# [590029302]Exp[6]_ScriptLog

## Experiment 6: Shell Loops

#### Name: Tanmay Amit Verma Roll No.: 590029302 Date: 2025-09-23

#### Aim:

- To understand and implement shell loops (`for`, `while`, `until`) in Bash.
- To practice loop control constructs (`break`, `continue`) and loop-based file processing.

#### Requirements

- A Linux system with bash shell.
- A text editor (nano, vim) and permission to create and execute shell scripts.

## Theory

Loops allow repeated execution of commands until a condition is met. Common loop constructs in Bash include `for` (iterate over items), `while` (repeat while condition true), and `until` (repeat until condition becomes true). Loop control statements like `break` and `continue` change the flow inside loops. Loops are essential for automating repetitive tasks such as processing multiple files, generating sequences, and collecting user input.

## Procedure & Observations

## Exercise 1: Palindrome Check

#### Task Statement:

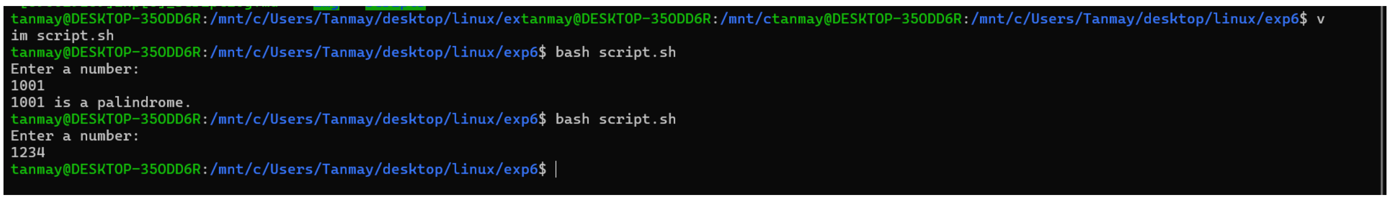
Write a `while` loop that checks whether a number is a palindrome or not.

#### Command(s):

```
#!/bin/bash
echo "Enter a number: "
read num
rev=0
temp=$num

while [ $temp -gt 0 ]
do
    digit=$((temp % 10))
    rev=$((rev * 10 + digit))
    temp=$((temp / 10))
```

```
done

if [ $num -eq $rev ]
then
    echo "$num is a palindrome."
else
    echo "$num is not a palindrome."
fi
```

### Output:



```
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/extanmay@DESKTOP-350DD6R:/mnt/ctanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ v
im script.sh
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash script.sh
Enter a number:
1001
1001 is a palindrome.
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash script.sh
Enter a number:
1234
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ |
```

---

## Exercise 2: GCD and LCM check

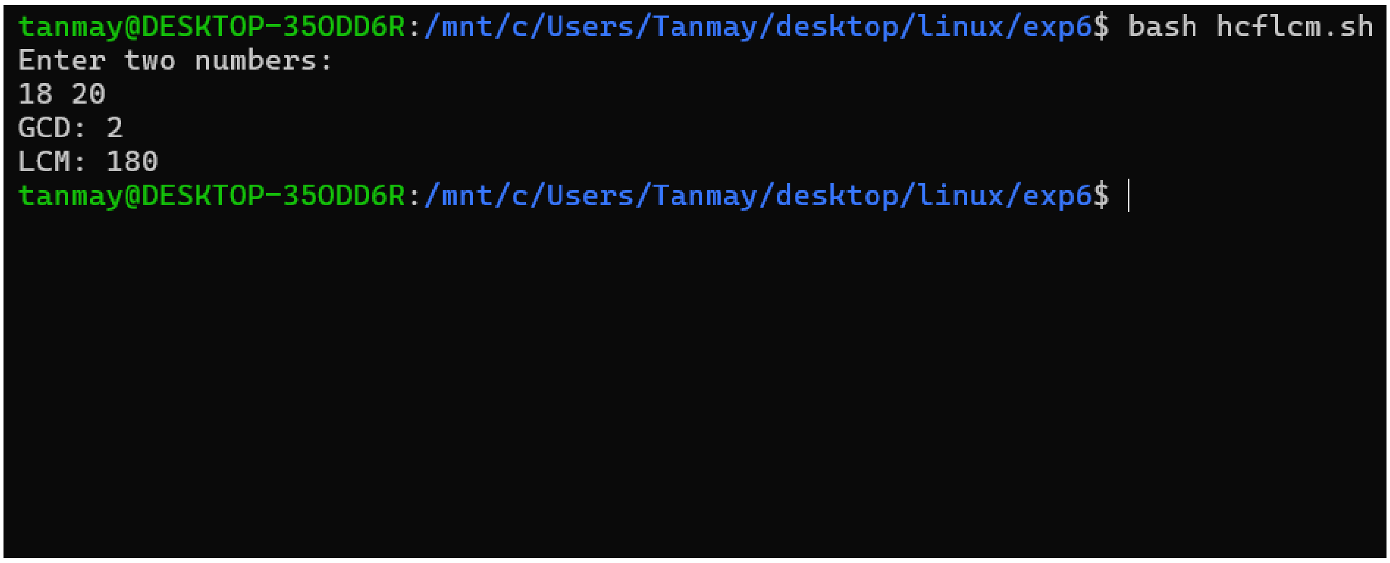
### Task Statement:

