CSC 360 - P2 Design Document

Name: Jordan Grubisich

ID: V00951272

1. Each customer in the system will have their own individual thread which will be created by the main thread. Each thread will simulate the arrival of its corresponding customer, that customers wait time in the queue and the interaction between the customer and the clerk. The customer thread will signal to other waiting threads once it has finished being served.

2. The threads will generally work independently of one another, the only interaction between threads will be the signal broadcasted once a customer is finished being served.

3. There are 5 mutexes in total:

    a. busMutex – guards operations that make changes to the business customer queue.
    b. ecoMutex – guards operations that make changes to the economy customer queue.
    c. clerkMutex – guards operations that make changes to the clerks array.
    d. custMutex – guards functional operations within the concurrent customer thread.
    e. timeMutex – guards operations that edit the wait time variables.

4. Once the customer threads are created, the main thread will wait idly for the customer threads to join.

5. Customers will be represented by a struct containing the customer's ID, flight class, arrival time and service time. This struct will be passed to each customer thread as it is created.

6. By implementing locks and unlocks of the above mentioned mutexes, concurrent modification of protected data structures will be prevented.

7. There is 1 convar in total:

    a. The convar "free_clerk" will represent the condition that a clerk is available for service.
    b. The mutex "custMutex" will be associated with it as it protects nearly all of the operations in the customer thread that modify protected data structures.
    c. After pthread_cond_wait() has been unblocked and the mutex re-acquired, the operation of serving the customer will begin which includes removing them from the queue, recording their wait time in the queue and simulating their interaction with the available clerk.

8. Customer Thread Algorithm:

Wait until time to arrive;
Indicate that you, a customer, have arrived and state your ID;
Join the queue corresponding to your class;
If you are in economy and there are more business customer than free clerks available
    OR if there are no free clerks available
    OR if you are not at the front of the queue
Then wait until another customer signals that a clerk is free and re-check the above conditions;
When it is your time to be served, lock clerkMutex;
Gather the ID of the clerk that will be serving you;
Unlock clerkMutex;
Leave the queue in which you were waiting;
Record the time spent waiting in the queue;
Announce that you are being served, the service start time, your ID and the clerk's ID;
Begin being served by clerk for your indicated service time;
Announce that you have finished being served, the service end time, your ID and the clerk's ID;
Lock clerkMutex;
Set the status of the clerk who served you back to free;
Unlock clerkMutex;
If there are still customers waiting in either queue, signal to them that a clerk is now free and
    unlock custMutex;
exit and join main thread;