OnlineInfoOlympiad (B)

Cotos Stefan

Facultatea de Informatică, Universitatea Alexandru Ioan Cuza

1 Introducere

1.1 Viziunea generală

Acest proiect va fi alcătuit dintr-un program server și un program client, așadar este un model client-server, în care serverul implementat tratează în mod concurent clienții. Se va realiza o olimpiadă de informatică on-line, la care vor participa un număr fix de clienți care primesc anumite probleme pe care trebuie să le rezolve, dupa să trimită fișierele, iar la final după ce serverul corectează toate rezolvările, toți clienții vor primii o listă cu rezultatele.

1.2 Obiectivele proiectului

Are ca scop impementarea corectă a concurenței în server, pentru a primii mesaje de la orice utilizator care trimite anumite informații în orice moment și transmiterea corectă a datelor între clienți și server, fără pierderi. Clienții vor trebuii să rezolve cât de bine pot ei problema pentru a obține un punctaj cât mai mare și un loc cât mai sus în clasament. Serverul trebuie să aibă configurate diferite fișiere pentru transmiterea anumitor informații către clienți. Și trebuie să analizeze și să corecteze în mod cât mai corect rezolvările primite de la toți utilizatorii.

2 Tehnologii Aplicate

În acest proiect folosesc protocolul **TCP** pentru comunicarea între server și clienți, pentru că vreau o comunicare sigură în care nu se pierd date în timpul transmiterii datelor de la server la clienții. Este un **TCP** în care serverul tratează clienții în mod concurent, adică atunci când sunt mai mulți clienți conectați și unul dintre aceștia trimite un mesaj, informația este receptată un acel moment, iar de restul se ocupă când vor trimite și aceștia date. Acest procedeu folosit se numește multiplexare, care este realizat cu ajutorul funcției **'select()'**. Această funcție permite să monitorizeze multipli descriptori, lucru care este esențial în acest proiect. Acest proiect utilizează și baze de date **SQLite**, în care se afla toate problemele pregătite pentru clienți, și diferite fișiere input, output pentru corectare. Folosesc și funcția **'rand()'** pentru a extrage o problemă diferită pentru fiecare client. De asemenea mai folosesc și **semnale**, pentru a monitoriza dacă cumva un client vrea să se deconecteze. O altă funcție folosită este **dup2()**, o folosesc în momentul în care serverul corectează problemele, pentru a vedea dacă scriptul care le compilează și le rulează s-a terminat cu succes.

