CS 3305A: Operating Systems
Department of Computer Science
Western University
Assignment 5/6
Fall 2020
Due Date: Dec 3rd 2020

Purpose

The goals of this assignment are the following:

- Get hands-on experience in developing mutual exclusion / semaphore / critical section techniques/algorithms.
- Gain more experience with the C programming language from an OS's mutual exclusion / semaphore / critical section perspective.

Assignment Description

Using C programming language, you will be developing a mutual exclusion algorithm for a process synchronization problem. You need to make sure that your mutual exclusion algorithm ensures that only one process can access the critical section portion of your code at a given point in time. You are allowed to use any mutual exclusion / semaphore related C systems calls.

a) Description of the problem is given below:

Assume that there are a set of n bank accounts ($n \geq 1$) shared by a set of x clients ($x \geq 1$). Clients can perform three different types of transactions with each bank account: deposit, withdraw or transfer funds. If a particular transaction results in a negative account balance, the transaction should be ignored (i.e. an account balance should never be less than 0).

b) Structure of the input file:

In the following example, there are two bank accounts (a1 and a2) shared by a total of ten clients (c1 to c10). The clients are allowed to deposit money into both the accounts, withdraw money from both the accounts, and transfer money between the two accounts. The initial balances of the accounts are specified in the input file. An input file is provided below for illustrative purposes.

```
a1 b 5000
a2 b 3500
c1 d a1 100 w a2 500 t a1 a2 25
c2 w a1 2500 t a1 a2 150
...
...
c9 w a1 1000 w a2 500
c10 d a1 50 d a2 200

Illustration:

(i) a1 b 5000
```

The above line specifies the initial balance of account #1 as $5000

```
(ii) c1 d a1 100 w a2 500 t a1 a2 25
```

The above line specifies the operations performed by client #1. client #1 deposits $100 into Account #1, then withdraws $500 from Account #2, and then transfers $25 from Account #1 to Account #2

The input file name will be provided to the program as a command line argument (similar to Assignment 4). Hardcoding the input file is not allowed. A different input file will be used to evaluate your program for marking purposes where the structure of this input file will remain the same, and only the data will be different.

You must output the balances of each bank account after all the transactions have been performed. For each bank account, your output should display the account followed by the account balance. For example:

```
a1 b 500
a2 b 300
```

Your C program should output results to the screen **and** into a text file "assignment_6_output_file.txt".

c) Makefile:

You are also responsible for creating and submitting a Makefile. If you are not familiar with Makefiles, consult this quick tutorial. You can also refer to the Makefile provided to you in Assignment 4.

- When running **make** it should compile your program and build output to **a6**
- To run your program after compiling: ./a6 [Input filename]

- **make clean** should remove the output file generated and the compiled executable

Sample Input and Output File

A sample input file "assignment_6_input_file.txt" has been provided to you with the expected output in "expected_output.txt".

For testing purposes. we have designed the sample input in a way where the final account balances will always remain the same after completing the transactions. However, when testing your program with another input file, note that due to the non-deterministic nature of threads, the final account balances may vary when running your program multiple times.

Computing Platform for Assignments

You are responsible for ensuring that your program compiles and runs without error on the computing platform mentioned on below. **Marks will be deducted** if your program fails to compile or your program runs into errors on the specified computing platform (see below).

- Students have virtual access to the MC 244 lab, which contains 30 Fedora 28 systems. Linux machines available to you are: **linux01.gaul.csd.uwo.ca** through **linux30.gaul.csd.uwo.ca**.
- It is your responsibility to ensure that your code compiles and runs on the above systems. You can SSH into MC 244 machines.
- If you are off campus, you have to SSH to **compute.gaul.csd.uwo.ca** first (this server is also known as sylvia.gaul.csd.uwo.ca, in honour of Dr. Sylvia Osborn), and then to one of the MC 244 systems **(linux01.gaul.csd.uwo.ca** through **linux30.gaul.csd.uwo.ca**).
- https://wiki.sci.uwo.ca/sts/computer-science/gaul

Assignment Submission

You must submit your Assignment through OWL. Be sure to test your code on one of MC 244 systems (see "Computing Platform for Assignments" section above).

Submission Information:

- Test your program on the computing platform mentioned above. Your program should not fail to compile or run into errors
- **Do NOT** submit a compressed file
- Submit a Makefile that follows the requirements in c) of the Assignment Description above

Marks will be deducted if the above guidelines are not followed.