Tema 2 - Documentație OnlineInfoOlympiad

Renghiuc Bianca Elena A2

08.12.2022

Abstract

În această documentație voi prezenta modul de implementare al aplicației Online-InfoOlympiad, printr-un model server-client ce folosește un protocol TCP concurent cu thread-uri. Serverul va folosi o bază de date SQLite pentru a trimite probleme clientului, iar după îi va corecta sursele si îl va puncta corespunzător.

1 Introducere

OnlineInfoOlympiad presupune implementarea unui model client-server pentru o olimpiadă de informatică on-line. Serverul va trimite clienților cerințele unor probleme alese random dintr-o bază de date și alte detalii referitoare la acestea. Clienții vor raspunde serverului cu un fișier text în care se află sursa la problema ce trebuie rezolvată, iar serverul va compila și puncta aceste surse (în funcție de corectitudinea obținerii unor output-uri, pentru anumite input-uri, și în funcție de timplul de executare al algoritmului).

2 Tehnologiile utilizate

Pentru a mă asigura că toate mesajele ajung de la client la server și invers, fără a se pierde, voi folosi protocolul TCP. Folosirea acestuia este necesară pentru a ne asigura că fiecare client primește problema ce trebuie rezolvată. De asemenea, trebuie să asigurăm și primirea rezolvării de la client pentru a putea fi compilată și corectată.

Pentru ca serverul nostru sa fie concurent vom folosi thread-uri. Astfel, clientii nu vor fi preluați în ordinea conectării, ci pe măsură ce trimit date către server. Răspunsurile clienților vor fi gestionate simultan, nefiind nevoie să ne ocupăm pe rând de câte un client în parte, ca la serverele de tip iterativ.

De asemenea, voi folosi o bază de date SQLite pe care o voi gestiona cu ajutorul librăriei sqlite3.h. Aici voi crea un tabel în care se vor regăsi următoarele coloane: cerința problemei, diferite input-uri și output-uri, timpul de rezolvare și timpul de executare al codului.

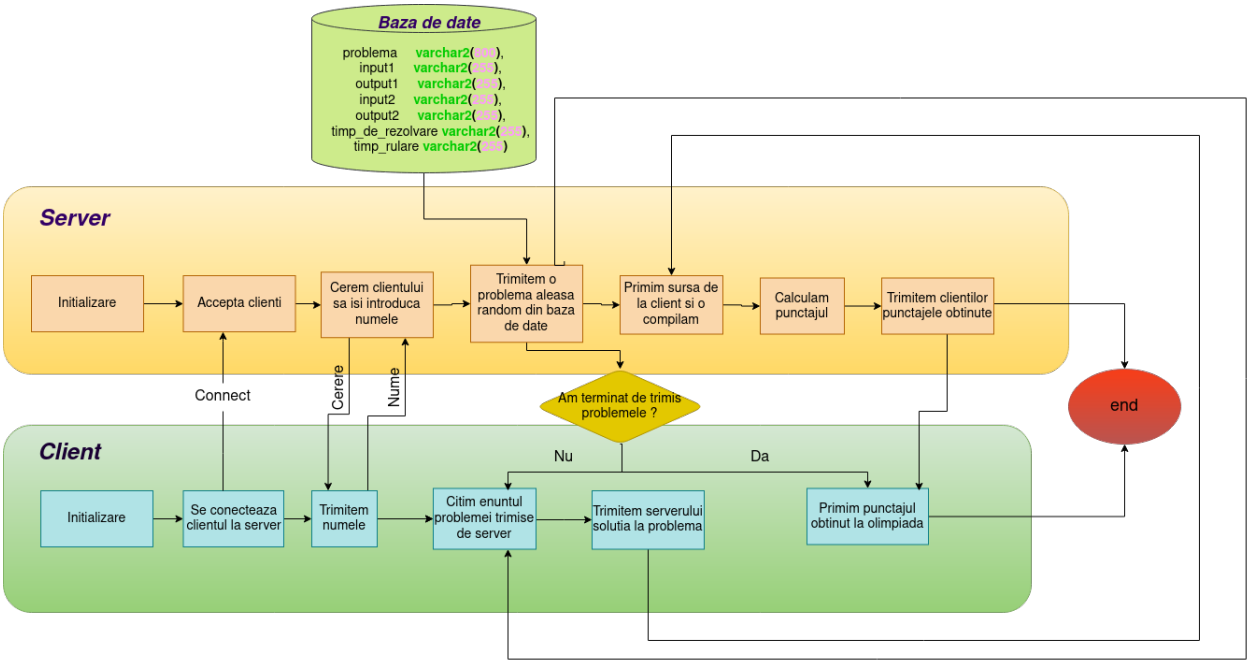
3 Arhitectura aplicației

Aplicația este formată din 3 fișiere: cel pentru server, cel pentru client și cel care conține baza de date.

Partea de client se va ocupa de trimiterea unei cereri de conectare și de trimiterea surselor problemelor ce trebuiesc rezolvate.

Partea de server se va ocupa de acceptarea clienților ce se conectează în timpul de conectare, de crearea thred-urilor ce ne vor ajuta să transmitem către clienți cerințele problemelor. După care, se va ocupa de corectarea surselor clienților și de calcularea punctajelor.

