**📘 What are File Permissions and Ownership in Linux?**

Every file and directory in Linux has:

1. **Owner** – The user who created the file.
2. **Group** – A set of users who can also access the file.
3. **Permissions** – Define what the owner, group, and others can do with the file.

**📂 File Ownership**

Each file or folder has **3 types of ownership**:

| **Ownership** | **Description** |
| --- | --- |
| **User (u)** | The **owner** of the file (creator). |
| **Group (g)** | A group of users with shared access. |
| **Others (o)** | All other users on the system. |

Use this command to check:

ls -l filename

Example:

-rw-r--r-- 1 koushik devs 1340 May 2 file.txt

* **koushik** = owner
* **devs** = group

**🔐 File Permissions**

There are 3 types of permissions:

| **Symbol** | **Permission** | **What it Allows** |
| --- | --- | --- |
| **r** | Read | View file content |
| **w** | Write | Edit/modify the file |
| **x** | Execute | Run the file (for scripts or binaries) |

Each permission applies to **user**, **group**, and **others**.

**📊 Permission Format (Example Explained)**

-rwxr-xr--

Breakdown:

| **Part** | **Meaning** |
| --- | --- |
| - | File type (- = file, d = directory) |
| rwx | User (owner) permissions: read, write, execute |
| r-x | Group permissions: read, execute |
| r-- | Others: read only |

**📘 chown – Full Detailed Explanation**

**🔑 Definition:**

chown stands for **"change ownership"**.  
It is used to **change the owner and/or group** of a file or directory.

**📚 Syntax:**

chown [OPTIONS] NEW\_OWNER[:NEW\_GROUP] FILE

**📌 Key Points:**

* **NEW\_OWNER** → the user who will become the new owner.
* **NEW\_GROUP** → the group that will be associated (optional).
* If only owner is specified: chown koushik file.txt
* If owner and group are specified: chown koushik:devops file.txt
* You need **sudo/root** privileges to run this command.

**🧪 Examples:**

**1. ✅ Change file owner:**

sudo chown ram file.txt

Now **ram** becomes the owner of file.txt.

**2. ✅ Change file owner and group:**

sudo chown ram:devops file.txt

Owner → ram, Group → devops

**3. ✅ Change only group:**

sudo chown :devops file.txt

Owner remains unchanged, group changes to devops.

**4. ✅ Change ownership of a directory:**

sudo chown koushik:devs mydir

**5. ✅ Recursive change (apply to all inside folder):**

sudo chown -R koushik:devs /var/www/

|  |
| --- |
| -R Recursively change ownership of directory and all files inside |

-v Verbose output (shows what changed)

**🔐 Why Use chown in Real Time?**

* To fix **permission issues** after copying files.
* To ensure files created by a **service or script** belong to the correct user.
* During **DevOps deployments**, you might change ownership to www-data, nginx, or jenkins users.

**📌 For Interview:**

"chown is a Linux command used to change the ownership of files or directories. We can assign a new user or group, or both, and use -R to apply recursively for directories."

**📘 What is chmod?**

chmod stands for **"change mode"**.  
It is used to **change the permissions** (read, write, execute) of a file or directory for **owner, group, and others**.

**📚 Syntax:**

chmod [OPTIONS] PERMISSIONS FILE

**🔑 File Permission Basics:**

There are **3 types of permissions**:

| **Symbol** | **Meaning** | **Value** |
| --- | --- | --- |
| r | Read | 4 |
| w | Write | 2 |
| x | Execute | 1 |

There are **3 levels of users**:

| **User** | **Symbol** |
| --- | --- |
| Owner | u |
| Group | g |
| Others | o |
| All | a |

**🧪 Two Ways to Use chmod:**

**✅ 1. Symbolic Mode (using letters)**

**1. 🔼 Add Permissions (+)**

chmod u+x file.txt # Add execute for user

chmod g+w file.txt # Add write for group

chmod o+r file.txt # Add read for others

chmod a+x file.txt # Add execute for all (user, group, others)

