Virginia Tech
Bradley Department of Electrical and Computer Engineering
ECE-3574: Applied Software Engineering * Fall 2012

Homework 6

Submission Details

You must submit the solutions for this homework as an electronic submissions using Scholar (under ECE3574 → Assignments → Homework 6). The submission must be a gzipped tar file (.tar.gz) with your source code. Include all necessary project files, but no binary or compiled files. Your program will be run to evaluate its correctness, and the source code will be reviewed for adherence to the Qt programming style. Your program must run on Ubuntu 12.04 and compile/build using the GNU C/C++ compiler and the qmake/make tools. The following information must be included at the top of each of your source files as comments: your full name, your student ID number, your email address, class (ECE 3574), and the title of the assignment (Homework 6). The submitted file must be given a name in the following form: ***LAST_FIRST_hw6.tar.gz*** where LAST is your last or family name and FIRST is your first or given name. You are only allowed to make one submission. Paper, email or Drop Box submissions will not be accepted. All work must be submitted by the announced due date/time. **Late submissions will not be accepted!** (Don't do it! You have been warned!)

Questions

Use the Homework 6 forum in the Discussion Board area of the class web site to ask questions about this assignment. Do not post questions that contain specific information about the solution.

Honor Code

As stated in the syllabus, in working on homework and projects, discussion and cooperative learning are allowed. However, copying or otherwise using another person’s detailed solutions to assigned problems is an honor code violation. See syllabus for details.

Homework:

Write a multi-threaded console application to perform [MxN] X [NxM] matrix multiplication using POSIX threads (pthreads). The program's name should be “matrix-multiply”.

Specifications

1. You need to create threads using the POSIX pthreads APIs.

2. You need to perform matrix multiplication on two input matrices [A] x [B]. For example:

Let A be a 3 x 2 matrix:

$$A = \begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \\ a_{31} & a_{23} \end{bmatrix}$$

Let B be 2 x 3 matrix:

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

Then, [A] x [B] = [C], which is the 3 x 3 matrix:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

3. Each element (C_{11}, C_{12} ,...., C_{33}) of the resultant matrix C should be calculated by a separate thread. For example the value of C_{11} needs to be calculated by thread_1, the value of C_{12} needs to be calculated by thread_2 and so on. You need to create MxM threads to complete this multiplication.

4. The input matrices A and B will be described in two input files that are specified on the command line. Each line of the input file will describe each row of a matrix. For example, to specify the matrices A and B:

$$A = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \quad B = \begin{bmatrix} 8 & 7 & 6 \\ 5 & 4 & 3 \end{bmatrix}$$

the 2 input files will have the format:

Input-file1	Input-file2

1 4
2 5
3 6
	8 7 6
	5 4 3

Your program will accept the two input files and the output file as command-line arguments:

```
%matrix-multiply input-file1 input-file2 outfile
```

The input files will contain the matrices A and B, respectively, for any values of M and N. M and N are not given. They should be inferred from the input files. You should also check for the correctness of the input files. The input files may contain either integer values or floating point values (but not both).

The output file will contain the resultant matrix C (computed by your program) in the same format as the input files. It will contain the same kind of values (integer or floating point) as the input files. For floating point values, use four (4) decimal places, like this: 0.1234

5. Reading the two input files should be done in parallel, using two threads spawned by the main thread.

6. Define A, B and C as global datatype variables so that they can be easily shared between all the threads.

7. Once the multiplication is done, the parent thread (e.g., main thread) may write the resultant C matrix in the output file.

8. You are NOT allowed to use QThreads for this homework. You need to use the POSIX "pthread" APIs. You would have to `#include <pthread.h>` in your code and during compilation use `-pthread` for linking to the pthread libraries. For example, if you write a multi-threaded application using pthreads in a file named `main.cpp`, the compilation step may be the following command:

```
%  g++ -pthread main.cpp -o matrix-multiply
```

9. You should use qmake for building this project and you are free to reuse classes from previous homeworks and the book's library (e.g., ArgumentList).

10. Make sure to properly implement error handling in your program. (missing files, lack of write permission, etc.)

11. Make sure you implement appropriate synchronization mechanisms (mutexes, semaphores, etc) if/where needed. Your main cpp file should have a comment at the top of the file, where you explain where you used such mechanisms and why, or why you didn't have to use any. This should be around 50-200 words.

Submission

1. Please ensure that you submit before the deadline.
2. Include the ".pro" files for your projects. You should use "make dist" to create the archive.

Grading

- Reading input files: 25%
- Product matrix computation: 40%
- Output: 10%
- Error handling: 10%
- Synchronization mechanisms: 5%
- Style: 10%