Lab Experiment- Shell Scripting & MakeFile

# Objective:

This lab manual is designed to provide hands-on experience with Linux Shell scripting and Makefiles.

# Prerequisites:

* Basic understanding of Linux commands
* Access to a Linux environment (can be a virtual machine or WSL on Windows)
* Text editor (e.g., nano, vim, or gedit)

# Lab 1: Basic Shell Scripting

## Exercise 1.1: Hello World

* Create a new file named hello.sh
* Add the shebang line: #!/bin/bash
* Write a command to print "Hello, World!"
* Make the script executable using chmod
* Run the script

## Exercise 1.2: Variables and User Input

* Create a script that asks for the user's name
* Store the input in a variable
* Print a greeting using the stored name
* Run the script with different inputs

## Exercise 1.3: Command-line Arguments

* Create a script that accepts two numbers as command-line arguments
* Calculate and print the sum of these numbers
* Run the script with different number pairs

# Lab 2: Control Structures

## Exercise 2.1: If-Else Statement

1. Write a script that checks if a number (provided as an argument) is even or odd

2. Use an if-else statement to print the result

3. Test with various numbers

## Exercise 2.2: For Loop

1. Create a script that prints the first 10 multiples of a number (provided as an argument)

2. Use a for loop to calculate and print the multiples

3. Test with different numbers

## Exercise 2.3: While Loop

1. Write a script that implements a simple guessing game

2. Generate a random number between 1 and 10

3. Use a while loop to allow the user to guess until correct

4. Provide "higher" or "lower" hints

# Lab 3: Functions and Arrays

## Exercise 3.1: Functions

1. Create a function that calculates the factorial of a number

2. Call this function with different numbers in your script

3. Print the results

## Exercise 3.2: Arrays

1. Create an array of fruits

2. Write a function that prints each fruit in the array

3. Add a new fruit to the array and call the function again

## Exercise 3.3: Associative Arrays

1. Create an associative array of country-capital pairs

2. Write a function that asks the user for a country and returns its capital

3. Implement error handling for countries not in the array

# Lab 4: File Operations and Text Processing

## Exercise 4.1: File Reading

1. Create a text file with several lines of content

2. Write a script that reads this file line by line

3. Print each line prefixed with its line number

## Exercise 4.2: Text Processing

1. Create a log file with entries of the format: "YYYY-MM-DD username action"

2. Write a script that:

a. Counts the total number of entries

b. Lists unique usernames

c. Counts actions per user

## Exercise 4.3: File Backup

1. Write a script that creates a backup of a specified directory

2. The backup should be a compressed tar file with the current date in its name

3. Implement error handling for cases where the directory doesn't exist

# Lab 5: Introduction to Makefiles

## Exercise 5.1: Basic Makefile

1. Create a simple C program with a main.c and functions.c

2. Write a Makefile to compile these into an executable

3. Include targets for 'all', 'clean', and individual object files

## Exercise 5.2: Advanced Makefile

1. Extend the previous Makefile to handle multiple source files automatically

2. Add a 'debug' target that compiles with debugging symbols

3. Implement dependency tracking for header files

## Exercise 5.3: Makefile for a Shell Script Project

1. Create a project with multiple shell scripts

2. Write a Makefile that:

a. Checks scripts for syntax errors

b. Runs unit tests (if available)

c. Installs scripts to a specified directory

## Helping Material

* [Shell Documentation](https://www.gnu.org/software/bash/manual/bash.html)
* [Test operators in Bash](https://linuxhandbook.com/bash-test-operators/)
* [Make Documentation](https://www.gnu.org/software/make/manual/make.html)