CS570 Summer 2017 Assignment 1

This page last modified 23 May 2017

You will do this by developing a project that manages multiple threads writing to a shared file (you aren't building a chat server, the chat bots write to the shared/common file).

You shall implement a program where several chat bots will run, each in their own thread, simultaneously but asynchronously with each other. Each bot shall write the specified text message to one, common shared resource, a file named QUOTE.txt. In order to prevent the data from getting corrupted by other bots, the bots shall use an appropriate IPC mechanism/algorithm.

1. When your program starts, it shall do the following:
    1. Create a file, named QUOTE.txt, in the current directory (cwd).
    2. Have your running process write it's pid (Process ID) followed by a Carriage Return and Newline in to the file.
    3. Close the file QUOTE.txt
    4. Create a semaphore named FLAG which the threads will use to manage access to the file QUOTE.txt.
    5. Create 7 threads. Use the POSIX version of threads (i.e., pthread_create())
    6. Block/wait for all seven 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 (even numbered threads - once every two seconds, odd numbered threads – once every 3 seconds) get the semaphore FLAG; once the thread has FLAG, it will proceed to do the following:
        1. Open the file QUOTE.txt and write the thread's tid (thread id) followed by "The Quote" (followed by a Carriage Return and Newline)
        2. Write to the console (print to stdout) "Thread <*thread id*> is running" followed by a newline
        3. Close the file QUOTE.txt
        4. Release the semaphore FLAG
    2. Repeat the above 7 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().

I will test your program 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 examples on Makefiles), 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 "bots".

- 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 "bots" 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 in pairs (team of 2) unless approved by the instructor.

The assignment is due 1800 (6:00 PM) on Monday, 5 June 2015

TURNING IN YOUR WORK:
Your project files shall be loaded onto Assignment #1 on Blackboard and onto the class account of one of the team members.

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

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

Be sure to state which class account is available for testing in the turnin on Blackboard and in the README file.

The Quote:

Even numbered threads: "Controlling complexity is the essence of computer programming.
--Brian Kernigan

Odd numbered threads: "Computer science is no more about computers than astronomy is about telescopes."
--Edsger Dijkstra