Apply a Euclidean algorithm for GCD and LCM in bash script using loops.

### Command(s):

```
#!/bin/bash
echo "Enter two numbers: "
read a b

x=$a
y=$b
while [ $y -ne 0 ]
do
    temp=$y
    y=$((x % y))
    x=$temp
done
gcd=$x

lcm=$(( (a * b) / gcd ))

echo "GCD: $gcd"
echo "LCM: $lcm"
```

### Output:



```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash hcflcm.sh
Enter two numbers:
18 20
GCD: 2
LCM: 180
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ |
```

## Exercise 3: Sorting Numbers

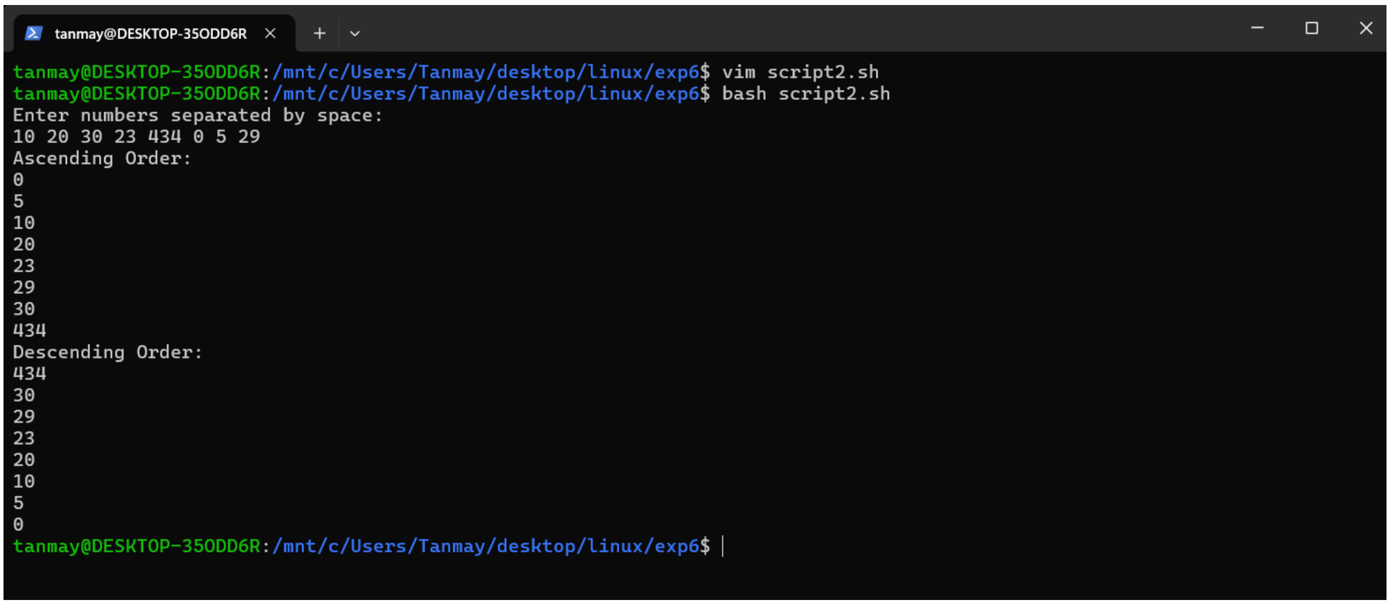
### Task Statement:

Use arithmetic C-style loop for numeric iteration.