**2. 🔽 Remove Permissions (-)**

chmod u-w file.txt # Remove write from user

chmod g-x file.txt # Remove execute from group

chmod o-r file.txt # Remove read from others

chmod a-w file.txt # Remove write from all

**3. 🟰 Set Exact Permissions (=)**

This **removes all existing** permissions and **sets only given** ones:

chmod u=r file.txt # User can only read

chmod g=rw file.txt # Group can read and write only

chmod o= file.txt # Others have no permission

chmod a=r file.txt # Everyone can only read

**✅ Combined Examples:**

chmod ug+r file.txt # Add read for user and group

chmod u+x,g-w file.txt # Add execute to user, remove write from group

chmod u=rw,g=r,o= file.txt # Set multiple exact permissions

**✅ 2. Numeric Mode (using numbers)**

Each permission has a value:

* r = 4, w = 2, x = 1

You add values for each user type:

chmod 755 file.sh

Means:

* Owner → 7 → rwx → 4+2+1
* Group → 5 → r-x → 4+0+1
* Others → 5 → r-x

More examples:

chmod 777 file.txt # All full access

chmod 644 file.txt # Owner: rw-, Group: r--, Others: r--

chmod 700 file.sh # Only owner can read, write, execute

**🔐 1. What is Authentication?**

Authentication is the process of **verifying identity**. In Linux or cloud systems (like AWS), when we connect to a server (using SSH), the system checks whether we are **allowed** or not.

**🔑 2. Types of SSH Authentication**

| **Type** | **Description** |
| --- | --- |
| **Password-based** | You enter a password to access the server |
| **Key-based (SSH Key)** | You use a cryptographic key pair (private/public) to log in |

**🔐 Password-based Authentication**

**✅ How it works:**

* You run ssh user@server
* Server asks: “What is your password?”
* You type the **user password**.
* If correct → access granted.

**✅ Advantages:**

* Simple and quick to set up
* No need to generate SSH keys

**❌ Disadvantages:**

* Less secure (passwords can be guessed, stolen, or brute-forced)
* Not scalable for many users or servers
* Cannot be automated easily

**📄 Location:**

/etc/ssh/sshd\_config

✅ This is the **SSH daemon (server) configuration file** — it controls how SSH behaves on the server side.

**🔧 1. Enable or Disable Password Authentication**

**✅ To Allow Password Login:**

PasswordAuthentication yes

**❌ To Disable Password Login (for key-based only):**

PasswordAuthentication no

🔒 Recommended for secure environments using only SSH keys.

sshd -t (for testing)

**Restart the SSH service:**

sudo systemctl restart sshd

Or:

sudo service ssh restart

**✅ You can now log in using passwords!**

**🔑 Key-based Authentication**

**✅ How it works**

**✅ Step-by-Step: Where to Paste the Public Key**

**🔑 Public key file example:**

If you generated a key pair using:

ssh-keygen -f mykey

You will get:

* mykey → private key (keep secure)
* mykey.pub → public key (you will paste this)

**User should send his public key to admin to give access.**

**📍 Where to paste the .pub key?**

✅ Paste the contents of your mykey.pub into the following file on the **remote server**:

/home/username/.ssh/authorized\_keys

**🔧 Steps:**

**1. Connect to the remote server (if you still have password access):**

ssh username@remote-server

**2. Create the .ssh directory (if not already exists):**

mkdir -p .ssh/authorized\_keys

chmod 700 .ssh

**3. Paste your public key into authorized\_keys:**

nano .ssh/authorized\_keys

# OR

vi .ssh/authorized\_keys

📌 Paste the **entire contents** of mykey.pub into the file.

