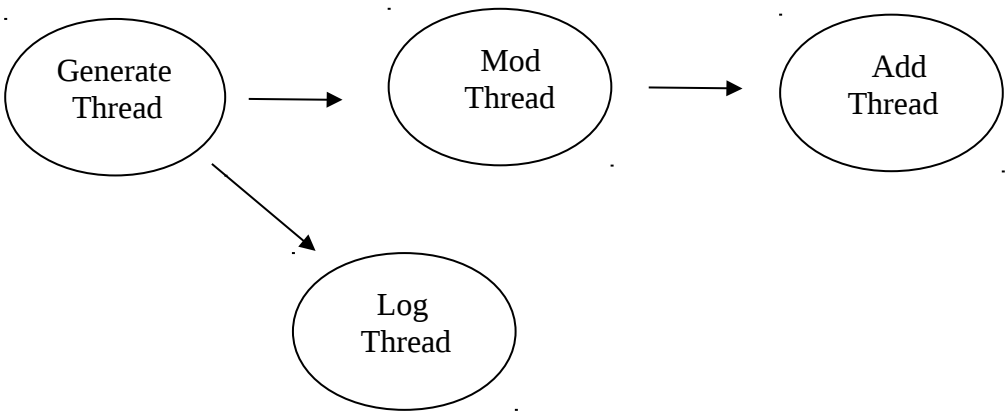
CSE 333 - OPERATING SYSTEMS
Programming Assignment # 3 DUE DATE: 27/12/2018 - 23:59

In this assignment, you will write a program that uses threads and synchronization to generate a square matrix with randomly created numbers. You will implement a program with four types of threads, structured like:

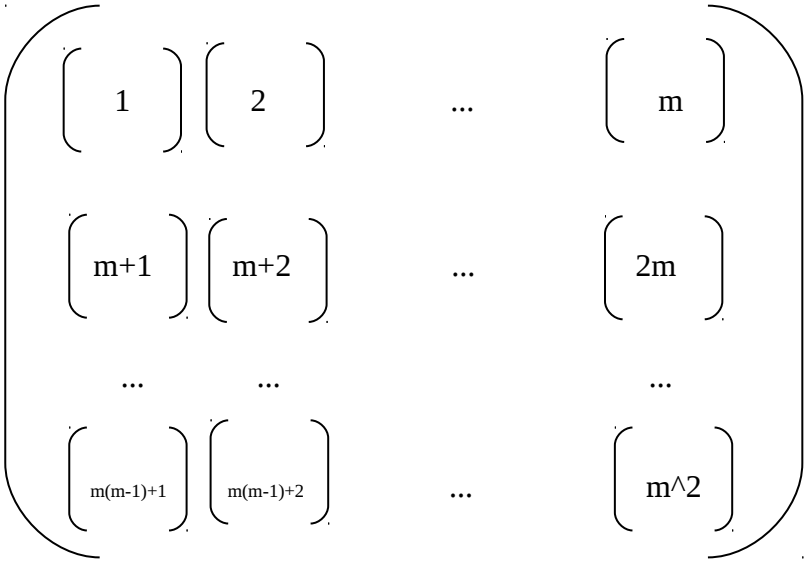


- The Generate Thread will generate a matrix of size 5x5 and fill it with random numbers.
- The Log Thread will get the 5x5 matrix generated by Generate Thread and allocate a bigger matrix to put multiple smaller matrices inside.
- The Mod Thread will get the 5x5 matrix generated by Generate Thread and find modula of each number by the first number in the matrix (the number at the zeroth row and zeroth column).
- The Add Thread will get the output of Mod Thread and find a local sum of 5x5 matrix and update a global sum.

Program Details

- The user will execute the program specifying the size of the matrix (N – a multiply of 5) to be generated and the number of threads to be created of each kind.
- You will then create the specified number of Generate Threads. Generate Threads will create a 5x5 matrix and fill it with random values (you can think the upper limit is 100 for random values), and put it inside a global queue. These submatrices will be used to create the bigger matrix whose size is specified by the user at program execution. So the Generate Threads has to create (N/5)*(N/5) submatrices. If you have a smaller number of threads, then each thread has to create another

submatrix until the specified number is achieved. The order in which the submatrices will be generated is as follows:



Note that the Generate Thread to be executed has to find which submatrix to generate before generation. So each Generate Thread has to check the last submatrix assigned and then generate the one after that. You have to keep track of and store this information with the submatrices created.

