CS3103 Operating Systems
2016-2017 A

Project: Shared Corporate Bank Account Problem

Due Date: Monday, Nov 21st, 2016. Turn in hardcopy in class, and submit source code in Canvas.

I. Project Organization

This project will study the coordination of multiple threads. The project can be done in groups, where each group can have a maximum of 3 members. Each group should do the following pieces to complete the project. Each piece is explained below:

• Design	20 points
• Code	35 points
• Output	25 points
• Summary	20 points

Design
List each semaphore or mutex used, its purpose, and its initial value. How synchronization can be achieved. List and describe all the functions/objects/files used in this project. Provide pseudo code of your design.

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 pseudo code.

Output
Output will be graded by the output you submitted.

Summary
The summary section should discuss your simulation, any difficulties encountered, what was learned, and results. Each group member's responsibility and efforts need to be listed in detail. It should be at least one page in length.

Language/Platform

This project can be done on Unix, Linux, or Windows.
The project can be written in C, C++ or Java.

If using Java, you can use Java Semaphores (java.util.concurrent.Semaphore) to achieve the synchronization. If using C or C++, you can use POSIX pthreads and semaphores, or use WIN32 library.

II. Project Description

The reader/writer problem is a classical synchronization problem in which a shared variable/data is being accessed by multiple processes or threads. The situation involves *n* writer threads that revise the value, and *m* reader threads that read the value out.

In this project, you will complete a program which is described as follows. A company has a shared corporate account that can be accessed by many employees. Each employee can do the **deposit**, **withdrawal** and **balance checking** operations to the corporate account, with the help of a bank staff. The corporate account is opened in a bank *B*. The program creates M bank staff threads in the bank *B*, and each staff thread will be assigned a "staff id" in the range of *0 ... m – 1*.

When an employee comes to bank *B*, the program should generate an "employee id". Any bank staff that is available can serve the employee for the deposit, withdrawal and balance checking operations. On the other hand, if there is no bank staff that is currently free, the employee must wait in a queue until some bank staff finishes his work. The employee queue is a buffer that should be **synchronized** among the threads that put the employee into queue, and the thread who brings out the next employee in queue to an available bank staff to be served (similar to the producer/customer problem). Threads should sleep for a period of time T_in, T_out to simulate the time required for an employee to step into the queue, and step out of the queue. To be aware of the queueing procedure, the program should show the working status of the bank staffs, such as "bank staff i is serving the employee j", or "bank staff i is idle now" while running.

Each bank staff can operate on the shared corporate account. Both the deposit and withdrawal operations are the "writers" to the account, while the balance checking operation is the "reader". According to the reader/writer problem, the amount of shared corporate account should be **synchronized** among multiple reader/writer threads. No thread may access the shared account for either reading or writing, while another thread is writing. However, two or more readers can access the shared account at the same time.

Each staff thread should sleep for a period of time T_d, T_w, T_b to simulate the time required to do the deposit, withdrawal and balance checking operations, respectively. For example, to serve a deposit/withdrawal operation, the bank staff first reads out the current amount of the shared corporate account, and then takes T_d/T_w time to implement the deposit/withdrawal operation, and finally, revises the current amount of the shared corporate account. After each deposit, withdrawal and balance checking operation, the current amount of the shared corporate account should be shown in the interface.

Here is an example of input file for you to test the correctness and efficiency of your program. Note that the efficiency of the program is influenced by how to handle the "reader" threads. Efficiency of the program can be improved by allowing the reader threads to access the shared account simultaneously.

Input file 1:

```
M 5
T_d 3
T_w 3
T_b 3
T_in 1
T_out 1
Company1 balance $10,000
Company2 balance $100

Time 0: Employee 1 Company2; deposit $1000
Time 1: Employee 2 Company1; check balance
Time 3: Employee 3 Company1; withdrawal $500
Time 4: Employee 4 Company1; deposit $3000
Time 4: Employee 5 Company2; deposit $4000
Time 4: Employee 6 Company1; check balance
Time 4: Employee 7 Company2; check balance
Time 4: Employee 8 Company1; withdrawal $1500
Time 5: Employee 9 Company1; check balance
Time 6: Employee 10 Company2; withdrawal $1000
```

III. Project Guidelines

Bonus!

Extra functions are encouraged and welcome in the developed system. Your group can get bonus points if some extra functions are implemented, such as:

1. You can develop a reasonable method so that all the customers get served in the shortest amount of time;
2. The system comes with a nice interface to illustrate the process;
3. You implement creative/innovative way of applying synchronization.

Please state clearly in your final report what bonus features have you implemented in your project.

Submission

Submit your project with hard copy and soft copy on Nov 21st, 2016. Include in your submission the following files:

1) A Word document for the design of the project;
2) The source files;
3) A Word document for the summary of the project;
4) Two sample outputs when you run the project.

Each group need to upload the source code (in the form of a zip file) and report at the CS3103 Canvas course website.

Go to the “Assignments” => “Project” => “Submit Assignment” and upload your file.

Academic Honesty

All work must be developed by each group separately. Please write your own code. **All submitted source code will be scanned by anti-plagiarism software**. If the code does not work, please indicate in the report clearly.

Resources

The web has many articles on threads. There are also books available on threads.