Barbershop Simulation

This simulation program uses C programming language to model the interactions between a barber and customers. The program involves the management of a waiting room, customer arrival and departure, as well as the barber's service duration. Using various semaphores, threads, and randomization, the program simulates real-world scenarios found in a barbershop.

GROUP MEMBERS

- QURAT UL AIN 2021-CE-02
- NOOR FATIMA 2021-CE-07
- ALIYA ZAHRA 2021-CE-16
- ASNA ASLAM 2021-CE-21

Overview of the Program

The program is designed to run for a user-specified duration, during which it simulates the operation of a barbershop. It consists of a barber thread and multiple customer threads, controlled by semaphores to enforce synchronization and achieve the intended behavior.

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

void barber();
void customer();

sem_t seatMutex;
sem_t customers;
sem_t smfBarber;

int runningTime;
int numberOfFreeSeats;
int customersCount;
int customerWait;
int getHCn;

pthread_t barberThread;
pthread_t customersThreads[20];
```

Barber's Workflow

1

Customer Arrival

The barber waits for customers to enter the waiting room.

2

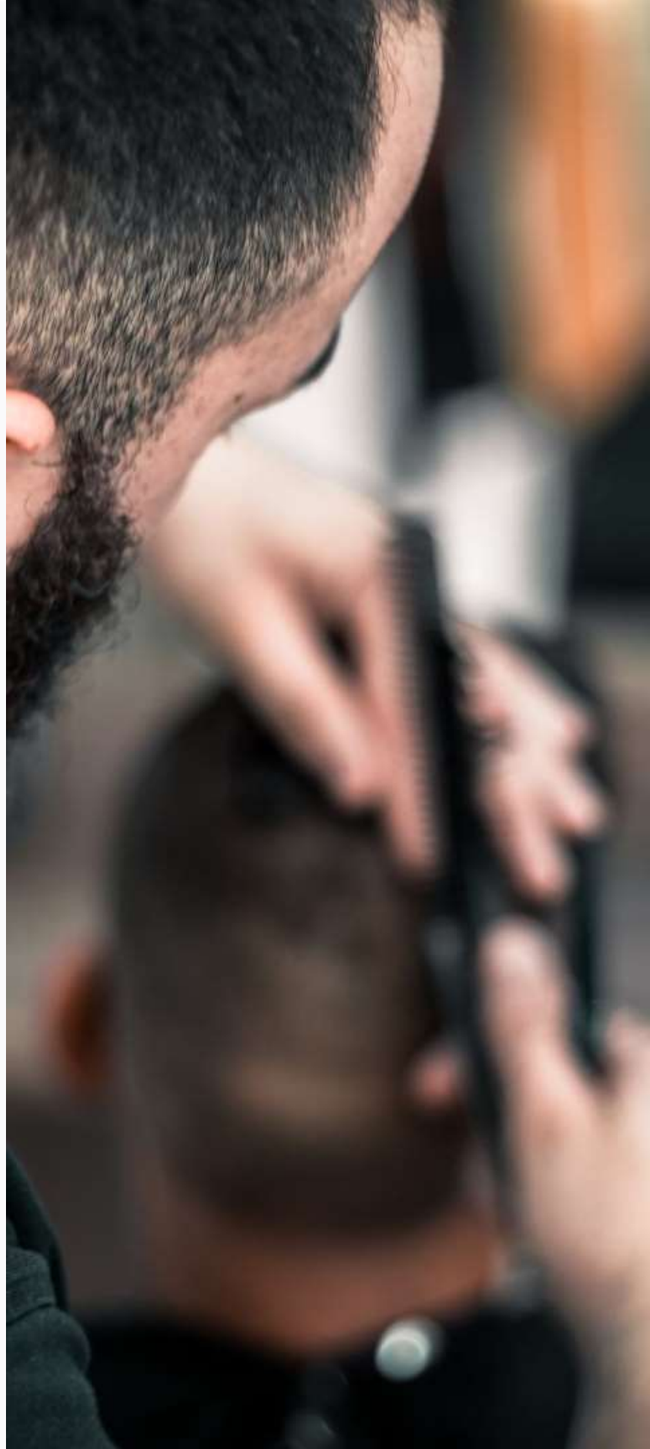
Haircut Process

The barber takes a new customer, allocates time for the haircut, and provides the service.

3

Free Seat Update

The number of free seats in the waiting room is updated after each interaction.



Barber Thread Behavior

1

Serve Customer

Barber serves the customer and notifies the waiting room status.

2

SeatMutex Lock

Ensures mutual exclusion for updating the seat count in the waiting room.

3

Working Duration

Randomized working time for performing the haircut service.

Barber’s Function Code

