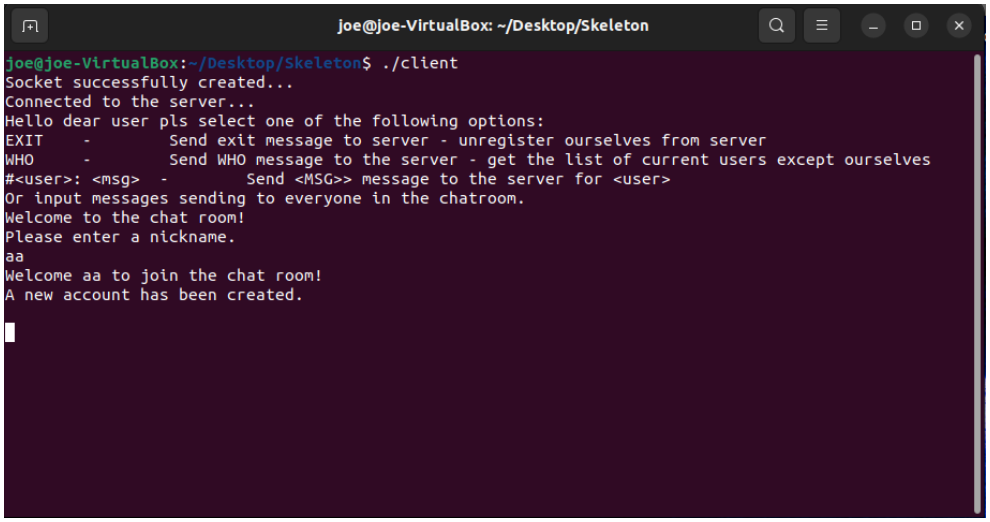
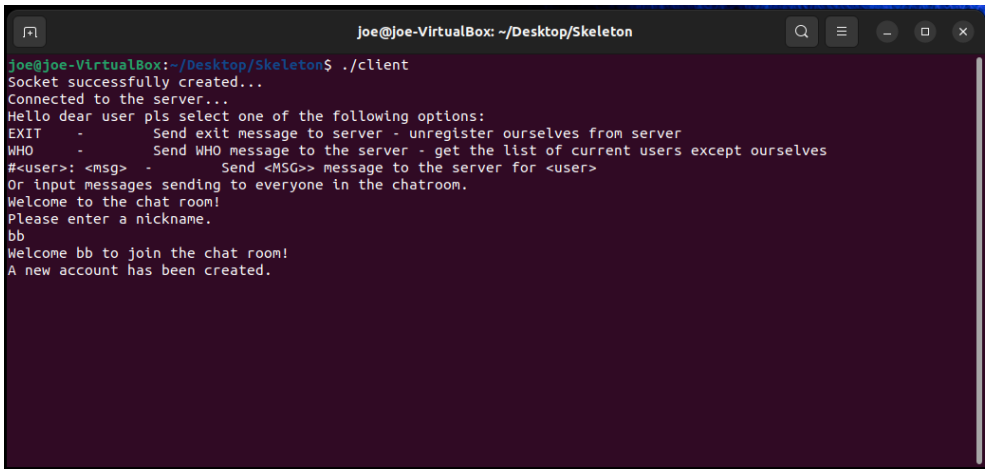
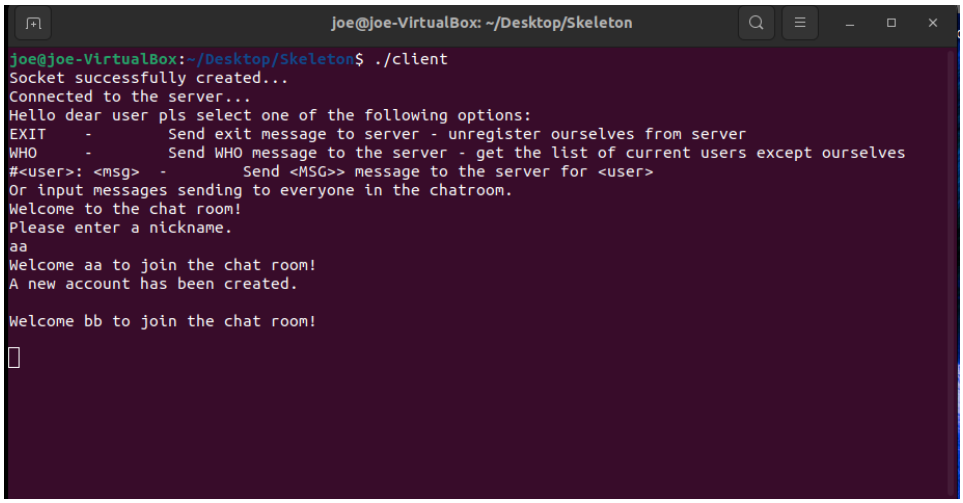
Implementation of Register



