Chess

Dominte Florin Iulian

Universitatea Alexandru Ioan Cuza (dominte.florin.iulian@gmail.com)

1 Introducere

Proiectul "Chess" consta in crearea unei aplicatii de sah care ruleaza pe un server concurent, care va superviza fiecare partida de sah in parte. Acest server trebuie sa anunte castigatorul fiecarei partide.

2 Tehnologiile utilizate

In cadrul proiectului, se vor folosi urmatoarele tehnologii:

1. limbajul de programare "C";
2. un server concurent cu protocolul TCP;
3. pthreads;
4. sockets() pentru comunicarea intre client-server;

3 Arhitectura aplicatiei

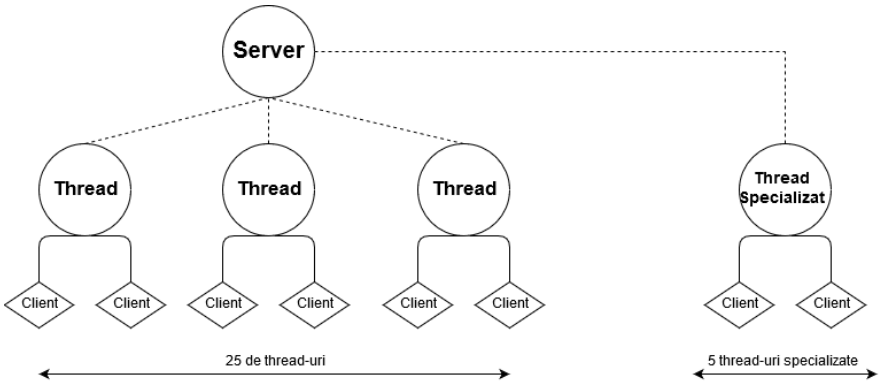

Fig. 1. Diagrama proiectului initial

Diagrama anterioara prezinta modelul teoretic initial.

In aceasta instanta, thread-urile normale vor accepta, fiecare, cate 2 clienti, care vor apela functia de "log in".
Dupa ce ambii clienti s-au conectat cu conturi valide, jocul de sah va incepe, thread-ul deconectandu-se(pthread detach) de la server.

Thread-urile specializate se vor ocupa de partidele de sah "custom". Cei 2 clienti vor cere accesul de la server de a se conecta pe acelasi thread pentru a juca o partida impreuna.

Thread-urile normale ofera posibilitatea de a juca cu persoane "la intamplare", pe cand cele specializate, cu ajutorul datelor clientilor, vor crea partide intre jucatori.

Totusi, ideea anterioara avea cateva probleme.
Astfel, avand un singur thread pentru doi clienti, functia de "log in" si toate comenzile lor for fi iterative, nu concurente.
De asemenea, nu exista optiunea de a rejuca cu alte persoane fara a inchide clientul.

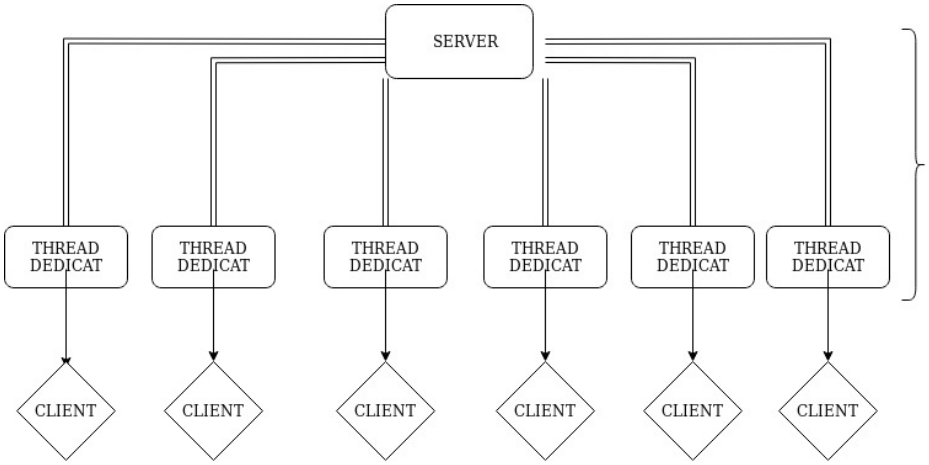

Fig. 2. Diagrama proiectului actuala

In aceasta diagrama se prezinta modelul aplicat.
Acum, pentru fiecare client conectat, o sa avem un thread dedicat, cu posibilitatea de "log in", de a alege "masa" de sah unde vor juca partida.
Comunicarea intre client-thread se va face prin socket-uri, iar comunicarea intre thread-uri se va face prin niste "camere de joc" in care vor putea citi si edita mutarile de sah.

Cu ajutorul functiilor pthread-mutex-lock/pthread-mutex-unlock vom reglementa asincronitatea dintre cei doi jucatori.

4 Detalii de implementare

Pentru comunicarea intre procese se vor folosi socket-uri. Clientii nu vor comunica direct intre ei, ci prin thread-urile lor dedicate.

