CSE5/7343 - Programming Assignment 3

Due Sunday, April 2, 2017 at 11:59 p.m.

Uploaded to Canvas

You will implement the programming assignment using C on the genuse machines provided by the school. When you are done, create a text file <yourname_ID>.txt in the programming directory and put your full name and ID number in the file. Include this file when you turn in your programming assignment. Do not turn in your assignment as separate files. Include all the files in a .zip folder and upload it to Canvas.

Banker's Algorithm (Chapter 7 – Deadlocks – p. 345-346)

For this project, you will write a program that implements the banker's algorithm discussed in Section 7.5.3. Several customers request and release resources from the bank. The banker will grant a request only if it leaves the system in a safe state. A request that leaves the system in an unsafe state will be denied. This programming assignment combines three separate topics: (1) multithreading, (2) preventing race conditions, and (3) deadlock avoidance.

The Banker

The banker will consider requests from *n* customers for *m* resources types as outlined in Section 7.5.3. The banker will keep track of the resources using the following data structures:

```
/* these may be any values >= 0 */
#define NUMBER OF CUSTOMERS 5
#define NUMBER OF RESOURCES 3

/* the available amount of each resource */
int available[NUMBER OF RESOURCES];
```

```
/*the maximum demand of each customer */
int maximum[NUMBER OF CUSTOMERS][NUMBER OF RESOURCES];

/* the amount currently allocated to each customer */
int allocation[NUMBER OF CUSTOMERS][NUMBER OF RESOURCES];

/* the remaining need of each customer */
int need[NUMBER OF CUSTOMERS][NUMBER OF RESOURCES];
```

The Customers

Create *n* customer threads that request and release resources from the bank. The customers will continually loop, requesting and then releasing random numbers of resources. The customers' requests for resources will be bounded by their respective values in the need array. The banker will grant a request if it satisfies the safety algorithm outlined in Section 7.5.3.1. If a request does not leave the system in a safe state, the banker will deny it. Function prototypes for requesting and releasing resources are as follows:

```
int request resources(int customer num, int request[]);
int release resources(int customer num, int release[]);
```

These two functions should return 0 if successful (the request has been granted) and -1 if unsuccessful. Multiple threads (customers) will concurrently access shared data through these two functions. Therefore, access must be controlled through mutex locks to prevent race conditions. The use of Pthreads mutex locks is covered in Section 5.9.4.

Implementation

You should invoke your program by passing the number of resources of each type on the command line. For example, if there were three resource types, with ten instances of the first type, five of the second type, and seven of the third type, you would invoke your program follows:

./a.out 10 5 7

The available array would be initialized to these values. You may initialize the maximum array (which holds the maximum demand of each customer) using any method you find convenient.

Reminder: Although most or all of this programming assignment should "just work" in many other environments (cygwin, OSX, solaris, etc.), note that (1) I will not be able to assist in setting up or debugging problems caused by differences in the environment and (2) statements like "it worked on my home machine" will not be considered in the grading process. If you choose to do development in an unsupported environment, it is *your responsibility* to leave adequate time to port your solution to the supported environment, test it there, and fix any problems that manifest.