### Command(s):

```
#!/bin/bash
echo "Enter numbers separated by space: "
read -a arr

echo "Ascending Order: "
printf "%s\n" "${arr[@]}" | sort -n

echo "Descending Order: "
printf "%s\n" "${arr[@]}" | sort -nr
```

### Output:



```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ vim script2.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash script2.sh
Enter numbers separated by space:
10 20 30 23 434 0 5 29
Ascending Order:
0
5
10
20
23
29
30
434
Descending Order:
434
30
29
23
20
10
5
0
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ |
```

## Assignment 1

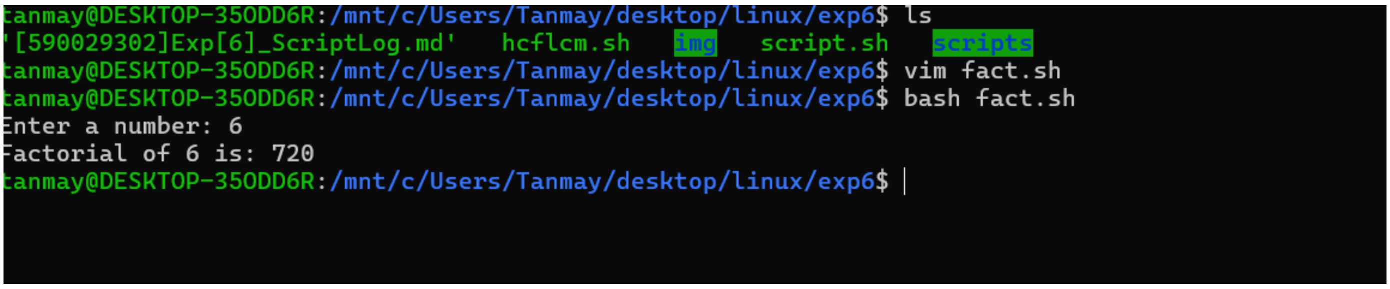
### Task Statement:

Write a function to calculate the factorial of a number using a loop

### Command(s):

```
#!/bin/bash

echo -n "Enter a number: "
read num

fact=1


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

echo "Factorial of $num is: $fact"
```

### Output:



```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ ls
'[590029302]Exp[6]_ScriptLog.md'   hcflcm.sh   img   script.sh   scripts
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ vim fact.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash fact.sh
Enter a number: 6
Factorial of 6 is: 720
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ |
```

## Task 2

### Task Statement:

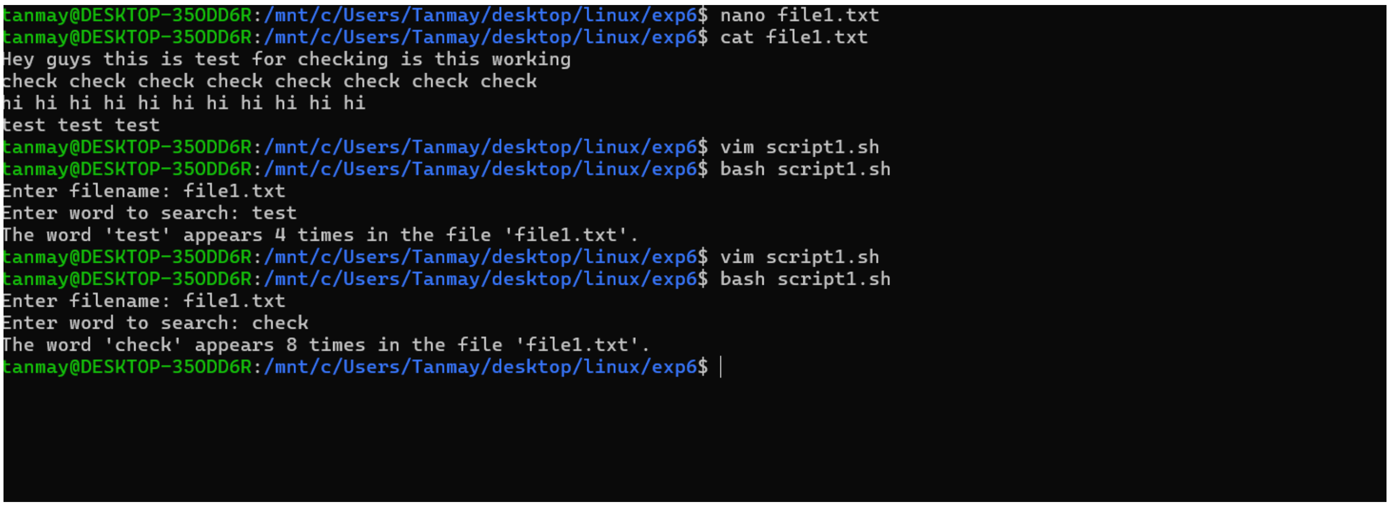
Write a scripts that reads a filename and counts how many times a given word appears in it.

