Computer Science and Engineering

Summer-2023

Report of Project-5

Project Name: Dental Clinic Problem

Course Title: Operating System

Course Code: CSE325

Section: 3

Group No: 05

Group Members:

Amin Ocin

2021-3-60-135

Tasnova Haque Mazumdar

2021-3-60-235

Akash Ahmed

2021-3-60-242

Submitted to:

Dr. Md. Nawab Yousuf Ali

Professor

Department of Computer Science and Engineering

East-West University

Dental Clinic Problem

We implemented a solution that synchronizes the activities of the dentist and the patients, by using POSIX threads, mutex locks, and semaphores.

Project Description:

A fresh graduate from a dental school opened his own dental clinic and started to make a living. This dental clinic has only one dentist, one dental chair, and n chairs for waiting for patients if there are any to sit on the chair. Initially, all the chairs were empty. If there is no patient, the dentist sleeps in his own chair. When a patient arrives, he has to wake up the dentist. If there are many patients and the dentist is attending to a patient, then the remaining patients either wait if there are empty chairs in the waiting room or leave if no chairs are empty. Implement a synchronization method to solve the problem. Using POSIX threads, mutex locks, and semaphores implement a solution that synchronizes the activities of the dentist, and the patients. The total number of patients and the number of chairs are passed as command line arguments. Once a patient thread receives treatment from the dentist, it should terminate. Once all the patient threads are terminated, the dentist thread and the main program should be terminated. Your program should work for any number of patients and chairs. Allocate memory for data structures dynamically based on the input parameter(s).

Overview:

A newly graduated dental student opened his own dental clinic, which has one dentist, one dental chair, and n chairs for patients. First, the dentist sleeps in his own chair if there is no patient present at the clinic. In any case, he needs to awaken. Furthermore, patients come to the holding-up region. Assuming there is no void, holding up seats accessible they will leave the center. If not, they'll sit in the waiting area. After completing one patient's treatment the patients will leave and the following patients from the pausing region will show up to the dental seat. At the point when all patients, get treated the specialist will end and go to sleep once more. In this process, the methods are:

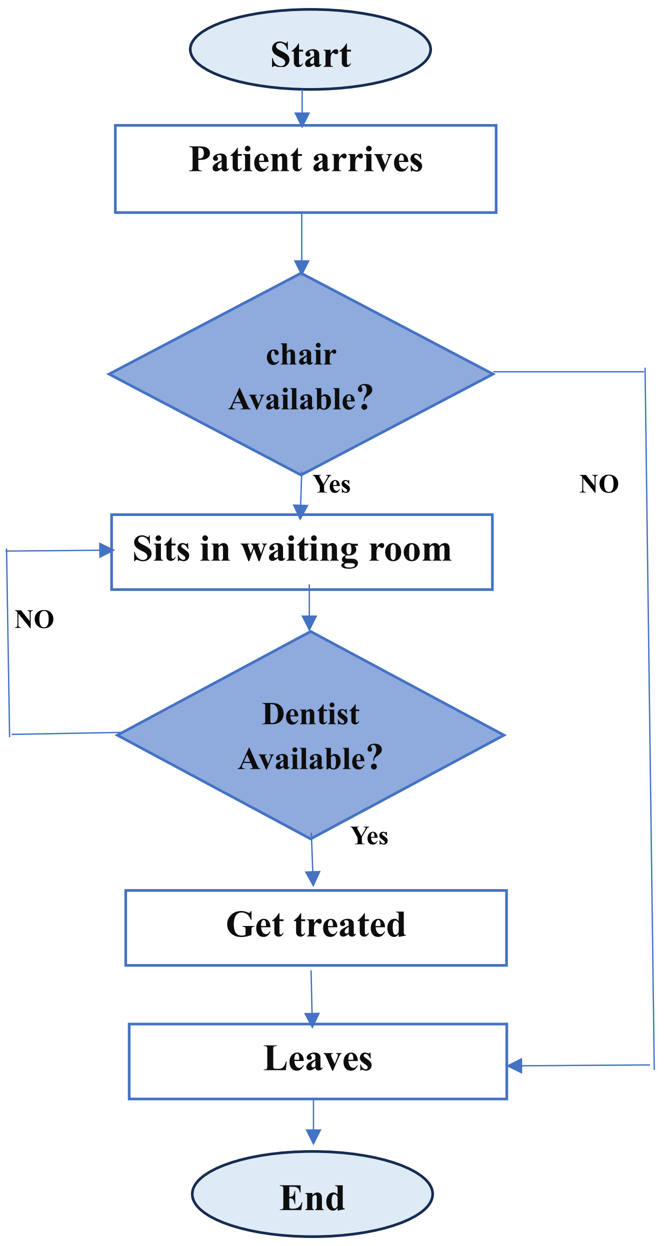
1. Pthread_create()
2. Pthread_join()
3. Pthread mutex init()
4. Sem_init()
5. Sem_wait()
6. Sem_post()
7. Pthread mutex destroy()

Operating System: Linux

To run this " projectCode.c" we used this command line argument in terminal:

```
gcc.projectCode.c -o thread -lpthread./thread
```

