LAB1-C

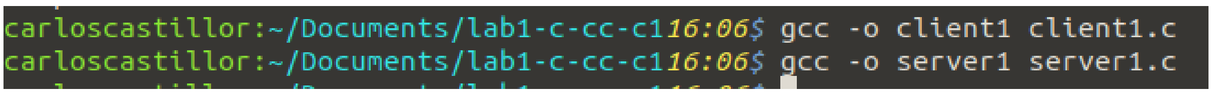
Claudio Maiorana Arvelo, Carlos Castillo Rivero
Githubs: CMaio, carloscastillor

OPERATING SYSTEMS
2020-2021
10 February 2021

Index

PART 1 - Implement Version1 connection oriented

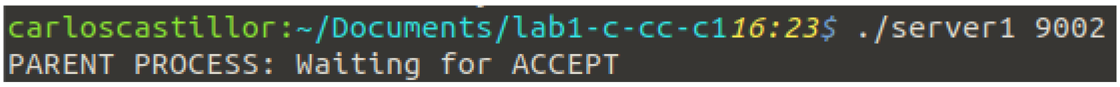
- Screenshots and explanation of the compiling process



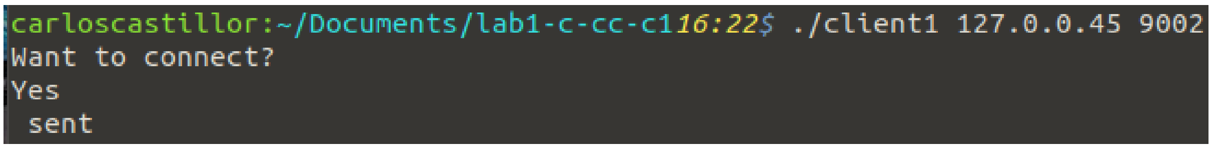
```
carloscastillor:~/Documents/lab1-c-cc-c116:06$ gcc -o client1 client1.c
carloscastillor:~/Documents/lab1-c-cc-c116:06$ gcc -o server1 server1.c
```

In order to compile and generate the programs, the GCC compiler is used.

- Screenshots and explanation of the execution process

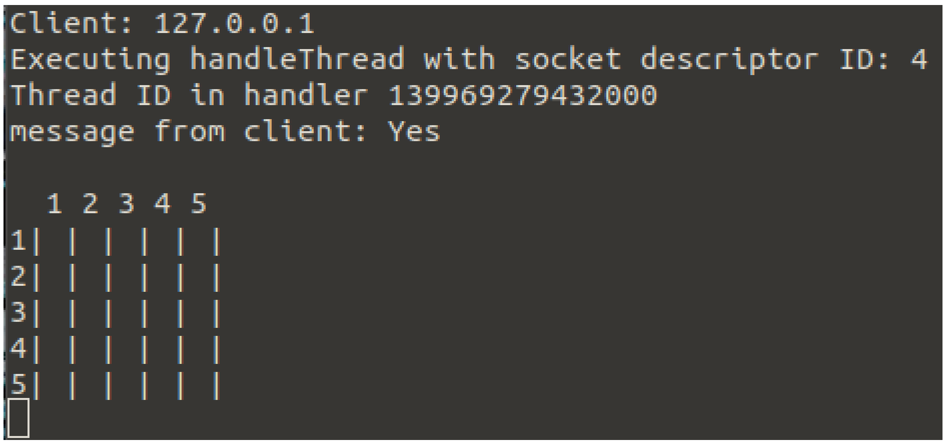


```
carloscastillor:~/Documents/lab1-c-cc-c116:23$ ./server1 9002
PARENT PROCESS: Waiting for ACCEPT
```

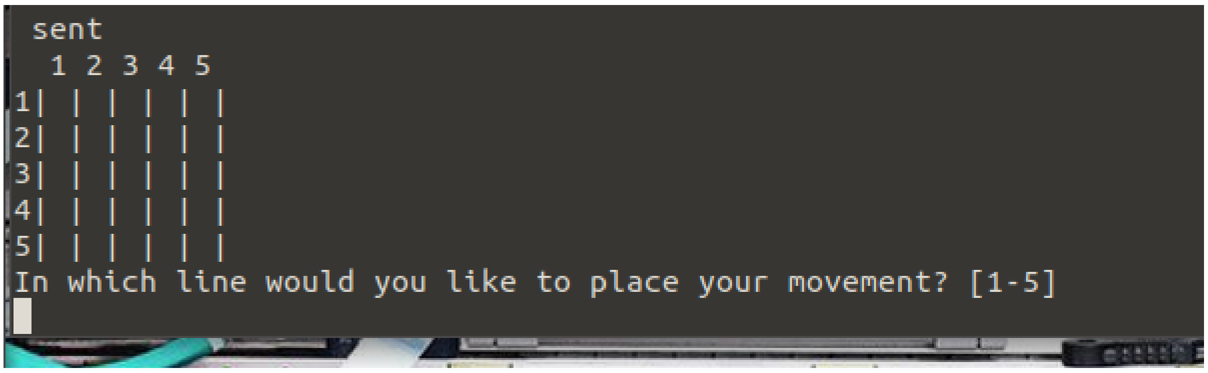


```
carloscastillor:~/Documents/lab1-c-cc-c116:22$ ./client1 127.0.0.45 9002
Want to connect?
Yes
 sent
```

In order to start playing, you must execute the server1 program first, using the command "./server1" and specify a port, then the process will wait for any client to answer.
Once executing a client1 program using the command "./client1" you must specify an ip address and the port to connect, this will be the same specified in the server1 execution. If the constraints are correct then it will ask if you want to connect with the server.

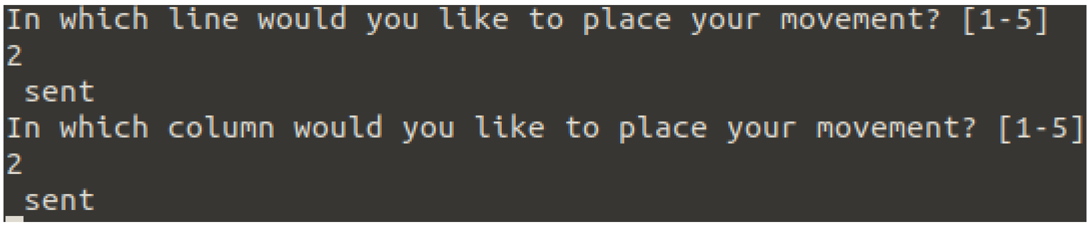


```
Client: 127.0.0.1
Executing handleThread with socket descriptor ID: 4
Thread ID in handler 139969279432000
message from client: Yes

  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
```

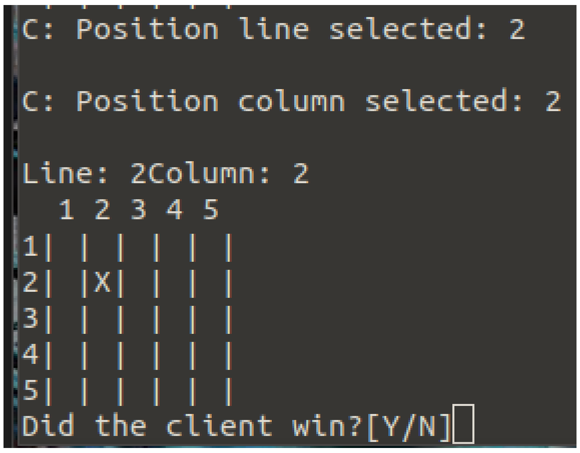


```
 sent
  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
In which line would you like to place your movement? [1-5]
```

If the client types anything, it will send it to the server, which will start the game. First of all it shows the current board to both the client and the server and asks the client to make a move. Each move consists of a combination of a row and a column.

