Championship - Technical Report

Soci Lucian

1 Introduction

This project holds significant importance as it seamlessly combines networking functionalities with user-friendly features, fostering an accessible experience for participants. By providing a platform for clients to login and register to a centralized database, we have created a secure and organized environment for users to engage in various championships. The user-friendly design not only facilitates effortless registration and login processes but also enhances overall accessibility, ensuring that participants can easily navigate through the system. This simplicity is crucial in attracting a diverse user base, as it empowers individuals with varying technical backgrounds to participate in championships without unnecessary complications. In essence, this project promotes inclusivity and engagement, making it a valuable contribution to the realm of online gaming and database management.

2 Technologies used

In my implementation, the client-server application depends on the TCP/IP protocol to establish a well-organized communication between clients and the server. The server is able to handle multiple client connections at once through multithreading.

TCP/IP stands for Transmission Control Protocol/ Internet Protocol. It is a set of conventions or rules and methods that are used to interconnect network devices on the Internet. It chooses how the information will be traded over the web through end-to-end communications that incorporate how the information ought to be organized into bundles (bundles of data), addressed, sent, and received at the goal. This communication protocol can also be utilized to interconnect organize devices in a private network such as an intranet or an extranet.

The TCP allows applications to create channels of communications across a network. It also permits a message to be separated into smaller packets before they are transmitted over the web and after that collected in the right order at the destination address. So, it guarantees the solid transmission of data across the channel which is crucial for this project.

A Thread is a single sequence stream within a process. Because threads have some of the properties of processes, they are sometimes called lightweight processes. Threads operate faster than processes, making them useful to this project.

3 Application Architecture

The application will consist on the following parts: the server and the client. The client focuses on reading input from the user and showing the server's response to the used commands.

The client performs the following tasks based on the command:

- **Login**: Processes a provided username and password, forwarding them to the server for validation, subsequently enabling the client to execute various additional commands.
- **Register**: Initiates a request to the server for validation and the creation of a new user, utilizing a specific username and password entered by the user. The client can only execute this command if he is not currently logged in.
- **Info**: Initiates a request to the server for information about available championships.
- **Create Championship**: Requests the server to add a championship of specified rules to it's database. Only available to "Admin" type users.
- **Join**: Initiates a request to the server to participate in a specified championship.
- **Change Date**: Initiates a request to the server to reschedule a championship. Only available to "Admin" type users.
- **Leave Championship**: Initiates a request to the server to abandon a specified championship.
- **Logout**: Logs out the current user.
- **Quit**: The client will close the connection and stop it's functioning.

The server performs the following tasks based on information sent by the client:

- **Login**: If the user attempts to login, the server will request the client to provide an username and a password. It will then search in the database if that specific user exists and if his password is correct. Logging in as an admin will require an additional code.
- **Register**: the server attempts to insert in the database a new user with an username and a password with the condition that the username is not already existing in the database.
- **Info**: Displays information about available championships to the client.
- **Create Championship**: Requests additional information regarding the rules and adds the championship of specified rules to it's database if the command is provided by an admin type user. Displays an appropriate message to the client otherwise.
- **Join**: Requests the specific championship the client wishes to join and adds the user to the championship's database.
- **Change Date**: Requests the specific championship, the date and hour and implements the changes.
- **Leave Championship**: Requests the specific championship and modifies the championship database.

4 Implementation details

At the start of the connection, the client will provide either the register or the login commands implemented in the following code:

```
int a=0; // admin
int o=0; // obisnuit
int i;
int l=-1; // logged in or not

while(1){
        bzero(buffer, 255);
        n = read(newsockfd, buffer, 255);
        if(n < 0)
                error("Error on reading");
        printf("Client: %s", buffer);

int c1 = strncmp("Register", buffer, 8);
        if(c1 == 0){
            n=write(newsockfd, "Username: ", 10);
            n = read(newsockfd, buffer, 255);
            buffer[strcspn(buffer, "\n")] = '\0';

            int usernameTaken=0;

            for (int i = 0; i < num_users; i++) {
            if (strncmp(Users[i].username,
                buffer, strlen(Users[i].username)) == 0)
                usernameTaken = 1;
            }

            if(usernameTaken==1)
            n=write(newsockfd, "Username already taken", 22);
                else{
            strcpy(Users[num_users].username, buffer);

            n=write(newsockfd, "Password: ", 10);
            n = read(newsockfd, buffer, 255);
            buffer[strcspn(buffer, "\n")] = '\0';
            strcpy(Users[num_users].password, buffer);

            n=write(newsockfd, "obisnuit or admin: ", 19);
            n = read(newsockfd, buffer, 255);
            buffer[strcspn(buffer, "\n")] = '\0';
            strcpy(Users[num_users].tip, buffer);
```

```
                    n=write(newsockfd, "Inregistrat cu succes", 21);
                    num_users++;
                        }
                    }

        int c2 = strncmp("Login", buffer, 5);
                if(c2 == 0){
                    l=-1;
                    n=write(newsockfd, "username:", 10);
                    n = read(newsockfd, buffer, 255);
                    buffer[strcspn(buffer, "\n")] = '\0';
                    for(i=0; i<num_users; i++){ // checks for existing username
                        if (strncmp(Users[i].username,
                            buffer, strlen(Users[i].username))==0){ l=i;
                                                                    break;}
                    }
                    if(l!=-1){
                        n=write(newsockfd, "password:", 10);
                        n = read(newsockfd, buffer, 255);
                        buffer[strcspn(buffer, "\n")] = '\0';
                        if (strncmp(Users[l].password,
                            buffer, strlen(Users[l].password))==0){
                            if (strncmp("admin", Users[l].tip, 5)==0){
                                n=write(newsockfd, "admincode:", 11); // "0000"
                                n = read(newsockfd, buffer, 255);
                                if (strncmp("0000", buffer, 4)==0){
                                a=1;
                                o=1;
                                n=write(newsockfd, "Logged in", 9);
                                }
                                else n=write(newsockfd, "cod gresit", 10);
                             }
                             else{                                    o=1;
                                n=write(newsockfd, "Logged in", 9);
                                }
                        }
                        else n=write(newsockfd, "Parola gresita", 14);
                    } else n=write(newsockfd, "Username gresit", 16);
        }
       \\ ...
        }
```

Only after the process of logging in, can the client access the rest of the commands. The server ensures this by checking for the "a" and "o" integer before executing other commands. For example:

```
int c3 = strncmp("Info", buffer, 4);
                if(c3 == 0){
            if (a == 1 || o == 1) {
            \\ ...
            }   else n = write(newsockfd, "Not currently logged in", 23);
```

5 Conclusions

This application presents a simple and user-friendly interface to create and participate in different championships. It could be improved by adding better security measures such as two factor authentification.

References

Documentation about TCP servers and multithreading:
$https://www.geeksforgeeks.org/tcp-ip-in-computer-networking/$
$https://www.geeksforgeeks.org/multithreading-in-c/$