Operating Systems

**CS4348**

**Project #2: Exploring Multiple** **Threads**

**Due Date: Saturday, October 28, 2023**

## I. Project Organization

This project will study the coordination of multiple threads.

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 Linux server.

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

If using C or C++, you must use POSIX pthreads and semaphores.

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.

### Hotel Simulation

A hotel is simulated by using threads and semaphores to model customer and employee behavior.

This project is similar to the “barbershop” example in the textbook. The following rules apply:

**Overview**

The hotel to be simulated has two employees at the front desk to register guests and two bellhops to handle guests’ bags. A guest will first visit the front desk to get a room number. The front desk employee will find an available room and assign it to the guest. If the guest has less than 3 bags, the guest proceeds directly to the room. Otherwise, the guest visits the bellhop to drop off the bags. The guest will later meet the bellhop in the room to get the bags, at which time a tip is given.

Threads:

Guest:

1. 25 guests visit the hotel (1 thread per guest created at start of simulation).
2. Each guest has a random number of bags (0-5).
3. A guest must check in to the hotel at the front desk.
4. Upon check in, a guest gets a room number from the front desk employee.
5. A guest with more than 2 bags requires a bellhop.
6. The guest enters the assigned room.
7. Receives bags from bellhop and gives tip (if more than 2 bags).
8. Retires for the evening.

Front Desk:

1. Two employees at the front desk (1 thread each).
2. Checks in a guest, finds available room, and gives room number to guest.

Bellhop:

1. Two bellhops (1 thread each).
2. Gets bags from guest.
3. The same bellhop that took the bags delivers the bags to the guest after the guest is in the room.
4. Accepts tip from guest.

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 print when it is created, and main should print when it joins the customer threads.
6. Each thread should only print its own activities. The guest threads print guest actions and the employee threads print their own actions.
7. Your output must include the same information, same wording, and the same set of steps as the sample output. Of course, each run can be different depending on the order of thread execution and the random assignments made.

**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. Thread output sample for 3 guests. The wording in your output should exactly match the sample.

Simulation starts

Front desk employee 0 created

Front desk employee 1 created

Bellhop 0 created

Bellhop 1 created

Guest 0 created

Guest 1 created

Guest 2 created

Guest 0 enters hotel with 1 bag

Guest 1 enters hotel with 4 bags

Guest 2 enters hotel with 3 bags

Front desk employee 0 registers guest 0 and assigns room 1

Front desk employee 1 registers guest 1 and assigns room 2

Guest 0 receives room key for room 1 from front desk employee 0

Guest 1 receives room key for room 2 from front desk employee 1

Front desk employee 0 registers guest 2 and assigns room 3

Guest 0 enters room 1

Guest 2 receives room key for room 3 from front desk employee 0

Guest 1 requests help with bags

Guest 0 retires for the evening

Guest 0 joined

Guest 2 requests help with bags

Bellhop 1 receives bags from guest 2

Bellhop 0 receives bags from guest 1

Guest 1 enters room 2

Guest 2 enters room 3

Bellhop 0 delivers bags to guest 1

Bellhop 1 delivers bags to guest 2

Guest 1 receives bags from bellhop 0 and gives tip

Guest 2 receives bags from bellhop 1 and gives tip

Guest 2 retires for the evening

Guest 1 retires for the evening

Guest 1 joined

Guest 2 joined

Simulation ends

## 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.
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.

### 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.

### 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.

### Resources

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