**🧭 What is a Path in Linux?**

A **path** is the **location of a file or directory** in the Linux file system.  
There are two types:

**1️ Absolute Path 🛤️**

It starts from the **root directory /** and shows the **complete location** of a file or folder, no matter where you are currently.

**📌 Format:**

/parent\_folder/sub\_folder/filename

**✅ Example:**

cd /home/koushik/projects/devops/

* This path **starts from / (root)**.
* It will **always go to the exact location**, no matter where you are.

**2️ Relative Path 🧭**

It is **relative to your current working directory**.  
You don’t start from /, you just give the path based on **where you are now**.

**📌 Format:**

folder\_name/

**✅ Examples:**

* cd logs/ → Go into logs directory inside your current directory
* cd ../ → Go **one level up**
* cd ../../configs → Go up 2 levels, then into configs

**🔍 Key Differences:**

| **Feature** | **Absolute Path** | **Relative Path** |
| --- | --- | --- |
| Starts from | Root / | Current directory (. or no symbol) |
| Always valid | ✅ Yes | ❌ Depends on where you are |
| Begins with / | ✅ Yes | ❌ No |
| Used in scripts | Preferred for accuracy | Used for short paths inside same folder |
| Example | /home/koushik/devops/file.txt | ../devops/file.txt |

**🧑‍💼 🎯 Interview-Ready Answer:**

"An **absolute path** is the full path that starts from the root /, like /home/user/file.txt. A **relative path** is based on the current location, like ../folder/. DevOps engineers use both depending on whether the script needs a fixed or dynamic path."

**👤 1. Normal User (Regular User)**

A normal user is a **non-administrative user** created to perform limited tasks like reading files, running apps, or editing their own files.

**✅ Characteristics:**

* Limited **permissions** (cannot modify system files or configurations).
* Has a **home directory** (e.g., /home/koushik).
* Cannot install packages or restart services without sudo.
* Safer to use for daily tasks (avoids accidental system changes).

**🔍 Example:**

koushik@ubuntu:~$ whoami

koushik

**👑 2. Root User (Superuser)**

The **root user** is the **administrator** of the system.  
They have **full control** over everything — they can install, delete, modify any file, user, or service.

**✅ Characteristics:**

* Full **read/write/execute** permissions.
* Can run **all commands** without restrictions.
* **Home directory** is usually /root
* Dangerous if misused — a wrong command can crash the system.

**🔍 Example:**

root@ubuntu:~# whoami

root

**🔐 Switching Between Users:**

| **Command** | **Description** |
| --- | --- |
| su | Switch user (to root if no username) |
| sudo | Run a command with root privileges |
| sudo su | Become root temporarily |
| exit | Go back to normal user from root |

**🧑‍💼 🎯 Interview-Ready Answer:**

"In Linux, a **normal user** has limited access and can only work in their own space. The **root user**, or superuser, has full control of the system and can make administrative changes. We use sudo to safely perform root tasks without logging in as root directly."

**🛡️ What is sudo in Linux?**

sudo stands for **“Superuser Do”**.  
It allows a **normal user** to run commands with **root (superuser) privileges** without logging in as the root user.

**📋 Syntax:**

sudo [command]

**🧪 Examples:**

| **Command** | **Meaning** |
| --- | --- |
| sudo apt update | Update package list (requires root) |
| sudo reboot | Restart the system |
| sudo mkdir /var/log/test | Create a folder where only root has permission |
| sudo useradd devuser | Add a new user (admin task) |
| sudo rm -rf /foldername | Force delete system folder (be careful ⚠️) |

**⚠️ Best Practices:**

* Only use sudo when needed — it prevents accidental damage.
* Always **double-check commands** before running with sudo.
* Use sudo -k to clear the sudo timestamp (forces password next time).

Great! Let’s understand sudo su clearly:

**🔄 What does sudo su mean?**

✅ It combines two powerful commands:

* **sudo** – Run a command with superuser privileges
* **su (substitute user)** – Switch to another user (default is root)

**💡 So, sudo su means:**

👉 “Run the su command as **root** using sudo permissions,”  
which effectively **switches you to the root user shell**.

**📌 When do you use it?**

* When you need to become the **root user** to perform multiple admin tasks
* Useful for **interactive sessions** where you're running several commands as root

**⚠️ Example:**

sudo su

🧑‍💻 Output: Your prompt changes to # instead of $, indicating **you are now root**.

Now you can run any command **without typing sudo again** for each one.

**🚪 To Exit Root:**

exit

**🔐 Tip for DevOps/Best Practice:**

Using sudo su gives **full root access**, which is **not always recommended**.  
It's safer to use sudo for **individual commands** unless absolutely needed.

