**Programming Project #2**
**CS5352 Advanced Operating Systems Design**

**Due Date: 3/16, 11:59 p.m., via Blackboard.**
**Late submissions are accepted till 3/23, 11:59 p.m., with 10% penalty each day.**

## IPC (InterProcess Communication) Programming (in C programming language)

In this programming project, you are asked to develop a multithreaded program and a Makefile to automate the compilation on Linux platform. This assignment assists you for better understanding of processes and threads management, multithreaded programming, and enhancing programming skills and experience with programming on a Unix-like environment.

You are asked to implement a multithreaded producer-consumer problem with PThreads library in this programming assignment. The producer-consumer is a common problem that requires cooperating processes or threads. In this problem, a producer produces items and put into a shared buffer, then these items are consumed by consumers. Synchronization and mutual exclusion are required to solve the problem correctly as discussed in the lecture and described in the textbook (please see section 2.3 and corresponding lectures). Figure 2-32 in the textbook shows a solution with one producer and one consumer. In this part of assignment, you are asked to implement this problem with support of multiple producers and consumers threads with PThreads library on a Linux platform.

Requirements:

1. Ensure appropriate synchronization and mutual exclusion using PThreads mutex locks and conditional variables as discussed in the class and shown in Figure 2-32.
2. Support multiple producer threads and multiple consumer threads. The number of producer threads and consumer threads can be either hard coded in the program or provided through command line arguments.
3. Print out the producer/consumer actions, e.g., "producer *i* produced one item", "consumer *j* consumed one item", "producer *i* found the buffer is full and waiting for a consumer to consume", and "consumer *j* found the buffer is empty and waiting for a producer to produce".
4. You can fix the shared buffer with 8 items, and produce 64 items from each producer thread. Test with 1 producer thread and 1 consumer thread, 2 producer threads and 2 consumer threads, and 4 producer and 4 consumer threads.
5. Develop a Makefile to automate the compilation process.

Hints:

- Implement a producer routine and a consumer routine, in which, a mutex lock and a conditional variable used to synchronize accessing the shared buffer. Figure 2-32 in the textbook shows a solution with one producer and one consumer.
- In the main function, create multiple producer and consumer threads separately, and call for producer/consumer routine. Don't forget to join threads and deallocate data structures such as mutex locks and conditional variables created as shown in Figure 2-32.

- Use Reference Materials regarding Pthreads listed at the end of this project description to help with Pthreads APIs (all APIs you need are already shown in Figure 2-32).

**Expected Submission:**

You should submit a single tarball/zipped file through the Blackboard containing the following, and please name your submission file starting with LastName_FirstName_Project#2.

- Source codes including the Makefile.
- Output files for the result of test cases.

How to report your results

- You can simply use bash redirection ('>') to produce log files like what we showed in class, and then submit the log files (plain text files)

How to submit your codes and results

- First, on Oak machine, you can create a tar ball to include all your source codes, logs, and other files, like one of below commands
  - `tar cvf mycs4352project.tar *`
  - `tar cvfz mycs4352project.tgz *`
- Then you can use "scp" or any SFTP client, e.g. FileZilla to download the tar/zipped file to your local machine, after that you can submit to Blackboard just like you submit your other homework solutions
  - E.g. you can download "pscp.exe" from the Putty website, then run a command like below to copy/download "mycs4352.tar" file to your local machine (you'll need to replace the path name "home/TTU/yonchen/cs4352/public_gitlab_repo/mycs4352.tar" with your own file name
  - `pscp.exe yonchen@oak.cs.ttu.edu:~/cs4352/public_gitlab_repo/mycs4352.tar .`
  - `pscp.exe yonchen@oak.cs.ttu.edu:/home/TTU/yonchen/cs4352/public_gitlab_repo/mycs4352.tar .`

**Grading Criteria:**

**Please note that, if needed, we may request an in-person, 5-10 mins quick demo from you.**

| Percentage % | Criteria |
|---|---|
| 10% | Inline comments to briefly describe your code |
| 30% | Correct use of system calls/library procedures of creating threads, correct use of primitives for synchronization and mutual exclusion, e.g. mutex locks, conditional variables |
| 20% | Correctness of result. Source code can be compiled and executed. |
| 20% | Successfully carry out specified test cases |

| | |
|---|---|
| 20% | Correctness and features of Makefile (at least automate compilation and cleanup) |

**Source Code Samples**

Please checkout source code samples with executing the following command on the Oak machine:

```
git clone https://discl.cs.ttu.edu/gitlab/yongchen/cs4352.git
```

**Reference Materials:**

- Linux system programming:

  Book: *Linux System Programming*

  Online:
  Tutorial for Beginners, http://www.ee.surrey.ac.uk/Teaching/Unix/
  Advanced Linux Programming, http://www.advancedlinuxprogramming.com/alp-folder/advanced-linux-programming.pdf

  Stack Overflow, http://stackoverflow.com/
  www.unix.com
  The Geek Stuff, http://www.thegeekstuff.com/

- PThreads**:**
  - POSIX Threads Programming, https://computing.llnl.gov/tutorials/pthreads/
  - Linux Tutorial: POSIX Threads, http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
  - Complete Pthreads API: http://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread.h.html

- Makefile**:**
  - http://www.gnu.org/software/make/manual/make.pdf or
  - http://www.gnu.org/software/make/manual/html_node/index.html