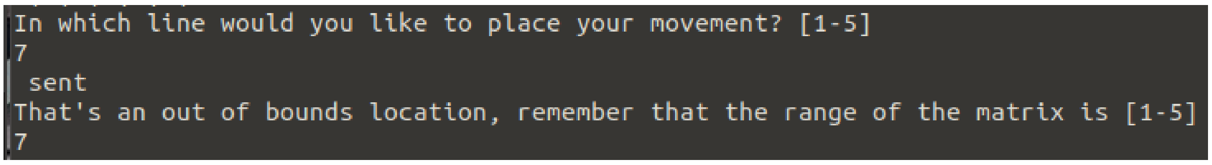


```
In which line would you like to place your movement? [1-5]
2
 sent
In which column would you like to place your movement? [1-5]
2
 sent
```

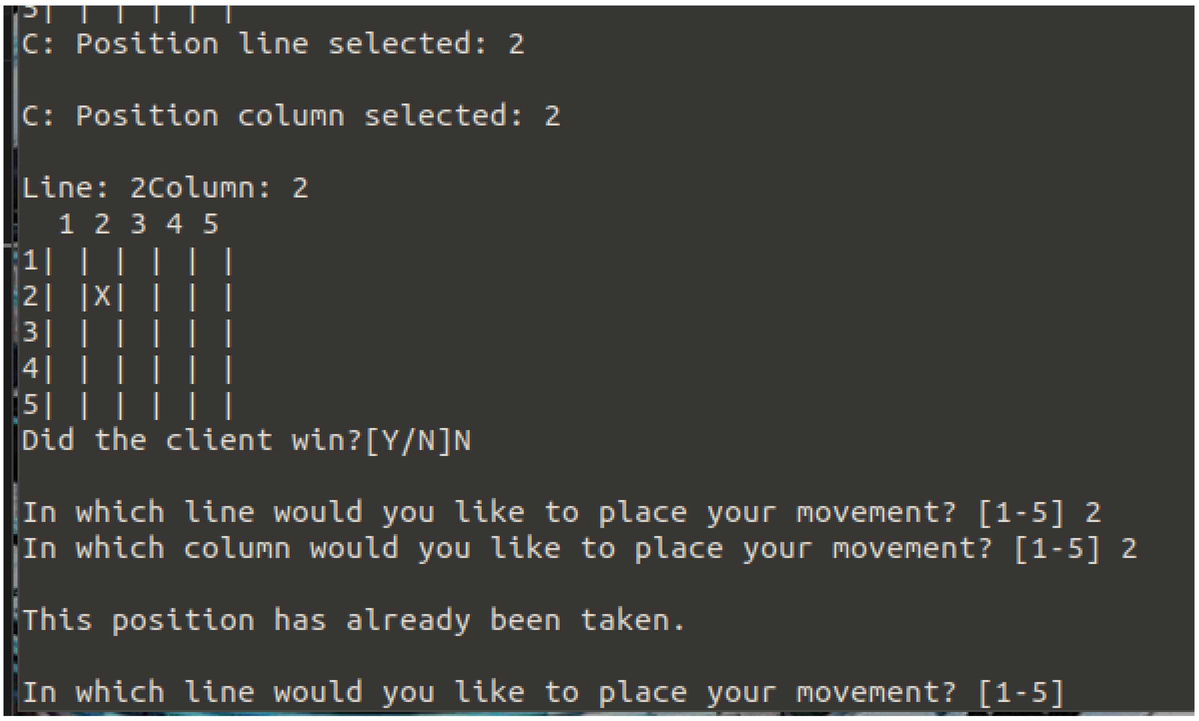


```
C: Position line selected: 2

C: Position column selected: 2

Line: 2Column: 2
  1 2 3 4 5
1| | | | | |
2| |X| | | |
3| | | | | |
4| | | | | |
5| | | | | |
Did the client win?[Y/N]
```

If the client inserts a valid move, it will be sent to the server and it will introduce it inside the board.

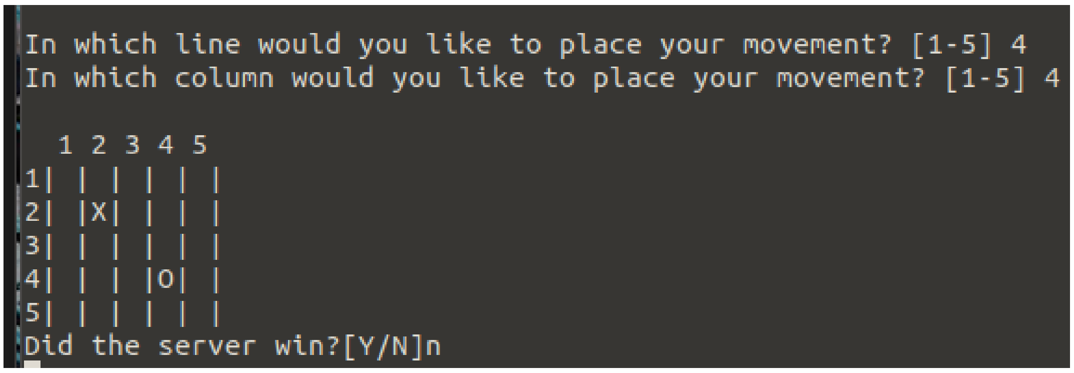


```
In which line would you like to place your movement? [1-5]
7
 sent
That's an out of bounds location, remember that the range of the matrix is [1-5]
7
```



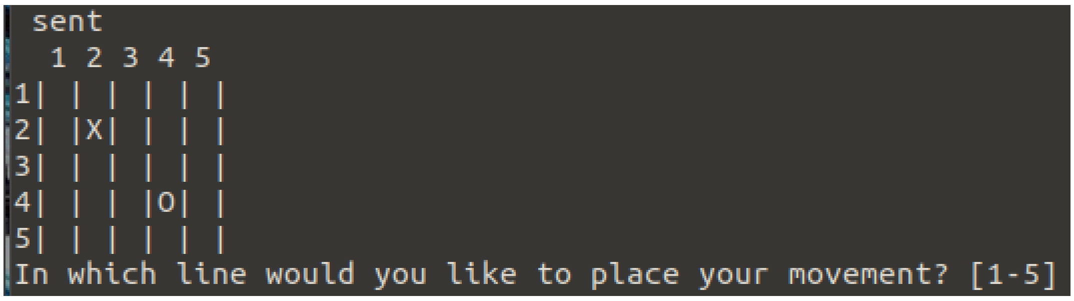
```
C: Position line selected: 2

C: Position column selected: 2

Line: 2Column: 2
  1 2 3 4 5
1| | | | | |
2| |X| | | |
3| | | | | |
4| | | | | |
5| | | | | |
Did the client win?[Y/N]N

In which line would you like to place your movement? [1-5] 2
In which column would you like to place your movement? [1-5] 2

This position has already been taken.

In which line would you like to place your movement? [1-5]
```

If the move is not valid it will ask to insert a new one. If it is an out of bounds location will show image number 1, if it is an already occupied position it will show image number 2.



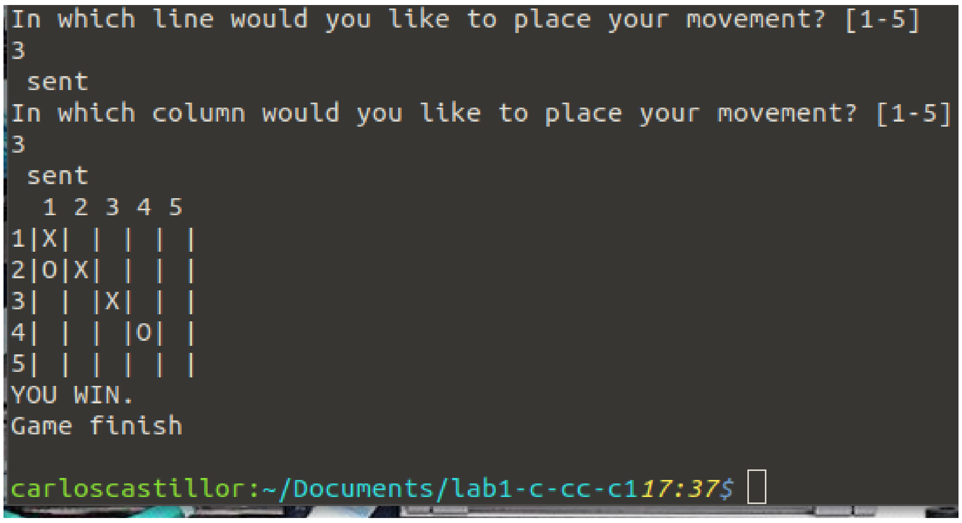
```
In which line would you like to place your movement? [1-5] 4
In which column would you like to place your movement? [1-5] 4

  1 2 3 4 5
1| | | | | |
2| |X| | | |
3| | | | | |
4| | | |O| |
5| | | | | |
Did the server win?[Y/N]n
```

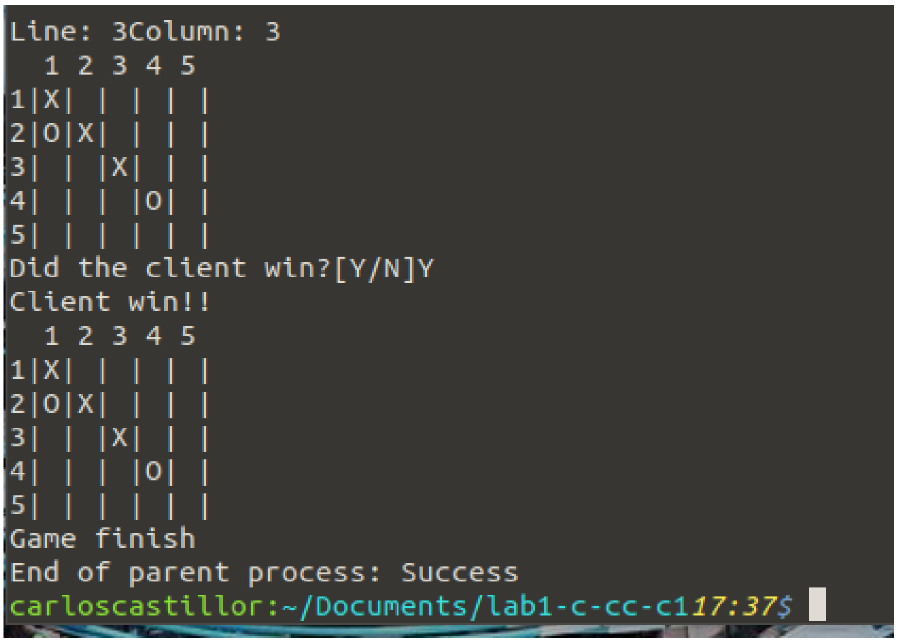
Once the move is already set on the board, the server will ask the user if that client move wins the game or not. If it is not a win move it will ask the server to introduce one itself and if it is valid will ask it if this move wins the game or not as well.


```
 sent
  1 2 3 4 5
1| | | | | |
2| |X| | | |
3| | | | | |
4| | | |O| |
5| | | | | |
In which line would you like to place your movement? [1-5]
```

Then it prints the current board in the client terminal and asks to introduce a new move.

