EAST WEST UNIVERSITY

Department of Computer Science and Engineering

Project Report

Course Name : Operating Systems

Course Code: CSE325

Section No: 02

Project : Dentist Problem

Group No : 07

Submitted By:

Name	ID
MD. Muntasir Ahmed Rifat	2022-1-60-333
Saba Tasnim Khan	2022-3-60-049
MD. Sarafat Alam Mozumder	2022-3-60-061

Submitted To:

Dr. Md. Nawab Yousuf Ali

Professor

Department of Computer Science and Engineering

Date of submission: 20th May, 2025

Problem Statement

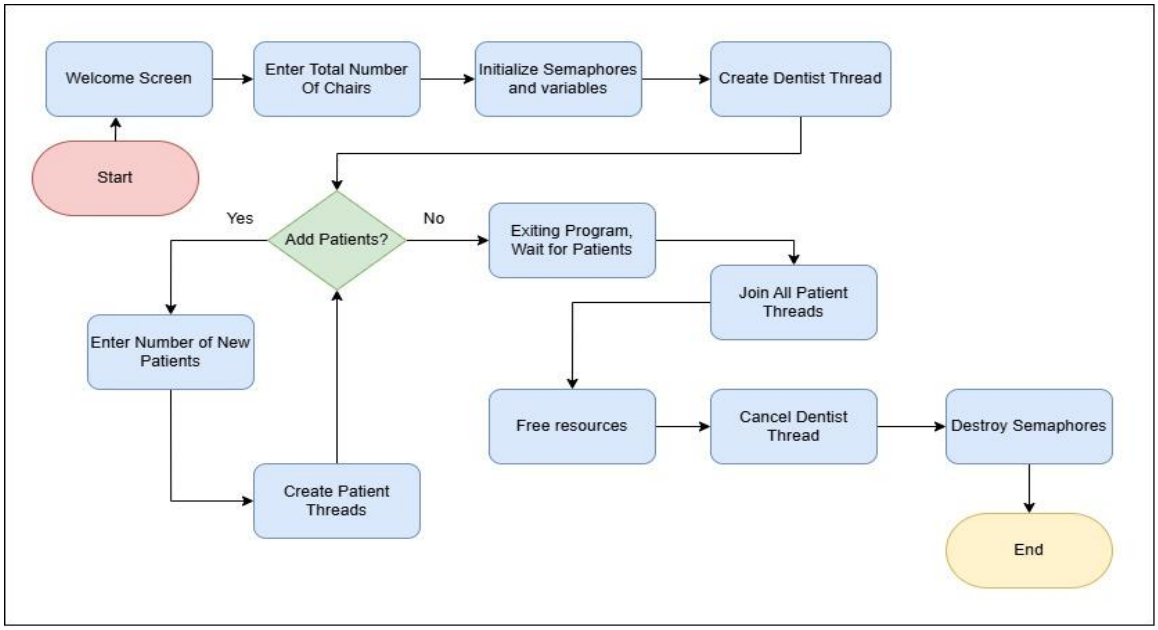
Dr. Smith, a new dental school graduate, has opened a small clinic with limited resources. The clinic has one dentist (Dr. Smith), one treatment chair, and n chairs in the waiting room. A receptionist handles appointments and patient records. Initially, all the chairs are empty.

