MULTI-USER CHAT SYSTEM

Using SOCKET Programming

Aman Roy | Gyanesh Anand

2016011 | 2016039

21-03-2018

Problem Statement

Design and implementation of a multi user chat system that is like a real-time instant message board system, much like IRC. The users should be able to talk to one another and to everyone else. The system needs to make sure which message is meant for whom – for individual user or for everyone. The message meant for everyone should be flashed on the system for everyone to view. The messages meant for individual users need to be handled by the individual users themselves. Popular Linux IPC mechanism have been used for the same.

Basic Ideas Behind the Code : Logics

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.

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

It would run on the local machine only and not over the Internet as AF_UNIX family of sockets have been used.

Implementation Details

Server & Client

The socket is created for the server.
The address is bound to the server using a file on the system.
Server starts listening to the client at master socket.
Server selects the client using the select function.
The clients connects to the server on the given file path.
The users can connect to the system.

INPUT Method

TO input given is the format.

USER_NO Message

In order to send to all give 3 as user id

3 message

In order to send to client 4
4 message

The client ids start from 4

CLOSING THE CLIENT

Press CTRL+C.
It also handles the abrupt client end error.

How to compile & Run the Code

Run the **make**
In order to run server do **./server**
In order to connect the clients do **./client**

Error Values & their Interpretations

ERRORS HANDLED

1. Error will be there if the client tries to send message to himself. This has been handled.
2. Error can be there if there is no space between user ID and message in the code . It has been handled and automatically rectified and sent to required client.
3. All other errors in send , recv , listen , socket etc have been caught and handled with.

ERRORS EXPECTED

If the user gives the wrong ID then the server fails so make sure that valid ids are only given.

REFERENCES

https://github.com/csepracticals/SerVerDesign/blob/master/sts/mx_tcp_server.c

Beej : UNIX SOCKETS