

Department of Computer Science and Engineering

Course Code: CSE325

Course Title: Operating Systems

Section: 02

Semester: Summer 23

**Project Report**

**Submitted to:**

**Dr. Md. Nawab Yousuf Ali**

Professor

Department of Computer Science and Engineering

East West University

**Submitted by:**

**Name:** B M Shahria Alam

**ID:** 2021-3-60-016

**Name:** Tasmiah Rahman Orpa

**ID:** 2021-3-60-021

**Name:** Sidratul Moontaha

**ID:** 2021-3-60-048

**Date of submission:** 5th September, 2023.

**Dental Clinic Problem**

Using POSIX threads, mutex locks, and semaphores, I implemented a solution that synchronizes the activities of the doctor and the.

**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 are 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 they 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, 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 fresh dental graduate student gets help in his own dental clinic where only one dentist, one dental chair and n chairs available for patients. First if there is no patient in the clinic, the dentist sleeps in his own chair. Otherwise he has to wake up. Secondly patients come to waiting area. If there is no empty waiting chairs available they will leave the clinic. Otherwise they will wait in the waiting area. After complete one patients treatment the patients will leave and the next patients from the waiting area will arrive to the dental chair. When all patients, get treated the doctor will terminated and goes to sleep again. 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()

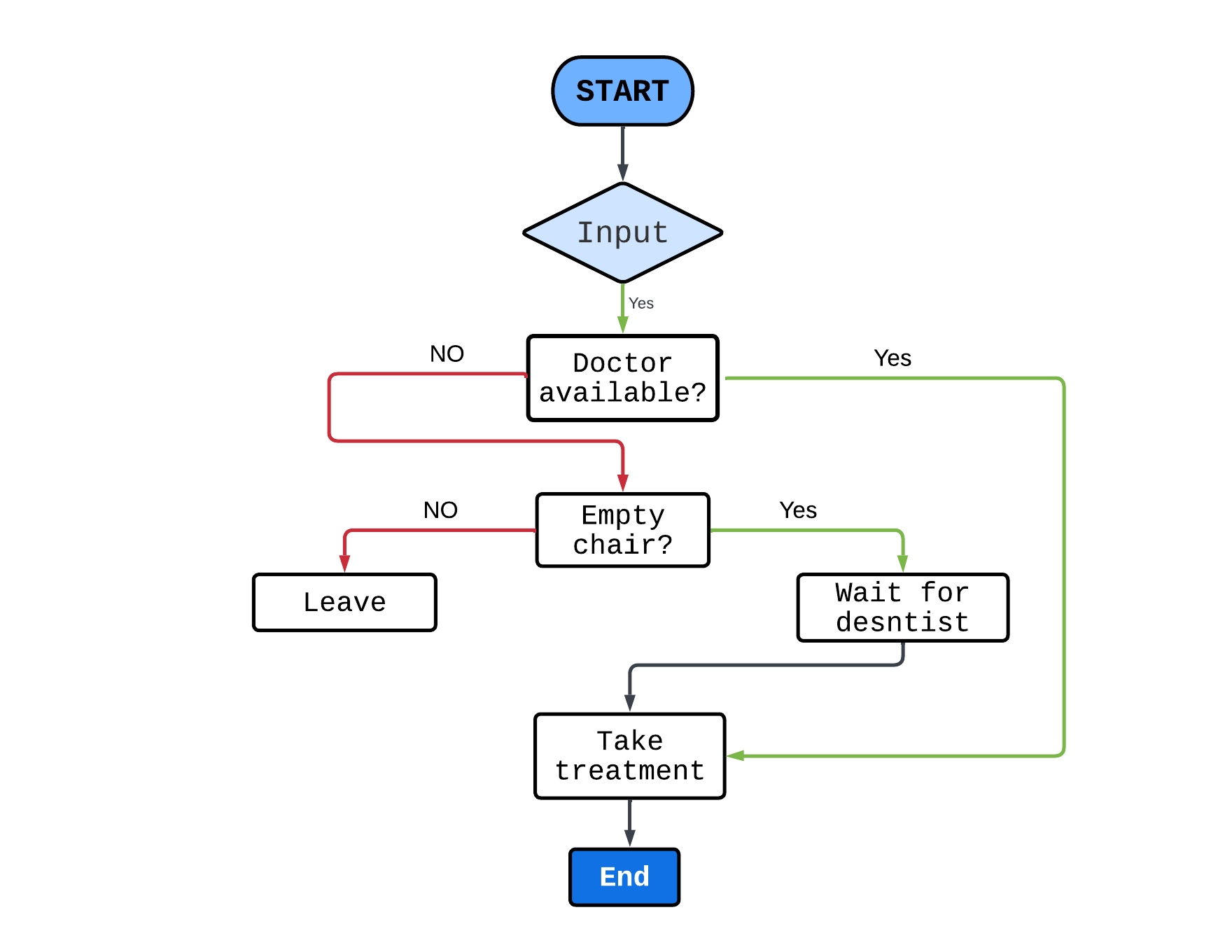
7. Pthread mutex destroy()

