CMSC 216 Threads Exercise Summer 2022

Threads **Deadline: Tue, Aug 2, 11:55 pm**

1 Overview

In this exercise you will use threads to write a program that computes the maximum and the sum of data present in an array. For this assignment:

- There is only one deadline (no late submission).
- There are no public, release or secret tests.
- There is no project distribution (usual folder we post).
- For this exercise, you should work in a group (of at most four students). Although you could work by yourself, we prefer groups (it reduces the amount of grading we have to do). Select group members during lecture, lab, or use Piazza.
- This is an open exercise, and unlike projects, you can develop your code along with other students and discuss the exercise with others.
- You may use any code we have posted for our lectures/labs.
- If you have questions about this assignment, please post them in Piazza. This will simplify the process of keeping the class informed about updates. It is your responsibility to check Piazza often regarding any possible updates and clarifications.

2 Submission

- Create a file named authors.txt with the names and directory ids (e.g., terp1, NOT your student number) of the members in your group.
- Create a folder with the .c code, the report and the authors.txt file. Zip that folder and upload it to the submit server using the submit server web interface (you will not be able to submit using the submit command).

  PLEASE ONLY ONE SUBMISSION PER GROUP. If several members of your group submits the work, all group members will be penalized as we will have to do unnecessary grading.
- Remember, THERE IS NO LATE DEADLINE for this exercise.

3 Academic integrity statement

Please **carefully read** the academic honesty section of the course syllabus. We take academic integrity matters seriously. Please do not post assignment solutions online (e.g., Chegg, github) where others can see your work. Posting code online can lead to an academic case where you will be reported to the Office of Student Conduct.

This exercise has been used in the past and you may find implementations online. Notice we are aware of the code sources, so you will be part of an academic integrity case if you use any such sources (even if you modify them). If you violate academic integrity rules, we may ask for an XF in the course; no exceptions.

4 Specifications

- You need to write a program that will create an array with random integer values. Then one of two possible processings will be applied to this array:

  - Max - You need to compute the maximum value in the array. Notice that for a number of threads greater than one, you need to assign to each thread an array segment. Just divide the array in equal length segments (when possible) based on the number of threads. For example, for an array of 1001 elements and 2 threads, the first thread can process the first half (500) and the second thread the rest (501).
  - Sum - You need to correctly compute the sum of elements in the array based on the following formula: sum = (sum + a[i]) % 1000000, where a[i] represents an array element.

- The program relies on command line arguments to provide the following information:
  - Number of elements
  - Number of threads
  - Seed (for random number generator)
  - Task (1 for max, 2 for sum)
  - Print Results (Y or N)
- Feel free to provide any additional parameters you understand are needed.
- After completing a task (maximum or sum) the program will display the results (maximum value or sum) if requested by the user.
- After completing a task, the program will display the wall clock time, the user time and the system/kernel time that it took to compute the maximum or the sum. Notice that this time information does not include the time to generate the random array.
- You need to have threads that run concurrently. For example, starting a thread and executing a join on it and repeating the process for each thread is not valid (you are executing threads sequentially.)
- The time information should follow the results.
- Report - Run experiments where you explore the impact of threads while computing the maximum value and the sum. Create a table where you increase the number of threads and the data sets. Your goal is to see whether using additional threads improves performance. Provide a short (one paragraph) explanation of your results. The information in this report is what you will be discussing with your TA in lab. You can create this report in a text file (report.txt). You will not be penalize if you don't see any improvement while using threads. Just try to explain why.
- Remember that you may not use code from the internet.
- The current limits in grace will not allow you to create a large number of threads. You can change those limits (assuming you are using tcsh) by executing "limit maxproc" followed by a number (e.g., limit maxproc 400).
- Make sure you check the values returned by Pthreads functions; this will allow you to identify whether your code is failing.
- An example of using the srand() and rand() functions can be found in the lecture example (dyn_realloc.c) discussed while covering dynamic memory allocation.

5 Grading Criteria

Your project grade will be determined by the following:

Maximum using multiple threads	40 pts
Sum using multiple threads	45 pts
Report	15 pts

5.1 Style grading

We will not grade your code for style, but you should follow the usual C style guidelines available at:

http://www.cs.umd.edu/~nelson/classes/resources/cstyleguide/