CSE 4300 Programming Assignment 2
Assigned: Sep 15, 2016
Due: Oct 7, 2016, 11:59 pm (Midnight)

The purpose of this assignment is to introduce you to the following concepts:

1. C programming in Linux environment
2. Writing Multithreading programs
3. Use of synchronization primitives

Suggested Reading for this assignment (Please note that you can find many other great tutorials by simply searching on Google!).

https://computing.llnl.gov/tutorials/pthreads/
(Please read this! You can also find sample code here!)

You may find the followings useful for part of this project –

- pthread_mutex_t *mutuex;
- pthread_cond_t *condition;
- pthread_create ();
- queue data structure (This is not part of pthread!)

Part A (10 points)

Purpose: Writing and compiling a C program in Linux

Example:

Let us save this as hello.c

```
#include <stdio.h>

main()
{
printf("Hello, world!\n");
return 0;
}
```

To compile, type the following in command prompt:

```
gcc -o hello hello.c
```

The *-o* option informs the compiler of the desired name for the *executable* (i.e., runnable) file that it produces. The name used in this example is *hello*, but it could just as easily be anything else such as *hello.exe*.

To run, type the following-

```
./hello
```

Now as you know how to write and compile a C program, write a **C program** that can do the followings when you execute your program –

1. Change the current directory to another directory using chdir() command
2. Create a new directory under the new directory
3. List everything in the current directory

Part B

Given the proliferation of multicore architecture, learning multithreaded programming is of utmost importance these days. Hence, the goal of this part of the assignment is to learn multithreaded programming. We will also learn how to synchronize among multiple threads using mutex.

Please note that, to compile multithreaded program, you need to specify "-lpthread" as follows

```
gcc -lpthread -o output.txt program.c
```

Part B.1: (100 points)

The goal of this assignment is to write a multithreaded program that explores synchronization challenge. For this part, assume that, we have a shared variable last_threadID. This is initialized to 1 at the beginning. Now create 5 threads in your program and assign ID 1,2,3,4,5 to them respectively. You can pass the ID as a parameter when you create the threads.

Each of the threads will try to access the variable "last_threadID". Whenever a thread can acquire the variable, it will subtract the value of "last_threadID" from its own ID and output "MyTurn" if the difference is 0. If the difference is not 0, it will output "Not My Turn". In both cases, it will also output its thread ID. After that, it will set "last_threadID" equal to its own Id + 1. When thread 5 gets to it, it will reset the "last_threadID" to 1 again, and the cycle continues.

For example, if thread 1 gets the access first, it will print "My Turn: 1.
In contrast, if thread 3 gets the access first, it will print "Not My Turn: 3.

The goal is to ensure that each thread access the "last_threadID" variable in sequential order. Meaning, thread 1 will get the first chance, then thread 2, and so on. If a thread acquires "last_thredID" and the difference (Id- last_threadID) is greater than 0, this implies that it is not its turn yet. So it will release the variable and try again later.

You need to run the program until each thread prints at least 10 times. Once a thread prints for 10 times, it terminates.

Part B.2 (100 points)

In this part, you are going to implement the producer-consumer problem using pthreads. After implementing, let the program run until at least 1000 items are produced by the producer and at least 1000 items are consumed by the consumer.

You are going to create 2 producer threads and 1 consumer threads in the program.

Repeat the experiment for the following settings:

1. Make the queue size 15
2. Make the queue size 30
3. Make the queue size 50

For each setting, count the number of times the producer and consumer goes to sleep because of empty queue or full queue. Print the total number of times each goes to sleep at the end of the experiment.

What to submit

For each part, please submit a well-documented listing of your source code and output through HuslyCT. For Part A, you do not have to submit any output.
You can copy paste your code in word file if you want.