Documentatie proiect OnlineInfoOlympiad

Fechita Antonio

Facultatea de Informatica, UAIC

1 Introducere

Am ales acest proiect deoarece consider ca are o utilitate practica si de asemenea prezinta o buna modalitate pentru a exersa cunostintele acumulate despre comunicarea intre un server si un client prin socketi TCP.

In realizarea acestui proiect imi propun sa ofer utilizatorilor un mediu prin intermediul caruia pot concura la o olimpiada online de informatica la care vor obtine un punctaj pe baza solutiilor trimise de acestia pentru o problema de informatica aleasa aleator dintr-o lista de probleme si un clasament al rezultatelor anonim sau nu in care fiecare utilizator sa isi cunoasca punctajul personal.

2 Tehnologii utilizate

Pentru comunicarea dintre client si server am considerat ca protocolul TCP este mai potrivit decat UDP deoarece siguranta ca mesajele sunt primite corect si fara erori este mai importanta decat viteza cu care are loc acest transfer, intru cat utilizatorii sa nu fie depunctati datorita erorilor care pot altera codurile sursa trimise catre server.

3 Arhitectura Aplicatiei

Aplicatia este compusa din urmatoarele componente: Client, Server, fisier de configurare, fisier cu probleme, perechi de fisiere input – output pentru fiecare problema in parte, fisierele in care sunt stocate pentru fiecare utilizator outputurile obtinute de solutia lui si protocolul TCP pentru comunicarea dintre client si server.

3.1 Clientul

Clientul are rolul de a interactiona cu utilizatorul. Prin intermediul acestuia utilizatorul primeste enuntul problemei, exemple de input si output si timpul pe care il are la dispozitie pentru a rezolva problema propusa. De asemenea utilizatorului I se solicita sa scrie rezolvarea intr-un fisier predefinit, astfel incat clientul sa citeasca intr-un buffer continutul fisierului si sa il trimita mai departe catre server print-un socket TCP pentru a asigura corectitudinea mesajului primit de server.

