NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

(KARACHI CAMPUS)

FAST School of Computing Fall 2024

PROJECT REPORT

MEMBERS:

- Mishaal Bheraiya (22K-4254)
- Ayesha Ansari (22K-4453)

COURSE INSTRUCTOR:

- Miss Rabia Ansari
- Sir Monis

Project Topic:

Sleeping Barber Problem

1. Introduction

The sleeping barber problem is a classic synchronization problem involving a barber who sleeps when there are no customers and is woken up by a customer if he is asleep when the customer arrives. This problem is used to demonstrate process synchronization using semaphores.

This project implements a solution to the sleeping barber problem in the form of a Linux model. The implementation uses semaphores for managing access to the barber's resources, like the waiting room and the barber chair.

2. Tools and Environment Setup

Step 1: Installing Required Tools

Ensure you have the necessary tools installed on your Ubuntu system, primarily a C compiler like GCC and the pthread library. Run these commands on your terminal:

```
sudo apt-get update
sudo apt-get install build-essential
```

Step 2: Writing the Code

Create a new file named OSproject.c using a text editor like nano:

```
nano OSproject.c
```

Step 3: Compile and Run the Code

Compile:

```
gcc OSproject.c -o project
```

Run:

```
./project
```

3. Code Overview

- **Semaphores:** Used to control access and synchronize the barber and customers.
- **Customer and Barber Threads:** Functions that simulate the behavior of customers and the barber.

4. Detailed Implementation

Semaphore Initialization:

Four semaphores are used:

- waitingRoom: Controls access to the waiting room.
- barberChair: Ensures exclusive access to the barber chair.
- barberPillow: Used by customers to wake the barber.
- seatBelt: Keeps the customer in the chair until the haircut is done.

Barber and Customer Threads

Customer Function:

- Arrive at the barber shop and try to enter the waiting room.
- If entry is successful, attempt to acquire the barber chair.
- Wake the barber if he is asleep and wait for the haircut to finish.
- Leave the barber shop.

Barber Function:

- Continuously check for customer presence.
- Sleep if no customers are in the waiting room.
- Perform the haircut and release the customer.

5. Compilation and Running

Run the following commands:

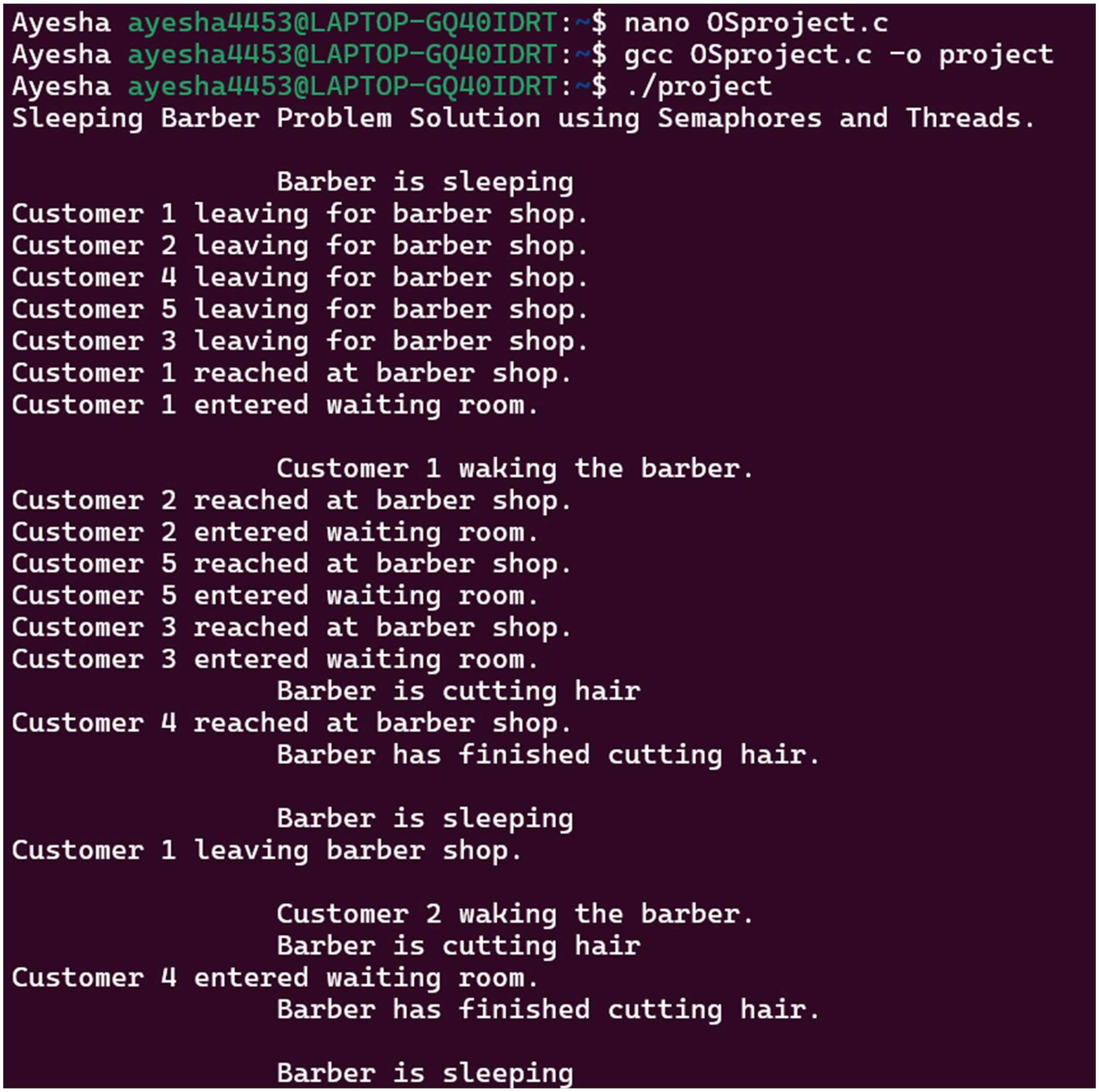
- gcc OSproject.c -o project
- ./project

