INTRODUCTION TO LINUX OPERATING SYSTEM

WHAT IS LINUX?

- Free, open-source OS based on Unix
- Created by Linus Torvalds (1991)
- Runs on servers, desktops, mobiles, embedded systems
- Known for stability, security, flexibility

OPEN SOURCE

- Source code freely available
- Anyone can view, modify, distribute
- Global community contributes to development

INTERFACES

- Command-line interface (CLI)
- Graphical interfaces: GNOME, KDE, Xfce
- Supports wide range of applications

KEY FEATURES

- Multi-user
- Multi-tasking
- CLI + GUI
- Hardware compatibility
- Strong security
- High stability
- Highly customizable
- Community-driven

LINUX SYSTEM ARCHITECTURE

[User Applications]

↓

[System Libraries]

↓

[Linux Kernel]

↓

[Hardware]

ARCHITECTURE OVERVIEW

- Kernel manages CPU, memory, devices
- User-space programs interact via system calls
- Runs on many hardware platforms
- Uses hierarchical filesystem

PASSWD

- Change a user's password
- Root can change any user’s password
- Options:
  - `-l` lock account
  - `-u` unlock
  - `-d` delete password

WHO

- Shows currently logged-in users
- Displays username, terminal, login time, host
- Options:
  - -a all info
  - -b last boot
  - -u user details

W

Shows:

- System time, uptime
- Logged-in users count
- Load averages
- USER, TTY, FROM, LOGIN@, IDLE, JCPU, PCPU

TTY

- Shows current terminal device
- Example: `/dev/pts/0`
- `tty -s` checks if session is a terminal

LOCK

- Locks a terminal session
- Prompts for confirmation
- Requires password to unlock
- Not available on all Linux distros

STTY

- Views or modifies terminal I/O settings
- Controls echo, speed, line discipline
- Example: `stty -a` shows current settings

SCRIPT

- Records all terminal activity
- Saves output to `typescript` by default
- Useful for logs, installations, debugging
- Start: `script filename`
- Stop: `exit`

CLEAR

- Clears the terminal screen
- Simply run: `clear`

UNAME

- Prints system information
- Options:
  - `-a` all info
  - `-m` machine type
  - `-n` hostname
  - `-r` kernel release
  - `-s` OS name
  - `-v` OS version

DATE

- Shows or sets system date/time
- Custom format with +FORMAT
- Example: `date "+%Y-%m-%d %H:%M:%S"`
- Set date: `date -s "2025-01-01 10:00:00"`

CAL

- Shows a calendar
- Default: current month
- Example: `cal 6 2022`

CALENDAR

- Reads reminders from `~/.calendar`
- Shows events for current date (if configured)

BC

- Precision calculator language
- Used in scripts for math
- Example: `echo "5*7" | bc`

ECHO

- Prints text or variable values
- Example: `echo "Hello"`
- Example: `echo $USER`

CD

- Changes directory
- Examples:
  - `cd` or `cd ~` → home
  - `cd ..` → parent
  - `cd /path/to/dir`
  - `cd -` → previous directory

- Options:
  - -P follow symlinks
  - -L don’t follow symlinks

MKDIR

- Creates directories
- Syntax: `mkdir [options] name`
- Options:
  - `-p` create parent dirs
  - `-m` set permissions
- Example: `mkdir -p -m 755 /path/to/dir`

RMDIR

- Removes an **empty** directory
- Fails if the directory is not empty
- Options:
  - `-p` remove parent dirs if they become empty
  - `--ignore-fail-on-non-empty` ignore errors
- Example: `rmdir mydirectory`

CP

- Copies files or directories
- Syntax: `cp [options] source destination`
- Options:
  - `-r` recursive copy (directories)
  - `-i` prompt before overwrite
  - `-v` verbose
  - `-p` preserve attributes
- Example: `cp myfile.txt mydir/`

RM

- Deletes files or directories
- Options:
  - `-r` delete directories recursively
  - `-f` force delete without prompts
  - `-i` confirm before delete
  - `-v` verbose
- **Dangerous**: deletion is permanent
- Example: `rm -r foldername`

MV

- Moves or renames files/directories
- Options:
  - `-v` verbose
  - `-i` prompt before overwrite
  - `-u` move only if newer
  - `-f` force overwrite

- Examples:
  - `mv file1.txt dir1/`
  - `mv file1.txt file2.txt`
  - `mv dir1/*.txt dir2/`

CAT

- Displays file content or concatenates files
- Options:
  - `-n` number all lines
  - `-b` number non-blank lines
  - `-s` squeeze blank lines
  - `-v` show non-printable characters

- Examples:
  - `cat file1.txt`
  - `cat file1.txt file2.txt`
  - `cat -n file1.txt`

LINUX FILE NAMING

- Case-sensitive
- Max length ~255 chars
- Avoid spaces; use _ or -
- Cannot use / or \
- Prefer lowercase, descriptive names

HIERARCHICAL DIRECTORY STRUCTURE

- Follows FHS (Filesystem Hierarchy Standard)
- Root at /
- Organized like an inverted tree
- Key dirs: `/bin`, `/etc`, `/home`, `/var`, `/usr`, `/tmp`, `/root`

ABSOLUTE VS RELATIVE PATH

ABSOLUTE PATH

- Starts from /
- Full location of a file
- Examples:
  - `/home/user/documents/file.txt`
  - `/var/log/syslog`
  - `/usr/bin/python3`

RELATIVE PATH

- Based on current directory
- Does **not** start with /
- Examples:
  - `./file.txt`
  - `../folder/file.txt`
  - `../../dir/file.txt`

LS

- Lists directory contents

LS -L

- Shows detailed listing
- Includes permissions, owner, group, size, date, filename
- Example entry:

```
-rw-r--r-- 1 user group 4096 Jan 1 00:00 file1.txt
```

LS -D

- Shows directory names **only**, not contents
- Example:
- `ls -d directory1`

chmod

Used to change file/directory permissions.