Intr-o partida de sah, ambii clienti impreuna cu thread-urile lor, vor avea access la o structura care va retine mutarile, iar partida se va desfasura in client.
Aceasta, la finalul fiecarei partide,va anunta ambii clienti cine a castiga si va fi reinitializata pentru a putea fi reutilizata.

Dupa finalizarea partidei, cei doi jucatori vor fi retrimisi la un ”menu” de unde vor putea alege alta masa sa joace o noua partida netrebuind sa inchida clientul, ca in conceptul anterior.

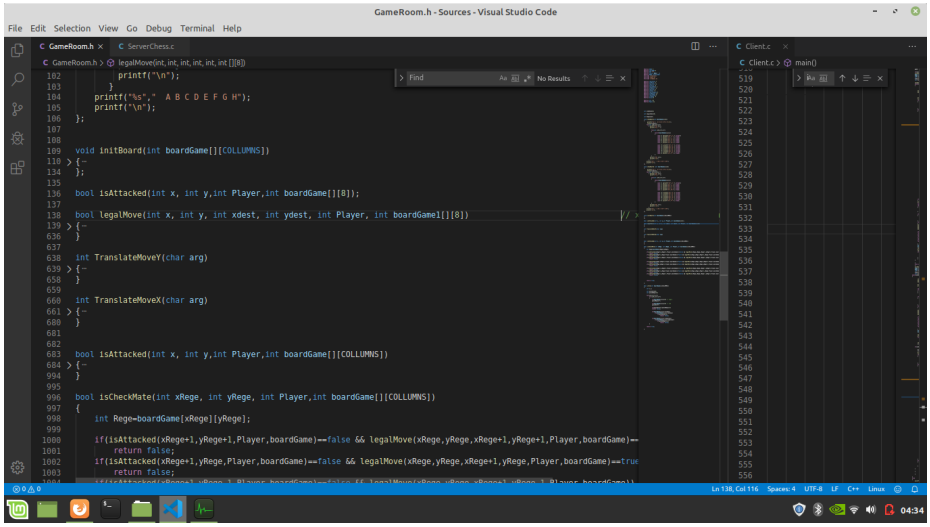

Fig. 3. Functiile de baza pentru jocul de sah

Acestea sunt functiile necesare pentru jocul de sah. Vor ajuta la initializarea tablei de sah, vor putea verifica daca o mutare este corecta si daca in urma acesteia regele nu intra in sah.
TranslateToX si TranslateToY sunt functiile necesare pentru a face trecerea mai rapida de la miscarea fizica, input-ul player-ului, la cel de codat.

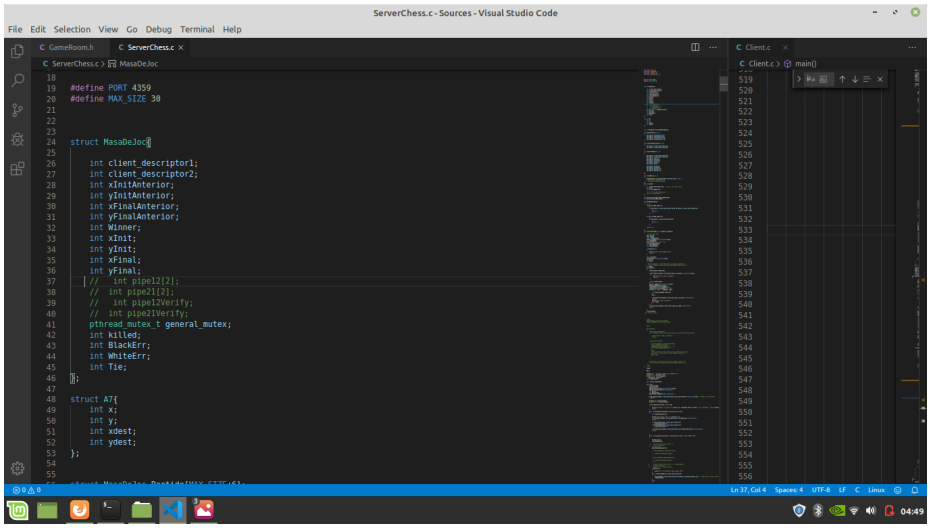

Fig. 4. Structura globala de date

Thread-urile vor scrie in acest struct toate miscarile si server-ul va decide prin variabila "Winner" cine a castigat. Prin variabila "Tie" se va anunta o remiza. De asemenea, se vor tine minte si client descriptorii celor doi care se vor logga, pentru a asigura faptul ca nu o sa avem intreruperi.

5 Concluzii

Lucruri pe care le-am putea adauga:

- cronometru
- sistem de punctare
- cresterea numarului de thread-uri
- interfata
- implementarea unei baze de date pentru statistici(victorii/infrangeri)
- posibilitatea de "turn back"
- posibilitatea de "flip coin"(pentru alegerea culorilor)

6 Bibliografie

- http://man7.org/linux/man-pages/man2/socket.2.html
- https://www.geeksforgeeks.org/thread-in-operating-system/
- https://www.sfml-dev.org/
- https://linux.die.net/man/3/pthread-mutex-lock