**COSC-5328\_48F Computing Network**

**Instructor - Dr. Lawrence Osborne**

**Project Report**

**Airline Reservation System Project**

**Submitted By:**

**Balram Agrawal (L20365063)**

**Shubhashish Mishra (L20365063)**

**Submitted On:**

**August 14, 2015 Acknowledgement:**

We would like to express my deepest appreciation to all those who provided us the possibility to complete this report. We extend a special gratitude to Dr Lawrence Osborne, professor of Computer Science department at Lamar University, whose contribution in stimulating suggestions and encouragement helped us to coordinate this project; especially in writing this report.

The supervision and support that he gave truly helped in the progression and smoothness of the project. Without her help we could not have presented this progress report upto the present standard.

Date: 08/14/2015

Balram Agrawal (L20365062)

Shubhashish Mishra (L20365063)

**Abstract:**

The main objective of the project is to develop a client/server database system for Airline Reservation using TCP socket that provide client with the facility to query the flight status, reservation for a particular flight and cancellation of reservation. The central servers hold all the data of the flight which are accessible to all the clients. Upon initiation of the TCP connection by the client, the authentication process allow only legitimate client to have access to the server and query, reserve or cancel reserved seat. The central server maintains the status of every flight by maintaining database in single file. The system maintains robustness by eliciting a reasonable error message for erroneous command.

Contents

[Introduction 5](#_Toc427318330)

[Problem Statement 6](#_Toc427318331)

[SOFTWARE REQUIREMENTS 6](#_Toc427318332)

[SYSTEM ARCHITECTURE & DESIGN 7](#_Toc427318333)

[Architecture 7](#_Toc427318334)

[System Design 8](#_Toc427318335)

[Authentication Module: 8](#_Toc427318336)

[Query module: 9](#_Toc427318337)

[Reservation Module 10](#_Toc427318338)

[Cancellation Module: 11](#_Toc427318339)

[UNIX Lock File write module: 12](#_Toc427318340)

[Screenshot 13](#_Toc427318341)

[Conclusion 18](#_Toc427318342)

[References 19](#_Toc427318343)

# Introduction

Client-server architecture (client/server) is network architecture in which each computer or process on the network is either a client or a server. Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network servers). Clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

Airline Reservation System Project is a client/server database system that facilitates query, reservation and cancellation of reserved flight online by the authorized user. The system provides an easy interface for the client to have access to the flight information online and perform necessary action. The system is primarily concerned with providing three main services to its client:

1. Query for number of seats available on flight x.
2. Reserve seats under username
3. Cancel seats for flight x under username

# Problem Statement

In the given project, we are supposed to develop a client/server database system that has a central computer which holds database for airlines reservation which run for 4 flights having capacities of 5, 2, 10 and 10 respectively. The system is supposed to handle request from travel agent from all over the world to the central computer and support following services through command as follows:

1. Query for number of seats available on flight x : Q x
2. Reserve seats I, j, etc. user name nm : R I j ... nm
3. Cancel seats for flight x under name n : C x n

The system is supposed to have any number concurrent access from the agent for reservation. The system also takes consideration of lock while updating the data whether its safe to update or not. If other client are currently reserving or cancelling the flight, the system lock the flight record for the duration of transaction so no other agent can take the available seat in the interim.

# Software Requirement

Operating System: UNIX/LINUX

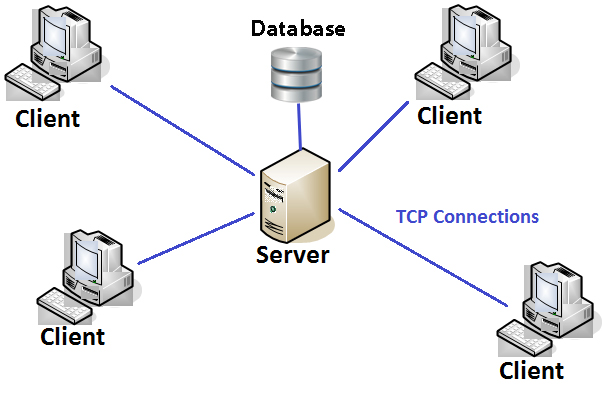
Technology: Python

Database: File system

# SYSTEM ARCHITECTURE & DESIGN

## Architecture

The system is based on Client/Server Architecture and the diagram below represents the flow of requests from clients to servers.