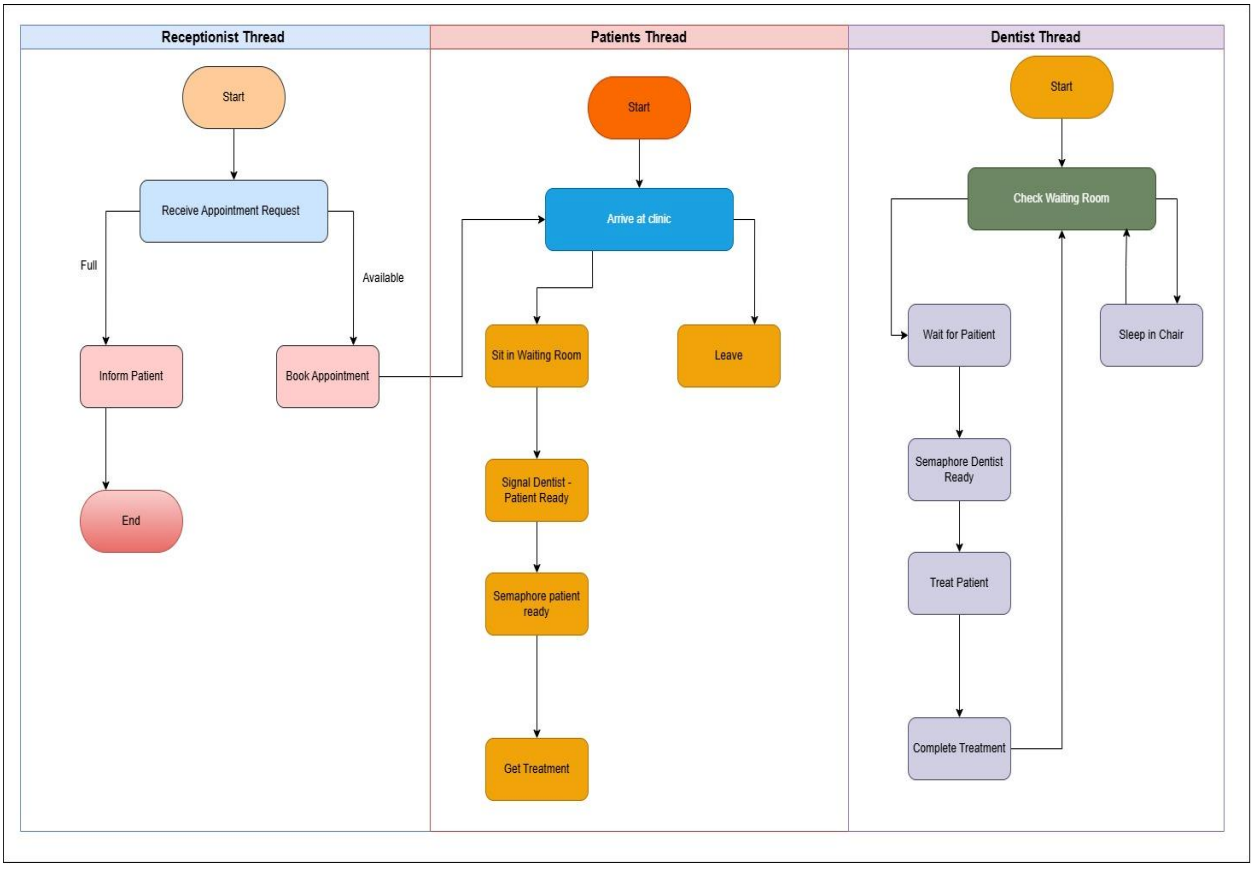
The clinic faces these challenges:

1. Dr. Smith's Availability:
    - Dr. Smith naps when no patients are present and needs to be woken when a patient arrives.
2. Patient Arrivals and Waiting Room:
    - Patients must either wait in available chairs or leave if the waiting room is full.
3. Appointments and Records:
    - The receptionist schedules appointments and keeps patient records.
4. Workflow Management:
    - Patients must be treated in order, and they should leave after treatment to make space for others.
5. Synchronization and Shutdown:
    - The system must handle interactions between Dr. Smith, the receptionist, and patients efficiently.
    - All operations should terminate properly after treating all patients.

The goal is to create a program using POSIX threads, mutex locks, and semaphores to manage patient arrivals, waiting room usage, appointments, and thread termination efficiently.

Flow Chart:


Welcome Screen
Enter Total Number Of Chairs
Initialize Semaphores and variables
Create Dentist Thread
Start
Yes
Add Patients?
No
Exiting Program, Wait for Patients
Join All Patient Threads
Enter Number of New Patients
Free resources
Cancel Dentist Thread
Destroy Semaphores
Create Patient Threads
End


Receptionist Thread
Patients Thread
Dentist Thread
Start
Receive Appointment Request
Full
Available
Inform Patient
Book Appointment
End
Start
Arrive at clinic
Sit in Waiting Room
Leave
Signal Dentist - Patient Ready
Semaphore patient ready
Get Treatment
Start
Check Waiting Room
Wait for Patient
Sleep in Chair
Semaphore Dentist Ready
Treat Patient
Complete Treatment