**4. Set proper permissions:**

chmod 600 .ssh/authorized\_keys

Sure, Koushik! To **enable password-based SSH login**, follow these steps:

**✅ Step-by-Step: Change PasswordAuthentication no to yes**

**1. Open the SSH configuration file:**

sudo nano /etc/ssh/sshd\_config

Or use vi if you prefer:

sudo vi /etc/ssh/sshd\_config

**2. Find this line:**

PasswordAuthentication no

🔁 Change it to:

PasswordAuthentication yes

👉 If the line is commented out (starts with #), remove the # too.

sshd -t (for testing)

**Restart the SSH service:**

sudo systemctl restart sshd

Or:

sudo service ssh restart

**✅ Then, from your local machine:**

Use your private key to log in:

ssh -i mykey username@remote-server

**🔒 Important Points:**

| **File** | **Purpose** | **Permission** |
| --- | --- | --- |
| ~/.ssh/ | SSH config dir | 700 |
| ~/.ssh/authorized\_keys | Stores allowed public keys | 600 |

**🔧 Bonus: Copy key automatically (if ssh-copy-id available):**

ssh-copy-id -i ~/.ssh/mykey.pub username@remote-server

**✅ Advantages:**

* More secure than passwords
* Can disable password login completely
* Better for automation (scripts, DevOps tools, CI/CD)
* Works with **Git**, **AWS EC2**, **Ansible**, etc.

**❌ Disadvantages:**

* Need to manage key files
* Private key must be protected carefully
* A bit technical for beginners

**🛡️ Summary Table:**

| **Feature** | **Password Auth** | **Key-Based Auth** |
| --- | --- | --- |
| Setup Simplicity | Easy | Slightly complex |
| Security | Weaker | Stronger |
| Automation Friendly | ❌ No | ✅ Yes |
| Multi-Server Usage | Tedious | Scalable |
| Risk | Password leak | Key theft if unprotected |
| SSH Command | ssh user@ip | ssh -i key user@ip |

**🧠 Interview Tip:**

"**Password-based authentication** uses a user password to log in, but is less secure and harder to scale. **Key-based authentication** uses a private/public key pair, offering better security, automation support, and is commonly used in DevOps and cloud environments."

To give a user **sudo (admin) access** using the /etc/sudoers file, you can add them under the **wheel group** or **directly as a sudo user**.

Here’s how to do it **properly and safely**.

**✅ Option 1: Add the User to the wheel Group (Recommended on RHEL/CentOS/Fedora)**

**Step 1: Add the user to the wheel group**

sudo usermod -aG wheel username

**Step 2: Make sure this line is enabled in /etc/sudoers:**

Open the file using **visudo** (safe editor for sudoers):

sudo visudo

Make sure this line is **uncommented**:

%wheel ALL=(ALL) ALL

✅ Now any user in the wheel group can use sudo!

**✅ Option 2: Add the User Directly to /etc/sudoers (Not recommended, but possible)**

Open with visudo:

sudo visudo

Then add a line at the end:

username ALL=(ALL) ALL

This gives the user username full sudo privileges.

**⚠️ Important Notes:**

* **Always use visudo** to edit /etc/sudoers. It checks for syntax errors before saving.
* Wrong syntax can **lock you out** of root/sudo access!
* **wheel group** is the safest and standard way on RHEL-based systems.

**✅ What is Package Management?**

**Package Management** in Linux is the process of installing, upgrading, configuring, and removing software **packages** (applications, libraries, tools) on your system.

**🔧 What is a Package?**

A **package** is a compressed archive file that contains:

* The application or tool itself
* Metadata (version, dependencies, etc.)
* Scripts to install and configure it

**🎯 Why is Package Management important?**

As a DevOps engineer, package management helps you:

* Automate software installation (e.g., via Ansible, Dockerfiles)
* Ensure version control and consistency across servers
* Resolve dependencies automatically

**🧰 Types of Package Managers:**

**1. Debian-based (Ubuntu, Kali, etc.)**

* **.deb files**
* Tools:
  + apt (Advanced Package Tool)
  + dpkg

**2. Red Hat-based (CentOS, RHEL, Fedora, Amazon Linux)**

* **.rpm files**
* Tools:
  + yum (Yellowdog Updater, Modified)
  + dnf (New version of yum)
  + rpm

