

## C\_PROIECT

### 1. Ce face proiectul

Acest program server-client stabilește o conexiune între un server și un client, permițând clientului să trimită mesaje către server. În plus, clientul criptează mesajele folosind un cifru Caesar înainte de a le trimite. Mai jos este o explicație pas cu pas:

#### **Server (server.c):**

- Serverul creează un socket și îl leagă la un port specific (8080 în acest caz).
- Apoi începe să asculte conexiuni de intrare.
- Odată ce un client se conectează, serverul acceptă conexiunea și începe să citească mesaje de la client.
- Serverul citește un total de 5 mesaje (definite de MAX\_MSGS) de la client. Fiecare mesaj este afișat în consolă.
- După ce primește toate mesajele, serverul trimite un răspuns înapoi la client indicând faptul că toate mesajele au fost primite.

#### **Client (client.c):**

- Clientul creează, de asemenea, un socket și se conectează la adresa și portul serverului.
- Apoi intră într-un ciclu în care îi cere utilizatorului să introducă un mesaj de 5 ori.
- Fiecare mesaj este criptat folosind un cifru Caesar, care deplasează fiecare literă în mesaj cu o anumită valoare (3 în acest caz). Mesajul criptat este afișat în consolă.
- Mesajul criptat este apoi trimis la server.
- După ce toate mesajele sunt trimise, clientul citește și afișează răspunsul de la server.

Cifrul Caesar este o tehnică simplă de criptare în care fiecare literă din textul simplu este deplasată un anumit număr de poziții în sus sau în jos pe alfabet. În acest caz, deplasarea este de 3 poziții în jos. De exemplu, cuvântul "hello" ar fi criptat ca "khoor".

### 2. Program server -

#### 2.1 -cod comentat

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <netinet/in.h>
#include <unistd.h>

#define PORT 8413
#define MAX_MSGS 5

int main() {
    int server_fd, new_socket;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    int msg_count = 0;
    printf("Server started on port 8413 made by gr3_13 Alina Atanasiu\n");

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt, sizeof(opt))) {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    if (listen(server_fd, 3) < 0) {
        perror("listen");
        exit(EXIT_FAILURE);
    }
}
```

```

if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) {
    perror("accept");
    exit(EXIT_FAILURE);
}

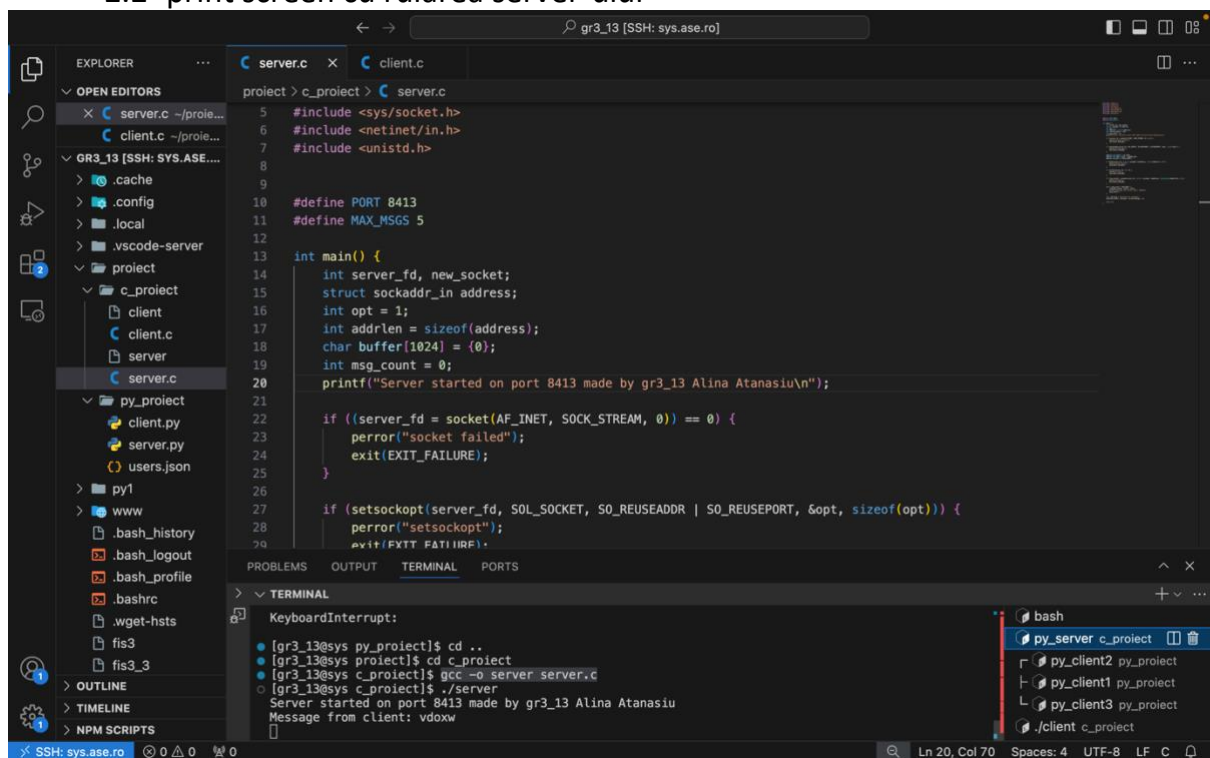
while (msg_count < MAX_MSGS) {
    read(new_socket, buffer, 1024);
    printf("Message from client: %s\n", buffer);
    msg_count++;
}