In client side, when user start running the client, program will ask for the nickname. According to the requirement, client side need to send the message REGISTERaa.

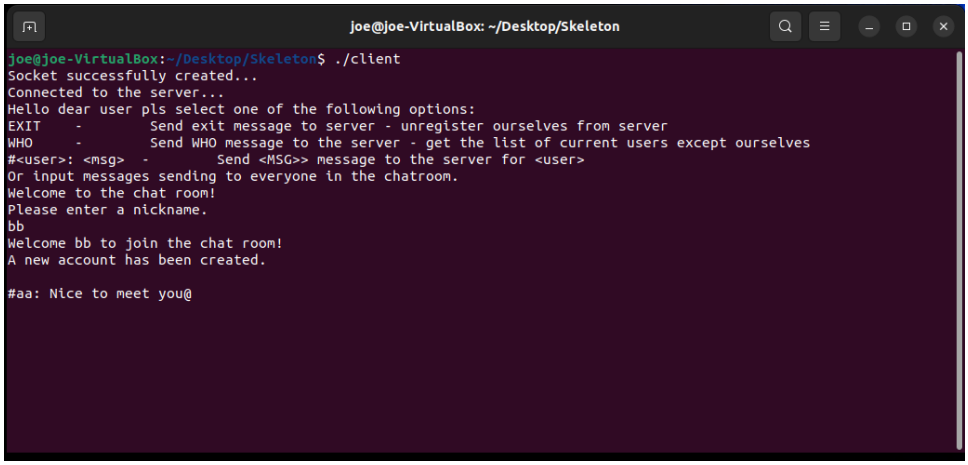
In server side, the server found that aa has not existed, so it is a new user, and we need to construct a new user_info_t to store user information. Then, we also need to create a text file for message box and broadcast to all users and notify aa a new account has been created.

