Project 1: "Text ChatRoullette"

DUE DATE: February 18, 2014 (11:59am)

Description

In this assignment you will be asked to implement an IRC like chat server and a chat client that enables a client to chat with "random" people and supports multiple such sessions. The project will be completed in either C/C++ and will require the use of Unix Socket programming. This assignment is worth - **100 points.**

Submission

As a part of the submission you need to provide the following:

1. All the code for the chat client and the chat server (.c and .h files)
2. A *makefile* that compiles the client and the server code (Tutorial on writing a makefile can be found here: http://mrbook.org/tutorials/make/)
3. A ReadMe (txt) file that spells out exactly how to compile and run the client and the server

Combine (via zip or tar) everything up into a single archive file named "your-wpi-login_project1.zip".

Submit your document electronically via Instruct Assist (https://cerebro.cs.wpi.edu/cs513) by 11:59am on the day the assignment is due. Make sure you choose "Project 1" under project drop-down in IA before uploading the zip file.

Requirements

Chat Server: The goal here is to implement a chat server called "Text ChatRoulette Server (TRS)", which is similar to IRC chat servers. TRS should have the following functionalities:

- Unlike a typical IRC chat server, TRS will maintain a single logical queue of all possible clients who want to chat at any given time.
- When a client expresses the desire to chat, the server automatically creates a *channel* and puts together **two** random clients. The two clients chatting with one another on a channel will be collectively referred to as *chatting partners*.
- The TRS should facilitate the chat between the two clients.
- The TRS should be capable of managing multiple chat channels simultaneously, and all the data being transferred between them.
- The TRS should have an *admin* entity, which can issue commands directly to it and control the chatting process.

Chat Client: The client is a program running on a user machine that connects to the TRS to chat with random people. The chat client should have the following functionalities:

- The chat client should be able to connect to the chat server and express its interest in chatting.
- The clients are identified by nicknames, which may not be unique.
- The chat client should be able to initiate and quit a chat whenever it pleases
- The chat client should be able to transfer images and other files (of up to size 100MB) to their chatting partners.
- There is a no UI requirement for the client.
- The client should be able to *report* its chatting partner if it finds it to be misbehaving in anyway.
- If the TRS dies or shuts down or the chatting partner quits, the clients should stop the chatting gracefully, without throwing an exception.

Commands (**minimal set to be supported**):
Between TRS and Chat Client.

- CONNECT: Issued by the client, the command is accepted by TRS at a specific TCP port. This expresses the intent of the client to be part of the chat queue. *Remember to uniquely identify the clients, as client nicknames may not be unique.*
- ACKN: Issued by the TRS to acknowledge the addition of the client to the queue, or if the queue is full then inform the client that the CONNECT failed.
- CHAT: Issued by the client, this command asks the TRS to connect the client with another random client in a common chat channel.
- IN_SESSION: Issued by the TRS to the clients informing them that they are talking to each other on a specific channel.
- QUIT: Issued by the client, this tells the TRS that the client wants to quit its current channel. TRS puts both the chatting partners back on the chat queue, and waits for them to issue a separate CHAT command. The channel the clients were a part of is destroyed.
- TRANSFER: Issued by the client, this tells the TRS that the client wants to transfer the file to another client it is chatting with on a channel.
- FLAG: Issued by the client, this tells the TRS that a client wants to report its current chatting partner is misbehaving is some form.
- HELP: Issued by the client, this asks the TRS to list all the commands it accepts from clients.

Between TRS and the admin.

- STATS: Issued by the admin to the TRS which results in TRS providing a file with the following information: (1) number of clients in the chat queue, (2) number of clients chatting currently, (3) data usage for each channel being used, (4) total number of users flagged chatting partners and their names and their status (i.e., are they in the queue or in a chat).
- THROWOUT: Issued by the admin to the TRS, which results in the TRS throwing out a client from a channel, and destroying the channel they were using.

- BLOCK: Issued by the admin to the TRS, to prevent a client from being randomly picked into a channel. This could be a punishment to suspend chat privileges for previous misbehaviors.
- UNBLOCK: The opposite of the previous command, which allows a previously blocked client to be randomly picked for future chatting.
- START: Starts the TRS and allows clients to connect to it for chatting.
- END: Destroys all active channels, empties the chat queue freeing all resources. This has to be done gracefully, which means informing all clients that their chat session is over.

Notes:

- The parameters for the commands above have deliberately not been specified. You need to provide appropriate parameters for the commands so that you can ensure all the functionality described above is enabled.
- The commands described above are not necessarily the full set of commands you may need. You might have to introduce additional commands for example for catching errors, faults, exceptions, acknowledgements etc. For example, if a client tries to send a file > 100MB, an error needs to occur, and the sending client informed.
- Make sure that client does not have to do anything about identifying the IP address or port number of its chatting partner. The TRS takes care of it all. The only thing the client should know is the hostname of the TRS to connect to.
- You must chose a TRS port number larger that 1023 (to be safe, choose a server port number larger than 5000).
- When you connect to the TRS make sure you use the hostname to connect and not the IP address. Please use the command *hostname* to find out where you are connected.
- In writing your code, make sure to check for an error return from your system calls or method invocations, and display an appropriate message. In C this means checking and handling error return codes from your system calls.
- If you need to kill a background process after you have started it, you can use the UNIX kill command. Use the UNIX *ps* command to find the process id of your server
- Make sure you perform error handling and close every socket that you use in your program.
- If you abort your program, the socket may still hang around and the next time you try and bind a new socket to the port ID you previously used (but never closed), you may get an error.
- On UNIX systems, you can run the command *netstat* to see which port numbers are currently assigned.

Grading

Grade breakup (%) assuming all documents are present:

1. Chat Server = 50%

    a) Implementing the list of commands (including chat queue) – 25%
    b) Handling multiple chat channels simultaneously – 15%
    c) Gracefully handling chat client and TRS shutdowns and errors – 10%
2. Chat Client = 40%
    a) Implement basic command list for the client – 20%
    b) Chat with other clients and send files to them – 10%
    c) Gracefully handling chat client and TRS shutdowns and errors – 10%
3. makefile = 5%
4. README = 5%

The grade will be a ZERO if:

1. If the code does not compile or gives a run-time error
2. If README with detailed instructions on compiling the running the code is not present
3. If makefile is erroneous and the README does not provide a way tell us how to compile and run the code

Teamwork

All submissions have to done in teams or 2 or 3 (preferable two). You can be in existing teams in the class or create new ones as needed. Please inform the instructor of your team members.

Teams will have to demo their project in the end. More details on the demo is forthcoming.

Resource:

- Many resources are available online for Network Programming using C/C++.
- One of the best is **Beej's Guide to Network Programming: Using Internet Sockets** http://beej.us/guide/bgnet/