Distribution=made by taking Linux core + some tools
cat /etc/os-release

Kernel=core app; allocates resources & talks to HW
uname -r (-a)
Latest: dnf list kernel
Install: dnf install kernel-devel --best
sudo dnf update kernel → reboot
sudo dnf –y update = update all

Shell=app that interprets the commands
Current: ps $$ or echo $0
Default: echo "$SHELL"
List: chsh -l or cat /etc/shells
Change: chsh -s ***shell_name*** → log out

Terminal=app where we type the commands
ps -p$PPID (term app creates shell, so it is the parent of the shell)
echo $TERM (term type; tells apps how to interact with term)

Prompt = system symbol of cmd line (**#,$,%,:**)
Continuation prompt: > (continuation of previous line)
Breaking cmd in various lines: \ or |
Separating 2 commands at one line: ; or &&
Autocomplete opens with tab + ↑↓ + enter
Depends on the context (*cd +tab vs cp +tab*)

CTRL+shift+n = open shell in new window
CTRL+shift+t = open shell in new tab
CTRL + l = clear screen
CTRL + r = history block search
CTRL+D = terminate the shell

ALT+b/f = move backward/forward word by word
CTRL + u = cut/erase the whole line
CTRL + k = cut/erase line right from the cursor
CTRL + w = cut/erase word left
ALT + d = cut/erase word right
CTRL + y = paste (1st buff)
CTRL+SHIFT+c = copy highlight text (2nd buff)
CTRL+SHIFT+v = paste 2nd buff; after usage=1st buff

ALT + c = capitalise first letter of the word
ALT + u = uppercase the rest of the word
ALT + l =lowercase rest of the word

who - show who is logged on
whoami - print userid
pwd - print current directory (= echo $PWD)
man *cmd* = manual (*cmd -h* or *cmd --help*)
type *cmd* - type of a cmd tool
-a : all occurrences of cmd name
which *cmd* - which binary are you executing?
which cmd vs sudo which python
whereis *cmd* - location of the binary/source/man files
history – last 15 commands
-100 = last 100 commands
-i = include all information
echo $HISTFILE → ~/.history
!+number_hist_line (!!=repeat last cmd -> sudo !!)
echo - send argument to stdout
-n = doesn't add new line character
cat - send content of file to stdout
-n = add number to all output lines
head - show 10 first lines of file
-n K = first K lines instead of 10
-c K = first K bytes
-n/c -K = all but the last K lines/bytes
tail - show last 10 lines of file
-n K = the last K lines instead of 10
-c K = last K bytes
-n/c +K = starting with K lines/bytes
-f = output appended data as the file grows;

/ = root directory
./ = current directory
../ = upper (parent) directory
~ = user home directory
.name = hidden dirs/files start with dot!
name~ = backup files
\ = escape character (split cmd line, special char)
$ = preceding variable name ("\$" to print $)
$0 = name of the running process.
$(cmd) = cmd substitution
$((...)) = arithmetic expansion operator
# sizeof
| = pipe→use output of cmd 1 as input to cmd 2
0< = stdin 1> = stdout 2> = stderr &>= stdout&err
stderr by default is going to the console as stdout
cat < file > file_content 2> error_content
> = stdout redirection → overwriting the output file
>> = stdout redirection → appending to output file
< = take stdin from file (wc < file, <file wc, wc file, cat file | wc)
2>&1 = redirect (add) errors to stdout
/dev/null = null device; discard all data & ret success

cd - navigate between dirs
= with NO arguments takes us to ~
- = toggle between the last two dirs.
mkdir - make a directory
-p = make parent directories as needed
touch - creates empty file/updates access & modif time
cp - copy a file/ directory
mv - move/rename files/directory
cp/mv -options source destination
-r : recursive mode used for directories
-i : interactive confirm file overwriting
-v : verbose see copy progress
-p: preserve file permission/attributes
rm - eliminate files
-f : force, never prompt

chmod - change file read/write/execute permissions
ugo = user/group/other (a=all)
rwx= read/write/execute
u(**rwx**)/g(**rwx**)/o(**rwx**)->9 binary->3 decimal->ex:737
ex: u+r+w,g-w,o+wrx (NO space in parameters)

ls – print the contents of the current dir
- 1 = 1 output per line
- s = size
- l = long = all information
- a = all -> hidden directories/files start with dot!
- H = follow symbolic links
- R = list subdirectories recursively
- d = do not enter inside directories
- S = sort by file size
- t = sort by modification time, newest first
- X = sort alphabetically by entry extension
- r = reverse order while sorting

