Julia Sinclair
V00890683
CSC 360 Fall 2022
Assignment 2

Design Document

1. How many threads are you going to use? Specify the task that you intend each thread to perform.

In my program, there is one main thread and then a thread for each customer specified in the input file. The main thread reads the file input, creates the customer threads, and calculates the average waiting times. The customer threads sleep until the customer's arrival time, add themselves to their respective queue, dequeue when appropriate, change a clerk's status to busy and then free again after service time, and calculate its own waiting time.

2. Do the threads work independently? Or, is there an overall "controller" thread?

The main thread creates the customer threads. However, once created the customer threads they will work independently as each customer is independent.

3. How many mutexes are you going to use? Specify the operation that each mutex will guard.

I am going to use five mutexes; customer, clerk, economy, business and time.
Customer mutex will be an array mutex, where customer[0] represents economy and customer[1] represents business. Customer mutex will be used in conjunction with a condition variable. See Question 7 for more details. Clerk mutex will guard changing a clerk's availability in the clerk array. Mutex locks will be used when changing clerk[i]'s value from 0 to 1 (now busy) and from 1 to 0 (now available). The economy and business mutexes will protect their respective class queues; mutex locks will be needed for (1) enqueueing, (2) dequeuing and (3) checking if a customer is at the head of its queue. Time mutex will also be an array mutex, where time[0] represents economy waiting time and time[1] represents business waiting time. Time mutex will guard adding a customer's individual waiting time to its class's overall waiting time.

4. Will the main thread be idle? If not, what will it be doing?

After the customer threads have been created, the main thread will be idle. It will wait until all customer threads exit.

5. How are you going to represent customers? What type of data structure will you use?

Customers will be represented using the customer structure provided in tutorial. This struct holds 4 integers; user_id the customers given ID, class_type the customer class (0 for economy and 1 for business), arrival time when customer arrives, and service time how long customer needs with a clerk. All customers will be held in a customer array when read from file, then each customer in the array will be parsed into its own thread.

6. How are you going to ensure that data structures in your program will not be modified concurrently?

Data structures in my program will not be modified concurrently because all critical sections, where data could be changed by multiple threads at one time, will be guarded with mutexes using mutex lock/unlock. Mutex locks ensure that only one thread can read and write in the code section between mutex lock and mutex unlock at one time.

7. How many convars are you going to use? For each convar:

I will only use 1 convar

(a) Describe the condition that the convar will represent.

The convar will represent a clerk becoming available. For cond_wait, customers will have to wait for a clerk to become free. Cond_broadcast will signal all waiting customers that a clerk has become free.

(b) Which mutex is associated with the convar? Why?

The associated mutex is the customer mutex. This mutex will make sure that customers of the same class are not checking that it is their ‘turn’ to be with a clerk at the same time. This prevents multiple customers from passing the clerk ‘turn’ conditions and dequeuing when there are not enough available clerks.

(c) What operation should be performed once pthread cond wait() has been unblocked and re-acquired the mutex?

Once pthread_cond_wait() has unblocked, the customer should recheck if it is their ‘turn’ with a clerk. The customer will recheck (1) if they are at the start of their class queue, (2) that there is a clerk available, and (3) if they are in economy, that there is enough clerks available to serve all of the customers in business class and themselves. If customer pass does not pass this check, then back waiting in the pthread_cond_wait(). If they do pass, then mark the available clerk as unavailable, dequeue from the customer’s class queue, and the clerk serves the customer.

8. Briefly sketch the overall algorithm you will use.

Main Thread

Read file
Store the total number of customers
Calculate number of business customers and number of economy customers
Initialize every clerk in clerk array to 0
Initialize machine time
Create a thread for each customer with their information
Wait for every customer thread to exit
Destroy mutexes and convar
Calculate average wait times
Print average wait times

Customer Thread

Sleep until arrive time
Print that a customer arrives
Business or Economy Mutex Lock
Join end of class queue
Business or Economy Mutex Unlock
Store time when customer joined class queue
Customer Mutex Lock
Check if customer is head of its queue, that there is an available clerk and if customer is in economy class, that there is enough clerks available to serve all business customers
If customer does not pass check, wait for signal
Customer Mutex Unlock
Clerk Mutex Lock
Mark the available clerk is now busy
Clerk Mutex Unlock
Business or Economy Mutex Lock
Remove customer from queue
Business or Economy Mutex Unlock
Store time when customer left class queue
Print that a clerk started serving customer
Sleep for service time
Print that clerk is finished serving customer
Clerk Mutex Lock
Mark the clerk that served customer is now available
Clerk Mutex Unlock
Signal all waiting customers that a clerk is available
Calculate waiting time for customer
Time Mutex Unlock
Add waiting time to class’s overall waiting time
Time Mutex Unlock
Exit