Rețele de calculatoare
Proiect - Connect Four

Ganea A. Florin 2A5
23 decembrie 2020

1. Introducere

Connect Four este un joc care se joacă în doi oameni, aceștia stabilind, de regulă, culoarea și jucătorul care începe prin tragere la sorți. Jocul conține un grilaj cu șase linii și șapte coloane așezat în poziție verticală în care jucătorii introduc, pe rând, câte un disc corespunzător culorii stabilite la început care cad pe coloana aleasă pe poziția cea mai de jos neocupată. Obiectivul unui jucător este de a fi primul care formează o linie din patru discuri proprii pe orizontală, pe verticală sau pe diagonală.

Pentru simularea jocului Connect Four am creat o aplicație de tip client care interacționează cu un server care stabilește jucătorul care face prima mutare și le asignează jucătorilor o culoare(roșu sau galben). Totodată, serverul are rolul de a ține scorul și oferă posibilitatea utilizatorilor de a juca mai multe reprize.

1.1 Motivaţie

Alegerea acestei teme a fost influențată de faptul că îmi plac board games, în special cele strategice (e.g. șahul), cât și de faptul că nu am avut în trecut prilejul de a juca acest joc. Prin urmare, această posibilitate de a îmbina utilul cu plăcutul m-a determinat să aleg jocul Connect Four.

2. Tehnologii utilizate

i) TCP

Pentru crearea serverului, am utilizat o conexiune TCP (Transmission Control Protocol) în detrimentul conexiunii UDP (User Datagram Protocol), deoarece:

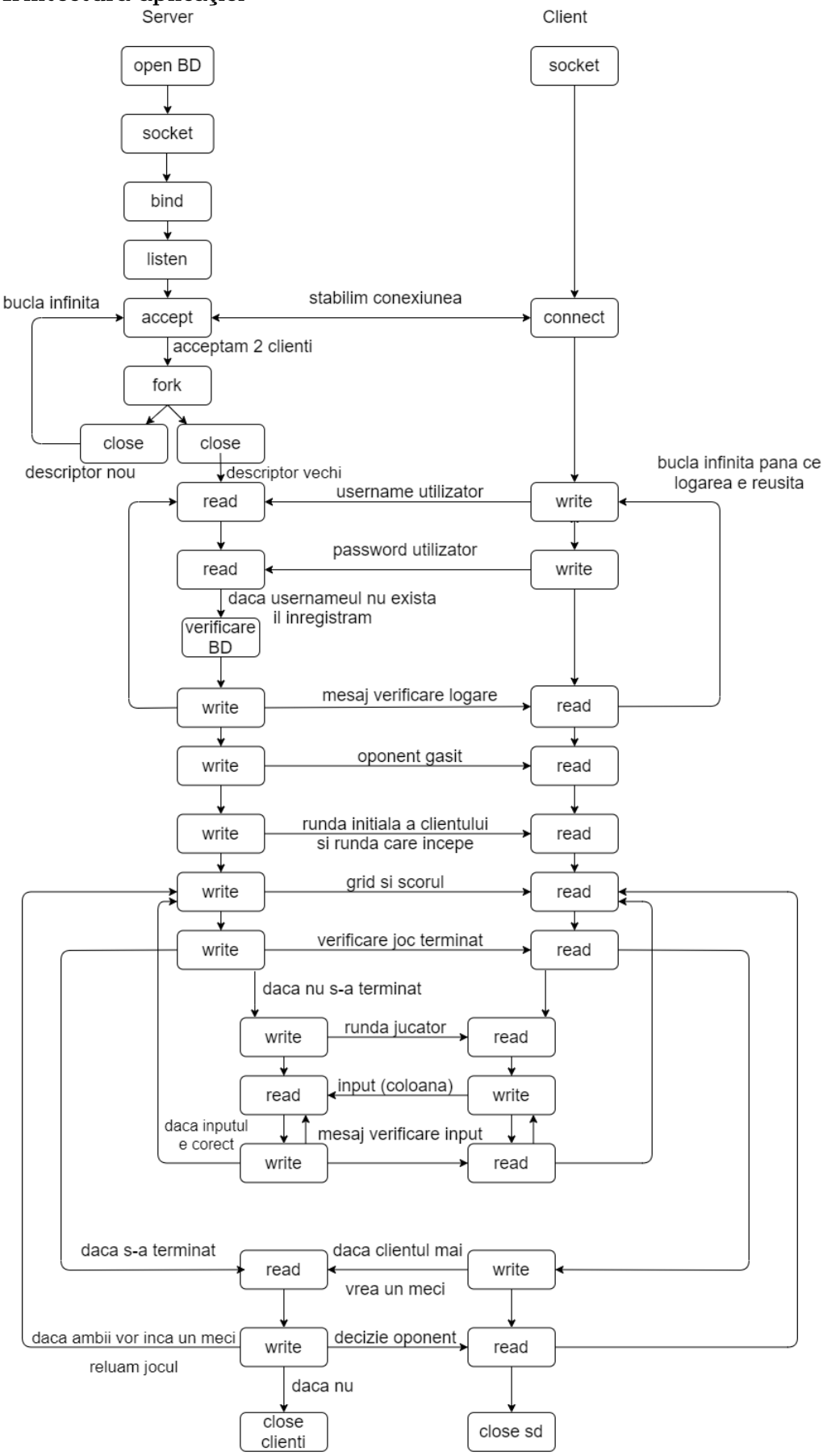
-UDP nu poate garanta faptul că pachetele au ajuns complete și la timpul potrivit, spre deosebire de TCP, acest lucru fiind esențial într-un joc interactiv precum Connect Four, unde jucătorii trebuie să trimită pe rând mutarea și această să fie verificată.

