Airline Check-in System – Design Document

CSC360 Fall 2018
Zhe(Kevin) Chen
V00819544

1. **How many threads are you going to use? Specify the task intend each thread to perform.**
   There will be thread for each customer and each thread(customer) will call usleep() to simulate its service time. There will also be thread for each clerk and each thread(clerk) will signal customer in queue and then serve customer.
2. **Do the threads work independently? Or is there an overall controller thread?**
   Customer and clerk threads are working independently since each customer is independent. There will be no controller thread.
3. **How many mutexes are you going to use?**
   There will be mutex for each queue.
4. **Will the main thread be idle?**
   Yes, the main thread will be idle after created all the threads needed, it waits for all the customers to be served(exit).
5. **How are you going to represent customers? Data structure.**
   There will be several data in the Customer data struct: customer id/class type/arrival time/start time/service time/clerk id(served). Queue(both business and economic class) that holds customers can be implemented using FIFO linked list.
6. **How are you going to ensure that data structures in your program will not be modified concurrently?**
   Pthread mutex is the solution to prevent such thing to happen from our design, it works in a way that only one process can enter its critical section such as write to queues.
7. **How many convars are you going to use? a) Describe the condition that the convar will represent. b) Which mutex is associated with the convar will represent. c) What operation should be performed once pthread_cond_wait() has been unblocked and re-acquired the mutex.**
   At this moment I'm thinking to use four conditional variable for four clerks to signal customer that's waiting in the queue.
   a) When any clerk is available(idle) this clerk will signal its conditional variable.
   b) The queue mutex is associated with clerk conditional variable. This is because when the customer being called it will be leaving the queue which requires read and write to the queue.
   c) When pthread_cond_wait() has been unblocked and re-acquired the mutex, this customer thread will be served by the clerk who signaled it.
8. **Briefly sketch the overall algorithm you will use.**
   1) **Main thread**
   set up the initial values and attributes
   read file and create customer thread for each customer which contains their information, create them all at once
   create four clerk threads
   while there is still unfinished customer thread do nothing

calculate and print all the stats

2) **Customer thread**
(assume it already has all the information on this specific customer)
sleep for its arrival time
mutex(based on class type) lock
enter a queue based on its class type
wait for the signal
check if this customer is actually the one being called
if no go back to wait (done in while loop)
mutex unlock
record the clerk that signaled this customer
record the current time as service start time
update stats such as overall waiting time and queue waiting time
get served(sleep for its service time)
signal clerk that service is finished

3) **Clerk thread**
loop
check if any queue has customer
if yes
mutex lock
signal that queue's mutex
mutex unlock
mutex lock
wait for customer to check in
mutex unlock
else do nothing