

# Experiment 8: Shell Programming

**Name: Biswanbandya Mohanty Roll No.: 590029274 Date: 2025-11-1**

## Aim:

- To extend shell programming concepts by using conditional statements, advanced scripting constructs, and command-line arguments.
- To practice writing scripts that perform decision-making and parameter handling.

## Requirements

- A Linux system with bash shell.
- Text editor and permission to create/execute shell scripts.

## Theory

Conditional execution in shell scripts allows branching logic using `if`, `elif`, `else`, and `case` statements. Scripts can accept command-line arguments using `$1`, `$2`, ... and `$@` for all arguments. Control flow constructs combined with user input and arguments allow dynamic and reusable scripts.

## Procedure & Observations

### Exercise 1: Using `if-else`

#### Task Statement:

Write a script to check whether a given number is positive, negative, or zero.

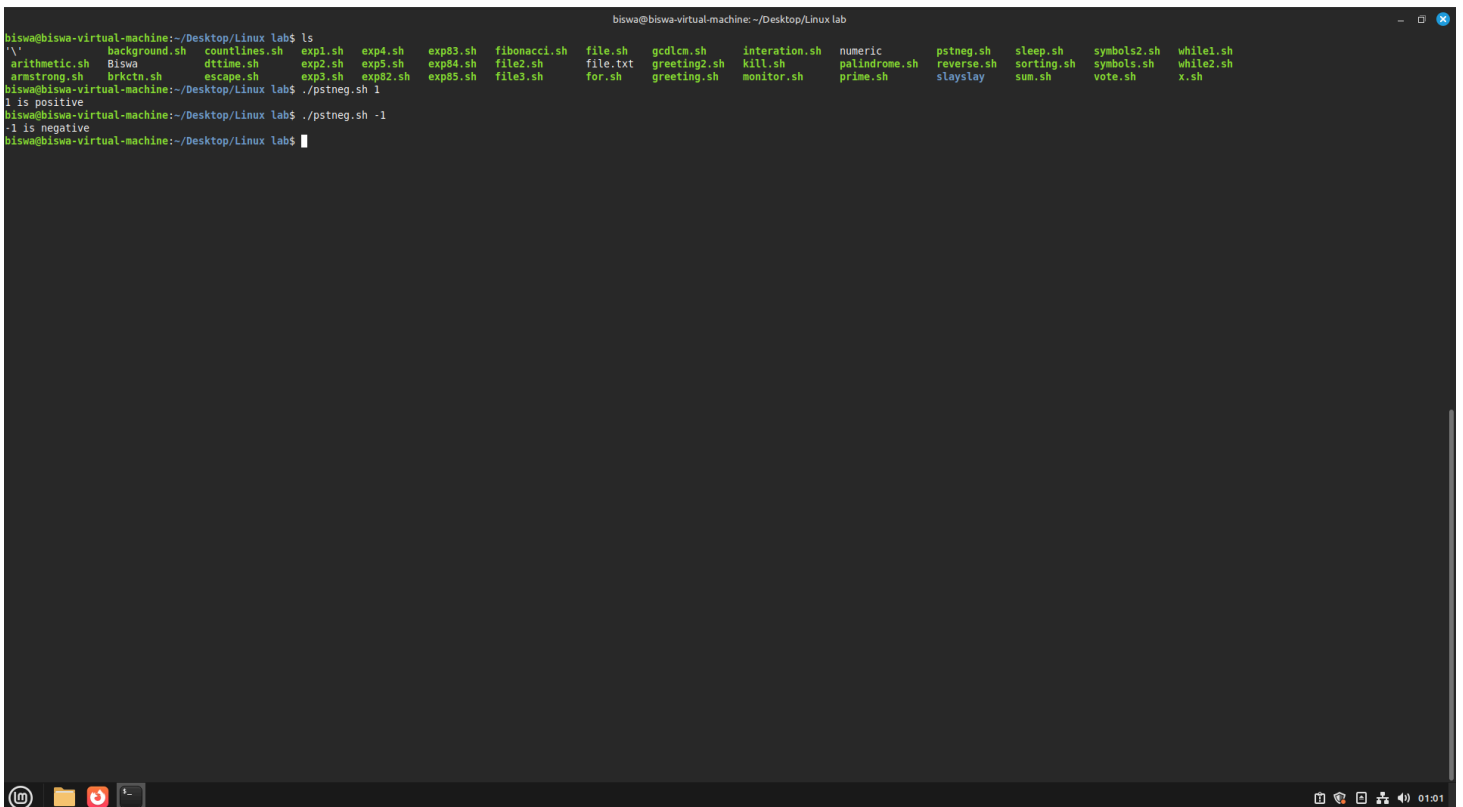
#### Explanation:

We used an `if-elif-else` construct to compare the number against 0.