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h> // For sleep()

sem_t dentistReady;
sem_t seatCountWriteAccess;
sem_t patientReady;
int numberOfFreeSeats;
int patientsWaiting = 0;
int doctorSleeping = 0;

void* dentist1(void* arg) {
    while (1) {
        sem_wait(&seatCountWriteAccess);
        if (patientsWaiting == 0) {
            if (!doctorSleeping) {
                doctorSleeping = 1; // Set the sleeping state
                printf("Doctor Smith is sleeping!\n");
            }
            sem_post(&seatCountWriteAccess);
            sleep(1); // Pause before checking again
        } else {
            doctorSleeping = 0; // Wake up the doctor
            sem_post(&seatCountWriteAccess);
            sem_wait(&patientReady); // Wait for a patient
            sem_wait(&seatCountWriteAccess);
            numberOfFreeSeats++;
            patientsWaiting--; // Decrement patients waiting
            sem_post(&dentistReady);
            sem_post(&seatCountWriteAccess);

            printf("Dentist is treating a patient\n");
            sleep(1); // Simulate treatment time
            printf("Dentist has completed the treatment\n");
        }
    }
}

void* patient(void* arg) {
    int patientId = (int)arg;
    printf("Patient %d arrived at clinic\n", patientId);
    sem_wait(&seatCountWriteAccess);
    if (numberOfFreeSeats > 0) {
        printf("Patient %d is waiting in the waiting room\n", patientId);
        numberOfFreeSeats--;
        patientsWaiting++;
        sem_post(&patientReady);
        sem_post(&seatCountWriteAccess);
        sem_wait(&dentistReady);
        printf("Patient %d is getting treatment\n", patientId);
    } else {
        sem_post(&seatCountWriteAccess);
        printf("No free seats. Patient %d is leaving.\n", patientId);
    }
    return NULL;
}
```

```c
int main() {
    int totalNumberOfPatients;
    int totalNumberOfChairs;
    int morePatients;

    printf("\t\t\t\tWelcome\t\t\t\n");
    printf("\t\t Dental Clinic Problem\t\t\t\n\n\n");
    printf("Enter the total number of chairs: ");
    scanf("%d", &totalNumberOfChairs);

    numberOfFreeSeats = totalNumberOfChairs;

    pthread_t dentistThread;
    pthread_t* patientsThread = NULL; // Start without patients
    int* patientNumber = NULL;
    int currentPatientCount = 0; // Track current patient count

    sem_init(&dentistReady, 0, 0);
    sem_init(&seatCountWriteAccess, 0, 1);
    sem_init(&patientReady, 0, 0);

    pthread_create(&dentistThread, NULL, dentist1, NULL); // Create dentist thread

    while (1) {
        printf("\nDo you want to add patients? (1 for Yes, 0 for No): ");
        scanf("%d", &morePatients);

        if (morePatients == 0) {
            printf("Exiting program. Waiting for all patients to be treated...\n");
            break;
        }

        printf("Enter the number of new patients: ");
        scanf("%d", &totalNumberOfPatients);

        patientsThread = realloc(patientsThread, sizeof(pthread_t) * (currentPatientCount +
        totalNumberOfPatients));
        patientNumber = realloc(patientNumber, sizeof(int) * (currentPatientCount +
          totalNumberOfPatients));

        for (int i = 0; i < totalNumberOfPatients; i++) {
            patientNumber[currentPatientCount + i] = currentPatientCount + i + 1;
            pthread_create(&patientsThread[currentPatientCount + i], NULL, patient,
            (void*)&patientNumber[currentPatientCount + i]);
        }

        currentPatientCount += totalNumberOfPatients;
    }

    for (int i = 0; i < currentPatientCount; i++) {
        pthread_join(patientsThread[i], NULL);
    }
    free(patientsThread);
    free(patientNumber);

    printf("All patients have been treated or left. Shutting down...\n");
    pthread_cancel(dentistThread); // Terminate dentist thread
    pthread_join(dentistThread, NULL);

    sem_destroy(&dentistReady);
    sem_destroy(&seatCountWriteAccess);
    sem_destroy(&patientReady);