Baza de date va fi gestionată tot de către server și se va ocupa de stocarea cerințelor și detaliilor despre problemele de la olimpiadă.



4 Detalii de implementare

Vom face ca socket-ul din server să fie nonblocant pentru a putea accepta numar nelimitat de clienți.

```
1 int flag = fcntl(sd, F_GETFL, 0);
2 fcntl(sd, F_SETFL, flag | O_NONBLOCK);
```

Acceptarea clienților se face într-o buclă while(1), iar când timpul a expirat, nu mai pot fi acceptați clienți.

```
if (start_olimpiada < time(NULL) - 30)
        {
            printf("Nu mai pot fi acceptati clienti.\n");
            fflush(stdout);
            break;
        }
        else
        {
            // acceptam un client
            client = accept(sd, (struct sockaddr *)&from, &length);
            if (client < 1)
            {
                if (errno == EWOULDBLOCK)
                    continue;
                else
                {
                    perror("Eroare la acceptarea conexiunii.");
                    continue;
                }
            }
        }
```

Vom deschide baza de date pentru a putea extrage date despre probleme.

```
int baza = sqlite3_open("data", &bd);
    //deschidem baza de date
    if (baza != SQLITE_OK)
    {
        fprintf(stderr, "Cannot open database: %s\n", sqlite3_errmsg(bd));
        sqlite3_close(bd);
    }
```

Luăm din baza de date cerința unei probleme, pentru a o putea trimite clienților.

```
bzero(numar_problema, 4);
bzero(sel, 800);
bzero(temp, 800);

strcpy(sel, "SELECT problema FROM olimpiada LIMIT 1 OFFSET ");
strcat(sel, numar_problema);
strcat(sel, ";");
```

```
8
9   baza = sqlite3_exec(bd, sel, callback, temp, &err);
10
11  if (baza != SQLITE_OK)
12  {
13      fprintf(stderr, "SQL error1: %s\n", err);
14      sqlite3_free(err);
15  }
```

Punem valorile din coloanele input1, output1 din tabelul olimpiada în două fișiere de input și de output pentru a putea rula sursele clienților. Ulterior voi face același lucru și cu coloanele input2, output2.

```
1   bzero(sel, 100);
2   bzero(_in, 100);
3   strcpy(sel, "SELECT input1 FROM olimpiada LIMIT 1 OFFSET ");
4
5   strcat(sel, numar_problema);
6   strcat(sel, ";");
7
8   rc = sqlite3_exec(bd, sel, callback, _in, &err);
9
10
11  if (rc != SQLITE_OK)
12  {
13      fprintf(stderr, "SQL error3: %s\n", err);
14      sqlite3_free(err);
15  }
16
17  FILE *fp1;
18  char filename1[100]="input.txt";
19
20  fp1 = fopen(filename1,"w+");
21
22  if ( fp1 )
23      {
24          fputs(_in,fp1);
25      }
26  else
27      {
28          printf("Failed to open the file\n");
29      }
30
```

Serverul va compila sursele clienților folosindu-se de comanda system() și va cal-

cula timpul de exutare al programului trimis de clienți, folisindu-se de gettimeofday().

```
gettimeofday(&start, NULL);

char nume_fara_cpp[30];
strcpy(nume_fara_cpp, nume);
nume_fara_cpp[strlen(nume_fara_cpp)-4]='\0';

char buff[100]="";
strcat(buff, "g++ -o "); //g++ -o file file.cpp
strcat(buff, nume_fara_cpp);
strcat(buff, " ");
strcat(buff, nume);
strcat(buff, "\0");
system(buff);
bzero(buff, 100);

strcat(buff, "./");  //./file < input.txt > output.txt
strcat(buff, nume_fara_cpp);
strcat(buff, " < input.txt > output_pb.txt\0");
system(buff);
bzero(buff, 100);

gettimeofday(&stop, NULL);
```

Dacă output-ul corect este același cu cel obținut prin rularea algoritmului clientului, atunci îi creștem punctajul.

```
int line, col, diff;
diff = compareFile(fp2, fp3, &line, &col); //functie implementata

fclose(fp2);
fclose(fp3);
fclose(fp1);
if (diff == 0)
{
    punctaj+=40;
}
```

5 Concluzie

Aplicația ar putea fi îmbunătățită prin afișarea unui clasament al punctajelor sau prin afișarea clienților ce obțin premii (primii 3).

Un alt aspect ce ar putea apărea ar fi prezența unui timer care să îi spună clientului cât timp mai are până la finalizarea probei.

Pentru a avea o viteza de transmitere a datelor mai bună, am fi putut utiliza protocolului UDP în loc de cel TCP. Dar, deși ar fi fost mai avantajos din punctul de vedere al vitezei de transmisie, protocolul UDP nu oferă siguranța transmiterii tuturor datelor.

Alte îmbunătățiri ar putea fi să se creeze o interfață grafică, sau afișarea la sfârșitul olimpiadei pentru fiecare client a modului de corectare și a greșelilor făcute.

References

[1] Server TCP concurent
`https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c`

[2] Client TCP concurent
`https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c`

[3] SQLite
`https://www.geeksforgeeks.org/sql-using-c-c-and-sqlite/`