Programming Assignment #4

This assignment will give you some practice working with the POSIX threads. Your task is to simply square an MxM matrix. For example, squaring matrix A will result in the following:

$$A=\begin{bmatrix} -4 & -1 & 2 & 1 \\ 6 & 0 & -9 & 10 \\ -3 & -10 & 6 & 0 \\ -10 & 6 & 7 & 9 \end{bmatrix} \qquad A^2=\begin{bmatrix} -6 & -10 & 20 & -5 \\ -97 & 144 & 28 & 96 \\ -66 & -57 & 120 & -103 \\ -35 & -6 & 31 & 131 \end{bmatrix}$$

Another example:

$$A=\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad A^2=\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

The interesting part of this assignment is that each value in the resulting matrix will be calculated in a separate thread. This means that if you have a 3x3 matrix, you will create 9 threads that will execute concurrently, one thread for each of the 9 values. The main thread will create all of the other threads. Each thread will perform the actual arithmetic of multiplying one row of the matrix by one column of the matrix. (This assignment is almost identical to the previous assignment that uses processes instead of threads.)

You will not be using any global data in the program. All of the necessary data for the threads will be passed to them via the parameters argument of the `pthread_create` function. You'll need to create a structure that contains all of the information that a thread will need, e.g. a pointer to the row and column, width, etc. and put this in a header file `matrix-thread.h`. I'm not giving you the structure because you need to figure that out. That's part of the assignment. I don't expect everyone to have the same structure, either. As before, you are given a skeleton program for the main thread to start with. This skeleton includes the code to print a matrix as well as read a matrix in from a text file on disk.

Details
The main thread will read the one command line argument (the name of the file containing the matrix) and read the matrix into memory. Then, the main thread will create N^2 threads and pass each thread the appropriate information. After the main thread creates all of the other threads, it will wait for each thread to finish. Once all threads have finished, the main thread will print the newly calculated matrix. The easiest (and safest) way of having the threads make their results known to the main thread is for the main thread to setup an empty NxN matrix of integers and give each thread a pointer to one of the locations in the array where the thread will put the result of the multiplication. This is easier with threads than processes because all threads have access to the process' memory without help from operating system (i.e. no shared memory is required).

Thread Responsibilities
The main thread and other threads have distinct roles.

The main thread is responsible for:

- Allocating memory for the input matrix
- Reading the input matrix from a file into its memory.
- Printing the input matrix.
- Allocating memory for the resulting matrix.
- Creating all of the other threads and supplying them with their parameters.
- Waiting for all of the threads to finish.
- Printing the resulting squared matrix.
- Cleaning up any memory or resources the main thread created.

The other threads are responsible for:

- Retrieving their parameters passed to the thread function.
- Given a row and column, multiply them to get an integer value.
- Storing the integer in a location specified by the main thread (it was passed in the structure).
- Cleaning up any resources that the thread may have created.

Notes
This assignment is obviously very similar to the last assignment so much of the logic is the same. The main difference is that you are using threads instead of processes to do the work. You can use as much code from your previous assignment that you want. If you used some helper functions, you should be able to reuse them very easily. I'm not going to specify how to setup your structure or exactly what information you need to pass to the threads. You should be able to figure that out on your own by now. The only thing that is not permitted is to use any global variables anywhere. You don't need them because you are passing the data to the thread function. There are many different ways to structure the data, and all of them are similar. Don't hesitate to make helper functions in your thread code. Also, realize that you do not need any synchronization in this assignment as all of the threads are writing to different locations in the array like the previous assignment.

What to submit
You must submit your `main-thread.c`, `matrix-thread.c`, `matrix-thread.h`, `refman.pdf`, and the `typescript` file. These file must be zipped up and uploaded to appropriate submission page. Note that you are not submitting any other files.

Files	Description
main-thread.c	This is the source code to the main thread that will create multiple threads. This file contains **main.** You must document the file (file header comments) and all functions (function header comments), including **main,** using the appropriate Doxygen tags.
matrix-thread.c	This is the code for the functions that will will calculate the vector multiplications. At a minimum, this will contain the thread function that the main thread will pass to `pthread_create`. You must document the file (file header comments) and all functions (function header comments) using the appropriate Doxygen tags.
matrix-thread.h	The header file containing the structure used to communicate the parameters to the threads and the name of the thread function. This needs to be included by both .c files.
typescript	This is the session that was captured with the script command.
refman.pdf	The PDF produced from the Doxygen tags.

Make sure your name and other info is on all documents and files.