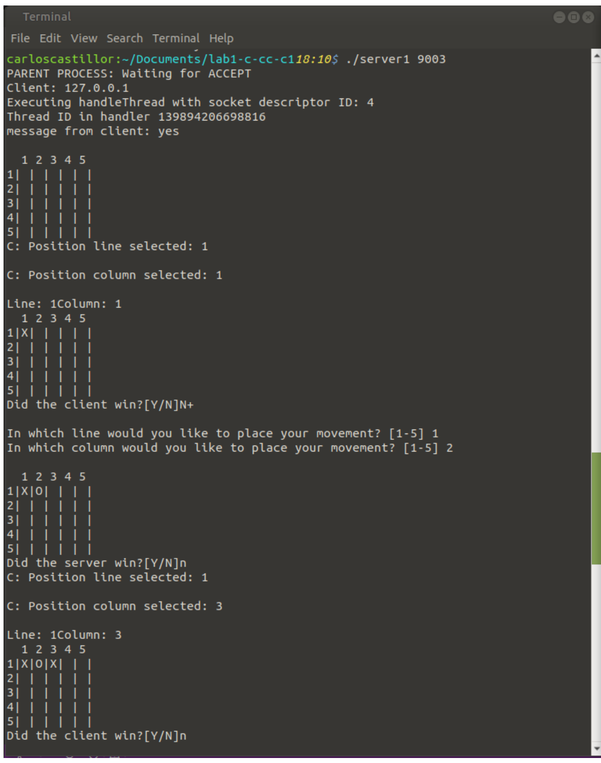
At this point the execution keeps that methodology until it is decided that a player wins.


```
In which line would you like to place your movement? [1-5]
3
 sent
In which column would you like to place your movement? [1-5]
3
 sent
  1 2 3 4 5
1|X| | | | |
2|O|X| | | |
3| | |X| | |
4| | | |O| |
5| | | | | |
YOU WIN.
Game finish

carloscastillor:~/Documents/lab1-c-cc-c117:37$
```

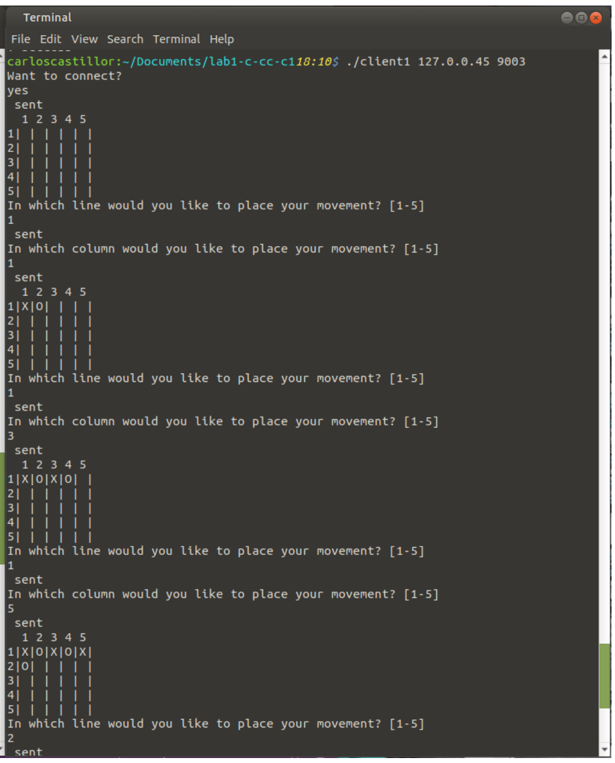

```
Line: 3Column: 3
  1 2 3 4 5
1|X| | | | |
2|O|X| | | |
3| | |X| | |
4| | | |O| |
5| | | | | |
Did the client win?[Y/N]Y
Client win!!
  1 2 3 4 5
1|X| | | | |
2|O|X| | | |
3| | |X| | |
4| | | |O| |
5| | | | | |
Game finish
End of parent process: Success
carloscastillor:~/Documents/lab1-c-cc-c117:37$
```

If either the client or the server is the winner it will show a custom message and the final board result in each terminal. And finally the execution process finishes.

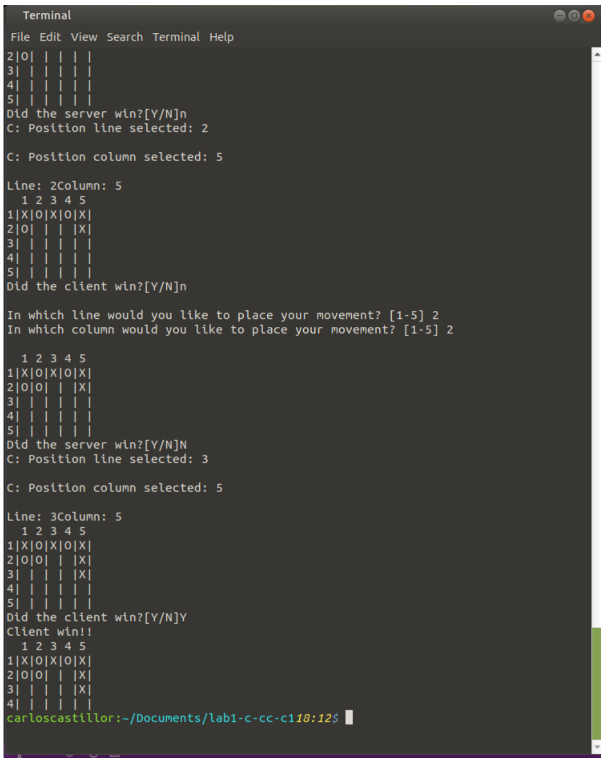
- Other example:



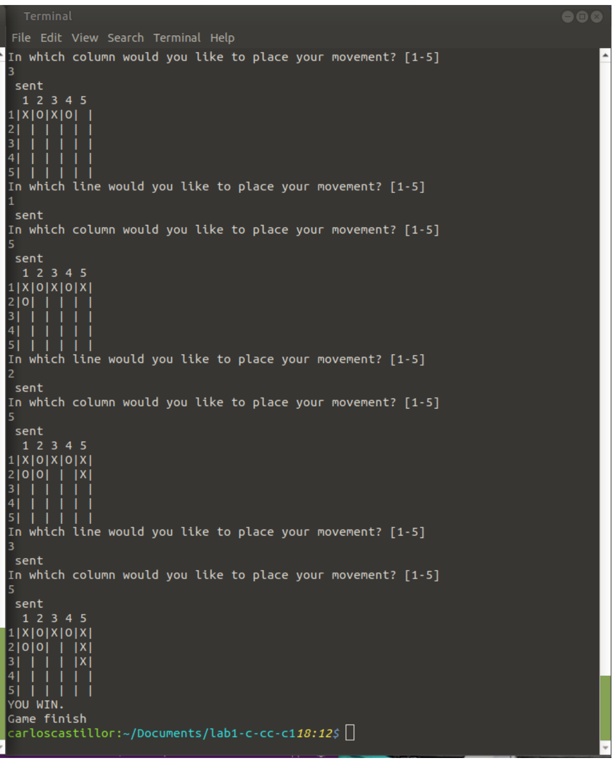
```
carloscastillor:~/Documents/lab1-c-cc-c118:10$ ./server1 9003
PARENT PROCESS: Waiting for ACCEPT
Client: 127.0.0.1
Executing handleThread with socket descriptor ID: 4
Thread ID in handler 139894206698816
message from client: yes

  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
C: Position line selected: 1

C: Position column selected: 1

Line: 1Column: 1
  1 2 3 4 5
1|X| | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
Did the client win?[Y/N]N+

In which line would you like to place your movement? [1-5] 1
In which column would you like to place your movement? [1-5] 2

  1 2 3 4 5
1|X|O| | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
Did the server win?[Y/N]n
C: Position line selected: 1

C: Position column selected: 3

Line: 1Column: 3
  1 2 3 4 5
1|X|O|X| | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
Did the client win?[Y/N]n
```



```
carloscastillor:~/Documents/lab1-c-cc-c118:10$ ./client1 127.0.0.45 9003
Want to connect?
yes
 sent
  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
In which line would you like to place your movement? [1-5]
1
 sent
In which column would you like to place your movement? [1-5]
1
 sent
  1 2 3 4 5
1|X|O| | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
In which line would you like to place your movement? [1-5]
1
 sent
In which column would you like to place your movement? [1-5]
3
 sent
  1 2 3 4 5
1|X|O|X|O| |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
In which line would you like to place your movement? [1-5]
1
 sent
In which column would you like to place your movement? [1-5]
5
 sent
  1 2 3 4 5
1|X|O|X|O|X|
2|O| | | | |
3| | | | | |
4| | | | | |
5| | | | | |
In which line would you like to place your movement? [1-5]
2
 sent
```