**💡 Most Common Commands:**

**🔸 APT (Debian/Ubuntu):**

| **Task** | **Command** |
| --- | --- |
| Install a package | sudo apt install <package> |
| Update package list | sudo apt update |
| Upgrade packages | sudo apt upgrade |
| Remove a package | sudo apt remove <package> |
| Search package | apt search <name> |
| Show package info | apt show <name> |

**🔹 YUM / DNF (RHEL/CentOS):**

| **Task** | **Command** |
| --- | --- |
| Install a package | sudo yum install <package> or sudo dnf install <package> |
| Update all packages | sudo yum update |
| Remove a package | sudo yum remove <package> |
| List all packages | yum list installed |
| Search package | yum search <name> |
| Package info | yum info <name> |
| Available packages | yum list available |

**✅ Process of Installing a Package in Linux**

Let’s break it down into **clear steps**:

**🔹 Step 1: Know Your Linux Distribution**

Check if your system is:

* **Debian-based** (like Ubuntu, Kali, Linux Mint)
* **RHEL-based** (like CentOS, Red Hat, Fedora, Amazon Linux)

Command to check:

cat /etc/os-release

**🔹 Step 2: Update Your Package Repository**

Before installing, always update the local list of available packages.

**For Debian-based:**

sudo apt update

**For RHEL-based:**

sudo yum update # OR

sudo dnf update

**🔹 Step 3: Search for the Package (Optional)**

To make sure the package exists in the repository.

**APT:**

apt search <package-name>

**YUM:**

yum search <package-name>

**🔹 Step 4: Install the Package**

**For Debian/Ubuntu (APT):**

sudo apt install <package-name>

**For RHEL/CentOS (YUM or DNF):**

sudo yum install <package-name>

# or

sudo dnf install <package-name>

Example:

sudo apt install nginx

sudo yum install git

**🔹 Step 5: Verify Installation**

You can check if the package was installed:

<package-name> --version

Or check with:

which <package-name>

Example:

git --version

which nginx

**🔹 Step 6: (Optional) Enable and Start the Service**

If it’s a service like nginx, mysql, etc.:

sudo systemctl enable nginx

sudo systemctl start nginx

**🔹 Step 7: (Optional) Check Status**

sudo systemctl status nginx

**🔐 Note:**

* Some packages may need **dependencies** – package managers automatically install them.
* Use sudo for admin rights.
* Always update before installation to avoid broken packages.

**What is /etc/yum.repos.d/?**

It is a **directory** where all the **YUM repository configuration files** (.repo files) are stored.

Each .repo file tells YUM **where to find and download packages** (URLs or mirrors).

**📁 Path:**

/etc/yum.repos.d/

Inside this folder, you'll see files like:

* CentOS-Base.repo
* epel.repo
* remi.repo

Each file contains information about one or more **repositories**.

**What is Service Management in Linux?**

In Linux, **services** are background processes (also called daemons) that start during boot and run continuously—for example:  
🔸 nginx, httpd, sshd, mysql, etc.

Service management refers to **starting, stopping, enabling, disabling, and checking the status** of these services.

**🔧 Tools Used:**

Modern Linux distros (like RHEL 7+/CentOS 7+/Ubuntu 16+ etc.) use:

**systemctl (Systemd service manager)**

**✅ 1. sudo systemctl start <service-name>**

Starts the service **temporarily** (until next reboot).

🧪 Example:

sudo systemctl start nginx

**✅ 2. sudo systemctl stop <service-name>**

Stops the running service.

🧪 Example:

sudo systemctl stop nginx

**✅ 3. sudo systemctl restart <service-name>**

Stops and starts the service again.  
Use when you make changes in configuration files.

