Assignment #5

Hide Assignment Information

Instructions

Assignment overview

A bank has four accounts and four ATMs distributed across the town. Requests to deposit or withdraw funds to any of the four accounts could occur from any of the four ATMs at any time. For this assignment, write a C program which will accept a command-line parameter indicating a file that contains the starting balance of all four accounts and all the transactions that each of the four ATMs perform in order. The program will then process the transactions from the four ATMs concurrently (one thread per ATM). However, the program must employ mutual exclusion to ensure that no account is overdrawn.

Purpose

The goals of this assignment are the following:

- Learn how to use mutual exclusion when multi-threading to guarantee data consistency
- Get experience with the mutual exclusion system functions (You may use Mutex locks or Semaphores)
- Gain more experience with the C programming language from an OS perspective

Computing platform

You are welcome to develop your program on your own workstation if you wish, but you are responsible for ensuring that your program compiles and runs without error on the Gaul computing platform. Marks will be deducted if your program fails to compile, or your program runs into errors on Gaul.

- Gaul tips

Instructions

You will write a program called `assignment-5.c`. This program will:

- Accept a command-line parameter
    1. The command-line parameter indicates the name of the file to open
    2. Here is a small sample. A larger sample is attached to this assignment. This is not necessarily the input file that will be used for grading. You can assume that the input will have variable length but always follow this format (eg. 3 fields per line. The first 4 lines will have the starting balance for all 4 accounts. The remaining lines will represent all transactions)

        1.
           ```
           ATM,Account,Amount
           -,1,1500
           -,2,2500
           -,3,3500
           -,4,4500
           2,3,1200
           1,1,-500
           2,4,300
           1,2,-100
           4,3,700
           3,1,200
           ```
        2. The file can be interpreted as follows:
            1. Line 1-4: Account 1 begins with $1500. Account 2 begins with $2500, Account 3 begins with $3500, and Account 4 begins with $4500
            2. Line 5+:
                1. Thread 2 executes the following tasks in order: Deposit $1200 into Account 3, deposit $300 into Account 4

        2. Thread 1 executes the following tasks in order: Withdraw $500 from Account 1, withdraw $100 from Account 2
        3. Thread 4 executes the following tasks in order: Deposit $700 into Account 3,
        4. Thread 3 executes the following tasks in order: Deposit $200 into Account 1
- Your program will then
    1. Read the file. You can load the entire file at the beginning or process the file as you go. (If you choose to process the file as you go, you will probably need to ensure each thread opens the file and processes its own transactions and ignores the rest.)
    2. Print the starting balance of all four accounts
    3. Launch four threads
        1. Each thread will process all transactions simultaneously and they must employ **mutual exclusion** to execute its instructions safely. If the lock cannot be obtained, the thread must block until the lock is available.
    4. Print the ending balance of all four accounts

Output

For example, executing `./assignment-5 bank_transactions.csv` against the sample input above should produce the following output (Order may not be sequential) (Final balances may not be consistent across multiple runs) (The sample here is truncated for brevity):

```
Balance
Account 1: $1500
Account 2: $2500
Account 3: $3500
Account 4: $4500

Thread 2: Deposit $1200 into Account 3
Thread 1: Withdraw $500 from Account 1
Thread 2: Deposit $300 into Account 4
Thread 1: Withdraw $100 from Account 2
Thread 4: Deposit $700 into Account 3
Thread 3: Deposit $200 into Account 1
```

```
...
Thread 1: Withdraw $35000 from Account 2 - INSUFFICIENT FUNDS
...

Balance
Account 1: $1200
Account 2: $1800
Account 3: $7500
Account 4: $7100
```

Helpful hints

- To read from a file, you should use the relevant functions in the stdio.h library such as `fopen`, `fscanf`, and `fclose`
- While debugging your program, print out the beginning and the end of your critical sections so you can verify when locks are obtained and released

Submitting

When you are finished your assignment, follow these steps:

1. **Cite your work:** Did you use code that you did not write yourself? Just like an essay, if you did not write it yourself, you must properly attribute the author. Leveraging existing code for your assignment is acceptable but it should represent a minor portion of your submission. Specifically, any code that fulfills the learning outcomes detailed in the Purpose section above should be almost entirely your own. Contact the course instructor if you are unsure.
    1. If you used sample code from the course notes or the textbook, in the comments, state which sample program you used or which page(s) from the lecture notes you used.
    2. If you used code from an online source (eg. StackOverflow, GeeksForGeeks, etc.), in the comments, provide the link where the code was found.
    3. If you used an LLM such as ChatGPT, in the comments, provide the link to the prompts you used to generate the code.
    4. Using code from another student is a violation of the policy on academic integrity and is not permitted.

2. Verify your code runs on Gaul. Create screenshots and place them in the `Assignment-5` directory. Remove any spurious files (eg. the executable version of your program). The directory should have a single file called `assignment-5.c` and any screenshots. Supporting files like `Makefiles` or `READMEs` are acceptable.
3. If necessary, run the following command to get out of the `Assignment-5` directory: `cd ..`
4. Package your assignment into a tarball: `tar -cvf Assignment-5.tar Assignment-5`
5. Verify the contents of your tarball (`tar -tvf Assignment-5.tar`) (`du -sh Assignment-5.tar`). The tarball should have your `assignment-5.c` file and all screenshots. **If your tarball is 10kb in size** you have an empty tarball and you made an error on this step. Make sure you are properly creating your tarball with the right files in it.
6. Use an SFTP program to download the tarball to your local workstation and then upload it to OWL.

Due on Apr 1, 2025 11:55 PM

Available on Mar 19, 2025 12:00 AM. **Access restricted before availability starts.**

Available until Apr 3, 2025 11:55 PM. **Access restricted after availability ends.**

Attachments

bank_transactions.csv (345 Bytes)

Download All Files

Show Rubrics

Submit Assignment

Allowed File Extensions

tar

Files to submit

(0) file(s) to submit

After uploading, you must click Submit to complete the submission.

Add a File Record Audio Record Video

Comments

Submit Cancel