1. Basic Linux Commands

Q. What are some essential Linux commands every beginner should know?

➔ **ls , cd , pwd , mkdir , touch cat , cp , mv [rename and move file] , rm , rmdir , head and tail**

Q. How do you check the current working directory?

➔ **pwd** command [present working directory]

Q. How do you list all files, including hidden ones, in a directory?

➔ **ls -a** simply lists all files (including hidden ones) without any details.

ls -lart lists all files (including hidden ones) with detailed information, sorted by modification time in reverse order.

Q. What is the difference between ls and ls -l?

➔ **ls** gives a simple listing of file and directory names.

ls -l provides a detailed view with additional information about each file or directory.

Q. How do you create, delete, copy, and move a file in Linux?

➔ cmd : **touch, cat, nano, echo** ------ for create a file

Eg. Touch filename

Cmd **: rm , rmdir** ----------for delete a file or directory

Eg. rm filename , rmdir directory name

Cmd **: cp** ---------- copy the file

Eg. cp filename1 filename2 (copy filename1 into filename2)

Cmd : **mv**---------- for move the file

Eg. mv filename1 directory name (move filename1 into directory name)

2. File and Directory Management

Q. What does the touch command do?

⇨ Cmd **: Touch** command use to create a file

Q. How do you rename a file in Linux?

⇨ Cmd : **mv** command is used to rename a file.

Q. What is the command to count the number of lines, words, and characters in a file?

⇨ Cmd : **wc command is used to count the line , words and characters in a file.**

Example :-

wc -l myfile.txt ---------- # Counts only lines

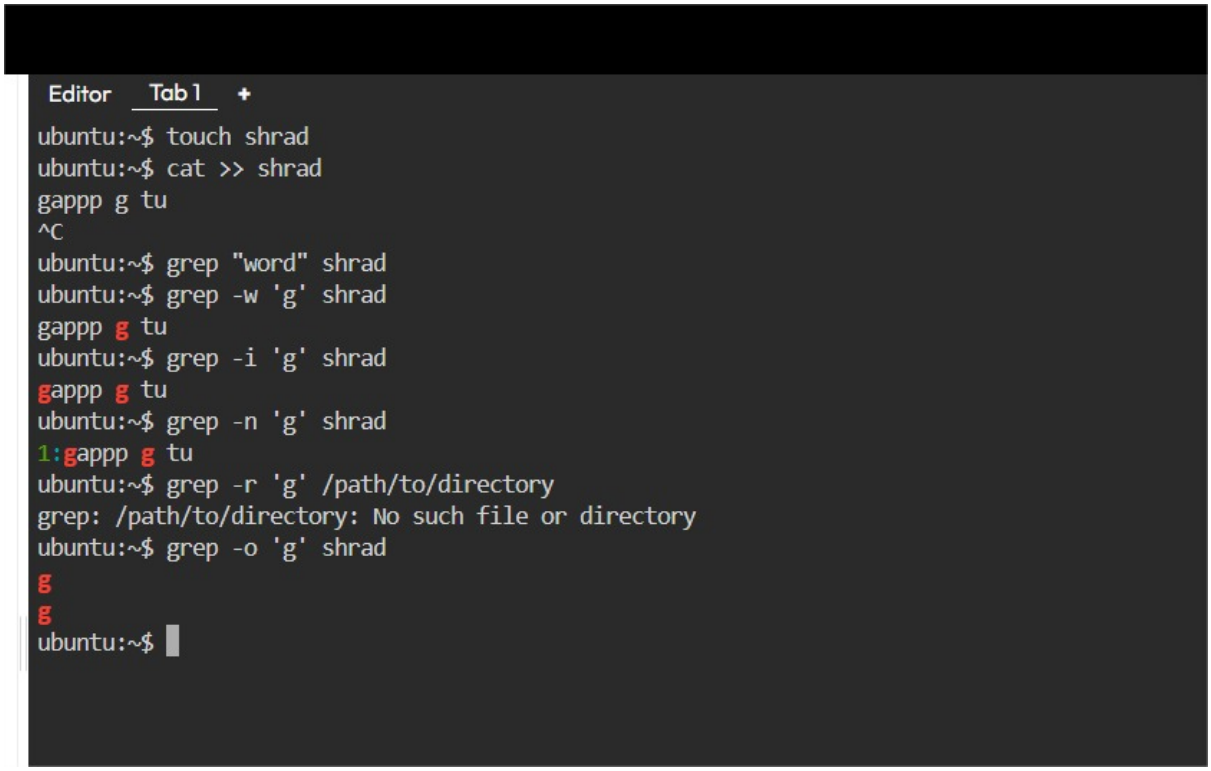
wc -w myfile.txt ---------- # Counts only words

wc -c , wc -m myfile.txt ---------- # Counts only characters

Q. How do you find a specific word inside a file?

⇨ Cmd – **grep "word" filename**

Syntax – **grep -w ' character name ' filename -------- (i- is a character)**



grep -w ‘ g ’ filename = shows only single character which is ‘ g ’ , as shown in img.

grep -i ‘ g ’ filename = it shows every the specific character present in every word in the sentence , as shown in img.

grep -n ‘ g ’ filename = it gives no. for a lines , as shown in img.

Q. Explain the difference between absolute and relative paths.

Aspect	**Absolute Path**	**Relative Path**
Definition	Specifies the full path from the root directory.	Specifies the path relative to the current working directory.
Starts with	/ (root directory)	The current working directory (does not start with /)
Example	/home/user/documents/file.txt	documents/file.txt (if in /home/user/)
Unambiguous	Yes, always points to the same location.	No, depends on your current directory.