3 Structura Aplicației

3.1 Diagrama aplicației

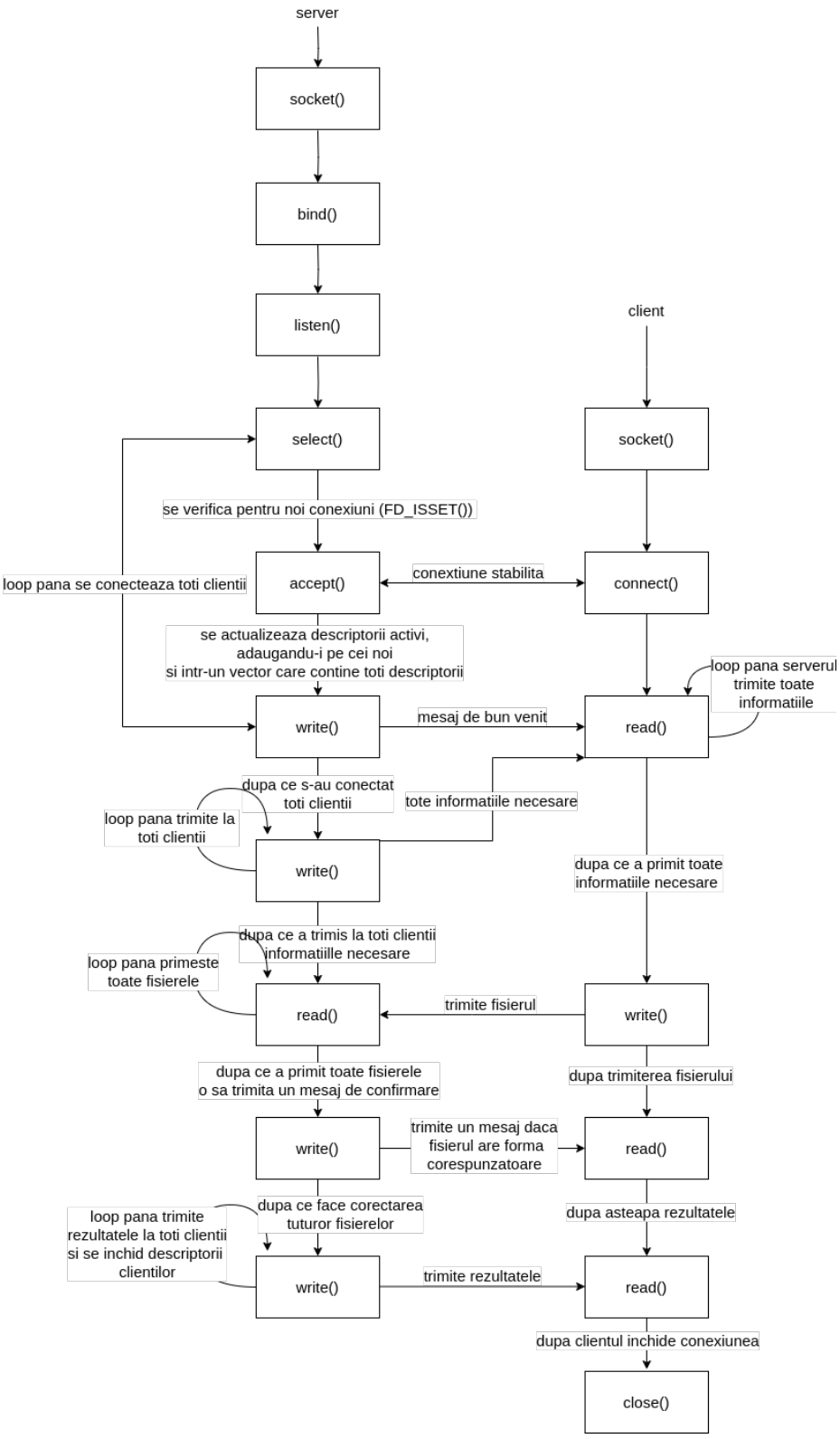

Fig. 1. Diagrama detaliată a aplicației.

3.2 Expunerea conceptelor

La început serverul se ocupă de crearea unui socket prin care o să comunice cu clienții, atașeaza socket-ul folosind **bind()**, și după începe 'ascultarea', adică dacă sunt clienți care o să vină să se conecteze, cu **listen()**. După acești pași, în server se folosește funcția **select()**, pentru implementarea multiplexării și se începe acceptarea clienților. Clientul de asemenea creează un socket prin care va comunica cu serverul la un anumit **PORT** și un anumit **IP**. Și între server și respectivii clienți se vor trimite mesajele aferente pentru acest proiect cu funcțiile **read()** și **write()**, exemplificate detaliat în diagrama de mai sus. La final, după ce clienții au terminat de trimis și primit date, se vor deconecta și serverul de asemenea va închide conexiunea catre aceștia.

4 Aspecte de Implementare

4.1 Exemplificarea anumitor secțiuni din cod

O să exemplific anumite porțiuni din cod care sunt specifice acestui proiect.

```
int rc = sqlite3_open("probleme.db", &db); //"probleme.db" este o baza de date care contine

if (rc != SQLITE_OK)
{
    fprintf(stderr, "[server]Nu se poate deschide baza de date: %s\n", sqlite3_errmsg(db));
    return rc;
}
```

```
nr = 1 + (rand() % 10);
numbers[j]=nr;

strcpy(comm, "SELECT problema,input,output,timp FROM probleme WHERE id = ");
sprintf(num, "%d", nr);
strcat(comm, num);
strcat(comm, ";");
const char *selectDataSQL = comm;
rc = sqlite3_exec(db, selectDataSQL, callback, 0, &errMsg);

if (rc != SQLITE_OK)
{
    fprintf(stderr, "Eroare la SQL: %s\n", errMsg);
    sqlite3_free(errMsg);
}
```