Syntax: `chmod [options] mode file`

Modes:

- Numeric (e.g., 644 → owner RW, group R, others R)
- Symbolic (e.g., `u+w`, `g-r`, `o=x`)

Examples:

```
chmod 644 file1.txt
chmod 755 dir1
chmod u-w file1.txt
chmod go+x dir1
```

PWD

Shows the current working directory.

`CD`

Changes directory.

`cd /path` → go to a specific directory.

CD (HOME)

`cd ~` takes you to your home directory.

CD (PARENT)

`cd ..` moves one level up.

LS

Lists files in the current directory.

LS (OPTIONS)

Common options:

-l → long list

-t → sort by time

-S → sort by size

-h → human-readable sizes

-r → reverse order

Combine: `ls -ltr`

LS -LTR

Long listing sorted by time, reversed.

WILDCARD (*)

* matches any set of characters in filenames.

MKDIR

Creates a new directory.

RMDIR

Removes an empty directory.

DISPLAYING FILES

Tools: `cat`, `less`, `head`, `tail`.

CAT

Shows entire file on screen.
Good for short files.

LESS

Scrollable file viewer.
Enter → scroll line
Space → scroll page
y → back one line
b → back one page
`/text` → search
q → quit.

HEAD

Shows top of a file.

Default: first 10 lines.

`head -n50 file.txt` → first 50 lines.

TAIL

Same as head, but shows the bottom lines.

CP

Copies a file.

MV

Moves a file to another location.

MV (RENAME)

Also renames files.

RM

Removes a file.

RM -R

Removes directories and files recursively.
Dangerous—permanent deletion.

FILE PERMISSIONS

Each file has read, write, execute permissions.

`ls -l file` shows permissions.

PERMISSION TYPES

r → read

w → write

x → execute (for directories: list contents)

PERMISSION GROUPS

User → owner

Group → group members

World → everyone else

CHMOD

Changes permissions (if you own the file).
Format: `chmod [u/g/o/a]+[r/w/x] file`
Example: give execute to all users.

PS

Shows processes you are running.

TOP

Shows CPU usage of all processes.

KILL

Terminates a process.

I/O REDIRECTION & PIPING

`a | b` → output of a → input of b

`a > out.txt` → write output to file

`a < in.txt` → read input from file.

PIPING EXAMPLES

Combine commands using | to filter or transform output.

WC

Counts lines, words, characters.
Output columns: lines, words, chars.

umask

Sets default permissions for newly created files/directories.

Syntax: `umask [mode]`

Mode is octal (e.g., 022 prevents write for group/others).

Examples:

```
umask 022
umask -S
umask -p
```

chown

Changes owner/group of files.

Syntax: `chown owner:group file`

Examples:

```
chown new_owner file1.txt
chown :new_group dir1
chown new_owner:new_group file1.txt
chown -R new_owner:new_group dir1
```

chgrp

Changes group ownership of files/directories.

Syntax: `chgrp new_group file`

Examples:

```
chgrp new_group file1.txt
chgrp new_group dir1
chgrp -R new_group dir1
```

touch

Creates an empty file or updates timestamps.

Syntax: `touch [options] file`

Options:

- `-a` update access time
- `-m` update modification time
- `-c` don’t create if missing
- `-r` copy timestamps from another file

Examples:

```
touch file1.txt
touch -m file1.txt
touch -r file2.txt file1.txt
```

Displays file content page by page.

Syntax: `more [options] file`

Options:

- `-n` show line numbers
- `-p` show given percentage per page
- `-c` clear screen before each page
- `-d` show “more” prompt message

Examples:

```
more file1.txt
more -n file1.txt
```

```
more -p 25 file1.txt
more -c file1.txt
```

Searches files/directories based on conditions.

Syntax: `find path [options] [expression]`

Common:

- `-name` match file name
- `-type` file type (`f`, `d`)
- `-mtime` modified time
- `-exec` run command on results

Examples:

```
find . -name file1.txt
find . -type d -name dir1
```

```
find . -type f -mtime -7
find . -name file1.txt -exec rm {} \;
```

INODE

Every file/dir has an inode storing metadata.

Contains:

- File type
- Permissions
- Owner + group
- Timestamps (access/modify/change)

- File size
- Disk block pointers
- Hard link count

Inodes identify files; filenames are just links pointing to inodes.

HARD LINK

A hard link points directly to an inode.
Two filenames → same inode, same data.
Created with: `ln file1 file2`

SOFT LINK (SYMBOLIC LINK)

A soft link is a separate file storing a path to another file.

Created with: `ln -s target linkname`

HARD LINK VS SOFT LINK

CREATION

- Hard link → `ln file1 file2`
- Soft link → `ln -s target link`

FILE SYSTEM RELATIONSHIP

- Hard link: points to the same inode; same actual file
- Soft link: separate file containing path to original

EFFECT OF DELETING ORIGINAL

- Hard link: still works; file data remains
- Soft link: breaks; points to nothing

COMPATIBILITY

- Hard link: same filesystem only
- Soft link: works across filesystems/partitions

SPACE USAGE

- Hard link: no extra space (same inode)
- Soft link: needs extra space (stores path)