**✅ 1. Introduction to Shell Scripting**

**🔹 What is Shell Scripting?**

Shell scripting is writing a sequence of commands in a text file to be executed by the Unix/Linux shell. It automates repetitive tasks, handles file processing, monitors systems, and more.

* Shell: Command-line interface (bash, zsh, sh, etc.)
* Shell script: A text file with .sh extension (optional)

**🔹 Why Shell Scripting?**

* Automate system tasks (e.g., backups, deployments)
* Process log files
* Manage users and permissions
* Monitor services

**✅ 2. First Shell Script**

**🔹 Shebang**

The **first line** of a shell script defines which shell should execute the script:

bash

#!/bin/bash

This is called a **shebang**. It tells the system to use bash to interpret the script.

**🔹 Sample Script**

bash

#!/bin/bash

echo "Hello, World!"

**🔹 How to Run the Script**

1. **Make it executable**:

bash

chmod +x myscript.sh

1. **Execute it**:

bash

./myscript.sh

**✅ 3. Variables in Shell**

**🔹 What are Variables?**

Variables are used to **store data** (strings, numbers) to be used later in scripts.

**🔹 Declaring Variables**

bash

name="Alice" # No space around '='

**🔹 Using Variables**

bash

echo "Hello, $name"

Use double quotes to preserve spaces and special characters.

**🔹 Reading User Input**

bash

read -p "Enter your name: " username

echo "Hi $username!"

**✅ 4. Data Types**

Shell does not have strong typing like other languages. Everything is treated as **strings**. You can perform arithmetic using built-in tools (expr, let, $(( ))).

**🔹 Arithmetic Example**

bash

a=10

b=5

sum=$((a + b))

echo "Sum is $sum"

**✅ 5. Conditional Statements**

**🔹 Purpose**

Used to execute commands based on **conditions**.

**🔹 if-else Syntax**

bash

if [ condition ]; then

# code

elif [ condition ]; then

# code

else

# code

fi

**🔹 Example**

bash

age=20

if [ $age -ge 18 ]; then

echo "Adult"

else

echo "Minor"

fi

**🔹 Common Operators**

| **Type** | **Operators** | **Example** |
| --- | --- | --- |
| Integer | -eq, -ne, -gt, -lt, -ge, -le | [ $a -gt 5 ] |
| String | =, !=, -z, -n | [ "$str" = "abc" ] |
| File Test | -f, -d, -r, -w, -x, -e | [ -f file.txt ] |

**✅ 6. Loops**

**🔹 for Loop**

Used to repeat tasks a specific number of times.

bash

for i in 1 2 3; do

echo "Value: $i"

done

**🔹 while Loop**

Repeats while the condition is true.

bash

count=1

while [ $count -le 5 ]; do

echo "Count: $count"

((count++))

done

**🔹 until Loop**

Repeats until the condition becomes true.

bash

n=1

until [ $n -gt 3 ]; do

echo "n is $n"

((n++))

done

**✅ 7. Functions**

**🔹 What is a Function?**

A **function** is a reusable block of code that performs a specific task.

**🔹 Defining a Function**

bash

greet() {

echo "Hello $1"

}

greet "World"

* $1, $2 refer to passed arguments

**✅ 8. Arrays**

**🔹 Declaring an Array**

bash

fruits=("apple" "banana" "cherry")

**🔹 Accessing Elements**

bash

echo ${fruits[0]} # apple

**🔹 Looping Through Arrays**

bash

for fruit in "${fruits[@]}"; do

echo "$fruit"

done

**🔹 Array Length**

bash

echo ${#fruits[@]} # 3

**✅ 9. String Manipulation**

bash

str="HelloWorld"

echo ${#str} # Length

echo ${str:0:5} # Substring (Hello)

echo ${str/World/Shell} # Replace (HelloShell)

**✅ 10. File I/O**

**🔹 Write to File**

bash

echo "Hello" > file.txt # Overwrites

echo "Next line" >> file.txt # Appends

**🔹 Read from File**

bash

while read line; do

echo "$line"

done < file.txt

**✅ 11. Text Processing Tools**

**🔹 grep: Search in files**

bash

grep "error" log.txt

**🔹 sed: Stream editor**

bash

sed 's/old/new/' file.txt # Replace first occurrence

sed 's/old/new/g' file.txt # Replace all

**🔹 awk: Pattern scanning**

bash

awk '{print $1}' file.txt

**✅ 12. Command Substitution**

bash

date\_today=$(date)

echo "Today is $date\_today"

**✅ 13. Error Handling**

**🔹 Exit Status**

bash

command

if [ $? -eq 0 ]; then

echo "Success"

else

echo "Failure"

fi

* $? stores the exit status of the last command

**✅ 14. Background Processes**

bash

sleep 10 &

jobs # List jobs

fg %1 # Bring back to foreground

**✅ 15. Scheduling with cron**

**🔹 Edit crontab**

bash

crontab -e

**🔹 Cron Syntax**

sql

\* \* \* \* \* command

│ │ │ │ │

│ │ │ │ └── Day of Week (0-6)

│ │ │ └──── Month (1-12)

│ │ └────── Day of Month (1-31)

│ └──────── Hour (0-23)

└────────── Minute (0-59)

**🔹 Example**

bash

0 6 \* \* \* /home/user/script.sh

**✅ 16. Script Debugging**

Add this at the top:

bash

set -x # Show commands as they are executed

Or run:

bash

bash -x myscript.sh