Quizz Game

Iacob Daria-Roxana

Facultatea de Informatică Iași, Rețele de calculatoare
daria.iacob@info.uaic.ro

1 Introducere

Quizz Game este un joc interactiv realizat prin implementarea unui server concurent în limbajul C/C++ care poate suporta un număr nelimitat de clienți.

Modul de desfășurare al aplicației este următorul: o sesiune de joc este formată din trei clienți, care pornește în momentul în care s-au conectat la server numărul necesar de participanți. Chestionarul este format din opt întrebări iar timpul alocat primirii unui răspuns este de douăzeci de secunde. Jucătorii vor fi punctați în funcție de viteza în care au răspuns la întrebare. Fiecare jucător are posibilitatea de a părăsi jocul. La finalul unei anumite sesiuni de joc numele câștigătorului va fi anunțat către toți participanții rămași.

Am ales acest proiect deoarece îl consider potrivit pentru a aprofunda și a pune în practică noțiunile învățate la curs și laborator. De altfel, îmi doream să implementez un joc întrucât consider că implică mai multă creativitate, originalitate și poate fi recreativ. Restricția legată de utilizarea unei baze de date SQLite sau a unor fișiere XML am considerat-o drept o provocare de a ieși din zona de confort.

Scopul jocului este de a îmbina elementul distractiv cu asimilarea unor informații noi ce pot contribui la îmbunătățirea culturii generale a jucătorilor deoarece întrebările adresate sunt din domenii diverse: istorie, geografie, literatură și biologie. În plus, întrebările sunt trimise în limba engleză, astfel participând la acest joc clienții își pot perfecționa abilitățile de înțelegere și comunicare în această limbă.

2 Tehnologii utilizate

2.1 Protocol

Comunicarea dintre server și client se realizează prin intermediul protocolului TCP. Protocolul corespunde nivelului transport din stiva TCP/IP și realizează o conectare *full duplex* între două puncte terminale, fiecare punct fiind definit de către o adresă IP și de către un port. TCP controlează mărimea segmentului de date și efectuează evitarea congestionării traficului de rețea.

Am făcut această alegere deoarece TCP este un protocol de transport orientat conexiune, de încredere, care oferă siguranță și garantează livrarea unui flux de date trimis de la un punct terminal la altul fără duplicarea sau pierderea de date.

Caracteristicile protocolului TCP utilizate în cadrul proiectului sunt:

1. Retransmiterea pachetelor, niciun pachet nu este pierdut. În cadrul jocului această proprietate este necesară deoarece pe de o parte, nu ar fi benefic ca jucătorii să primească doar porțiuni din întrebările adresate iar pe de cealaltă parte, răspunsul oferit de jucător să ajungă incomplet la server fiindcă acesta nu ar putea fi punctat, iar clasamentul final ar fi incorect.
2. Reasamblarea pachetelor în ordinea corectă la destinație. Această trăsătură este importantă pentru a asigura corectitudinea transmiterii întrebărilor și a răspunsurilor. Spre exemplu, nu ar fi favorabilă amestecarea literelor deoarece mesajul ar putea deveni indescifrabil.

2.2 Stocarea datelor

Întrebările cu variantele de răspuns sunt stocate într-o bază de date SQLite de unde vor fi preluate pentru a fi transmise jucătorilor. Pentru fiecare întrebare, în baza de date se va reține și răspunsul corect corespunzător.

De asemenea, pentru fiecare rundă punctajele aferente jucătorilor sunt păstrate într-o bază de date pentru a putea afla la final câștigătorul jocului cât și punctajul acumulat de fiecare participant în parte.

SQLite este o bibliotecă C care implementează un motor de baze de date SQL. Am ales să utilizez SQLite pentru că este accesibilă, ușor de folosit și administrat, are o perfomanță bună, oferă consistența datelor prin respectarea setului de proprietăți ACID. În plus, am considerat că este o ocazie prielnică de a pune în practică cunoștintele dobândite la cursul de Baze de Date.

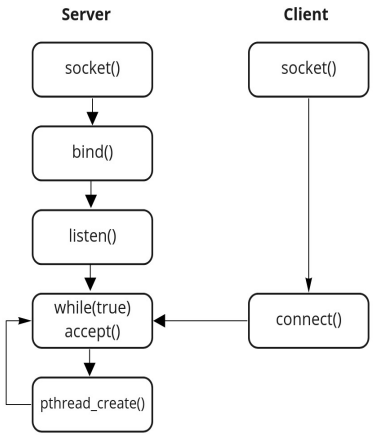
2.3 Interfața grafică

Pentru a realiza o interfață grafică pentru client voi folosi librăria GTK și aplicația Glade care permite dezvoltarea de interfețe dintr-un mediu vizual. Interfața grafică creată în Glade este salvată într-un fișier XML care prin intermediul obiectului GtkBuilder poate fi încărcat în program.