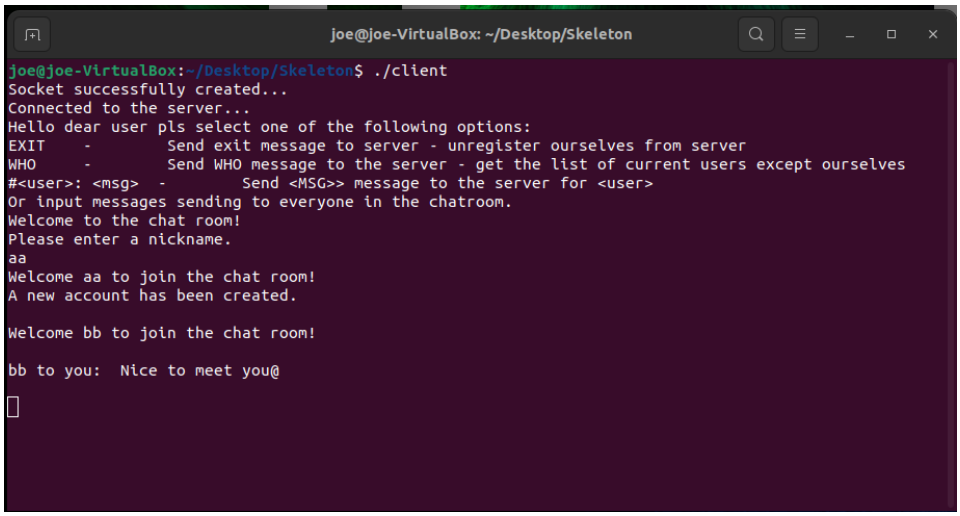




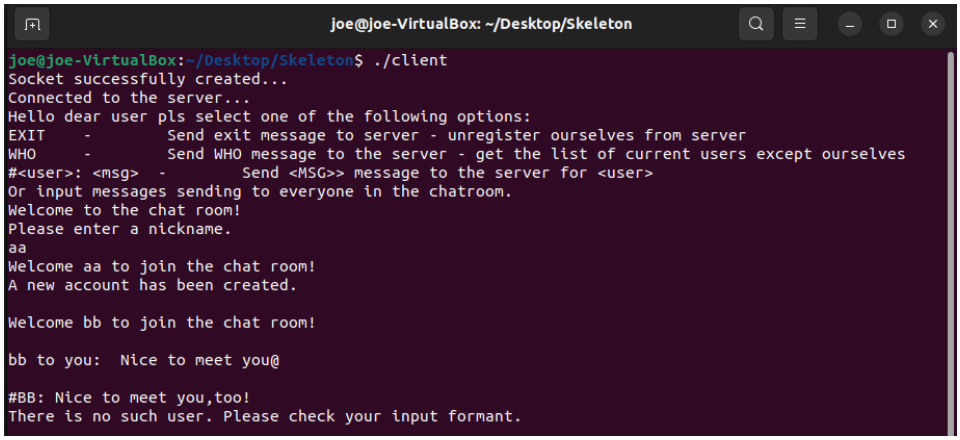
It is the same for bb, while aa will receive the broadcasting message.

