Operating Systems

**CS4348**

**Project #2: Threads**

**Due Date: Saturday, October 26, 2019**

## 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 20 points
* Output 20 points
* Summary 20 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.

Other approaches require approval.

### Bank Simulation

A bank 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:

Customer:

1. 5 created initially, one thread each.
2. Each customer will make 3 visits to the bank.
3. Each customer starts with a balance of $1000.
4. On each visit a customer is randomly assigned one of the following tasks:
   1. make a deposit of a random amount from $100 to $500 in increments of $100
   2. make a withdrawal of a random amount from $100 to $500 in increments of $100
      * withdrawals may exceed the balance
   3. request a loan of a random amount from $100 to $500 in increments of $100
5. Steps for each task are defined in the task table.

Bank Teller:

1. Two created initially, one thread each.
2. Serves next customer in the teller line.
3. Processes customer request and updates customer balance.

Loan Officer:

1. One created initially as one thread.
2. Serves next customer in line for a loan.
3. Approves loan request and updates customer balance by adding the loan amount.

Main

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

Other rules:

1. Each activity of each thread should be printed with identification (e.g., customer 1).
2. All mutual exclusion and coordination must be achieved with semaphores.
3. A thread may not use sleeping as a means of coordination.
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 be obtained and used as a basis for program logic.
7. Each customer thread should print when it is created and when it is joined.
8. All activities of a thread should only be output by that thread.
9. Threads use sleep to simulate task time but scaled so that 1 minute only delays 1/10 second.

Output:

Your output must be a logical ordering of events. Your output should match the wording of the sample output. The order of statements, worker assigned, amounts involved and transactions processed may vary.

It is okay to have all threads print ‘created’ before processing begins, but in my output processing begins as soon as a thread is created.

1. Thread activity

Loan Officer created 0

Customer 0 created

Teller 0 created

Teller 0 begins serving customer 0

Customer 0 requests of teller 0 to make a withdrawal of $300

Teller 1 created

Customer 1 created

Loan Officer serving customer 1

Customer 1 requests of loan officer to apply for a loan of $100

Customer 2 created

Teller 1 begins serving customer 2

Customer 2 requests of teller 1 to make a deposit of $500

Teller 0 processes withdrawal of $300 for customer 0

Loan Officer approves loan for customer 1

Teller 1 processes deposit of $500 for customer 2

Customer 3 created

Customer 4 created

Customer 0 gets cash and receipt from teller 0

Customer 1 gets loan from loan officer

Customer 2 gets receipt from teller 1

Teller 0 begins serving customer 4

Customer 4 requests of teller 0 to make a withdrawal of $400

Loan Officer serving customer 3

Customer 3 requests of loan officer to apply for a loan of $300

Teller 1 begins serving customer 0

Customer 0 requests of teller 1 to make a withdrawal of $100

Teller 0 processes withdrawal of $400 for customer 4

Loan Officer approves loan for customer 3

Teller 1 processes withdrawal of $100 for customer 0

Customer 3 gets loan from loan officer

Customer 4 gets cash and receipt from teller 0

Customer 0 gets cash and receipt from teller 1

Loan Officer serving customer 3

Customer 3 requests of loan officer to apply for a loan of $300

Teller 0 begins serving customer 1

Customer 1 requests of teller 0 to make a deposit of $300

Teller 1 begins serving customer 2

Customer 2 requests of teller 1 to make a withdrawal of $200

Loan Officer approves loan for customer 3

Teller 0 processes deposit of $300 for customer 1

Teller 1 processes withdrawal of $200 for customer 2

Customer 3 gets loan from loan officer

Customer 1 gets receipt from teller 0

Customer 2 gets cash and receipt from teller 1

Loan Officer serving customer 0

Customer 0 requests of loan officer to apply for a loan of $300

Teller 0 begins serving customer 4

Teller 1 begins serving customer 3

Customer 4 requests of teller 0 to make a deposit of $300

Customer 3 requests of teller 1 to make a withdrawal of $300

Loan Officer approves loan for customer 0

Teller 1 processes withdrawal of $300 for customer 3

Teller 0 processes deposit of $300 for customer 4

Customer 0 gets loan from loan officer

Customer 3 gets cash and receipt from teller 1

Customer 4 gets receipt from teller 0

Customer 0 departs the bank

Customer 3 departs the bank

Customer 0 is joined by main

Teller 1 begins serving customer 1

Customer 1 requests of teller 1 to make a withdrawal of $500

Teller 0 begins serving customer 2

Customer 2 requests of teller 0 to make a deposit of $300

Teller 1 processes withdrawal of $500 for customer 1

Teller 0 processes deposit of $300 for customer 2

Customer 2 gets receipt from teller 0

Customer 1 gets cash and receipt from teller 1

Customer 2 departs the bank

Customer 1 departs the bank

Teller 0 begins serving customer 4

Customer 4 requests of teller 0 to make a deposit of $500

Customer 1 is joined by main

Customer 2 is joined by main

Customer 3 is joined by main

Teller 0 processes deposit of $500 for customer 4

Customer 4 gets receipt from teller 0

Customer 4 departs the bank

Customer 4 is joined by main

1. At the end of the simulation, display a report as follows:

Bank Simulation Summary

Ending balance Loan Amount

Customer 0 900 300

Customer 1 900 100

Customer 2 1600 0

Customer 3 1300 600

Customer 4 1400 0

Totals 6100 1000

Task Table

|  |  |  |
| --- | --- | --- |
| Customer | Teller | Time (minutes) |
| Task: Make a deposit  Request deposit    Get receipt | Process deposit | 1  4  1 |
| Task: Make a withdrawal  Request withdrawal    Get cash and receipt | Process withdrawal | 1  4  1 |
| Task: Request a loan  Request loan    Get cash and receipt | Process loan | 1  4  1 |

## 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.

### Resources

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