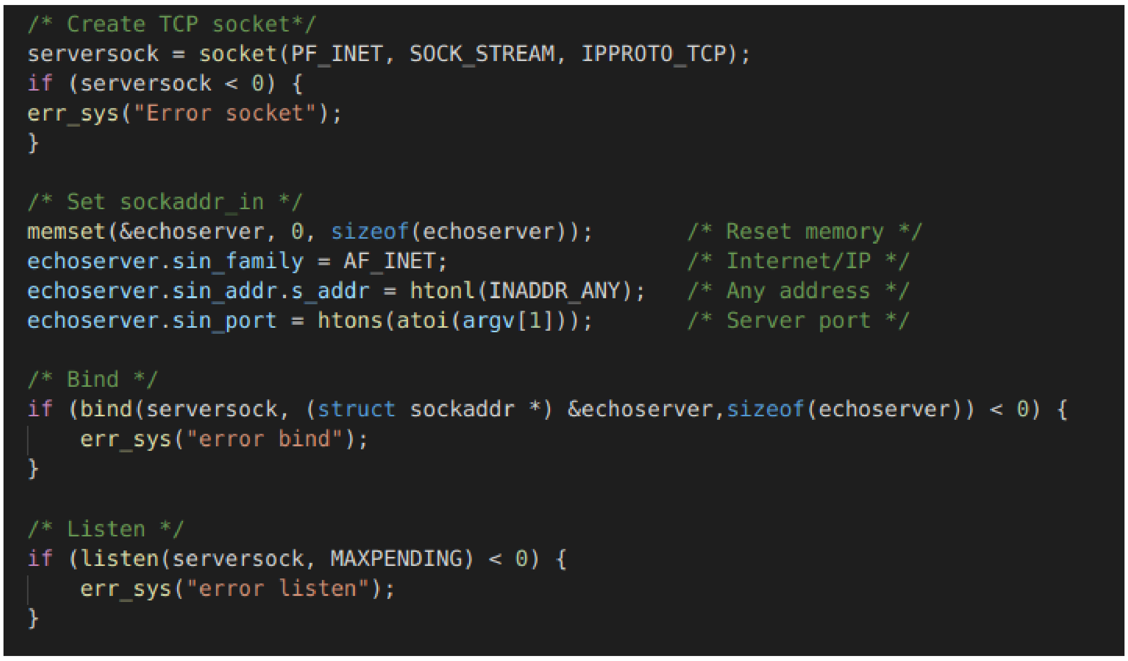
```
2|O| | | | |
3| | | | | |
4| | | | | |
5| | | | | |
Did the server win?[Y/N]n
C: Position line selected: 2

C: Position column selected: 5

Line: 2Column: 5
  1 2 3 4 5
1|X|O|X|O|X|
2|O| | | |X|
3| | | | | |
4| | | | | |
5| | | | | |
Did the client win?[Y/N]n

In which line would you like to place your movement? [1-5] 2
In which column would you like to place your movement? [1-5] 2

  1 2 3 4 5
1|X|O|X|O|X|
2|O|O| | |X|
3| | | | | |
4| | | | | |
5| | | | | |
Did the server win?[Y/N]N
C: Position line selected: 3

C: Position column selected: 5

Line: 3Column: 5
  1 2 3 4 5
1|X|O|X|O|X|
2|O|O| | |X|
3| | | | |X|
4| | | | | |
5| | | | | |
Did the client win?[Y/N]Y
Client win!!
  1 2 3 4 5
1|X|O|X|O|X|
2|O|O| | |X|
3| | | | |X|
4| | | | | |
carloscastillor:~/Documents/lab1-c-cc-c118:12$
```



```
In which column would you like to place your movement? [1-5]
3
 sent
  1 2 3 4 5
1|X|O|X|O| |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
In which line would you like to place your movement? [1-5]
1
 sent
In which column would you like to place your movement? [1-5]
5
 sent
  1 2 3 4 5
1|X|O|X|O|X|
2|O| | | | |
3| | | | | |
4| | | | | |
5| | | | | |
In which line would you like to place your movement? [1-5]
2
 sent
In which column would you like to place your movement? [1-5]
5
 sent
  1 2 3 4 5
1|X|O|X|O|X|
2|O|O| | |X|
3| | | | | |
4| | | | | |
5| | | | | |
In which line would you like to place your movement? [1-5]
3
 sent
In which column would you like to place your movement? [1-5]
5
 sent
  1 2 3 4 5
1|X|O|X|O|X|
2|O|O| | |X|
3| | | | |X|
4| | | | | |
5| | | | | |
YOU WIN.
Game finish
carloscastillor:~/Documents/lab1-c-cc-c118:12$
```

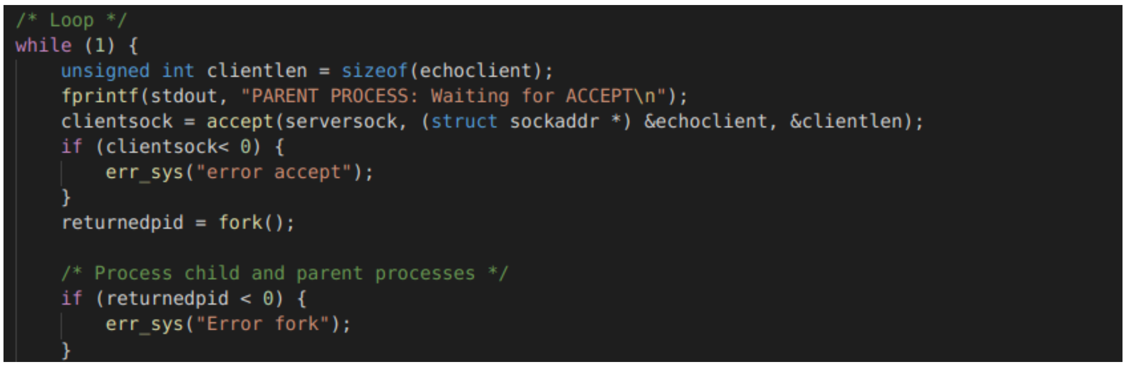
- Comments on the relevant parts of the code.

Server1



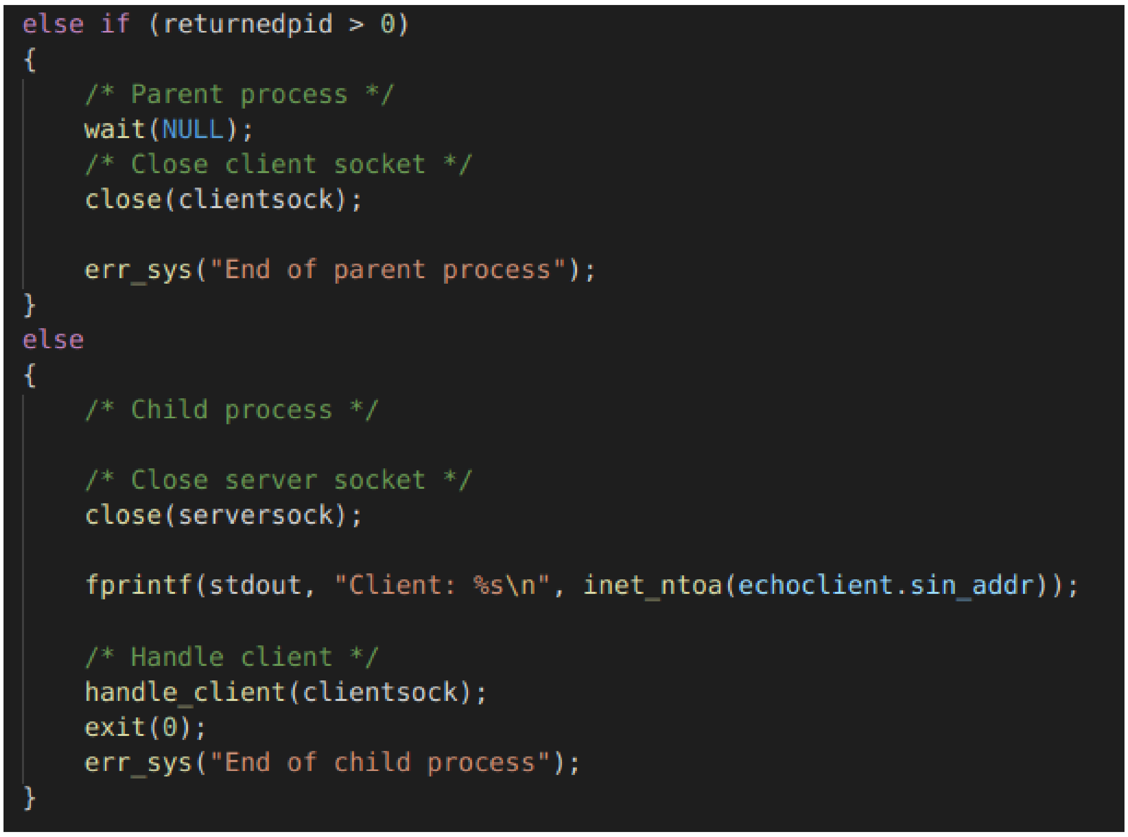
```
/* Create TCP socket*/
serversock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if (serversock < 0) {
err_sys("Error socket");
}

/* Set sockaddr_in */
memset(&echoserver, 0, sizeof(echoserver));       /* Reset memory */
echoserver.sin_family = AF_INET;                  /* Internet/IP */
echoserver.sin_addr.s_addr = htonl(INADDR_ANY);   /* Any address */
echoserver.sin_port = htons(atoi(argv[1]));       /* Server port */

/* Bind */
if (bind(serversock, (struct sockaddr *) &echoserver,sizeof(echoserver)) < 0) {
    err_sys("error bind");
}

/* Listen */
if (listen(serversock, MAXPENDING) < 0) {
    err_sys("error listen");
}
```

