OPERATING SYSTEMS PROJECT

Grup 1 – Ahmet Elbir

MÜDAFER KAYMAK

20011093

Mudafer.kaymak@std.yildiz.edu.tr

SUBJECT

The project contains multi-user messaging application in c. It has two parts: Server and client.

Server listens clients for incoming messages, assigns unique ID for clients and forward messages to specified client. Clients works simultanouesly in different terminals, send messages to other clients and receives messages from other clients.

SOLUTION

In this project, we were tasked with developing a messaging application. To achieve this, server and client programs were written to establish communication between them. Clients were created to communicate with the server, where requests received from clients were processed on the server and then distributed back to the clients. To achieve this i used socket programming. Multithreading was employed on both sides to enable simultaneous handling of incoming and outgoing operations.

FUNCTIONALITY

```
typedef struct {
    int client_socket;
    int client_id;
    char profile[50];
} ClientInfo;
```

In the struct above i keep the clients info which includes socket for connection, ID for distincions between clients and profile for client username.

Client Side

receive_messages():

in an finite loop it listens to server and check if there is any message receiving.

main:

By socket programming, connects clients with server. Then it takes login message from server and show it to client after that it takes clients username and sends back to server and receiver clients ID from server. It creates a thread for receiving messages. Ask clients to select their process for next like send message and exit. After input it applies these process.

Server Side

```
typedef struct {
    int client_socket;
    int client_id;
    char profile[50];
} ClientInfo;
```

In the struct above i keep the clients info which includes socket for connection, ID for distincions between clients and profile for client username.

handle_client

Sends login message to clients then reads client's username, parse login format and check if it is in true format. Sends clients' unique ID and until client disconnects it listens the client in case of any message request.

Main

By socket programming provide connections between server and clients. Creates new threads for each client.

OUTPUTS

I have opened 3 clients

```
Server listening on port 8080...
Client 0 connected.
Client 1 connected.
Client 2 connected.
```

Server says that your client ID is 1. Now i want to send a message to ID 2

```
Your client ID is: 1using /login Username
Enter your choice (1: Send, 3: Exit): 1
Enter the recipient's client ID: 2
Enter your private message: merhaba naber nasilsin
```

Client whose ID is 2 got the message like that

```
Your client ID is: 2using /login Username
Enter your choice (1: Send, 3: Exit): Server says: merhaba naber nasilsin -User1
```

Client can get message while sending a message. In this example he gets a message from client whose ID is 0