char *message = "Received all messages";
send(new_socket, message, strlen(message), 0);

return 0;
}

```

## 2.2 -print screen cu rularea server-ului



## 2.3- portul cu identificare protocolului si procesul

```
server.c  client.c
project > c_project > client.c
1  #include <stdio.h>
2  #include <sys/socket.h>
3  #include <arpa/inet.h>
4  #include <unistd.h>
5  #include <string.h>
6
7  #define PORT 8413
8  #define SHIFT 3
9
10 void caesar_encrypt(char* message, int shift) {
11     char c;
12     for (int i = 0; message[i] != '\0'; ++i) {
13         c = message[i];
14         if (c >= 'a' && c <= 'z') {
15             message[i] = (c - 'a' + shift) % 26 + 'a';
16         } else if (c >= 'A' && c <= 'Z') {
17             message[i] = (c - 'A' + shift) % 26 + 'A';
18         }
19     }
20 }
21
```

```
gr3_13@sys ~]$ pstree -p -u gr3_13 | grep server
sh(3602204)---node(3602210)---node(3602257)---bash(3603481)---server(3604460)
gr3_13@sys ~]$ netstat -a -n -p |grep -w 8413
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:8413          0.0.0.0:*               LISTEN     3606380/./server
tcp        0      0 127.0.0.1:59920      127.0.0.1:8413          ESTABLISHED 3606488/./client
tcp        0      0 127.0.0.1:59920      127.0.0.1:59920        ESTABLISHED 3606380/./server
```

### 3. Program client -

#### 3.1 -cod comentat

```
#include <stdio.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <string.h>

#define PORT 8413

#define SHIFT 3

void caesar_encrypt(char* message, int shift) {

    char c;

    for (int i = 0; message[i] != '\0'; ++i) {

        c = message[i];

        if (c >= 'a' && c <= 'z') {

            message[i] = (c - 'a' + shift) % 26 + 'a';

        } else if (c >= 'A' && c <= 'Z') {

            message[i] = (c - 'A' + shift) % 26 + 'A';

        }

    }

}
```

```

int main() {
    int sock = 0;
    struct sockaddr_in serv_addr;
    char message[1024] = {0};
    char buffer[1024] = {0};

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\n Socket creation error \n");
        return -1;
    }

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
        printf("\nInvalid address/ Address not supported \n");
        return -1;
    }

    if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
        printf("\nConnection Failed \n");
        return -1;
    }