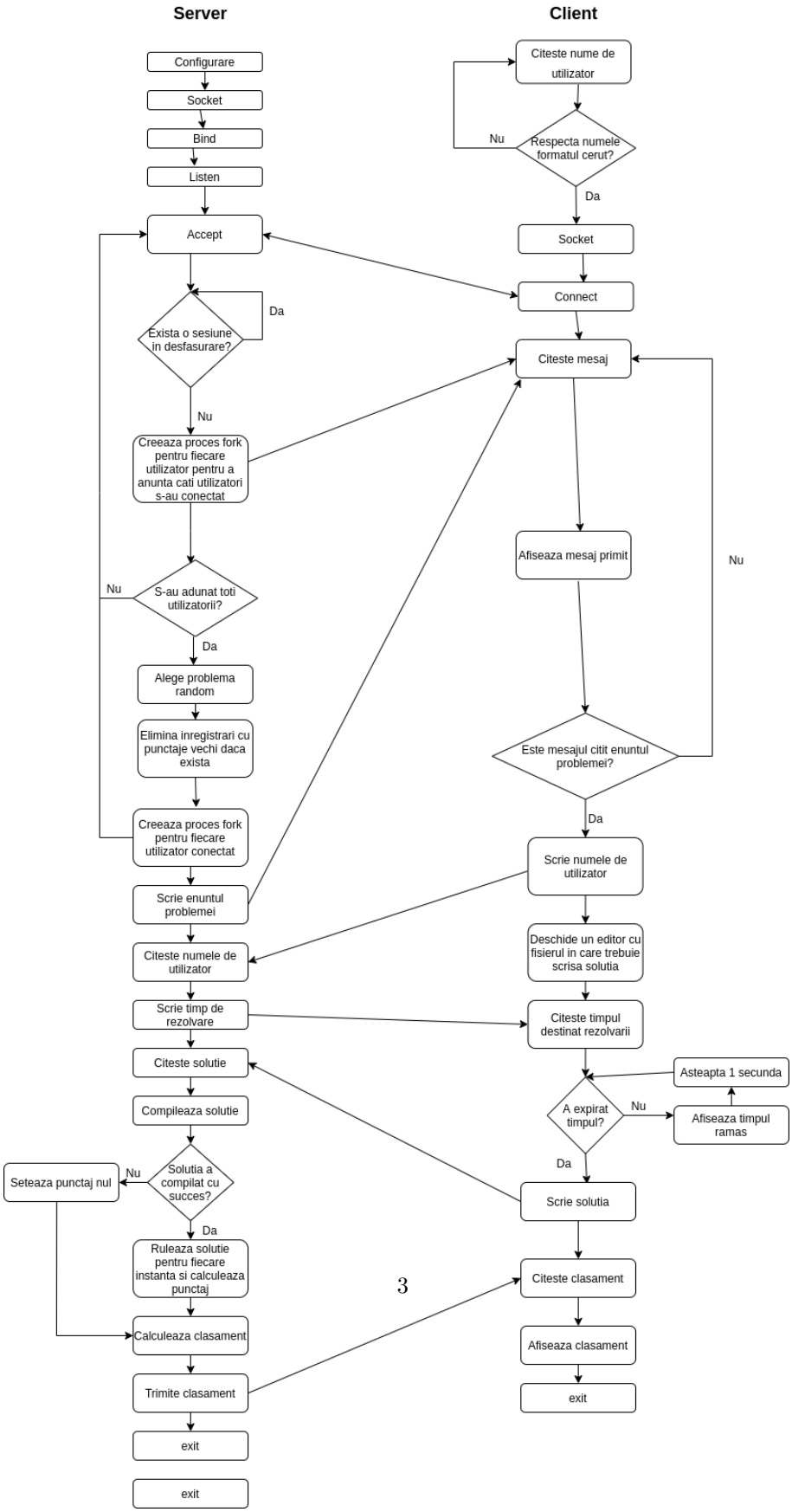
3.2 Serverul

Serverul de tip TCP concurent alege la intamplare o problema din fisierul cu lista de probleme si o trimite catre toti clientii in momentul in care s-a strans numarul asteptat de participanti din fisierul de configurare, in caz contrar utilizatorii suplimentari sunt anuntati de catre server ca s-a atins deja numarul maxim de utilizatori si sunt pusi in asteptare pentru urmatoarea sesiune. Serverul compileaza fiecare cod sursa pe care il primeste de la clienti, apoi ruleaza solutiile si compara rezultatele obtinute de acea solutie cu cele din fisierul de output corespondent fisierului de input folosit pentru testarea corectitudinii. Dupa aceasta testare serverul calculeaza un punctaj pentru fiecare utilizator in parte si le trimite tuturor participantilor lista cu rezultatele obtinute de toti utilizatorii.

3.3 Fisierul de configurare

Fisierul de configurare contine informatii precum numar numarul de participanti asteptati, setarea de anonimitate a rezultatelor, iar pentru fiecare problema in parte timpul alocat rezolvarii in secunde, numarul de perechi de fisiere input-output pentru testare, dar si timpii in care se poate incadra o solutie la executie pentru puncte suplimentare.