```
void barber() {
    int workingTime;
    while(1) {
        sem_wait(&customers);
        sem_wait(&seatMutex);
        numberOfFreeSeats += 1;
        workingTime = (rand() % 5) + 1;
        printf("Barber took a new customer, and he will take %d seconds for haircut.\n",workingTime);
        printf("\tNumber of free seats: %d\n",numberOfFreeSeats);
        sem_post(&smfBarber);
        sem_post(&seatMutex);
        sleep(workingTime);
    }
}
```

Customer's Experience

1 Waiting Time Calculation

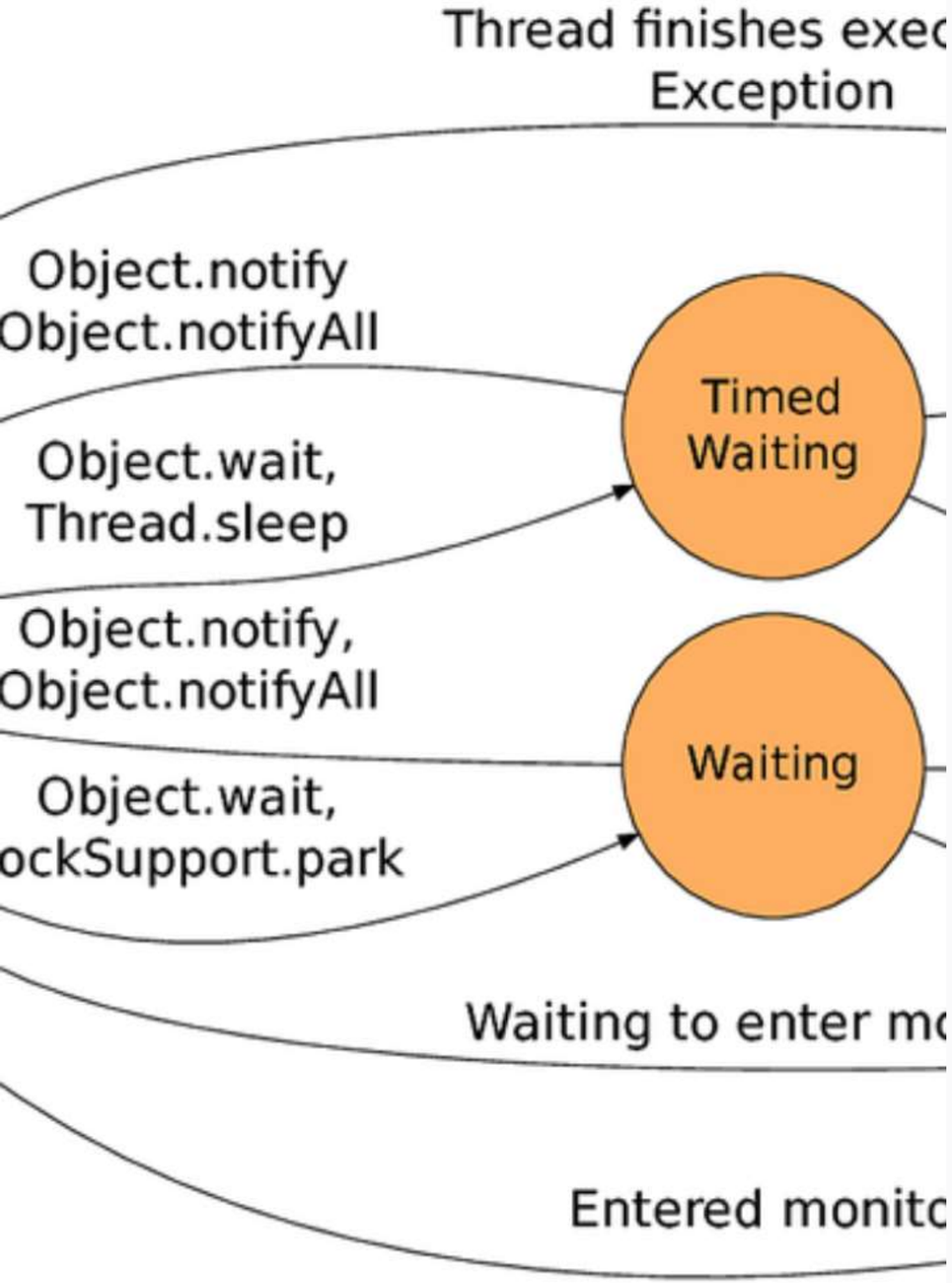
If the waiting room is full, the customer calculates the waiting time before returning.

2 Seating Availability

The customer checks the number of free seats and decides on joining the queue.

3 Haircut Success

The customer is informed whether they got a successful haircut or need to retry.



Customer Thread Actions

- Customer's Thread Activities
- Barber's Synchronization

Customer’s Function Code

```
void customer() {
    int waitingTime;
    int notEnd = 1;
    while(notEnd == 1) {
        sem_wait(&seatMutex);
        if(numberOfFreeSeats <= 0){
            waitingTime = (rand() % customerWait) + 1;
            printf("Customer %u left without haircut, and will come back after %d
            seconds to try again.\n", pthread_self(),waitingTime);
            sem_post(&seatMutex);
            sleep(waitingTime);
        }
        else{
            numberOfFreeSeats -= 1;
            printf("Customer %u is waiting.\n",pthread_self());
            printf("\tNumber of free seats: %d\n",numberOfFreeSeats);
            sem_post(&customers);
            sem_post(&seatMutex);
            sem_wait(&smfBarber);
            printf("Customer %u get a haircut :)\n",pthread_self());
            notEnd = 0;
            getHCn += 1;
        }