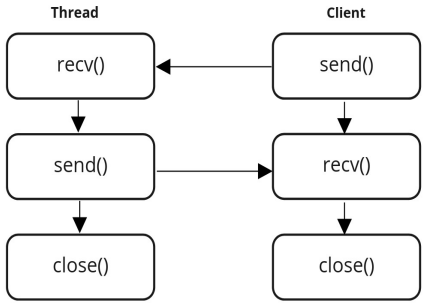
Am ales să realizez o interfață grafică pentru a face interacțiunea dintre utilizator și computer mai ușoară și mai eficientă. Aceasta oferă atractivitate, o înțelegere mai bună a aplicației deoarece oferă un suport vizual, accesarea funcționalităților fiind mai ușoară.

3 Arhitectura aplicației

Serverul TCP concurent este implementat prin intermediul firelor de execuție. Thread-ul principal este blocat într-un apel *accept*(), iar când un client este acceptat conexiunii va crea un thread nou cu ajutorul primitivei *pthread_create*() care îl va servi. Astfel, serverul va putea comunica cu mai mulți clienți simultan. După conectarea la server și crearea thread-ului ce se va ocupa de respectivul client poate începe comunicarea între cele două puncte terminale.



(a) Creare thread pentru fiecare client conectat.



(b) După conectarea la server a unui client, comunicarea se realizează la nivel de thread.

Pentru a începe jocul, clientul trebuie să trimită o cerere pentru a înștiința serverul că este pregătit de joc în care va trimite și numele de utilizator pe care îl va avea pe parcursul jocului. În cazul în care clientul se răzgândește și nu mai dorește să continue jocul poate trimite o cerere de închidere a conexiunii.

Atunci când un thread primește o cerere de începere a jocului din partea unui client, va realiza o configurare pentru acesta ce constă în atribuirea unei sesiuni de joc și a unui număr de jucător în cadrul sesiunii (din intervalul $[1, 3]$). Astfel, fiecare client va fi identificat în mod unic prin aceste două atribute. Secțiunea de cod ce se ocupă de configurarea informațiilor este protejată de un *mutex* pentru a oferi accesul la resursele partajate unui singur thread la un moment dat.

Pentru a verifica dacă jocul poate începe, fiecare thread va accesa un *flag* comun (beneficiem de memorie partajată în cadrul thread-urilor) care este setat pe *true* în momentul în care pentru sesiunea de joc corespunzătoare s-au adunat numărul maxim de participanți. Odată ce jocul poate porni, se va încărca pe rând câte o întrebare din cele opt ale chestionarului dintr-o bază de date SQLite și vor fi trimise în același timp jucătorilor. În aceeași interogare va fi extrasă din baza de date și varianta corectă corespunzătoare fiecărei întrebări.