3.4 Diagrama aplicatiei



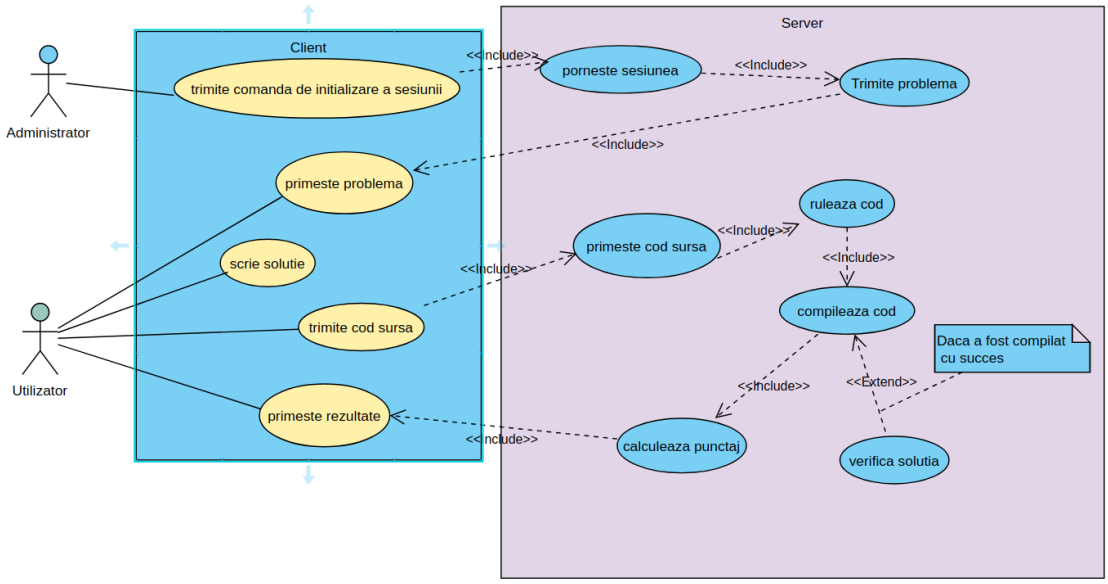
4 Detalii de implementare

4.1 Cod specific proiectului

```
//        Urmatorul fragment de cod din server creeaza pentru fiecare utilizator un fisier
//care contine codul sursa trimis de catre acesta, care urmeaza sa fie compilat si rulat.

//pregatim fisier pentru codul sursa al utilizatorului
int myOwnPid = getpid(); //folosim pid-ul procesului care
char numeFisierCodSursa[50]; //se ocupa de un anumit user pe post de identificator
sprintf(numeFisierCodSursa, "userInput%d.c", myOwnPid);
int fd = open(numeFisierCodSursa, O_RDWR | O_TRUNC | O_CREAT, 0777);
write(fd, msg, strlen(msg)); //scriem in noul fisier codul sursa primit de la client
char numeExecutabil[50];
sprintf(numeExecutabil, "userInput%d", myOwnPid); //

//compilam codul intr-un alt proces copil
int pid1;
if ((pid1 = fork()) == -1) {
  perror("Eroare la fork()");

} else if (pid1 == 0) {
  // copil
  close(client);

  execlp("gcc", "gcc", numeFisierCodSursa, "-o", numeExecutabil, NULL);
  exit(0);
}
// parinte
wait(NULL); //asteptam finalizarea compilarii efectuata de copilul procesului curent

//codul a fost compilat daca a fost sintactic corect, urmeaza sa fie rulat
//in urmatorul proces copil

int pid2;
if ((pid2 = fork()) == -1) {
  perror("Eroare la al doilea fork()\n");
  exit(-1);
} else if (pid2 == 0) {
  // copil
  //verifica daca exista fisierul de executat
  int a;
  if ((a = access(numeExecutabil, F_OK)) != -1) {
    perror("Codul sursa a fost compilat cu succes!");
    //urmeaza pasul de rulare
    //...
```

```
  } else {
    printf("Codul sursa nu a fost compilat cu succes!\n");
  }

} else if (pid2 > 0) {
  // parinte (copilul primului parinte)
  wait(NULL); //asteptam terminarea pasului de rulare a executabilului

// Functia urmatoare alege la intamplare indexul unei probleme din lista de probleme si
//parcurge lista de probleme salvand intr-un buffer enuntul problemei alese

int alegeRandomProblema(char problema[500],int nrProbleme,char listaProbleme[10005])
//pune in bufferul "problema" textul unei probleme alese la intamplare
{
  srand(time(NULL));
  int indexProblemaSelectata = rand()%nrProbleme + 1;
  //alegem la intamplare numarul unei probleme din lista problemelor

  int n=strlen(listaProbleme);
  int k=0;
  int problemaCurenta=0;
  //tine evidenta problemei pe care o citim la un moment dat

  for(int i=0;i<n;i++)
  {
      if(listaProbleme[i]=='#') //problemele din fisier sunt separate prin simbolul '#'
      problemaCurenta++;
      if(problemaCurenta==indexProblemaSelectata && listaProbleme[i]!='#')
      //daca citim problema aleasa din lista o salvam in bufferul problema

      {
          problema[k]=listaProbleme[i];
          k++;
      }
      else if(problemaCurenta>indexProblemaSelectata)
      //daca am terminat de citit problema aleasa ne putem opri din citit lista de probleme

          {
                  problema[k]=NULL;
                  break;
          }

  }
  return indexProblemaSelectata;  //returnam un identificator al problemei selectate
}
```

4.2 Scenarii de utilizare



Utilizatorul este indrumat de aplicatie sa scrie rezolvarea problemei primite intr-un fisier creat de client, si exista urmatoarele scenarii:

- utilizatorul trimite o solutie care compileaza → serverul calculeaza un punctaj in functie de corectitudinea si viteza de executie a solutiei utilizatorului
- utilizatorul trimite o solutie care nu compileaza → serverul calculeaza un punctaj nul
- utilizatorul sterge sau redenumeste fisierul in care ar trebui scrisa rezolvarea → mesajul trimis de client catre server este considerat a fi un cod sursa gresit sintactic, iar utilizatorul primeste punctaj nul, fiind atentionat printr-ul alt mesaj ca fisierul solutiei a fost alterat
- utilizatorul se conecteaza la server dupa momentul in care au fost trimise catre clienti subiectele → primeste de la server un mesaj care il anunta ca este in asteptare pentru urmatoarea sesiune

5 Concluzii

Proiectul ar putea fi imbunatatit prin adaugarea unui sistem de logare al utilizatorilor astfel incat un anumit utilizator sa aiba acces la un istoric al rezultatelor sale la olimpiade la care a participat in trecut, si alte statistici personale. De asemenea ar putea exista un utilizator special care sa aiba rol de administrator avand acces la o serie mai larga de comenzi prin intermediul carora sa poata modifica anumite date din fisierul de configurare, sau fisierul cu lista de probleme, ori ar putea elimina in timpul competitiei anumiti utilizatori daca considera ca este necesar. De asemenea am putea utiliza threaduri pentru a servi utilizatorii in

locul proceselor create prin fork(). Utilizatorii care s-ar conecta dupa inceperea concursului ar putea primi un timp estimativ in care va fi disponibila urmatoarea sesiune la care pot participa. O alta imbunatatire ar fi implementarea unei interfate grafice mai intuitiva pentru utilizator.

6 Bibliografie

- TCP vs UDP: https://www.vpnmentor.com/blog/tcp-vs-udp/
- Kill: https://man7.org/linux/man-pages/man2/kill.2.html
- Remove: https://www.tutorialspoint.com/c_standard_library/c_function_remove.htm
- Rename: https://www.tutorialspoint.com/c_standard_library/c_function_rename.htm
- sprintf: https://www.tutorialspoint.com/c_standard_library/c_function_sprintf.htm
- waitpid: https://linux.die.net/man/2/waitpid
- TCP concurent: https://profs.info.uaic.ro/ georgiana.calancea/Laboratorul_7.pdf