Jordan University of Science and Technology

Computer Engineering Department

CPE 473: Operating Systems

Programming Assignment #2: Threading

Notes:

1- **The due date is 2/1/2022 at 11:55 PM. The early deadline is 26/12/2021 at 11:55 PM**
2- **Late submissions (i.e. after the due date) will not be accepted.**
3- **Submissions before the early deadline will be rewarded a *20% bonus***
4- **Submit the source code files and the PDF report. Write your name and ID at the top**
5- **Groups are allowed (up to 3 students, it is OK to be from different sections)**
6- **Each group member should submit the files to eLearning individually**

In this assignment, you will implement a multi-threaded program (using C/C++) that will perform the **Matrix Multiplication Problem (cross product)**. The program will create T worker threads to perform the operation (T will be passed to the program with the Linux command line). Each of the threads needs to perform part of the total matrix multiplication operation.

Your program should have 3 global shared variables: numOfOdd, numOfEven, and totalCells. These will be incremented by all the threads when a thread finds a cell in the output matrix. numOfOdd and numOfEven will be incremented when the output cell is odd or even, respectively. The variable "*totalCells*" will be incremented for each cell in the output matrix regardless of being odd/even.

When any of the threads starts executing, it will print its number (0 to T-1), and then the range of loops that it is operating on from the total matrix multiplication problem. When all threads are done, the main thread will print the final values in the counter variables: numOfOdd, numOfEven, and totalCells. Do not print the output matrix itself. Note that the main needs to wait until all worker threads are done.

You should write two versions of the program: The first one doesn't consider race conditions, and the other one is thread-safe. The input will be provided in an input file (in.txt), and the output should be printed to an output file (out.txt). The number of worker threads will be passed through the command line, as mentioned earlier.

The input will start with an integer number (N) representing the dimension of the input matrices (assume both matrices are square, with the same dimensions). This will be Followed by two NxN matrices representing the two input matrices A, and B.

Tasks:

In this assignment, **you will submit your source code files for the thread-safe and thread-unsafe versions, in addition to a report (PDF file).** The report should show the following:

1. Screenshot of the main code
2. Screenshot of the thread function(s)
3. Screenshot highlighting the parts of the code that were added to make the code thread-safe, with explanations on the need for them
4. Screenshot of the output of the two versions of your code (thread-safe vs. non-thread-safe), when running passing the following number of threads (T): 1, 2, 4, 8, 16, 32.
5. Based on your code, how many computing units (e.g. cores, hyper-threads) does your machine have? Provide screenshots of how you arrived at this conclusion, and a screenshot of the actual properties of your machine to validate your conclusion. It is OK if your conclusion doesn't match the actual properties, as long as your conclusion is reasonable.

Hints:

1. **Read this document carefully multiple times to make sure you understand it well. Do you still have more questions? Ask me during my office hours, I'll be happy to help!**
2. To learn more about matrix multiplication, look at resources over the internet (e.g. link). We only need the parts related to the simple case: Cross product on two square matrices: **AxB=C**
3. Plan well before coding. Especially on how to divide the multiplication over worker threads. Think of which matrices/variables are read-only, write-only, or read/write?
4. For passing the number of threads (T) to the code, you will need to use *argc*, and *argv* as parameters for the main function. For example, the Linux command for running your code with two worker threads (i.e. T=2) will be something like: "./a.out 2"
5. You can assume that the number of threads (T) will always be a power of 2 (e.g. 1, 2, 4, 8, 16, …)
6. For answering Task #5 regarding the number of computing units (e.g. cores, hyper-threads) in your machine, search about "diminishing returns". You also might need to use the Linux command "*time*" while answering Task #4, and use input matrices of large sizes (e.g. N>=1024).
7. I suggest you create another code to automate generating input matrices of bigger sizes. Don't know how? Ask me during my office hours. Moreover, I have already provided a sample input file with N=1024 to help you get started.
8. You will, obviously, need to use pthread library and Linux. I suggest you use the threads coding slides to help you with the syntax of the pthread library

Sample Input:

```
4
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2

3 3 3 3
3 3 3 3
3 3 3 3
3 3 3 3
```

Sample Output:

```
ThreadID=0, startLoop=0, endLoop=2
ThreadID=1, startLoop=2, endLoop=4
numOfEven=16, numOfOdd=0, totalCells=16
```

Output matrix (Don’t print it with the program’s output):

```
24 24 24 24
24 24 24 24
24 24 24 24
24 24 24 24
```