Project 4211: Gopher Chat Room

Student: Ngan Nguyen

X500: nguy4068

**1. Required features**

**\* Account Registration:**

- User can register for the account using the command: REGISTER name password

- Both name and password should have length within 4 to 8 characters, all characters should be alphanumeric

- Invalid format for name and password will be caught at client side, and message will be printed out message to tell the invalid parts

- Registration for pre-existed account will be caught on server side, a message will be printed out to let user know

- It’s ok to leave registration command inside the script and rerun it, because only a valid login will trigger the connection between client and server

**\* Login:**

- User can login to the chat room using the command: LOGIN name password

- If either name or password does not match the credentials of any existed user found in the storage, message will be printed out for user

- User does not need to register if they have already registered before, however, it's fine to leave them in the scripts, as only a valid login is needed for connection

**\* Send messages:**

- User can send messages to other users using this command: SEND message

- The limitation of message size is 256, if a message size exceeds 256 bytes, it will be aborted on client side

- Message can have any characters

**\* Receive messages:**

- User will receive message from other user in this format: name: message

- Every received messages will be printed out

**\* Send private message:**

- Users can send private message to only their target recipient using this command: SENDA username message

- The limitation of message size is 256, if a message size exceeds 256 bytes, it will be aborted on client side

- Only one target recipient is allowed

- If the target user does not exist, then error will be printed out for client

- The error look like this: “user haha is invalid”

**\* Receive private message:**

- User will receive message from other user in this format: name: message

**\* Send anonymous messages:**

- Users can send anonymous message to other users using this format: SENDA message

- The limitation of message size is 256, if a message size exceeds 256 bytes, it will be aborted on client side

**\* Receive anonymous messages:**

- Users receive anonymous messages in this format: ANONYMOUS: message

**\* Send file public:**

- Users can send file to everyone using this command: SENDF file

- The file size should not exceed 10MB, if it exceeds 10MB, then the sending will be aborted

- User should only send file within the current directory that they run the client program

**\* Send file private:**

- Users can send file to a target receiver using this command: SENDF2 name file

- The file size should not exceed 10MB, if it exceeds 10MB, then the sending will be aborted

- User should only send file within the current directory that they run the client program

**\* Receive file:**

- Users receive file in this format: name: file filename

- Users can open and check the file with exactly the same name with [filename]

- If multiple files and messages are sent concurrently, users will potentially receive the message first, and then the file will come later

**\* List all online users:**

- Users can ask for the list of current online users using this command: LIST

- Users will receive the list of online users in this format:

List of online users:

user1 user2 user3 ...

**\*Logout:**

- Users can disconnect from server using this command: LOGOUT

- The socket file descriptor of the corresponding user will be removed on server side

- The program will terminate after 5 seconds

**\*Reset server:**

- Server can be reset from server side using this command: ./server reset

- Once server is reset, all of the users stored in database will be removed, as well as offline storage for each of them

- A further instruction can be found at the section "How to run program"

**\*Server side monitoring:**

- Every receipts of messages, files from clients will be printed out

- Message from clients will be printed out, file size received from client will also be printed out

- Forwarding of message from server will be printed out

- All these messages will have timestamps preceding it

- Below is an example of the logged message:

Mon Apr 25 04:51:31 2022: Sent file of size 1538658 to mario

- When connection is removed, a message will be printed out in this form: connection with [user] is closed

**2. Add-on Feature:**

**\* Offline chat room:**

- Sending of messages, files, regardless of public, private, or anonymous mode will be saved for existed users if they are valid intended users and they are offline

- As soon as the user login, all of the messages happening during their offline time will be logged out to the screen

- Both messages, and files will be sent to the offline user that just comes back to online

**3. How to run the program:**

**\*Fire up server side:**

- Server is stored in root directory at EC2 instance with IP address: 18.118.113.160

- To login, use the following command: "ssh -i 4211.pem ubuntu@18.118.113.160"

- First you need to copy the server.cpp file and the make file to your desire folder in the EC2 instance

- Then in this folder, create two additional empty files: users.txt, and storage.txt

- You can simply use these following commands:

- **'touch users.txt'**

- **'touch storage.txt'**

- At the room directory, run this following command: **make server**

- After that, run this command: **./server 8001 4068**

- If you use a different EC2 instance, please add the inbound rule for your intended port, and port 4068

- Port 4068 is specifically chosen as the listen socket for upcoming file transfers to avoid blocking of normal message transfers

- So two inbound rules will need to be added, one for your chosen port, and one for 4068

**\*Fire up client side:**

**All examples for client scripts are stored in testcases folder, you can look at it for syntax reference while doing testing**

- You can directly run client code from your local machine

- cd to the download project folder

- Type in this following command: **make client**

- After it, you should be able to find there are 4 sample client folders that I have provided:

- To run the client, cd to one of the client folder, ex: 'cd client1' use this command:

**../../ client 18.118.113.160 8001 4068 user.txt**

- Similarly, if you want to run the client2 folder, cd client2 and then:

**../../ client 18.118.113.160 8001 4068 user.txt**

**-** Notice that the ../../ indicates that the client executable file is outside current working directory 2 levels, you should leave it like that, or manually bring the executable file to your client folder

- 8001 is the port for message channel, 4068 is the port for file channel, if you want, you can change it, but the port should be similar to those you provided for users

- You can include a LOGOUT command in the script, or you can simply sleep for 30 seconds before program exists, or use can use Ctrl+C to terminate the client as well

**4. How to test the add-on features:**

- First run the script of a user to register for account

- Then run the script of another user

- You can use the script that I have provided inside the client1 and client3 folder

- If you use it, then run the user script in client1 first, and wait for it to log out

- Then run the user script in client3, wait for it to finish

- Then run the user script in client1 again

- You should be able to see all of the messages (The total 7 files) be sent to user 1