-În ciuda faptului că UDP este mai rapid, acest lucru nu ne interesează deoarece datele transmise între client și server sunt relativ mici (integers și șiruri de caractere de lungime maximă 256), faptul că pachetele au ajuns în întregime și în ordinea corectă este mult mai important decât viteza în cazul nostru.

ii) Concurența

Pentru a asigura concurența serverului, am utilizat primitiva fork() pentru a separa perechile de utilizatori care joacă, cu scopul ca aceste perechi să fie independente unele de altele (e.g. un participant de la jocul 1 nu trebuie să depindă de un participant de la jocul 3).

3. Arhitectura aplicaţiei



4. Detalii de implementaree

Serverul începe prin a crea o bază de date (dacă aceasta nu există) și o deschide, după care creează socketul, atașează serverul la socket și începe să "asculte" dacă vin clienți. Apoi, într-o buclă infinită acceptă 2 clienți și apelează primitiva fork(), în părinte se vor închide conexiunile, iar în copil se va desfășura aplicația propriu-zisă. Clienții sunt întâmpinați cu un mesaj de bun-venit și aceștia sunt nevoiți să se logheze. În cazul în care contul nu există, acesta va fi adăugat în bază de date:

```
Bine ati venit! Va rugam sa va autentificati.
Daca nu aveti un cont, acesta va fi inregistrat automat.
Va rugam introduceti un username.
florin123
Va rugam introduceti parola.
 panem
V-ati inregistrat cu succes!
Asteptati un oponent.
```

Figura 4.1. Mesaj înregistrare

În cazul în care contul există, dar parola este greșită, clientul are posibilitatea să încerce din nou. Dacă reușește să se logheze, jocul poate începe:

```
Va rugam introduceti un username.
florin
Va rugam introduceti parola.
 pix12
Parola incorecta, va rugam reincercati!
Va rugam introduceti un username.
florin
Va rugam introduceti parola.
 pix123
V-ati logat cu succes!
Asteptati un oponent.
A fost gasit un oponent. Meciul va incepe imediat.
```

Figura 4.2. Autentificare eșuată și autentificare reușită

Autentificarea se realizează astfel:

```
do
{
    read_string(client, nickname_c1); //citim usernameul primului client
    read_string(client, password_c1); //citim parola acestuia
    login_check = username_check(db, path, nickname_c1, password_c1); //verificam in baza de date
    if(login_check == -2) //daca nu exista in baza de date il adaugam
      insert_data(path, nickname_c1, password_c1);

    send_message_username_check(client, login_check); //trimitem clientului daca logarea a esuat sau a reusit
} while (login_check == 0);
```

