CSCI3240 Project3: Client-Server Programming

1 Introduction

The objective of this project is to learn TCP client-server interaction using socket interface in C programming language. After completing this project, you will have a basic understanding of the steps required to develop a networking application.

2 Project Specification

In this project, you are required to do socket programming in C language (Linux environment) to implement a pair of client and server that provides following functionalities:

- The server will act as a datastore for employees.
- A employee record on the server will contain employee attributes such as First Name, Last Name, Zip Code, Depearment, and Salary. The employee record should be saved in a csv file named ***records.csv*** (same as lab 6). Each employee record should be stored as a seperate line in the file. Each attributes should be seperated by comma.
  Example contain of records.csv file:

  ```
  Pablo,Picasso,37128,Arts,65000
  Jack,Sparrow,07801,Movies,40000
  Jackie,Chan,12345,Martial Arts,50000
  Charles,Babbage,37128,Computer Science,55000
  ```

- Only server should access the ***records.csv*** file.
- The client will be interactive, and allow user to perfrom following actions:
  1. Add new record to the server (i.e. server should receive the information on the new student from the client. Next, server should append the new record into the ***records.csv*** file.)
  2. Request the server to serach the information on employees by name (full name). The server should send the matching record(s) back to the client. Note: There can be multiple employee with same name.

3. Request the server to serach the information on employees by zip code. The server should send the matching record(s) back to the client.
4. Request the server to serach the information on a employee by salary and a comparision operator. The comparison operator contains one of the operators in the set:

   `{ ">", "<", ">=", "<=", "==" }`

   Example scenarios:

   i. If the *Salary* value is 50,000 and the *Comparision Operator* is "$>=$", your function should find all the employees whose salary is greater than or equal to 50,000.

   ii. Similarly, if the provided *Salary* value is 40,000 and the *Comparision Operator* is "==", your function should find all the employees whose salary is equal to 40,000.

   The server should send the matching record(s) back to the client.
5. Close the connection with the server (The server should not terminate).

Specifically, your client and server programs entail to achieve the following requirements:

1. Your client program needs to take two arguments that specify the name of server and the port that it is trying to connect to. Your program for server needs to take an argument that specifies the port that it is listening to. You can use 3240 as a port number.

2. The server program will start first and keep listening to the specified port. Your client will connect to the port that your server is listening to, and a socket between your client and server is constructed.

3. After the connection with the server is established, your client program will first prompt a message that asks the user to select an option. Options:
   (1) Add Record
   (2) Search By Name
   (3) Search by Zip Code
   (4) Search by Salary
   (5) Terminate.

4. If option 1 is selected, your client should ask the user for the employee's first name, last name, zip code, depertment, and salary. Next, the record should be sent to the server. Then, your server should add the new employee record to ***records.csv*** file. Your server should then send an acknowledgment message to the client stating that the new record has be added. The client should display the message.

5. If option 2 is selected, your client should ask the user the first name and the last name of the employee to search. The client should send this information to the server. The server should then search for the matching record(s) with same first and last name in the ***records.csv*** file and send them back to the client. The client should display them. If no matching record is found, the server should send a message to the client indicating that the record was not found; the client should display it.

6. If option 3 is selected, your client should ask the user the zip code. The client should send this information to the server. The server should then search for the employee(s) living in the requested zip code and send it back to the client. The client should display it. If no matching record is found, the

server should send a message to the client indicating that the record was not found; the client should display it.

7. If option 4 is selected, your client should ask the user the salary and a comparision type. The client should send this information to the server. The server should then search for the record(s) that satisfies the search criteria and send it back to the client. The client should display it. If no matching record is found, the server should send a message to the client indicating that the record was not found; the client should display it.

6. If option 5 is selected, the client should close the connection and terminate (sever should not terminate).

7. The client should be able to interact with the server until option 5 is selected.

3 Programming Notes

I suggest you to start modifying the sample client and server code provide in Lab 9 document.

This project will help you to understand the basic client-server interaction using TCP sockets.

You can use `azuread jupyterhub server` for this project. You can run server program in one terminal window and a client program in another terminal window.

