



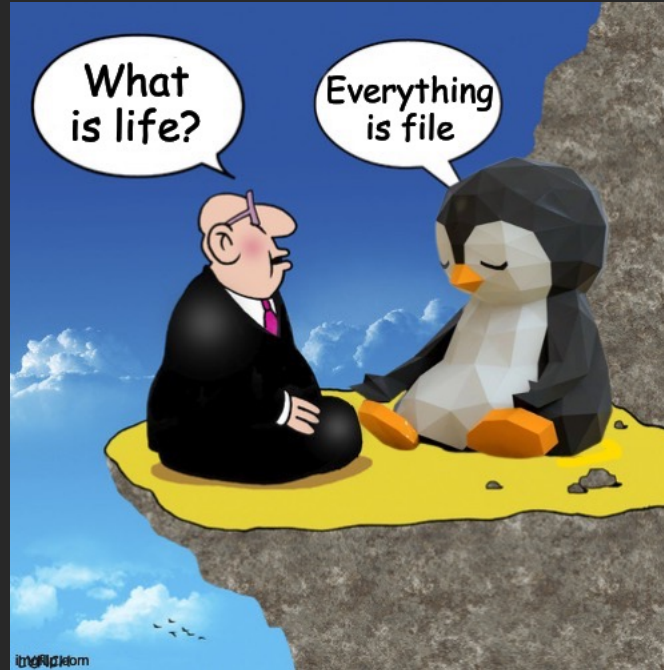
# What is a File?

In Linux, **everything is treated as a file** – ordinary files, directories, devices, sockets, even processes.

A file is a named collection of data stored on disk.

Examples:

- Text files – contain readable text.
- Binary files – contain executable or machine data.
- Directories – special files that list other files.



# Long Format (ls -l) Explained

The long format shows multiple properties of each file:

```
ls -l
```

Example output:

```
-rw-r--r-- 1 student users 4096 Oct 22 10:30 notes.txt
```

Parts of this line:

1. **Type & permissions** – file type and access rights
2. **Links** – number of hard links
3. **Owner** – user who owns the file
4. **Group** – group owning the file
5. **Size** – file size in bytes
6. **Date & time** – last modification
7. **Name** – file name

This view helps you identify files and their properties at a glance.

---

# Home Directory Meaning

Every user has a personal **home directory** – their private workspace.

It's where:

- Your personal files and folders are stored.
- Configuration files (dotfiles) live.
- You usually start when logging in.

Path example:

```
/home/student
```

Shortcut:

```
~
```

`~` always expands to your current user's home directory.

---

## Where am I?

Show your current working directory:

```
pwd
```

Example output:

```
/home/student
```

This tells you your exact location in the filesystem hierarchy.

---



## Human-friendly details

Add `-h` for human-readable sizes (KB, MB, GB):

```
ls -lh
```

Sort by modification time (newest last):

```
ls -lt
```

Reverse order (newest first):

```
ls -ltr
```

List one entry per line:

```
ls -1
```

You can combine flags in one command – for example, `ls -lah`.

---

## Colored output & file types

Many distros colorize `ls` output automatically (directories in blue, executables in green).

Each leading character in `ls -l` shows type:

Symbol	Type
-	regular file
d	directory
l	symbolic link
c	character device
b	block device

---



## Changing directories – cd

Move to another location:

```
cd /etc
```

Return to your home directory:

```
cd
```

Go up one level (parent directory):

```
cd ..
```

Return to the previous working directory:

```
cd -
```

---

## Home directory shortcut

`~` always represents your home directory.

```
cd ~
```

To access subfolders in home, append paths:

```
cd ~/Documents
```

`~user` accesses another user's home (if permitted).

---



# Absolute vs Relative Paths

Absolute paths start from `/` (root).

Relative paths start from your current directory.

Examples:

Type	Example	Meaning
Absolute	<code>/home/student/docs</code>	Always points to the same location
Relative	<code>../docs</code>	Moves relative to where you are

Tip: Use `pwd` before running a command to confirm your location.

---

# Globbering – Wildcards

The shell expands wildcard patterns automatically before running the command.

Pattern	Matches
*	any number of any characters
?	any single character
[abc]	any one of a, b, or c
[0-9]	any digit
[!x]	anything <i>except</i> x

Example:

```
ls *.txt
```

lists all files ending in `.txt` in the current directory.

---

## Brace Expansion {}

Create multiple arguments or names in one command.

```
echo file_{a,b,c}.txt
```

→ expands to `file_a.txt file_b.txt file_c.txt`

Make multiple directories:

```
mkdir project/{src,bin,docs}
```

`{}` saves typing repetitive parts.

---

# Environment Variables

Special variables store information about your shell environment.

Show your home directory path:

```
echo $HOME
```

Show your PATH (where executables are searched):

```
echo $PATH
```

**PATH** is a colon-separated list of directories.

---







Thanks !

 By ElnurBDa