- The Log Threads will read from the global queue that is being updated by the Generate Threads. Then the first Log Thread to run will allocate a matrix of size NxN array and all of them fill the matrix with the numbers from submatrices in the appropriate slots.
- The Mod Threads will also read from the global queue that is being updated by the Generate Threads. Each Mod Thread will replace the values inside submatrices with the remainder of when they are divided by the first number in the submatrix (the number at the zeroth row and zeroth column). Note that the first numbers in all of the submatrices will be zeros after this operation. The updated version of the submatrices will be put inside another global queue.
- Add Threads will read from the global queue that is being updated by Mod Threads. They will find the local sum for each matrix and update a global sum value.
- The main thread will wait for all the threads to finish execution and print the matrix of size NxN (produces by the Log Threads) and the global sum on an output file.
- All the threads have to print information about their job on the screen (see the example execution below).

- Note that when a Generate Thread starts generating a submatrix, it has to get which submatrix to generate. You have to make sure that no multiple Generate Threads create same submatrix.
- Note that there are two global queues requested and the queues can be accessed by many threads of different types so a synchronization should be done seperately for each queue. This can be achieved by using the semophores.
- Note that multiple Log Threads have to be executed concurrently.
- Note that when Log Threads and Mod Threads can access the global queue at the same time since they are not updating the value.
- The main thread is responsible for creating threads and waiting for the completion of them.
- Both global queues have to be created at the very first time a thread needs to use it by that thread.
- Preserving consistency and preventing deadlocks are major issues to be considered.
- Multiple simultaneous operations on different parts of the same matrix and queue should be allowed in your solution.
- Your program will be executed as follows:
    - Example:

      ./project3.out -d 30 -n 15 5 8 6
    - The -d option represents the size of the matrix to be created (a multiple of 5), -n option represents the numbers of threads to be created of each type (Generate, Log, Mod, Add, respectively).
- Your program should produce output.
    - Example:

      <Thread-type and ID> <Output>

      Generator_1 “Generator_1 generated following matrix: [3,2,3,4,5,
      6,7,8,9,10,
      1,2,3,4,5,
      6,7,8,9,10,
      1,2,3,4,5]
      This matrix is [0,1] submatrix” // Labeled 2 in above shape

      Generator_2 “Generator_2 generated following matrix: [2,2,3,4,5,
      10,9,8,7,6,

5,4,3,2,1,
6,7,8,9,10,
1,2,3,4,5]
This matrix is [0,0] submatrix" // Labeled 1 in above shape

Mod_2 "Mod_2 generated following matrix: [0,0,1,0,1,
0,1,0,1,0,
1,0,1,0,1,
0,1,0,1,0,
1,0,1,0,1]
This matrix is generated by [0,0] submatrix"

Mod_1 "Mod_1 generated following matrix: [0,2,0,1,2,
0,1,2,0,1,
1,2,0,1,2,
0,1,2,0,1,
1,2,0,1,2]
This matrix is generated by [0,1] submatrix"

Add_1 "Add_1 has local sum: 25 by [0,1] submatrix, global sum before/after update: 0/25"

Add_2 "Add_2 has local sum: 12 by [0,0] submatrix, global sum before/after update: 37"

....

- The output file should be in the following format:

The matrix is

[2,2,3,4,5,3,2,3,4,5, ...
10,9,8,7,6,6,7,8,9,10, ...
5,4,3,2,1,1,2,3,4,5, ...
6,7,8,9,10,6,7,8,9,10, ...

1,2,3,4,5 1,2,3,4,5 .....

.....

.....]

The global sum is : 65482.

Notes:

- The project will be done in Linux operating system using C programming language.
- You must use PThread library and synchronization appropriately in your code.
- Take into account materials and examples covered in the lab sessions.
- Consider all necessary error checking for the programs.
- No late homework will be accepted!
- In case of any form of copying and cheating on solutions, all parties/groups will get ZERO grade. You should submit your own work.
- You have to work in groups of two.
- If after 3 projects, your overall grade is higher than 100, it will be considered as 100.

What to submit?

A softcopy of your source codes which are **EXTENSIVELY** commented and appropriately structured and a project report (minimum 2-page) that contains the detailed information about your implementation should be emailed to cse333.projects@gmail.com. All the files should be submitted as one zip file. You should use your student numbers as the name of the file: student#1_student#2_project3.zip