### Command(s):

```
#!/bin/bash

echo -n "Enter filename: "
read filename

if [[ ! -f "$filename" ]]; then
```

```
    echo "File does not exist!"
    exit 1
fi

echo -n "Enter word to search: "
read word

count=$(grep -o -w "$word" "$filename" | wc -l)

echo "The word '$word' appears $count times in the file '$filename'."
```

### Output:



```
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ nano file1.txt
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ cat file1.txt
Hey guys this is test for checking is this working
check check check check check check check check
hi hi hi hi hi hi hi hi hi hi hi
test test test
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ vim script1.sh
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash script1.sh
Enter filename: file1.txt
Enter word to search: test
The word 'test' appears 4 times in the file 'file1.txt'.
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ vim script1.sh
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash script1.sh
Enter filename: file1.txt
Enter word to search: check
The word 'check' appears 8 times in the file 'file1.txt'.
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ |
```

---

## Task 3

### Task Statement:

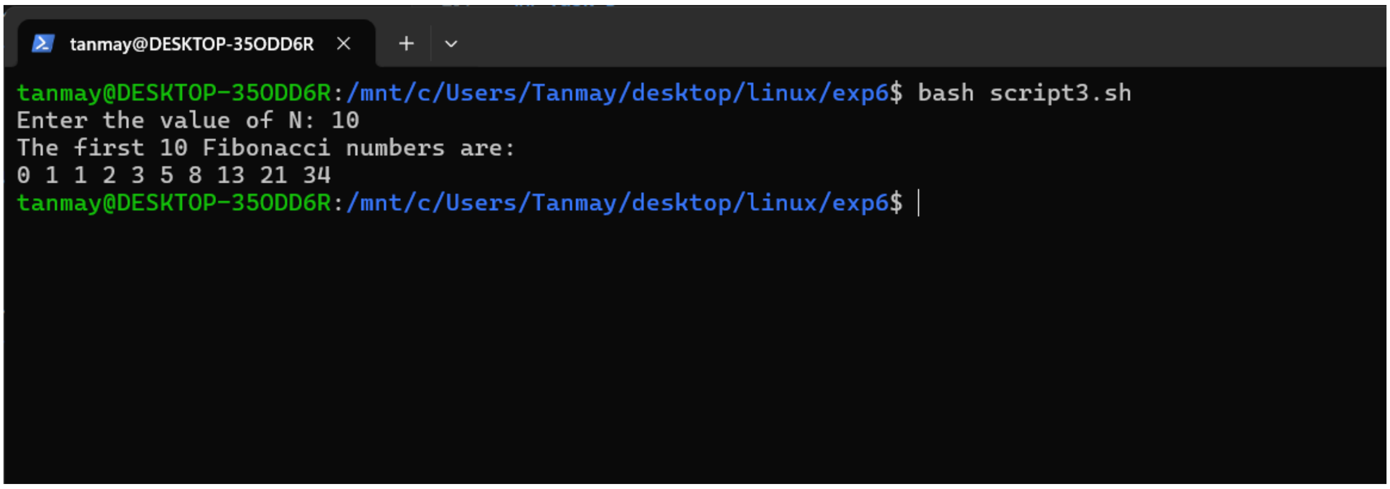
Write a script which generates the first N fibonacci numbers using a while loop.

### Command(s):

```
#!/bin/bash

echo -n "Enter the value of N: "
read N

a=0
b=1
i=1

echo "The first $N Fibonacci numbers are:"

while [ $i -le $N ]
```

```
do
    echo -n "$a "
    fn=$((a + b))
    a=$b
    b=$fn
    i=$((i + 1))
done

echo
```

**Output:**



tanmay@DESKTOP-35ODD6R

```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash script3.sh
Enter the value of N: 10
The first 10 Fibonacci numbers are:
0 1 1 2 3 5 8 13 21 34
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ |
```

---

## Task 4

### Task Statement:

Write a script that validates whether the entered string is a proper email address using a regular expression.