About the server code here we can say that a socket is created for the server. To avoid problems, we reset the memory, set the IP, specify which ips can access and the port of the server and wait for a connection .



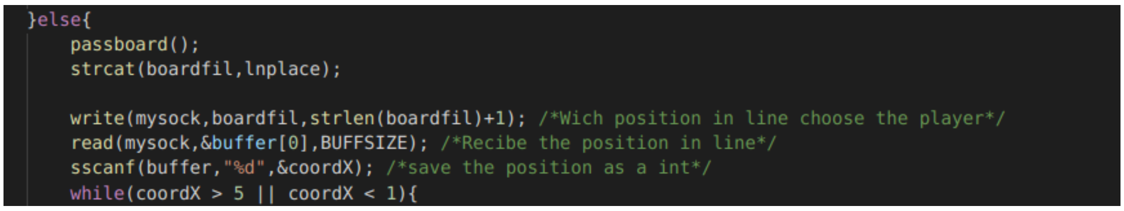
```
/* Loop */
while (1) {
    unsigned int clientlen = sizeof(echoclient);
    fprintf(stdout, "PARENT PROCESS: Waiting for ACCEPT\n");
    clientsock = accept(serversock, (struct sockaddr *) &echoclient, &clientlen);
    if (clientsock< 0) {
        err_sys("error accept");
    }
    returnedpid = fork();

    /* Process child and parent processes */
    if (returnedpid < 0) {
        err_sys("Error fork");
    }
```

When a client tries to connect to the server, we create a socket for it and start a fork for the execution of the game.



```c
else if (returnedpid > 0)
{
    /* Parent process */
    wait(NULL);
    /* Close client socket */
    close(clientsock);

    err_sys("End of parent process");
}
else
{
    /* Child process */

    /* Close server socket */
    close(serversock);

    fprintf(stdout, "Client: %s\n", inet_ntoa(echoclient.sin_addr));

    /* Handle client */
    handle_client(clientsock);
    exit(0);
    err_sys("End of child process");
}
```

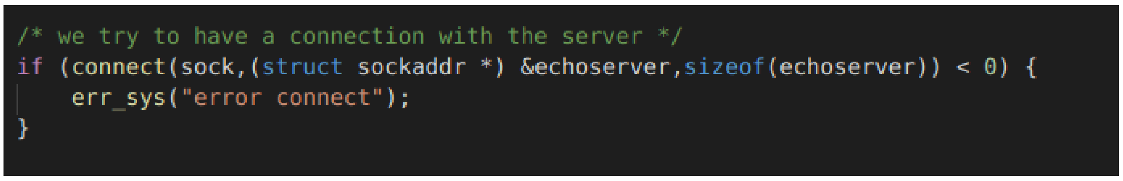
When the child process is created the game is started and when the game finishes we close the socket of the client and, because the client process is finished, the wait(NULL) end and the socket of the server is closed too.



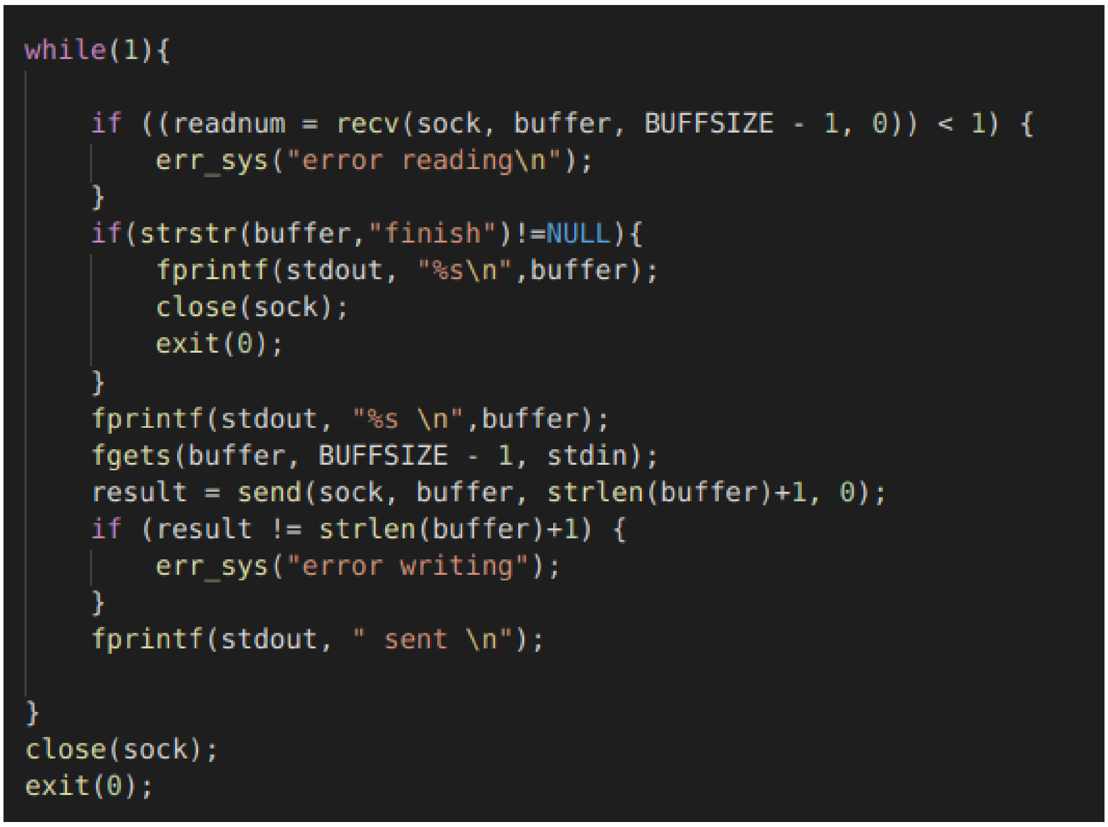
```c
}else{
    passboard();
    strcat(boardfil,lnplace);

    write(mysock,boardfil,strlen(boardfil)+1); /*Wich position in line choose the player*/
    read(mysock,&buffer[0],BUFFSIZE); /*Recibe the position in line*/
    sscanf(buffer,"%d",&coordX); /*save the position as a int*/
    while(coordX > 5 || coordX < 1){
```

Here we can see the part of the code that has the implementation of the game. This part is the management that the server has to do when it's the turn of the client. First we add the current board in a string, then we concatenate this string with a string that contains the line of the next movement asked and then this current board is sent to the client with the method write. Later we wait until the read method receives the line number and parse it to an int to check later if the move is correct along with the column number.

Client1



```c
/* we try to have a connection with the server */
if (connect(sock,(struct sockaddr *) &echoserver,sizeof(echoserver)) < 0) {
    err_sys("error connect");
}
```

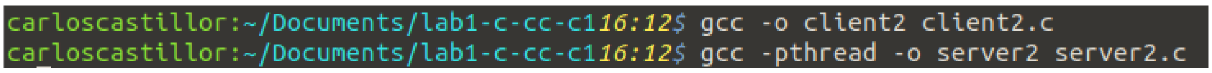


```c
while(1){

    if ((readnum = recv(sock, buffer, BUFFSIZE - 1, 0)) < 1) {
        err_sys("error reading\n");
    }
    if(strstr(buffer,"finish")!=NULL){
        fprintf(stdout, "%s\n",buffer);
        close(sock);
        exit(0);
    }
    fprintf(stdout, "%s \n",buffer);
    fgets(buffer, BUFFSIZE - 1, stdin);
    result = send(sock, buffer, strlen(buffer)+1, 0);
    if (result != strlen(buffer)+1) {
        err_sys("error writing");
    }
    fprintf(stdout, " sent \n");

}
close(sock);
exit(0);
```

In the case of the code of the client it's more or less the same as the server in terms of connectivity but one thing to say is that when the configuration of the connection that the client socket has it tries to connect to the server and if the client can't it close the program , the only thing to remark is this screenshot above. This loop is for the part when the game is running. Once receiving a string, look if the string has the word “finish”, meaning that the game has ended and then the client socket can be closed, if it doesn't contain the word “finish” then it will ask to write a number for the new move.