Aspect	Absolute Path	Relative Path
Usage	When the full path is needed, independent of where you are in the filesystem.	When you want to refer to a file or directory relative to your current location.
Traversal	Always starts from the root and follows the entire hierarchy.	Starts from the current working directory and moves through relative directories.
Efficiency	Can be longer since it includes the entire path.	Can be shorter if you're working within a specific directory structure.

Example:

- **Absolute Path**: /home/user/documents/file.txt (full path starting from root /)
- **Relative Path**: documents/file.txt (relative to the current directory, e.g., /home/user/)

3. User and Permission Management

Q. How do you check the current logged-in user?

➔ Cmd : **whoami** command is used for current logged-in user.

- ⇨ **`whoami` :** Shows the current logged-in username.
- ⇨ **`who :`** Lists all users currently logged in.
- ⇨ **`id :`** Displays detailed user information.
- ⇨ **`logname`** : Displays the login name of the user who started the session.

Q. What is the difference between su and sudo?

Aspect	su (Substitute User)	sudo (Superuser Do)
Purpose	Switches to another user (commonly root) for the entire session.	Executes a command with root privileges without switching users.
Command Syntax	su [username] or just su (for root)	sudo [command]
Password	Requires the password of the user you are switching to (root, by default).	Requires the password of the user executing the command.
User Switching	Switches to the target user, typically root, for the entire session.	Does not switch users, just runs the command as root.
Session Duration	You stay logged in as the target user until you exit.	Root privileges are granted only for the specific command executed.
Scope of Access	Grants full access to the system as the target user (root).	Grants temporary root access for a single command.
Security	Less secure, as it gives persistent access to the target user.	More secure, as it limits root access to a single command.
Command Logging	Commands executed are not typically logged (unless configured).	Commands executed via sudo are logged for auditing purposes.
Use Case	When you need full root access for an entire session.	When you need to run specific commands with root privileges.
Typical Output	Changes the shell to reflect the new user.	Executes the command and returns control to the regular user.

Example:

- **Using su**:
- su

  (Switches to the root user, and you remain root until exit.)

- **Using sudo**:
- sudo apt-get update

  (Executes the apt-get update command as root and then returns to your normal user prompt.)

  Summary:

- **su**: Switches users (typically to root) for the duration of the session.
- **sudo**: Grants root privileges to execute individual commands without switching

Q. How to check the list of created users ?

➔ **ls /home**

Q. How do you change file permissions using chmod?

➔ **Chmod** command is used to change the permission .

you can use either symbolic notation or numeric (octal) notation.

Using Symbolic Notation

In symbolic notation, you specify the permissions using letters:

- **r** for read
- **w** for write
- **x** for execute

You can assign or remove permissions to the **user** (owner), **group**, and **others** using the **following syntax**:

chmod [option][operator][permissions] file

Where:

- **[who]** can be: **options**
  - **u** for user (owner)
  - **g** for group

    - **o** for others
    - **a** for all (user, group, others)
- **[operator]** can be:
    - **+** to add permissions
    - **-** to remove permissions
    - **=** to set specific permissions, overriding existing ones
- **[permissions]** are:
    - **r** (read) - **4**
    - **w** (write) - **2**
    - **x** (execute) - **1**

Q. Explain file ownership and the chown command.

➔ **chown** is used to change the ownership**.**

Q. What does the umask command do?

➔ The umask command in Unix-like operating systems sets the default permissions for newly created files and directories.

How It Works:

When you create a new file or directory, Linux assigns **default permissions**.

- Default for **files** :- ‘ 666 ’ (read & write for everyone)

- Default for **directories** :- ‘ 777 ’ (read, write & execute for everyone)

- The **umask value subtracts permissions** from these defaults to set the final permissions.

Example:-

If `umask` is `022`:

- **Files:-** `666 - 022 = 644` (**rw-r--r--**) → Owner can read/write, others can only read.
- **Directories:-** `777 - 022 = 755` (**rwxr-xr-x**) → Owner has full access, others can read & execute.

Check & Set umask :-

- **Check** current umask:- `umask`
- **Change** umask:- `umask 027` (Restricts group and others)

syntax of umask :-

⇨ **umask [OPTION] [MASK]**

Common Usages :-

⇨ **Check current umask value :-**

```bash
umask
```

Example output: `0022` (Octal format)

⇨ **Set a new umask value :-**

```bash
umask 027
```

⇨ **This sets the umask to `027`, meaning :-**

- Files: `666 - 027 = 640` (**rw-r-----**)
- Directories: `777 - 027 = 750` (**rwxr-x---**)

⇨ **Use symbolic notation (like chmod) :-**

```bash
umask u=rwx,g=rx,o=
```

Equivalent to `umask 027`.

⇨ **Make umask permanent :-**

Add `umask 027` to your shell configuration file (`~/.bashrc`, `~/.profile`, etc.).

4. Process and System Monitoring

Q. How do you check running processes in Linux?

⇨ In Linux, you can check the running processes using several commands, with the most common being :-

⇨ **1. ps Command :-**

-The ps (process status) command shows a snapshot of the **current processes running on the system.**

Some common uses are :-

- **ps** : Displays a simple list of the current processes running in your terminal session.
- **ps aux** : Displays all running processes on the system, including those of other users.
- **ps -ef** : Another way to display all running processes, showing more details like parent process IDs (PPIDs).

⇨ 2. top Command :-

-The top command provides a real-time, dynamic view of the system's processes.
-It updates continuously, showing you CPU, memory usage, and the processes consuming resources.
top: Displays a real-time view of running processes.
-You can press **q** to exit the top command.

⇨ 3. htop Command :-

-**htop is an improved version of top, providing a more user-friendly, interactive interface with colors and the ability to scroll and kill processes directly.**
-It might need to be installed first.
htop: Displays an interactive process viewer.
-You can install it with:

```
sudo apt install htop   # On Debian/Ubuntu-based systems
sudo yum install htop   # On RHEL/CentOS-based systems
```

⇨ 4. pgrep Command :-

-If you're looking for processes related to a specific name, you can use **pgrep to search for processes by name.**
pgrep <process_name>: Finds processes by name.
Example: pgrep firefox will show the process IDs of all running Firefox processes.