Implementation of Direct messages to online users

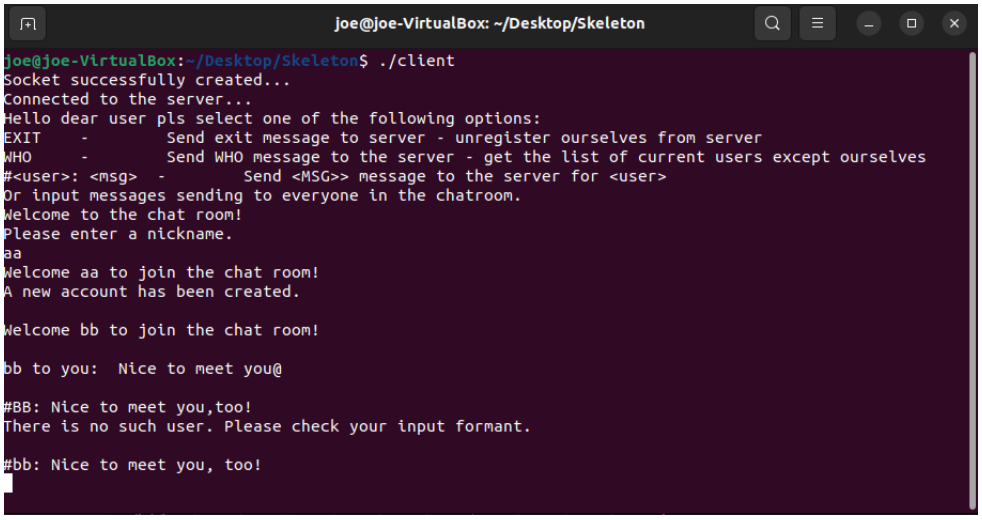


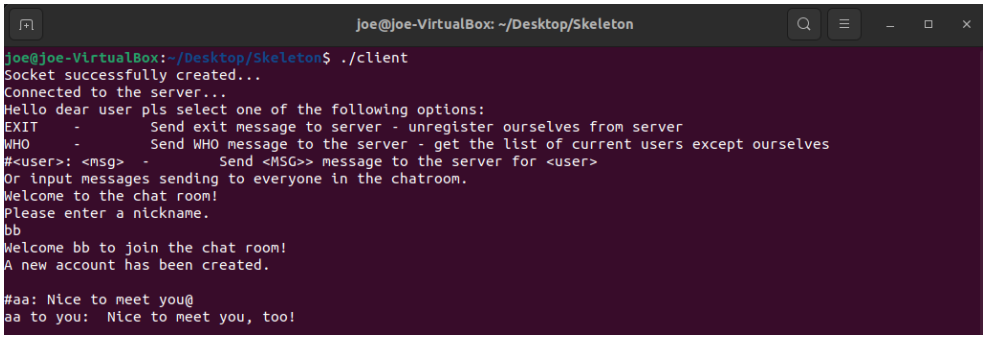


bb sends the message to server, then the server identify the # symbol and know it is direct message. Server side will split the message to send name, destination name, destination socket and message. If aa is found (i.e. the destsock is not -1), and the state is online, server will send to data to aa.



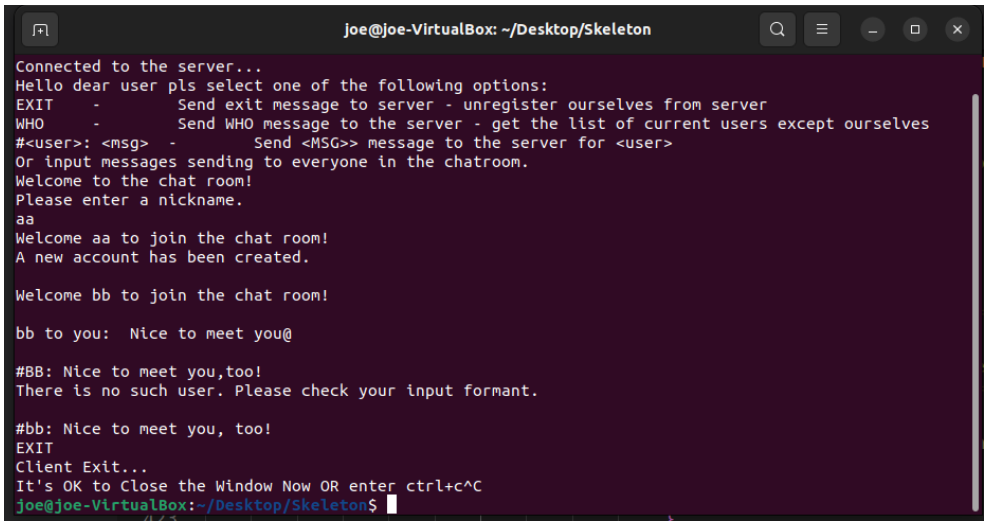
Server side does the same thing, but since BB does not exist (destsock = -1), server send the warning message to aa.





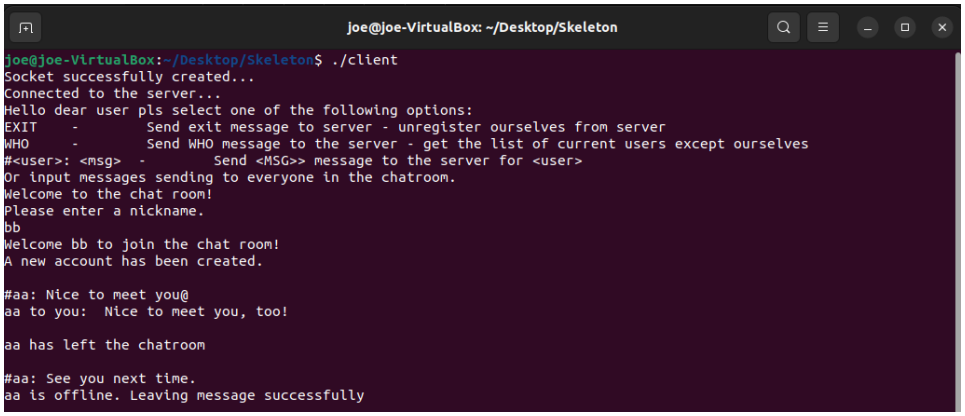
This time aa type the name bb correctly. Server and client side do the same thing as bb to aa.