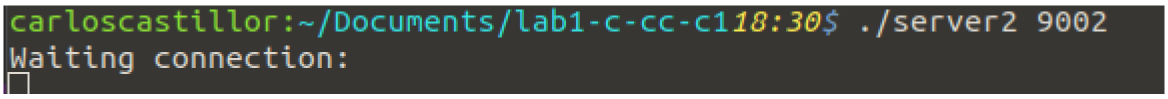
PART 2 - Implement Version2 connection oriented

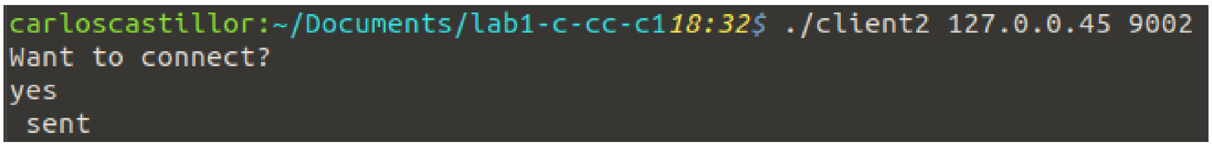
- Screenshots and explanation of the compiling process


```
carloscastillor:~/Documents/lab1-c-cc-c116:12$ gcc -o client2 client2.c
carloscastillor:~/Documents/lab1-c-cc-c116:12$ gcc -pthread -o server2 server2.c
```

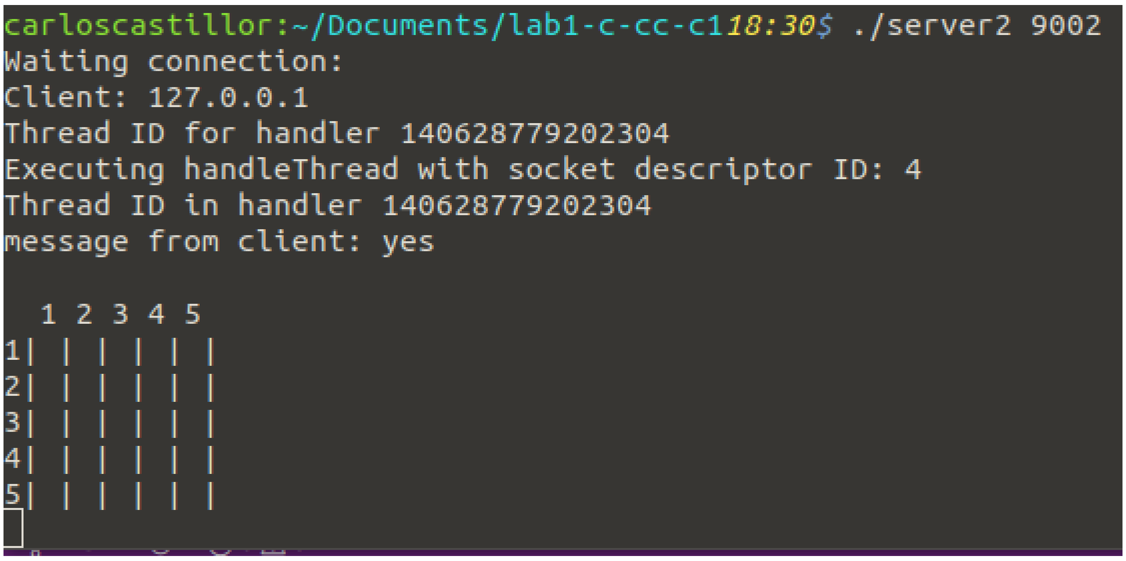
In this part the GCC compiler is also used to compile and generate the programs, but in order to compile the server2.c file another option in the command is used. This other option is “-pthread”, and is necessary because the compiler needs to know if this library is used in the implementation of the code.

- Screenshots and explanation of the execution process

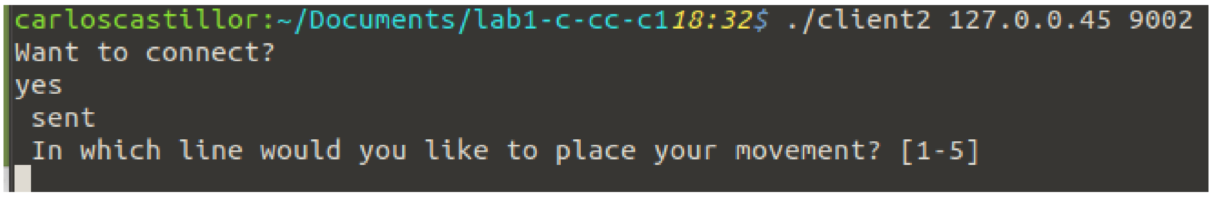

```
carloscastillor:~/Documents/lab1-c-cc-c118:30$ ./server2 9002
Waiting connection:
```


```
carloscastillor:~/Documents/lab1-c-cc-c118:32$ ./client2 127.0.0.45 9002
Want to connect?
yes
 sent
```

In order to start playing, you must execute the server2 program first, using the command “./server2” and specify a port; then the process will wait for any client to answer.
Once executing a client2 program using the command “./client2” you must specify an ip address and the port to connect, this will be the same specified in the server2 execution. If the constraints are correct then it will ask if you want to connect with the server.

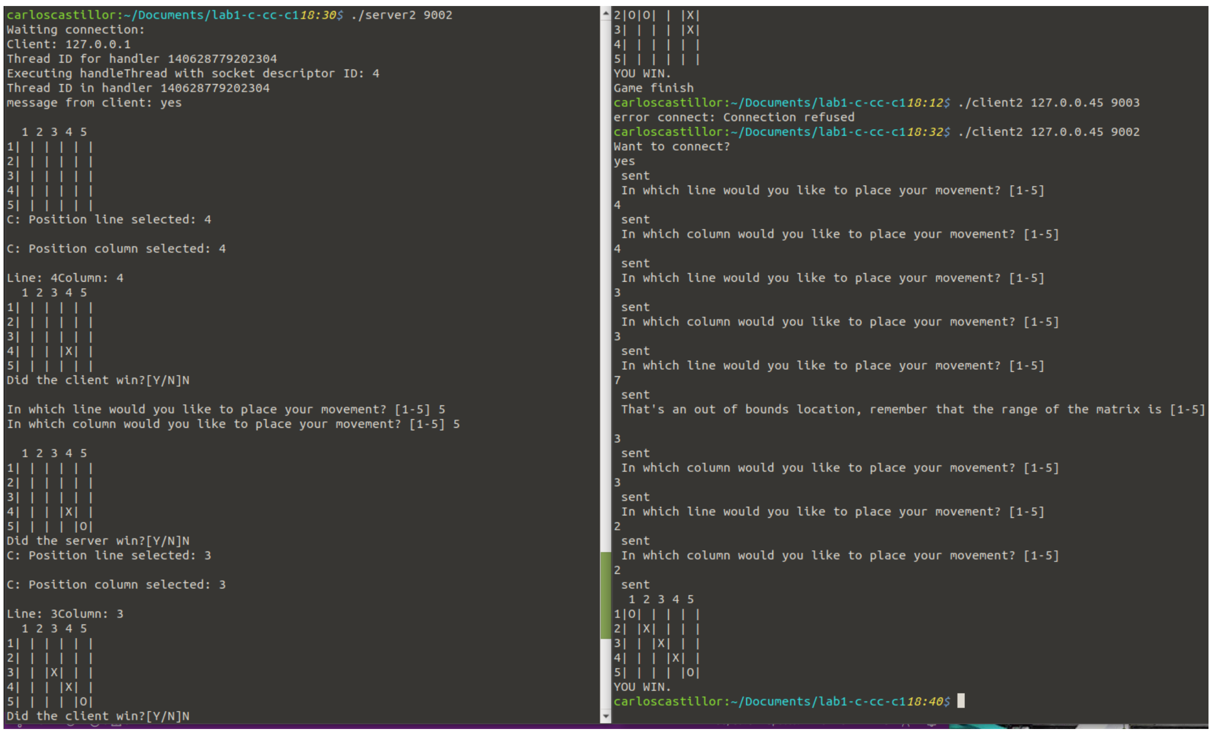


```
carloscastillor:~/Documents/lab1-c-cc-c118:30$ ./server2 9002
Waiting connection:
Client: 127.0.0.1
Thread ID for handler 140628779202304
Executing handleThread with socket descriptor ID: 4
Thread ID in handler 140628779202304
message from client: yes

  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
```