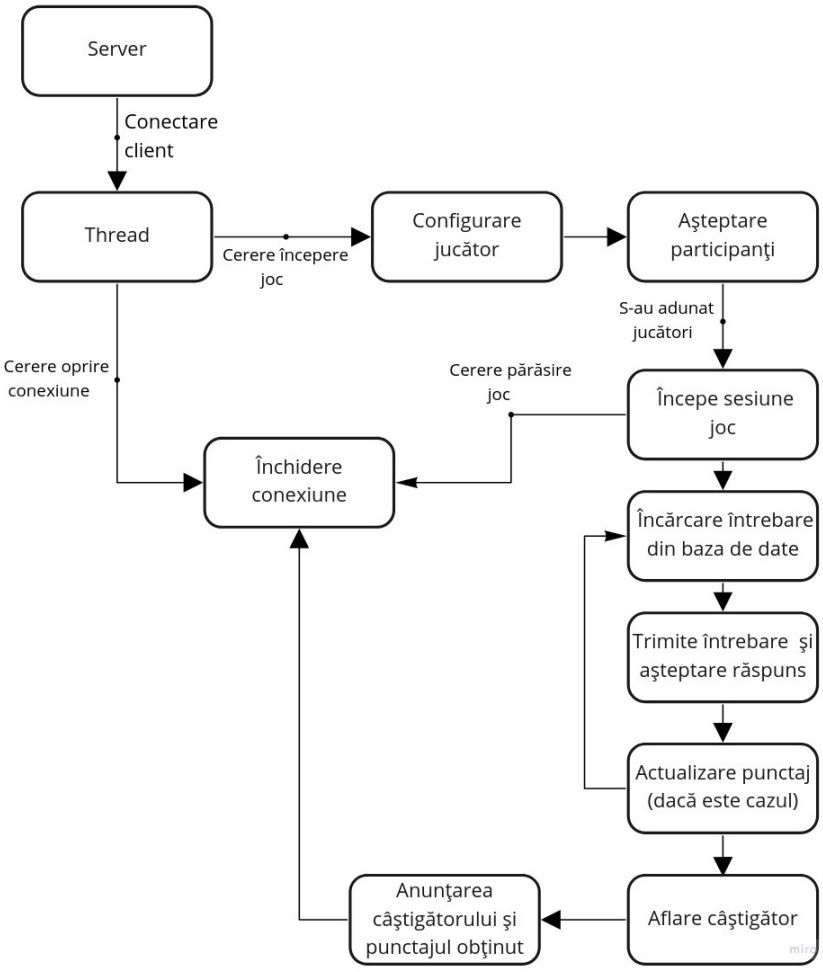


Fig. 2: Diagrama aplicației la nivel de server.

Jucătorii au douăzeci de secunde timp pentru a răspunde, iar dacă soluția trimisă este corectă, jucătorul va fi punctat în funcție de viteza în care a raspuns. Următoarea întrebare va fi trimisă după expirarea timpului chiar dacă jucătorii au răspuns mai rapid, pentru a realiza sincronizarea clienților.

Punctajele jucătorilor sunt de asemenea stocate într-o bază de date SQLite pentru a putea afla la finalul chestionarului câstigătorul sesiunii respective cât și punctajul pe care fiecare client a reușit să-l acumuleze. La începutul jocului este inserat în baza de date un tuplu pentru fiecare client, inițial cu punctajul 0, iar mai apoi de fiecare dată când jucătorul a oferit un răspuns corect se va realiza o operație de *update*. Dacă un participant dorește să părăsească jocul cât timp acesta este în derulare, punctajul respectivului jucător va deveni -1,

iar jocul va continua fără probleme atât timp cât rămâne cel puțin un jucător în acea rundă. După anunțarea numelui câștigătorului către toți participanții rămăși dintr-o anumită sesiune, conexiunea între client și server poate fi închisă.

Operațiile de inserare, actualizare, ștergere a valorilor din baza de date necesare pe parcursul jocului sunt protejate de lacăte speciale pentru a garanta consistența datelor si buna funcționare a aplicației. Totodată, aceste operații vor fi realizate de un thread separat, pentru a nu bloca thread-ul corespunzător clientului.

4 Detalii de implementare

Serverul TCP concurent este implementat prin intermediul tehnicii de multithreading. După crearea unui socket() care va fi folosit în continuare pentru a face bind(), listen() și accept(), serverul este pregătit pentru a accepta conexiuni și a asigna un thread care se va ocupa de fiecare client. Comunicarea între server și client se realizează folosind socket-uri.

```
int server()
{   /*...*/
    while(true){
        int clientDescriptor;
        threadInfo* threadArg;
        if((clientDescriptor=accept(serverD,(struct sockaddr*)
   &client,&addrlen))==-1){
            perror("Error at accept().\n");
            continue;
        }
        threadArg=(struct threadInfo*)malloc(sizeof(struct
    threadInfo));
        threadArg->idThread=id++;
        threadArg->clientDescriptor=clientDescriptor;
        pthread_t t;
        if(pthread_create(&t,NULL,&threadRoutine,threadArg)){
            perror("[server]Error at pthread_create().\n");
        }
    }
}
```

Fiecare thread are asociată o structură în care sunt reținute informații despre jucător:

```
typedef struct threadInfo{
    unsigned int idThread;
    int clientDescriptor;
    int session;
    int playerNumber;
    char username[50];
    int points;
    int startQN;
}threadInfo;
```

Serverul și clientul comunică pe baza unui protocol, care definește următoarele specificații: cererile trimise către server sunt șiruri de caractere delimitate de *new line*, fiecare punct terminal va prefixa mesajul pe care urmează să-l trimită cu dimensiunea acestuia, iar clientul va afișa fiecare răspuns primit de la server.

