

How to Start:

I started by looking at the “The Definitive Guide to Linux Network Programming by Keir Davis, John W. Turner, and Nathan Yocom” book. I copied and altered the codes from chapter 2 of the book just to clearly understand the functions and code blocks clearly. First I tried some important functions like `socket()` and `bind()`. Then I continued by coding other parts.

Features:

I wanted to code a chat program but it shouldn't be something simple, so I decided to add nickname feature. Since I also wanted some clients to stay anonymous if they want to, I had to add the join feature. When you run the client type `“/join <nickname>”` you will have a nickname you choose and can send messages to other clients. Anonymous clients can see the messages, but they can't send either type of message.

To send messages you can either type `“/msg <message>”` or `“/enc <message>”`. “enc” is short for encrypted and when you send a message it uses a key to encrypt your message by XORing its bytes (I coded a file named “encryption.c”, you can see the details of the implementation). Every time a new client joins, key changes randomly (using time seed) so that clients cannot decrypt the messages that has been sent when they were anonymous.

In the encryption part I got help from “Her Yönüyle C by Tefik Kızılören” book (I needed some help with the functions). I used the same book when I was coding the server functions like `decodeCommand()`, `broadcast_message()` etc. I did not take any code blocks from this book, I just learned and used the functions.

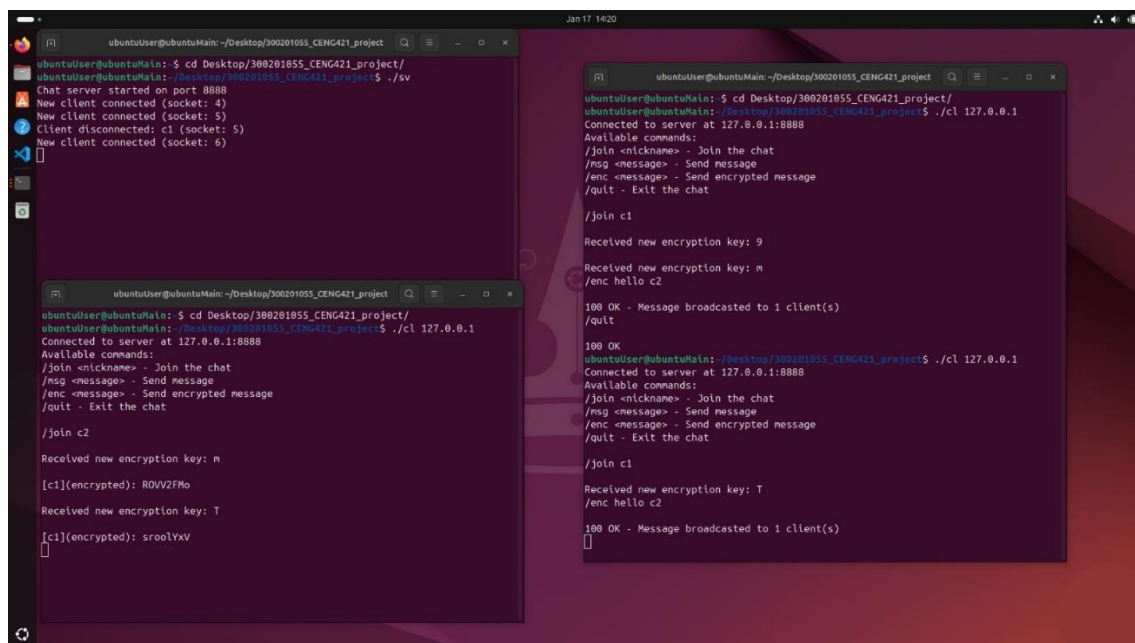
When you type `“/quit”` client disconnects. Server shows log messages for both client connections and disconnections. Also errors are shown as log messages in client's or server's terminal (according to error cause).

Where I Struggled:

The most challenging part was of course sending the encrypted messages. At first, I tried the XOR encryption method with ASCII values. But since ASCII table includes characters like 'NULL', it wasn't a good idea. Then I created my own table that includes only alphanumeric values, space and dot and solved this problem. Another issue I encountered was sending messages that include space in them. It was a really big problem that causes 'Segmentation Fault'. This was because my code had the assumption `buffer[0] = cmd` and `buffer[1] = msg` and the delimiter was space character. So I had to make an adjustment to change the delimiter after assigning `buffer[0]`. I used <https://stackoverflow.com/> to learn how to solve these problems. I did not directly take any code from there (since the bugged code there wasn't same with my code) but took the ideas and implemented according to those ideas.

