Experiment-03

Experiment Name: To know about shell scripting and write a basic program.

1. Understanding Shell Scripting Basics

- **What is Shell Scripting?**
  A shell script is a text file containing commands that are executed by the shell (e.g., `bash`). These scripts automate tasks, making them faster and more efficient to run.
- **Script File Structure**:
  Shell scripts typically start with `#!/bin/bash`, which specifies the shell type (`bash` in this case) to interpret the script.

Why we need to know about shell scripting?

Shell scripting gives CSE students practical experience with operating system concepts, providing skills that are critical for both academic understanding and future careers in technology.

- **Automation**: Saves time by automating repetitive tasks like backups, updates, and monitoring.
- **System Administration**: Essential for managing user accounts, permissions, and system health on UNIX/Linux systems.
- DevOps Essentials: Vital for setting up CI/CD pipelines and infrastructure management in DevOps.
- Resource Efficiency: Lightweight and faster execution compared to GUI tools, ideal for system tasks.
- Enhanced Control: Provides flexibility with direct OS control, conditional logic, and command chaining.
- Integration: Works well with other tools and languages, creating versatile workflows across systems.
- Career Advantage: Highly valued in IT roles such as system administration, DevOps, and data analysis.

2. Creating a Shell Script

- **Step 1**: Open a text editor (e.g., `nano`, `vim`) and create a new file.
- **Step 2**: Begin with the "shebang" line `#!/bin/bash` to specify the interpreter.

- **Step 3**: Write commands line by line, just as you would enter them in the terminal.

  Example of a simple shell script (`hello.sh`):

```
#!/bin/bash

echo "Hello, World!"
```

3. Making the Script Executable

- **Step 1**: Save the file and exit the text editor.
- **Step 2**: In the terminal, make the script executable by running:

```
chmod +x hello.sh
```

4. Running the Script

- Execute the script by typing:

```
./hello.sh
```

5. Using Variables

- **Defining Variables**:
  Shell scripts allow variables to store and reuse values. Variables do not need a `data type` declaration.
- **Using Variables**: Use `$` to refer to variable values.

  Example:

```
#!/bin/bash

name="Linux User"

echo "Hello, $name!"
```

6. Adding Control Structures

- **If Statements**: Adds conditional logic to the script.

  Example:

```
#!/bin/bash

num=10

if [ $num -gt 5 ]; then

   echo "$num is greater than 5"
```

```
fi
```

Loops: Automate repetitive tasks with loops (`for`, `while`).

Example (`for` loop):

```
#!/bin/bash
for i in {1..5}; do
    echo "Iteration $i"
done
```

7. Using Functions

- Functions group commands into reusable units.

  Example:

```
#!/bin/bash
greet() {
    echo "Hello, $1!"
}
greet "User"
```

8. Accepting User Input

- Use `read` to get input from the user.

  Example:

```
#!/bin/bash
echo "Enter your name:"
read name
echo "Hello, $name!"
```

9. Working with Files and Directories

- Use commands within scripts to handle files (`cp`, `mv`, `rm`, etc.) and directories.

  Example:

```
#!/bin/bash
mkdir test_directory
touch test_directory/newfile.txt
echo "File created!"
```

Program:

a) Check if a Number is Even or Odd

```
#!/bin/bash
# Program to check if a number is even or odd

echo "Enter a number:"
read num

if [ $((num % 2)) -eq 0 ]; then
    echo "$num is even"
else
    echo "$num is odd"
fi
```

b) Check if a Year is a Leap Year

```
#!/bin/bash
# Program to check if a year is a leap year

echo "Enter a year:"
read year

if (( year % 400 == 0 )); then
```

```
    echo "$year is a leap year"
elif (( year % 100 == 0 )); then
    echo "$year is not a leap year"
elif (( year % 4 == 0 )); then
    echo "$year is a leap year"
else
    echo "$year is not a leap year"
fi
```

c) Find the Factorial of a Number

```
#!/bin/bash
# Program to find the factorial of a number

echo "Enter a number:"
read num
factorial=1

for (( i=1; i<=num; i++ ))
do
    factorial=$((factorial * i))
done

echo "The factorial of $num is $factorial"
```

d) Swap Two Integers

```
#!/bin/bash
# Program to swap two integers

echo "Enter first integer:"
```

```
read a
echo "Enter second integer:"
read b

echo "Before swapping: a = $a, b = $b"
# Swap the values
temp=$a
a=$b
b=$temp

echo "After swapping: a = $a, b = $b"
```

Practice:

1. Write a Shell program to find the smallest digit from a number.
2. Write a Shell program to find the second largest digit from a number.
3. Write a Shell program to find the sum of digits of a number.
4. Write a Shell program to check the given integer is Armstrong number or not.