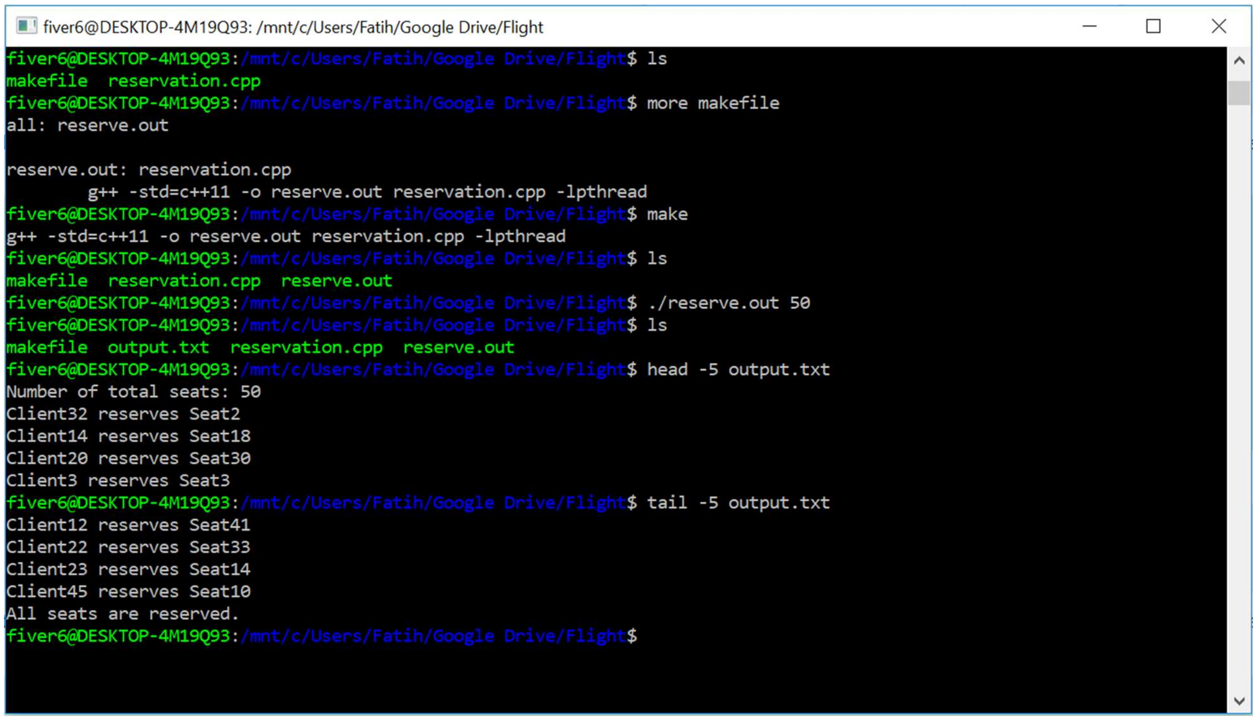
FLIGHT RESERVATION SYSTEM

The screenshot below shows how to compile and run the FRS (Flight Reservation System).


fiver6@DESKTOP-4M19Q93: /mnt/c/Users/Fatih/Google Drive/Flight

```
fiver6@DESKTOP-4M19Q93:/mnt/c/Users/Fatih/Google Drive/Flight$ ls
makefile  reservation.cpp
fiver6@DESKTOP-4M19Q93:/mnt/c/Users/Fatih/Google Drive/Flight$ more makefile
all: reserve.out

reserve.out: reservation.cpp
        g++ -std=c++11 -o reserve.out reservation.cpp -lpthread
fiver6@DESKTOP-4M19Q93:/mnt/c/Users/Fatih/Google Drive/Flight$ make
g++ -std=c++11 -o reserve.out reservation.cpp -lpthread
fiver6@DESKTOP-4M19Q93:/mnt/c/Users/Fatih/Google Drive/Flight$ ls
makefile  reservation.cpp  reserve.out
fiver6@DESKTOP-4M19Q93:/mnt/c/Users/Fatih/Google Drive/Flight$ ./reserve.out 50
fiver6@DESKTOP-4M19Q93:/mnt/c/Users/Fatih/Google Drive/Flight$ ls
makefile  output.txt  reservation.cpp  reserve.out
fiver6@DESKTOP-4M19Q93:/mnt/c/Users/Fatih/Google Drive/Flight$ head -5 output.txt
Number of total seats: 50
Client32 reserves Seat2
Client14 reserves Seat18
Client20 reserves Seat30
Client3 reserves Seat3
fiver6@DESKTOP-4M19Q93:/mnt/c/Users/Fatih/Google Drive/Flight$ tail -5 output.txt
Client12 reserves Seat41
Client22 reserves Seat33
Client23 reserves Seat14
Client45 reserves Seat10
All seats are reserved.
fiver6@DESKTOP-4M19Q93:/mnt/c/Users/Fatih/Google Drive/Flight$
```

DEVELOPMENT ENVIRONMENT:

Operating System: Ubuntu 16.04.5 LTS

Development Language: C++

*g++ is used for compilation with flag std set to c++11

How Does It Work?

The program takes one parameter which is the available seat number. Then in the main thread program creates client and server threads as many as the number of available seats. Then main thread makes the clients thread execute. Then each client sleeps some random amount of time. Then clients choose a random seat to make a reservation request. Each client make sure seat is available before they make a reservation request and each client tries until it succeeds with the reservation. Reservation requests are handled by server threads. Server threads updates the reservation table for the requested seat number and logs the result.

There are two mutex locks in the program. One for to protect output file to be written simultaneously by server threads. The other one for to prevent client threads to make request for the same seat number. If a client thread has the lock for the related seat, other clients wait until the lock is released to check the reservation state of the related seat.