CS 3733.001-002

15-pt part of the Final Exam

(posed/submitted like an assignment under 30% FINAL EXAMS … on Blackboard)

You will take (or already took) the 85-point part of the final exam online on Blackboard with Proctorio. The 15pt part of the final will be this question, which is designed like a very simple assignment so that you can better understand and use *socket programming* and *signal waiting* mechanisms by actually implementing and testing them.

Basically, as described below, you are asked to develop/implement two programs (client and server programs) in **C using TCP (10pt)**. Additionally, you will implement how to wait for SIGINT in the beginning of the client (**5pt**). After the user presses Ctrl-C, the client continues as a regular client and interacts with the server.

Test your programs on our Linux machines and then submit them through BB Learn like an assignment **under 30% FINAL EXAMS… before the due date posted on Blackboard.** See the last page of this document for more directions.

No late submission will be accepted!

Get the sample TCP client and server programs in C from **http://www.cs.utsa.edu/~korkmaz/teaching/cs3733** and modify them! Follow the link "socket programming examples in C and Java" under Online Materials.

You can directly get the TCP sample programs in C from
http://www.cs.utsa.edu/~korkmaz/teaching/resources-os-ug/tk-socket-examples/SOCKET/C/capitalize-tcp/

*Note: if you are asked to enter username and passwd, simply type **cs3733** for both…*

Here is the description of the client and server programs that you need to develop in C using TCP:

Suppose we have a simple employee database, where the server keeps employees' info in an array of

```
struct employee_info {
      char ID[7];          char name[10];        double salary;
}
```

To simplify the tasks, the server creates a static array of 10 employees with arbitrary values that you can manually put for ID, name, and salary in the server program as follows

```
struct employee_info DB[10] = { {“abc123”,”Turgay”, 100000.0}, …};
```

Then the server waits for clients to connect. When a client connects to the server, the server creates a child process. The server (parent process part) goes back and waits for other clients while the server (child process part) starts interacting with client.

The client process can send two types of messages: “GETSALARY ID” or “STOP” . If the server (child process part) gets “GETSALARY ID” then it looks up the DB array, and sends the name and salary of the queried employee if the ID is in DB array. Otherwise (ID is not in DB array), send an error msg. You need to specify the format of that reply messages! The server (child process part) then waits for the next query until it gets “STOP” msg. In that case, both the client and the server (child process part) close sockets and terminate. **Client-server part is 10pt.**

In addition, you should code and show how the client process waits for a signal SIGINT using **sigsuspend()** in the beginning of the client program. Remember that you can generate SIGINT by pressing Ctrl-C. This signal waiting part is **5pt** and will be in the beginning of the client program. After that, the client will interact with the server as described above.

Here are the details of the client-server interaction:

server_host> server server_port	**client_host > client server_host serve_port**
Create a static array of struct employee_info with 10 employee (use arbitrary input make sure ID’s are unique) !	Print a message that client is waiting for SIGINT Ctrl-C … wait for Ctrl-C using **sigsuspend()** … Print a message that Ctrl-C is pressed…
The **server** always waits for TCP connection requests in an infinite loop.	
Upon receiving a TCP connection request, the server (parent process) creates a child process, which serves the client. The parent process will continue to wait for another client.	The **client** will first establish a TCP connection with the server.
Child server process Wait for a request from the client If the server gets “GETSALARY ID”, Check if the given ID is in the array If so, create a reply message containing employee name and salary, and send it to the client If not, send an error message If the server gets “STOP”, close connection and quit.	The client gets an input from the user: 1. Get SALARY for an employee 2. Stop If the user enters option 1, get ID from the user and send “GETSALARY ID” request to the server. wait for a reply and print it. If the user enters option 2, sent “STOP” request to the server. close connection and quit.

Notes:

- **Include extra printf statements** to print what messages are sent or received at the server and client sites so that we can easily follow your protocol.
- Make sure you close the socket descriptors that are not needed in the parent and child part of the server.

- You don't need to handle every possible error, but make sure you check what each system call returns etc. and accordingly take minimum action (e.g., simply quit if there is an error)

WHAT TO Submit:

Put all your work under a directory called **yourabc123-final**.

As in the provided sample TCP programs in C, make sure you have your source codes as **server.c**, **client.c** , **Makefile**. Compile and test them on our fox0X. Simply copy/paste the outputs of your client and server programs into **out-client.txt** and **out-server.txt** for at least 3 queries.

Then

1. Remove executables, just have the above 5 files Makefile *.txt *.c
2. Zip yourabc123-final directory/folder
3. Make sure yourabc123-final.zip contain the above files Makefile *.txt *.c
4. Submit it through BB Learn (30% FINAL EXAMS… /Implement Socket and Signal Programs for Final).