

# **Experiment 8: Shell Programming (Continued)**

**Name: Aarush Prasad Roll No.: 590027630 Date: 2025-10-30**

## **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:

```
Virtual Box [Running] - Oracle VirtualBox
aarush07@aarush07-VirtualBox:~/Linux lab$ vim ifelse.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat ifelse.sh
#!/bin/bash
num=$1
if [ $num -gt 0 ]; then
    echo "$num is positive"
elif [ $num -lt 0 ]; then
    echo "$num is negative"
else
    echo "$num is zero"
fi
aarush07@aarush07-VirtualBox:~/Linux lab$ ./ifelse.sh 7
7 is positive
aarush07@aarush07-VirtualBox:~/Linux lab$ ./ifelse.sh -98
-98 is negative
aarush07@aarush07-VirtualBox:~/Linux lab$ ./ifelse.sh 0
0 is zero
aarush07@aarush07-VirtualBox:~/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.

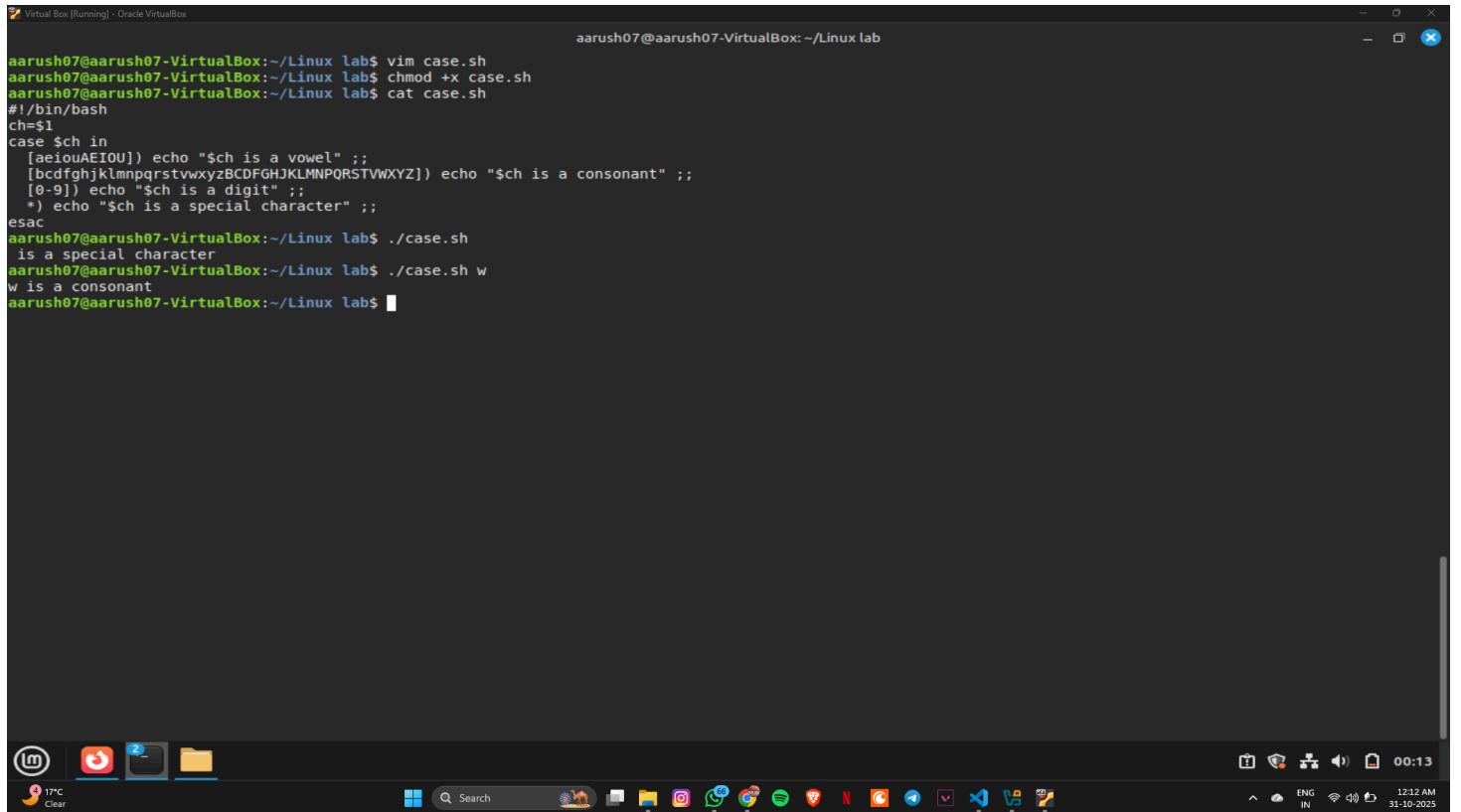
# Explanation:

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

## Command(s):

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

## Output:



```
aarush07@aarush07-VirtualBox:~/Linux lab$ vim case.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ chmod +x case.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat case.sh
#!/bin/bash
ch=$1
case $ch in
[aeiouAEIOU]) echo "$ch is a vowel" ;;
[bcdfghjklmnpqrstuvwxyzBCDFGHJKLMNPQRSTUVWXYZ]) echo "$ch is a consonant" ;;
[@-9]) echo "$ch is a digit" ;;
*) echo "$ch is a special character" ;;
esac
aarush07@aarush07-VirtualBox:~/Linux lab$ ./case.sh
is a special character
aarush07@aarush07-VirtualBox:~/Linux lab$ ./case.sh w
w is a consonant
aarush07@aarush07-VirtualBox:~/Linux lab$
```

# 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.

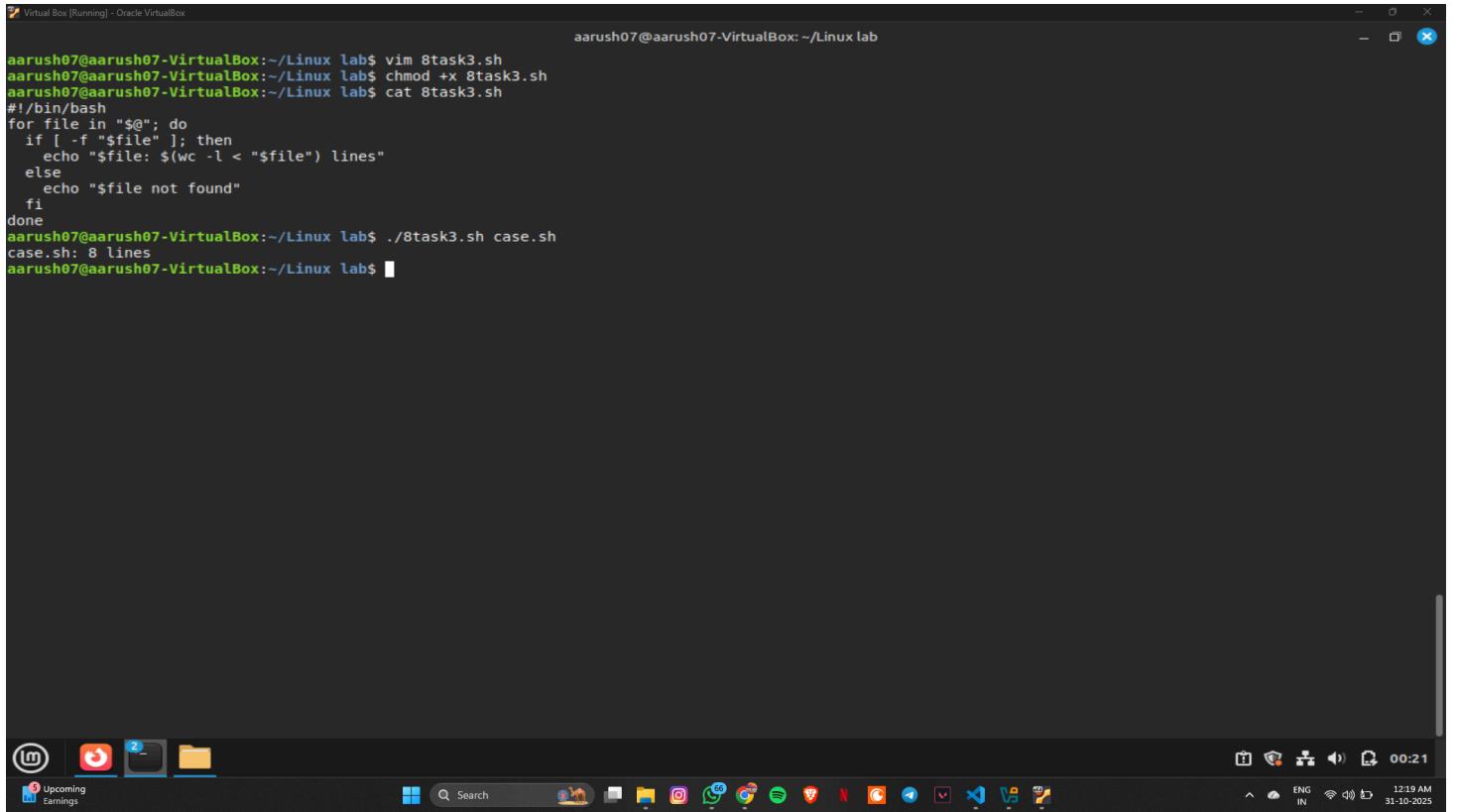
## Explanation:

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:



```
aarush07@aarush07-VirtualBox:~/Linux lab$ vim 8task3.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ chmod +x 8task3.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat 8task3.sh
#!/bin/bash
for file in "$@"; do
    if [ -f "$file" ]; then
        echo "$file: $(wc -l < "$file") lines"
    else
        echo "$file not found"
    fi
done
aarush07@aarush07-VirtualBox:~/Linux lab$ ./8task3.sh case.sh
case.sh: 8 lines
aarush07@aarush07-VirtualBox:~/Linux lab$
```

The screenshot shows a Linux desktop environment with a terminal window open in Oracle VirtualBox. The terminal window title is "Virtual Box [Running] - Oracle VirtualBox". The terminal content shows the execution of a shell script named 8task3.sh, which reads a file named case.sh and prints its line count. The desktop taskbar at the bottom shows various application icons and the system tray.

## Exercise 4: Nested conditionals

### Task Statement:

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

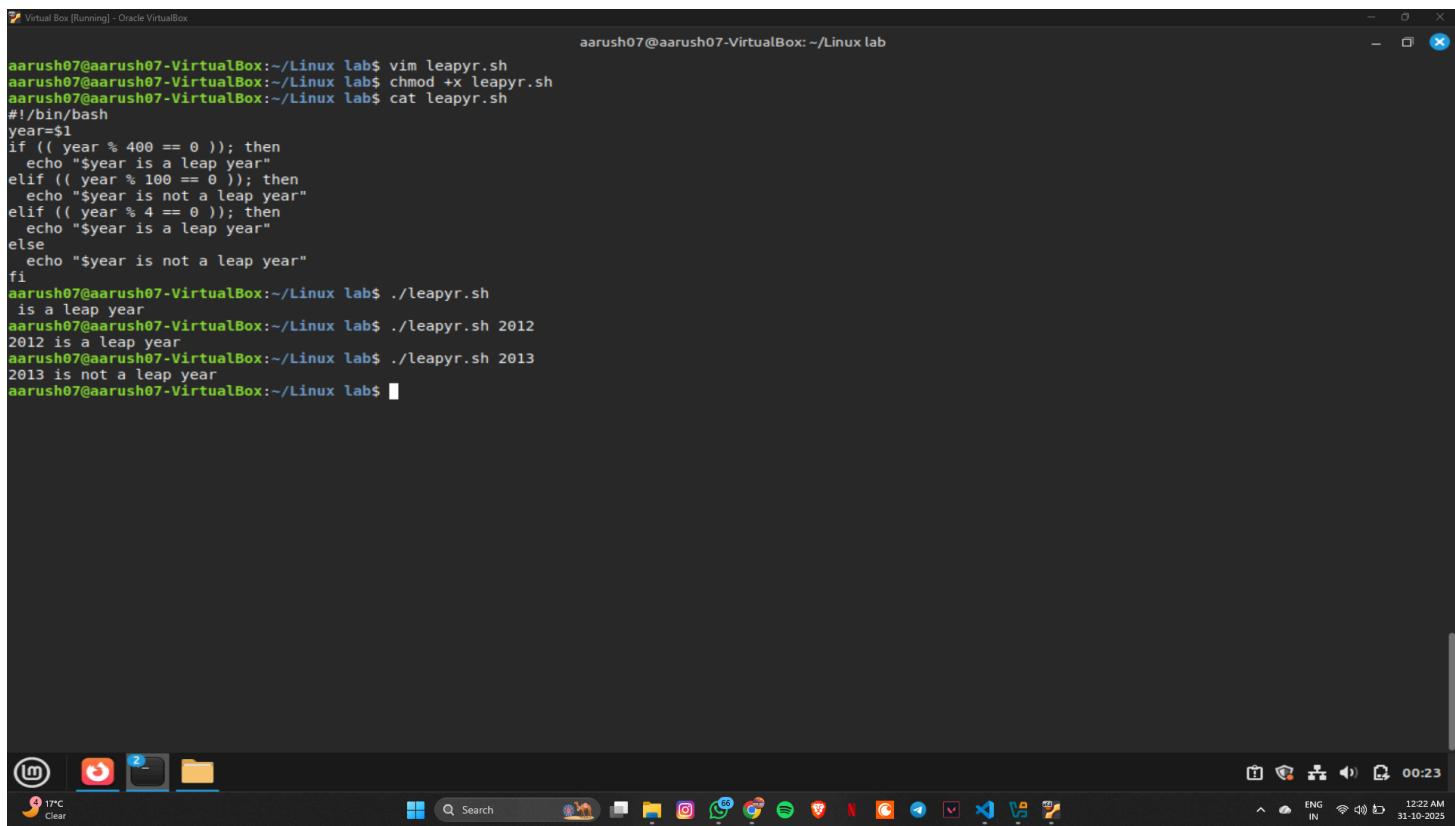
### Explanation:

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 "Virtual Box [Running] - Oracle VirtualBox". The terminal output is as follows:

```
aarush07@aarush07-VirtualBox:~/Linux lab$ vim leapyr.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ chmod +x leapyr.sh
aarush07@aarush07-VirtualBox:~/Linux lab$ cat leapyr.sh
#!/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 "$year is not a leap year"
fi
aarush07@aarush07-VirtualBox:~/Linux lab$ ./leapyr.sh
is a leap year
aarush07@aarush07-VirtualBox:~/Linux lab$ ./leapyr.sh 2012
2012 is a leap year
aarush07@aarush07-VirtualBox:~/Linux lab$ ./leapyr.sh 2013
2013 is not a leap year
aarush07@aarush07-VirtualBox:~/Linux lab$
```

The terminal window is part of a desktop environment, with a taskbar at the bottom showing various application icons and system status indicators.

## 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.

## 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.