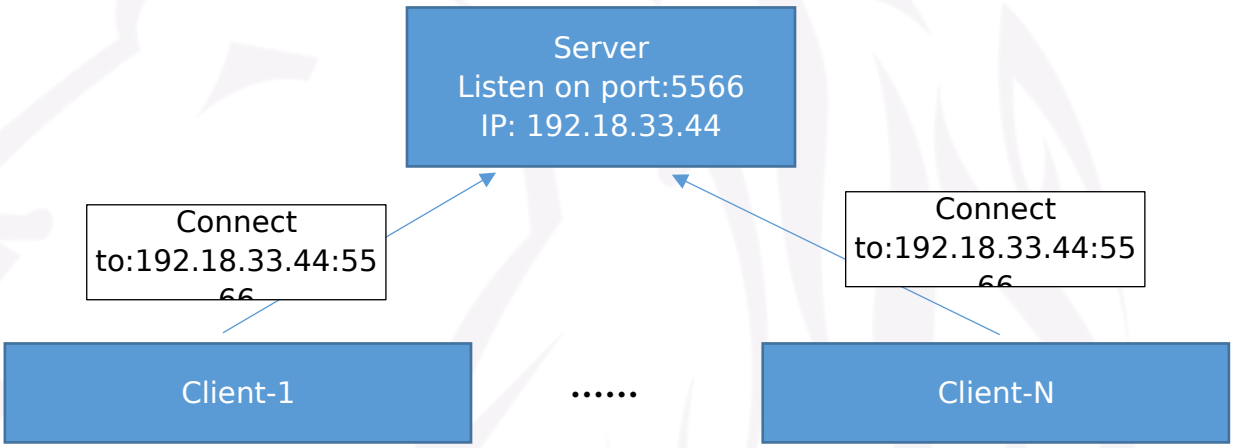
CS-232: Operating Systems

Assignment 02

Duration: 1.5 Weeks (Please submit it by Sunday 1st of April before 1159 pm)

This assignment is a practice in socket programming, threads, and synchronization. You are supposed to implement a rudimentary chatting application. The application will run on a client/server model which means that you'll build two different programs: a server and a client. For simplicity we'd say that one instance of the server will be running and multiple clients can connect to the server and chat to each other using this server program as an intermediary. The server and clients may run on the same PC or on different ones.



TO DO:

- You basically run the server passing it a port number:
  ./server 5566
  The server program creates a socket and starts listening on the port passed to it at command line.

- You run a client passing it the server programs computer IP, the port it's listening on, and a name for your client:
  ./client 192.18.33.44 5566 client1
  The client should make a socket and try to make a connection to server program using the server's PC's IP and the port number the server is listening on. Once the connection is established, the client should send its identifier ("client1" in this case) to the server. In case there is no server running, the client should terminate with an appropriate message.

- The server program should accept connections from multiple clients. For each connection established the server should store the client's identifier and keep a record of which port is being used for communication with this client.

- In such applications, usually, a sever creates a separate thread for each connection with the client.

- If a new connection request comes from a a client with a client identifier which is already in use, the server should inform the client with an appropriate error message and close that connection.

- The client can send "commands" to the server. Each command starts with a '/' character. When the server program receives a command from a client, it should take the appropriate action and send a response to the client. Few commands:

  /list - if the client sends a "/list" command to the server, it means the client is asking for a list of all the connected clients. Upon receiving this command, the server should put the names of all the clients connected to itself in string and send it back to the client. The client shall receive this string and display all these names to the user.

  /msg – users can type this command to send messages to other clients. The general syntax is "/msg clientname message". i.e. if the user at client-1 types "/msg client-2 hi there", the server should receive this message from client-1 and send it to client-2. The user at client-2 should receive the message "hi there" along with the information that it came from client-1. You should do proper error handling for this command at server and client end i.e. missing destination (client) identifier or incorrect destination identifier should generate appropriate response.

  /quit – users can type this command to disconnect from the server and quit the client. Upon reception of this command from a client, the server should close the connection with that client and terminate that particular thread.

  You'll need socket programming, multi-threading, dynamic memory allocation/de-allocation, and synchronization for this assignment.

  Some resources can be found on LMS in the Books folder for the lab section. The chapters on threading and socket programming in the book (Advanced programing in unix environment) APUE 3rd edition should help as well as the links in the .txt file in the same folder. The rest you can find yourself.

  You'll also need string manipulation functions in C: https://en.wikibooks.org/wiki/C_Programming/String_manipulation

IMPORTANT: This assignment may take more time than the previous one. So kindly start early !!

You've got two weekends, so don't expect any extensions on this one.

Submission guidelines:

1. You are to work in groups of two for this assignment. Both members get the same grade. Both should submit the assignment.
2. You should submit a single tar-zipped file whose name should be your ID.
3. When extracted it should extract to a single directory whose name should be your ID.
4. In that directory there should be only files for the source code of each part and a makefile to build them.
5. Write clean code, comment it, follow good coding practices like, giving sensible names to variables, de-allocate all resources after their use, etc.

Make sure that you DO NOT create any memory leaks, dangling pointers, zombie threads, etc.