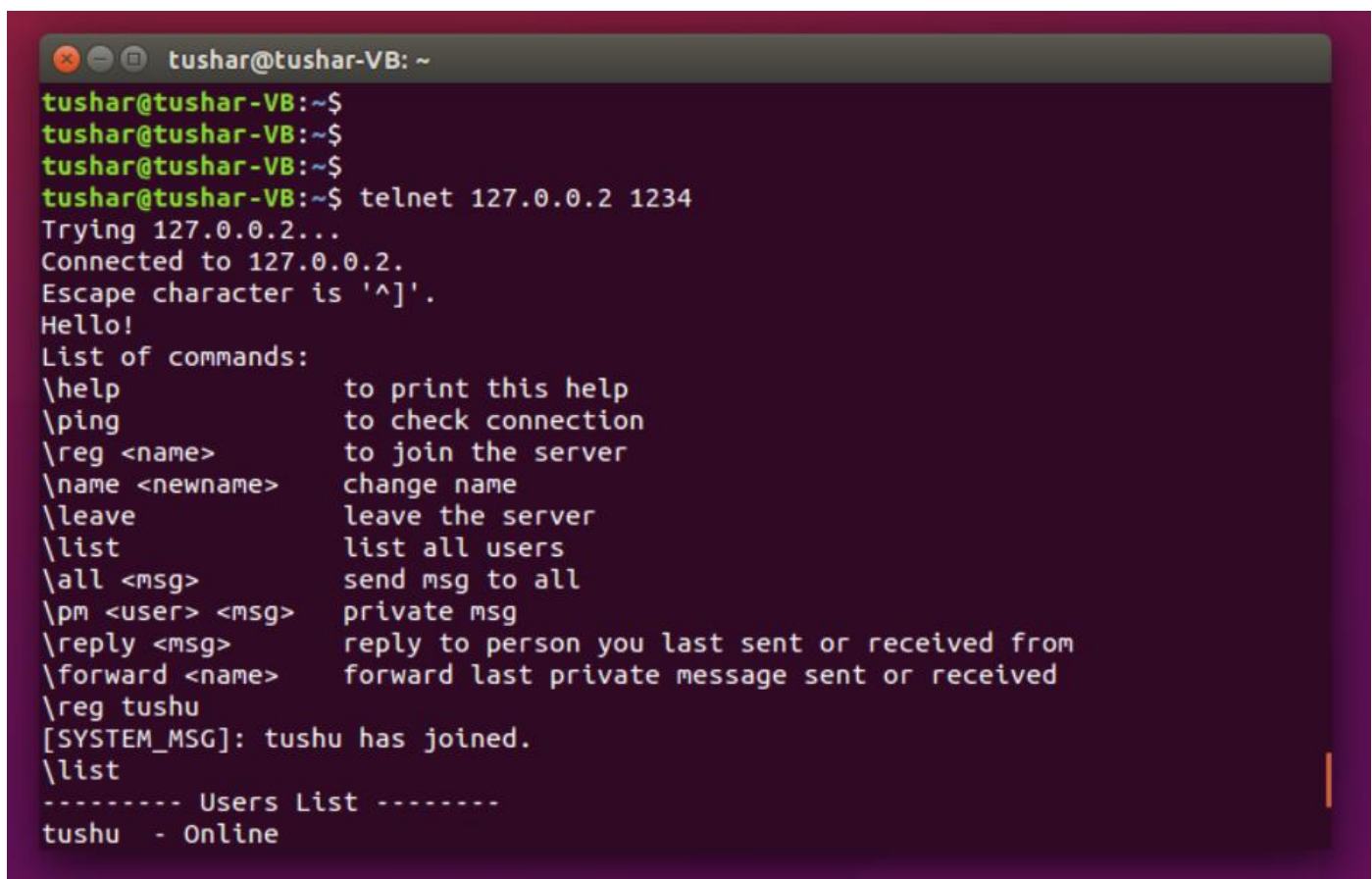


Design features of Chat Server

The server listens to both message queue and new client requests at the same time so as to take the required action. Select system call is used for this behavior. When a new user connects to the server, the server forks itself and the new child created handles all further communication with the client. All communication between children and parent-children takes place through message queues. For distinguishing between different types of messages on message queue mtype is used.

1. For **part(a)**, if a client wants to join the server, it sends a connection request to server. The server provides a unique user id to the client and adds information about it in a client list it maintains. The server then creates a fork, and any further message from the client is directed to the child. The client has to first register if it wants to communicate with other users. When the client wants to leave the server, it sends the leave command to the child. And the child conveys this information to the parent using message queue which frees up the corresponding memory in clients' array. All other clients are notified too.
2. For **part(b)**, the client asks its corresponding child for the list, receiving which the child asks to its parent for the current updated list of all the clients using message queue. Parent sends the information back to the child using message queue. And the client concatenates the user list in a string format before sending it back to the user who asked for it.
3. For **part(c)**, when a user wants to send a private message, the corresponding child retrieves an updated list of all the clients and check if receiver is a valid client connected to the chat server. If it is, then it sends the child associated with the receiver to convey sender's message. This child then sends the message to its client. And the process is complete. For broadcasting a message to every user, the process is similar. Client gets an updated list of users, and sends a message over message queue to all associated children. Those children then carry forward the message to their client.
4. The mentioned way for implementation in **part(d)** has been used to implement all the features as explained previously.

5. For **part(e)**, three new functions, “ping”, “reply” and “forward” are implemented. “ping” allows a user to check connection with the server. Sending upon which it receives back “pong” from the server, if the connection is not broken. A user can use the “reply” feature to reply to the user it has previously contacted by either sending or receiving a private message. It saves time for the Client as it doesn’t have to write the receiver’s name again. User can also forward the previously received or sent message to another user by using the “forward” command. These are implemented by storing the previous correspondence details in other state variables and taking action using them if need arises.



```
tushar@tushar-VB: ~  
tushar@tushar-VB:~$  
tushar@tushar-VB:~$  
tushar@tushar-VB:~$ telnet 127.0.0.2 1234  
Trying 127.0.0.2...  
Connected to 127.0.0.2.  
Escape character is '^]'.  
Hello!  
List of commands:  
\help          to print this help  
\ping          to check connection  
\reg <name>    to join the server  
\name <newname> change name  
\leave        leave the server  
\list         list all users  
\all <msg>    send msg to all  
\pm <user> <msg> private msg  
\reply <msg>  reply to person you last sent or received from  
\forward <name> forward last private message sent or received  
\reg tushu  
[SYSTEM_MSG]: tushu has joined.  
\list  
----- Users List -----  
tushu  - Online
```

All the commands can also be seen in the above screenshot of a telnet client connected to the server.