In this scenario overall system has three main actors:

1. Client: Clients in the system are the agents and user that connects to the system to get service. The number of client is indefinite for the system i.e. any range from 1 to server capacity.

2. Server: It is the central component of the system that is responsible for handling client request and serving them. It retrieves the flight data stored on database as a file and honor request based on that.

3. Database: The database is stored in file with each information for each flight. For any query request, reservation and cancellation, the data is retrieved and served back.

## System Design

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

We have deployed following module for efficient working of the system:

### Authentication Module:

This module is explicitly associated to authenticate the client and only allow legitimate travel agent to log on to the system and have access to the reservation system. The server terminates the connection if the username and password don’t match up.

# authentication feature

auth = 1

i = 100

user = clientsocket.recv(1024)

print user

while 1:

passwd = clientsocket.recv(1024)

if passwd != None:

break

print passwd

try:

i = username.index(user)

print i

if password[i] != passwd:

auth = 0

print "wrong password"

except ValueError:

print "No such username"

auth = 0

if auth == 0:

clientsocket.send("Bad username/password combination")

clientsocket.close()

else:

clientsocket.send("!!!Authentication successful!!!")

### Query module:

Query module is related to handle the query request of the travel agent. The module first determines which flight is client interested to query and determines the status of the flight based on the reserved/empty field of the flight in the database. If 0, which means empty, the module add the flight information to the reply string and sent back to the client.

if data[0].lower() == 'q': # query section

sq = ""

#f = open('flight.txt', 'r')

print "query request"

que = data.split(' ')

print que

if que[1] == '1': # for flight 1

for q in range(0,5):

#print element[q][2]

if element[q][2] =='0':

sq = sq + '\t'.join(element[q]) + '\n'

elif que[1] =='2': # for flight 2

for q in range(5,7):

if element [q][2] =='0':

sq = sq + '\t'.join(element[q]) + '\n'

elif que[1] =='3': # for flight 3

for q in range(7,17):

if element [q][2] =='0':

sq = sq + '\t'.join(element[q]) + '\n'

elif que[1] =='4': # for flight 4

for q in range(17,27):

if element [q][2] =='0':

sq = sq + '\t'.join(element[q]) + '\n'

if sq =="":

sq = "No seat available on flight"

### Reservation Module

This module is responsible for reserving the empty seat of the respective flight queried. The reservation is immediately followed by the query so, the context define which flight must be reserved. The module change the empty/reserved field and username of the flight in the database. The module change empty/reserve field of respective seat to 1 and username to the respective username provided in the command.

elif data[0].lower() =='r': # Reserve section

sq = "Reservation of flight "

print "Reserve seat"

reserve = data.split(' ')

flight = que[1]

sq = sq+flight+" done Successfully!!!"

for a in range(1, len(reserve)-1):

if flight == '1':

for b in range(0,5):

if reserve[a] == element[b][1]:

element[b][2]= '1'

element[b][3]= reserve [len(reserve)-1]

if flight == '2':

for b in range(5,7):

if reserve[a] == element[b][1]:

element[b][2]= '1'

element[b][3]= reserve [len(reserve)-1]

if flight == '3':

for b in range(7,17):

if reserve[a] == element[b][1]:

element[b][2]= '1'

element[b][3]= reserve [len(reserve)-1]

if flight == '4':

for b in range(17,27):

if reserve[a] == element[b][1]:

element[b][2]= '1'

element[b][3]= reserve [len(reserve)-1]

### Cancellation Module:

This module is related to cancelling of the allocated seat to the user. The module cancels the entire seat allocated to the given username for the given flight. The module change empty/reserved field and username field of the flight back to the 0 and ‘non’ respectively.

