**OPERATING SYSTEM ASSIGNMENT**

Name: Chinmyee Prabhu

Class: CS-AIML

PRN: 12211768

Roll No. : 13

**Aim:**

Write a shell script to cover following concepts  
1. Basic Arithmetic  
2. Command Line arguments  
3. String functions  
4. Conditional statements and loops  
5. Executing Linux commands in a shell  
6. Functions  
7. Arrays

**Solution:**

1. Basic Arithmetic

Code:

#!/bin/bash

# Function to perform addition

addition() {

    echo "$1 + $2" | bc

}

# Function to perform subtraction

subtraction() {

    echo "$1 - $2" | bc

}

# Function to perform multiplication

multiplication() {

    echo "$1 \* $2" | bc

}

# Function to perform division

division() {

    echo "scale=2; $1 / $2" | bc

}

# Main function

main() {

    echo "Welcome to Simple Calculator"

    echo "1. Addition"

    echo "2. Subtraction"

    echo "3. Multiplication"

    echo "4. Division"

    echo "Enter your choice (1/2/3/4): "

    read choice

    echo "Enter first number: "

    read num1

    echo "Enter second number: "

    read num2

    case $choice in

        1) result=$(addition $num1 $num2);;

        2) result=$(subtraction $num1 $num2);;

        3) result=$(multiplication $num1 $num2);;

        4) result=$(division $num1 $num2);;

        \*) echo "Invalid choice"; exit;;

    esac

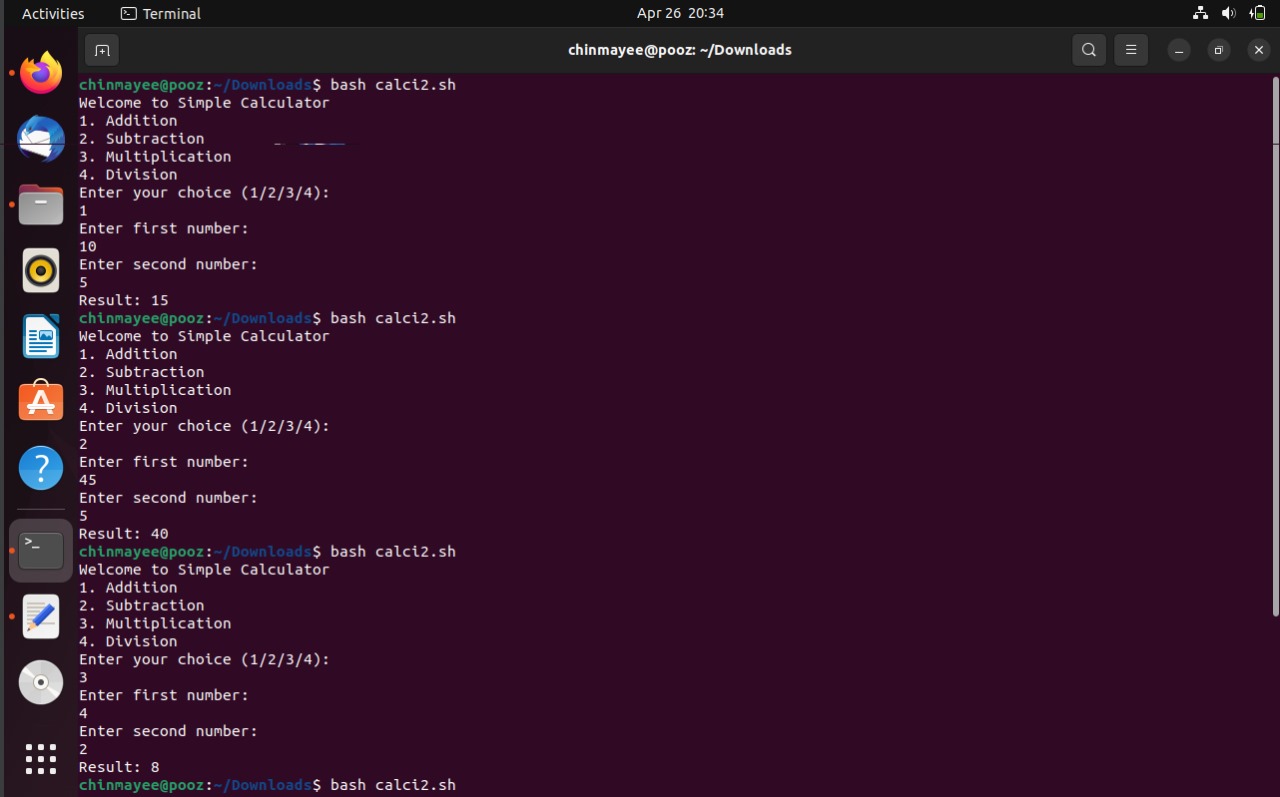
    echo "Result: $result"