## Command(s):

```
#!/bin/bash
num=$1
if [ $num -gt 0 ]; then
    echo "$num is positive"
elif [ $num -lt 0 ]; then
    echo "$num is negative"
else
    echo "$num is zero"
fi
```

## Output:

A screenshot of a terminal window titled "biswa@biswa-virtual-machine: ~/Desktop/Linux lab". The terminal shows the execution of a script named "pstneg.sh". The user enters "1" as input, and the script outputs "1 is positive". Then, the user enters "-1" as input, and the script outputs "-1 is negative". The terminal window has a dark background with light green text. At the top, there is a list of various shell scripts available in the directory, such as "arithmetic.sh", "background.sh", "countlines.sh", etc. The bottom of the window shows a standard Linux desktop taskbar with icons for a file manager, a terminal, and other applications, along with a system clock showing "01:01".

```
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ls
\
arithmetic.sh  Biswa      dttime.sh  exp1.sh  exp4.sh  exp83.sh  fibonacci.sh  file.sh  gcdlcm.sh  interation.sh  numeric  pstneg.sh  sleep.sh  symbols2.sh  while1.sh
armstrong.sh  brkctn.sh  escape.sh  exp2.sh  exp5.sh  exp84.sh  file2.sh  file.txt  greeting2.sh  kill.sh  palindrome.sh  reverse.sh  sorting.sh  symbols.sh  while2.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./pstneg.sh 1
1 is positive
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./pstneg.sh -1
-1 is negative
biswa@biswa-virtual-machine:~/Desktop/Linux lab$
```

## Exercise 2: Using case

### Task Statement:

Write a script that takes a character as input and classifies it as vowel, consonant, digit, or special character.

## Explanation:

The `case` statement provides pattern matching for multiple options.

## Command(s):

```
#!/bin/bash
ch=$1
case $ch in
  [aeiouAEIOU]) echo "$ch is a vowel" ;;
  [bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]) echo "$ch is a consonant" ;;
  [0-9]) echo "$ch is a digit" ;;
  *) echo "$ch is a special character" ;;
esac
```

## Output:



## Exercise 3: Command-line arguments

### Task Statement:

Write a script that accepts filename(s) as arguments and prints the number of lines in each file.

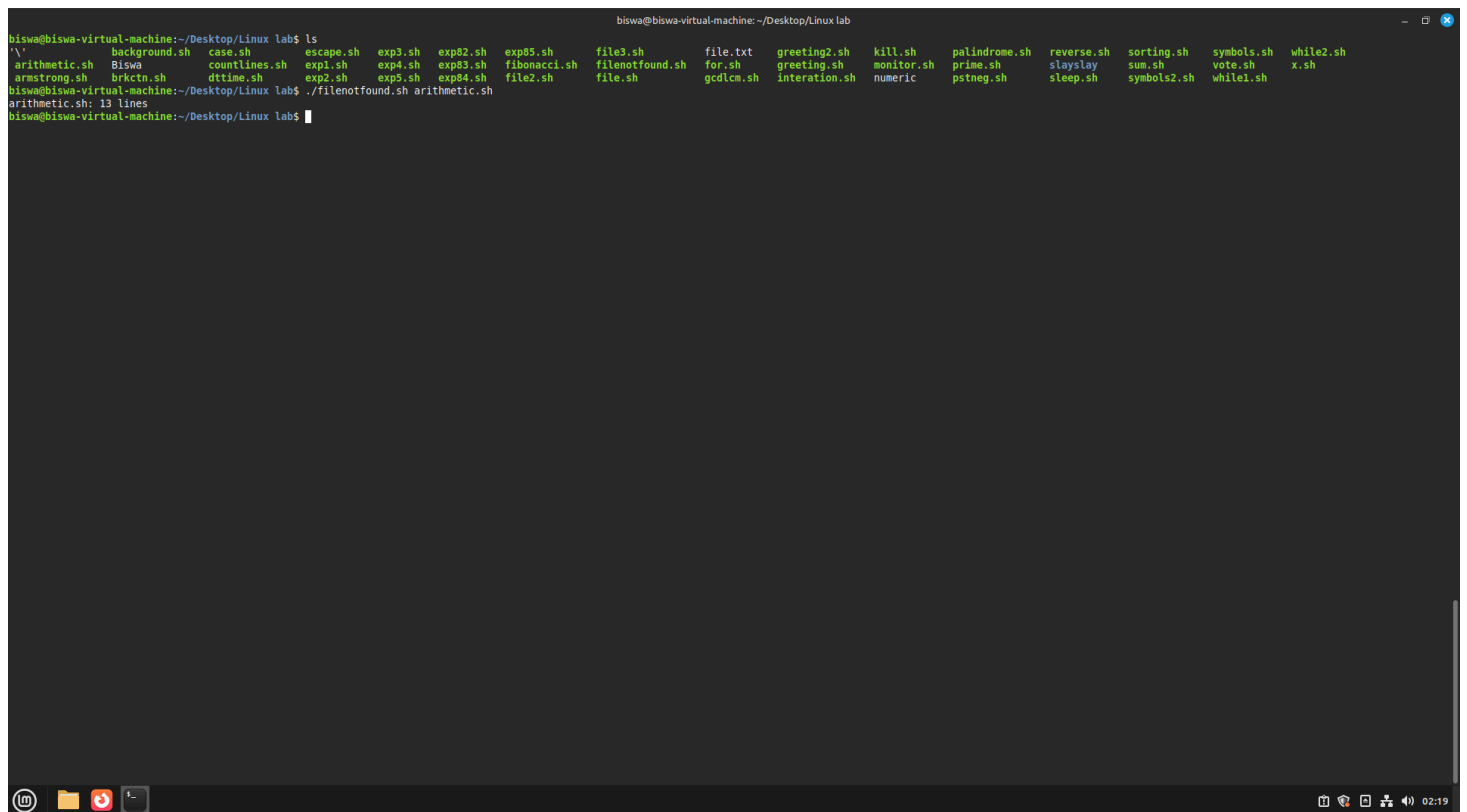
### Explanation:

Command-line arguments are accessed using `$@`. Looping through each argument allows file-wise operations.

## Command(s):

```
#!/bin/bash
for file in "$@"; do
    if [ -f "$file" ]; then
        echo "$file: $(wc -l < "$file") lines"
    else
        echo "$file not found"
    fi
done
```

## Output:

A terminal window titled 'biswa@biswa-virtual-machine: ~/Desktop/Linux lab' showing the execution of a script. The user runs 'ls' and lists various shell scripts. Then, they run './filenotfound.sh arithmetic.sh', which outputs 'arithmetic.sh: 13 lines'. The terminal window has a dark background with light-colored text. The bottom of the window shows a taskbar with icons for a file manager, a terminal, and other applications, along with system status icons on the right.

```
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ls
'\''
arithmetic.sh  background.sh  case.sh          escape.sh      exp3.sh      exp82.sh      exp85.sh      file3.sh      file.txt      greeting2.sh  kill.sh        palindrome.sh  reverse.sh    sorting.sh    symbols.sh    while2.sh
armstrong.sh   Biswa          countlines.sh   exp1.sh       exp4.sh      exp83.sh      fibonacci1.sh file4.sh      for.sh        greeting3.sh  monitor.sh     prime.sh      slayslay      sum.sh        vote.sh       x.sh
brkctn.sh      dttime.sh     exp2.sh         exp5.sh       exp84.sh      file2.sh      file.sh        gcdlcm.sh    interaction.sh numeric        pstneg.sh     sleep.sh      symbols2.sh  while1.sh

biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./filenotfound.sh arithmetic.sh
arithmetic.sh: 13 lines
biswa@biswa-virtual-machine:~/Desktop/Linux lab$
```

## Exercise 4: Nested conditionals

### Task Statement:

Write a script to check if a year is a leap year.