Outputs:

In the output below, c1 sends the same encrypted message after rejoining but message is received differently by c2.



```
ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project
ubuntuUser@ubuntuMain: $ cd Desktop/300201055_CENG421_project/
ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project$ ./sv
Chat server started on port 8888
New client connected (socket: 4)
New client connected (socket: 5)
Client disconnected: c1 (socket: 5)
New client connected (socket: 6)

ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project
ubuntuUser@ubuntuMain: $ cd Desktop/300201055_CENG421_project/
ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project$ ./cl 127.0.0.1
Connected to server at 127.0.0.1:8888
Available commands:
/join <nickname> - Join the chat
/msg <message> - Send message
/enc <message> - Send encrypted message
/quit - Exit the chat

/join c2

Received new encryption key: n
[c1](encrypted): ROVV2Fm0

Received new encryption key: T
[c1](encrypted): sroolYxV

ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project
ubuntuUser@ubuntuMain: $ cd Desktop/300201055_CENG421_project/
ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project$ ./cl 127.0.0.1
Connected to server at 127.0.0.1:8888
Available commands:
/join <nickname> - Join the chat
/msg <message> - Send message
/enc <message> - Send encrypted message
/quit - Exit the chat

/join c1

Received new encryption key: 9
Received new encryption key: m
/enc hello c2

100 OK - Message broadcasted to 1 client(s)
/quit

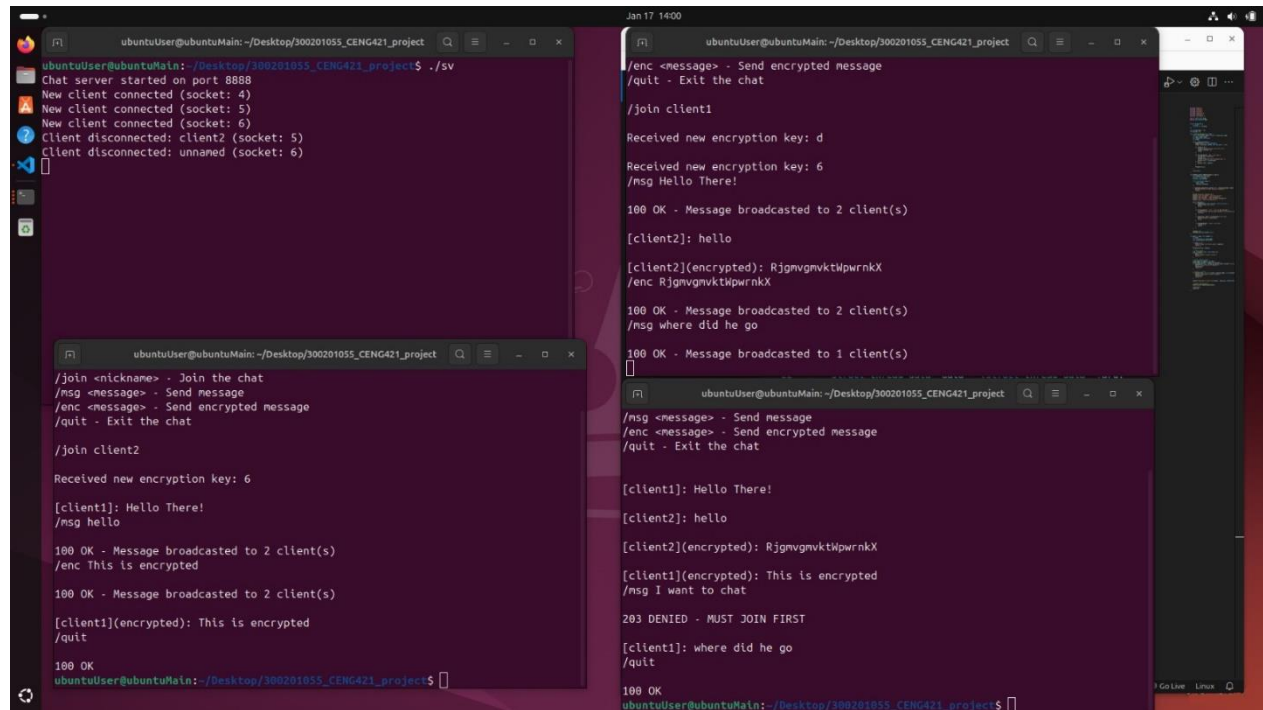
100 OK
ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project$ ./cl 127.0.0.1
Connected to server at 127.0.0.1:8888
Available commands:
/join <nickname> - Join the chat
/msg <message> - Send message
/enc <message> - Send encrypted message
/quit - Exit the chat

/join c1

Received new encryption key: T
/enc hello c2

100 OK - Message broadcasted to 1 client(s)
```

In the output below, client counter changes and log messages are printed when new clients connect or disconnect.