    }
}
```

Semaphore Utilization

SeatMutex Semaphore

Controls access to the waiting room seat count and customer allocation.

Barber Semaphore

Regulates the barber's availability to serve the customers.

Customers Semaphore

Manages the customer threads and their arrival at the waiting room.

Main Thread Execution

Running Time Input

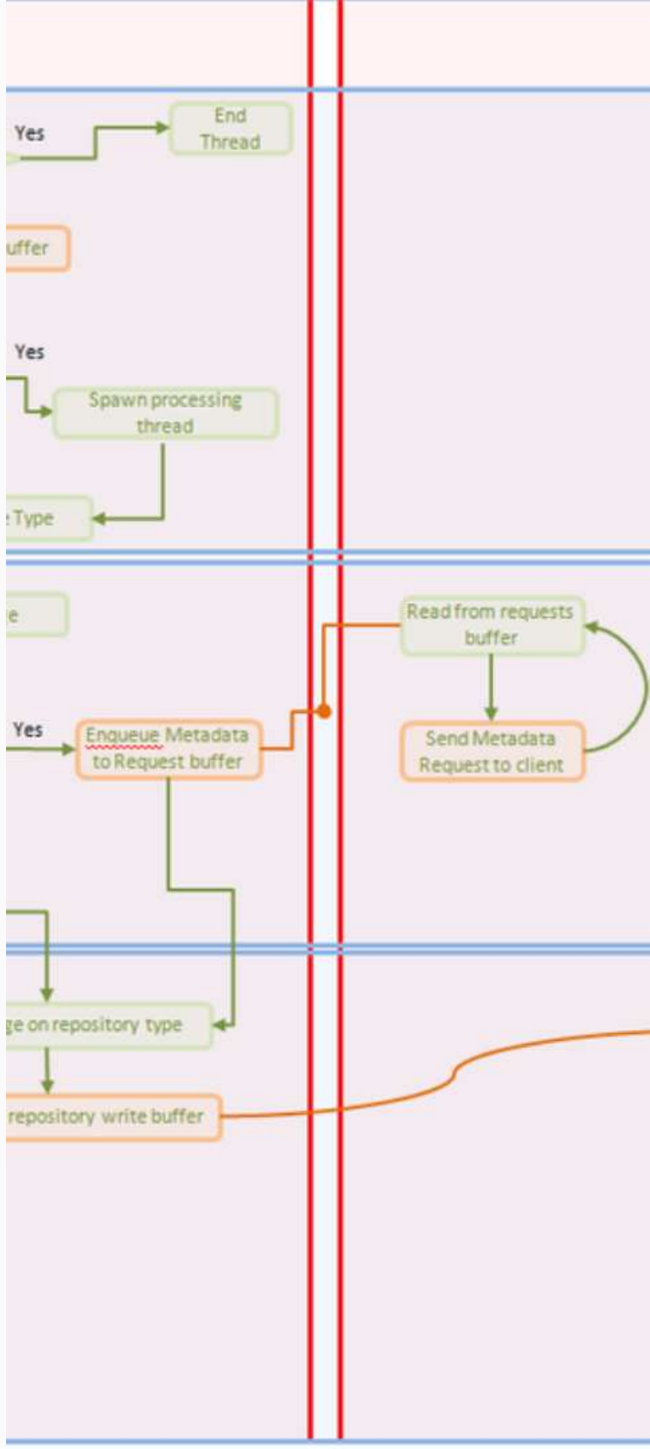
User determines the duration for which the program will run the simulation.

Free Seats Allocation

The allocated number of free seats in the waiting room controls customer entry.

Customer Count

User specifies the number of customers to participate in the simulation program.



Main Function Code

```c
int main() {
    printf("Enter the running time of the program: ");
    scanf("%d",&runningTime);
    printf("Enter the number of free seats: ");
    scanf("%d",&numberOfFreeSeats);
    printf("Enter the customers count: ");
    scanf("%d",&customersCount);
    printf("Enter the maximun waiting time for the customer to come again: ");
    scanf("%d",&customerWait);
    getHCn = 0;
    printf("\nProgram is beginning\n\n");
    sem_init(&seatMutex,0,1);
    sem_init(&customers,0,0);
    sem_init(&smfBarber,0,0);
    pthread_create(&barberThread, NULL, barber, NULL);
    printf("Barber has been created.\n");
    for (int i = 0; i < customersCount; i++){
        pthread_create(&customersThreads[i], NULL, customer, NULL);
        printf("Customer %u has been created.\n",customersThreads[i]);
    }
    sleep(runningTime);
    printf("\n\nEnd of the day :)\n");
    printf("%d out of %d customers get haircut.",getHCn,customersCount);
    exit(0);
}
```


Thank
you