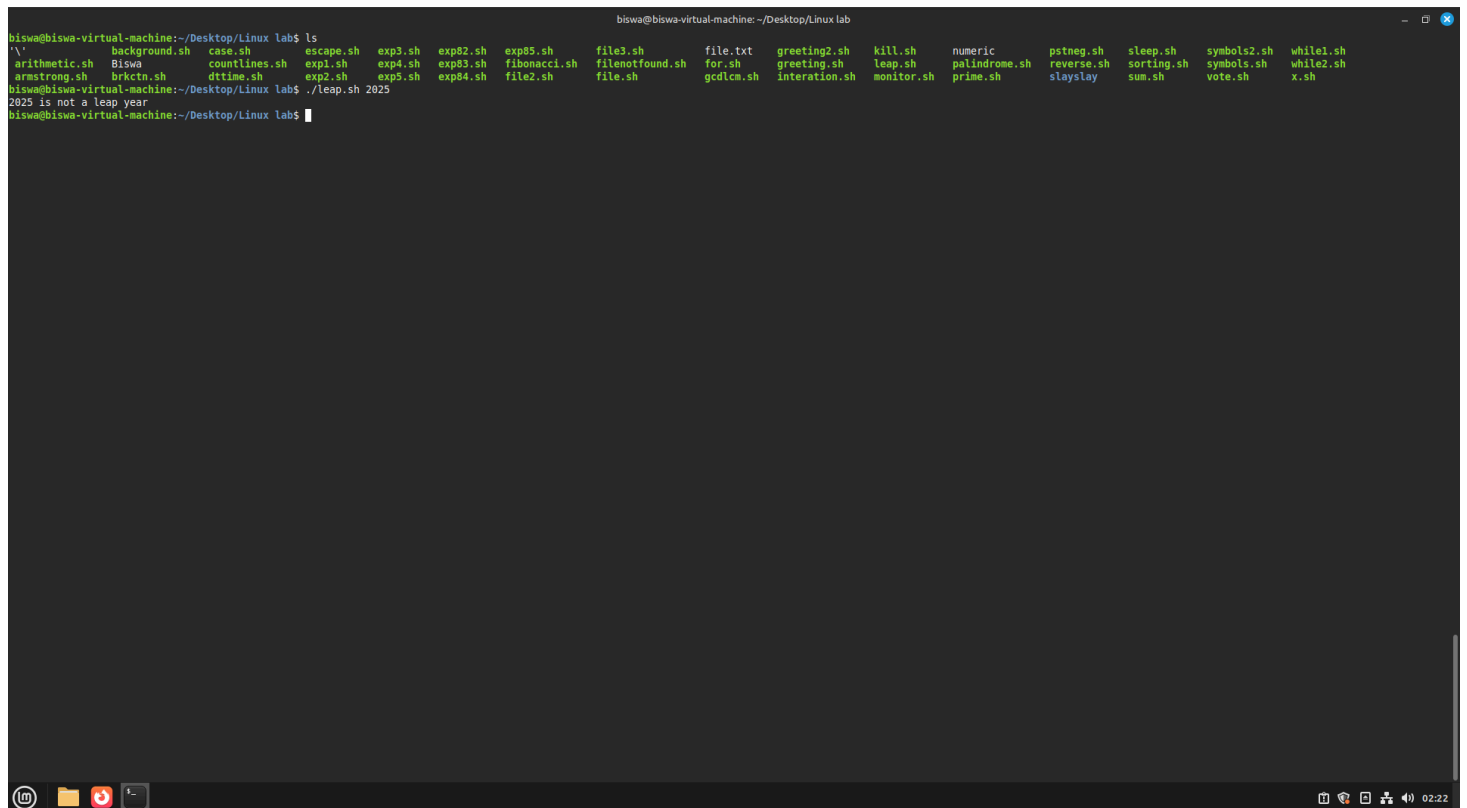
# Explanation:

A leap year is divisible by 4, but if divisible by 100 it must also be divisible by 400.

# Command(s):

```
#!/bin/bash
year=$1
if (( year % 400 == 0 )); then
    echo "$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
```

# Output:



The screenshot shows a terminal window titled "biswa@biswa-virtual-machine: ~/Desktop/Linux lab". The user has run the command `ls`, which lists various shell scripts in the directory, including `arithmetic.sh`, `background.sh`, `case.sh`, `escape.sh`, `exp3.sh`, `exp82.sh`, `exp85.sh`, `file3.sh`, `file.txt`, `greeting2.sh`, `kill.sh`, `numeric`, `pstneg.sh`, `sleep.sh`, `symbols2.sh`, and `while1.sh`. The user then runs the command `./leap.sh 2025`, and the output is `2025 is not a leap year`. The terminal window has a dark background and a light-colored text. The bottom of the window shows a taskbar with icons for a file manager, a terminal, and a clock showing 02:22.

# Task 1

## Task Statement:

Write a script that starts a background job (e.g., `sleep 60`), lists all jobs, brings the job to the foreground, and then terminates it.

## Command(s):

```
#!/bin/bash
```

```
sleep 60 &
```

```
jobs
```

```
fg %1
```

```
kill %1
```

## Output:

```
biswa@biswa-virtual-machine: ~/Desktop/Linux lab
ls
'\
arithmetic.sh  background2.sh  brkctn.sh  dttime.sh  exp2.sh  exp5.sh  exp84.sh  file2.sh  file.sh  gcdlcm.sh  interation.sh  monitor.sh  prime.sh  slayslay  sum.sh  vote.sh  x.sh
armstrong.sh  Biswa          countlines.sh  expl.sh  exp3.sh  exp82.sh  exp85.sh  file3.sh  file.txt  greeting2.sh  kill.sh  numeric  pstneg.sh  sleep.sh  symbols2.sh  while1.sh
armstrong.sh  Biswa          countlines.sh  expl.sh  exp4.sh  exp83.sh  fibonacci.sh  filenotfound.sh  for.sh  greeting.sh  leap.sh  palindrome.sh  reverse.sh  sorting.sh  symbols.sh  while2.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./background2.sh
[1]+  Running                  sleep 60 &
./background2.sh: line 5: fg: no job control
./background2.sh: line 6: kill: %1: no such job
biswa@biswa-virtual-machine:~/Desktop/Linux lab$
```

# Task 2

## Task Statement:

Create a script that compares two files and displays whether their contents are identical or different.

