CMPE 322 PROJECT 2

Elif Çalışkan
2016400183

Problem Description:

Implementing a simple flight reservation system by using threads.

Design and Methodology:

I used a mutex array for every seat. Since there cannot be more than 100 seats, I initialized its size to 100. I created the client threads in a for loop up to the given argument. I sent every client their client number while creating the threads. After waiting for every client to terminate, the message "All seats are taken" is printed and the program finishes.

In client function, server thread is created, and client's id is sent. Then client sleeps for 50-200 milliseconds. After sleeping, the client picks a random seat in a while loop. If the seat is locked, it means the seat is taken and it chooses another seat. If it can acquire its lock, then the reservation queue is locked, write_mutex is acquired. Because the client wants to add its name to the reservation list. So a Reservation node is created with the according seat and client numbers. This node is appended to the reservationQueue. After adding the node, write_mutex is unlocked. Since mutex is used, other clients cannot write their names before this client. Then it waits for the server to terminate.

In printServer function, server checks if the first node of the reservationQueue belongs to its client. If it is its time to write, write_mutex is acquired and the first node is written to output. After writing, the first node becomes the next one and head is changed. After removing the first element, lock is released. Since the queue is a global variable, I used write_mutex for changing it. It means, there cannot be an addition to the queue when removing is done or there cannot be more than one addition or remove operation.

My code takes about 8-9 seconds to run for 100 seats. Since I am using a linked list the creation of nodes takes time.

To sum up, I checked for available seats at client threads and logged the reservation at server threads. For keeping the earlier reservation before the others, I used a reservation queue. This is a linked list. When a seat is chosen, reservation node is created and appended to queue. Server constantly checks whether the first reservation belongs to its client. When it is found, server writes the reservation to output.txt. I used a mutex array for seats, I locked every chosen seat. I used write_mutex for queue. For removing the head in server or appending a reservation in client, the lock should be acquired.

The platform I used to develop:

I wrote the code in Ubuntu. After that, I used G++ compiler tools for testing the code.

How to run the program:

make
./main seat_number

output.txt gets created