LINUX

BASIC LINUX COMMANDS

Definition:
Linux commands are instructions used in the **terminal** to interact with the operating system for file, process, and system management.

Why Linux Commands Are Needed (Purpose)

- Most servers run on **Linux**
- Used in **production support & DevOps**
- Faster than GUI
- Essential for **Cognizant training & interviews**

File & Directory Commands

pwd

- Shows current directory
- Example: pwd

---

ls

- Lists files and folders
- ls -l → detailed view
- ls -a → shows hidden files

---

cd

- Change directory
- cd folderName
- cd .. → go back

---

mkdir

- Create directory
- mkdir test

---

rmdir

- Remove empty directory
- rmdir test

---

touch

- Create empty file
- touch file.txt

---

rm

- Delete files/directories
- rm file.txt
- rm -r folderName

File Viewing Commands

cat

- View file content
- cat file.txt

---

more / less

- View large files

- less file.txt (preferred)

---

head / tail

- First / last lines of file
- head -n 5 file.txt
- tail -n 10 file.txt

Search & Filter Commands

grep

- Search text in file
- grep "error" log.txt

---

find

- Find files
- find /home -name file.txt

File Operations

cp

- Copy files
- cp a.txt b.txt

---

mv

- Move or rename files
- mv old.txt new.txt

Permission & Ownership

chmod

- Change permissions
- chmod 755 file.sh

---

chown

- Change ownership
- chown user:group file.txt

System & Process Commands

ps

- Show running processes
- ps -ef

---

top

- Live process monitor

---

kill

- Stop process
- kill -9 PID

Disk & Memory

df

- Disk space usage
- df -h

---

free

- Memory usage
- free -m

Network Commands
ping

- Check connectivity
- ping google.com

---

ifconfig / ip a

- Network details

---

netstat / ss

- Network connections

Package & Help
sudo

- Run command as admin

---

man

- Command manual
- man ls

Real-Life Example
Production issue:

- cd /logs
- grep "error" app.log
- tail -100 app.log

Interview Explanation (How to Say It)
"Linux commands help manage files, processes, and systems efficiently through the terminal."

Common Mistakes Freshers Make

- Using rm -rf blindly
- Forgetting permissions
- Not checking current directory
- Killing wrong process

One-Line Summary
Linux commands are essential for server and production-level work.

FILE MANAGEMENT (LINUX)

Definition:
File Management in Linux is the process of **creating, organizing, viewing, copying, moving, and deleting files and directories**.

Why File Management is Needed (Purpose)

- To organize application data and logs
- To manage files on servers efficiently
- To support **production & maintenance work**
- To troubleshoot issues quickly

Basic File Management Commands
Create Files & Directories

- touch file.txt → create file
- mkdir folder → create directory
- mkdir -p a/b/c → nested directories

View Files

- cat file.txt → view full file
- less file.txt → large files
- head -n 10 file.txt → first 10 lines
- tail -n 20 file.txt → last 20 lines

Copy Files & Directories

- cp a.txt b.txt → copy file
- cp -r dir1 dir2 → copy directory

Move / Rename Files

- mv old.txt new.txt → rename
- mv file.txt /tmp/ → move

Delete Files & Directories

- rm file.txt → delete file
- rm -r folder → delete directory
- ⚠️ rm -rf and rm -rf / → dangerous (no undo)

File Permissions (Basics)

- Read (r), Write (w), Execute (x)
- Example: chmod 755 script.sh
- Owner | Group | Others

Real-Life Example

Office files:

- Create folders per project
- Move files to correct folders
- Delete old unused files

Technical Example (Production)

Log management:

```
cd /var/log
ls
grep "ERROR" app.log
cp app.log backup.log
```

Interview Explanation (How to Say It)

“File management in Linux involves handling files and directories using commands like ls, cp, mv, rm, and chmod.”

Common Interview Questions

- Difference between rm and rmdir?
- How to copy a directory?
- How to view large files?
- How to change file permissions?