## Command(s):

```
#!/bin/bash

read -p "Enterfile 1: " file1
read -p "Enterfile 2: " file2

if cmp -s "$file1" "$file2"; then
    echo "Files are identical."
else
    echo "Files are different."
fi
```

## Output:

A terminal window titled 'biswa@biswa-virtual-machine: ~/Desktop/Linux lab' showing the execution of a script. The terminal output is as follows:

```
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ls
'\
arithmetic.sh  Biswa      countlines.sh  exp2.sh  exp82.sh  fibonacci.sh  file.sh  greeting2.sh  leap.sh  numeric  process.sh  sleep.sh  symbols.sh  x.sh
armstrong.sh  brkctn.sh  dttime.sh     exp3.sh  exp83.sh  file2.sh     file.txt  greeting.sh  memory.log.txt  palindrom.sh  pstneg.sh  sorting.sh  vote.sh
background2.sh case.sh    escape.sh     exp4.sh  exp84.sh  file3.sh     for.sh    interation.sh  monitor.sh  pattern.sh  reverse.sh  sum.sh     while1.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ vim filecheck.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ chmod +x filecheck.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./filecheck.sh
Enterfile 1: arithmetic.sh
Enterfile 2: case.sh
Files are different.
biswa@biswa-virtual-machine:~/Desktop/Linux lab$
```

# Task 3

## Task Statement:

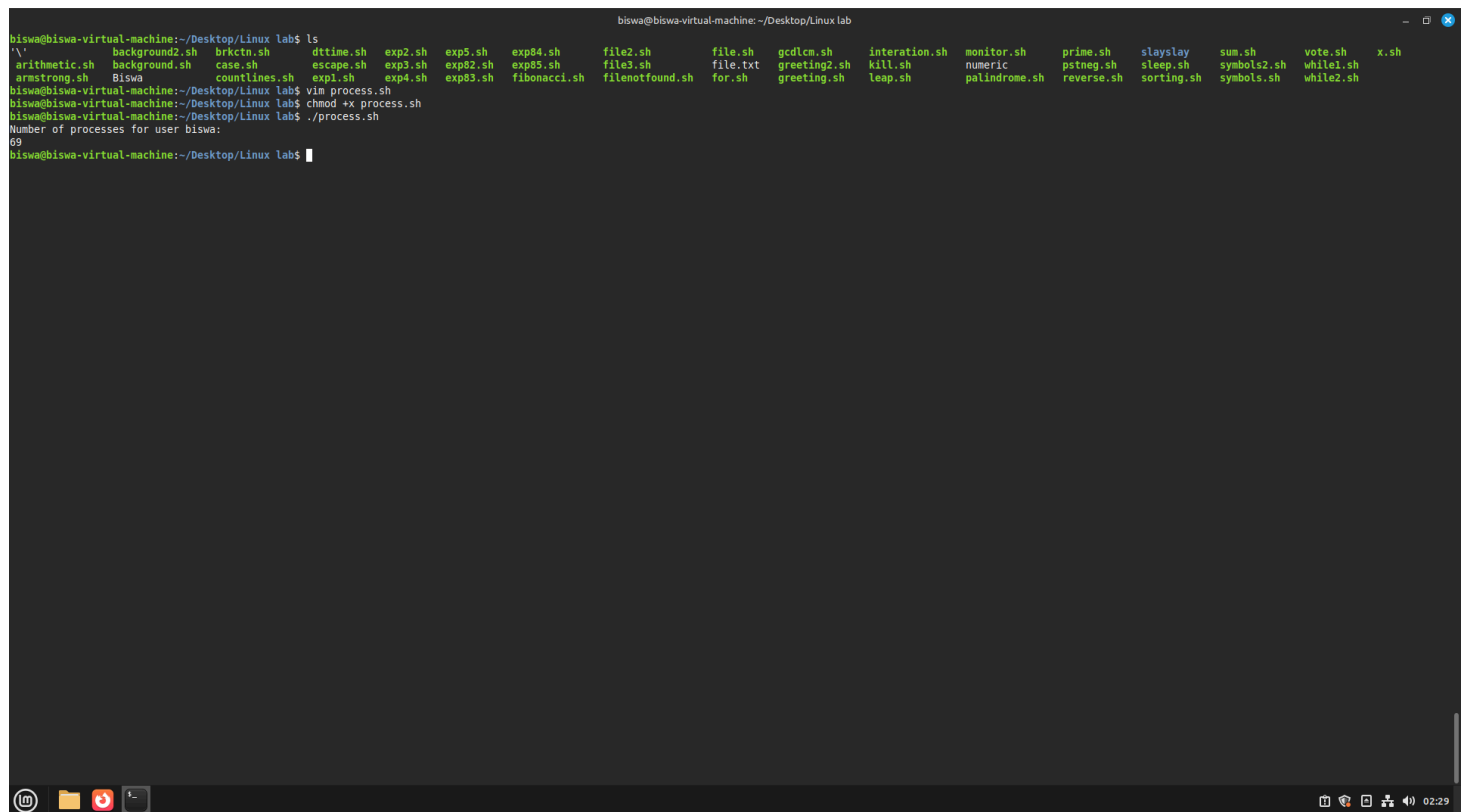
Write a script that counts the number of processes currently being run by your user.