Serverul știe să răspundă la urmatoarele comenzi:

- "start:username" - prin intermediul acestei cereri clientul anunță serverul că dorește să înceapă jocul, menționând și numele de utilizator pe care vrea să-l aibă pe parcursul jocului;
- "R:<letter>" - pentru a răspunde la o întrebare din cadrul chestionarului, clientul trebuie să trimită litera corespunzătoare răspunsului corect sub această formă; Această comandă poate fi trimisă doar după ce a fost efectuată anterior comanda *"start:username"*, altfel un mesaj de eroare va fi transmis.
- "leave" - pentru a-și exprima dorința de a părăsi jocul cât timp acesta se află în derulare clientul poate trimite această comandă pentru a înștiința serverul. De asemenea, dacă clientul trimite această comandă înaintea celei de start, se consideră că acesta nu mai dorește să participe la joc și se eliberează conexiunea dintre cele două puncte terminale.

```
void *threadRoutine(void* arg)
{
    struct threadInfo info; info=*((struct threadInfo*)arg);
    while(true){
        myRecv(info.clientDescriptor);
        if(strncmp(command,"start:",6)==0){
            setUsername(command,length,&info);
            configurePlayer(&info);
            while(flag[info.session]==false){
                //verifica daca este ceva de citit pe socket
            }
            startQuiz(info); //incepe jocul
        }
        else if(strncmp(command,"leave",5)==0){
                handlerLeave(info);
        }
        else{
            mySend(info.clientDescriptor); //mesaj de eroare
        }
    }
}
```

Partea de configurare a jucătorului constă în:

```
void configurePlayer(struct threadInfo* info)
{
    pthread_mutex_lock(&lock);
    info->session=sessionNumber;
    counter++; //s-a mai alaturat un jucator
    info->playerNumber=counter;
    if(counter==3){
        flag[sessionNumber]=true; //poate incepe sesiunea de joc
        sessionNumber++;
        counter=0;
    }
    pthread_mutex_unlock(&lock);
}
```

Din momentul în care a început jocul, comenzile disponibile sunt:

```
void startQuiz(struct threadInfo info)
{
    for(int i=info.startQN;i<info.startQN+8;i++)
    {
        correctAnswer=sendQuestion(info,i); //interogare baza de date
        /*asteapta douazeci de secunde raspunsul*/
        myRecv(info.clientDescriptor);
        if(strncmp(command,"R:",2)==0){
            if(command[2]==correctAnswer){
                    updateScore(info);
                }
            }
        }
        else if(strncmp(command,"leave",5)==0){
            handlerLeave(info);
        }
    }
}
```

Răspunsurile trimise către client vor fi de forma:

- "Q:<question>" - întrebările urmate de variantele de răspuns vor fi trimise către participanții jocului în această formă. Fiecare întrebare are patru variante de răspuns (a,b,c și d) iar răspunsul corect este format dintr-o singură opțiune.
- "info:<text>" - dacă înaintea începerii jocului sau pe parcursul acestuia serverul dorește să trimită un mesaj către client în care să-l înștiințeze despre momentul în care va începe jocul sau alte detalii necesare, notificarea va fi transmisă în acest mod;

– "error:<text>" - acest tip de răspuns va fi folosit pentru trimiterea unui mesaj de eroare către client;

– winner::<text>"-anunță numele câștigătorului rundei cât și punctajul obținut de client. În cazul în care clientul este câștigător un mesaj aferent este trimis.

– "end:<text>" - anunță sfârșitul jocului; dacă pe parcursul jocului clientul trimite o cerere de părăsire, mesajul în care serverul menționează că a primit cererea este trimis tot în această formă.

5 Concluzii

Soluția propusă ar putea fi îmbunătățită prin divizarea chestionarelor pe diverse domenii, clientul având oportunitatea de a-l alege pe acela la care consideră că va putea acumula un punctaj cât mai mare.

O altă optimizare ar consta în crearea unui cont de utilizator pentru fiecare client care se conectează la server. Astfel, s-ar putea reține pentru fiecare jucător scorul pe care l-a obținut la rundele la care a participat anterior, întrebările la care a răspuns precum și varianta corectă corespunzătoare. În funcție de numărul total de puncte pe care îl acumulează, clientul poate avea acces la un magazin de recompense care oferă următoarele beneficii: eliminarea a două variante de răspuns, oferirea unui indiciu sau trimiterea răspunsului corect(cea mai costisitoare opțiune).

În plus, săptămânal s-ar putea organiza o sesiune de joc între primii patru jucători aflați în fruntea clasamentului săptămânii anterioare, detaliile legate de organizare fiind trimise prin e-mail participanților. Aceste tipuri de competiții ar putea fi organizate și pentru participanții aflați în fruntea clasamentului pe o perioadă mai îndelungată de timp.

6 Bibliografie

1. Specificaţia protocolului TCP (traducere de Cătălin Bulancea)
2. Andrew S. Tannenbaum ("2004"). Retele de calculatoare. Editura Byblos.
3. Allen, Grant; Owens, Mike (5 noiembrie 2010). The Definitive Guide to SQLite
4. Krause, Andrew (23 April 2007), Foundations of GTK+ Development