Championship

Ionela Cotiuga

Facultatea de Informatica, Iasi
secretariat@info.uaic.ro
https://www.info.uaic.ro/

Abstract. Prezentarea proiectul **Championship**. Descrierea proiectului si scopul acestuia, tehnologiile utilizate in implementarea lui, arhitectura aplicatiei, diagarama, conceptele implicate si detaliile de implementare.

Keywords: Client · Server · TCP · Fork.

1 Introducere

Proiectul Championship este o aplicatie client/server ce va putea administra diferite campionate. In aplicatia server vor fi indeplinite urmatoarele functionalitati: inregistrarea utilizatorilor, logarea si delogarea acestora, inregistrarea unui campionat , inscrierea unui utilizator intr-un campionat. Mai multi clienti se pot conecta la server si vor putea introduce comenzi diferite, iar raspunsurile vor fi primite in mod concurent.
Pentru implementarea proiectului, tehnologia utilizata este TCP-ul concurect folsindu-se de fork-uri per client. Pentru fiecare client se va crea cate un proces copil in cadrul caruia se va citi requestul de la client si i se va trimite raspunsul aferent.

2 Tehnologiile utilizate

TCP concurent (Fork per client) Pentru implementarea proiectului Championship se utilizeaza protocolul TCP care permite comunicarea client-server. Scopul protocolului TCP este acela de a controla transferul de date in asa fel incat acesta sa fie de incredere. Comunicarea intre client si server se va face cu ajutorul unui socket. Pentru fiecare client ce se va conecta la server se va crea cate un proces copil in cadrul caruia se va citi requestul de la client si i se va trimite raspunsul aferent.

2.1 Aplicatia client

Va contine: crearea socketului si conectarea la server; apoi ni se prezinta comenzile posibile pe care le putem apela. Utilizatorul va scrie comanda dorita, si in functie de aceasta, va primi un raspuns de la server.

2.2 Aplicatia server

Se creeaza un socket si se asteapta conectarea clientilor la server; dupa ce acceptam cererea de conexiune de la client, vom apela functia fork() pentru a crea un proces copil, iar in bucla while(1) vom trata clientii.

Cu ajutorul functiei raspunde() vom apela alte functii(`create_account`, login, logout, exit, `sign_in_championship`, `register_championship`) in functie de requestul primit de la client.

2.3 Sqlite3

Pentru stocarea detaliilor campionatelor se foloseste baza de date sqlite3. In fisierul championships.db exista 2 tabele:
1. championships: este baza de date cu 4 coloane (game_name, nr_players, rules, bracket_type) care stocheaza datele despre campionatele inregistrate. Aceasta este accesata si modificata de fiecare data cand un administrator inregistreaza un nou campionat;
2. inscrieri: (game_name, user_name, score, email_address). In acest tabel sunt stocate datele despre utilizatorii inscrisi in campionate. Pentru fiecare joc, se pot inscrie un numar limitat de jucatori (doar cati au fost declarati ca nr_players in detaliile jocului).

3 Arhitectura aplicatiei

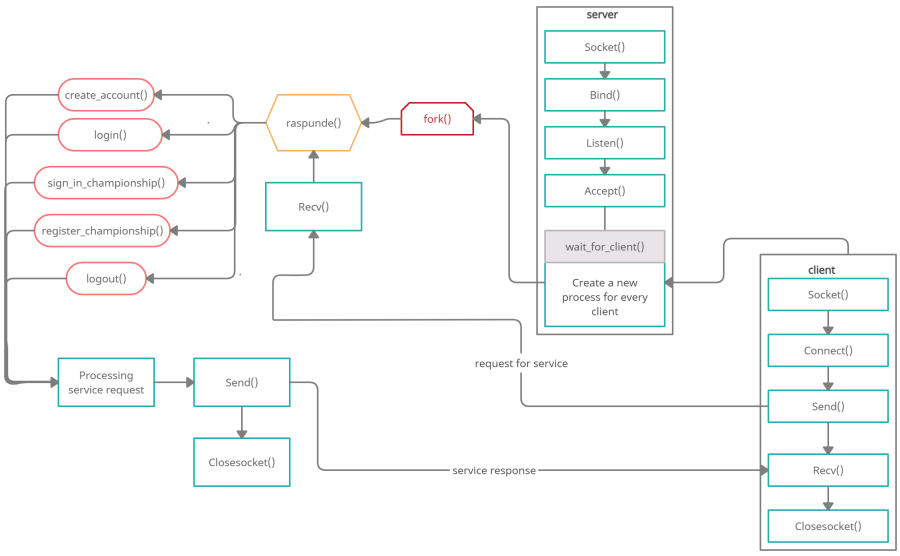

Fig.1. Diagrama cuprinde cele doua aplicatii, client si server, si functiile apelate in cadrul acestora. Pentru fiecare client se porneste cate un proces copil. Procesele nou create apeleaza aceeasi functie, raspunde(). In cadrul functiei raspunde(), se hotaraste ce functie se apeleaza mai departe, depinzand de requestul clientului. Functiile `create_account`, login, logout, exit `sign_in_championship`, `register_championship` proceseaza requestul primit si trimit in final un raspuns catre client.

