Operating Systems

**CS4348**

**Project #2: Threads and Semaphores**

**Due Date: Saturday, April 6, 2024**

## I. Project Organization

This project will study the coordination of multiple threads using semaphores.

You should do the following pieces to complete your project. Each piece is explained below:

* Design 40 points
* Code 25 points
* Output 25 points
* Summary 10 points

# Design

The design should consist of two things: (1) a list of every semaphore, its purpose, and its initial value, and (2) pseudocode for each function. The pseudocode should be similar to the pseudocode shown in the textbook for the barbershop problem. Every wait and signal call must be included in the pseudocode.

# Code

Your code should be nicely formatted with plenty of comments. The code should be easy to read, properly indented, employ good naming standards, good structure, and should correctly implement the design. Your code should match your pseudocode.

# Output

Output will be graded by running your program.

# Summary

The summary section should discuss your simulation, any difficulties encountered, what was learned, and results. It should be at least one page in length.

## II. Project Description

**Language/Platform**

This project must target a Unix platform and execute properly on our cs1 or csgrads1 Linux server.

The project must be written in C, C++, or Java.

If using C or C++, you must use POSIX pthreads and semaphores (no mutexes, locks, etc.)

If using Java, you must use Java Threads and Java Semaphores (java.util.concurrent.Semaphore).

You should not use the “synchronized” keyword in Java.

You should not use any Java classes that have built-in mutual exclusion.

Any mechanisms for thread coordination other than the semaphore are not allowed.

### Doctor’s Office Simulation

This project will simulate a visit to the doctor’s office. It is similar to the “barbershop” example in the textbook.

**Overview**

The clinic to be simulated has doctors, each of which has their own nurse. Each doctor has an office of his or her own in which to visit patients. Patients will enter the clinic to see a doctor, which should be randomly assigned. Initially, a patient enters the waiting room and waits to register with the receptionist. Once registered, the patient sits in the waiting room until the nurse calls. The receptionist lets the nurse know a patient is waiting. The nurse directs the patient to the doctor’s office and tells the doctor that a patient is waiting. The doctor visits the patient and listens to the patient’s symptoms. The doctor advises the patient on the action to take. The patient then leaves.

**Threads**

Receptionist – one thread

Doctor – one thread each, maximum of 3 doctors

Nurse – one per doctor thread, identifier of doctor and corresponding nurse should match

Patient – one thread each, up to 15 patients

**Inputs**

The program should receive the number of doctors and patients as command-line inputs.

**Other rules:**

1. All mutual exclusion and coordination must be achieved with semaphores.
2. A thread may not use sleeping as a means of coordination.
3. Busy waiting (polling) is not allowed.
4. Mutual exclusion should be kept to a minimum to allow the most concurrency.
5. Each thread should only print its own activities. The patient threads prints patient actions and the doctor threads prints doctor actions, etc.
6. Your output must include the same information and the same set of steps as the sample output.

**Output**

1. Each step of each task of each thread should be printed to the screen with identifying numbers so it is clear which threads are involved.
2. Begin by printing the number of patients, nurses, and doctors in this run.
3. Thread activity output sample. The wording in your output should exactly match the sample:

Run with 3 patients, 3 nurses, 3 doctors

Patient 0 enters waiting room, waits for receptionist

Receptionist registers patient 0

Patient 0 leaves receptionist and sits in waiting room

Patient 2 enters waiting room, waits for receptionist

Nurse 0 takes patient 0 to doctor's office

Receptionist registers patient 2

Patient 0 enters doctor 0's office

Patient 2 leaves receptionist and sits in waiting room

Patient 1 enters waiting room, waits for receptionist

Nurse 2 takes patient 2 to doctor's office

Receptionist registers patient 1

Patient 2 enters doctor 2's office

Doctor 0 listens to symptoms from patient 0

Patient 1 leaves receptionist and sits in waiting room

Patient 0 receives advice from doctor 0

Doctor 2 listens to symptoms from patient 2

Patient 2 receives advice from doctor 2

Nurse 1 takes patient 1 to doctor's office

Patient 1 enters doctor 1's office

Doctor 1 listens to symptoms from patient 1

Patient 1 receives advice from doctor 1

Patient 0 leaves

Patient 2 leaves

Patient 1 leaves

Simulation complete

## III. Project Guidelines

### Submitting

Your final project should work correctly on cs1 or csgrads1.

Submit your project on eLearning. Include in your submission the following files:

1. ‘design.xxx’ where xxx is doc, docx, or pdf.
2. ‘summary.xxx’ where xxx is doc, docx, or pdf.
3. ‘project2.c’, ‘project2.cpp’, or ‘Project2.java’ along with any other source files.

### Partial or Missing Submissions

It is your responsibility to upload all of the right files on time. It is recommended that you double-check the files you upload to make sure they are the right ones. Once the deadline passes, changes to the submission are not accepted without a late penalty.

### Academic Honesty

This is an individual project. All work must be your own. Comparison software may be used to compare the work of all students. Similar work will be reported to the Office of Community Standards and Conduct for investigation.

### Resources

The web has many articles on threads and there are books available on threads. The course website also contains example source code.

### Grading

The written portions will be graded subjectively based on completeness and quality. The code will be graded based on points allocated for each key part of the processing as determined by the instructor.

The output will be graded based on expected results for the input programs. A script will be used to automatically compile your program, execute it, and compare the output to the expected output. For this reason, it is very important that your program compiles and executes successfully on the cs1 server (or csgrads1) as shown below:

Java:  
 compile: javac Project2.java  
 run: java Project2

C/C++:

compile: gcc project2.c -o project2 or g++ project2.cpp -o project2  
 run: project2