elif data[0].lower() == 'c': #cancel reservation

print "Cancel reservation"

cancel = data.split(' ')

sq = "Reservation of flight "+cancel[1]+" calcelled for "+cancel[2]

if cancel[1] == '1':

for e in range(0,5):

if element[e][2]=='1' and element[e][3]==cancel[2]:

element[e][2]= '0'

element[e][3]= "non"

if cancel[1] == '2':

for e in range(5,7):

if element[e][2]=='1' and element[e][3]==cancel[2]:

element[e][2]= '0'

element[e][3]= "non"

if cancel[1] == '3':

for e in range(7,17):

if element[e][2]=='1' and element[e][3]==cancel[2]:

element[e][2]= '0'

element[e][3]= "non"

if cancel[1] == '1':

for e in range(17,27):

if element[e][2]=='1' and element[e][3]==cancel[2]:

element[e][2]= '0'

element[e][3]= "non"

### UNIX Lock File write module:

This module explicitly handles the case for mutual exclusion so that no race condition arises while reserving the flight. The system lock the database file before any update starts to happen making it accessible to a single client while writing and have problem of race condition solved. The system uses UNIX system lock called ‘flock’ when write is about to happen. The module have no effect for reading the file.

#writing to file

f = open('flight.txt', 'w')

try:

fcntl.flock(f,fcntl.LOCK\_EX | fcntl.LOCK\_NB)

except IOError:

wr = '0'

print "\ncannot be written in file\n"

else:

wr = '1'

for i in range(len(element)):

text = '\t'.join(element[i]) + '\n'

f.write(text)

#time.sleep(8)

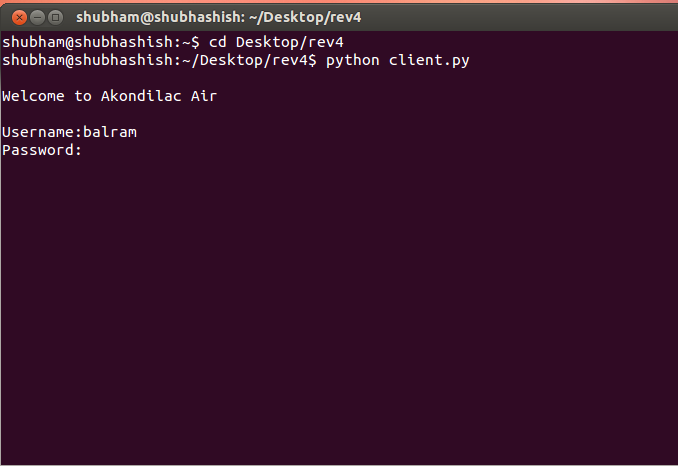
fcntl.flock(f, fcntl.LOCK\_UN)

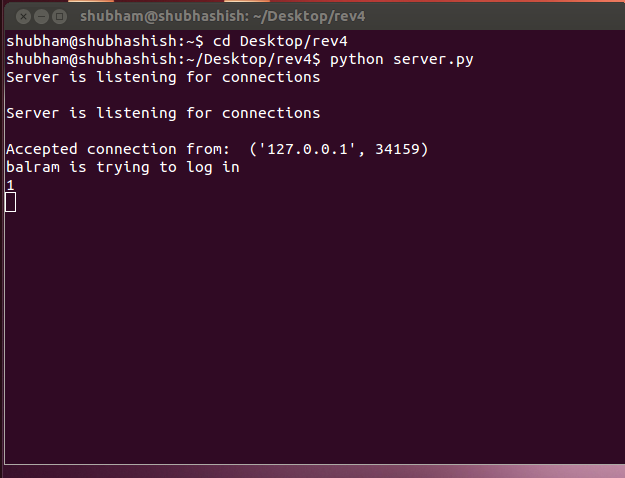
f.close()

if wr == '0':

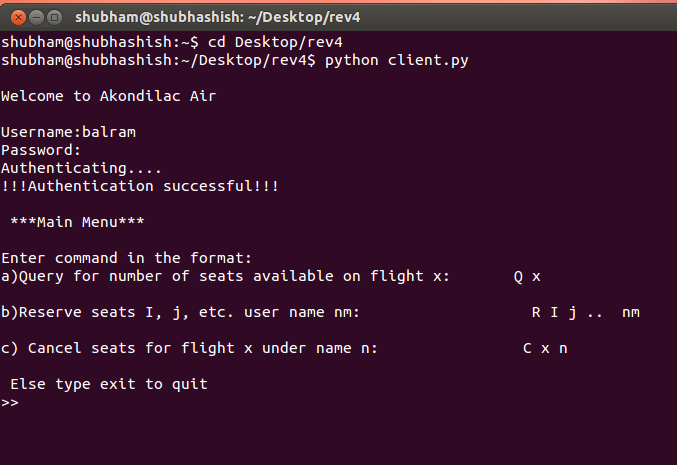
sq = "Reservation/Cancellation in process by other client.\nTry after sometime "

