

[590023785]Exp6_Scrip\[590023785]Exp6_Scrip.md

Experiment 6: Shell Loops

Name: Gayatri Bhatt Roll No.: 590023785 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: Simple for loop

Task Statement:

Write a for loop that prints numbers 1 to 5.

Command(s):

```
for i in 1 2 3 4 5; do
    echo "Number: $i"
done
```

Output:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 6ex1.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 6ex1.sh
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ |
```

Exercise 2: for loop over files

Task Statement:

Process all `.txt` files in a directory and count lines in each.

Command(s):

```
for f in *.txt; do
    echo "File: $f - Lines: $(wc -l < "$f")"
done
```

Output:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 6ex2.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 6ex2.sh
File: script.txt - Lines: 4
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ |
```

Exercise 3: C-style for loop

Task Statement:

Use arithmetic C-style loop for numeric iteration.

Command(s):

```
for ((i=0;i<5;i++)); do
    echo "i=$i"
done
```

Output:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 6ex3.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 6ex3.sh
i=0
i=1
i=2
i=3
i=4
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ |
```

Exercise 4: while loop and reading input

Task Statement:

Write a while loop that reads lines from a file or from user input.

Command(s):

```
# Read from file
while read -r line; do
    echo "Line: $line"
done < sample.txt

# Read from user with exit condition
while true; do
    read -p "Enter a number (0 to exit): " n
    if [[ $n -eq 0 ]]; then
        echo "Exiting..."; break
    fi
    echo "You entered: $n"
done
```

Output:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 6ex4.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 6ex4.sh
6ex4.sh: line 5: sample.txt: No such file or directory
Enter a number (0 to exit): 7
You entered: 7
Enter a number (0 to exit): 89
You entered: 89
Enter a number (0 to exit): 0
Exiting...
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ |
```

Exercise 5: until loop

Task Statement:

Use an until loop to run until a condition becomes true.

Command(s):

```
count=1
until [ $count -gt 5 ]; do
    echo "count=$count"
    ((count++))
done
```

Output:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 6ex5.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 6ex5.sh
count=1
count=2
count=3
count=4
count=5
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ |
```

Exercise 6: break and continue

Task Statement:

Demonstrate break and continue inside a loop.

Command(s):

```
for i in {1..10}; do
    if [[ $i -eq 5 ]]; then
        echo "Reached 5, breaking"; break
    fi
    if (( i % 2 == 0 )); then
        echo "Skipping even $i"; continue
    fi
    echo "Processing $i"
done
```

Output:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 6ex6.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 6ex6.sh
Processing 1
Skipping even 2
Processing 3
Skipping even 4
Reached 5, breaking
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ |
```

Exercise 7: Nested loops

Task Statement:

Create nested loops to generate a multiplication table.

Command(s):

```
for i in {1..3}; do
    for j in {1..3}; do
        echo -n "${i*j}) "
    done
    echo
done
```

Output:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 6ex7.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 6ex7.sh
1 2 3
2 4 6
3 6 9
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ |
```

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:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$ vim 6t1.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$ bash 6t1.sh
Enter a number: 7
Factorial of 7 is: 5040
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$ |
```

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:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe  
riments/Experiments Pdfs$ vim 6t2.sh  
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe  
riments/Experiments Pdfs$ bash 6t2.sh  
Enter filename: gayatri  
File does not exist!  
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe  
riments/Experiments Pdfs$ |
```

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:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ vim 6t3.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ bash 6t3.sh
Enter the value of N: 5
The first 5 Fibonacci numbers are:
0 1 1 2 3
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ |
```

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:

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ vim 6t4.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ bash 6t4.sh
Enter an email address: gayatribhatt@gmail.com
✓Valid email
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ |
```

Task 5

Task Statement:

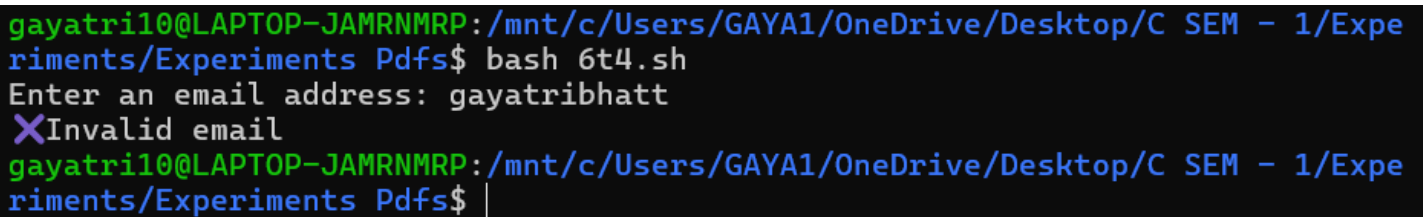
Write a script with an intentional error, run it with "bash -x" and explain the debug output

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:



```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$ bash 6t4.sh
Enter an email address: gayatriibhatt
✗ Invalid email
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$ |
```

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.