Use Case	Command	Description
Background Process	sleep 60 &	Start a background process
Background Process	echo $!	Capture PID of background process
Background Process	ps -p $!	Verify background process is running
Background Process	kill $!	Terminate background process
Process Monitoring	ps -ef	Full-format listing of all processes
Process Monitoring	ps aux	Detailed info of all user processes
Process Monitoring	top	Real-time system resource usage
Process Monitoring	htop	User-friendly version of top
Process Control	kill -9	Forcefully terminate a process
Process Control	kill -15	Gracefully terminate a process
Process Priority	nice -n 10 sleep 100	Start process with given priority
Process Priority	renice -n 5 -p 1234	Change priority of a running process
Disk Monitoring	df -h	Disk usage in human-readable format
Memory Monitoring	free -m	Memory usage in MB
Network Monitoring	netstat -ntlp	Active TCP connections and listening ports
Kernel Logs	dmesg	View kernel ring buffer messages
Process Search	pgrep firefox	Find PID by process name
Process Control	pkill firefox	Kill process by name
Process Monitoring	pstree	Tree structure of running processes
System Control	shutdown -r now	Shutdown and reboot immediately
System Control	reboot	Restart the system
System Control	halt	Stop all processes
Scheduling	crontab -l	List scheduled jobs
Scheduling	crontab -e	Edit scheduled jobs
Scheduling	crontab -r	Remove scheduled jobs
Permissions	chmod g-r file.txt	Remove group read permission
Permissions	setfacl -m u:bob:rw file.txt	Set ACL for user bob
Permissions	chmod u+s file	Set SUID on a file
Permissions	chmod g+s file	Set SGID on a file
Permissions	chmod +t /shared	Set sticky bit on a directory
File Search	find / -name "file.txt"	Search file by name
File Search	locate config	Fast file search using database
Session Management	screen -S session	Start a screen session
Session Management	tmux new -s devops	Start a tmux session
Redirection	echo 'hello' > file.txt	Overwrite output to file
Redirection	echo 'world' >> file.txt	Append output to file
Redirection	sort < file.txt	Input redirection from file
Redirection	command 2> error.txt	Redirect stderr to file
Redirection	command &> all.txt	Redirect stdout and stderr
Redirection	ls -l | tee file.txt	View and save output simultaneously
Pipes	ps aux | grep nginx | wc -l	Chained command example
File Operations	ls -l	List files with details
File Operations	cat file.txt	Display file contents
File Operations	head -n 10 file.txt	Show first 10 lines of a file
File Operations	tail -n 10 file.txt	Show last 10 lines of a file
File Operations	tail -f /var/log/syslog	Follow log updates in real-

File Operations	tail -f /var/log/syslog	Follow log updates in real-time
File Operations	cp file1 file2	Copy a file
File Operations	mv old.txt new.txt	Rename or move a file
File Operations	rm file.txt	Delete a file
File Operations	mkdir newdir	Create a directory
File Operations	rm -rf directory/	Delete a directory recursively
Customization	alias ll='ls -lah'	Create a shortcut for a command
Customization	source ~/.bashrc	Reload shell configuration
Variables	export VAR=value	Set an environment variable
Variables	unset VAR	Unset a variable
Variables	readonly VAR=value	Make variable read-only
Variables	echo $VAR	Display variable value
Conditionals	if [-f file]; then ...	Check if file exists
Conditionals	[$a -eq $b]	Numeric comparison
Conditionals	["$a" = "$b"]	String comparison
Redirection	tee file.txt	Write output to file and display it
Redirection	wc -l < file.txt	Count lines using input redirection
Text Processing	grep 'pattern' file.txt	Search text in files
Text Processing	awk '{print $1}'	Pattern scanning and processing
Text Processing	cut -d':' -f1 /etc/passwd	Cut specific fields
Permissions	chmod 777 /shared	Set full permissions on a directory
Permissions	ls -l /bin/ping	Check SUID on binary
Permissions	ls -l /srv/devops	Check SGID on directory

Log Analyzer and Archiver Script

Scenario:

Your team stores logs in /var/log/myapp/. Your job is to create a script that:

1. Checks if the log directory /var/log/myapp/ exists.

2. If it exists, count how many .log files are there. Take user's input for the backup directory

3. For each .log file:

   * Print the filename.

   * Print how many lines contain the word ERROR.

4. Archive all .log files into a .tar.gz file with the current date in its name (e.g.,

logs_2025-12-19.tar.gz).