Implementation of Exit

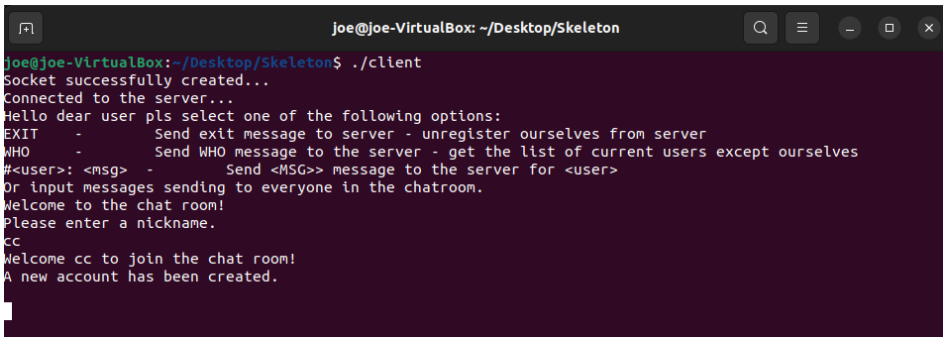


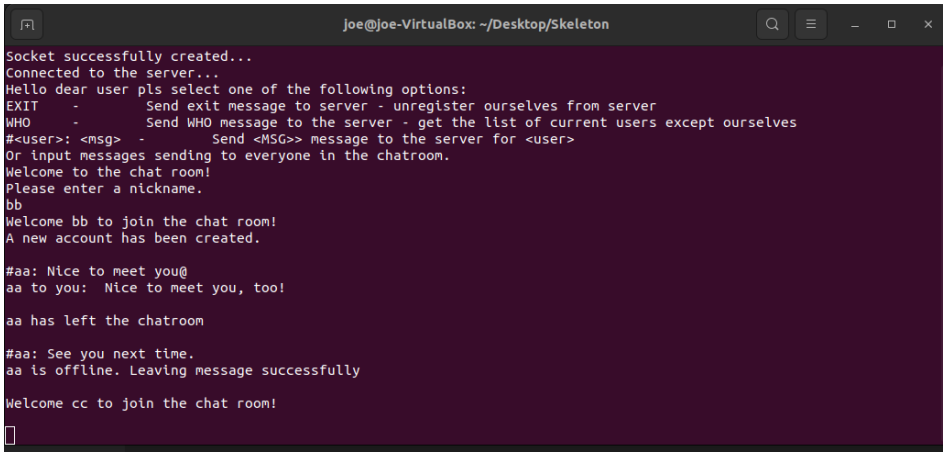
aa exits, the server recognize the “EXIT” keyword. Then, server modifies the state and close the socket.

