EECS 111

System Software

Spring 2017

Project 2: Thread & Synchronization

Due Date (May 12th, 2017 11:59 PM)

Instructions

In this project, your job is to understand thread synchronization and mutex lock and write a code to solve some problem. You can unzip the project file by using this command: `tar -xvf ./eecs111-proj2.tar.gz`. Then you will see the following files in the ***p2_student*** directory:

1. `main.cpp`: This is a top file for this project
2. `p2_threads.h`: This is a header file to handle threads
3. `p2_threads.cpp`: This is a file to handle threads
4. `utils.h`: This is a header file for utility functions
5. `utils.cpp`: This is a file header file for utility functions
6. `types_p2.h`: This is a header file for `Person class`
7. `types_p2.cpp`: This is a source code file for `Person class`
8. `Makefile`: This is a compilation script

You need to complete the following files: `main.cpp, p2_process.cpp,` and `Makefile`. The main object of this project is to create a program that solve the problem below.

After you complete the `Makefile` file, You should be able to build the project code by using following command: `make`. After build, **you must have `p2_exec` file as an executable.** Also, it is suggested to remove all the compiled object files and executable file with the following command: `make clean`. You can change the source code in the given files. In addition, you can add any new header and source code files for this assignment. But, please make sure that your program can be compiled with '`make`' command and satisfies the descriptions below.

In this project, you have to get 1 argument. You have to print error message and usage example if you didn't provide the argument. This is an example for the error message.

```
[ERROR] Expecting 1 argument, but got (X).
[USAGE] p2_exec <number>
```

Here is the description for the problem:

Suppose that a university wants to show off how progressive it is and ends its long standing practice of gender-segregated restrooms on campus. However, as a concession to propriety, it makes a policy that when a woman is in the restroom only other women may enter, but not men, and vice versa. On the door of every restrooms there will be a sign with a sliding marker that will indicate one of three possible states it is currently in:

`Empty`, `WomenPresent`, and `MenPresent`

For this assignment, you will need to complete a working program that will address the above problem and that must compile and run. The program must execute until all the people finish using the restroom.

- The program must contain the following functions: `woman_wants_to_enter`, `man_wants_to_enter`, `woman_leaves`, and `man_leaves`.

- The program must display the following during its execution: the time stamp, the state of the restroom (empty, occupied by women and if so how many, occupied by men and if so how many), the status of the queue including whether it is empty or not and, if not empty what genders are in queue. The example command line output for each case is as follows:

```
➢ [XX ms][Restroom] State: Empty, Queue status: Total: x (Men: x, 
  Women: x)
➢ [XX ms][Restroom] State: WomenPresent (Number: x), Queue status: 
  Total: x (Men: x, Women: x)
➢ [XX ms][Restroom] State: MenPresent (Number: x), Queue status: Total: 
  x (Men: x, Women: x)
```

- The above example output is only for the restroom status. There may be many additional command line outputs.

- During run-time, you need to generate a sequence of people based on the input argument. If 10 is the input argument, then your program must generate a sequence of total of 20 people that includes 10 men and 10 women.

- You must randomly assign the gender to a person and send that person to the restroom. After that person has been sent to the restroom, you must wait for a random time interval (between 1 milliseconds – 5 milliseconds) until you can send the next person to the restroom. Your program must display the following output:

```
➢ [XX ms][Input] A person goes into the queue
```

- When a person goes into the restroom, you must randomly assign the time to stay in the restroom (3 milliseconds – 10 milliseconds). Your program must display the following output:

```
➢ [XX ms][Queue] A man goes into the restroom (Stay xx ms).
➢ [XX ms][Queue] A woman goes into the restroom (Stay xx ms).
```

- Your program must generate different sequence. In other words, your program must show different behaviors for each run of the program.

- Your program should not have a starvation problem.

- TIP: You must have at least two threads. Except main process.

Grading

Your project code will be automatically graded. There are two reasons for this:

1. A grader program can test your code a lot more thoroughly than a TA can, yielding more fair results.
2. An autograder can test your code a lot faster than a TA can.

Of course, there is a downside. Everything that will be tested needs to have a standard interface that the grader can use, leaving slightly less room for you to be creative. Your code must strictly follow these interfaces.
Since your submissions will be processed by a program, there are some very important things you must do, as well as things you must not do:

1. Your executable filename must be `p2_exec`.
2. You must store all your files in **p2_<your student id>** directory (i.e. p2_84733922).
3. You must not change the directory structure. In other words, please don't change the filename except the top directory. The top directory name (student id) will be used to track your submission status.

Submission

A drop box folder is provided in EEE website. You need to compress all the files in folder into a single archive file (zip, rar, etc.) and upload it. The deadline for uploading the files is the project deadline.

Task

1. (5%) Following the submission format
2. (10%) Compile your code with your Makefile without any problem.
3. (10%) Command line output matches with the description for any type of input.
4. (15%) Randomly generate the input sequence and randomly assign the time to stay in the restroom
5. (40%) Your program work properly and does not have deadlock.
6. (20%) Your program does not have starvation problem.

Note

Even though you can generate correct output, that does not mean that your code consider some extreme case. Please verify your code with some corner case.

There will be more discussion in the discussion session. Please don't forget to come to the discussion session.

If you have any question regarding the project, please send an e-mail to following address: haeseunl@uci.edu. You MUST cc the professor and TA when you send an e-mail.