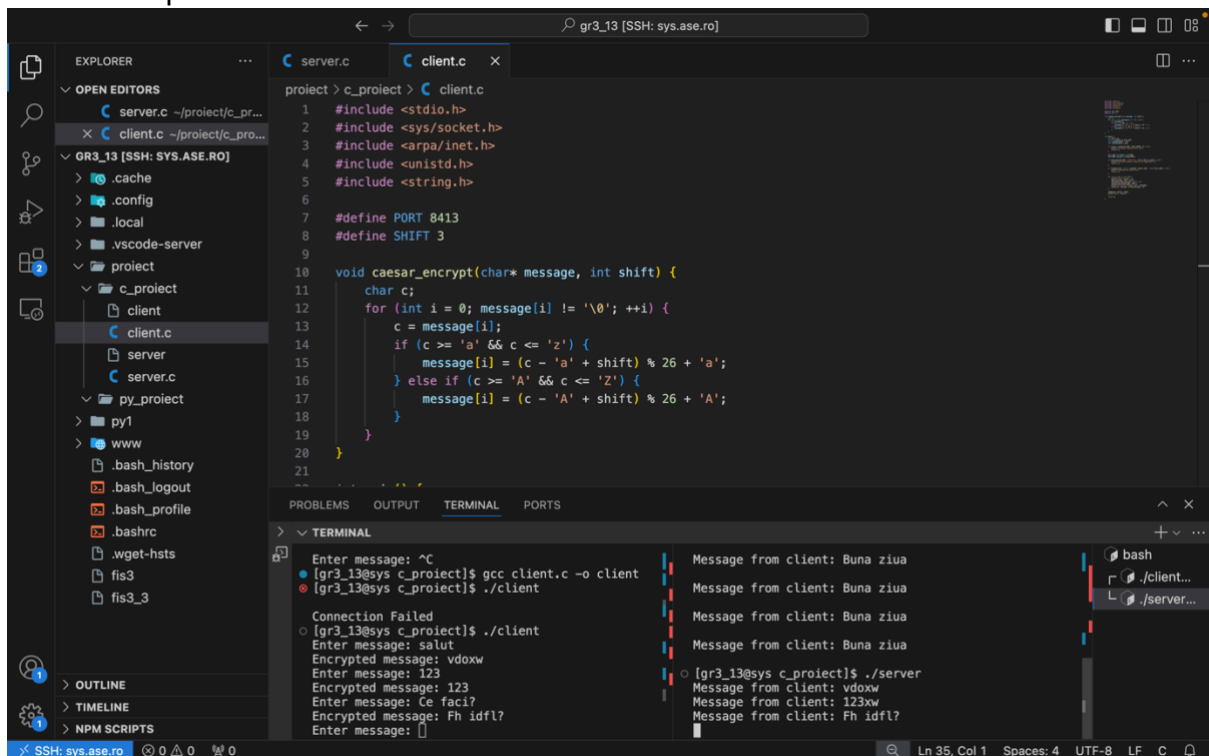
    for (int i = 0; i < 5; i++) {
        printf("Enter message: ");
        fgets(message, 1024, stdin);
        message[strcspn(message, "\n")] = 0;
        caesar_encrypt(message, SHIFT);
        printf("Encrypted message: %s\n", message);
        send(sock, message, strlen(message), 0);
    }

    read(sock, buffer, 1024);
    printf("%s\n", buffer);

    return 0;
}

```

### 3.2 -print screen cu rularea client-ului



The screenshot shows the Visual Studio Code editor with a project named 'c\_project' open. The file explorer on the left shows the project structure, including 'client.c'. The main editor window displays the code for 'client.c', which includes headers for `<stdio.h>`, `<sys/socket.h>`, `<arpa/inet.h>`, `<unistd.h>`, and `<string.h>`. It defines a port of 8413 and a shift of 3. The `caesar_encrypt` function is implemented, which takes a message and a shift, and returns the encrypted message. The terminal at the bottom shows the execution of the client program. It prompts for a message, which is entered as 'Buna ziua'. The program then sends the message to the server, and the server responds with the encrypted message 'vdoxw'. The terminal also shows the server's output, which displays the received message and the encrypted message.

## PY\_PROIECT

### 1. Ce face proiectul

Acest proiect este un sistem simplu de chat care utilizează socket-uri pentru comunicare între server și clienți.

Serverul acceptă conexiuni de la clienți, primește mesaje de la ei și le transmite tuturor celorlalți clienți. Când un client se conectează, serverul îi cere un nume de utilizator (nickname) și o parolă, pe care le stochează pentru a le folosi ulterior. Clientul se conectează la server, trimite mesajele introduse de utilizator și primește și afișează mesajele de la ceilalți clienți. Înainte de a se conecta, clientul cere utilizatorului un nume de utilizator și o parolă, pe care le verifică într-un fișier JSON local.

În ambele scripturi, comunicarea este gestionată în thread-uri separate pentru a permite trimiterea și primirea simultană a mesajelor.

Dacă un client se deconectează sau apare o eroare în comunicare, serverul îl elimină din lista de clienți și anunță ceilalți clienți că acesta a părăsit chat-ul.

### 2. Program server

#### 2.1 -cod comentat

