Assignment III

Network Programming

This assignment is on network programming. It is accompanied by a pair of programs server.c and client.c.
The server and client can be executed in different systems (but for convenience you can have them in same system with each client process associated with a standalone terminal).
Before attempting, you must read through the assignment, and in particular, all the use cases and implementational details described within the program.

References:
1. Unix Network Programming, W. Richard Stevens, PHI.
2. Online man pages
3. Sockets Tutorial : http://www.linuxhowtos.org/C_C++/socket.htm
4. Client Server Model Architecture :
   http://www.tutorialspoint.com/unix_sockets/client_server_model.htm

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 5 concurrent connections. Each client process will open a new connection with the server and to handle each client request, the server will create a new child PROCESS.

API's allowed to use : socket, bind, listen, connect, accept, close etc.
Programming Language : C(Only) for implementation
Concurrency and Synchronization : POSIX threads and mutex, Shared Memory, binary Semaphore.

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

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

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

2. **Data Transfer Phase** : Client should send a query message to server asking the details of available clients. Server should send the details of all the online clients. After receiving details, client can transfer messages to any other client of choice by using its unique id. (Note that this is a one to one communication).
Design a buffer queue at server using shared memory architecture with a separate process handling the message delivery. Use appropriate locking mechanism to handle this critical section where messages to be delivered have to be written and processed for delivery.
There can be a situation when a 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 client is now disconnected and that message should be discarded.

3. **Connection Termination** : In order to disconnect, the client should send an EXIT message to the server. The server should notify all other clients with the details of client which is going to disconnect. Then terminate the client process.

4. **Broadcast:** A client should be able to send a broadcast message by typing "+broadcast <msg_string>". The <msg_string> should be delivered to all clients connected to the server.

Important :
This is mostly a design problem and a large fraction of marks are allocated for innovating thinking and most efficient way to implement the problem. Try to come up with all possible boundary cases and appropriate test cases.

Deliverables :
1. All codes should be in a tar.gz format with name starting with your roll no. **<Roll_NO.>tar.gz**
2. A proper design document enlisting a proper explanation of the approach you used to solve the problem and also the boundary cases you have identified and handled.

Mark Distribution :

1. Design Document understanding of problem statement - 10 Marks
2. Identification of boundary cases, test cases - 20 Marks
3. Implementation of problem statement - 50 Marks
4. Viva - 10 Marks
5. Coding style and Code Modularity - 10 Marks

Please do not copy code from internet or from your classmates. If found copied then straight away zero will be given and strict action will be taken.

Come up with new ideas and spend time designing and implementing problem.

Happy Coding !!