Pattern matching @command line
* = match all files and subdirectories (show subdir content)
*x = restrict to files and subdirectories starting with x
x = restrict to files and subdirectories containing with x
*x = restrict to files and subdirectories ending with x The
* = any number of unknown characters,
? = only one unknown character
^ = negation (*(^/)=any pattern not having "/" inside)
If restriction result is empty NO filter is used

List just directories : ls -d *(/) ; ls -d */ ; echo */
List just files: ls -a *(^/)
List hidden dir/files = ls -ld .*
Get files/dirs with abs path: ls -d -1 $PWD/*
For entering 2nd level: ls -d -1 $PWD/*/*

' ' - single quotes=do not touch this text
" " - double=perform shell variable expansion
` ` - evaluate & replace=cmd substitution (`≠')
== $(cmd) BUT ≠ $cmd

wc - print line, word, and byte counts
-c = print the byte counts
-m = print the character counts
-l = print the newline counts
-w = print the word counts
seq - print sequence of numbers (*start step stop*)
-f = format (*-f %5.1f, -f%3.1e, -f "Line: %g"*)
-s = delimiter (default = \n)
less - interactively show content of a file
-N = show line number
-S = truncate lines wider than window
Use this while reading:
G / g = go to end/beginning of file
q = quit
/ = forward Search (? = backward search)
^pattern : pattern @ beginning of line
pattern$: pattern @ end of line
n – next match (N = previous match)

{ } → parameter expansion
{a,b{1..3},c} = a b1 b2 b3 c
mv log{,.OLD} = mv log log.OLD
echo {00..8..2} = 00 02 04 06 08 echo {D..T..4}
→ variable identification
VAR=AB; echo $VAR12; echo ${VAR}12
→ Text replacement, after find & xargs = {}
→ Block of code = { cmd1; cmd2; . . . cmdN; }

( ) → evaluate & replace
→ array creation = array=(1 2 3)
→ subshell creation = pwd; (cd /; pwd); pwd
(( )) → arithmetic operations:
((a = 42)) ((a++)) echo $((a + b + (14 * c)))
for ((i=0; i<10; i++))

[ ] → test commands (*man test*)
["$foo" -lt 3] or [[$bar =~ ^123]]
→ range or character class
ba[rz], foo[[:alnum:]], qu[[=u=]]x
→ part of an array assignment
f=(3 4); f[42]=bar; echo $f,$f[2],$f[3],$f[42]
[[ ]] → Extended test construct builtin

find - search for files *[path] [conditions]*
-type + f=file, d=directory
-name = find by name (-iname = case insensitive)
find . -type f -name "text_file"*
-maxdepth/mindepth = max/min dir levels *(Level 1=./)*
-perm p = with permissions p *(p is integer ex: 757)*
-not = ! = invert the match
-size +/-n= file larger/smaller than n (-empty)
-mmin N = files modified within N minutes
-mtime N : files modified within N days
-newermt YYYY-MM-dd = modified on or after date
-exec cmd = execute command on every found file
-ok cmd = prompt before executing on a file
*find *.txt -exec ls {} \; -exec **sh -c** "head {} | tr A B" \;*
All occurrences of {} are replaced by the filename.

dir=(*) = store dir content in array
du -a --max-depth=1 = disk usage
df . = amount of available disk space for current dir
tree -f -L 2 = contents of dirs in a tree-like format.
export GIT_EDITOR=vim -> kwrite