⇨ 5. pidof Command :-

pidof returns the process ID of a running program.
pidof <program_name>: Shows the process ID of a specific program.

Example: pidof chrome will show the process ID of Chrome.

Q. What is the difference between ps, top, and htop?

• Feature	`ps`	`top`	`htop`
Type of View	Static (snapshot of processes)	Dynamic (real-time updates)	Dynamic (real-time updates, interactive)
Interactivity	No (just displays information)	Limited (can sort by CPU/memory)	Yes (interactive with colors, sort, kill processes)
Output Style	Plain text	Plain text	Colorful and more structured
Use Case	When you need a snapshot of processes at a specific moment	When you want to monitor processes in real-time	When you want an interactive, user-friendly process monitor
Default Installation	Pre-installed in most systems	Pre-installed in most systems	May need to be installed (`sudo apt install htop`)
Sorting Processes	No sorting options	Yes (by CPU, memory, etc.)	Yes (by CPU, memory, etc., with better control)
Resource Usage	Low (only runs once)	Moderate (updates in real-time)	Moderate (updates in real-time with more features)
Process Details	Basic info (PID, user, etc.)	More details (CPU, memory usage)	Most detailed and interactive

When to Use:

- **ps** : For a quick, one-time snapshot of processes.
- **top** : For real-time monitoring of system performance.
- **htop** : For an interactive and user-friendly experience with real-time process monitoring.

Q. How do you kill a process in Linux?

1. Use- **kill <PID>** to terminate a process.
2. Use- **kill -9 <PID>** to forcefully terminate it.
3. Use- **pkill <process_name>** or **killall <process_name>** to kill by name.

Q. What is the purpose of the nice and renice commands?

➔ 1. **nice**: Starts a process with a specified priority.

2. renice: Changes the priority of an already running process.

Q. How do you check system uptime?

⇨ 1. **uptime**: Simple and quick way to check uptime along with other system info.
2. **top**: Shows uptime along with detailed resource usage.
3. **cat /proc/uptime**: Displays uptime in seconds.
4. **who -b**: Shows the last system boot time.

Q. Difference between supervisor and hypervisor ?

Feature	Supervisor	Hypervisor
Definition	A software that controls the execution of programs, typically in a single operating system environment.	A software layer that allows multiple virtual machines to run on a single physical machine.
Function	Manages system resources and execution of processes.	Manages and allocates resources for virtual machines (VMs).
Scope	Works in a single environment (operating system).	Works across multiple environments (virtual machines).
Example	OS kernel, like in traditional operating systems.	VMware, Hyper-V, KVM (for virtual machines).

Usage	Used for managing processes and resources within a single OS.	Used for virtualization, creating and running multiple OS instances on a single physical machine.
Resource Management	Manages system resources like CPU, memory, and I/O for a single OS.	Manages hardware resources to allocate them to different virtual machines.
Level of Operation	Runs at a low level within a single OS environment.	Runs at a higher level, creating isolated environments (VMs) on top of the hardware.

5. Networking Basics

Q. How do you check your IP address in Linux?

1. **ip a** or **ip addr**: Preferred modern command to check local IP addresses.
2. **ifconfig**: Older command, still useful if installed.
3. **hostname -I**: Quick way to get the local IP address.
4. **nmcli device show**: For systems using NetworkManager.
5. **curl ifconfig.me**: To check your public IP address.
6. **nslookup** : to show the ip address of specific websites or link
   Syntax : nslookup example.com

   Example : 1. nslookup google.com
   2. nslookup cloudblitz.in

Q. What command is used to test network connectivity?

1. **ping**: To test basic network connectivity.
   ⇨ Syntax : **ping <hostname_or_IP_address>**
   Example : To test connectivity to a remote server or website (like Google):

   cmd: ping google.com

   Or to test connectivity to a specific IP address:

   Cmd: ping 8.8.8.8

Example Output :

PING google.com (142.250.80.78) 56(84) bytes of data.

64 bytes from 142.250.80.78: icmp_seq=1 ttl=56 time=12.3 ms

64 bytes from 142.250.80.78: icmp_seq=2 ttl=56 time=11.8 ms

- If the **ping** command **returns a response with times (in milliseconds), it means the network connection is working.**
- Press Ctrl + C to stop the ping command.

2. **traceroute**: To trace the route packets take to a destination.
   ⇨ **Syntax:** `traceroute <hostname_or_IP_address>`
   Example:
   Bash
   Cmd: **traceroute google.com**

3. **netstat**: To check for open ports and active network connections.
   ⇨ **Syntax** : **netstat -tuln**
   - This will list all active listening ports and their associated IP addresses.

4. **telnet**: To test specific port connectivity.
   ⇨ **Syntax** : **telnet <hostname_or_IP_address> <port>**
   Example : To check if port 80 (HTTP) on google.com is reachable
   Cmd: **telnet google.com 80**

5. **curl**: The **curl** command is often used to test HTTP or HTTPS connectivity. You can use it to make requests to a web server and see if the server responds
   ➔ **Syntax** : **curl <URL>**

   Example :
   Cmd: curl google.com
   - This command will display the raw HTML of the google.com page if the connection is successful.

Q . What is the difference between wget and curl?

⇨ **wget** is simpler for downloading files, especially when downloading entire sites or large sets of files.

curl is more powerful and flexible for transferring data, making API requests, and handling various protocols beyond just file downloads.

