Lab 5 Report

Devin Westbye

Muhammad Ahmad

Timothy MacDougall

The program generates 5 threads using the pthreads library. The threads share data using the following global arrays:

**int available[NUM\_RESOURCES]** - the number of each resource that is available

**int maximum[NUM\_CUSTOMERS][NUM\_RESOURCES]** - the maximum demand of each customer

**int allocation[NUM\_CUSTOMERS][NUM\_RESOURCES]** - the amount currently allocated to each customer

**int need[NUM\_CUSTOMERS][NUM\_RESOURCES]** - the remaining need of each customer

The threads run the function requestReleaseRepeat which randomly generates the maximum amount that each process will need. The process or thread then requests a random amount of resources bounded by the need using the request\_Res function. Calling request\_Res is the critical section of the code, so a mutex lock is used to prevent race conditions. Once the process is given all the resources it needs, it releases all its allocated resources using the release\_Res function. This also requires a mutex lock.

The request\_Res function first checks that the request does not exceed the need of the process or the available resources. If it does, the request is denied. It then tentatively allocates resources to the process and runs the is\_safe function which uses the banker’s algorithm to determine if the resulting change leaves the system in a safe state. If it does not, the resources are reverted to the previous safe state and the request is denied. If it does then the request is granted.