Programming Assignment #1

Due Jun 11 by 8am **Points** 50 **Submitting** a file upload
Available after May 24 at 8am

CS570 Summer 2021 Assignment 1

This page last modified 8 June 2021

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.

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 each will simply write to a shared/common file).

You must work in pairs (team of 2) unless approved by the instructor.

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 into 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

1. 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:
2. Open the file QUOTE.txt and write the thread's tid (thread id) followed by "The Quote" (followed by a Carriage Return and Newline)
3. Write to the console (print to stdout) "Thread *<thread id>* is running" followed by a newline
4. Close the file QUOTE.txt
5. Release the semaphore FLAG

2. Repeat the above 7 times (they run a total of 8 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 Canvas for examples on Makefiles), and a README file (see Canvas 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 [https://computing.llnl.gov/tutorials/pthreads/]
- The system call "system()" will NOT be allowed

The assignment is due 1700 (5:00 PM) on Monday, 14 June 2021

TURNING IN YOUR WORK:

Your project files shall be loaded onto Assignment #1 on Canvas, in the class account of one of the team members. Be sure to write each

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

Before loading files onto Canvas, create a single zip file or a tarball (tar file) with all of your project files. Then, Attach File (upload it) under Assignment Submission in Assignment #1 (only one team member turns in the assignment on Canvas). Next, Attach the README file. Before submitting your project, include the names and class accounts of both team members and identify which account to be used for testing, then submit your project.

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

Sample QUOTE file:

```
Process ID: 32548
Thread ID 2: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 1: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 5: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 6: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 7: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 3: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 4: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 2: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 6: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 4: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 1: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 5: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 7: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 3: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 2: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 6: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 4: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 1: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 5: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 6: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 4: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 2: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 7: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 3: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 6: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 4: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 2: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 1: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 5: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 7: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 3: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 6: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 4: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 2: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 1: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 7: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 3: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 5: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 4: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 6: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 2: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 4: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 6: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 2: "Controlling complexity is the essence of computer programming." --Brian Kernigan
Thread ID 1: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 3: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 7: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 5: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 1: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 7: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 3: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 5: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 1: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 7: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 3: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
Thread ID 5: "Computer science is no more about computers than astronomy is about telescopes." -Edsger Dijkstra
```

Sample terminal output:

```
bots
Creating thread, in main(): 1
Creating thread, in main(): 2
Creating thread, in main(): 3
Thread 2 is running
Creating thread, in main(): 4
Creating thread, in main(): 5
Thread 1 is running
Creating thread, in main(): 6
Thread 5 is running
Creating thread, in main(): 7
Thread 6 is running
Thread 7 is running
Thread 3 is running
Thread 4 is running
Thread 2 is running
Thread 6 is running
Thread 4 is running
Thread 1 is running
Thread 5 is running
Thread 7 is running
Thread 3 is running
Thread 2 is running
Thread 6 is running
Thread 4 is running
Thread 1 is running
Thread 5 is running
Thread 6 is running
Thread 4 is running
Thread 2 is running
Thread 7 is running
Thread 3 is running
Thread 6 is running
Thread 4 is running
Thread 2 is running
Thread 1 is running
Thread 5 is running
Thread 7 is running
Thread 3 is running
Thread 6 is running
Thread 4 is running
Thread 2 is running
Thread 1 is running
Thread 7 is running
Thread 3 is running
Thread 5 is running
Thread 4 is running
Thread 6 is running
Thread 2 is running
Thread 4 is running
Thread 6 is running
```

Assignment #1 - Asynchronous Threads

Criteria	Ratings		Pts
README File	**5 pts** **Full Points** README file includes content as specified in the README format doc	**0 pts** **Minimum Points** README exists but contains minimal info	5 pts
Makefile	**5 pts** **Full Points** Makefile correctly compiles the project using at least two rules: (1) make the project and (2) clean up files	**0 pts** **Minimum Points** Minimal or no Makefile	5 pts
Threads are created correctly	**6 pts** **Full Points** Used pthread_create and managed each threads exit	**0 pts** **Minimum Points** did not use or used incorrectly	6 pts
All threads run correctly	**9 pts** **Full Points** All threads run the correct number of times and in the correct order using the synchronization tools	**0 pts** **Minimum Points** Can create threads but they are totally ad hoc	9 pts
Threads Use Semaphore	**12 pts** **Full Points** Semaphore is properly created, destroyed, and used by threads, including the use of: sem_init(), sem_wait(), sem_post(), and sem_destroy()	**0 pts** **Minimum Points** did not use or used incorrectly	12 pts
Process Block/Waits	**5 pts** **Full Points** Process properly block waits for threads to complete	**0 pts** **Minimum Points** Process does not manage threads	5 pts
Process Output and Control	**8 pts** **Full Points** Process writes its PID correctly into the shared file. Process performs a clean exit	**0 pts** **Minimum Points** Process fails to write its PID correctly into the shared file. Process fails to exit as specified in the requirements	8 pts
			Total Points: 50

Submission

✓ Submitted!
Jun 10 at 10:46pm
Submission Details
Download a1.zip
Download README.txt

Grade: 50 (50 pts possible)
Graded Anonymously: no
View Rubric Evaluation

Comments:
cssc4206
Dong Lin, Jun 10 at 10:46pm
Excellent, well done!
Guy Leonard, Jun 21 at 12:56pm