Figura 4.3. Citirea usernamelui și a parolei și verificarea acestora

```c
int username_check(sqlite3 *db, const char *path, char username[256], char password[256])
{
    int ok = -2;
    //ok==1 user logat
    //ok==0 username corect, parola gresita
    //ok==-2 usernameul nu exista, il inregistram
    sqlite3_stmt *stmt;
    sqlite3_prepare_v2(db, "select * from LOGIN", -1, &stmt, NULL);
    //BD are atributele ID (col. 0), Username(1) si Password(2)
    while (sqlite3_step(stmt) != SQLITE_DONE) //schimbam randul
    {   //daca gasim usernameul pe a 2-a coloana
        if (strcmp(sqlite3_column_text(stmt, 1), username) == 0)
        { //daca parola corespunde
            if (strcmp(sqlite3_column_text(stmt, 2), password) == 0)
            //daca am gasit usernameul si pe acelasi rand gasimt parola
                ok = 1;
            //daca am gasit usernameul dar parola nu corespunde
            else
                ok = 0;
            break;
        }
    }
    //daca nu am gasit usernameul ok ramane -2.

    sqlite3_close(db); //inchidem BD
    return ok;
}
```

Figura 4.4. Funcția care verifică parola și usernameul

Serverul decide care client începe jocul prin funcția rand() și le transmite clienților care dintre ei are prima mutare. Jocul este simulat printr-o matrice existentă în server inițializată cu 0 pe care o modificăm cu valorile 1 sau 2 în funcție de cel care a efectuat o mutare și pe care o trimitem clienților. Dacă e tura lui, clientul poate introduce o coloană, pe care o trimite serverului. Acesta are rolul de a verifica corectitudinea inputului, de a modifica matricea și de a trimite un mesaj.

```
Ati primit culoarea rosie prin randomizare. Veti incepe jocul.

A B C D E F G

_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
florin123 0 - 0 mihai
Introduceti coloana:
```

Figura 4.5. Jocul din perspectiva utilizatoruluil

Dacă coloana introdusă nu este corectă, afișăm eroarea și oferim posibilitatea de a încerca din nou.