Common Mistakes Freshers Make

- Using rm -rf without checking
- Forgetting file permissions
- Confusing files and directories

- Working in wrong directory

One-Line Summary
File management helps organize and control data efficiently in Linux systems.

DIRECTORY MANAGEMENT (Linux)

Definition:
Directory Management is the process of **creating, navigating, organizing, and deleting folders (directories)** in Linux.

Why Directory Management is Needed (Purpose)
- To organize project files and logs
- To manage server data properly
- To navigate Linux systems efficiently
- To support production and maintenance tasks

Basic Directory Management Commands
Create Directories
- mkdir test → create directory
- mkdir dir1 dir2 → multiple directories
- mkdir -p a/b/c → nested directories

Navigate Directories
- pwd → current directory
- cd folderName → go inside
- cd .. → go back
- cd ~ → home directory

List Directory Contents
- ls → list files
- ls -l → detailed view
- ls -a → hidden files

Delete Directories
- rmdir test → empty directory
- rm -r test → non-empty directory

Directory Permissions (Basics)
- r → read (list files)
- w → write (create/delete files)
- x → execute (enter directory)
  Example: chmod 755 mydir

Real-Life Example
Think of **folders in your laptop**:
- Create folders for subjects
- Move files to correct folders
- Delete unused folders

Technical Example (Production)
Managing app folders:

```
cd /opt
mkdir app_logs
cd app_logs
ls
```

Interview Explanation (How to Say It)
"Directory management in Linux involves creating, navigating, listing, and deleting folders using commands like mkdir, cd, ls, and rm."

Common Interview Questions

- Difference between rm and rmdir?
- What does mkdir -p do?
- What does execute permission mean for directories?

Common Mistakes Freshers Make

- Deleting wrong directory
- Forgetting -r while deleting
- Confusing file and directory permissions
- Working without checking pwd

One-Line Summary
Directory management helps organize and navigate the Linux file system efficiently.

FILE PERMISSIONS and ACCESS MODES (Linux)

Definition:
File permissions and access modes control **who can read, write, or execute** a file or directory in Linux.

Why Permissions Are Needed (Purpose)

- To protect files from unauthorized access
- To ensure system security
- To control user actions on files/directories
- To prevent accidental data loss

Permission Types (Access Modes)
Three Permission Types

- **Read (r)** → View file / list directory
- **Write (w)** → Modify file / create-delete files
- **Execute (x)** → Run file / enter directory

Permission Levels (Who)
Three User Levels

1. **Owner (u)** – file creator
2. **Group (g)** – group members
3. **Others (o)** – everyone else

Permission Representation
Symbolic Mode
Example:
-rwxr-xr--
Breakdown:

- rwx → Owner
- r-x → Group
- r-- → Others

Numeric (Octal) Mode

Permission	Value
r	4
w	2
x	1

Example:
chmod 755 file.txt
→ Owner: rwx (7)
→ Group: r-x (5)
→ Others: r-x (5)

Changing Permissions
chmod Command

- Symbolic: chmod u+x file.sh
- Numeric: chmod 644 file.txt

Ownership Command
chown user:group file.txt

File vs Directory Permissions
Key Difference

Permission	File	Directory
Read	View content	List files
Write	Modify content	Create/Delete files
Execute	Run file	Enter directory

Real-Life Example
House access:

- Read → see inside
- Write → change furniture
- Execute → enter house

Technical Example
Script execution:
chmod +x deploy.sh
./deploy.sh

Interview Explanation (How to Say It)
“Linux permissions define read, write, and execute access for owner, group, and others using symbolic or numeric modes.”

Common Interview Questions

- What does 755 mean?
- Difference between chmod and chown?
- What is execute permission for directory?

Common Mistakes Freshers Make

- Giving 777 permissions unnecessarily

- Forgetting execute permission on scripts
- Confusing file and directory permissions
- Changing permissions without understanding

One-Line Summary
File permissions control access and protect Linux systems.

