..) and then enter a specific subdirectory (<dir>).

1. **mkdir -p fruits/apples**

🡪To create nested folders.

1. **mv <curr\_file> <new\_file\_name>**

🡪It rename the name of the file

1. **touch pear**

**touch apple**

**mkdir fruits**

**mv pear apple fruits**

🡪To move a file which is present in the current directory to the new directory.

1. **cp <file\_name> <new\_file\_name>**

🡪To create a copy of the file

1. **cp -r <folder\_name> <new\_folder\_name>**

🡪To create a copy of the folder

1. **Find:**

🡪The find command can be used to find the files or folders matching a particular search pattern

**find . -name '\*.js'**

🡪Find all the files under the current tree that have the .js extension and print the relative path of each file matching.

🡪It's important to use quotes around special characters like \* to avoid the shell interpreting them.

**find . -type d -name src**

🡪Find directories under the current tree matching the name "src"

**find folder1 folder2 -name filename.txt**

🡪It searched for files or directories inside 2 folders and looks for a given name.

**find . -type d -name node\_modules -o -type d -name public**

🡪Looks for directories only in the current directory

**find . -type f -name "\*.txt" -not -path "\*/node\_modules/\*"**

🡪type f – for files

🡪type d – for directories

🡪Looks for the files or directories in the current directory with the given extension and exclude given extension files in the node modules.

**find . -type f -size +100c**

🡪Looks for the files only in the current directory with files larger than 100 bytes.

**find . -type f -size +100k -size -1M**

🡪 Search files bigger than 100KB but smaller than 1MB

**find . -type f -mtime +3**

🡪 Search files edited more than 3 days ago

**find . -type f -mtime -1**

🡪 Search files edited in the last 24 hours

**find . -type f -mtime -1 -delete**

🡪You can delete all the files matching a search by adding the -delete option.

**find . -type f -exec cat {} \;**

🡪Looks for files only in the current directory. For each file found, execute **cat** to display the contents.

🡪{} – Placeholder for the file name.

🡪\; -- Terminates the -exec command

1. **ln**

🡪 The ln command is part of the Linux file system commands.

🡪It is used to create links. It’s like a pointer to another file.

🡪You might be familiar with windows shortcuts. They are similar.

🡪We have 2 types of links: **hard links**  and **soft links**

**Hard links:**

🡪Hard links are rarely used . They have a few limitations: you can’t link to directories, and you can’t link to external file systems(disks).

🡪A hard link is created using

**ln <original> <link>**

🡪 For example, say you have a file called recipes.txt. You can create a hard link to it using:

ln file.txt myfile.txt

🡪 The new hard link you created is indistinguishable from a regular file.

🡪 Now any time you edit any of those files, the content will be updated for both.

🡪 If you delete the original file, the link will still contain the original file content, as that's not removed until there is one hard link pointing to it.

**Soft links:**

🡪Soft links are different. They are more powerful as you can link to other filesystems and to directories, but when the original is removed, the link will be broken.

🡪You create soft links using the **-s** option of **ln.**

**ln -s original> <link>**

🡪 For example, say you have a file called recipes.txt. You can create a soft link to it using.

ln -s file.txt myfile.txt

🡪In this case you can see there’s a special **l** flag when you list the file using **ls-al,** and the file name has a @ at the end, and it’s colored differently if you have colors enabled.

🡪 Now if you delete the original file, the links will be broken, and the shell will tell you "No such file or directory" if you try to access it.

1. **gzip**

🡪You can compress a file using gzip compression using **gzip** command.

**gzip filename**

gzip – Compresses the file.

Filename – The file to be compressed.

🡪The output will be **filename.gz** , and the original file will be removed.

**gzip -k filename**

gzip – Compresses the file.

Filename – The file to be compressed.

🡪The output will be **filename.gz** , and the original file will not removed.

**gzip -d filename.gz**

🡪To decompress the file.

**gzip file1 file2 file3**

🡪To compress multiple files.

**gzip -v aws.txt**

🡪It is used for debugging.