6. Code:

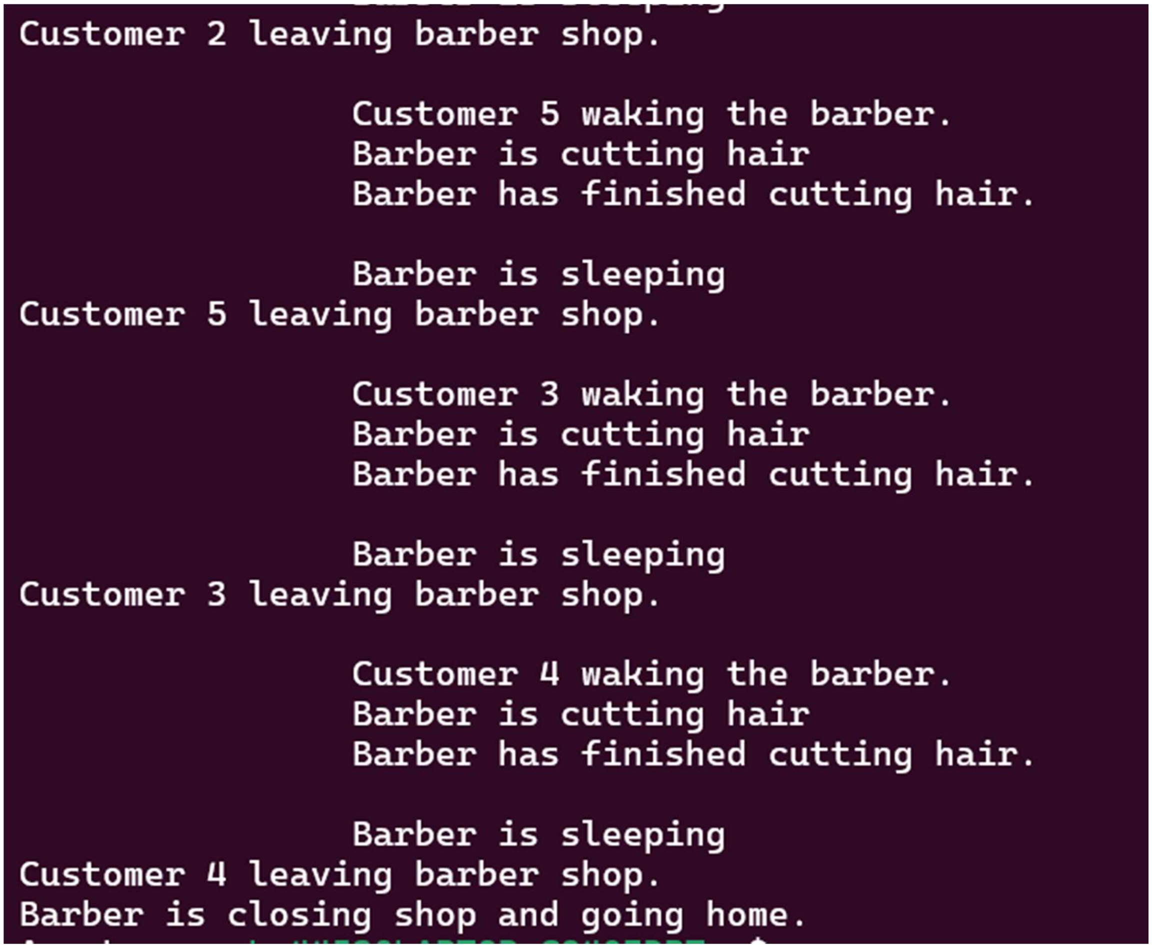
```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>

#define MAX_CUSTOMERS 10

sem_t waitingRoom;
sem_t barberChair;
sem_t barberPillow;
sem_t seatBelt;
int flag = 0;
int temp = 5;

void *customer (void *num) {
   int c = *(int *)num;
   printf("Customer %d leaving for barber shop.\n", c);
   sleep(5);
   printf("Customer %d reached at barber shop.\n", c);
   sem_wait(&waitingRoom);
   printf("Customer %d entered waiting room.\n", c);

   sem_wait(&barberChair);
   sem_post(&waitingRoom);
   printf("\n\t\tCustomer %d waking the barber.\n", c);
   sem_post(&barberPillow);
   sem_wait(&seatBelt);
   sem_post(&barberChair);
```

```
    printf("Customer %d leaving barber shop.\n", c);
    return NULL;
}

void *barber(void *data) {
    while (!flag) {
        printf("\n\t\tBarber is sleeping\n");
        sem_wait(&barberPillow);
        if (!flag) {
            printf("\t\tBarber is cutting hair\n");
            sleep(5);
            printf("\t\tBarber has finished cutting hair.\n");
            temp--;
            sem_post(&seatBelt);
        } else {
            printf("Barber is closing shop and going home.\n");
        }
    }
    return NULL;
}

int main(void) {
    pthread_t barber_id;
    pthread_t customer_id[MAX_CUSTOMERS];
    int numCustomers = 5;
    int numChairs = 3;
    int i;
    int cus[MAX_CUSTOMERS];

    printf("Sleeping Barber Problem Solution using Semaphores and Threads.\n");

    sem_init(&waitingRoom, 0, numChairs);
    sem_init(&barberChair, 0, 1);
    sem_init(&barberPillow, 0, 0);
    sem_init(&seatBelt, 0, 0);

    pthread_create(&barber_id, NULL, barber, NULL);

    for (i = 0; i < numCustomers; i++) {
        cus[i] = i + 1;
        pthread_create(&customer_id[i], NULL, customer, (void *)&cus[i]);
    }

    for (i = 0; i < numCustomers; i++) {
        pthread_join(customer_id[i], NULL);
    }

    if(temp == 0) {
        flag = 1;
        sem_post(&barberPillow);
    }

    pthread_join(barber_id, NULL);

    sem_destroy(&waitingRoom);
    sem_destroy(&barberChair);
    sem_destroy(&barberPillow);
    sem_destroy(&seatBelt);

    return 0;
}
```

