CMPE 156/L, Winter 2019

Final Programming Project
Due: TBA

1 Description

The objective of this assignment is to create a TCP-based chat program composed of a client for the end users and a server program which will be used to connect clients so that they can exchange messages. The client should operate in three modes: asking the server for information (INFO), waiting to be contacted by another client (WAIT), or exchanging messages with another client (CHAT).

On start up the client must connect first connect to the server. The user can then either tell the server it will be awaiting for a connection, request a list of users that are awaiting for connection, or request to connect with one of those users. If the user requests to wait, the client will wait until it receives a request for connection, then change to the conversation mode (CHAT). In conversation mode the clients can send messages to each other until one of them drops from the conversation, which results in both clients returning to the server (INFO). You can look at the IRC protocol as defined in RFC 1459 for inspiration on how to write your protocol.

1.1 Operation

The server should be concurrent in processing connections from clients. It is started with a port number. Example:

```
./server 1234
```

The client should be started with the IP address and port to the server and an id. Example:

```
./client 128.114.104.54 1234 student1
```

All commands should be in the format "`/<word>`", in which `<word>` is a sequence of lower case letters. A command that appears in the middle of a message can be ignored and treated as part of the conversation, e.g., the message "Use the command `/quit` to leave chat" sent in CHAT mode should be displayed at the other client, not cause the user end the application.

The client program should ignore commands that follow the format but have not been implemented, displaying the message "`Command /<word> not recognized`". For simplicity of implementation, it will be enough to check that the first character in the input is the slash (`/`) for purposes of identifying whether the user has sent a command or not.

The client should also have a prompt "`<id>> `" available at all times, where `<id>` is the id provided when starting the client. While not at INFO mode the client should support the command `/quit` to quit the application.

It should also support the following commands while in INFO mode:

- `/list` - get a list of identifiers of awaiting clients in alphabetical order.
- `/wait` - switches to WAIT mode, while telling the server to add `<id>` to the list of awaiting clients to be shown by `/list`.

The id should contain only letters and numbers. The client should also provide the server with a port number that it will use to receive the connection, this should be done without direct input from the user.

- `/connect <id>` - start a conversation with the client identified by `<id>`. Use Control-C `^C` to end the connected conversation.

The server should provide the client with the address information necessary for the connection and remove it from its waiting list, or say that the other client is not waiting anymore if it is not in the waiting list anymore (e.g. because another client started a conversation with it in the mean time). The other client should be able to refuse the connection if already engaged in conversation.

A client can quit from any state by the quit command.

- `/quit` - Leave program.

1.2 Conversation

While in CHAT mode the two clients should exchange messages directly, **without** any bridging by the server. Messages should be printed as they arrive. Don't worry about the user interface getting messed up if both clients are typing at the same time. The communication should work as expected, not crash and lose any characters. You can assume that messages will only include printable characters that do not cause line breaks: letters, digits, spaces and symbols (e.g. !@#$).

2 Example usage

This depicts one use of the chat program, as seen in the client interface by client with id `my`. Notice the empty prompts while waiting for messages from the server or other client. Notice also how the other use disappears from the waiting list after getting a connection.

```
my> /list
1) Alice
2) bob
3) UsEr13
my> /connect bob
Connected to bob
my> Hi bob
my>
bob: Hi how are you?
my> Testing my chat program
my>
bob: Oh ok
my> Bye
my> ^C
Left conversation with bob
my> /list
1) Alice
2) UsEr13
my> /wait
Waiting for connection.
my>
Connection from bob.
my>
bob: Is the waiting part also working?
my> Yes
my> ^C
Left conversation with bob
my> /wait
Waiting for connection.
my> ^C
Stopped waiting.
my> /quit
```