long list format
ls -l ≡ ll

lll USE TAB! ls → list ls -a → shows . .. and hidden files

pwd → present working directory
cd → change directory
mkdir → make directory, rmdir → remove directory
lsblk → disk partition it tells
sudo apt update → updates package index
sudo apt upgrade → upgrade to latest version
sudo apt full-upgrade → allow removal of existing packages to install new.

conda activate <my env = zephyr1>
conda deactivate
conda ~~install~~ install <package name>

root user :- home ≡ /root
regular user :- home ≡ /home/medhansh
(This is relative path) username

. ≡ Shortcut for current directory
.. ≡ Shortcut to parent directory.

(eg)
HOME → documents
→ Music
to go from doc to music
do :- cd ../Music/
this is 'home'

ls -X ⇒ lists same extension directories together alphabetically

ls -x ⇒ lists alphabetically but row wise (ls, alphabetically column wise)

ls -i ⇒ also prints inode no

mv can change name

ln file1 file2 makes hard links (same inode as file1)

file2 ≡ file1
↳ created

↑ & file2 = file1 basically.

name does not change inode no.

Vim usually makes a copy, changes are applied there, and the old inode is released new inode gets the file.

but if files are hardlinked, vim cannot release that old inode.

ls -R → recursively list (the sort)

ls -r → simply lists in reverse of the Sort

(eg: (alphabetically)$^{-1}$)
↳ ls -r

rmdir -pv
↳ removes specific and parent dir if they are empty

cp -r (recursive) has to be done when directories are copied.
cp -v will display only the files been copied
-i prompts before overwriting

---

mv [option] <source> <destination>

-v :- details about files being moved including source and destination (shows 'renamed' instead of moved)
-n → no clobber ⇒ prevents overwriting of existing files. If destination file exists, move is not performed.
i → interactive
f → force the move operation

---

rm (remove)
rm -d = rmdir (remove empty dir)
rm -f (forces the removal of files without confirmation)
-v

f → force the move operation

rm (remove)
rm -d = rmdir (remove empty dir)
rm -f (forces the removal of files without confirmation)
-v (verbose)

echo
-E disables escape character interpretation which is default
-e enables \n, \t etc

touch
Stat tells us status:-
Access, modification, Change, Birth } timings ; if you access or modify change anyways happens

touch file (will create file if not present)
touch -c file (will only touch if present wont create)
-a → updates only access
-m → updates only modify
touch -t [[CC]YY]MMDDhhmm[.SS]
touch -t 202411070900.30 Tree.cpp

prev command will touch. Access, modify & Todo.cpp to 2024-11-07 9:00:30.00

cat

concatenates and displays content of files

-v → shows non-printable characters as escape sequences (excluding tabs and new lines)

-b → numbers only non empty lines

-n → number all output lines

-s → suppresses repeated empty lines and outputs single blank line for consecutive empty lines

cat joke* → joke, joke1, joke2, joke3 ✓

less

view file contents also as

-i → case insensitive ✓ (only if all lowercase)

-N → line numbers

/ to search words, q to quit

-I pretty case insensitive

more

more +19 → start at line 19

-5 → upon pressing b, go back by 5 lines

head (and tail)

-n (number) bigfile

↳ will show 5 lines from top (bottom)

default is 10.

more

more +19 → start at line 19

-5 → upon pressing b, go back by 5 lines

head (and tail)

-n (number) bigfile

↳ will show n lines from top (bottom)

default is 10.

- tail -f follows original file (inode)
- tail -F follows file name!

→ When using nano (inode no. stays same) so tail -f and tail -F both update to alongside nano

→ When using vim inode no. changes if there are no hard links to it (replaced file) so tail -f ~~loses~~ follows prev inode only. tail -F works, but it shows file replaced and follows new file of same name!

Advanced Commands

* wc (word count)

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

--files0-from=(F)

filenames are separated by null characters \0 instead of new lines → eg:- file1.txt

eg:-

find . -name "*.txt" -print0 > file1.txt
(separated by)

wc --files0-from=file1.txt

- -c ⇒ Byte count ≡ --byte
- -m Character count
- -l newline count
- -w word count
- -L print length of longest line

newline, word, byte count is default

Regular Expressions.

"^" ⇒ Beginning of line

-w word count
-L print length of longest line
newline, word, bytecount is default

Regular Expressions.

"^" ⇒ Beginning of line
(not if inside [ ])

"$" ⇒ end of line
"." ⇒ match any single character
"\" ⇒ escape a special character
"|" ⇒ or operation

Quantifiers:-
* → 0 or more times (preceding item)
? → 0 or 1
+ → 1 or more
{n} exactly n times
{n,} atleast n times
{,m} atmost m times
{n,m} n to m times

( ) → group together
{ } → [ ] → any character from this range.

ab[xyz]c => abxc, abyc, abzc

[^ ...] match
↳ anything not here

[a-z] → all small case

[A-Z] → all capital

[0-9] → match digit

---

grep [OPTIONS].. pattern [File]....

grep [OPTIONS]... -e ——

grep —— -f ——

[option]

grep -f pattern file .... [File]....

[option]

grep -e Patterns.... file

-i = ignore case

-v = invert match (show lines which do not match the pattern)

-r or -R = recursively search directories

-n = show line number with matching lines

-c Cant number of matching lines.

-H Print the filename for each match
ex: grep -H "wolf" file1.txt file2.txt

-o print only matched parts of line

-E use extended regular expressions

match the pattern)

-r or -R = recursively search directories

-n = show line numbers with matching lines

-c Count number of matching lines.

-H Print the filename for each match
es: grep -H "wolf" file1.txt file2.txt

-o print only matched parts of line

-E use extended regular expressions

-w use match only whole words
(es 'Per" will not match Perhaps)

-A (Num) display lines of text after matching line

-B (Num) " " before " "

-C (Num) both before and after

or -Num print Num lines of output context (before and below)

-C 0 ≡ -0

This doesn't work if used with -0 (gives warning)
es:
--} operators.

-f ⇒ interpret pattern as fixed strings
es "Patterns." will not match to "Patterns8"