Homework 1: A Chat Room Service

50 points (plus 5 BONUS points)
BONUS part of HW 1 needs to be submitted individually
Due: February 1st, 2018, at 11:59pm

1 Overview

The objective of this assignment is to build a **Chat Rooms Service**. The system can have several Chat rooms, and clients join chat rooms by issuing `JOIN` commands. Clients can also create, list and delete chat rooms remotely by issuing `CREATE`, `LIST` and `DELETE` commands. All communication between chat rooms and chat clients will be using a socket interface.

1.1 Chat Room Server

The program/process implementing the **Chat Room Server** (call it `crsd`) would do the following steps in a loop, in any order:

- Listen on a well-known port to accept `CREATE`, `DELETE`, `LIST` or `JOIN` requests from Chat clients.
- For a `CREATE <chatroom name>` request, check whether the given chatroom exists already. If not, create a new master socket (careful about picking the port number!) Create an entry for the new chat room in the local database, and store the name and the port number of the new chat room. Inform the client about the result of the command.
- For a `JOIN <chatroom name>` request, check whether the chat room exists. If it does, return the port number of the master socket of that chat room and the current number members in the chat room. The client will then connect to the chat room through that port.
- For a `LIST` request, return the names of all chatrooms.
- For a `DELETE <chatroom name>` request, check whether the given chatroom exists. If it does, send the warning message `"chat room being deleted"` to all connected clients before terminating their connections, closing the master socket, and deleting the entry. Inform the client about the result.

- Incoming chat messages are handled on slave sockets that are derived from the chat-room specific master socket. Whenever a chat message comes in, forward that message to all clients that are part of the chat room.
- Clients leave the chat room by (unceremoniously) terminating the connection to the server. It is up to the server to handle this and manage the chat room membership accordingly.

1.2 Chat Room Client

The program/process implementing the **Chat Room Client** (call it `crc`) would issue requests to and exchange messages with the chat room server. The client program takes the host name and the port number of the chat room server as first and second command line argument, respectively. The client program provides a simple command line interface of your design, which reads both commands to create, delete, or join chat rooms, and messages to the currently joined chat room.
The chat room client would create, delete, and join chat rooms in the following way:

- To create a chat room, connect to the well-known port of the chat server, and send a `CREATE <chatroom name>` message. Display the reply and close the connection.
- To delete a chat room, connect to the well-known port of the chat server, and send a `DELETE <chatroom name>` message. Display the reply and close the connection.
- To list all chat rooms, connect to the well-known port of the chat server, and send a `LIST` message. Display the reply and close the connection.
- To join a chat room, connect to the well-known port of the chat server, and send a `JOIN <chatroom name>` message. Read the information with the port number of the chat room and the current number of members. Display the information.
- If the `JOIN` operation was successful, connect to the port returned by the server, and begin exchanging messages.
- Leave the chat room by terminating the connection.

Note: There is a little complication, as the client program has to read input both from the command line and from the connection to the chat room. You have to find a solution to handle this. (One way is to use separate threads to handle the two input sources, or you can simply use `select()` to handle them in a single-thread solution.)

2 What to Hand In

The running system will consist of the chat room server (`crsd`) and the chat room client (`crc`). As platform, you should be using linux workstations, and develop the program in C/C++.

2.1 Design

Before you start hacking away, plot down a design document. The result should be a system level design document, which you hand in along with the source code. Do not get carried away with it, but make sure it convinces the reader that you know how to attack the problem. List and describe the components of the system: Chat Client Program, Chat Room Server Program, and their interaction. In particular, describe how you implement the server program (iterative, multi-threaded using thread, multi-threaded using processes, multi-threaded using `select()`, others.)

2.2 Source code

Hand in the source code, comprising of a:

- Makefile
- a file `crsd.c` or `crsd.C`
- a file `crc.c` or `crc.C`
- README file that explains how to compile the program and run it

The code should be easy to read (read: well-commented!). The instructor reserves the right to deduct points for code that he/she considers undecipherable.
Points will be give as follows: 5 pts (design document), 5 pts (compilation), 40 pts (5 test cases - details on the exact inputs and outputs will be provided in class)

3 BONUS

Read Section 1.5 of the textbook. Using the material from this section, formulate 5 multiple choice questions, where each question can have 0-5 correct answers. Give the correct answer for each question and a brief explanation for it. If no answer is correct, add "No correct answer" as an option. Each question correctly formulated, answered and explained is worth 1 point.
An example is the following

Questions 1: Section 1.5 of the textbook is about the following main challenges of distributed systems:

- Heterogeneity
- Security
- Concurrency
- Scalability
- No correct answer

Answer: A, B, C
Explanation: Because these are the subsections of the chapter.