Protocol usage and Message syntax

# Client -> Server

#     NAME--<name>

#     MESSAGE:<str>

#     QUIT

# Server -> Client

#     WELCOME--<name>

#     HAS\_CONNECTED--<name>  #tell user who's chatroom he joined

#     JOIN--<name> #tell user who joined his chatroom

#     VALID\_NAME--<name> #tell user he register success

#     OTHER\_SIDE\_LEFT #other side people left

#     WAIT\_OTHER\_SIDE\_CONNECTED #wait another people join in chatroom

#     PLEASE\_REGISTER\_NAME #require people register

#     REJECT #reject user connect

#     ERROR\_NAME #user name has error

#     MESSAGE:<text>

#     VALID\_MESSAGE:<message> #send user message

#     SYSTEMERROR:<str> #respond user send message

*Application overview*

Users join or create a chat room by register a user name，also will verify the style of name ，it will give feedback by different commands and instructions to show the status of sever. To achieve chat between two users, use multithreading to serve multiple chat rooms at the same time, set the maximum number of threads, limit the number of users simultaneously served by the server, when the user leaves, it will automatically close the socket link and release the occupied thread resources. When the number of simultaneous users reaches the maximum number, the other links will be rejected. When a user suddenly disconnects, the chat room will be ended and another user will be reminded. Fixed some bugs, for example, when there is only one person in the chat room, sending a message will cause the program to crash, and now it will send a feedback to the user as an error. When someone in the chat room leaves, the other person will still occupy thread resources and send messages will cause a crash. Now when one party leaves, the server will release the resources occupied by the user, and notify the other party that the other party has left and the server is disconnected Link, enter any character and exit. Treat certain information as special instructions instead of returning it to the client as a message. When one of the chatting parties quits, the link between the chatting parties and the server will end, instead of using text as the user name. The sever improved the extendibility and stability

Security and privacy concerns

Because the output transmission is not encrypted, it is easy to be caught to leak the security information. It can not guarantee the safety and privacy of communication, and it can also be easily attacked to make server confusion and collapse by people.

The client archive the registration and format verification of the users’ name, enter chat information and process the server's returned instruction. The users no longer focus on the protocol content and focus more on the client's chat function. When the server turned off unexpectedly, it will handle the error and will not cause the program to crash unexpectedly. Dealing with where the program may go wrong, some unexpected errors will not cause the program to crash unexpectedly. When the server cannot be reached, it will try to connect continuously, you can also enter QUIT to exit.