7. Output



```
Ayesha ayesha4453@LAPTOP-GQ40IDRT:~$ nano OSproject.c
Ayesha ayesha4453@LAPTOP-GQ40IDRT:~$ gcc OSproject.c -o project
Ayesha ayesha4453@LAPTOP-GQ40IDRT:~$ ./project
Sleeping Barber Problem Solution using Semaphores and Threads.

                Barber is sleeping
Customer 1 leaving for barber shop.
Customer 2 leaving for barber shop.
Customer 4 leaving for barber shop.
Customer 5 leaving for barber shop.
Customer 3 leaving for barber shop.
Customer 1 reached at barber shop.
Customer 1 entered waiting room.

                Customer 1 waking the barber.
Customer 2 reached at barber shop.
Customer 2 entered waiting room.
Customer 5 reached at barber shop.
Customer 5 entered waiting room.
Customer 3 reached at barber shop.
Customer 3 entered waiting room.
                Barber is cutting hair
Customer 4 reached at barber shop.
                Barber has finished cutting hair.

                Barber is sleeping
Customer 1 leaving barber shop.

                Customer 2 waking the barber.
                Barber is cutting hair
Customer 4 entered waiting room.
                Barber has finished cutting hair.

                Barber is sleeping
```



```
Customer 2 leaving barber shop.

                Customer 5 waking the barber.
                Barber is cutting hair
                Barber has finished cutting hair.

                Barber is sleeping
Customer 5 leaving barber shop.

                Customer 3 waking the barber.
                Barber is cutting hair
                Barber has finished cutting hair.

                Barber is sleeping
Customer 3 leaving barber shop.

                Customer 4 waking the barber.
                Barber is cutting hair
                Barber has finished cutting hair.

                Barber is sleeping
Customer 4 leaving barber shop.
Barber is closing shop and going home.
```

8. Conclusion

This project effectively demonstrates the use of semaphores in the Linux environment to solve the sleeping barber problem, showcasing process synchronization and thread management.