📘 Basic Linux Commands

1️⃣ `pwd` – Print Working Directory

- **Command:** `pwd`
- **Purpose:**
  Displays the **current directory path** you are working in.

Example:

```
pwd
```

Output Example:

```
/home/ec2-user
```

2️⃣ `ls` – List Directory Contents

- **Command:** `ls`
- **Purpose:**
  Lists files and directories in the current directory.

Common Usage:

```
ls
```

- **Useful Options:**
  - `ls -l` → long listing
  - `ls -a` → show hidden files
  - `ls -lh` → human-readable size

3️⃣ `ll` – Long Listing Format

- **Command:** `ll`
  (Alias for `ls -l` in most Linux systems)

- **Purpose:**
  Displays **detailed information** about files and directories.

🔹 Example Output

```
drwxr-xr-x. 2 ec2-user ec2-user 6 Jan 14 10:20 cloud
```

4️⃣ Understanding `ll` Output (7 Fields)

Field No	Field Value	Description
1	`drwxr-xr-x.`	File type & permissions
2	`2`	Number of hard links
3	`ec2-user`	Owner of the file/directory
4	`ec2-user`	Group of the file/directory
5	`6`	Size in bytes
6	`Jan 14 10:20`	Last modification timestamp
7	`cloud`	Name of the file/directory

5️⃣ File Type Indicator (First Character)

The **first character** in the permissions field shows the **file type**:

Symbol	Meaning
`-`	Regular file
`d`	Directory
`l`	Soft (symbolic) link

🔹 Examples

```
-rw-r--r--   → Regular file
drwxr-xr-x   → Directory
lrwxrwxrwx   → Soft link
```

6️⃣ mkdir – Make Directory

Command:
`mkdir`

Purpose:
Creates new directories.

Examples:

```
mkdir ashvini
```

Creates **a directory** with the given name.

```
mkdir ashvini arjun abhi
```

Creates **3 directories** with the given names.

```
mkdir -p test1/test2/test3/test4/test5
```

Creates **nested directories**.
`-p` → parent directories are created automatically if they don't exist.

```
mkdir test{1..10}
```

Creates directories from **test1 to test10**.

7️⃣ rmdir – Remove Empty Directory

Command:
`rmdir`

Purpose:
Deletes **empty directories only**.

Examples:

```
rmdir ashvini
```

Deletes a directory **only if they are empty**.

```
rmdir ashvini arjun abhi
```

Deletes 3 directories **only if they are empty**.

```
rmdir -p test1/test2/test3/test4/test5
```

Deletes nested empty directories using the `-p` (parent) option.

```
rmdir test{1..10}
```

Deletes directories from **test1 to test10** (must be empty).

8️⃣ rm -r – Remove Non-Empty Directories

Command:
`rm -r`

Purpose:
Deletes **files and directories recursively**, even if they are not empty.

Examples:

```
rm -r test1 test2 test3 test4
```

Deletes the listed directories and their contents.

```
rm -r test{1..10}
```

Deletes directories from **test1 to test10** with all contents.

```
rm -r test1?
```

Deletes directories starting with `test1` followed by **one character**
(e.g., `test11`, `test12`).

```
rm -r *
```

⚠️ Deletes **all files and directories** in the current directory.

9️⃣ cd – Change Directory

Command:
`cd`

Purpose:
Used to move between directories.

📌 **Assumption:**
You are working inside:
`/home/ec2-user/ashvini`

Examples:

```
cd test2/subodh/arjun/abhi
```

Moves to **abhi** directory.
`pwd → /home/ec2-user/ashvini/test2/subodh/arjun/abhi`

```
cd ../..
```

Moves **two levels up**.
`pwd → /home/ec2-user/ashvini/test2/subodh`

```
cd ../../test1/ashvini
```

Moves to **ashvini** directory under test1.

```
cd /home/ec2-user/ashvini/test2/subodh/arjun
```

Moves using **absolute path**.

```
cd -
```

Switches to the **previous working directory**.

🔟 man – Manual Pages

Command:
`man`

Purpose:
Displays **manual/documentation** of any Linux command.

Examples:

```
man ls
man pwd
man mkdir
man rmdir
```

1️⃣1️⃣ touch – Create Empty Files

Command:
`touch`

Purpose:
Creates empty files (or updates timestamps if file exists).

Examples:

```
touch file.txt
```

Creates an empty file.

```
touch test1.txt dummy.txt redhat.txt
```

Creates multiple empty files.

```
touch test{1..10}.txt
```

Creates files from **test1.txt to test10.txt**.