1. **Tar**

🡪The tar command is used to create an archive, grouping multiple files in a single file.

**tar -cf archive.tar file1 file2**

🡪tar → Creates or extracts archive files.

🡪-c → Creates a new archive.

🡪-f archive.tar → Specifies the output file (archive.tar).

🡪file1 file2 → The files to be added to the archive.

**tar -tf archive.tar**

🡪To verify the archive contents.

**tar -xf archive.tar**

🡪To extract the archive.

**tar -czf archive.tar.gz file1 file2**

🡪Compress while archiving with gzip.

🡪tar → Creates or extracts archive files.

🡪-c → **Creates** a new archive.

🡪-z → **Compresses** the archive using **gzip**.

🡪-f archive.tar.gz → Specifies the output file (archive.tar.gz).

🡪file1 file2 → The files to be added to the archive.

**tar -tzf archive.tar.gz**

🡪To verify the archived contents in the zip file.

**tar -xzf archive.tar.gz**

🡪To extract the archived contents in the zip file.

1. **Alias**

**alias ll='ls -al'**

🡪alias → Defines a shortcut for a command.

🡪ll → The custom shortcut name.

🡪'ls -al' → Expands to ls -al, which:

* -a → Shows **hidden files**.
* -l → Displays **detailed file information**.

**alias lsthis="ls $PWD"**

**alias lscurrent='ls $PWD'**

🡪$PWD refers to the current folder the shell is into. If you now navigate away to a new folder, lscurrent lists the files in the new folder, lsthis still lists the files in the folder you were when you defined the alias.

1. **Cat**

**cat file**

🡪It simply prints the content in the file.

**cat file1 file2**

🡪It prints the content from the multiple files.

**cat file1 file2 > file3**

🡪It is used to concat the content of the multiple files into a new file.

**cat file1 file2 >> file3**

🡪It is used to append the content of the multiple files into a new file, creating if it does not exist.

**cat -n file1**

🡪cat → Displays the contents of a file.

🡪-n → Adds line numbers to the output.

🡪file1 → The file being displayed.

1. **Less**

**less filename**

🡪The less command is one I use a lot. It shows you the content stored inside a file, in a nice and interactive UI.

🡪Once you are inside a less session, you can quit by pressing q.

1. **wc**

**wc filename**

🡪The wc (**word count**) command in Linux is used to count **lines, words, characters, and bytes** in a file.

**wc -l test.txt**

🡪To count the lines in a file.

**wc -w test.txt**

🡪To count the words in a file.

**wc -c test.txt**

🡪To count the bytes in a file.

**wc -m test.txt**

🡪To count the characters in a file.

**wc -L test.txt**

🡪To find longest line length in a file.

1. **grep**

**grep document.getElementById index.md**

🡪grep → Searches for a pattern in a file.

🡪document.getElementById → The string to search for.

🡪index.md → The file to search in.

**grep -i document.getelementbyid index.md**

🡪Case-insensitive search.

**grep -n document.getElementById index.md**

🡪It will show the line numbers.

**grep -nC 2 document.getElementById index.md**

🡪grep → Searches for a pattern in a file.

🡪-n → Shows **line numbers** where matches are found.

🡪-C 2 → Displays **2 lines of context** **before and after** the match.

🡪document.getElementById → The search pattern.

🡪index.md → The file being searched.

1. **sort**

🡪The sort command helps us sorting the file with names.

🡪Use the r option to reverse the order.

🡪Sorting by default is case sensitive, and alphabetic. Use the --ignore-case option to sort case insensitive, and the -n option to sort using a numeric order.

🡪You can use -u option to remove them.

**ls | sort**

🡪ls → Lists files and directories in the current directory.

🡪| (pipe) → Passes the output of ls to the sort command.

🡪sort → Sorts the list alphabetically.

1. **uniq**

🡪Uniq is a command useful to sort lines of text.

**uniq filename**

🡪**uniq** removes **consecutive duplicate lines** from a sorted file.

🡪**Issue**: uniq only works on **sorted** data.