**🔍 What does sudo su - do?**

sudo su - means:

**"Use sudo to run su as root, and start a login shell for the root user."**

So, it:

* Switches to the **root user** (su)
* Loads the **root user's environment variables**, path, and settings (-)

**✅ Difference between sudo su and sudo su -:**

| **Command** | **What it does** |
| --- | --- |
| sudo su | Switches to root **without** loading root's environment |
| sudo su - | Switches to root **and loads** root’s full login environment |

**🧪 Example:**

sudo su -

You'll now be:

* In root’s home directory (/root)
* Using root’s environment and PATH
* Seeing # in your shell prompt

**🔐 Why use sudo su -?**

✅ Best when:

* You’re doing **administrative tasks** that rely on **root's environment**
* You want to **simulate logging in directly as root**

**🚪 To exit:**

exit

**📁 What is** ls **Command?**

ls stands for **“list”**.  
It is used to **list files and directories** in the current directory or a specified path.

**📌 Basic Syntax:**

ls [options] [directory or file path]

**🔍 Basic Examples:**

| **Command** | **Description** |
| --- | --- |
| ls | Lists files/folders in current directory |
| ls /home/koushik | Lists contents of that path |
| ls Documents | Lists files inside Documents folder |

**✅ Most Commonly Used ls Options:**

| **Option** | **Meaning** |
| --- | --- |
| -l | Long listing format (permissions, owner, date, size) |
| -a | Shows all files including hidden (. files) |
| -h | Human readable size (used with -l) like KB, MB |
| -R | Recursively lists subdirectories too |
| -t | Sort by modification time (latest first) |
| -r | Reverse order of sorting |
| -S | Sort by file size |
| -i | Show inode number of files |
| -d | List only the directory name, not contents |

**🧪 Examples of Combined Options:**

ls -l # Detailed info with size, date, owner, etc.

ls -la # Includes hidden files

ls -lh # Human-readable sizes

ls -ltr # Sorted by time (oldest first)

ls -lhS # Sort by size (largest to smallest)

**📦 Real-Time DevOps Usage:**

| **Use Case** | **Command** |
| --- | --- |
| Check logs folder size/info | ls -lh /var/log |
| View hidden config files | ls -a ~ |
| Explore recursive app folders | ls -R /opt/tomcat |
| See file modified recently | ls -lt /etc/nginx/ |

**📌 What is touch Command?**

touch is used to **create empty files** or **update the timestamp** (access and modification time) of existing files.

**🧠 Basic Syntax:**

touch [options] filename

**✅ Common Use Cases:**

| **Command** | **Description** |
| --- | --- |
| touch file.txt | Creates a new empty file named file.txt |
| touch a.txt b.txt | Creates multiple files at once |
| touch log.txt | If file exists, updates its timestamp |

**🧪 Real-Time Examples:**

touch hello.txt # Creates 'hello.txt'

touch devops1.txt devops2.txt # Creates two files

touch /tmp/mylog.txt # Creates file in another folder

**🔁 Timestamp Update:**

If the file **already exists**, touch updates:

* **Access time (atime)** – last read time
* **Modification time (mtime)** – last content change

**⚙️ Useful Options:**

| **Option** | **Description** |
| --- | --- |
| -c | Do not create a file if it doesn’t exist |
| -t | Set a specific timestamp (YYYYMMDDhhmm) |
| -r file1 | Use timestamp of another file |
| -a | Update only access time |
| -m | Update only modification time |

**🔧 Examples:**

touch -c test.txt # Only update if exists, else do nothing

touch -t 202504290915 my.txt # Set specific date/time

touch -r old.txt new.txt # Set new.txt's time same as old.txt

**🔐 Real-time DevOps Use Cases:**

| **Scenario** | **Command** |
| --- | --- |
| Creating test or log files | touch /var/log/test.log |
| Timestamp syncing for deployment | touch -r lastbuild.txt app.js |
| Script creating temp files | touch tempfile.txt |

**🐱 What is cat Command?**

cat stands for **“concatenate”**.  
It is used to **display the contents of a file**, **create a file**, or **combine multiple files** into one.

**📌 Basic Syntax:**

cat [options] [filename]

**✅ Common Use Cases:**

| **Command** | **Description** |
| --- | --- |
| cat file.txt | Shows contents of file.txt |
| cat > newfile.txt | Creates a new file, takes input from user |
| cat file1 file2 | Displays contents of both files |
| cat file1 > file2 | Copies content from file1 to file2 |
| cat file1 >> file2 | Appends content of file1 to file2 |

**✍️ Example: Create a file**

cat > demo.txt

Then type your content and press Ctrl + D to save.

**👓 Example: View a file**

cat demo.txt

**➕ Example: Append to an existing file**

cat >> demo.txt

**🔍 Useful Options:**

| **Option** | **Description** |
| --- | --- |
| -n | Show line numbers |
| -b | Show line numbers (skip blank lines) |
| -E | Show $ at end of each line |
| -s | Remove extra blank lines |
| -T | Show tab characters as ^I |

**💡 Examples with Options:**

cat -n file.txt # Show contents with line numbers

cat -b file.txt # Skip blank lines while numbering

cat -E file.txt # See line ends with $

**💼 Real-Time DevOps Usage:**

| **Task** | **Command** |
| --- | --- |
| View logs quickly | cat /var/log/syslog |
| Combine two config files | cat a.conf b.conf > final.conf |
| Check file created by script | cat output.txt |

**📁 What is mkdir?**

mkdir stands for **"make directory"**.  
It is used to **create new directories (folders)** in Linux.