}

# Call main function

main

Output:



2. Command Line arguments

Code:

#!/bin/bash

# Check if command-line arguments are provided

if [ $# -eq 0 ]; then

    echo "Usage: $0 <name>"

    exit 1

fi

# Retrieve command-line argument

name=$1

# Function to greet the user

greet() {

    echo "Hello, $1!"

}

# Main function

main() {

    echo "Welcome to Greeting Script"

    echo "Please enter your age:"

    read -p "Age: " age

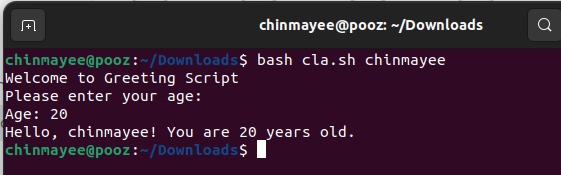
    echo "Hello, $name! You are $age years old."

}

# Call main function

main

Output:



3. String functions Code:

#!/bin/bash

# Function to check if a string is empty

is\_empty() {

    if [ -z "$1" ]; then

        echo "String is empty"

    else

        echo "String is not empty"

    fi

}

# Function to get the length of a string

get\_length() {

    echo "Length of the string is ${#1}"

}

# Function to convert string to lowercase

to\_lowercase() {

    echo "${1,,}"

}

# Function to convert string to uppercase

to\_uppercase() {

    echo "${1^^}"

}

# Function to concatenate two strings

concatenate() {

    echo "$1$2"

}

# Main function

main() {

    echo "Enter a string:"

    read input\_string

    is\_empty "$input\_string"

    get\_length "$input\_string"

    echo "String in lowercase: $(to\_lowercase "$input\_string")"

    echo "String in uppercase: $(to\_uppercase "$input\_string")"

    echo "Enter another string to concatenate with the first one:"

    read second\_string

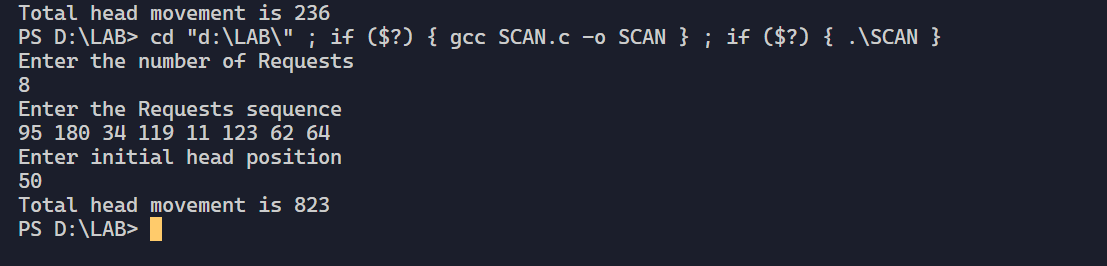
    echo "Concatenated string: $(concatenate "$input\_string" "$second\_string")"

}

# Call main function

main

Output:



4. . Conditional statements and loops

Code:

