Python:

```
#SERVER –mesaj inversat returnat de server clientului
import socket
from _thread import *
import threading
print_lock = threading.Lock()
def threaded(c):
while True:
# datele primite de la client
data = c.recv(1024)
if not data:
print('Bye')
print_lock.release()
break
# inversăm șirul primit de la client
data = data[::-1]
# trimitem înapoi șirul inversat către client
c.send(data)
# închidem conexiunea
c.close()
# definim funcția principală
def Main():
host = ""
port = 8604
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((host, port))
print("socket binded to port", port)
s.listen(5)
```

```
print("socket is listening")
while True:
c, addr = s.accept()
print_lock.acquire()
print('Connected to:', addr[0], ':', addr[1])
start_new_thread(threaded, (c,))
s.close()
if __name__ == '__main__':
Main()
#CLIENT
import socket
def Main():
host = '37.120.249.45'
port = 8604
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
message = "Hi, I'm on!"
while True:
s.send(message.encode('ascii'))
data = s.recv(1024)
print('Received from the server:', str(data.decode('ascii')))
ans = input('\nDo you want to continue (y/n): ')
if ans == 'y':
continue
else:
break
# închidem conexiunea
s.close()
```

```
if __name__ == '__main__':
Main()
```

C:

Trimiterea de mesaje – modificarea mesajului primit prin insertia “*” dupa fiecare caracter din mesaj

```
//SERVER
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define PORT 8604
void addStars(char *message) {
int len = strlen(message);
int i, j = 0;
char newMessage[2 * len + 1]; // Declaram un nou sir pentru a stoca mesajul cu caracterele '*' adaugate
for (i = 0; i < len; ++i) {
newMessage[j++] = message[i];
newMessage[j++] = '*';
}
newMessage[j] = '\0'; // Adăugăm terminatorul de șir
strcpy(message, newMessage);
}
int main() {
int server_fd, new_socket, valread;
```

```
struct sockaddr_in address;

int opt = 1;

int addrlen = sizeof(address);

char buffer[1024] = {0};

// Crearea socket-ului

if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {

perror("Socket failed");

exit(EXIT_FAILURE);

}

// Setarea optiunilor pentru socket

if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt))) {

perror("Setsockopt failed"); // Afisam un mesaj de eroare in caz de esec

exit(EXIT_FAILURE); // Iesim din program cu cod de eroare

}

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"); // Afisam un mesaj de eroare in caz de esec

exit(EXIT_FAILURE); // Iesim din program cu cod de eroare

}

if (listen(server_fd, 3) < 0) {

perror("Listen failed"); // Afisam un mesaj de eroare in caz de esec

exit(EXIT_FAILURE); // Iesim din program cu cod de eroare

}

if ((new_socket = accept(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen))
< 0) {

perror("Accept failed"); // Afisam un mesaj de eroare in caz de esec
exit(EXIT_FAILURE); // Iesim din program cu cod de eroare
}
printf("Conexiune realizata pe portul %d.\n", PORT);
// Citim mesajul de la client
valread = read(new_socket, buffer, 1024);
printf("Mesajul primit de la client: %s\n", buffer);
addStars(buffer);
printf("Mesajul primit modificat: %s\n", buffer);
send(new_socket, buffer, strlen(buffer), 0);
printf("Mesaj trimis inapoi catre client.\n");
close(new_socket);
close(server_fd);
return 0; // Incheiem executia programului cu succes
}
//---------------------------------------------
//CLIENT
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define PORT 8604
#define SERVER_IP "37.120.249.45"
int main() {
int sock = 0, valread;
struct sockaddr_in serv_addr;
char buffer[1024] = {0};
```

```
// Creare socket
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
perror("Socket creation error");
exit(EXIT_FAILURE);
}
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(PORT);
// Convertirea adresei IP in format binar
if (inet_pton(AF_INET, SERVER_IP, &serv_addr.sin_addr) <= 0) {
perror("Invalid address/ Address not supported");
exit(EXIT_FAILURE);
}
// Conectare la server
if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
perror("Connection failed");
exit(EXIT_FAILURE);
}
printf("Introduceti mesaj: ");
fgets(buffer, sizeof(buffer), stdin);
// Trimitere mesaj la server
send(sock, buffer, strlen(buffer), 0);
printf("Mesaj trimis la server.\n");
// Primit si afisare raspuns de la server
valread = read(sock, buffer, 1024);
printf("Raspuns de la server: %s\n", buffer);
close(sock);
return 0;
}
```