# What is a Shell?

## Introduction

A shell is a command-line interpreter that provides an interface between the user and the Linux operating system. It allows users to execute commands, run scripts, and manage system resources. The shell interprets user input, processes commands, and returns output. Some popular shell types in Linux include:

* **Bash (Bourne Again Shell)** - Default shell on most Linux distributions.
* **Zsh (Z Shell)** - An extended version of Bash with improved features.
* **Fish (Friendly Interactive Shell)** - User-friendly and feature-rich shell.
* **Tcsh (TENEX C Shell)** - Based on C Shell (csh), commonly used for scripting.
* **Ksh (Korn Shell)** - A powerful shell often used in enterprise environments.

## What is Bash Scripting?

Bash scripting is the process of writing scripts using the Bash shell to automate tasks. A Bash script is a file containing a series of commands that are executed sequentially. Scripts are used to automate repetitive tasks, manage system operations, and perform complex system configurations.

### Why Use Bash Scripts?

* **Automation** - Reduces manual intervention for repetitive tasks.
* **System Administration** - Helps in managing users, processes, logs, and system updates.
* **Backup and Maintenance** - Enables scheduled backups and system cleanup.
* **Task Scheduling** - Can be executed via cron jobs or systemd timers.
* **Security & Access Control** - Helps in restricting or granting access using scripts.

# Common Practical Examples

### 1. Check system uptime

uptime

### 2. Display logged-in users

who

### 3. Show disk usage

df -h

### 4. Monitor memory usage

free -m

### 5. List running processes

ps aux

### 6. Find a specific process

ps aux | grep apache

### 7. Kill a process by PID

kill -9 <PID>

### 8. Check network connectivity

ping -c 4 google.com

### 9. Display open ports

netstat -tulnp

### 10. Check system logs

dmesg | tail -50

### 11. Create a simple Bash script

echo "#!/bin/bash" > myscript.sh  
echo "echo Hello, World!" >> myscript.sh  
chmod +x myscript.sh  
./myscript.sh

### 12. Automate file backup

tar -czvf backup.tar.gz /home/user/

### 13. Find and delete old files

find /var/log -name "\*.log" -type f -mtime +30 -exec rm {} \;

### 14. Loop through files in a directory

for file in /path/to/directory/\*; do echo "Processing $file"; done

### 15. Read user input in a script

read -p "Enter your name: " name  
echo "Hello, $name!"

### 16. Conditional statement in Bash

if [ -f /etc/passwd ]; then echo "File exists"; else echo "File not found"; fi

### 17. Running a command as another user

sudo -u user command

### 18. Schedule a cron job to run daily at midnight

crontab -e  
0 0 \* \* \* /path/to/script.sh

### 19. Monitor system resource usage

top

### 20. Log script output to a file

./script.sh > output.log 2>&1

# Additional Notes

* Always use #!/bin/bash at the beginning of Bash scripts to ensure execution in the correct shell.
* Scripts should have execute permissions (chmod +x script.sh).
* Use set -e to stop script execution on errors.
* Use trap to handle unexpected interruptions.
* Use comments (#) to document scripts for maintainability.

# Configuration

### Setting Bash as Default Shell

chsh -s /bin/bash

### Creating a Basic Bash Profile (~/.bashrc)

echo 'export PS1="\u@\h:\w$ "' >> ~/.bashrc  
echo 'alias ll="ls -la"' >> ~/.bashrc  
source ~/.bashrc

### Creating a System-Wide Bash Script

1. Create the script:

echo "#!/bin/bash" > /usr/local/bin/myscript  
echo "echo System Info: $(uname -a)" >> /usr/local/bin/myscript

1. Assign execution rights:

chmod +x /usr/local/bin/myscript

1. Run the script:

./myscript  
source myscript # this option does not require +x rights