**Operating System:** Linux

**To run this “ProjectCode.c” file, we used these command line arguments in terminal**:

gcc ProjectCode.c -o thread -lpthread ./thread

**Flowchart**



**Patients has 3 parts:**

**1.** Patients has to start programming and seek treatment from the doctor, Patients needs to get a waiting seat. After coming to get seat, if patient does not find any waiting seat, then has to go back to programming.

**2.** If a patient gets a seat, then has to wait in the queue for tutor to be treated.

**3.** After getting treatment, the patient will leave the clinic.

**Doctor has 3 parts:**

**1.** The Doctor waits for patients to come in the clinic.

**2.** Prioritize patients according to who has come first.

**3.** If all patients leave, then the doctor goes to sleep again.

**Project Code**

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <stdlib.h>

#include <stdbool.h>

#include <unistd.h>

#include <time.h>

sem\_t sem1;

sem\_t sem2;

sem\_t sem3;

pthread\_mutex\_t mutex;

int limit=0, x=0, count=0, tchair=0, num, flag=0;

void \*Patients(void \*arg)

{

sem\_wait(&sem1);

int num = (int\*)arg;

pthread\_mutex\_lock(&mutex);

if (count >tchair)

{

printf("\nNo more chairs available. Patient %d is leaving from the DENTAL CLINIC.", num);

pthread\_mutex\_unlock(&mutex);

pthread\_exit(num);

}

if (count!=0)

{

printf("Patient %d is waiting in the chair in the CLINIC.\n",num);

}

count++;

pthread\_mutex\_unlock(&mutex);

sem\_wait(&sem2);

flag++;

sleep(1);

if (flag==1)

{

printf("\n\nPatient %d wakes up the dentist\n",num);

}

x=num;

sem\_post(&sem3);

}

void \*Dentist(void \*arg)

{

while (true)

{

if (limit)

{

printf("\nTHE DENTIST TERMINATES.\n");

break;

}

sem\_wait(&sem3);

printf("\nThe dentist is treating patient '%d' \n", x);

printf("Patient %d is treated successfully.\n", x);

printf("Patient %d is leaving.\n", x);

sem\_post(&sem2);

}

}

int main()

{

pthread\_t dentist, \*arr\_patients;

int patient,chair;

printf("\n\n\n\*\*\*\*\*\*\*\*\*\*\*\* DENTAL CLINIC PROBLEM \*\*\*\*\*\*\*\*\*\*\*\*\n\n\n");

printf("Number of total patients: ");

scanf("%d", &patient);

printf("Number of total chairs: ");

scanf("%d", &chair);

printf("\n");

tchair= chair;

sem\_init(&sem1, 0, patient);

sem\_init(&sem2, 0, 1);

sem\_init(&sem3, 0, 0);

pthread\_mutex\_init(&mutex, 0);

if (patient==0) {printf("Dentist is sleeping\n");}

arr\_patients=(pthread\_t\*)malloc(patient\*sizeof(pthread\_t));

pthread\_create(&dentist,NULL,Dentist,NULL);

for (int i=0;i<patient;i++)

{

pthread\_create(&arr\_patients[i],NULL,Patients, (void\*)i+1);

}

for (int i=0;i<patient;i++)

{

pthread\_join(arr\_patients[i], NULL);

}

limit=1;

sem\_post(&sem3);

pthread\_join(dentist, NULL);

pthread\_mutex\_destroy(&mutex);

sem\_destroy(&sem1);

sem\_destroy(&sem2);

sem\_destroy(&sem3);

}