Feature	wget	curl
Purpose	Download files from the web.	Transfer data with URLs, interact with APIs.
Protocols Supported	HTTP, HTTPS, FTP.	HTTP, HTTPS, FTP, SFTP, SCP, IMAP, and many more.
Download Files	Automatically saves downloaded files.	Outputs to terminal by default unless redirected.
Recursive Download	Yes, can download websites or entire directories.	No built-in support for recursive downloads.
Redirect Handling	Follows redirects automatically.	Needs -L to follow redirects.
Resuming Downloads	Supports resume with -c option.	Supports resume with -C - option.
Working with APIs	Not as flexible with HTTP methods.	Very flexible, supports GET, POST, PUT, DELETE, etc.
Custom HTTP Headers	Not easy to customize.	Easily customize headers with -H.
Output	Saves the file with its original name.	Can output to terminal or save to a file using -o or -O.
Default Use Case	Simple file downloads.	Data transfers, API interaction, and file downloads with more control.

Q . How do you check open ports on your system?

➔ To check open ports on your system, there are different ways to do it depending on the operating system you are using.

Here are some common methods:

For Linux or macOS:

1. **Using netstat:**
    - Open a terminal.
    - Run the following command:

      sudo netstat -tuln

This shows the listening ports with their associated services. The flags are:

- **-t:** Show TCP ports
- **-u:** Show UDP ports
- **-l:** Show only listening ports
- **-n:** Show numeric addresses (no domain resolution)

2. **Using ss (Socket Stat):**
    - Another command that's often recommended on Linux is **ss**, which provides more detailed output and is faster than netstat:

      sudo ss -tuln

3. **Using lsof:**
    - You can also use the **lsof** command to check open ports:

      sudo lsof -i -P -n

This will list all open files and their associated ports. The **-i** flag indicates network files, **-P** prevents port name resolution, and **-n** avoids host name resolution.

For Windows:

1. **Using netstat:**
    - Open Command Prompt as Administrator.
    - Run the following command:

      netstat -an | find "LISTEN"

This shows a list of all listening ports. The -**a** flag lists all connections, and -**n** shows the addresses numerically.

2. **Using PowerShell:**
    - You can use PowerShell to list open ports:

```
Get-NetTCPConnection | Where-Object { $_.State -eq 'Listen' }
```

This will show all TCP connections that are in the "Listen" state.

For Both Windows and Linux (Using nmap):

- **nmap** (Network Mapper) is a powerful network scanning tool that can be used on both Linux and Windows. You need to install it first (it's usually pre-installed on Linux).
    - Run the following command to scan for open ports on your system:

      ```
      nmap -p- localhost
      ```

This will scan all ports on your local machine.

For macOS (Additional Tool - lsof):

- You can also use **lsof** on macOS, which is similar to Linux. Here's a command to check open ports:

  ```
  sudo lsof -i -P -n
  ```

Q. How do you find the DNS records of a domain in Linux?

➔ To find the DNS records of a domain in Linux, you can use several tools.

Here are the most common methods:

1. Using dig (Domain Information Groper)

- dig is a powerful and flexible tool for querying DNS records.

- It can retrieve different types of DNS records for a domain, such as **A, MX, NS, TXT**, etc.

- To get the **A** record (IPv4 address) for a domain:

  ```
  dig example.com
  ```

- To get all available DNS records (like A, MX, TXT, etc.):

  ```
  dig example.com ANY
  ```

- To query a specific type of DNS record (e.g., **MX** for mail servers):

  ```
  dig example.com MX
  ```

- To get the **NS** (name server) records:

  dig example.com NS

- To get the **TXT** records (often used for SPF, DKIM, etc.):

  dig example.com TXT

- If you want to query a specific DNS server:

  dig @8.8.8.8 example.com

2. Using nslookup

- nslookup is another utility that you can use to query DNS records.

- It is widely available on Linux systems and works similarly to dig.

- To get the **A** record (IP address) of a domain:

  nslookup example.com

- To get specific records like **MX**, **NS**, or **TXT**:

  nslookup -type=MX example.com

  nslookup -type=NS example.com

  nslookup -type=TXT example.com

- To query a specific DNS server:

  nslookup example.com 8.8.8.8

3. Using host

- host is another simple command-line tool to get DNS records.

- To get the **A** record of a domain:

  host example.com

- To get the **MX** records:

  host -t MX example.com

- To get the **NS** records:

  host -t NS example.com

4. Using whois (For Domain Information)

- whois can provide registration and DNS information for domains.

- It doesn't give detailed DNS record information like dig or nslookup, but it can give you details about the domain's registrar, name servers, and more.

- To find the registrar information and name servers for a domain:

  whois example.com

Example Commands:

Here are some practical examples:

- Get **A** record:

  dig example.com A

- Get **MX** records (Mail servers):

  dig example.com MX

- Get **NS** records (Name servers):

  dig example.com NS

6. Disk and Storage Management

Q. How do you check free disk space?

➔ To check free disk space on your system in Linux, there are several commands you can use.

Here are the most common ones:

1. Using df (Disk Free)

- The **df** command is the most commonly used tool to check disk space.

- It shows the amount of disk space used and available on your file systems.

- Basic command to **show disk space in human-readable format**:

  df -h

  - The -**h** flag makes the output human-readable, **showing sizes** in **KB, MB, or GB.**

- **To check disk space for a specific directory or file system:**

  df -h /path/to/directory

- To get more detailed information about the file systems:

  df -Th

  - The **-T** flag **shows the type of file system**.

2. Using du (Disk Usage)

While, **df** shows the **overall disk space usage**,

du shows the **disk usage of files and directories**.

- **To check the disk usage of a specific directory**:

  du -sh /path/to/directory

  - The **-s** flag gives you the **total size of the directory**.
  - The **-h** flag **makes it human-readable**.

