CS570 Summer 2019 Assignment 1

This page last modified 16 May 2019

Design, develop, and test a project that manages multiple threads writing to a single, common, shared file.

Design, develop, and test a program where four “card players” will run, each in their own thread, simultaneously but asynchronously with each other. Each card player shall write the specified text message to one, common shared resource, a file named STACK.txt. When writing to this file, the thread needs to prevent the data from getting corrupted by the other card player’s threads; to implement this, each card player’s thread shall use an appropriate IPC mechanism/algorithm.

1. When your program (process) starts, it shall do the following:
    1. Create a file, named STACK.txt, in the current directory (cwd).
    2. Write it’s pid (Process ID) followed by a Carriage Return and Newline into the file.
    3. Close the file STACK.txt
    4. Create a semaphore named FLAG which the threads will use to manage access to the file STACK.txt.
    5. Create 4 threads. Use the POSIX version of threads (i.e., pthread_create())
    6. Block/wait for all four threads to complete their work.
    7. Once all threads are done, destroy the semaphore, then exit gracefully, printing a friendly message to the console

2. Each thread shall perform the following (note, each thread is running concurrently):
    1. Periodically (see instructions at the end of this specification) get the semaphore FLAG; once the thread has FLAG, it will proceed to do the following:
        1. Open the file STACK.txt and write the defined output (see instructions at the end of this specification) followed by a Carriage Return and Newline into the file.
        2. Write to the console (print to stdout) “Thread <*thread id*> is running” followed by a newline
        3. Close the file STACK.txt
        4. Release the semaphore FLAG
    2. Repeat the above 12 times (each thread will run a total of 13 times).
    3. exit

You will need to use the following POSIX system calls for creating and managing the semaphores with: sem_init(), sem_wait(), sem_post(), and sem_destroy().

Your program will be tested by compiling it and executing it on edoras. Your program shall be written such that it compiles and executes cleanly when using the gcc, or g++ compiler You shall create a sub-directory named "a1" in your home directory. In it, you shall place your source file(s), your header file, your Makefile (see Blackboard for Makefile examples), and a README file (see Blackboard for README requirements). Your source files SHALL CONTAIN sufficient comments for making the source easy to read. Points will be taken off for poorly (or non) commented source. Name the executable "player".

- Create ~/a1 by hand.
- Create source files, an include file, a Makefile, and a README file. Put them into ~/a1.
- The Makefile shall create an executable named "player" in this same directory (~/a1).
- Here is a nice overview of threads [http://www.llnl.gov/computing/tutorials/pthreads/#Overview]

- The system call "system()" will NOT be allowed
- You must work individually or in pairs (individually or a team of 2 students)

The assignment is due 1800 (6:00 PM) on Monday, 10 June 2019

TURNING IN YOUR WORK:
Your project files shall be loaded onto Assignment #1 on Blackboard and in your class account on edoras (if pair programming, choose one student to turn this in).

Make sure that all of your files (all source files, Makefile, README file, test files, etc) are in the a1 sub-directory of the team's selected class account

Additionally, create a tarball (tar file) or zip file and attach the file (upload it) under Assignment Submission in Assignment #1 in Blackboard (only one team member turns in the assignment on Blackboard).

Be sure to state the class account to be used for testing your project in your README file and on Blackboard when you turn it in on Blackboard.

Thread Instructions:

Thread 1: runs once every 1/8 second (125 ms), prints "Diamond" followed by the A | 1 | 2 | 3 |… | 9 | 10 | J | Q | K depending on whether it's the first time or the 13th time it's run

Thread 2: runs once every 1/4 second (250 ms), prints "Club" followed by the A | 1 | 2 | 3 |… | 9 | 10 | J | Q | K depending on whether it's the first time or the 13th time it's run

Thread 3: runs once every 1/2 second (500 ms), prints "Heart" followed by the A | 1 | 2 | 3 |… | 9 | 10 | J | Q | K depending on whether it's the first time or the 13th time it's run

Thread 4: runs once every 3/4 second (750 ms), prints "Spade" followed by the A | 1 | 2 | 3 |… | 9 | 10 | J | Q | K depending on whether it's the first time or the 13th time it's run