Implementation of Offline message box

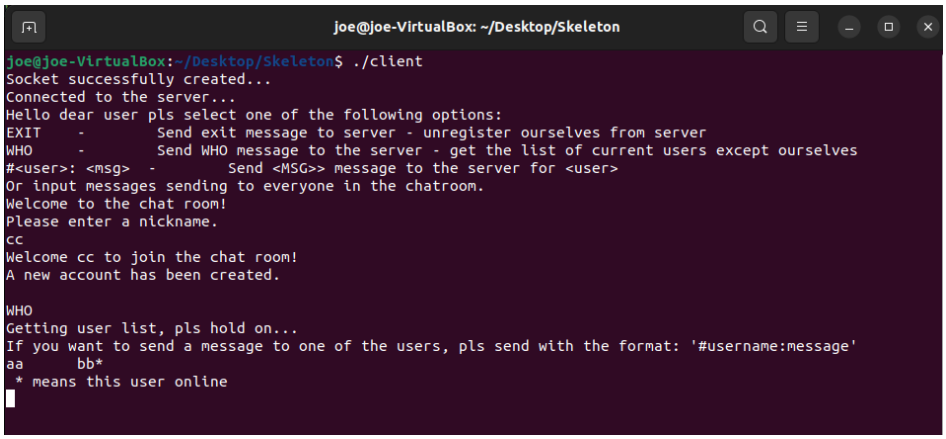


bb receives the broadcast message. bb sends direct message to aa. The message will store in aa.txt because server get the 0 state of aa (i.e. offline). Then, server send message to bb to tell the result.

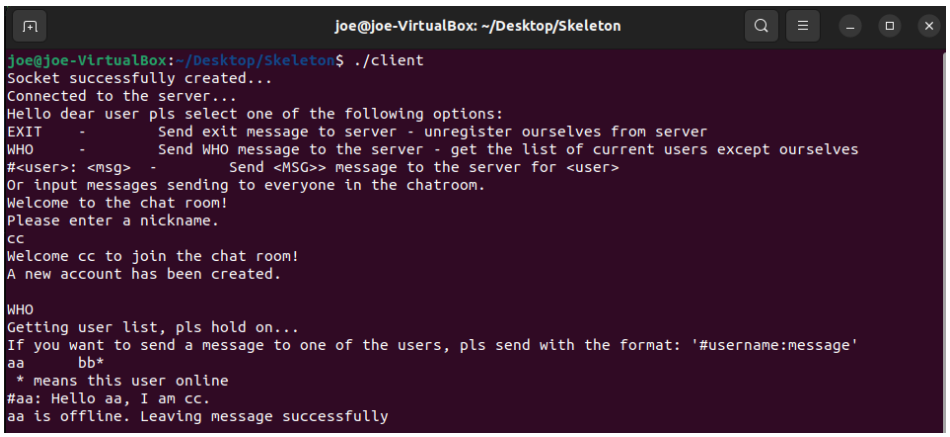




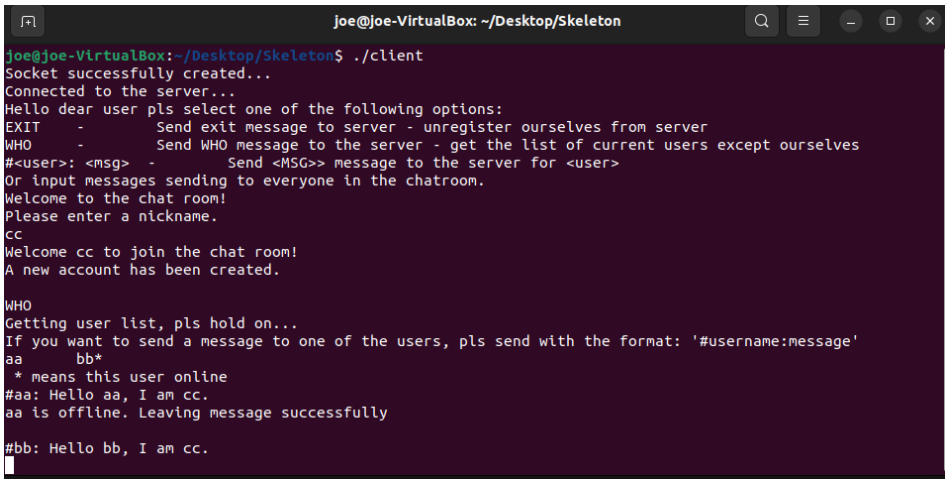
cc join the chat room and bb receive the broadcast message. It is the same as bb joined the chat room and aa received the message before.



