(Cmd --help)

LINUX Commands

REPL → Infinite loop
Shell (or) REPL console
R - Read
E - Evaluate
P - print
L - Loop

Cmd --help

Pwd → Print working directory
(or)
current working directory.

```
pwd --help
pwd -h (Invalid)
```

pwd → print the name of current working directory
[By default it is pwd -L]

options :

- -L : print the value of PWD
- -P : print without any $mybolic links
- -W : print Win32 Value.

Example :
```
pwd -L  /c/users
pwd -P  /c/users
pwd -W  C:/users
```

Macos : pwd --help does not work use command man pwd.

pwd → This represents what is current directory we are actually at.

ls → All the files and sub-directories will be printed.

```
ls -lh
ls -l
ls -a
```
ls -a → Shows all files including hidden files

Cd → Change directory

Ex: cd D:

Cd → This can help you to move into a folder and move out of a folder.

Ex:
cd developer → To move into developer folder

cd .. → move out of the folder
↳ refering to one folder to move back.

Cd .. → If you want to jump one folder back from current directory.

cd ../.. → If you want to make two jumps back from current directory.

To come back to home directory:

→ cd ~ (~ ⇒ tilda)

Cd ~ : from any directory this will help you to come back to home directory.

→ cd Developer/GIT

cd directory1/directory2/...
↳ we can actually move into internal subdirectories directly by seperating them with forward slash (/).

Note: ~ $\longrightarrow$ refers to home directory
(tilda) (cd ~)

/ $\longrightarrow$ refers to root directory (cd /)

Relative Path	Absolute Path
It describes location of file / folder wrt current folder	In it we mention the location from home directory (or) root directory ↓ The folder which is not having parent folder

```
cd / ——> root directory
cd ~ ——> home directory
```

Note: Home directory is subfolder of root directory

$\longrightarrow$ cat ~/main.dart
↳ print the content of file

Note: When you give absolute path of a file or a folder that means you will give the whole path of that file / folder, whereas in relative path we do jumps wrt to current folder.

Clear $\longrightarrow$ This clears the console (or) working space by actually moving you to top of current shell.

(*) ls

ls -lh → Similar to ls -l but it shows size in KB, MB (k, M)

ls -l → It shows more information like when file was created (date), who is owner of file, permissions of file (read, write)

ls -a → To show hidden files

ls -h → human readable format

ls --help → To sell all flags of ls

Creating files & folders:

→ mkdir <folder_name>
  ↳ Helps us to create new folder

→ touch <file_name>
  ↳ Helps to create a new blank file.

→ cat <file_name>
  ↳ prints the whole content of file.

→ rm <file_name>
  ↳ removes the file

Note: using rm we cannot remove subfolder.

→ rmdir <sub_folder>
↳ It removes subfolder

Note: When you perform remove operation using commands, the removed file will not be available in recycle bin too.

→ So, avoid doing remove operation using command prompt.

→ rmdir <sub_folder>
↳ It doesnot work when subfolder is not empty.

In order to delete we use:

→ rm -r subfolder
↳ The -r flag enables rm to recursively delete all the content of the folder and then delete the folder.

→ rm -rf
rm -rf/ → Try to avoid because it delets / root directory all the files in p.c gets erased.

Vim → Text editor

Linux terminal by default gives access to text editors → vim, NaNo, emac

To open :
Vim test.js / Vi newfile.txt
(Insert — i)

To exit:

Esq :q / Esq :wq

Insert — i (we can make changes)
Esq — Normal mode (we cant insert)

→ If we made changes,
you want to save changes & exit → Esq :wq

→ If you dont want to save changes & exit → :q!

≈≈ vim <filename>: this will create a file and then open it in vim editor in normal mode.
In normal mode we can do changes to file but we can read & navigate it.

Now to ~~go~~ start making changes we need insert mode press <u>i</u>. If you want to come back to <u>normal mode</u> press <u>ESC</u>.

→ Esc + :q → To exit a file

→ Esc + :q! → To exit a file without saving changes.

→ Esc + :wq → To exit file with saving changes

Cursor moves :
In normal mode
L → To right
H → To left
J → To down
K → To up

VIM adventures — game

To delete: press Esc ~~to~~ go normal mode ~~sed~~

→ dd : In normal mode it will delete the line at which cursor is excatly.

Back Page

In normal mode:

To go to start of file → gg

To go to last line → 'G'

w: In normal mode, it can jump one word

2w: " , it can jump two words

d2w : This will delete two words

yy: copy whole line

P : paste whole line

yw: It copies one word

Esc :%S/foo/bar/ : In normal mode to replace all occureances of foo with bar

Search

Vim ~/. vimrc   My-Vim-Config
↑ repository code

→ tail <file_name> ⟶ To get bunch of last lines.
tail -n 3 <file_name> ⟶ last 3 lines
tail -f <file_name>

→ head <file_name> ⟶ To get bunch of first lines.
head -n 10 <file_name>

→ echo "vamsi" ⟶ To print on terminal
→ echo " vamsi" > dump.txt ⟶ The result vamsi is dumped into file dump.txt
echo "vamsi" > dump.txt

dd: In normal mode deletes line
dd: In insert mode it will write

Piping → Takes o/p of one command and passes as I/p to another Command

→ ls | grep <folder>

The result of first command is given as input to second Command.
grep → does substring search

→ ps aux .
Shows all the processors running in system

→ ps aux | grep chrome
ps aux | grep mysql

<u>Dumping</u>

To dump some logs into a particular file

```
ls > test.txt      |   pwd > test.txt
~cat test.txt      |   ls >> test.txt
```

$\boxed{> \longrightarrow \text{replaced} \quad >> \longrightarrow \text{append}}$

<u>To execute C++ code</u>:

$$\text{g++ } \underbrace{\text{permutation.cpp}}_{\text{file}} \text{ --std=} \underset{\text{version}}{\underset{\downarrow}{\text{c++14}}} \ \underset{\text{binary}}{\underset{\downarrow}{\text{-o}}} \text{ run}$$

```
ls | grep run

./run
```

<u>Clubbing commands:</u>

```
g++ file.cpp --std=c++14 -o run && ./run
```

↳ This even happens in VS Code when we execute program.

→ ls | grep python : This will actually pass the o/p of ls as an input to grep, grep does substring search of python on o/p of ls.

→ ls > new.txt : Whatever is the result of ls will be dumped into new.txt, nothing will be printed on console. If new.txt has some content then that will be replaced.

⟶ ls >> new.txt : This appends the content to new.txt file.