Experiment 3

Deadline: 2024/5/5 23:59

- Programming Projects
  Textbook (version 9) P345

Task: Banker’s Algorithm

Notes:
(1) Please submit all your source codes.
(2) Please submit your report, showing your experimental procedures and results. Anything else that you would like to report correlated to this project is also welcomed.
(3) You may refer to any other materials besides the textbook for completing the tasks, and please cite them properly in your report.

Programming Projects

Banker's Algorithm

For this project, you will write a multithreaded 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. Both the Pthreads and Windows APIs provide mutex locks. The use of Pthreads mutex locks is covered in Section 5.9.4; mutex locks for Windows systems are described in the project entitled "Producer–Consumer Problem" at the end of Chapter 5.

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.