National University of Computer and Emerging Sciences

Operating Systems Lab (CL2006)

Date: March 12th 2025

Course Instructor(s) Miss Syeda Tehreem, Miss Fatima Gado

Sessional-I Exam (B)

Total Time: 1.5 Hours

Total Marks: 50

Total Questions: 03

Semester: Fall-2025

Campus: Karachi

Dept: Computer Science

Student Name	Roll No	Section	Student Signature

CLO # 1: Understand and Analyze Command Line tools for Linux OS and Shell scripts for system level programming to automate tasks such as file management, system backups and software installations. (Lab # 1 and Lab # 3)

Q1. [= 10wtg 20 marks]

Part (A): (Manually write the commands)

1. Which command displays the system architecture?
2. How can you check CPU information in Linux?
3. Write the command to check the Linux kernel version.
4. How can you display the current system date and time?
5. Write the command to list all files and directories in long format.
6. Which command is used to search for numbers in a file using "grep"?
7. How can you create three directories named "dir1", "dir2", and "dir3"?
8. Write the command to remove a file named "file1".
9. How can you remove an empty directory named "dir1"?
10. Write the command to change the timestamp of "file_or_dir" to a specific date and time

sol :

```
1. uname -m
2. cat /proc/cpuinfo
3.uname -r
4. date
5. ls -l
6. grep -E '[0-9]+' filename.txt
7. mkdir dir1 dir2 dir3
8. rm file1
9. rmdir dir1
10. touch -t 202503121200 file_or_dir
```

Part (B)

A company needs a shell script to help employees manage their contacts. The script should display a menu with the following options:

1. Add a new contact (user inputs name, phone number, and email).
2. View all contacts.
3. Search for a contact by name and display their details.
4. Delete a contact (confirm before deletion).

5. Exit the program.

The script should validate inputs (e.g., non-empty name), handle errors (e.g., contact not found), and store contacts in a text file for persistence. The menu should run in a loop until the user chooses to exit.

Sol:

```
#!/bin/bash

CONTACT_FILE="contacts.txt"

add_contact() {
  echo "Enter name:"
  read name
  if [[ -z "$name" ]]; then
    echo "Name cannot be empty."
    return
  fi
  echo "Enter phone number:"
  read phone
  echo "Enter email:"
  read email
  echo "$name:$phone:$email" >> "$CONTACT_FILE"
  echo "Contact added!"
}

view_contacts() {
  echo "Contacts:"
  cat "$CONTACT_FILE"
}

search_contact() {
  echo "Enter name to search:"
  read name
  grep "$name" "$CONTACT_FILE" || echo "Contact not found."
}

delete_contact() {
  echo "Enter name to delete:"
  read name
  grep -v "$name" "$CONTACT_FILE" > temp.txt && mv temp.txt "$CONTACT_FILE"
  echo "Contact deleted!"
}

while true; do
  echo "1. Add Contact"
  echo "2. View Contacts"
  echo "3. Search Contact"
  echo "4. Delete Contact"
```

```
    echo "5. Exit"
    read choice
    case $choice in
      1) add_contact ;;
      2) view_contacts ;;
      3) search_contact ;;
      4) delete_contact ;;
      5) exit ;;
      *) echo "Invalid choice" ;;
    esac
done
```

CLO # 2: Gain hands on experience in writing code that interacts with operating system services related process and files system, multi-thread programing and different synchronization primitives. (Lab # 4 and Lab # 6)

Q2. [=10wtg 20 marks]

Part (A)

Create a C program named numbercopy.c that transfers numerical data between processes using a pipe. The program takes two arguments: an input file containing a list of numbers and an output file where the numbers will be copied. The parent process reads the numbers from the input file and writes them to the pipe, while the child process retrieves the numbers from the pipe and writes them to the output file. For example, executing

- ./numbercopy data.txt copied_data.txt
- should result in copied_data.txt containing the same numbers as data.txt.
- Ensure proper error handling.

Sol:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char *argv[]) {
   if (argc != 3) {
      fprintf(stderr, "Usage: %s input_file output_file\n", argv[0]);
      exit(1);
   }

   int pipefd[2];
   pipe(pipefd);

   pid_t pid = fork();
   if (pid == 0) { // Child process
      close(pipefd[1]);
      int dest = open(argv[2], O_WRONLY | O_CREAT, 0644);
      char buffer[1024];
      int bytesRead;
      while ((bytesRead = read(pipefd[0], buffer, sizeof(buffer))) > 0) {
         write(dest, buffer, bytesRead);
```

```
            }
            close(dest);
            close(pipefd[0]);
        } else { // Parent process
            close(pipefd[0]);
            int src = open(argv[1], O_RDONLY);
            char buffer[1024];
            int bytesRead;
            while ((bytesRead = read(src, buffer, sizeof(buffer))) > 0) {
                write(pipefd[1], buffer, bytesRead);
            }
            close(src);
            close(pipefd[1]);
            wait(NULL);
        }
        return 0;
    }
```

Part(B): (Manually write the complete C code)

The Harmonic series is defined as:

$$H(N) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N}$$

Write a C program that performs the following operations:

- The parent process reads an integer N from the command line and passes it to the child process.
- The child process computes and prints the Harmonic Sum up to N terms.
- The parent process waits for the child process to complete before terminating.
- Implement error handling to manage invalid inputs, such as non-integer or negative values.

Sol:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {  // Ensure exactly one argument is passed
        printf("Usage: %s <positive_integer>\n", argv[0]);
        return 1;
    }

    int N = atoi(argv[1]);  // Convert string argument to integer

    if (N <= 0) {  // Validate input
        printf("Error: Please enter a positive integer.\n");
        return 1;
    }
    pid_t pid = fork();  // Create child process

    if (pid < 0) {
        printf("Fork failed!\n");
        return 1;
    } else if (pid == 0) {  // Child process
        double sum = 0.0;
        for (int i = 1; i <= N; i++) {
            sum += 1.0 / i;  // Harmonic sum calculation
        }
        printf("Harmonic Sum H(%d) = %.6f\n", N, sum);
        exit(0);
    } else {  // Parent process
        wait(NULL);  // Wait for child process to finish
    }

    return 0;
}
```

CLO # 3: Understand how to configure and customize Linux Kernel for installations, applying patches and performance optimizations. (Lab # 2)

Q3. [= 5 wtg 10 marks]

You are developing an electricity bill calculator that determines the total cost based on energy consumption (kWh) and unit rates. The system consists of a header file (electricity.h) that declares the function for bill calculation, a C file (calculate_bill.c) that implements the logic, and another C file (main.c) that takes user input, calls the function, and displays the total bill amount. Write a Makefile that compiles calculate_bill.c and main.c separately into object files, links them to produce an executable named bill_calculator, and ensures efficient compilation. Additionally, include a clean target to remove compiled files.

Sol:

```
bill_calculator: calculate_bill.o main.o
	gcc -o bill_calculator calculate_bill.o main.o

calculate_bill.o: calculate_bill.c electricity.h
	gcc -c calculate_bill.c

main.o: main.c electricity.h
	gcc -c main.c

clean:
	rm -f bill_calculator calculate_bill.o main.o
```