4 Example Client Output

```
$./client localhost 3240
(1) Add record
(2) Search by Name
(3) Search by Zip Code
(4) Search by Salary
(5) Terminate
Select an option [1,2,3,4 or 5]: 1

Enter First Name: Jack
Enter Last Name: Sparrow
Enter Zip Code: 32130
Enter Department: Accounting
Enter Salary: 30000
Message From Server:
Record added Sucessfully!!

(1) Add record
(2) Search by Name
(3) Search by Zip Code
```

```
(4) Search by Salary
(5) Terminate
Select an option [1,2,3,4 or 5]: 2

Enter first name: Jack
Enter last name: Sparrow
Message From Server:
Jack,Sparrow,32130,Accounting,30000
Jack,Sparrow,07801,Movies,40000

(1) Add record
(2) Search by Name
(3) Search by Zip Code
(4) Search by Salary
(5) Terminate
Select an option [1,2,3,4 or 5]: 2

Enter first name: Johnny
Enter last name: Cash
Message From Server:
No record found!!

(1) Add record
(2) Search by Name
(3) Search by Zip Code
(4) Search by Salary
(5) Terminate
Select an option [1,2,3,4 or 5]: 3

Enter Zip Code: 37128
Message From Server:
Pablo,Picasso,37128,Arts,65000
Charles,Babbage,37128,Computer Science,55000

(1) Add record
(2) Search by Name
(3) Search by Zip Code
(4) Search by Salary
(5) Terminate
Select an option [1,2,3,4 or 5]: 4

Enter Salary: 40000
Enter Comparision Type ['>','<','==','>=','<=']: >=
Message From Server:
```

```
Pablo,Picasso,37128,Arts,65000
Jack,Sparrow,07801,Movies,40000
Jackie,Chan,12345,Martial Arts,50000
Charles,Babbage,37128,Computer Science,55000

(1) Add record
(2) Search by Name
(3) Search by Zip Code
(4) Search by Salary
(5) Terminate
Select an option [1,2,3,4 or 5]: 4

Enter Salary: 40000
Enter Comparision Type ['>','<','==','>=','<=']: <
Message From Server:
Jack,Sparrow,32130,Accounting,30000

(1) Add record
(2) Search by Name
(3) Search by Zip Code
(4) Search by Salary
(5) Terminate
Select an option [1,2,3,4 or 5]: 5
Message From Server:
Connection Closed!!
```

5 Example Server Output

```
$./server 3240
Connected to (localhost, 52638)
(localhost, 52638) disconnected
```

6 Submission Instructions

1. You need to submit the following files.

- **a.** `client.c`: You will write the client code here.
- **b.** `server.c`: You will write the server code here.
- **c.** `other files`: It includes all the other header file or `C` files that you used for this project.(Example: `csapp.h and csapp.c`)

d. `Makefile`: Your makefile should compile both client and server program. Also, `make clean` command should clear all the intermediate object files and the excutables files.

e. `records.csv`: It will include the firstname, lastname, zip code, depeartment, and salary of employees.

2. Do not zip the submission.

3. Provide adequate comments on your c code.

4. If I need any clarification regarding your coding, you will need to come at my office hours to demonstrate and/or run the code.

5. Finally, after you are done, upload the requested files in **"Project3"** dropbox in D2L.

7 Grading Rubrics

1. [-10] If add record is not working correctly.

2. [-10] If search by name is not working correctly. [-5] if fails to display multiple matching records with same full name.

3. [-10] If search by zip code is not working correctly. [-5] if fails to display multiple matching records.

4. [-5] If search by salary is not working correctly. [-3] for each failed comparision type test cases.

5. [-5] If server terminates after clients closes the connection(Server can only be terminated forcefully using ctrl+C key).

6. [-5] If comments are missing and bad coding style.

7. [-10] If makefile is missing or does not work (make sure you double check).

8. [-5] If any required libary files are missing (such as csapp.c and csapp.h).

9. [-100] If server.c or client.c missing or does not compile

10. [-2] If scanf is used.

11. [-5] One day late penalty.

12. [-10] Two days late penalty. Submission will not be accepted after two days.