- **To check disk usage of all directories inside a directory:**

  du -h /path/to/directory

3. Using lsblk (List Block Devices)

The **lsblk** command can **display all block devices (including disks and partitions) and their sizes.**

- **To list all block devices with their sizes:**

  lsblk

- **To display the disk space in a human-readable format**:

  lsblk -o NAME,SIZE,FSTYPE,MOUNTPOINT

4. Using fdisk (Partition Table)

- fdisk is used to **show partition information**.

- While it's generally used for partition management, **it can also show free disk space and the size of each partition.**

- **To list all partitions and their sizes**:

  sudo fdisk -l

5. Using free (Memory Usage)

The free command is typically used to check memory usage, but it also shows swap space, which is important for disk space management.

- **To check memory and swap usage**:

  free -h

Example Commands:

- **Check free disk space** (with df):

  df -h

- **Check disk usage of a directory** (with du):

  du -sh /home/user

- **List all block devices** (with lsblk):

  lsblk

Summary:

- **df -h**: Check overall disk usage.
- **du -sh /path/to/directory**: Check the disk usage of a specific directory.
- **lsblk**: List all block devices and their sizes.
- **fdisk -l**: Display partition sizes.

Q. What is the difference between df and du?

➔ **df** when you want to check the total disk usage and available space on your entire file system or partitions.

du when you want to check how much disk space a particular file or directory is using.

Feature	df	du
Purpose	Shows overall disk space on mounted file systems	Shows disk space usage for specific files or directories
Scope	Provides information for entire file systems (partitions)	Provides information about specific files or directories
Output	Displays total, used, and available space for file systems	Displays the disk space usage of files/directories
Typical Use	Checking free space on disk/partition	Checking how much space a specific directory or file is using
Example Use	df -h to check free space on all partitions	du -sh /path/to/directory to check usage of a specific directory

Q. How do you find the size of a specific directory?

➔ To find the size of a specific directory in Linux, you can use the **du (Disk Usage)** command.

Here's how to do it:

1. Using du to Find the Size of a Directory

The du command will show the disk usage of the specified directory and all of its contents.

- **to show the size of a directory:**

  du /path/to/directory

This will give you the disk usage of the specified directory and each subdirectory within it.

- **Human-readable output** (KB, MB, GB):

  du -sh /path/to/directory

    - **-s:** the total size of the directory (don't show the size of each subdirectory).
    - **-h:** Make the output **human-readable** (shows the size in KB, MB, GB, etc.).

- **Show disk usage of all subdirectories** in the directory:

  du -h /path/to/directory

This will list the disk usage for each subdirectory and the total for the main directory at the end.

- **Including hidden files** (in case the directory contains hidden files):

  du -sh /path/to/directory/*

2. Example Usage

- To find the size of the **/home/user/Documents** directory:

  du -sh /home/user/Documents

Output:

2.5G /home/user/Documents

This means the directory Documents is using 2.5 GB of disk space.

- **To find the size of all directories inside /home/user:**

  du -sh /home/user/*

Output:

2.5G /home/user/Documents

1.0G /home/user/Downloads

500M /home/user/Music

3. Additional du Options

- **Display file sizes recursively for all files in the directory**:

  du -ah /path/to/directory

    - **-a:** Show file sizes in **addition to directory sizes.**

- **Sort the output by size**: You can combine **du** with the sort command to list directories by their size:

  du -sh /path/to/directory/* | sort -h

  - This will list directories and files inside /path/to/directory sorted by size in human-readable format.

Q. What is the purpose of the mount and umount commands?

➔ **Key Differences:**

Command	Purpose
mount	Mounts a file system, making it accessible in the file system hierarchy. Syntax : mount [options] <device> <mount_point> Example **: To mount a USB drive (e.g., /dev/sdb1) to the /mnt/usb directory** sudo **mount** /dev/sdb1 /mnt/usb
umount	Unmounts a file system, detaching it safely from the file system hierarchy. Syntax : umount [options] <mount_point_or_device> Example : **To unmount a USB drive mounted at /mnt/usb:** sudo **umount** /mnt/usb

Q. How do you format a disk partition in Linux?

➔ To format a disk partition in Linux, the main commands are:

1. **Identify the partition:**

   sudo fdisk -l

2. **Unmount the partition (if mounted):**

   sudo umount /dev/sdX1

3. **Format the partition (e.g., as ext4):**

```
sudo mkfs.ext4 /dev/sdX1
```

Replace /dev/sdX1 with your actual partition.

7. Shell Scripting Basics

Q. What is a shell script?

➔ Shell Scripting is nothing but the list of commands which are listed in the script in the order of execution.

Shell :-

Shell is a program which provides the interface between the user and operating system.

Tupes of shells :

1. C shell (csh)
2. Bourne shell (sh)
3. korn shell (ksh)
4. Bourne Again shell (bash)

Q. How do you make a shell script executable?

➔ To make a shell script executable, follow these easy steps:

1. **Write your shell script** : Create your shell script (like myscript.sh) using any text editor.
2. **Give permission to run it** : You need to tell your computer that the script can be run.

   To do this, open a terminal and type this command:

   ```
   chmod +x myscript.sh
   ```

This command allows your script to be executed (it adds "execute" permission).

3. **Run the script** : To run your script, type this in the terminal:

   ```
   ./myscript.sh
   ```

The ./ part just means "look in the current folder for the script."

In short:

- **chmod +x** : Makes the script executable.
- **./** : Runs the script.

Q. What is the shebang (#!) in a shell script?

➔

Q. How do you pass arguments to a script?

➔ To pass arguments to a shell script, you just type them after the script's name when running it.

Here's how:

Steps:

1. **Write your script**: Create a shell script (e.g., myscript.sh) that will use the arguments.