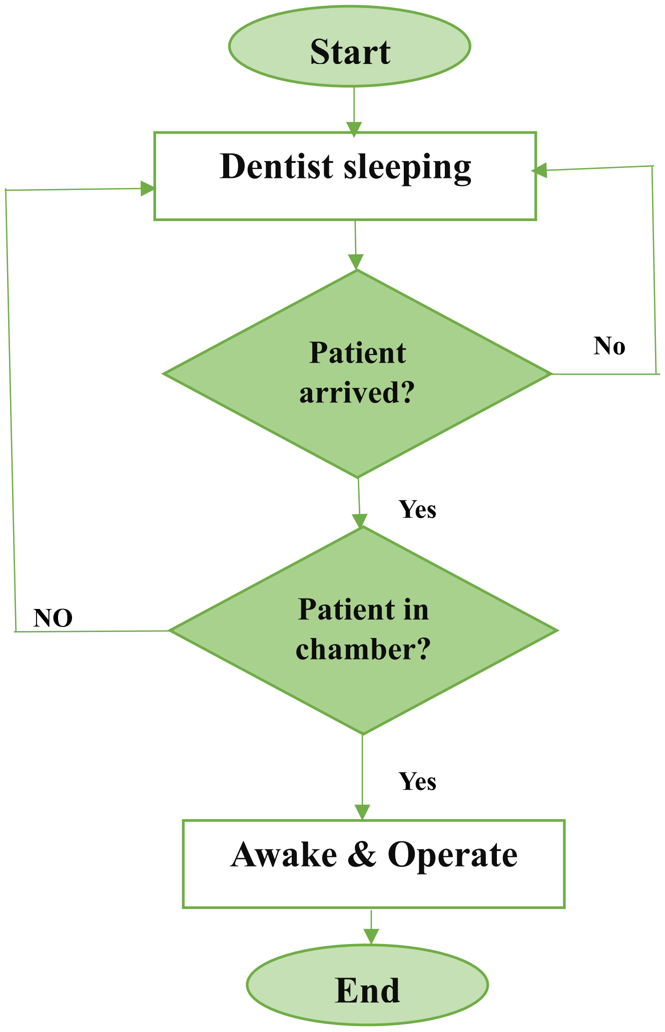
Patient Flowchart



The patient has 3 parts:

1. Patients have to check if there are available chairs to sit in the waiting room.
2. After sitting they have to check if the doctor is available or not.
3. If the doctor is available they take treatment and when done leave.

Dentist Flowchart



Dentist has 3 parts:

1. Dentist sleeps when there is no patient in the chamber
2. When a patient comes into the chamber he awakes up and treats the patient.
3. After the patient leaves the chamber he falls asleep again.

Project code:

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


pthread_mutex_t patient_mutex, waiting_mutex;
sem_t dentist_sem, patient_sem;

int total_patient, total_chair, patient_waiting = 0, patient_arrived = 0;

void *dentist(void *arg) {
    while(1){
        printf("-------- Dentist: Sleeping --------\n\n");

        sem_wait(&dentist_sem);

        printf("-------- Dentist: Awake, Operating on Patient --------
\n\n");

        sleep(rand() % 5 + 1);

        printf("-------- Dentist: Procedure Complete, Patient Leaving ----
----\n\n");

        sem_post(&patient_sem);

        patient_arrived++;

        if(patient_arrived >= total_patient) break;
    }
}

void *patient(void *arg) {

    int t = *(int *)arg;

    sleep(rand() % 10 + t);

    pthread_mutex_lock(&waiting_mutex);

    if(patient_waiting >= total_chair){
        printf("Patient arrived, no empty chair available. Leaving.\n\n");
        patient_arrived++;
```

```
 pthread_mutex_unlock(&waiting_mutex);

        pthread_exit(NULL);
    }

    patient_waiting++;

    printf("Patient arrived and waiting in the waiting room.\n\n");

    pthread_mutex_unlock(&waiting_mutex);

    pthread_mutex_lock(&patient_mutex);

    printf("Patient is in the dentist chamber.\n\n");

    patient_waiting--;

    sem_post(&dentist_sem);
    sem_wait(&patient_sem);


    printf("Patient leaving after treatment.\n\n");

    pthread_mutex_unlock(&patient_mutex);

}

int main() {
    printf("Enter the number of total_patient: ");
    scanf("%d", &total_patient);

    printf("Enter the number of total_chair: ");
    scanf("%d", &total_chair);

    pthread_t patient_threads[total_patient], dentist_thread;

    sem_init(&dentist_sem, 0, 0);
    sem_init(&patient_sem, 0, 0);

    pthread_mutex_init(&patient_mutex, NULL);
    pthread_mutex_init(&waiting_mutex, NULL);

    int t[total_patient];

    for(int i = 0; i < total_patient; i++){
        t[i] = i + 1;
        pthread_create(&patient_threads[i], NULL, patient, &t[i]);
    }

    pthread_create(&dentist_thread, NULL, dentist, NULL);
    pthread_join(dentist_thread, NULL);
```

```
for(int i = 0; i < total_patient; i++){
        pthread_join(patient_threads[i], NULL);
    }

    sem_destroy(&dentist_sem);
    sem_destroy(&patient_sem);

    pthread_mutex_destroy(&waiting_mutex);
    pthread_mutex_destroy(&patient_mutex);

    return 0;
}
```