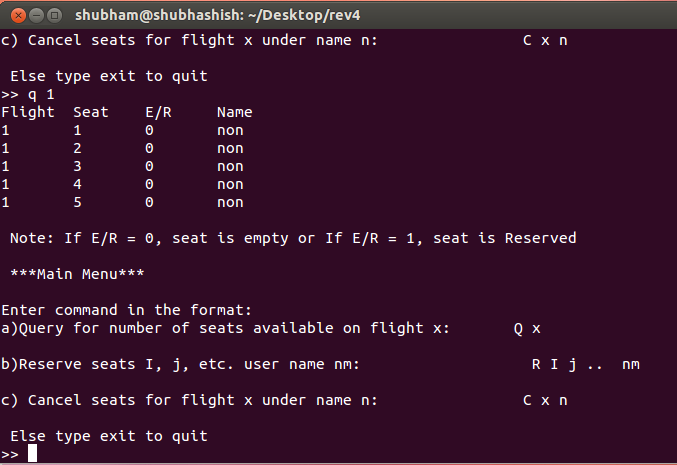
# Screenshot

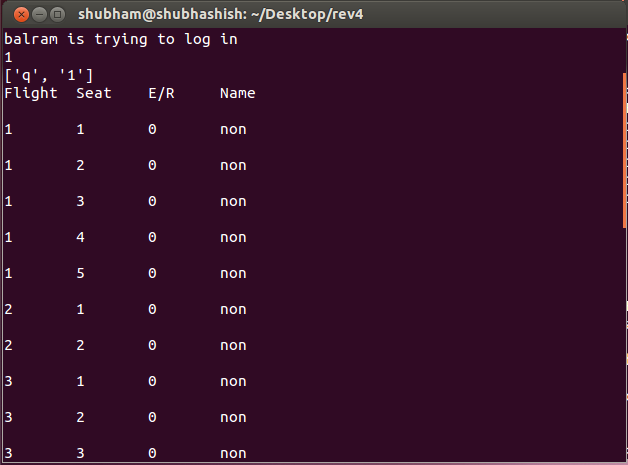
Login screen Client

Server

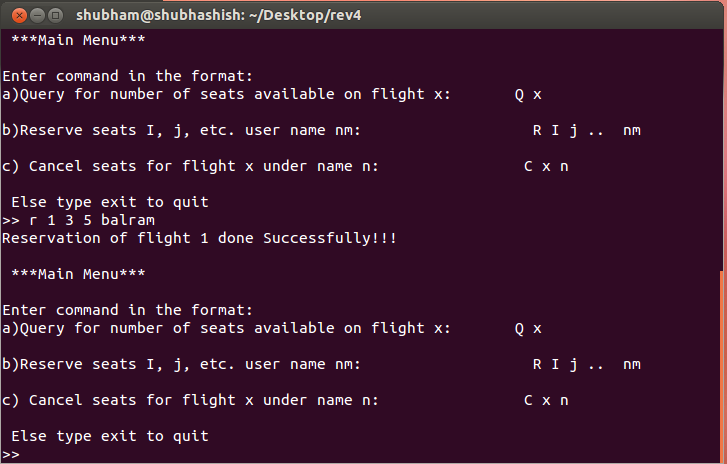
Main Menu Client

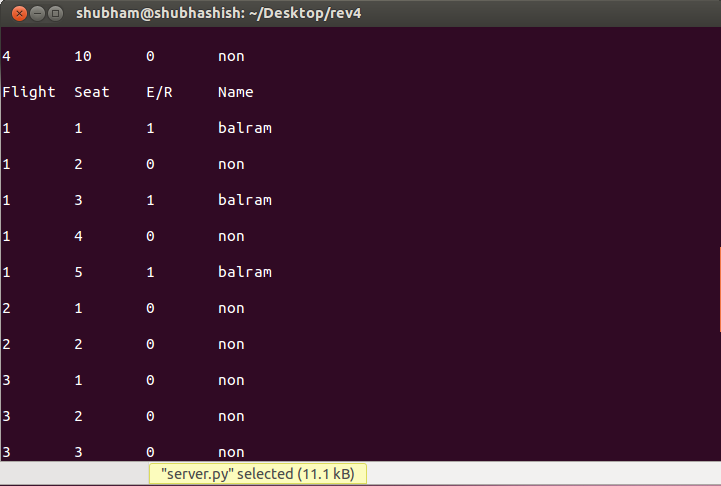


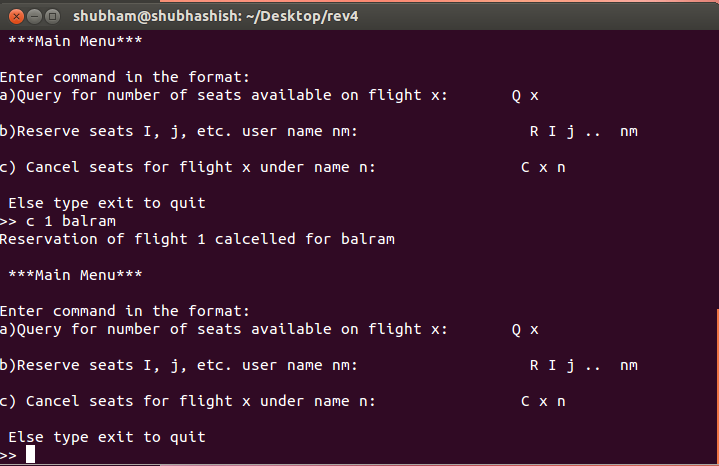
Query Client

Query Server

Reserve Client

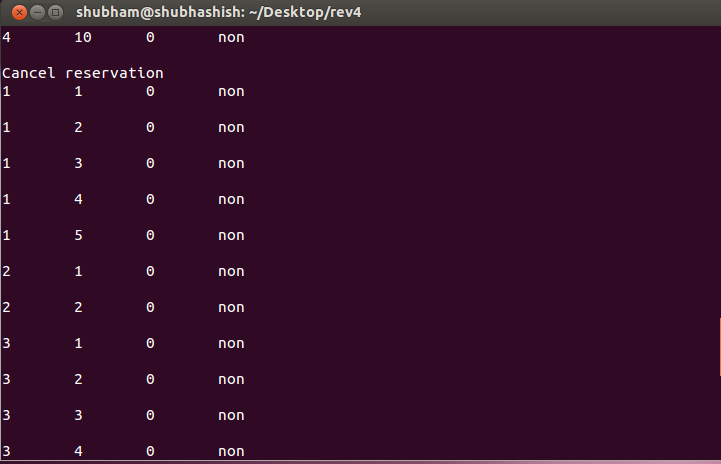


Reserve server

Cancel Client

Cancel Server

Cancel server



# Conclusion

The project was successfully developed as a Client-Server model with central computer working as a server and multiple travel agent around the world working as a client. The central computer is designed to take multiple client requests concurrently and service them with either query of flight, reservation of flight or cancellation of reserved flight. The communication between travel agent and the central computer was accomplished using TCP. To summarize, we developed a client/server database in an UNIX environment that provides facility of querying the flight, reserving the flight and cancelling the reserved flight.

# References

1. [**http://tilde.town/~cristo/file-locking-in-python.html**](http://tilde.town/~cristo/file-locking-in-python.html)
2. [**https://docs.python.org/2/library/fcntl.html**](https://docs.python.org/2/library/fcntl.html)
3. [**http://www.tutorialspoint.com/python/python\_command\_line\_arguments.htm**](http://www.tutorialspoint.com/python/python_command_line_arguments.htm)
4. [**https://docs.python.org/2/tutorial/errors.html**](https://docs.python.org/2/tutorial/errors.html)
5. **https://docs.python.org/2/tutorial/inputoutput.html**
6. **Text Book: Computer Networking: A Top-Down Approach, James F. Kurose and Keith W. Ross**