Assignment 7 - Part 1
March 10, 2023
Multi-Client Chat server

In this assignment you need to implement a client-server chat application using TCP sockets. There will be a single server and multiple clients communicating with the server. The server process can handle at most 10 concurrent connections. Each client process will open a new connection with the server and add the client socket id to its fd_set(). **Use *select* to handle multiple client requests.**

Task : Simple Chat with Broadcast Functionality.

1. **Connection Establishment** : The client should send the connection request to the server. The server should reply to the client with appropriate messages if a connection can be established or not. [10]

    Successful : If the connection can be established successfully, then generate appropriate identifiers for the client and store them at the server. Identifiers include - Unique Id (5 Digit Random Number). After the connection is established, send the client the Unique ID with a welcome message.

    Unsuccessful : If the number of clients connected are already 5 then no further client is allowed to connect and the server should inform the client that “Connection Limit Exceeded !!”.

    Client details table: Keep the details in memory at server side (Use proper data structure for simplicity)

2. **Chatting** : Client should send a query message to the server asking the details of online clients using “/active” call. Server should send the details of all the online clients [**each with a unique key**]. After receiving details, the client can chat with another client of choice by using its unique id, using the “/send” command. (Note that this is a one to one communication). [20]

    Message info table: Create another table which stores message details.

***NOTE**: There can be a situation when client A gets the list of online clients and before it can send any message to client B, client B goes offline. The sender in this case should be notified that the client is now disconnected and that the message should be discarded.*

3. **Broadcast** : A client should be able to send a broadcast message by typing “/broadcast”. The message should be delivered to all clients connected to the server. [20]

4. **Connection Termination** : In order to disconnect, the client should send a “/quit” message to the server. The server should notify all other clients with the details of the client which is going to disconnect. Then terminate the client process. [10]

***NOTE**: It can so happen that a client can hang up abruptly in the process of chatting. It could be because of some interruptions like Ctrl+C, Ctrl+Z. You are supposed to handle these two specific interrupts and make sure the client terminates properly by executing the '/quit' functionality.*

Functionalities to be included

1. **/active** : To display all the available active clients that are connected to the server.
2. **/send <dest client id> <Message>** : To send a message to the client corresponding to its unique id.
3. **/broadcast <Message>** : Message should be broadcasted to all the active clients.
4. **/quit** : The client will be removed from the server.

Things to remember:

Server Side

1. Use select() for the implementation.
2. Use send() and recv() system calls. It will make your life easier.
3. If client B wants to send a message to client A, B won't be able to send to A's socket directly. Instead use a message details table at the server side. Client B will send a message to server and then server will pass on the message to client A.
4. Server will check the message details table. If it finds an entry of a particular client with dest_id as its sock_id, then send the message to the client. Remove that entry.
5. Log messages on server (server's terminal).

Client Side

1. Reading from the standard input and writing to the server (send() system call) and reading from the server (recv() system call) will be handled by different processes (use *fork* here, we believe you know the reason).

Submission Guidelines:

1. Give proper comments explaining how you have implemented a part of code.
2. Submit file as RollNo_A7_P1_server.c for server, and RollNo_A7_P1_client.c for client.