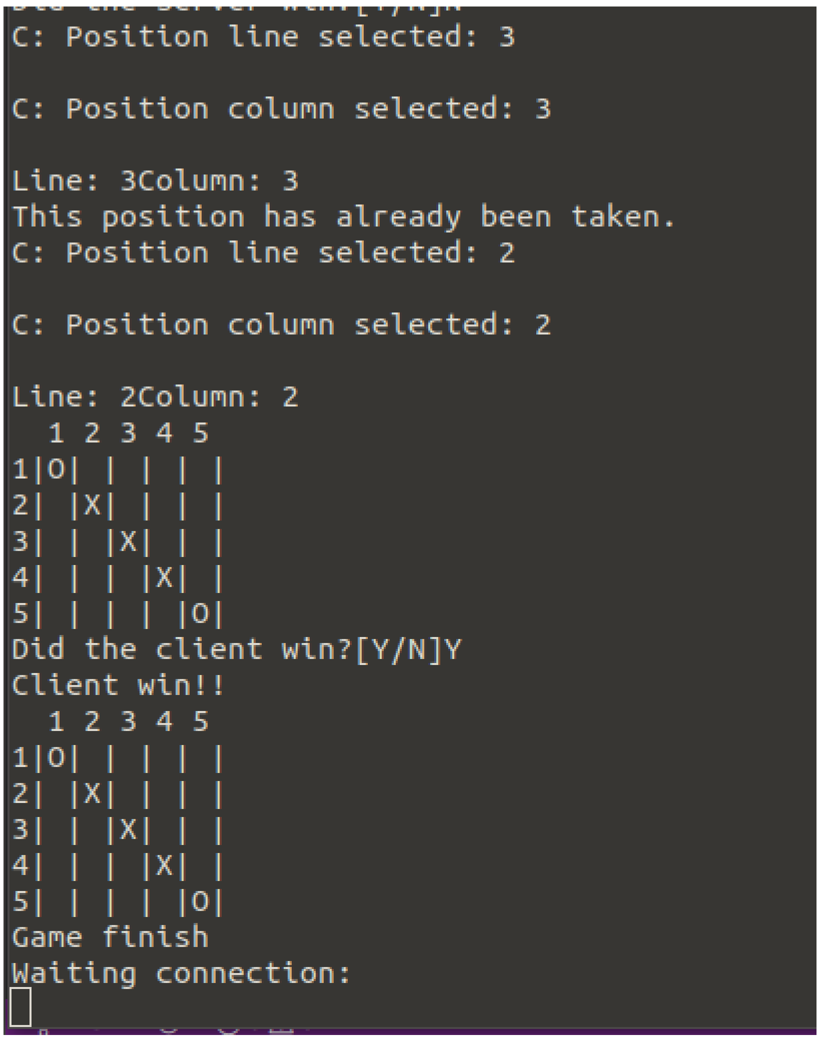
```
carloscastillor:~/Documents/lab1-c-cc-c118:32$ ./client2 127.0.0.45 9002
Want to connect?
yes
 sent
 In which line would you like to place your movement? [1-5]
```

Once answered the game will start, it will print in the server terminal the current board and ask the client to enter a move.



```
carloscastillor:~/Documents/lab1-c-cc-c118:30$ ./server2 9002
Waiting connection:
Client: 127.0.0.1
Thread ID for handler 140628779202304
Executing handleThread with socket descriptor ID: 4
Thread ID in handler 140628779202304
message from client: yes

  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
C: Position line selected: 4

C: Position column selected: 4

Line: 4Column: 4
  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | | | | |
4| | | |X| |
5| | | | | |
Did the client win?[Y/N]N

In which line would you like to place your movement? [1-5] 5
In which column would you like to place your movement? [1-5] 5

  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | | | | |
4| | | |X| |
5| | | | |O|
Did the server win?[Y/N]N
C: Position line selected: 3

C: Position column selected: 3

Line: 3Column: 3
  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | |X| | |
4| | | |X| |
5| | | | |O|
Did the client win?[Y/N]N
```

```
2|O|O| | |X|
3| | | | |X|
4| | | | | |
5| | | | | |
YOU WIN.
Game finish
carloscastillor:~/Documents/lab1-c-cc-c118:12$ ./client2 127.0.0.45 9003
error connect: Connection refused
carloscastillor:~/Documents/lab1-c-cc-c118:32$ ./client2 127.0.0.45 9002
Want to connect?
yes
 sent
 In which line would you like to place your movement? [1-5]
4
 sent
 In which column would you like to place your movement? [1-5]
4
 sent
 In which line would you like to place your movement? [1-5]
3
 sent
 In which column would you like to place your movement? [1-5]
3
 sent
 In which line would you like to place your movement? [1-5]
7
 sent
 That's an out of bounds location, remember that the range of the matrix is [1-5]

3
 sent
 In which column would you like to place your movement? [1-5]
3
 sent
 In which line would you like to place your movement? [1-5]
2
 sent
 In which column would you like to place your movement? [1-5]
2
 sent
  1 2 3 4 5
1|O| | | | |
2| |X| | | |
3| | |X| | |
4| | | |X| |
5| | | | |O|
YOU WIN.
carloscastillor:~/Documents/lab1-c-cc-c118:40$
```



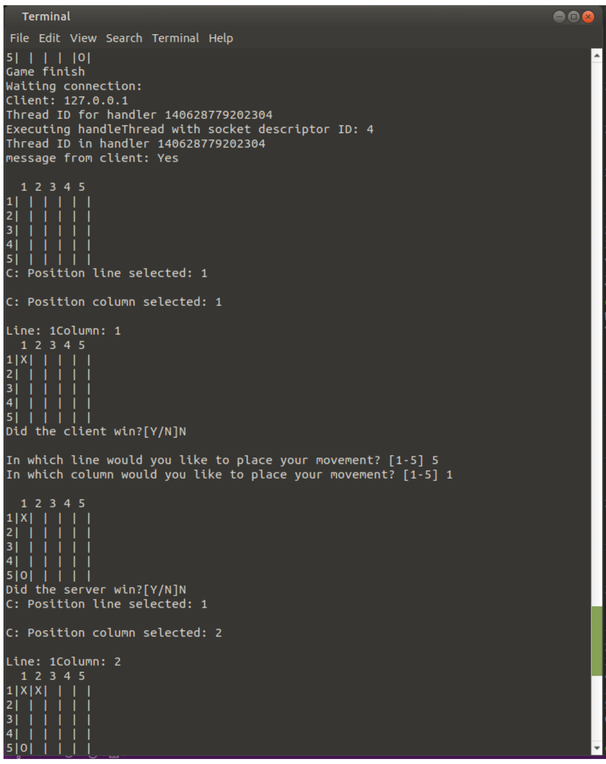
```
C: Position line selected: 3

C: Position column selected: 3

Line: 3Column: 3
This position has already been taken.
C: Position line selected: 2

C: Position column selected: 2

Line: 2Column: 2
  1 2 3 4 5
1|O| | | | |
2| |X| | | |
3| | |X| | |
4| | | |X| |
5| | | | |O|
Did the client win?[Y/N]Y
Client win!!
  1 2 3 4 5
1|O| | | | |
2| |X| | | |
3| | |X| | |
4| | | |X| |
5| | | | |O|
Game finish
Waiting connection:
```

All the gameplay logic is the same as in part1, the only difference is that the client is never shown the board game until the game finishes, so he will play blindly.

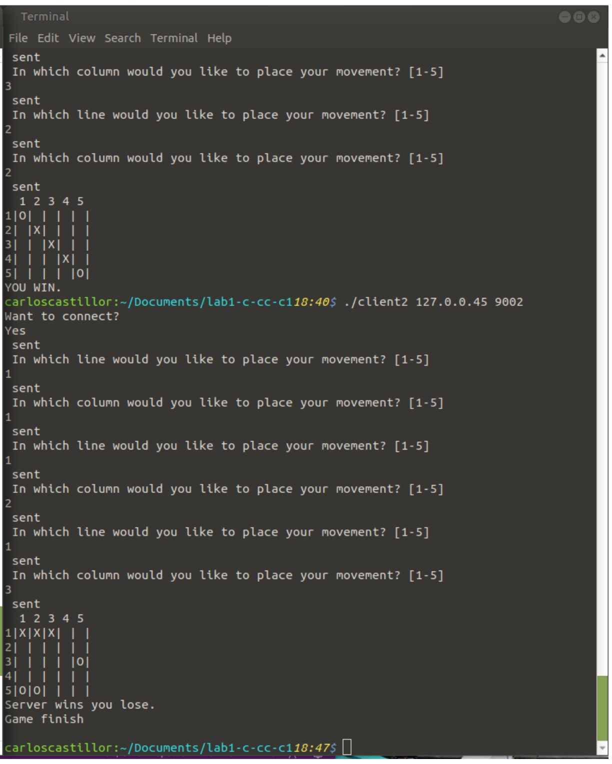
Also If either the client or the server is the winner it will show a custom message and the final board result in each terminal. And finally the execution process of the client will end, but the server one will keep running, waiting for a connection.

