NAME - AAKASH KUMAR SAGAR
ROLL NO - 2018323

MULTI-USER CHAT SYSTEM

Basic Ideas Behind the Code

There are two files present in the chat system.
1. Server: Handles all the incoming connections to the server.

2. Client: Used to create n numbers of clients.

I have used UNIX sockets for the implementation of the IPC.

I have used AF_UNIX family of sockets, therefore, it will run on local machine only

Implementation Details of the Code

The socket is build for the server.
By using a file of the system the address is bound to the server.
At master socket, the server begins to listen to the client.
The server uses the select function to select the client.
On the given file path, the clients connect to the server.
The users can connect to the system.

INPUT FORMAT

The input format is given below

user_id Message_to_client

If the message is sent to all the client input 3 as user_id
e.g:- 3 message_to_client

NAME - AAKASH KUMAR SAGAR
ROLL NO - 2018323

If the message is sent to an individual client input 4 or above as user_id
E.g:- 4 message
Note:- The client ids start from 4

In order to close a Client
Press CTRL+C.

How to compile & Run the Code

First, compile the server.c code and client.c code using command
gcc -o Cli Client.c
gcc -o Ser Server.c

Then open the desired number of terminals and run the **Ser** file on one terminal and run the **Cli** file on the remaining terminals

In order to run Ser type ./Ser
In order to run the Cli type ./Cli

Follow the above input format to communicate

ERRORS HANDLED

1.if the client tries to send a message to himself. This has been handled.
2. if there is no space between the user ID and message in the code. It has been handled and automatically rectified and sent to the required client.

ERRORS EXPECTED

If the user gives the wrong ID then the server fails so make sure that valid ids are used in communication

NAME - AAKASH KUMAR SAGAR
ROLL NO - 2018323