CSC 139, Spring 2015
Department of Computer Science, CSUS
2/19/2015

Programming Assignment 2

- **The due day: March 3.**
- **No credit for a program that does not compile or does not run.**

This assignment requires you to write a multi-threaded C program to carry out the matrix multiplication of two simple integer matrices (2×4, and 4×2) as specified in the first assignment, and to display the results. The sixteen elements of the two matrices are given at the command line. You are required to use the POSIX thread (*pthread*) library functions in your program. Make sure that your submitted program compiles and runs in Athena. In addition, you need to compare the multi-processed approach in the first assignment with the multi-threaded approach in this assignment and summarize the differences in terms of the system overhead (process vs. thread creation and communication) and their pros and cons.

1. In your program, include the header <**pthread.h**> (see Pages 172-174 of the textbook). When compiling your program, use a switch "-**lpthread**" to link pthread functions to your program.

2. The main thread in the program takes the input data of sixteen integer values from the command line and verifies for correctness. If the number of input is not correct, or any of the input data is not an integer (e.g., abc, –34a5), the main thread prompts the user with the message: "Incorrect number of input or incorrect type of input" and then exits. Signed integers should be allowed. You can reuse the input validation logic. The order in which input data are used in initializing matrices ***A*** and ***B*** is the same as in Assignment 1.

3. After reading the correct input data, the main thread then creates (**pthread_create**) five threads out of five functions of MC_0, MC_1, MC_2, MC_3 and *Display*. MC_0 through MC_3 calculate c_0 through c_3, respectively, in the resulting matrix ***C***. After finishing their calculations, MC_0 through MC_3 communicate their respective results independently to the thread *Display* that displays the resulting matrix ***C*** using **printf** in the same manner as in the previous assignment. All threads will exit after finishing their tasks (**pthread_exit**).

```
pthread_t  myThread1;
pthread_create(&myThread1, NULL, (void *) MC0, NULL);
```

4. The communication between MC_0 and *Display*, …, and between MC_3 and *Display* is now facilitated through the use of some shared variables in the process. If a need arises to maintain the integrity of a shared variable, a lock mechanism called **mutex** lock and its functions can be used in the following manner:

```
pthread_mutex_t  myLock1 = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_lock(&myLock1);
  x = ……;
pthread_mutex_unlock(&myLock1);
```

5. The main thread will wait until after all five threads finish their tasks (**pthread_join**), then it exits. Arrange some messages (via **printf**) from all threads before they exit. For the main thread, display a message before each thread is created, a message before issuing **pthread_join** calls to start the waiting stage, and a message before it exits. For each of the remaining threads, display a message including its tid (**pthread_self**()) right after being created, a message for its result (computation or communication results), and a message before its termination (**pthread_exit**).

6. Submission Requirements. Your program must include adequate commenting (**points deduction for programs with inadequate comments**). Compile your program using the following:

```
gcc   prog2.c  -o  prog2  -lpthread
```

a) Test your program to make sure it works correctly for at least the following four scenarios. Submit your source file to Assignment2 in SacCT.

```
Prog2 ……//incorrect number of input, say, 10 or 17 integers.
Prog2 ……//incorrect type of input, say, -3a4 in command line.
Prog2 ……//16 integers, say, 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Prog2 ……//16 integers, say, -16 +15 14 13 -12 -11 +10 -9 +8 +7 6 +5 4 3 2 1
```

b) Compare the multi-processed approach with the multi-threaded approach (the system overhead in process/thread creation and communication, and their pros and cons). Submit your comparison in a **.txt** file to Assignment2 in SacCT as well.