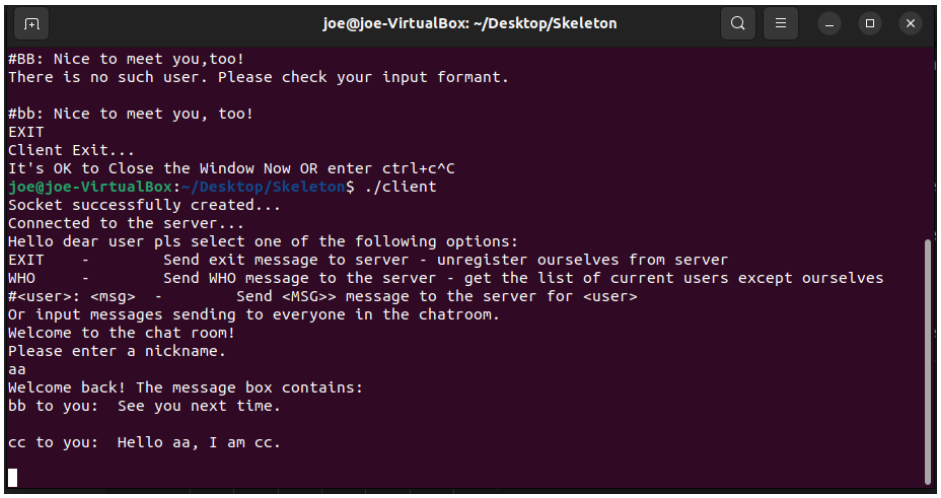
cc types WHO keyword to server. The server gets all username from listOfUsers and add * if the state of user is 1. Each username is separated by \t.

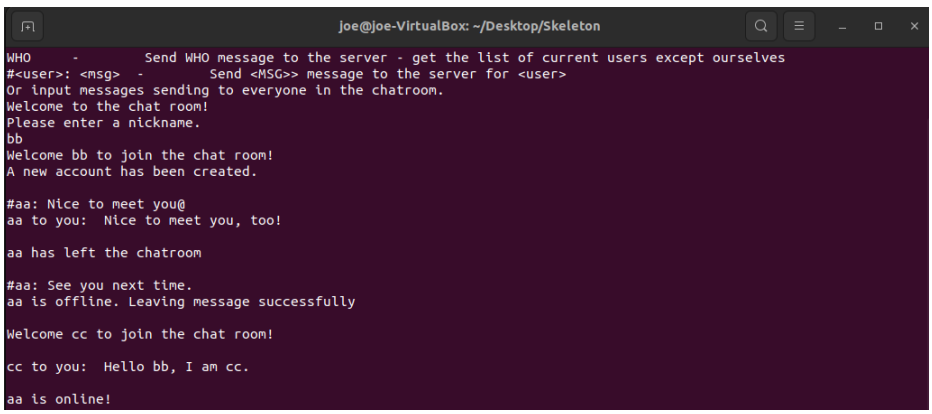


Since aa is offline, cc's message will append to aa.txt. It is the same case as bb sends "See you next time" to aa.

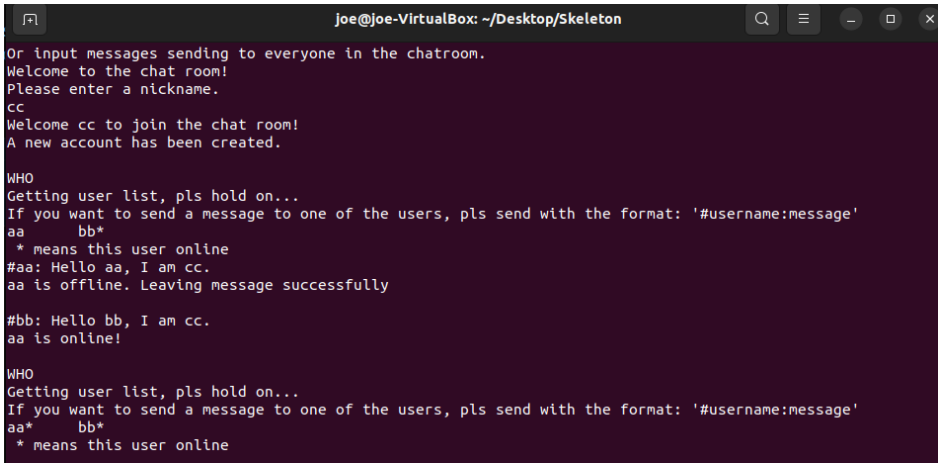


cc sends message to bb. The code logic is also the same as bb sends message to aa before.