The screenshot shows a terminal window with the following output:

```
ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project
ubuntuUser@ubuntuMain:~/Desktop/300201055_CENG421_project$ ./sv
Chat server started on port 8888
New client connected (socket: 4)
New client connected (socket: 5)
New client connected (socket: 6)
Client disconnected: client2 (socket: 5)
Client disconnected: unnamed (socket: 6)

/Join client2

Received new encryption key: 6

[client1]: Hello There!
/msg hello

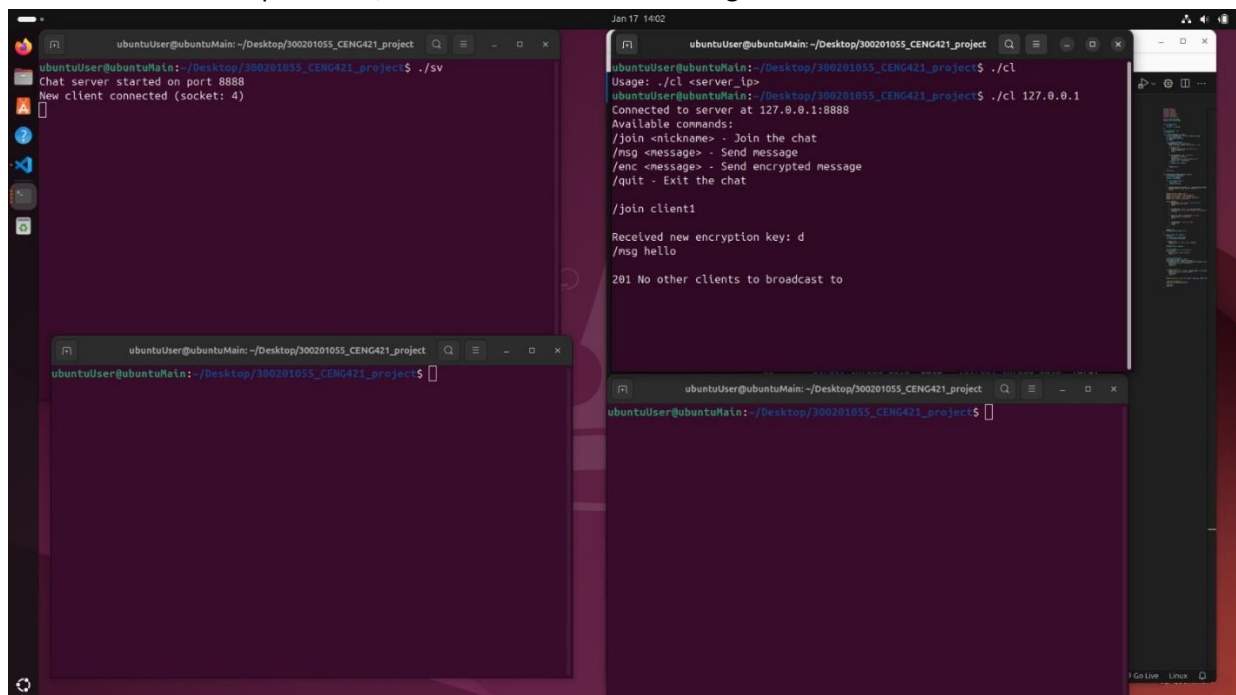
100 OK - Message broadcasted to 2 client(s)
/enc This is encrypted

100 OK - Message broadcasted to 2 client(s)

[client1](encrypted): This is encrypted
/quit

100 OK
ubuntuUser@ubuntuMain:~/Desktop/300201055_CENG421_project$
```

In the output below, client1 cannot send a message since no other clients are connected.