## Command(s):

```
#!/bin/bash

echo "Number of processes for user $USER:"
ps -u $USER | wc -l
```

## Output:



```
biswa@biswa-virtual-machine: ~/Desktop/Linux lab
ls
^V
arithmetic.sh  background2.sh  bskctn.sh      dttime.sh      exp2.sh      exp5.sh      exp84.sh      file2.sh      file.sh      gcdlcm.sh      interation.sh  monitor.sh      prime.sh      slayslay      sum.sh      vote.sh      x.sh
armstrong.sh   Biswa           countlines.sh  escape.sh      exp3.sh      exp82.sh      exp85.sh      file3.sh      file.txt     greeting2.sh  kill.sh        numeric        pstneg.sh    sleep.sh     symbols2.sh  while1.sh
Biswa         countlines.sh  expl.sh        exp4.sh      exp83.sh      fibonacci.sh  filenotfound.sh  for.sh      greeting.sh  leap.sh        palindrome.sh  reverse.sh      sorting.sh    symbols.sh   while2.sh

biswa@biswa-virtual-machine:~/Desktop/Linux lab$ vim process.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ chmod +x process.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./process.sh
Number of processes for user biswa:
09
biswa@biswa-virtual-machine:~/Desktop/Linux lab$
```



# Task 4

## Task Statement:

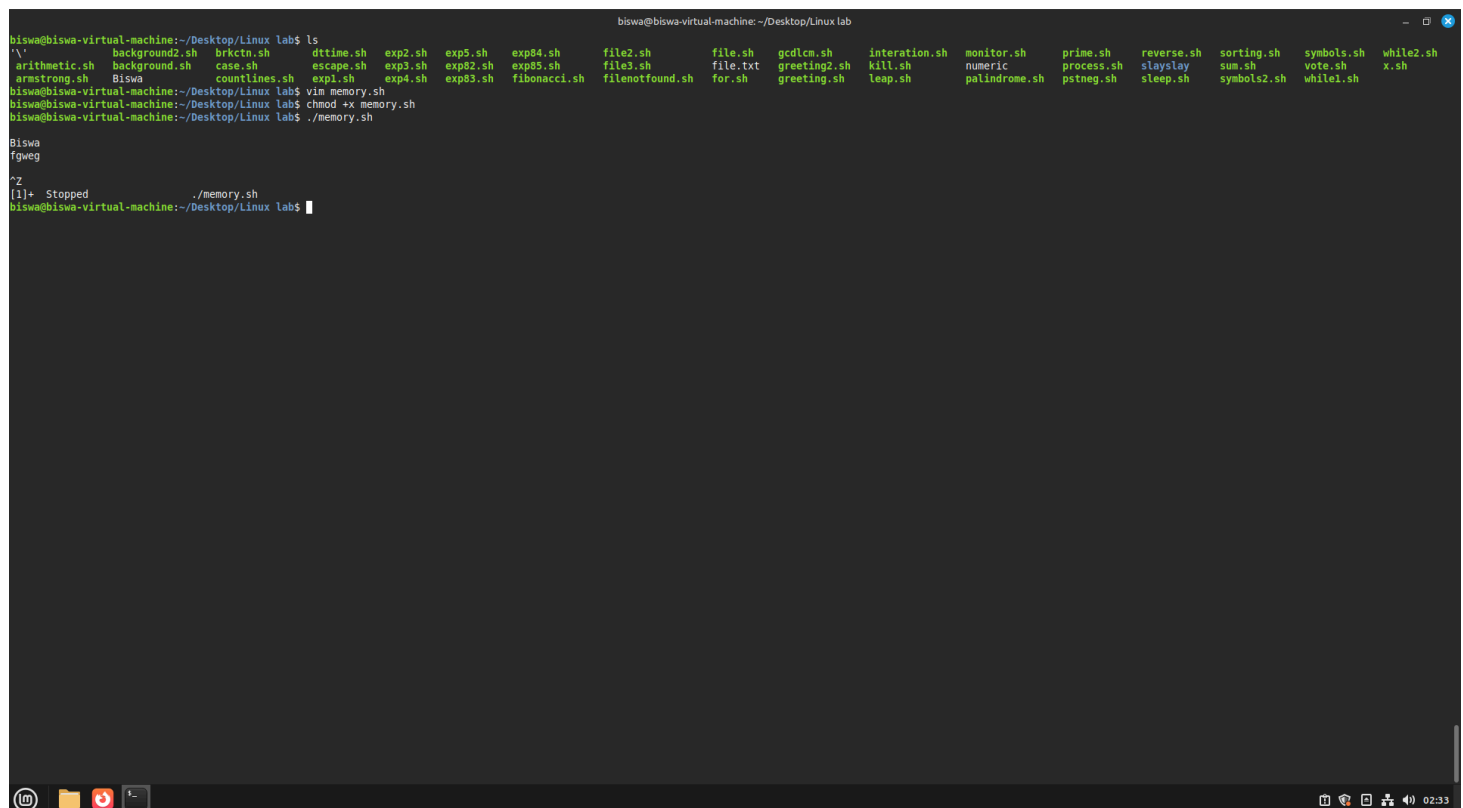
Develop a script that monitors memory usage every 5 seconds and logs it into a file.