BASIC LINUX UTILITIES

Definition:
Basic Linux utilities are built-in **command-line tools** used for **file handling, text processing, monitoring, and system management.**

Why Utilities Are Needed (Purpose)
- To manage files and directories efficiently
- To process text/logs for analysis
- To monitor system resources
- To support production and DevOps tasks

Common Linux Utilities
File & Directory Utilities
- ls → List files
- pwd → Current directory
- cp → Copy files
- mv → Move/rename files
- rm → Remove files
- mkdir → Create directories

Text Processing Utilities
- cat → View file content
- less / more → View large files
- head / tail → View first/last lines
- grep → Search text
- wc → Count words, lines, characters
- cut → Extract columns

Disk & Storage Utilities
- df → Disk space usage
- du → Directory/file size
- lsblk → List block devices

Process & System Utilities
- ps → View processes
- top → Monitor processes live
- kill → Stop process by PID
- uptime → System uptime
- free → Memory usage

Networking Utilities
- ping → Check connectivity
- ifconfig / ip a → Network info
- netstat / ss → Connections & ports

Archiving & Compression

- tar -cvf file.tar folder/ → Archive
- tar -xvf file.tar → Extract
- gzip / gunzip → Compress/Decompress

Real-Life Example

Monitoring logs:

```
cd /var/log
tail -f app.log | grep "ERROR"
```

- View last lines of log and filter errors

Technical Example

Copy backup logs:

```
cp /var/log/app.log /backup/app_$(date +%F).log
```

Interview Explanation (How to Say It)

"Linux utilities are command-line tools for file management, text processing, system monitoring, and networking, essential for production and development."

Common Interview Questions

- Name some Linux utilities
- How to check disk usage?
- How to search for a keyword in a file?
- How to monitor CPU & memory usage?

Common Mistakes Freshers Make

- Ignoring options/flags of commands
- Using rm carelessly
- Forgetting grep for filtering
- Not understanding process commands

One-Line Summary

Linux utilities simplify file handling, system monitoring, and text processing for effective operations.

PIPES and FILTERS (Linux)

Definition:

Pipes and Filters are Linux features used to **process data efficiently by combining commands.**

- **Filter:** Command that processes input and gives output.
- **Pipe (|):** Sends output of one command as input to another.

Why Pipes & Filters Are Needed (Purpose)

- To process data **without creating temporary files**
- To **combine simple commands** for complex tasks
- To save **time and system resources**
- Essential for **log analysis, production support, and automation**

Key Points (Core Concepts)

- **Pipe (|)** → Connects commands

- **Filters** → grep, sort, uniq, cut, head, tail, wc
- Can chain multiple commands
- Supports text processing and monitoring

Common Filters

Command	Purpose	Example
grep	Search text	grep "ERROR" log.txt
sort	Sort lines	sort names.txt
uniq	Remove duplicates	`sort names.txt
cut	Extract columns	cut -d',' -f2 file.csv
head	First n lines	head -n 5 file.txt
tail	Last n lines	tail -n 10 file.txt
wc	Count lines/words/characters	wc -l file.txt

Real-Life Example

Filtering production logs:

```
tail -f /var/log/app.log | grep "ERROR" | wc -l
```

- Continuously monitor log errors
- Count number of errors in real-time

Technical Example

List unique users from a log file:

```
cat access.log | cut -d' ' -f1 | sort | uniq
```

- cut → extract IP
- sort → sort IPs
- uniq → remove duplicates

Interview Explanation (How to Say It)

"Pipes and filters allow Linux commands to be combined so the output of one command becomes input to another, enabling efficient text processing."

Common Interview Questions

- What is a pipe in Linux?
- Name some filter commands.
- How to count errors in a log file?
- Difference between sort and uniq?

Common Mistakes Freshers Make

- Forgetting to sort before uniq
- Using > instead of | for pipes
- Ignoring options like -n or -f
- Overcomplicating simple commands

One-Line Summary

Pipes and filters let Linux users process data efficiently by chaining commands together.