5. Move the archive to backup directory and delete the original .log files after successful archiving.

6. Log all outputs and errors to a file named log_cleanup.log.

if [-d "$DIR"] 1. Check if /var/log/myapp exists

wc-l, find 2. Count .log files

for FILE in *.log 3. Loop through files

4. Analyze the content of files

echo: Print name

grep | wc-l — Print line count with word 'error'

↳ grep -c

tar-cvzf, date 5. Archive files

6. Move archived files to backup dir

mv, rm & clean up

7. Log everything

exec > logfile 2>&1

1.1 - Take user's input to take backup dir's name

read -p "Enter directory name" LOG_DIR

Shebang — /bin/sh or /bin/zsh or /sbin/python3

SKELETON

```
#!/bin/bash

# Define variables

# Take user's input for bkp dir

if [ -- ] ; then
    # Count file logic

    for File in .... ; do
        # Print filename
        # Count ERROR lines
    done

    # Archive logic

    # Move archive to bkp dir
    # Delete original logs
fi
```

if [ ] ; then

if

→ name = "Nashiti"
echo $name

String Comparisons

Use Case	Syntax	Example
Equals	["$a" = "$b"]	["$USER" = "root"]
Not Equals	["$a" != "$b"]	["$USER" != "admin"]
Is Empty	[-z "$var"]	[-z "$INPUT"]
Is Not Empty	[-n "$var"]	[-n "$RESULT"]

Numeric Comparisons

– No double quotes

Use Case	Syntax	Example
Equal	[$a -eq $b]	[$AGE -eq 30]
Not Equal	[$a -ne $b]	[$COUNT -ne 10]
Greater Than	[$a -gt $b]	[$CPU -gt 80]
Less Than	[$a -lt $b]	[$AGE -lt 18]
Greater or Equal	[$a -ge $b]	[$X -ge 100]
Less or Equal	[$a -le $b]	[$Y -le 5]

File Tests

– Double Quotes

Use Case	Syntax	Example
File exists	[-f "$file"]	[-f "data.txt"]
Directory exists	[-d "$dir"]	[-d "/var/log"]
File is readable	[-r "$file"]	[-r "secrets.txt"]
File is writable	[-w "$file"]	[-w "output.log"]
File is executable	[-x "$script"]	[-x "deploy.sh"]
Path exists (file/dir)	[-e "$path"]	[-e "/etc/passwd"]
File is non-empty	[-s "$file"]	[-s "logfile.log"]

Command Exit Status

Use Case	Syntax	
Last command succeeded	[$? -eq 0]	
Run command in if directly	if grep "ERROR" logfile.log > /dev/null; then	

LOGICAL OPERATORS

AND

```
if [ "$a" = "yes" ] && [ "$b" = "confirmed" ]; then
```

OR

```
if [ "$USER" = "admin" ] || [ "$USER" = "root" ]; then
```

NOT

```
if [ ! -f "error.log" ]; then
```

Brackets	Purpose	Used In	Returns Value?	Variable Scope	Example Use Case
[ ]	Test condition (POSIX)	if [condition]	Yes (0 or 1)	Current shell	["$a" = "b"], [-f file.txt]
[[ ]]	Advanced test (Bash-only)	if [[condition]]	Yes (0 or 1)	Current shell	[[$a == b]], [[$file == *.txt]]
(( ))	Arithmetic/logic test	if ((condition))	Yes (0 or 1)	Current shell	((a > 10)), ((x % 2 == 0))
( )	Subshell (child shell)	if (commands)	if exit = 0	Subshell (isolated)	if (cd /tmp && ls); then echo ok; fi
{ }	Command grouping	if { commands; }	if exit = 0	Current shell	if { cd /tmp; ls; }; then echo ok; fi

For loop Variations

Type	Syntax Example	Use Case
List loop	for x in a b c; do	Known values or file names
Brace loop	for i in {1..5}; do	Simple numeric or letter range
C-style loop	for ((i=0; i<5; i++))	Advanced counters and math
File loop	for f in *.txt; do	Iterating over matched files
Command output	for u in $(cut -d: -f1 /etc/passwd)	System commands with clean output
Array loop	for i in "${arr[@]}"	Structured values in a list
While-read loop	while read line; do	Reading files line-by-line safely

```
root@ip-172-31-27-45:~# cat finalscript.sh
#!/bin/bash
LOG_DIR="/var/log/myapp"
DATE=$(date +%Y-%m-%d)
ARCHIVE_NAME="logs_$DATE.tar.gz"
LOG_FILE="log_cleanup.log"
read -p "Enter bkp dire name" BACKUP_DIR

exec > "$LOG_FILE" 2>&1

if [ -d "$LOG_DIR" ]; then
        echo "LOG Dir Found"
        LOG_COUNT=$(find "$LOG_DIR" -type f -name "*.log" | wc -l)
        echo "Log Count = $LOG_COUNT"

        for FILE in "$LOG_DIR"/*.log ; do
            echo "Current File: $FILE"
            ERROR_COUNT=$(grep -c "ERROR" "$FILE")
            echo "$ERROR_COUNT errprs in $FILE"
        done

        tar -cvzf "$ARCHIVE_NAME" -C "$LOG_DIR" *.log

        mkdir -p $BACKUP_DIR
        mv "$ARCHIVE_NAME" "$BACKUP_DIR/"
        rm "$LOG_DIR"/*.log
else
        echo "$LOG_DIR does not exist"


fi
```

apt install acl