## Command(s):

```
#!/bin/bash

while true; do
    echo "Mem use  $(date)" >> memory_log.txt
    free -m >> memory_log.txt
    echo "------" >> memory_log.txt
    sleep 5
done
```

## Output:



The screenshot shows a terminal window titled "biswa@biswa-virtual-machine: ~/Desktop/Linux lab". The user has created a file named "memory.sh" and made it executable with "chmod +x memory.sh". They then ran the script with "./memory.sh". The script is a bash loop that runs indefinitely, logging memory usage and the current date to "memory\_log.txt" every 5 seconds. The terminal output shows the first few iterations of the script running. At the bottom of the terminal, there is a status bar showing the time as 02:33.

```
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ls
'\'      background2.sh  brkctn.sh  dttime.sh  exp2.sh  exp5.sh  exp84.sh  file2.sh  file.sh  gcdlcm.sh  interation.sh  monitor.sh  prime.sh  reverse.sh  sorting.sh  symbols.sh  while2.sh
arithmetic.sh  background.sh  case.sh  escape.sh  exp3.sh  exp62.sh  exp85.sh  file3.sh  file.txt  greeting2.sh  kill.sh  numeric  process.sh  slayslay  sum.sh  vote.sh  x.sh
armstrong.sh  Biswa  countlines.sh  expl.sh  exp4.sh  exp83.sh  fibonacci.sh  filenotfound.sh  for.sh  greeting.sh  leap.sh  palindrome.sh  pstneg.sh  sleep.sh  symbols2.sh  while1.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ vim memory.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ chmod +x memory.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./memory.sh

Biswa
fgweg

^Z
[1]+  Stopped                  ./memory.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$
```

# Task 5

## Task Statement:

Write a script that prompts for a filename and a search pattern, then displays the count of matching lines.