🧪 Example:

sudo systemctl restart sshd

**✅ 4. sudo systemctl reload <service-name>**

Reloads the configuration **without fully restarting** the service.  
(Not supported by all services.)

🧪 Example:

sudo systemctl reload apache2

**✅ 5. sudo systemctl status <service-name>**

Shows current status: running, stopped, failed, etc.

🧪 Example:

sudo systemctl status nginx

**✅ 6. sudo systemctl enable <service-name>**

Automatically starts the service **at boot time**.

🧪 Example:

sudo systemctl enable mysql

**✅ 7. sudo systemctl disable <service-name>**

Prevents the service from starting at boot.

🧪 Example:

sudo systemctl disable mysql

**✅ 8. sudo systemctl is-enabled <service-name>**

Checks if the service will start at boot.

🧪 Example:

sudo systemctl is-enabled sshd

**✅ 9. systemctl list-units --type=service**

Lists all currently loaded **services** and their states.

**✅ 10. journalctl -u <service-name>**

Shows logs/output of the service.

🧪 Example:

journalctl -u nginx

**✅ df Command in Linux – Disk Filesystem Usage (Detailed & Simple)**

The df (disk free) command is used to **check disk space usage** of file systems in Linux.

**📌 Definition:**

The df command reports the amount of **used and available disk space** on Linux file systems, including mount points.

**🧪 Basic Syntax:**

df [OPTION]... [FILE]...

**🔍 Sample Output:**

$ df -h

Filesystem Size Used Avail Use% Mounted on

/dev/sda1 50G 20G 28G 42% /

tmpfs 2.0G 0 2.0G 0% /dev/shm

**🔧 Important Columns Explained:**

| **Column** | **Meaning** |
| --- | --- |
| Filesystem | The name of the mounted file system (e.g., /dev/sda1) |
| Size | Total size of the file system |
| Used | How much space is used |
| Avail | How much space is available |
| Use% | Percentage of space used |
| Mounted on | Where the file system is mounted (directory path) |

**🛠️ Most Used Options:**

| **Option** | **Description** |
| --- | --- |
| -h | Human-readable format (e.g., GB, MB) |
| -T | Shows the **type of file system** |
| -a | Includes pseudo, duplicate, and inaccessible file systems |
| -i | Displays inode usage instead of block usage |
| --total | Adds a total at the end |

🧪 Examples:

df -h # Human readable sizes

df -T # Show filesystem type (ext4, xfs, etc.)

df -i # Show inode usage

df --total # Total usage summary

**✅ du Command in Linux — Full & Simple Explanation**

The du (disk usage) command is used to **check the size of directories and files** in Linux.

**📌 Definition:**

The du command **estimates** the **space used** by files and directories on the disk.

**🧪 Basic Syntax:**

du [OPTIONS] [FILE or DIRECTORY]

**🔍 Simple Example:**

du -h /home/koushik

This shows the disk usage of /home/koushik in human-readable format (KB, MB, GB).

**🔧 Important Options:**

| **Option** | **Description** |
| --- | --- |
| -h | Human-readable sizes (KB, MB, GB) |
| -s | Summary: Shows only the **total size** of each argument |
| -a | Shows size of **each file and directory** |
| -d N | Shows depth level up to N subdirectory levels |
| -c | Adds a **grand total** at the end |
| --max-depth=N | Limit report to N levels of subdirectories |

**📦 Examples:**

du -h # Size of current directory and its subdirectories

du -sh \* # Size of each file/folder in current directory

du -ah # Show size of all files and directories

du -sh /var/log # Total size of /var/log directory

du -h --max-depth=1 # Only one level depth shown

du -hc /etc /var/log # Show sizes and a grand total

**🧠 Key Differences Between df and du:**

| **Feature** | **df** | **du** |
| --- | --- | --- |
| Reports | Free and used space on disk | Space used by files/directories |
| Source | Filesystem level | File and directory level |
| Real-Time | May show slight delay | Checks actual file data |