The screenshot shows a terminal window with the following output:

```
ubuntuUser@ubuntuMain:~/Desktop/300201055_CENG421_project
ubuntuUser@ubuntuMain:~/Desktop/300201055_CENG421_project$ ./sv
Chat server started on port 8888
New client connected (socket: 4)

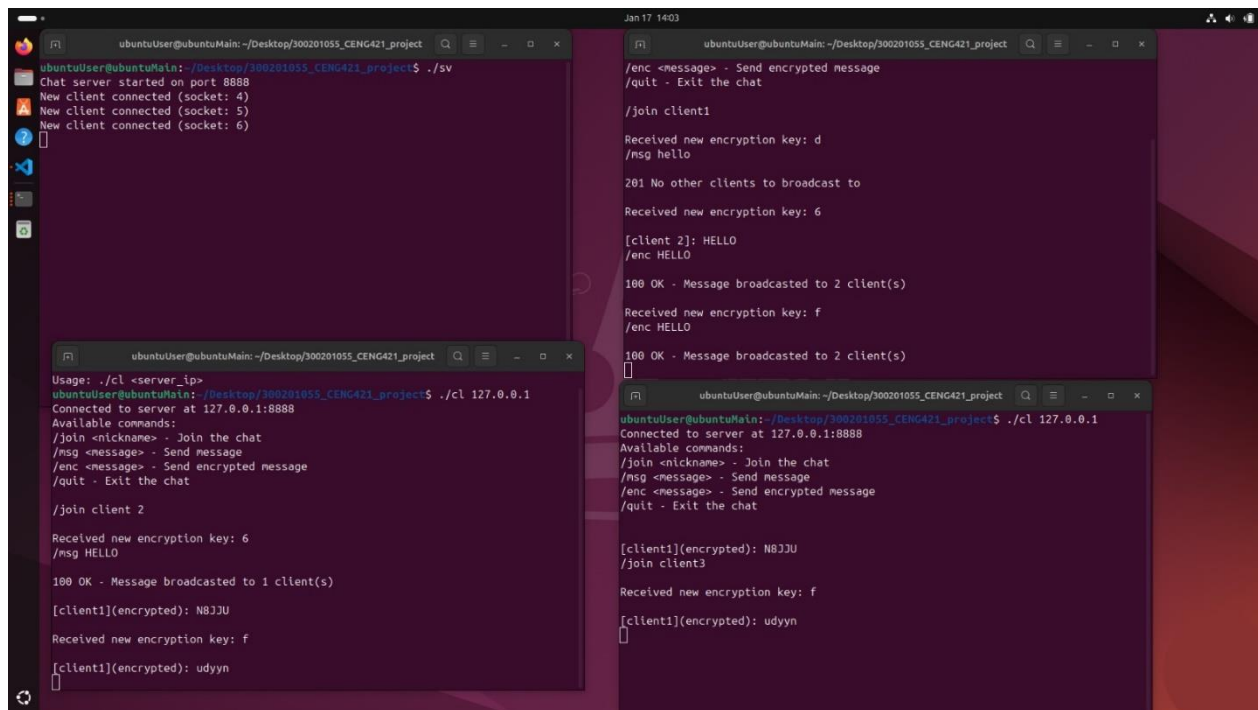
/Join client1

Received new encryption key: d
/msg hello

201 No other clients to broadcast to

ubuntuUser@ubuntuMain:~/Desktop/300201055_CENG421_project$
```

In the output below, simulation of a short greeting is shown.



The screenshot displays three terminal windows from an Ubuntu desktop environment. The top-left window shows the server's startup and initial client connections. The bottom-left window shows a client joining the chat and sending a message. The right window shows the server's response to the client's message, including encryption key distribution and message broadcasting.

```
ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project
Chat server started on port 8888
New client connected (socket: 4)
New client connected (socket: 5)
New client connected (socket: 6)

Usage: ./cl <server_ip>
ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project$ ./cl 127.0.0.1
Connected to server at 127.0.0.1:8888
Available commands:
/join <nickname> - Join the chat
/msg <message> - Send message
/enc <message> - Send encrypted message
/quit - Exit the chat

/join client 2

Received new encryption key: 6
/msg HELLO

100 OK - Message broadcasted to 1 client(s)
[client1](encrypted): N8JJU
Received new encryption key: f
[client1](encrypted): udyyn

ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project$ ./sv
/enc <message> - Send encrypted message
/quit - Exit the chat

/join client1

Received new encryption key: d
/msg hello

201 No other clients to broadcast to

Received new encryption key: 6
[client 2]: HELLO
/enc HELLO

100 OK - Message broadcasted to 2 client(s)
Received new encryption key: f
/enc HELLO

100 OK - Message broadcasted to 2 client(s)

ubuntuUser@ubuntuMain: ~/Desktop/300201055_CENG421_project$ ./cl 127.0.0.1
Connected to server at 127.0.0.1:8888
Available commands:
/join <nickname> - Join the chat
/msg <message> - Send message
/enc <message> - Send encrypted message
/quit - Exit the chat

[client1](encrypted): N8JJU
/join client3

Received new encryption key: f
[client1](encrypted): udyyn
```

To Run:

Server

```
gcc server.c encryption.c -o sv
./sv
```

Client

```
gcc client.c -o cl
./cl serverIP (if same -> 127.0.0.1)
```

References:

- “The Definitive Guide to Linux Network Programming by Keir Davis, John W. Turner, and Nathan Yocom” from <https://kalfaoglu.com/ceng421/The%20Definitive%20Guide%20to%20Linux%20Network%20Programming.pdf>
- “Her Yönüyle C by Tevfik Kızılören” from my roommate (hard copy)
- <https://stackoverflow.com/questions/21034107/what-situations-are-there-where-one-might-want-to-use-the-bitwise-xor-operator>
- <https://stackoverflow.com/questions/52207214/how-to-use-strncpy-correctly>
- <https://stackoverflow.com/questions/19641597/what-is-segmentation-fault-core-dumped>
- <https://stackoverflow.com/questions/59039123/segmentation-fault-core-dumped-issue-when-running-my-code/59039362>
- <https://commons.wikimedia.org/wiki/File:ASCII-Table-wide.svg>