   For example:

   ```
   #!/bin/bash

   echo "First argument: $1"

   echo "Second argument: $2"
   ```

2. **Make the script executable** (if you haven't already):

   ```
   chmod +x myscript.sh
   ```

3. **Run the script with arguments**: When you run the script, add the values you want to pass as arguments, like this:

   ```
   ./myscript.sh Hello World
   ```

What happens:

- $1 will be **"Hello"** (the first argument).
- $2 will be **"World"** (the second argument).

Example Output:

```
First argument: Hello

Second argument: World
```

In short:

- When running the script, just add words or values after the script name. These are the **arguments**.
- Inside the script, use $1, $2, etc., to get those values.

Q. What is the difference between " (double quotes) and ' (single quotes) in shell scripting?

➔ **Double quotes (“) :** allow variables and commands inside them to be evaluated.

Single quotes (‘) : treat everything inside them as plain text, without evaluating variables or commands.

Feature	Double Quotes (")	Single Quotes (')
Variable Expansion	Expands variables inside the quotes ($variable).	Does **not** expand variables.
Command Substitution	Allows command substitution ($(command)).	Does **not** allow command substitution.
Escape Characters	Escape characters like \n, \" are processed.	Escape characters (except for ') are ignored.
Use Case	Used when you need to evaluate variables or commands.	Used when you need literal, unchanged strings.

8. Package Management

Q. What is a package manager in Linux?

➔ - A **package manager** in Linux is a tool that helps you install, update, remove, and manage software packages (applications and libraries) on your system.

- It makes handling software easy by automating tasks like downloading, installing, and updating software from repositories.

Key Features of a Package Manager:

1. **Installation** : Installs software packages automatically.
2. **Updates** : Keeps software up-to-date with the latest versions.
3. **Dependencies** : Handles dependencies, which are other software libraries or packages that your software needs to work.
4. **Removal** : Uninstalls packages you no longer need.
5. **Repositories** : A package manager connects to repositories, which are servers containing precompiled software packages.

Common Package Managers:

- **APT (Advanced Package Tool)**: Used in **Debian-based** distributions like Ubuntu.
    - Command: sudo apt install package_name
- **YUM (Yellowdog Updater, Modified)**: Used in **Red Hat-based** distributions like CentOS and Fedora.
    - Command: sudo yum install package_name
- **DNF (Dandified YUM)**: The newer version of YUM, used in modern Fedora.
    - Command: sudo dnf install package_name
- **Pacman**: Used in **Arch Linux** and its derivatives.
    - Command: sudo pacman -S package_name
- **Zypper**: Used in **openSUSE**.
    - Command: sudo zypper install package_name

How It Works:

1. You use the package manager to install a software package.
2. The package manager looks for the software in the repository and checks if it has all the required dependencies.
3. It installs the software along with any needed dependencies.
4. If the software needs an update, the package manager will check for the latest version and install it.

Example:

To install **Firefox** on an Ubuntu-based system using APT:

```
sudo apt update                    # Updates the list of available packages
sudo apt install firefox           # Installs Firefox
```

in a simple way , a **package manager** simplifies the process of managing software on Linux by automating installations, updates, and removals, making it much easier for users to handle software management.

Q. What are the differences between apt, yum, and dnf?

- **APT** is for Debian-based systems.
- **YUM** is for older Red Hat-based systems.
- **DNF** is the modern version of YUM for newer Red Hat-based systems.

Feature	**APT (Advanced Package Tool)**	**YUM (Yellowdog Updater, Modified)**	**DNF (Dandified YUM)**
Used In	Debian-based systems (Ubuntu, Debian, Mint)	Red Hat-based systems (CentOS, RHEL, Fedora)	Newer Red Hat-based systems (Fedora, CentOS 8, RHEL 8)
Package Format	.deb packages	.rpm packages	.rpm packages
Command Example	sudo apt install package_name	sudo yum install package_name	sudo dnf install package_name

Speed	Fast, especially for everyday use	Slower, especially with large packages	Faster than YUM, improved performance
Dependency Handling	Handles dependencies automatically	Handles dependencies, but can be slower or struggle with conflicts	Improved dependency handling and faster performance
Features	Easy to use, handles updates well	Older, slower, good for stable, enterprise systems	Modern, more efficient, with better features than YUM
Used On	Desktop systems, personal computers	Enterprise and server systems	Latest server and workstation systems

Q. How do you install, update, and remove a package using a package manager?

➔ **1. Using apt (Debian-based systems like Ubuntu)**

- **Install a package:**

  sudo apt install <package-name>

- **Update a package:**

  sudo apt update # Update package list

  sudo apt upgrade # Upgrade installed packages

  sudo apt install <package-name> # Upgrade specific package

- **Remove a package:**

  sudo apt remove <package-name>

  sudo apt purge <package-name> # To remove configuration files too

Q. What is the purpose of the **dpkg** and **rpm** commands?

➔ The dpkg and rpm commands are both package management tools used in different Linux distributions for installing, managing, and removing software packages.

They serve a similar purpose but are used on different package management systems. Here's a breakdown:

dpkg (Debian Package)

- **Used by:** Debian-based distributions, such as Ubuntu, Linux Mint, and Debian itself.
- **Purpose:** dpkg is a low-level tool that handles .deb packages (Debian package files).
- **Common commands:**
    - dpkg -i <package.deb>: Install a .deb package.
    - dpkg -r <package>: Remove an installed package.
    - dpkg -l: List all installed packages.
    - dpkg -S <file>: Find which package a file belongs to.

dpkg works with individual package files (i.e., .deb files) and does not automatically resolve dependencies, which is where higher-level package managers like apt (Advanced Package Tool) come in.

rpm (Red Hat Package Manager)

- **Used by:** Red Hat-based distributions, such as Red Hat Enterprise Linux (RHEL), CentOS, Fedora, and openSUSE.
- **Purpose:** rpm is a low-level tool that handles .rpm packages (Red Hat package files).
- **Common commands:**
    - rpm -i <package.rpm>: Install an .rpm package.
    - rpm -e <package>: Remove an installed package.
    - rpm -q <package>: Query a package to check if it’s installed.
    - rpm -ql <package>: List files installed by a package.