4 Detalii de implementare

4.1 Scenarii de utilizare

- comunicarea se face prin executia de comenzi citite de la tastatura in client si executate in procesele copil create in server;
- comenzile sunt siruri de caractere delimitate de new line;
- in aplicatia client ni se prezinta comenzile pe care le putem alege: `create_account`, login, logout, exit `sign_in_championship`, `register_championship`;
- toate comenzile vor fi restrictionate de sectiunea de logare;
- protocolul minimal cuprinde comenzile:

- "register:user:name:password / register:admin:name:password" - de implementat pentru 2 tipuri de utilizatori(obisnuiti, administratori); se creeaza cate un fisier de configurare pt cele 2 tipuri de utilizatori care se conecteaza in aplicatie: un fisier pentru administratori admins.txt si un fisier pentru utilizatorii obisnuiti users.txt; diferenta dintre ei se face la introducerea numelui de utilizator, in cazul administratorilor, langa numele lor se va scrie si cuvantul "admin" iar in cazul utilizatorilor obisnuiti, langa numele lor se va scrie cuvantul "user".

- "login:user:name:password /login:admin:name:password" - dupa logare, utilizatorii obisnuiti vor putea primi informatii despre campionatele inregistrate iar administratorii pot inregistra noi campionate

- "logout"

- "exit"

- " `sign_champ: game_name|user_name|score|email_address`" - comanda va putea fi executata daca utilizatorul este autentificat in aplicatie; acesta va fi informat daca a fost acceptat in campionatul respectiv si va primi informatii aditionale despre partidele sale (ora, adversarul, etc...); utilizatorul are posibilitatea sa reprogrameze o sesiune de joc.

- "`reg_champ: game_name|nr_players|rules|bracket_type`" - comanda va putea fi executata daca utilizatorul administrator este autentificat in aplicatie; specificarea jocului, numarului de jucatori, diferite reguli sau structuri de campionat(single-elimination, double elimination), modul de extragere a partidelor(deciderea partidelor). -

4.2 Cod relevant particular proiectului

Functii importante din aplicatia server:

In bucla while() se accepta fiecare client care se conecteaza la server iar pentru fiecare dintre acestia se creeaza un proces copil, apeland funcgtia fork().

```
while (1)
{
    int client;
    int length = sizeof(from);

    fflush(stdout);

    if ((client = accept(sd, (struct sockaddr *)&from, &length)) < 0)
    {
        perror("[server]Eroare la accept().\n");
    }
    printf("[server]S-a conectat un nou client.\n");

    if (fork() == 0)
    {
        raspunde(client);
    }
}
```

- functia **raspunde()** citeste comanda introdusa de client si apeleaza alte functii, depinzand de requestul clientului:

```
 void raspunde(void *arg)
{
    /*.........*/

    char *com1, *com2, *com3, *com4, *com5, *com6;
    com1 = strstr(sir, "register:");
    com2 = strstr(sir, "login:");
    com3 = strstr(sir, "sign_champ:");
    com4 = strstr(sir, "reg_champ:");
    com5 = strstr(sir, "logout");
    com6 = strstr(sir, "exit");

    if (com1 != NULL && error == 0){
            create_account(client, username_data, utilizator);
        }
        else if (com2 != NULL && error == 0){
            login(client, username_data, utilizator);
        }

        else if (com3 != NULL){
```

```
            sign_in_championship(client, sir);
        }

        else if (com4 != NULL){
            register_championship(client, sir);
        }
        else if(com5 != NULL){
            logout(client);
        }
        else if(com6 != NULL){
            printf("[server]Clientul s-a deconectat de la server.\n");

    /*..........*/

}
```

5 Concluzii

Pentru o vizualizare mai buna a informatiilor despre campionate sau a inregistrarii utilizatorilor se poate realiza o interfata grafica a proiectului.

References

1. Computer Networks https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
2. Laboratory https://profs.info.uaic.ro/ ioana.bogdan/