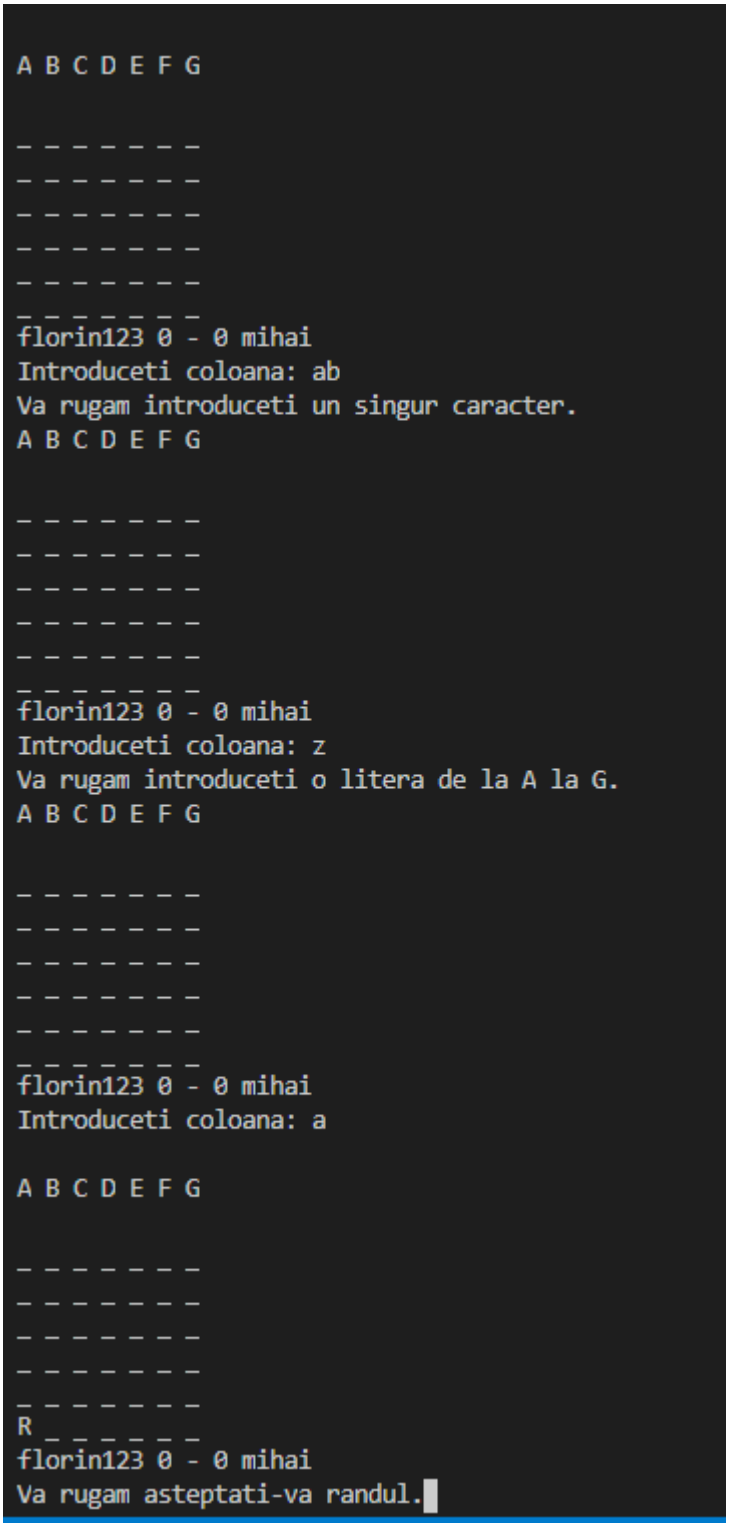


Figura 4.6. Mesaje pentru diferite inputuri

Dacă unul dintre utilizatori câștigă, actualizăm scorul și îi întrebăm dacă doresc să mai joace. În cazul în care ambii doresc, se începe un nou joc. În cazul în care unul dintre utilizatori nu dorește, îi deconectăm pe amândoi.

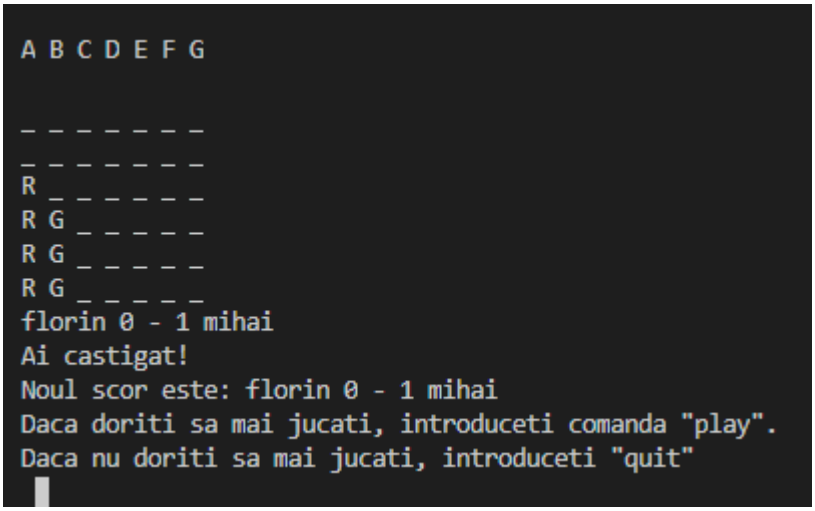


Figura 4.7. Mesaj rundă finalizată

5. Concluzie

În concluzie, aplicația creată până în prezent simulează un joc Connect Four, dar într-un mod minimal, aceasta putând fi îmbunătățită prin implementarea următoarelor caracteristici:

-interfață grafică;

-timp limită pentru mutare, dacă depășește timpul deconectăm utilizatorul;

-bază de date pentru istoricul dintre doi jucători (scorul all-time) și diferite statistici.

6. Webografie

wikipedia

guru99

laboratorul 7

enunt proiect