Multiuser Chatting Server Programming

CSE351 Computer Networks (Spring 2018)

Object

The goal of this project is to implement multiuser chatting server. By using this program, multiple users connect to multiuser chatting room and communicate each other simultaneously. You have to implement both of chatting client and server. You have to develop your own idea using what you did in socket programming pre-tutorial and webserver programming. Your program should be implemented in C++ or C language.

How it works (+ 80 points)

1. Users execute program with chatting server address, room number and nickname.

   ex)
   ./client 192.168.0.2:5000 10 Bob

2. Chatting server sends successful connection message to the user, and sends 'new user' message to the other users in the room.

   ex)
   message to the user → Hello Bob! This is room #10
   message to the other users in the room → Bob joined room 10

3. Sender input message data with receivers' name and message, and then the message should be sent to server by socket.

   - The message form should be like this.

   Receiver1, Receiver2, …, ReceiverN : Hello World!

   ex1) Steven, James, Sam : Hello World!

   ex2) All : Hello World!
   → **This message will be sent to all users in the server**

4. When the server receives message from sender, server parse the message to check whether receivers are in the room which sender connects to. Then server sends the message to each user

   check) if server cannot find receiver's name because of typo, sender should this message to the sender
   → There is no such user : Steve, Jame

5. When receiver receives message, it should show the message with sender's nickname

   ex)
   Bob : Hello World!

Hint

1. Each user open a connection to the server
2. Server should keep track of socket descriptor and nickname of each user (You can use data structure, you can use file system, and so on…)
3. To make it concurrent, implement you program with multi-threading
4. It is good idea to implement your own protocol for the communication between server and client

Additional Points

If you implement additional functionality, you can get extra points.

1. Changing room (+ 20 points)

   description)
   If user sends join message ("/join 10"), server should change room (to room number 10)

2. Leave message (+ 20 points)

   description)
   If user sends quit message ("/quit"), server should send leave message ("**Good bye Bob**"), and close the connection

   The other users in the room should receive leaving message like

   Bob is disconnected from room #10

3. Listing user (+ 20 points)

   description)
   When user sends list message (“/list”), server should send list of users in the room

   /list
   > This is list of users in room #10
   1. Steven
   2. James
   3. Sam

4. Solving nickname duplication problem (+ 10 points)

   description)
   There can be duplicate nickname. To solve this problem, check user list when a user initially connects to server, if there is duplicate nickname, then change the user name automatically like this

   James -> James-2

   Server should change connection message
   “Hello Bob, This is room #10” → “Hello Bob-2, This is room #10”

5. Message reservation (+ 20 points)

   description)
   Sender can set wating time for each message. For example,
   If sender sends message like this,

   Alice#60, Bob#30 : Hi

   Then, server should sleep 60 seconds for the message to Alice, and 30 seconds for the message to Bob.

6. Connection health checking (+ 30 points)

   description)
   Server sends hidden messages to each users for checking whether connection of the user is alive or not. Each user (client) responds to the server automatically. If server cannot receive alive message before timeout (3 seconds), server removes the connection and sends leaving message (“**Alice is disconnected from room #10**”) to the other users in the room

Testing and grading

Test your code in Ubuntu, and your code should be written in C++ or C.

Open many terminals, one for server, and the others for clients.

Grading policy)

1) Basic chatting implementation (without concept of room) : 40 points

2) Chatting room implementation : 40 points

3) Room Changing : 20 points

4) Leave Message : 20 points

5) Listing user : 20 points

6) Solving nickname duplication problem : 10 points

7) Message reservation : 20 points

8) Connection health checking : 30 points

Total : 200 points

Submission

- ✓ Submit your codes: jongyunlee@unist.ac.kr
- ✓ Deadline is 15th June
- ✓ The email should have the title in the format of "[CSE351] chat_<name>_<student id>"
  - ➢ You can submit only one file (in *.zip or *.tar.gz only)
  - ➢ Make the directory name "chat_<student id>"
- ✓ Your file should contains server.cpp, client.cpp
- ✓ At the head of server.cpp content, please remark what you implemented shortly.
  ex) /* 1. Basic chatting implementation, 2. Chatting room implementation, 3. Room changing… */