**📌 Basic Syntax:**

mkdir [options] directory\_name

**✅ Common Use Cases:**

| **Command** | **Description** |
| --- | --- |
| mkdir devops | Creates a directory named devops |
| mkdir test1 test2 | Creates multiple folders at once |
| mkdir /tmp/logs | Creates logs folder in /tmp |

**⚙️ Useful Options:**

| **Option** | **Description** |
| --- | --- |
| -p | Create parent directories if they don't exist |
| -v | Show message for each directory created (verbose) |
| -m | Set permissions while creating directory (mode) |

**🧪 Real-Time Examples:**

mkdir projects # Creates 'projects' folder

mkdir dev logs backups # Creates 3 folders

mkdir -p app/logs/error # Creates full path, even if intermediate folders missing

mkdir -v newfolder # Shows message: mkdir: created directory 'newfolder'

mkdir -m 755 newdir # Creates directory with permission 755

**💼 Real-Time DevOps Usage:**

| **Task** | **Command** |
| --- | --- |
| Create log folders for app | mkdir -p /var/logs/myapp/errors |
| Organize backups or deployments | mkdir /backup/2025-April |
| Script-based directory creation | mkdir -p $HOME/scripts/output |

**🔐 Notes:**

* If the folder already exists without -p, it shows an error.
* mkdir needs **write permissions** in the parent directory.

**🗑️ What is rmdir?**

rmdir stands for **"remove directory"**.  
It is used to **delete empty directories only** in Linux.

**📌 Basic Syntax:**

rmdir [options] directory\_name

**⚠️ Important Note:**

* rmdir **only removes empty folders**.
* If the directory has files or subfolders, it **won’t delete it**.

**✅ Common Use Cases:**

| **Command** | **Description** |
| --- | --- |
| rmdir demo | Deletes the folder demo if it's empty |
| rmdir test1 test2 | Deletes multiple empty folders |
| rmdir -p a/b/c | Removes directory c, then b, then a if all are empty |

**🧪 Examples:**

mkdir testdir

rmdir testdir # Works as it's empty

mkdir -p project/code

rmdir -p project/code # Removes 'code', then 'project' if both are empty

**🔥 Real-Time DevOps Usage:**

| **Use Case** | **Command** |
| --- | --- |
| Remove temp empty folders | rmdir /tmp/oldbuild/emptyfolder |
| Clean up unused empty dirs | rmdir -p /opt/tools/old/utils |

**🗑️ What is rm?**

rm stands for **“remove”**.  
It is used to **delete files and directories** in Linux.

**📌 Basic Syntax:**

rm [options] file\_or\_directory

**✅ Common Use Cases:**

| **Command** | **Description** |
| --- | --- |
| rm file.txt | Deletes the file named file.txt |
| rm -r folder | Deletes a folder **and its contents** |
| rm -f file.txt | Force delete without confirmation |
| rm -rf folder/ | Forcefully deletes folder + all inside |

**⚠️ Be Careful!**

* **rm deletes permanently** — no Recycle Bin!
* Especially with -rf, there is **no warning**.

**🔧 Useful Options:**

| **Option** | **Description** |
| --- | --- |
| -f | Force delete (no prompt) |
| -i | Ask before every delete (interactive mode) |
| -r | Recursively delete directories and contents |
| -v | Verbose mode (shows what is being deleted) |

**🧪 Examples:**

rm report.txt # Deletes a file

rm -i config.json # Asks before deleting

rm -r logs/ # Deletes directory and all inside

rm -rf /tmp/project # Deletes without any prompt (DANGEROUS!)

rm -rv myfolder/ # Verbose recursive delete

**💼 Real-Time DevOps Usage:**

| **Scenario** | **Command** |
| --- | --- |
| Remove old logs | rm -rf /var/log/myapp/old/ |
| Delete temporary build folders | rm -rf /tmp/build/ |
| Clean unused config files | rm -i \*.conf |

**🧠 Bonus Tip: Safe Alternative**

Use trash-cli tool in Linux if you want a Recycle Bin-like feature (trash-put, trash-list).

**🧾 What is CRUD?**

**CRUD** stands for:  
**C**reate, **R**ead, **U**pdate, and **D**elete.

It refers to the **four basic operations** you can do on **data** in a database or any storage system (like files, APIs, or cloud services).

**🔠 CRUD Operations Explained:**

| **Operation** | **Action** | **Example in Real Life** |
| --- | --- | --- |
| **Create** | Add new data | Add a new user to a website |
| **Read** | View existing data | View a user's profile |
| **Update** | Change existing data | Change a user's email address |
| **Delete** | Remove existing data | Delete a user from the system |