```
import socket
import threading
```

```

def broadcast(message, _client):
    for client in clients:
        if client != _client:
            client.send(message)

def handle(client):
    while True:
        try:
            message = client.recv(1024)
            broadcast(message, client)
        except:
            index = clients.index(client)
            clients.remove(client)
            client.close()

            nickname = nicknames[index]
            nicknames.remove(nickname)
            broadcast(f'{nickname} left the chat!'.encode('ascii'), _client=None)
            break

def receive():
    while True:
        client, address = server.accept()
        print(f'Connected with {str(address)}')

        client.send('NICK'.encode('ascii'))
        nickname = client.recv(1024).decode('ascii')
        client.send('PASS'.encode('ascii'))
        password = client.recv(1024).decode('ascii')

        nicknames.append(nickname)
        clients.append(client)

        print(f'Nickname of the client is {nickname}!')
        broadcast(f'{nickname} joined the chat!'.encode('ascii'), _client=None)
        client.send('Connected to the server!'.encode('ascii'))

        thread = threading.Thread(target=handle, args=(client,))
        thread.start()

clients = []

```

```

nicknames = []

host = '37.120.249.45'

port = 8413

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((host, port))

server.listen()

print(f"Server Started at port {port} made by gr3_13 Alina Atanasiu")
receive()

```

## 2.2 -print screen cu rularea server-ului

The screenshot shows a VS Code editor window with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with files like `server.py`, `users.json`, and `client.py`. The main editor displays the content of `server.py`, which is a Python script for a simple socket server. The terminal shows the output of running the script, including the message "Server Started at port 8413 made by gr3\_13 Alina Atanasiu" and several connection logs from clients.

```

server.py
43 clients = []
44 nicknames = []
45
46 host = '37.120.249.45'
47 port = 8413
48
49 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
50 server.bind((host, port))
51
52 server.listen()
53
54 print(f"Server Started at port {port} made by gr3_13 Alina Atanasiu")
55 receive()

```

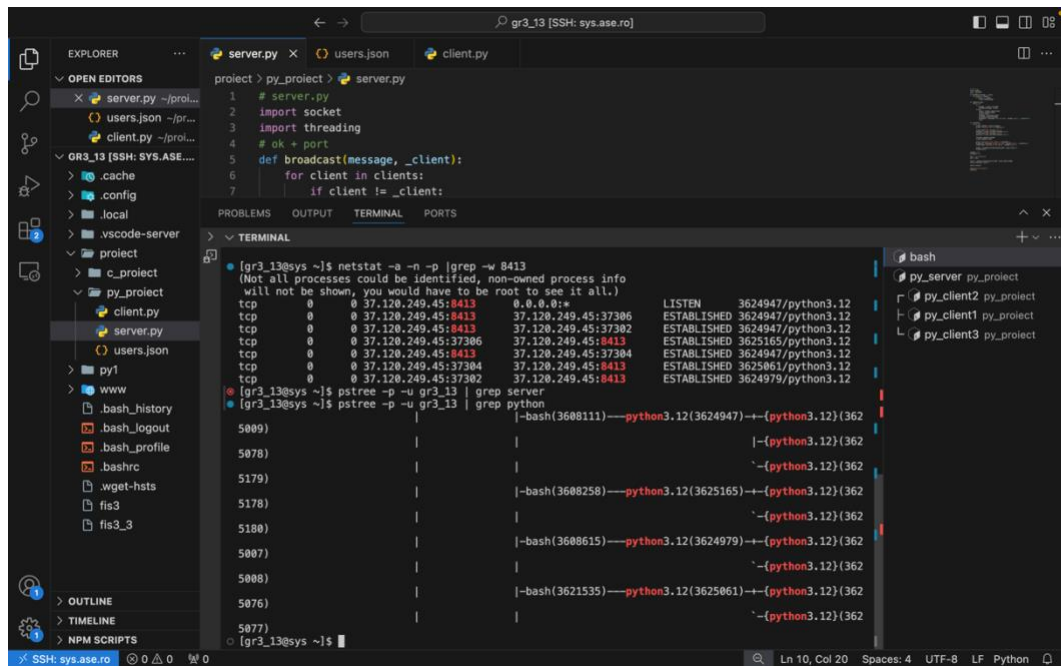
```