PROCESS MANAGEMENT (Linux)

Definition:
Process Management in Linux is the activity of **viewing, controlling, and monitoring running programs (processes)** on the system.

Why Process Management is Needed (Purpose)

- To monitor **system resource usage**
- To **kill unresponsive processes**
- To **optimize CPU and memory utilization**
- Essential for **production support and server management**

Key Concepts

- **Process:** A running instance of a program
- **PID (Process ID):** Unique identifier for each process
- **Parent Process:** Process that started another process
- **Foreground & Background:** Run process interactively or in background

---

Common Commands

Command	Purpose	Example
ps	Show running processes	ps -ef
top	Real-time process monitoring	top
htop	Interactive process monitoring	htop
kill	Terminate process	kill -9 1234
jobs	List background jobs	jobs
fg	Bring job to foreground	fg %1
bg	Resume job in background	bg %1
nice	Set process priority	nice -n 10 command
renice	Change priority of running process	renice 5 -p 1234

Real-Life Example

- Watching tasks on a busy computer
- Closing a frozen browser window using Task Manager equivalent

---

Technical Example (Production)
Check memory-heavy process and kill it:

```
ps -eo pid,comm,%mem --sort=-%mem | head -n 5
kill -9 2345
```

Interview Explanation (How to Say It)
“Process management in Linux involves monitoring, controlling, and optimizing running programs using commands like ps, top, kill, and nice.”

Common Interview Questions

- What is a PID?
- Difference between kill and kill -9?
- How to check CPU/memory usage of a process?
- Difference between foreground and background process?

Common Mistakes Freshers Make

- Killing wrong process
- Using kill without checking PID
- Confusing jobs vs ps

- Ignoring process priority

One-Line Summary
Process management controls running programs to optimize system performance and resource usage.

NETWORK COMMUNICATION and UTILITIES (Linux)

Definition:
Network communication and utilities in Linux are tools and commands used to **monitor, test, and manage network connections** on a system.

Why Network Utilities Are Needed (Purpose)
- To **check connectivity** between systems
- To **diagnose network issues**
- To manage **network configurations**
- Essential for **production support, DevOps, and server management**

Key Network Concepts
- **IP Address:** Unique address of a system
- **Port:** Endpoint for communication
- **Ping:** Test connectivity
- **DNS:** Domain name resolution
- **Sockets:** Connection endpoints

Common Linux Network Commands

Command	Purpose	Example
ping	Test connectivity	ping google.com
ifconfig / ip a	View network interfaces	ip a
netstat	View network connections	netstat -tulnp
ss	Socket statistics	ss -tulw
traceroute	Trace path to host	traceroute google.com
nslookup	DNS lookup	nslookup google.com
dig	Detailed DNS query	dig google.com
curl	Test HTTP request	curl https://example.com
wget	Download file from internet	wget https://example.com/file.zip
scp	Copy files over SSH	scp file.txt user@remote:/path
ssh	Remote login	ssh user@192.168.1.10

Real-Life Example
- Ping a server to check if it’s online
- Use traceroute to find network delay
- Copy files from local to remote server using scp

Technical Example (Production)
Check open ports and network connections:

```
ss -tuln
netstat -tulnp | grep 80
ping -c 4 192.168.1.10
```

Interview Explanation (How to Say It)
"Linux network utilities help monitor and troubleshoot connectivity, manage IPs, test network paths, and transfer files between systems."

Common Interview Questions

- How to check connectivity to a server?
- Difference between ifconfig and ip a?
- How to check open ports?
- Difference between ping and traceroute?

Common Mistakes Freshers Make

- Forgetting -c with ping for limited attempts
- Using netstat without sudo for all ports
- Confusing scp and rsync
- Not understanding IP vs hostname

One-Line Summary
Network utilities in Linux monitor, troubleshoot, and manage connectivity efficiently.