Aceste porțiuni de cod reprezintă deschiderea bazei de date folosite, și după alegerea unui număr random se folosește cuvântul cheie specific SQLite **'SELECT'** pentru a selecta o problema din baza de date, și este apelată o funcție denumită **'callback'**, în care se preia rezultatul interogării și se pun în diferite variabile pentru a fii utilizate mai târziu la trimiterea către participanți.

```
char rezolvare[100];
strcpy(rezolvare, "./exec.sh "); //un script care compileaza si ruleaza fiecare problema
strcat(rezolvare, "prob");
char n[2];
sprintf(n,"%d",numbers[i]);
strcat(rezolvare, n);
system(rezolvare);
```

```
time(&start_time);
do
{
  time(&current_time);
  elapsed_time = difftime(current_time, start_time);

  if (elapsed_time >= TIMP)
  {
    break;
  }

  fd_set actfds;
  FD_ZERO(&actfds);
  FD_SET(STDIN_FILENO, &actfds);

  struct timeval tv;
  tv.tv_sec = 0;
  tv.tv_usec = 0;

  int r = select(STDIN_FILENO + 1, &actfds, NULL, NULL, &tv);

  if (r > 0 && FD_ISSET(STDIN_FILENO, &actfds))
  {
    fflush(stdout);
    read(0, msgrasp, 100);
    ok = 1;
    break;
  }

  usleep(10000);
} while (1);
```

În prima imagine de mai sus putem vedea cum se execută un script **'exec.sh'** pentru compilarea și rularea problemelor primite de la participanți.

Iar în a doua secțiune de cod implementez un **'timer'** deoarece clienții au un timp limitat de a introduce fișierul cu rezolvarea, acesta este implementat cu funcția **select()** deoarece cu ajutorul acesteia, programul nu mai așteaptă la infinit clienții să introducă de la tastatură fișierul și are doar un timp limitat.

4.2 Protocolul aplicației

În acest proiect server-ul trimite întai un mesaj de bun venit clienților (participanților) conectați, trebuie să se conecteze un anumit număr de clienți pentru a începe olimpiada, după ce s-au conectat toți clienții, serverul le va trimite fiecăruia o problema aleasă random dintr-o bază de date, cum am precizat mai sus, împreuna cu fișierele de input, output respective pentru testarea corectitudinii rezolvării la final și diferite informații pe care un participant trebuie să le aibă în vedere. Clientul preia aceste informații, afișează cerința problemei, precum și fișierele de input și output. Și acum are la dispoziție un anumit interval de timp pentru a rezolva și trimite înapoi server-ului fișierul cu rezolvarea, numele fișierului trebuie să aibă un anumit format, menționat de către server, în caz contrar nu va fii acceptat. După expirarea timpului acordat, clienții nu vor mai putea trimite rezolvarea, iar dacă nu au trimis până în acel moment, vor avea punctajul 0. Serverul la primirea fișierul respectiv, afișează conținutul acestuia. Iar după primirea tuturor fișierelor, le compilează, le rulează, și calculează punctajul participanților după corectitudinea acestora în funcție dacă s-au compilat cu succes și de câte linii au fișierele primite față de cele cu rezolvarea corectă. Și la final, server-ul trimite participanților(clienților) lista cu rezultatele. La olimpiadă pot participa un număr fix de clienți, de aceea dacă în timpul olimpiadei, după ce toți s-au conectat, un alt client încearcă conectarea, aceasta îi va fi refuzată de către server. Iar dacă un client dorește să se deconecteze pe parcurs acesta o poate face după ce s-au conectat toti participanții, iar acest lucru nu va afecta desfășurarea olimpiadei, doar că respectivul participant va avea punctajul 0.

4.3 Scenarii reale de utilizare

Aceste idei s-ar putea utiliza și la o olimpiada adevarată, dacă va fi nevoie să se țină on-line, deoarece toți participanții trebuie să fie prezenți și 'conectați' la olimpiadă și să respecte anumite reguli impuse. Și ei vor avea la fel un anumit timp in care să rezolve problema/problemele. După care trebuie să trimită rezolvarea, primind după și ei un rezultat dupa ce toate problemele au fost analizate și corectate. Sau nu neapărat la olimpiade, la fel și la diferite concursuri din orice categorie se aplică acești pași.

5 Concluzii

La această soluție se poate îmbunatații modul de corectare a problemelor pentru a fii cât mai precis și a oferii participanților cele mai bune rezultate. Ca o

îmbunătățire semnificativă ar fii și implementarea unei interfețe grafice pentru automatizarea olimpiadei și o mai bună desfășurare a acesteia.

6 Referințe Bibliografice

1. https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
2. https://www.andreis.ro/teaching/computer-networks