Like dpkg, rpm also does not resolve dependencies automatically, which is typically handled by higher-level package managers like yum or dnf for Red Hat-based distributions.

Summary:

- **dpkg**: Used for .deb packages, common in Debian-based distributions (e.g., Ubuntu).
- **rpm**: Used for .rpm packages, common in Red Hat-based distributions (e.g., RHEL, CentOS).

Q. How do you find which package a command belongs to?

➔ For Ubuntu or Debian (Debian-based)

If the command is already installed:

dpkg -S /path/to/command

Example:

dpkg -S /bin/ls

→ This will tell you which package installed the ls command.

If the command is not installed and you want to know which package provides it:

sudo apt install apt-file

sudo apt-file update

apt-file search /bin/ls

→ It will show you the package name containing ls.

9. Log Management & Troubleshooting

Q. Where are system logs stored in Linux?

➔ In Linux, system logs are essential for monitoring and troubleshooting issues. They store information about system activities, applications, services, and errors.

☑ Location of System Logs

Most **system logs are stored in the /var/log/ directory**. This directory contains various log files for the system, services, and applications.

System Logs: /var/log/syslog or /var/log/messages

Contains general system messages, including startup messages and application logs.

Used for debugging and monitoring the system.

Q. How do you view logs in real time?

➔ 1. Using tail Command

The tail command is typically used to display the last few lines of a log file, but you can also use it to watch the log file in real-time as new entries are added.

- **To view logs in real-time:**

  tail -f /path/to/logfile

  Example (viewing the system log in real-time):

  tail -f /var/log/syslog

This command will show you the latest log entries as they are written to the file. To stop viewing the logs, press Ctrl+C.

Q. What is the purpose of the /var/log directory?

➔ The /var/log directory in Linux is a central location where system logs and application logs are stored.

Logs are important for troubleshooting, monitoring system performance, tracking security events, and maintaining system health.

Key Purposes of the /var/log Directory:

1. **System Logging**: It holds logs that record system events, such as boot-up messages, system errors, and kernel information.
2. **Application Logs**: Many software applications, services, and daemons store their operational logs in this directory to record events, errors, and status updates.
3. **Security Auditing**: Logs related to system security, including user logins, access attempts, and authentication failures, are stored here, helping with auditing and security monitoring.
4. **Troubleshooting and Debugging**: When issues occur, system administrators and developers check logs to identify problems, system failures, or unexpected behavior.

Common Log Files in /var/log:

Here are some of the typical log files you may find in the /var/log directory:

- **/var/log/syslog**: General system log, often used for tracking system messages, application errors, and other general system activities.
- **/var/log/messages**: Similar to syslog, it contains general system logs, particularly related to system hardware or services.
- **/var/log/auth.log**: Contains logs related to authentication, such as successful and failed login attempts, sudo actions, and user access.
- **/var/log/kern.log**: Contains logs generated by the kernel, such as hardware detection and kernel-related events.
- **/var/log/dmesg**: Contains kernel ring buffer messages, which include information from the system boot process and hardware detection.
- **/var/log/boot.log**: Contains logs from the system boot process, useful for troubleshooting boot issues.
- **/var/log/apt/**: Contains logs for package management activities, such as installation, removal, or updates when using APT-based systems (e.g., Ubuntu).
- **/var/log/httpd/ or /var/log/nginx/**: Logs for web servers (Apache or Nginx), including access logs and error logs.
- **/var/log/mail.log**: Logs related to email services, such as mail delivery, spam filtering, or email server errors.
- **/var/log/cron**: Logs related to scheduled tasks (cron jobs).

Why It's Important:

- **Monitoring and Security**: Logs help administrators detect issues like failed login attempts, system resource issues, or malicious activities.
- **Debugging and Troubleshooting**: Logs provide valuable insights when diagnosing problems with the system or applications.
- **Auditing and Compliance**: For security and legal reasons, log data can help track user activity and system changes.

Q. How do you check failed login attempts?

➔ For Ubuntu or Debian (Debian-based)

Logs are stored in /var/log/auth.log.

1. Check All Failed Logins

grep "Failed" /var/log/auth.log

- This will show all failed login attempts.

2. Check Failed SSH Logins

grep "Failed password" /var/log/auth.log

- This is useful for tracking SSH login failures.

Q. How do you monitor CPU and memory usage?

➔ To monitor **CPU** and **memory usage** on your Linux system, there are simple commands you can use.

These help you see how much of your system's resources (CPU and memory) are being used by different processes.

1. Using top Command:

- **What it does**: The top command gives you a live, real-time view of your system's CPU and memory usage.
- **How to use**: Open your terminal and type:

  Cmd : top

  - It shows a list of running programs and how much CPU and memory each one is using.
  - **CPU usage** is shown as a percentage (how much of your processor is being used).
  - **Memory usage** is shown as the percentage of RAM (how much of your system's memory is being used).
- **How to quit**: Press q to exit.

2. Using htop Command:

- **What it does**: htop is like a better version of top, with color and easier-to-read information.
- **How to use**: If it's not installed, you can install it first:
- sudo apt install htop # On Ubuntu/Debian
- sudo yum install htop # On CentOS/Red Hat

Then, run:

Cmd : htop

It shows CPU usage per core and memory usage in a nice, visual format.

- **How to quit**: Press F10 or q to exit.

3. Using free Command:

- **What it does**: The free command shows how much memory (RAM) is being used and how much is free.
- **How to use**:

  Cmd : free -h

  - The -h flag shows the data in a human-readable format (e.g., MB, GB).
  - It shows:
    - **Used memory**: How much memory is in use.
    - **Free memory**: How much memory is available.
    - **Swap**: The amount of hard drive space being used as virtual memory.