1. **Diff**

🡪diff is a handy command. Suppose you have 2 files, which contain almost the same information, but you can't find the difference between the two.

🡪diff will process the files and will tell you what's the difference

**diff file1 file2**

🡪It gives the difference between the 2 files.

**diff -y file1 file2**

🡪It will compare 2 files line by line.

**diff -u file1 file2**

🡪diff → Compares two files **line by line**.

🡪-u (**unified format**) → Provides a **clearer** and **more readable** output, showing **context lines** around changes.

🡪file1 file2 → The files being compared.

1. **Echo**

**🡪**It prints the output the argument passed to it.

**Ex:** echo “Hello”

**echo "hello" >> output.txt**

🡪We can append the output to a file.

**echo \***

🡪We can echo the files in the current folder.

**echo o\***

🡪We can echo the files in the current folder that start with the letter o.

**echo ~**

🡪You can print your home folder path.

**echo $(ls -al)**

**🡪** You can also execute commands, and print the result to the standard output.

1. **chown**

🡪The chown command is used to change the owner of the file or directory.

**chown <owner> <file>**

🡪The owner can change the owner to another user too, using the chown command.

**chown user1:group1 file.txt**

🡪Changes the file.txt to user1 and group to group1.

**chown user1 file1 file2**

🡪It changes ownership for multiple files.

**chown -R user1:group1 /home/user1**

🡪chown → Changes the **owner** and **group** of a file/directory.

🡪-R (**recursive**) → Applies changes **to all files and subdirectories** inside /home/user1.

🡪user1:group1 → Sets **user1** as the owner and **group1** as the group.

🡪/home/user1 → The directory being modified.

🡪If the owner does not exist.

🡪Create a owner using the following commands:

**useradd user1**

**passwd user1**

🡪To go back to the root user:

**exit**

1. **chmod(Change mode)**

🡪You can change the permissions given to a file using the chmod command.

🡪Then you have 3 set of values:

* The first set represents the permissions of the owner of the file.
* The second set represents the permission of the members of the group of the file.
* The third set represents the permissions of the everyone else.

🡪Those sets are composed by 3 values. **rwx** means read, write and exexute actions.

🡪chmod command is used in 2 ways: First is using symbolic arguments, the second is using numeric arguments.

🡪You type chmod followed by a space, and a letter:

* a stands for all
* u stands for user
* g stands for group
* o stands for others

🡪Then you type either + or - to add a permission, or to remove it. Then you enter one or more permissions symbols ( r , w , x ).

Example: **chmod a+rw filename**

🡪This number value can be a maximum of 7, and it's calculated in this way:

* 1 if has execution permission
* 2 if has write permission
* 4 if has read permission

🡪This gives us 4 combinations:

* 0 no permissions
* 1 can execute
* 2 can write
* 3 can write, execute
* 4 can read
* 5 can read, execute
* 6 can read, write
* 7 can read, write and execute

1. **umask (user mask):**

🡪The umask command sets the default permissions for the newly created files and directories.

**umask mode**

1. **du(disk usage):**

🡪The du command shows the size of files and directories.

**du**

🡪To shows the size of the files and directories separately.

**du \***

🡪Human-readable format.

**du -h**

🡪Shows the total size of the directory.

**du -sh**

🡪Show total size at the end.

**du -hc**

🡪Show sizes of all the files(not directories)

**du -ah**

🡪Limit the depth of the directory scanning.

**du -h –max-depth=1**

1. **df(disk filesystem):**

🡪 The df (**disk filesystem**) command shows the **available and used disk space** on mounted file systems.

**df**

🡪Using the -h option, it will show the those values in human-readable format.

**df -h**

1. **basename:**

🡪The basename command removes the directory and returns only the file or directory name.

**basename path**

1. **dirname:**

🡪The dirname command removes the filename and returns only the directory path.

**dirname path**

1. **ps(process status):**

🡪The ps command shows the information about active processes.

**ps**

🡪Show all processes.

**ps -e**

🡪Full format listing(detailed info)

**ps -ef**

🡪Show processes of a specific user.

**ps -u user1**