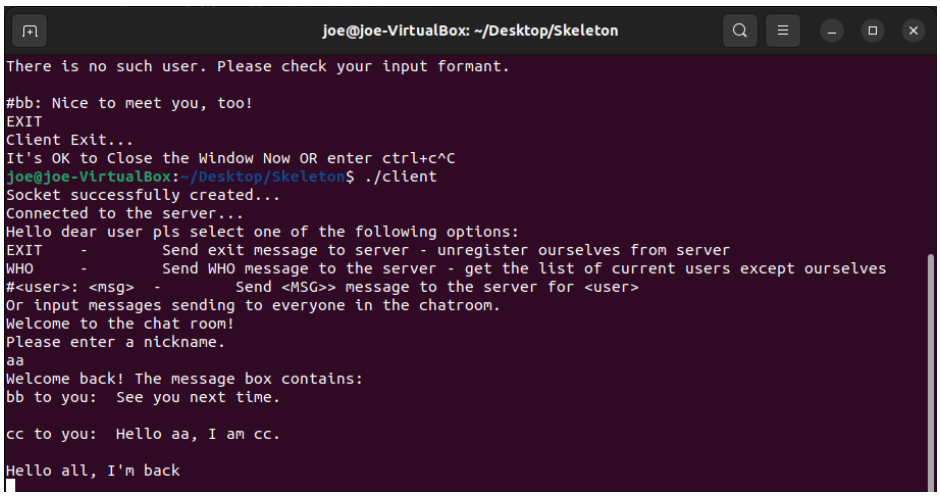


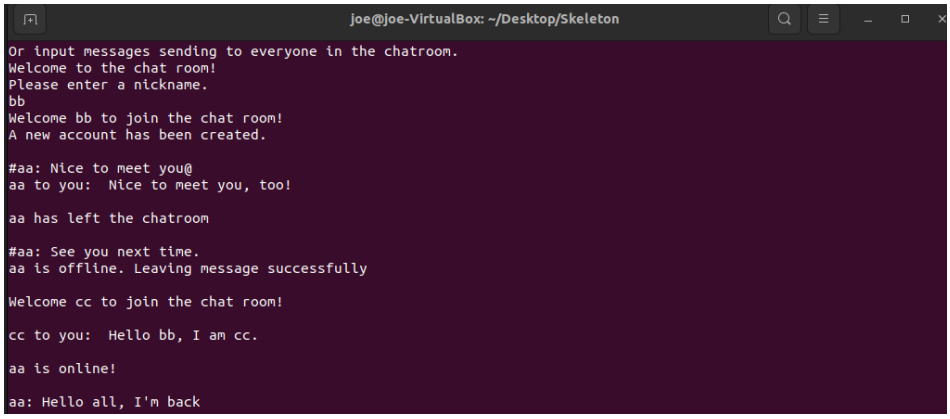


aa logins the chatroom. The server finds aa in the listOfUsers, so it is a login process. Then the server gives a different welcome message from registration, and prints out the content of aa.txt. At the same time, server broadcast a notification that aa is online.



cc checks the user list again. This time, aa and bb are all online, so the states are 1 and therefore they get a *.





aa broadcast to all users. Since there is no keyword (e.g., #, WHO, EXIT, REGISTER), the server classifies it as a broadcast message by default. Then, the server gets the sender name, and checks all users and sends the message to online users.