4. Using vmstat Command:

- **What it does**: vmstat gives you more detailed information about CPU, memory, and other system performance stats.
- **How to use**: Type this in the terminal:

  Cmd : vmstat 1

  - This shows real-time stats, updating every second.

In Summary:

- **top**: Shows CPU and memory usage for running programs.
- **htop**: A better version of top with color and a clearer display.
- **free -h**: Shows total, used, and free memory.
- **vmstat 1**: Shows detailed system stats every second.

10. File Compression and Archiving

Q. How do you create a tar archive?

➔ To create **a .tar archive**, you can use the **tar** command in a terminal or command line interface.

Here's the general syntax:

```
tar -cvf archive_name.tar /path/to/directory_or_files
```

Here's what each option means:

- **c** stands for **"create"** — this option tells tar to create a new archive.
- **v** stands for **"verbose"** — this option makes tar show the progress by listing the files as it archives them (optional).
- **f** stands for **"file"** — this option tells tar to use the file name that follows it.
- **archive_name.tar** is the name you want for the resulting .tar file.
- **/path/to/directory_or_files** is the path to the directory or file(s) you want to include in the archive.

Example:

1. If you want to archive a folder called documents, the command would be:

```
tar -cvf documents.tar documents/
```

This will create a documents.tar file containing everything in the documents directory.

2. If you want to include multiple files or directories, just list them after the tar file name:

```
tar -cvf archive.tar file1.txt file2.txt folder1/
```

This will create archive.tar with file1.txt, file2.txt, and folder1/.

Q. What is the difference between tar, gzip, and zip?

➔

Aspect	tar	gzip	zip
Primary Purpose	Archiving files (bundles multiple files/folders)	Compression (reduces file size)	Archiving and compression (both)
File Extension	.tar	.gz	.zip
Compression	No compression by default (can combine with gzip or bzip2)	Compresses a single file	Compresses and archives files in one step
How it Works	Packages multiple files or directories into a single archive file	Compresses a single file (or a .tar archive)	Combines both archiving and compression
Multi-File Handling	Yes, handles multiple files/folders (e.g., tar -cvf archive.tar files/)	No, works with a single file only	Yes, handles multiple files/folders (e.g., zip -r archive.zip folder/)
Command Example	tar -cvf archive.tar /path/to/dir	gzip file.txt	zip -r archive.zip /path/to/dir
Common Use Case	Archiving (grouping files together)	Compressing files for size reduction	Archiving and compressing files together
Compression Support	No (but can use with other tools like gzip)	Yes (single file compression)	Yes (compresses everything in the archive)
Platform Compatibility	Common on Linux/Unix systems	Common on Linux/Unix systems	Widely supported across all platforms (Windows, Linux, macOS)
Handling of Directories	Yes, can include directories and subdirectories	No, works on files only	Yes, handles directories and subdirectories

Quick Summary:

- **tar**: Archiver that bundles files without compression, but can work with tools like gzip for compression.
- **gzip**: A compression tool that reduces file size for a single file (can be used with tar).
- **zip**: An all-in-one tool for both archiving and compressing files, popular for cross-platform compatibility.

Q. How do you extract a .tar.gz file?

➔ To extract a .tar.gz file, you can use the tar command with specific options in the terminal.

The general syntax is:

```
tar -xvzf file.tar.gz
```

Here's what each option means:

- **x** stands for **"extract"** — this tells tar to extract the contents of the archive.
- **v** stands for **"verbose"** — this shows the progress by listing the files being extracted (optional).
- **z** stands for **"gzip**" — this tells **tar to decompress** the **.gz file** (since it's compressed with gzip).
- **f** stands for **"file"** — this specifies that the following name is the archive file to extract.
- **file.tar.gz** is the name of the archive you want to extract.

Example:

To extract a file named archive.tar.gz, you would run:

```
tar -xvzf archive.tar.gz
```

Notes:

- This command will extract the files into the current directory.
- If you want to extract the archive to a specific directory, you can use the -C option followed by the path to the desired directory:

```
tar -xvzf archive.tar.gz -C /path/to/extract/to/
```

Q. How do you compress a file using gzip?

➔ To compress a file using gzip, you can use the following command in the terminal:

gzip filename

Explanation:

- **gzip** is the command to compress files.
- **filename** is the name of the file you want to compress.

Example:

To compress a file called **example.txt**, you would run:

gzip example.txt

This will compress the file and create a compressed version of it called **example.txt.gz.** The original file **(example.txt)** will be replaced with the compressed file.

Q. How do you check the contents of an archive without extracting it?

➔ To check the contents of an archive without extracting it, you can use the tar command for **.tar, .tar.gz, .tar.bz2,** or similar archive types.

Here's how you can do it:

1. For .tar, .tar.gz, .tar.bz2, and .tgz files:

tar -tvf archive_name.tar

Explanation:

- **t** stands for **"list"** — this tells tar to list the contents of the archive without extracting them.
- **v** stands for **"verbose"** — this option provides detailed output, listing each file in the archive (optional).
- **f** stands for **"file"** — this specifies the file name that follows it.
- **archive_name.tar** is the name of the archive you want to inspect.

Example:

To check the contents of a archive.tar.gz file, you would run:

tar -tvzf archive.tar.gz

This will display the list of files and directories contained in the archive.tar.gz without extracting them.

---

2. For .zip files:

You can use the **unzip** command to list the contents of **a .zip** file **without extracting**:

unzip -l archive.zip

This will show a list of files inside the **archive.zip file.**

Summary:

- **For tar archives**: tar -tvf archive_name.tar (or use **-z** for **.tar.gz**, **-j** for .tar.bz2).
- **For zip archives**: unzip -l archive_name.zip.
