Networks Assignment 8

1. Group Members

Aditya Singh (18CS30005)
Itish Agarwal (18CS30021)

2. How to Use

Follow the given steps to use the application :-

-> Create an user-input table in the following format (called **user_info_table.txt**)

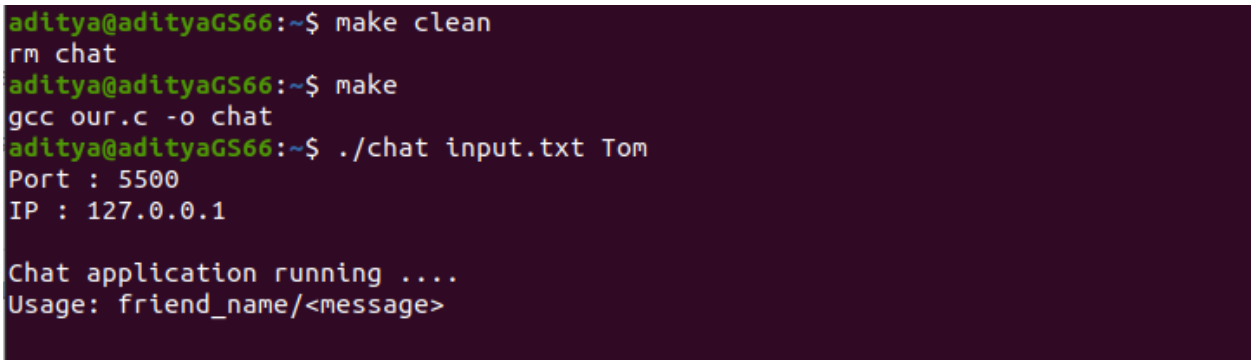
```
4
Tom      127.0.0.1  5500
Jerry    127.0.0.1  5501
Phinaes  127.0.0.1  5502
Ferb     127.0.0.1  5503
```

We have provided a sample **user_info_table.txt** in our zip folder.

First line should contain the number of users (max_peers). The next 'max_peers' number of rows indicate the username, the IP address and the port number of each peer respectively.
Instead of hardcoring the peer names and IP addresses of peers, we have made a peer_list.txt file from which peer information is read.

-> Type ‘**make**’ in the terminal to compile the C program file and create the chat executable file.

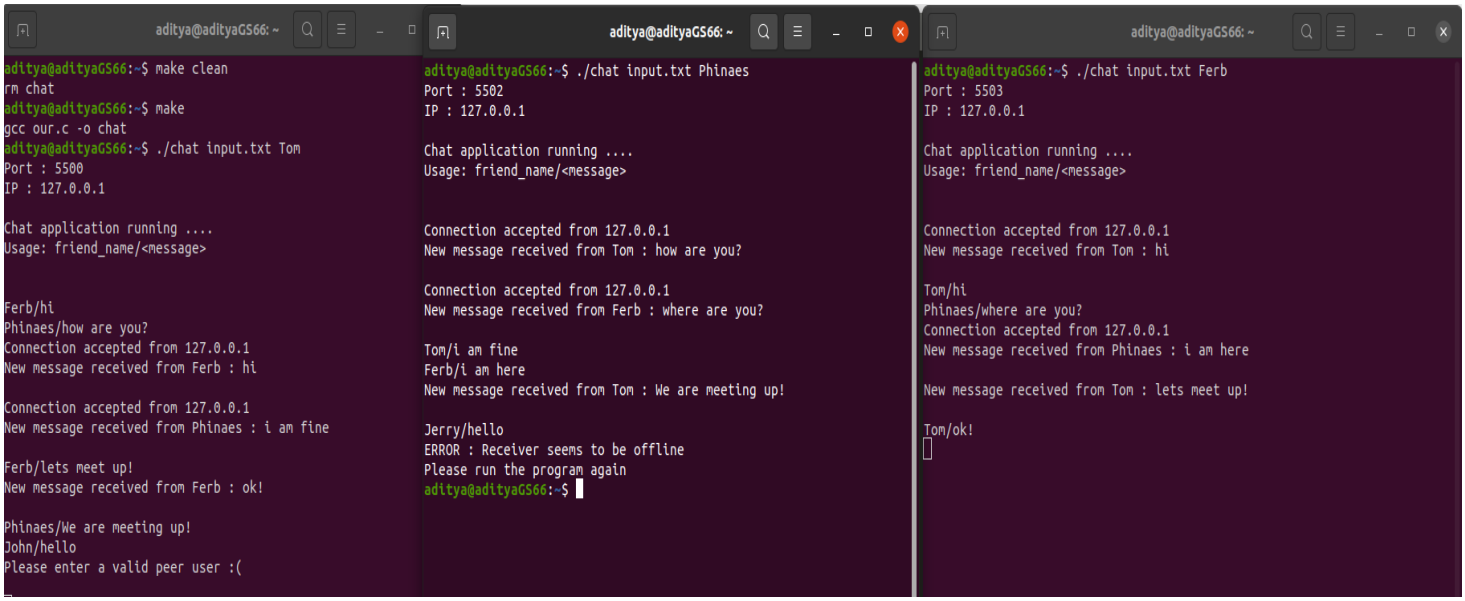
-> To run the chat application, run the command “ **./chat [user_info_table.txt] [peer name]** ”.
Chat instance of that user will start running. It displays the IP address and the port on which the user is using the chat program



-> Once the chat application is running, type the messages in the form of **friend_name/<msg>** where msg is the message you want to send.
Here is a sample input-output chat order between Tom and Jerry

./chat user_info_table.txt Tom (Terminal 1)	./chat user_info_table.txt Jerry (Terminal 2)
Port : 5500 IP : 127.0.0.1 Chat application running .... Usage: friend_name/<message>	Port : 5501 IP : 127.0.0.1 Chat application running .... Usage: friend_name/<message>
Jerry/Hi Jerry, how are you?	

	Connection accepted from 127.0.0.1 New message received from Tom : Hi Jerry, how are you?
	Tom/Hi Tom, I am fine
Connection accepted from 127.0.0.1 New message received from Jerry : Hi Tom, I am fine	



-> Use '**`make clean`**' and '**`make`**' to delete the chat executable file and start again.

3. Documentation

- From the user_info_table.txt, number of peers are read. Then username, IP addresses and ports of different users are read.

- Username of the peer currently using the service is taken as command line argument, ie, argv[2]
- peer_list.txt is taken as command line argument, ie, argv[1]

- Then we find out the name, ip address and port of the user whose username that we got from command line argument

- Then we create a listening server of this instance
  -> create socket, bind to the socket

- Now start listening on this server

- Now if a client wants to connect we will check if the request is a new connection request or the client has already connected before

- If the request is a new request, accept this request and open the socket. Otherwise, just send the message through the already opened socket and display the message

- Select system call is used to check which file descriptors are ready (either to write into or to read from)

- Maintain an array last_time[], which basically stores the last time that particular peer reads/writes some data into the chat application

- At each select system call, we check for each peer, if
  `current time(ie, time(NULL)) - last_time[peer] > LIMIT`

If this is true, we put a timeout message and close the socket corresponding to that client