    return 0;
}
```

Code Explanation

1. Introduction

- Dr. Smith treats patients sequentially while managing limited waiting room chairs.
- Patients arrive dynamically and either wait, receive treatment, or leave if the waiting room is full.
- Synchronization and proper thread management are achieved using POSIX threads, semaphores,

and mutexes.

2. Initialization

Semaphores:

- dentistReady: Signals when the dentist is ready to treat a patient.
- seatCountWriteAccess: Ensures mutual exclusion when modifying the number of available seats.
- patientReady: Signals when a patient is ready for treatment.

Variables:

- numberOfFreeSeats: Tracks the number of empty chairs in the waiting room.
- patientsWaiting: Tracks the number of patients currently waiting.
- doctorSleeping: Tracks whether the dentist is sleeping.

3. Dentist Thread

Function: dentist1

Continuously checks for patients.

- If no patients are waiting, the dentist sleeps and periodically checks again.

If patients are waiting:

- Wakes up.
- Signals readiness to treat the next patient.
- Simulates treatment and frees up a waiting room chair.

4. Patient Thread

Function: patient

- Represents a patient arriving at the clinic.

Checks for an available chair:

If a chair is available:

- Waits for the dentist to signal readiness.
- Receives treatment.

If no chairs are available:

- Leaves the clinic.

5. Main Function

Input:

- Total number of chairs in the waiting room.
- Number of patients arriving dynamically.

Dynamic Patient Handling:

- Allows the user to add new patients at runtime.
- Patient threads are created for each new patient.

Termination:

- After all patients are treated or leave, the program cleans up resources:
- Joins all patient threads.
- Cancels the dentist thread.
- Frees dynamically allocated memory.
- Destroys semaphores.

Complexity Analysis

Time Complexity:

- Each patient dentist threads thread and performs constant-time operations.
- For n patients, total time complexity is **O(n)**.

Space Complexity:

- Memory is used for patient threads and dynamically allocated arrays.
- For n patients, total space complexity is **O(n)**.

Output

```
muntasir@muntasir:~$ cd Project
muntasir@muntasir:~/Project$ ./clinic
                                        Welcome
                                ABC Dental Clinic


Enter the total number of chairs: 3

Do you want to add patients? (1 for Yes, 0 for No): Doctor Smith is sleeping!
1
Enter the number of new patients: 6
Patient 2 arrived at clinic
Patient 2 is waiting in the waiting room
Patient 3 arrived at clinic
Patient 3 is waiting in the waiting room

Do you want to add patients? (1 for Yes, 0 for No): Patient 1 arrived at clinic
Patient 1 is waiting in the waiting room
Patient 6 arrived at clinic
No free seats. Patient 6 is leaving.
Patient 4 arrived at clinic
No free seats. Patient 4 is leaving.
Patient 5 arrived at clinic
No free seats. Patient 5 is leaving.
Dentist is treating a patient
Patient 2 is getting treatment
Dentist has completed the treatment
Dentist is treating a patient
Patient 3 is getting treatment
Dentist has completed the treatment
Dentist is treating a patient
Patient 1 is getting treatment
Dentist has completed the treatment
Doctor Smith is sleeping!
```

```
Doctor Smith is sleeping!
1
Enter the number of new patients: 4

Do you want to add patients? (1 for Yes, 0 for No): Patient 8 arrived at clinic
Patient 8 is waiting in the waiting room
Patient 9 arrived at clinic
Patient 9 is waiting in the waiting room
Patient 10 arrived at clinic
Patient 10 is waiting in the waiting room
Patient 7 arrived at clinic
No free seats. Patient 7 is leaving.
Dentist is treating a patient
Patient 8 is getting treatment
Dentist has completed the treatment
Dentist is treating a patient
Patient 9 is getting treatment
Dentist has completed the treatment
Dentist is treating a patient
Patient 10 is getting treatment
Dentist has completed the treatment
Doctor Smith is sleeping!
```

```
muntasir@muntasir:~/Project$ ./clinic
                                    Welcome
                                ABC Dental Clinic


Enter the total number of chairs: 0

Do you want to add patients? (1 for Yes, 0 for No): Doctor Smith is sleeping!
1
Enter the number of new patients: 3

Do you want to add patients? (1 for Yes, 0 for No): Patient 2 arrived at clinic
No free seats. Patient 2 is leaving.
Patient 1 arrived at clinic
No free seats. Patient 1 is leaving.
Patient 3 arrived at clinic
No free seats. Patient 3 is leaving.
```

Conclusion

In conclusion, the solution effectively manages Dr. Smith’s dental clinic with limited resources using POSIX threads, semaphores, and mutex locks. It ensures smooth coordination between the dentist, patients, and receptionist. The system handles patient arrivals dynamically, manages the waiting room efficiently, and makes sure the dentist is always available when needed.

By synchronizing patient treatment and managing available seats, the program prevents issues like race conditions and ensures that all resources are properly cleaned up after use. This approach provides an efficient way to simulate clinic operations while demonstrating important concepts in thread management and synchronization