- Other Examples:

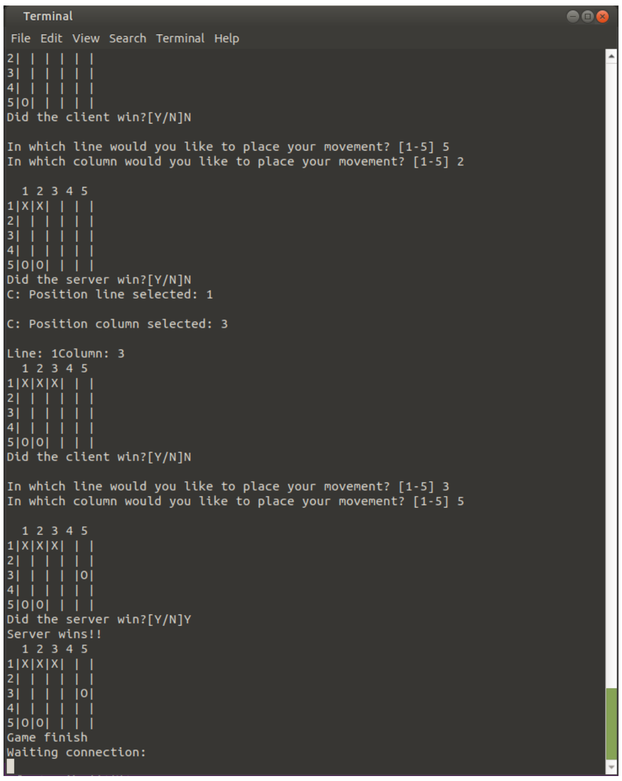
Prove that you can play again.



```
5| | | | |O|
Game finish
Waiting connection:
Client: 127.0.0.1
Thread ID for handler 140628779202304
Executing handleThread with socket descriptor ID: 4
Thread ID in handler 140628779202304
message from client: Yes

  1 2 3 4 5
1| | | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
C: Position line selected: 1

C: Position column selected: 1

Line: 1Column: 1
  1 2 3 4 5
1|X| | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5| | | | | |
Did the client win?[Y/N]N

In which line would you like to place your movement? [1-5] 5
In which column would you like to place your movement? [1-5] 1

  1 2 3 4 5
1|X| | | | |
2| | | | | |
3| | | | | |
4| | | | | |
5|O| | | | |
Did the server win?[Y/N]N
C: Position line selected: 1

C: Position column selected: 2

Line: 1Column: 2
  1 2 3 4 5
1|X|X| | | |
2| | | | | |
3| | | | | |
4| | | | | |
5|O| | | | |
```

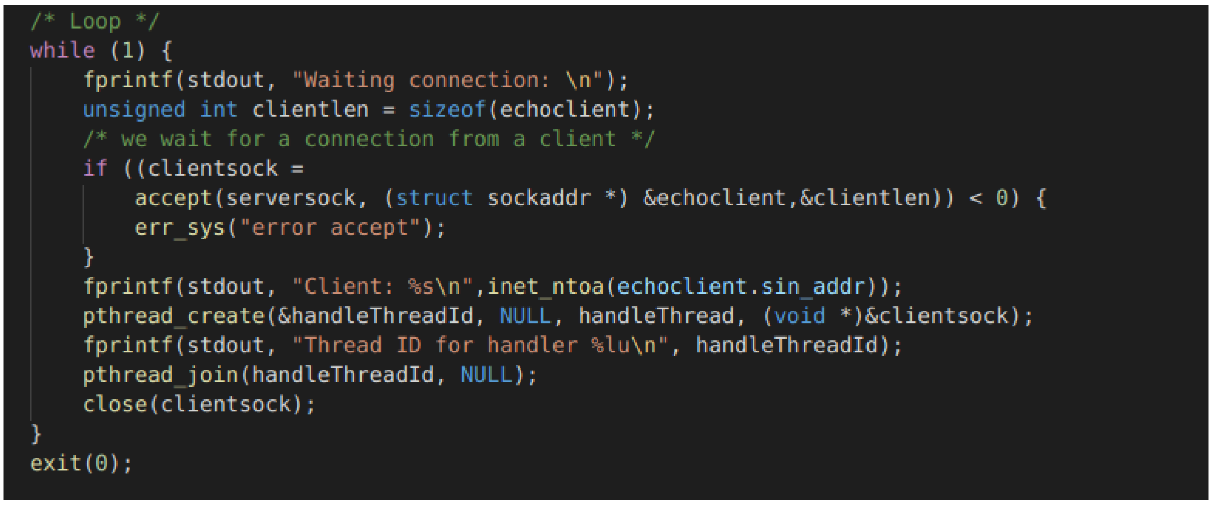


```
 sent
 In which column would you like to place your movement? [1-5]
3
 sent
 In which line would you like to place your movement? [1-5]
2
 sent
 In which column would you like to place your movement? [1-5]
2
 sent
  1 2 3 4 5
1|O| | | | |
2| |X| | | |
3| | |X| | |
4| | | |X| |
5| | | | |O|
YOU WIN.
carloscastillor:~/Documents/lab1-c-cc-c118:40$ ./client2 127.0.0.45 9002
Want to connect?
Yes
 sent
 In which line would you like to place your movement? [1-5]
1
 sent
 In which column would you like to place your movement? [1-5]
1
 sent
 In which line would you like to place your movement? [1-5]
1
 sent
 In which column would you like to place your movement? [1-5]
2
 sent
 In which line would you like to place your movement? [1-5]
1
 sent
 In which column would you like to place your movement? [1-5]
3
 sent
  1 2 3 4 5
1|X|X|X| | |
2| | | | | |
3| | | | |O|
4| | | | | |
5|O|O| | | |
Server wins you lose.
Game finish

carloscastillor:~/Documents/lab1-c-cc-c118:47$ 
```

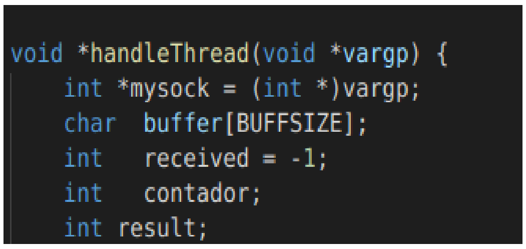


```
2| | | | | |
3| | | | | |
4| | | | | |
5|O| | | | |
Did the client win?[Y/N]N

In which line would you like to place your movement? [1-5] 5
In which column would you like to place your movement? [1-5] 2

  1 2 3 4 5
1|X|X| | | |
2| | | | | |
3| | | | | |
4| | | | | |
5|O|O| | | |
Did the server win?[Y/N]N
C: Position line selected: 1

C: Position column selected: 3

Line: 1Column: 3
  1 2 3 4 5
1|X|X|X| | |
2| | | | | |
3| | | | | |
4| | | | | |
5|O|O| | | |
Did the client win?[Y/N]N

In which line would you like to place your movement? [1-5] 3
In which column would you like to place your movement? [1-5] 5

  1 2 3 4 5
1|X|X|X| | |
2| | | | | |
3| | | | |O|
4| | | | | |
5|O|O| | | |
Did the server win?[Y/N]Y
Server wins!!
  1 2 3 4 5
1|X|X|X| | |
2| | | | | |
3| | | | |O|
4| | | | | |
5|O|O| | | |
Game finish
Waiting connection:
```

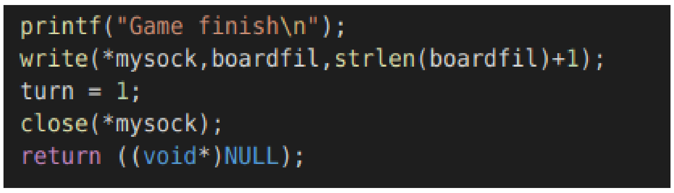
- Comments on the relevant parts of the code.



```
/* Loop */
while (1) {
    fprintf(stdout, "Waiting connection: \n");
    unsigned int clientlen = sizeof(echoclient);
    /* we wait for a connection from a client */
    if ((clientsock =
        accept(serversock, (struct sockaddr *) &echoclient,&clientlen)) < 0) {
        err_sys("error accept");
    }
    fprintf(stdout, "Client: %s\n",inet_ntoa(echoclient.sin_addr));
    pthread_create(&handleThreadId, NULL, handleThread, (void *)&clientsock);
    fprintf(stdout, "Thread ID for handler %lu\n", handleThreadId);
    pthread_join(handleThreadId, NULL);
    close(clientsock);
}
exit(0);
```



```
void *handleThread(void *vargp) {
    int *mysock = (int *)vargp;
    char  buffer[BUFFSIZE];
    int   received = -1;
    int   contador;
    int result;
```



```
printf("Game finish\n");
write(*mysock,boardfil,strlen(boardfil)+1);
turn = 1;
close(*mysock);
return ((void*)NULL);
```

The code for part 2 is mostly the same as the part 1, so there are only two things to talk about. First of all, in the loop for the server the process to play the game is done using threads instead of forks like in the part one. We wait for a client to connect to the server, create the socket and then use the function pthread_create to start a process to play the game. When the game ends we use the function pthread_join to terminate the process and close the clientsocket.
Then the other thing to remark is that the function handleThread it's startet like it's seen in the screenshot because of the function pthread_created and then to finish the game as we see in the other screenshot we return a void as null only to close correctly the client and wait for another connection in the server.