aOperating Systems

**CS 4348**

**Project #2: Threads**

**Due Date: Saturday, April 30, 2022**

## 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 may not use the “synchronized” keyword in Java for mutual exclusion.

You may not use Java data structures that have built-in mutual exclusion.

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

### Theater Simulation

A theater is simulated by using threads to model customer and employee behavior. The objective is to use threads for each person and to use semaphores to coordinate their activities.

This project is similar to the “barbershop” example in the textbook. The threads to be used are as follows:

Customer:

1. 50 customers are in line at the box office when the simulation starts (1 thread per customer).
2. Movie titles and seats available are read from an input file.
3. Each customer randomly chooses a movie and requests to buy a ticket from the box office.
4. If the movie the customer wants to see is sold out, the customer leaves.
5. The customer must see the ticket taker before entering the theater’s main lobby.
6. Each customer randomly chooses whether to visit the concession stand (50% chance to visit it).
7. Each customer randomly visiting the concession stand randomly chooses popcorn, soda, or both.

Box Office Agents:

1. Two created initially, one thread each.
2. Serves next customer in line.
3. Maintains count of tickets sold for each movie—does not sell ticket if movie is sold out.
4. If movie is sold out, this is communicated to the customer.

Ticket Taker:

1. Created initially, one thread.
2. Serves next customer in line.
3. Takes and tears ticket.

Concession Stand Worker:

1. Created initially, one thread.
2. Serves next customer in line.
3. Gets order from customer.
4. Fills order.

Main

1. Creates all threads and joins all customer threads.
2. When last customer has exited, ends the simulation.

Other rules:

1. Some actions take time (see table). The thread should sleep 1/60 of the time shown.
2. Each activity of each thread should be printed with identification (e.g., customer 1).
3. A thread may not use sleeping as a means of coordinating with other threads.
4. Busy waiting (polling) is not allowed.
5. Mutual exclusion should be kept to a minimum to allow the most concurrency.
6. The semaphore value may not obtained and used as a basis for program logic.
7. Each thread should print when it is created, and each customer should print when it is joined.
8. All activities of a thread should only be output by that thread.
9. Your output must include the same information and the same set of steps as the sample output.

Sample output:

Your project’s output should match the wording of the sample output.

Box office agent 0 created

Box office agent 1 created

Ticket taker created

Concession stand worker created

Theater is open

Customer 1 created, buying ticket to Toy Story

Customer 2 created, buying ticket to Beauty and the Beast

Box office agent 0 serving customer 1

Customer 0 created, buying ticket to Wall-E

Box office agent 1 serving customer 2

Box office agent 1 sold ticket for Beauty and the Beast to customer 2

Box office agent 0 sold ticket for Toy Story to customer 1

Customer 1 in line to see ticket taker

Customer 2 in line to see ticket taker

Box office agent 1 serving customer 0

Ticket taken from customer 1

Customer 1 enters theater to see Toy Story

Ticket taken from customer 2

Customer 2 enters theater to see Beauty and the Beast

Box office agent 1 sold ticket for Wall-E to customer 0

Customer 0 in line to see ticket taker

Ticket taken from customer 0

Customer 0 in line to buy Popcorn

Order for Popcorn taken from customer 0

Popcorn given to customer 0

Customer 0 receives Popcorn

Customer 0 enters theater to see Wall-E

Joined customer 0

Joined customer 1

Joined customer 2

Task Table

|  |  |  |
| --- | --- | --- |
| Actor | Step | Time (seconds) |
| Ticket Taker | Tear Ticket | 15 |
| Concession Stand Worker | Fill Order | 180 |
| Box Office Agent | Sell Ticket | 90 |

## III. Project Guidelines

### Submitting

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.
4. ‘readme.txt’ containing:
   1. the complete command line used to compile your program
   2. the complete command line used to run your program
   3. any other details the TA should know

### 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 Judicial Affairs for investigation.

### 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.

### Resources

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