[gr3_13@sys py_project]$ python3.12 server.py
Server Started at port 8413 made by gr3_13 Alina Atanasiu
Connected with ('37.120.249.45', 37324)
Nickname of the client is Ana11!
Connected with ('37.120.249.45', 37328)
Nickname of the client is Niku44!
Connected with ('37.120.249.45', 37330)
Nickname of the client is Ion!

```

## 2.3- portul cu identificare protocolului si procesul





### 3. Program client

#### 3.1 -cod comentat

```
import socket
import threading
import json

def check_user(nickname, password):
    with open('users.json', 'r') as file:
        users = json.load(file)
        for user in users:
            if user['nickname'] == nickname and user['password'] == password:
                return True
        return False

nickname = input("Enter your nickname: ")
password = input("Enter your password: ")

if not check_user(nickname, password):
    print("Invalid credentials!")
    exit()

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('37.120.249.45', 8413))

def receive():
```

```

while True:
    try:
        message = client.recv(1024).decode('ascii')
        if message == 'NICK':
            client.send(nickname.encode('ascii'))
        elif message == 'PASS':
            client.send(password.encode('ascii'))
        else:
            print(message)
    except:
        print("An error occurred!")
        client.close()
        break

def write():
    while True:
        message = f'{nickname}: {input("")}'
        client.send(message.encode('ascii'))

receive_thread = threading.Thread(target=receive)
receive_thread.start()

write_thread = threading.Thread(target=write)
write_thread.start()

```

json:

```

[
    {
        "nickname": "Ana11",
        "password": "p123"
    },
    {
        "nickname": "Niku44",
        "password": "p123"
    },
    {
        "nickname": "lon",
        "password": "p123"
    }
]

```

### 3.2 -print screen cu rularea client-ului

The screenshot displays a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project structure with files like `server.py`, `users.json`, and `client.py`. The main editor area has three tabs: `server.py`, `users.json`, and `client.py`. The `client.py` file is open and contains the following Python code:

```
1 # client.py
2 import socket
3 import threading
4 import json
5 # ok + port
6 def check_user(nickname, password):
7     with open('users.json', 'r') as file:
8         users = json.load(file)
9         for user in users:
10             if user['nickname'] == nickname and user['password'] == password:
11                 return True
12             return False
13
14 nickname = input("Enter your nickname: ")
15 password = input("Enter your password: ")
16
17 if not check_user(nickname, password):
18     print("Invalid credentials!")
19     exit()
20
21 client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
22 client.connect(('37.120.249.45', 8413))
23
24 def receive():
25     while True:
```

Below the editor, the TERMINAL panel is active, showing the execution of the client script. It displays a `KeyboardInterrupt` followed by the command `python 3.12 client.py`. The output shows a successful connection to the server and a chat log where three users (Ana11, Niku44, and Ion) join and exchange messages.

```
KeyboardInterrupt:
[gr3_13@sys py_project]$ python 3.12 client.py
Enter your nickname: Niku44
Enter your password: p123
Niku44 joined the chat!
Connected to the server!
Ana11: bine tu?

Enter your nickname: Ana11
Enter your password: p123
Ana11 joined the chat!
Connected to the server!
Ion: ce faci?
Niku44 joined the chat!
bine tu?

on3.12 client.py
Enter your nickname: Ion
Enter your password: p123
Ion joined the chat!
Connected to the server!
ce faci?
Niku44 joined the chat!
Ana11: bine tu?
```

The status bar at the bottom indicates the current position is at line 7, column 42, with 4 spaces, using UTF-8 encoding, LF line endings, and the Python language mode.