1️⃣2️⃣ rm – Remove Files

Command:
`rm`

Purpose:
Deletes files.

Examples:

```
rm file.txt
```

Deletes a single file.

```
rm test1.txt dummy.txt redhat.txt
```

Deletes multiple files.

```
rm test{1..10}.txt
```

Deletes files from **test1.txt to test10.txt**.

1️⃣3️⃣ cat – View / Create / Append File Content

Command:
`cat`

Purpose:
Reads file content or writes/appends content to files.

Examples:

```
cat file.txt
```

Displays file content.

```
cat > file.txt
```

Writes content to file (**overwrites existing content**).

```
cat >> file.txt
```

Appends content to the end of the file.

1️⃣4️⃣ tac – Read File in Reverse

Command:
`tac`

Purpose:
Displays file content **from bottom to top**.

Example:

```
tac file.txt
```

1️⃣5️⃣ cp – Copy Files and Directories

Command:
`cp`

Syntax:

```
cp [source] [destination]
```

🔹 **Copy Files**

📌 **pwd:** `/home/ec2-user/linux_tutorial`

```
cp test1/ashvini/arjun/file.txt test2/abhi/rahul/arnav/
```

Copies file from source to destination.

```
cd test1/ashvini/arjun/
cp test1.txt test{3..4}.txt ../../../test2/abhi/
```

Copies multiple files to another directory.

🔹 **Copy Directories**

```
cp -r test1/ashvini/arjun/ test2/
```

Recursively copies directory **arjun**.

```
mkdir test3
cp -r test1/ashvini/arjun/ test2/abhi/rahul/arnav/ test3/
```

Copies multiple directories into **test3**.

1️⃣6️⃣ mv – Move / Rename Files and Directories

Command:
`mv`

Syntax:

```
mv [source] [destination]
```

🔹 **Move Files**

```
mv test1/ashvini/arjun/file.txt test2/abhi/rahul/arnav/
```

Moves file to destination.

```
cd test1/ashvini/arjun/
mv test1.txt test{3..4}.txt ../../../test2/abhi/
```

Moves multiple files.

🔹 **Move Directories**

```
mv test1/ashvini/arjun/ test2/
```

Moves directory **arjun** to test2.

```
mkdir test3
mv test1/ashvini/arjun/ test2/abhi/rahul/arnav/ test3/
```

Moves multiple directories to **test3**.

1️⃣7️⃣ Absolute Path

Definition:
An **absolute path** is the complete path to a file or directory starting from the root directory `/`.

It always begins with `/`.

📌 **When to Use Absolute Path?**
If there is **less similarity or no similarity** between the current directory and target directory, it is better to use an absolute path.

🔹 Example Scenario

If you want to read `app.txt`, the full path is:

```
cat /home/ec2-user/path_lab/projectA/src/app.txt
```

🔹 Explanation:

Part	Meaning
/	Root directory
home	Home directory folder
ec2-user	User directory
path_lab	Main project folder
projectA	Sub-project
src	Source folder
app.txt	Target file

✅ This is called an **Absolute Path** because it starts from `/`.

📌 Key Points About Absolute Path

- Always starts with `/`
- Independent of current working directory
- Works from anywhere in the system
- Mostly used in scripts, cron jobs, and production environments

---

1️⃣8️⃣ Relative Path

Definition:
A **relative path** is the path to a file or directory relative to your current working directory.

It does **not start with `/`**.

📌 When to Use Relative Path?

If there is **more than two parent-level similarities** between the current directory and target directory, it is better to use a relative path.

🔹 Example Scenario

📌 Current working directory:

```
pwd
```

Output:

```
/home/ec2-user/path_lab/projectB/logs
```

Now you want to read:

```
/home/ec2-user/path_lab/projectA/src/app.txt
```

Instead of using full path, use relative path:

```
cat ../../projectA/src/app.txt
```

🔹 Understanding `../../`

Symbol	Meaning
..	Move one directory up
../..	Move two directories up

From:

```
/home/ec2-user/path_lab/projectB/logs
```

Step-by-step:

1. `..` → `/home/ec2-user/path_lab/projectB`
2. `..` → `/home/ec2-user/path_lab`
3. Then → `projectA/src/app.txt`

📌 Key Points About Relative Path

- Does NOT start with `/`
- Depends on current working directory
- Shorter and easier when working inside project folders
- Commonly used during development

🔎 Absolute Path vs Relative Path

Feature	Absolute Path	Relative Path
Starts With	`/`	Does not start with `/`
Depends on Current Location	❌ No	✅ Yes
Length	Usually longer	Usually shorter
Used In	Scripts, automation	Daily usage