### Command(s):

```
 #!/bin/bash

read -p "Enter an email address: " email

regex='^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'
if [[ $email =~ $regex ]]; then
    echo "✔ Valid email"
else
    echo "✖ Invalid email"
fi
```

### Output:

## Task 5

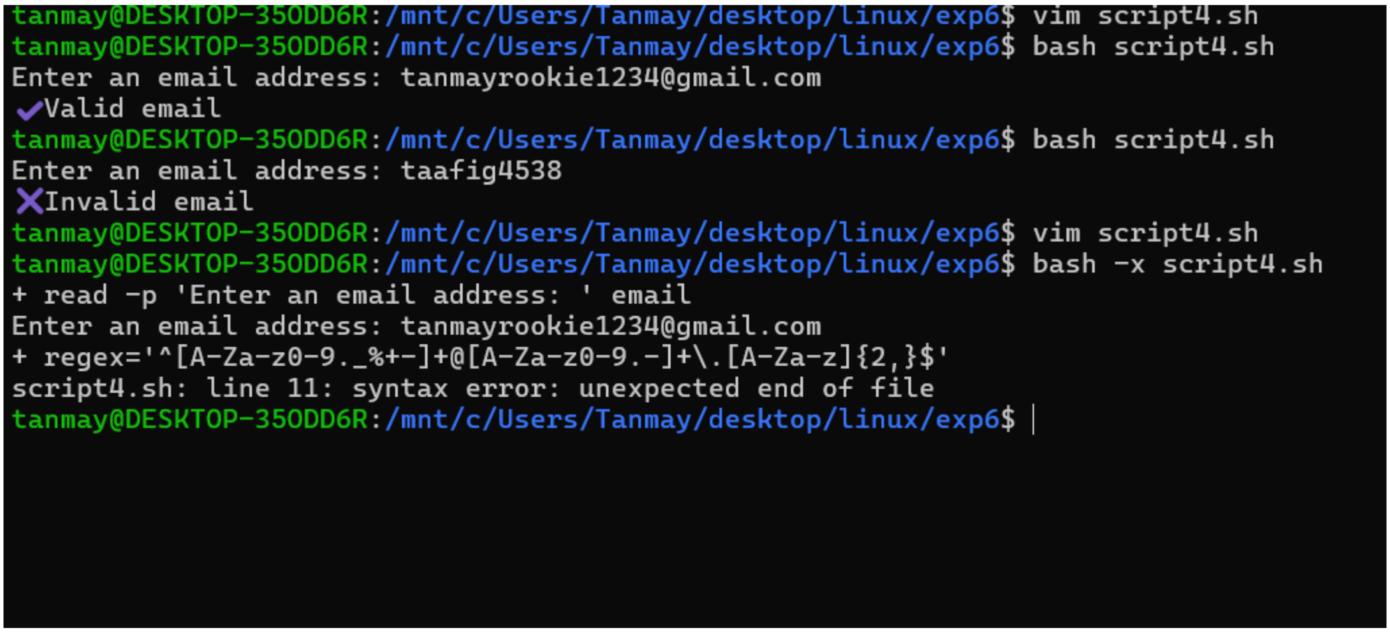
### Task Statement:

Write a script with an intentional error, run it with '"bash -x"' and explain the debug output

**Command(s):**

```
#used same code as above added an intentional error of removing fi in the end while closing the loop
read -p "Enter an email address: " email

regex='^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'
if [[ $email =~ $regex ]]; then
    echo "✓ Valid email"
else
    echo "✗ Invalid email"
i
```

**Output:**



```
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ vim script4.sh
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash script4.sh
Enter an email address: tanmayrookie1234@gmail.com
✔Valid email
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash script4.sh
Enter an email address: taafig4538
✖Invalid email
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ vim script4.sh
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$ bash -x script4.sh
+ read -p 'Enter an email address: ' email
Enter an email address: tanmayrookie1234@gmail.com
+ regex='^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'
script4.sh: line 11: syntax error: unexpected end of file
tanmay@DESKTOP-350DD6R:/mnt/c/Users/Tanmay/desktop/linux/exp6$
```

## Result

- Implemented `for`, `while`, and `until` loops and used loop control statements.
- Practiced reading input, processing files, and nested iteration.

## Challenges Faced & Learning Outcomes

- Challenge 1: Handling spaces and special characters when iterating filenames — learned to use quotes and `read -r`.
- Challenge 2: Remembering arithmetic syntax in Bash — used `(( ))` and `expr` where needed.

### Learning:

- Loops are powerful for automation in shell scripting. Correct quoting and use of control constructs prevent common bugs.

## Conclusion

The lab demonstrated practical loop constructs in Bash for automating repetitive tasks and processing data efficiently.