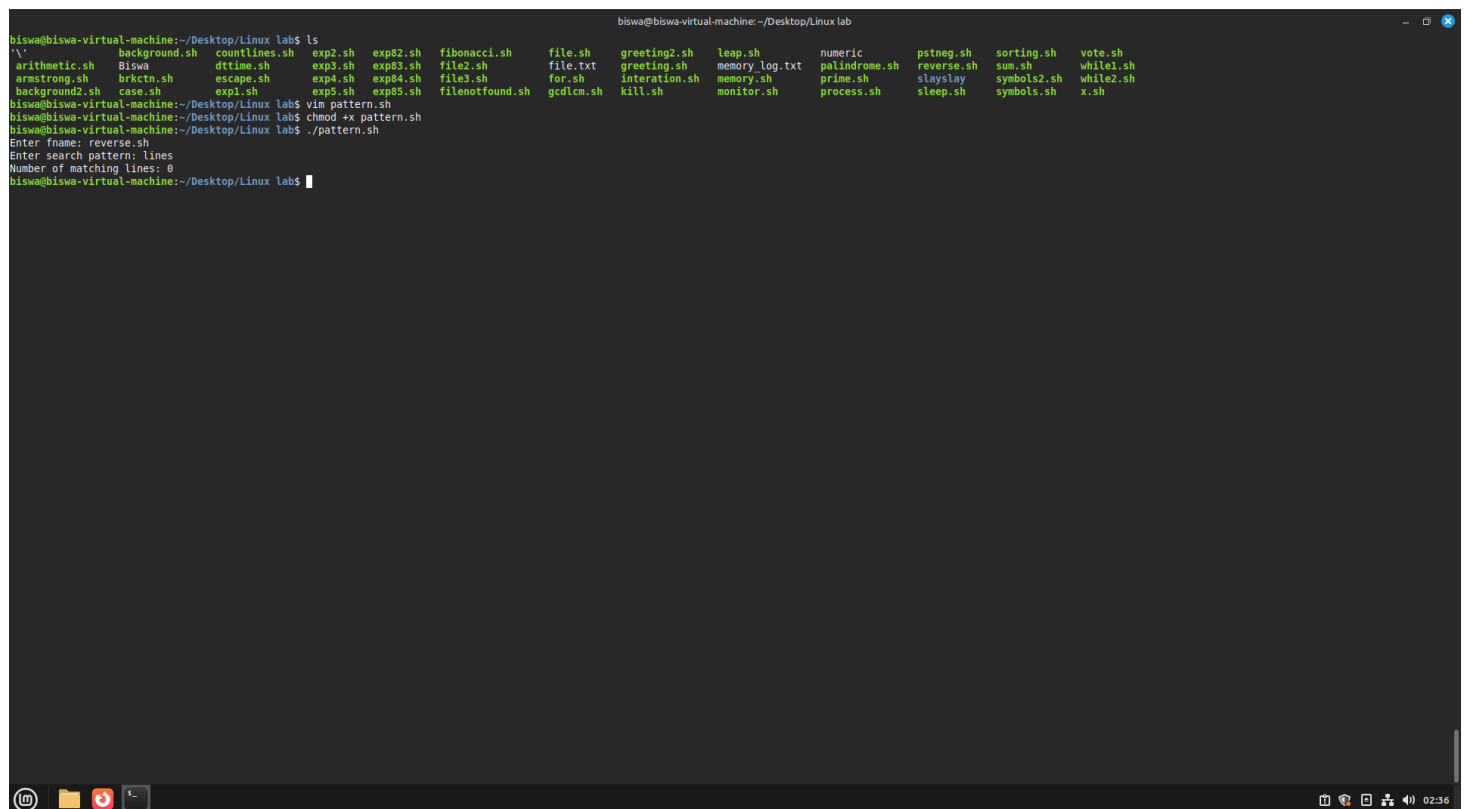
## Command(s):

```
#!/bin/bash

read -p "Enter fname: " file
read -p "Enter search pattern: " pattern

count=$(grep -c "$pattern" "$file")
echo "Number of matching lines: $count"
```

## Output:



The screenshot shows a terminal window titled "biswa@biswa-virtual-machine: ~/Desktop/Linux lab". The user has listed files in the directory, including various shell scripts like arithmetic.sh, background.sh, and many others. They then run the script ./pattern.sh. The script prompts for a filename, and the user enters "reverse.sh". It then prompts for a search pattern, and the user enters "Lines". Finally, it displays the output: "Number of matching lines: 0".

```
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ls
'\
arithmic.sh  Biswa      countlines.sh  exp2.sh  exp82.sh  fibonacci.sh  file.sh  greeting2.sh  leap.sh  numeric  pstneg.sh  sorting.sh  vote.sh
armstrong.sh brkctn.sh  dttime.sh     exp3.sh  exp83.sh  file2.sh     file.txt  greeting.sh  memory_log.txt  palindrome.sh  reverse.sh  sum.sh  while1.sh
background2.sh case.sh    escape.sh     exp4.sh  exp84.sh  file3.sh     for.sh    interation.sh  memory.sh  prime.sh  slayslay  symbols2.sh  while2.sh
background2.sh case.sh    exp1.sh      exp5.sh  exp85.sh  filenotfound.sh  gcdlcm.sh  kill.sh    monitor.sh  process.sh  sleep.sh  symbols.sh  x.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ vim pattern.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ chmod +x pattern.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./pattern.sh
Enter fname: reverse.sh
Enter search pattern: Lines
Number of matching lines: 0
biswa@biswa-virtual-machine:~/Desktop/Linux lab$
```

# Result

- Implemented conditional statements ( `if-else` , `case` ) in shell scripts.
- Practiced handling command-line arguments and nested conditions.
- Wrote reusable and flexible shell scripts.

## Challenges Faced & Learning Outcomes

- Challenge 1: Forgetting to quote variables in conditions — resolved by using `"$var"` to avoid word splitting.
- Challenge 2: Pattern matching in `case` — practiced with multiple examples.
- challenge 3: Could not run experiment 2 perfectly.

## Learning:

- Learned practical use of branching and decision-making in shell scripting.
- Understood command-line argument handling for automation.

## Conclusion

This experiment extended shell programming by introducing decision-making and parameter handling. The scripts demonstrate the flexibility of shell programming for different use cases.