#!/bin/bash

# For loop

echo "Enter the number of times to repeat using for loop:"

read for\_count

echo "Output using for loop:"

for (( i=1; i<=$for\_count; i++ )); do

    echo "Iteration: $i"

done

# While loop

echo "Enter the number of times to repeat using while loop:"

read while\_count

echo "Output using while loop:"

while [ $while\_count -gt 0 ]; do

    echo "Iteration: $while\_count"

    ((while\_count--))

done

# Until loop

echo "Enter the number of times to repeat using until loop:"

read until\_count

echo "Output using until loop:"

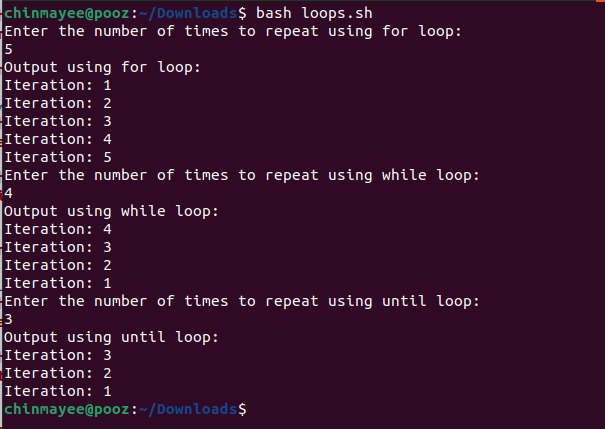
until [ $until\_count -eq 0 ]; do

    echo "Iteration: $until\_count"

    ((until\_count--))

done

Output:



5. Executing Linux commands in a shell

#!/bin/bash

# Prompt the user to enter a command

echo "Enter a Linux command:"

read command

# Execute the command entered by the user

echo "Executing command: $command"

$command

Output:



6. Functions

#!/bin/bash

# Define a function to greet the user

greet\_user() {

    echo "Hello, $1!"

}

# Prompt the user to enter their name

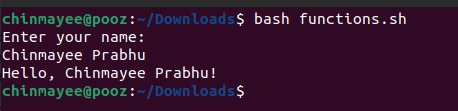
echo "Enter your name:"

read name

# Call the function with the user's name as an argument

greet\_user "$name"

Output:



7. Arrays

Code:

#!/bin/bash

# Prompt the user to enter the size of the array

echo "Enter the size of the array:"

read size

# Initialize an empty array

declare -a my\_array

# Populate the array with user input

echo "Enter $size elements, separated by spaces:"

read -a my\_array

# Display the elements of the array

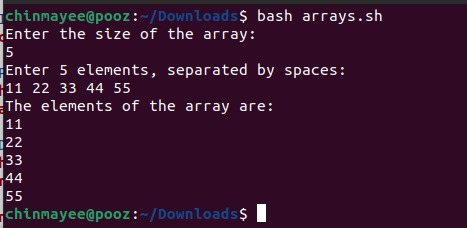
echo "The elements of the array are:"

for element in "${my\_array[@]}"; do

    echo "$element"

done

Output:



Conclusion:

Basic Arithmetic: The script demonstrates basic arithmetic operations such as addition, subtraction, multiplication, and division, showcasing fundamental mathematical computations in bash.

Command Line Arguments: It illustrates how to handle command line arguments passed to a bash script, enabling interaction with the script by providing input parameters from the command line.

String Functions: The script includes string manipulation operations such as concatenation, substring extraction, and string length determination, showcasing the versatility of bash for handling textual data.

Conditional Statements and Loops: It showcases the use of conditional statements (if-else) and loops (for, while) for implementing decision-making and repetitive tasks, facilitating flow control within the script.

Executing Linux Commands in a Shell: The script demonstrates the execution of Linux commands within a bash script, enabling automation of system tasks and interaction with the underlying operating system environment.

Functions: It showcases the definition and usage of functions in bash, allowing code modularity, reusability, and abstraction for organizing and structuring complex scripts.

Arrays: The script utilizes arrays to store and manipulate collections of data, showcasing how bash handles indexed and associative arrays for efficient data storage and retrieval.