Mypbinfo - Evaluator de probleme informatice

Uliuliuc Serafim

Decembrie 2020

1 Introducere

Proiectul acesta este foarte asemanator cu diverse platforme de învățare practică, prin rezolvarea de probleme, cum ar fi campion, pbinfo, etc. Astfel, a trebuit sa folosesc modelul client-server pentru a trimite o problemă de la server către clienți, iar după un anumit interval de timp să transmit rezolvarea (fișierul sursă) de la client înapoi la server, unde se va compila și apoi evalua după diferite perechi de input și output. Iar la final, serverul va trimite înapoi rezultatele cliienților obținute după evaluare.

2 Tehnologii utilizate

Pentru comunicare pe rețea am folosit protocolul TCP pentru a fi sigur de integritatea datelor trimise și receptate, nu se poate accepta eroare în trimiterea enunțului problemei și nici în trimiterea unui fișier sursă, poate cauza erori de compilare.

Asigurarea concurenței este îndeplinită de folosirea primitivei fork, pentru ca fiecare client să comunice cu procesul lui propriu din server (două procese pot surla simultan pe un procesor cu cel puțin două unități aritmetico-logice) și pentru a asigura o compilare individuală, unitară și simplă a surselor, asemenea pentru testare (fapt pentru care am folosit primitive din familia exec).

Problemele sunt stocate în fișiere text, care vor fi deschide odată cu pornirea serverului, iar descriptorii memorați într-un tablou.

Alegerea problemei ce se va trimite clinetului este aleasă în mid pseudo-aletor, dupa rezultatele returnate de primitiva random dupa operația modulo numărul de probleme disponibile.

Rezultatele vor fi scrise într-un fișier si apoi fiecare client va primi informațiile din acel fișier.

Am folosit pipe-uri pentru comunicare dintre procesele care rulează azplicația dată de către client și procesul care face evaluarea totală a soluției.

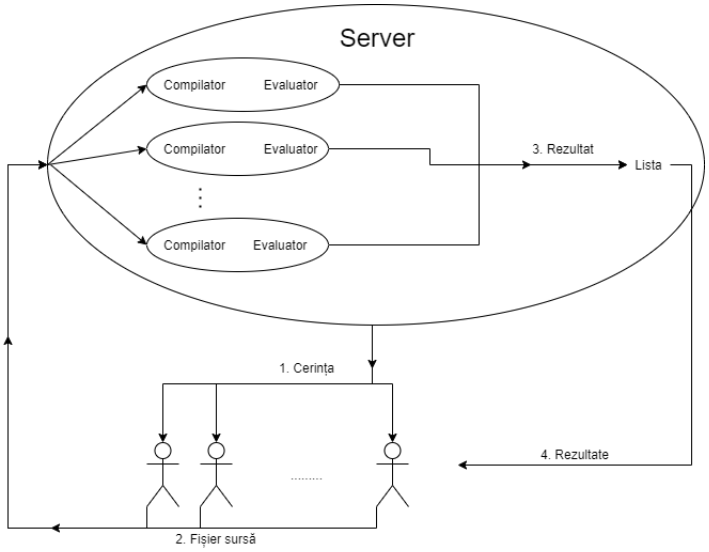
Primitivele dup2 asigură direcționarea fluxului de date stabilite standard (eroare, intrare și ieșire) către fișierele dorite, astfel se simplifică apelurile primitivelor din familia execl.

Folosirea primitivei pipe a asigurat comunicarea între două procese diferite, fără a mai trebui să creez un alt fișier pe disk.

Astfel, am folosit predominat primitivele read și write pentru cominicare dintre socketi și pentru citirea și, respectiv scrierea din anuminte fișiere.

3 Arhitectura aplicației

Figure 1: Diagrama generală de utilizare



La pornirea serverului se vor deschie mai multe fișiere: fișierul de configurare din care vor fi inițializate variabilele ce reprezintă numărul de clienți, numărul de probeme și fișierele ce conțin date despre fiecare problemă în parte (enunțul, timpul disponibil, exemplu de intrare-ieșire). Va seta informațiile necesare protocolului TCP și va aștepta numărul dat de clienți.

Odată ce un client se conectează la server, acesta creează un număr aleator ce semnifică indexul problemei pe care acest client trebuie să o rezolve, salveaza datele de conectare pentru a putea trimite la final rezultatele și nou proces ce se va ocupa de trimitrea informațiilor necesare (inclusiv PID-ul ca o metoda de identificare în lista finală).

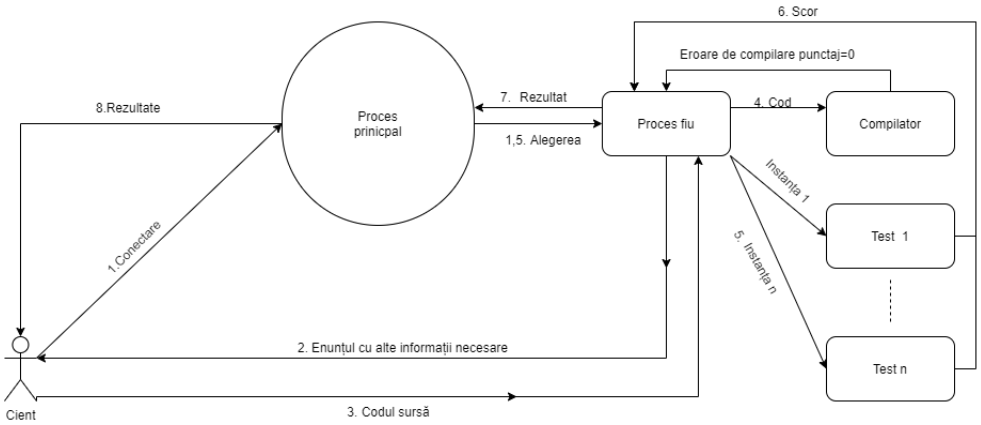
După expirarea timpului dat pentru rezolvarea problemei, clientul va trimite șirul de caractere ce reprezintă codul în C către server (procesul fiu care se ocupa de servierea acestuia) care va creea un alt proces cu scopul compilarii (in caz de eroare se va scrie punctajul 0 din faza aceasta), apoi pentru fiecare set de fișiere intrare-ieșire se va creea un alt proces care sa ruleze soluția compilată, după care se va face o sumă in funcție de dierențele dintre informațiile din fișierul cu teste si rezultatul dat.

La final, fiecare proces fiu relativ la cel principal va scrie intr-un fisier rezultatele obținute de fiecare identificat de PID, se închid toți descriptorii de fișiere și de socketi, și în final serverul.

În caz de eroare la apelul unei primitive, procesul va seta variabila errno corespunzător și își va încheia execuția.

4 Detalii de implementare

Figure 2: Diagrama detaliată de (scenarii) funcționare a aplicației



Modul în care se face alegerea unei probleme dintr-o listă de probleme:

```
problema=random()%nr_total_probleme;
```

Trasnsiterea fișierelor sursă, poate fi simulată cu ușurință prin trimiterea de siruri de caractere, partea de recepționare fiind asemănătoare:

```
while(read(fisier,&buf,MAX_SIZE)>0)
{
        write(socket,buf,strlen(buf));
}
```

Pentru partea de compilate am folosit un nou proces care apelează primitiva exec, iar răspunsul la compiiilare (fișierul 2, cel de eroare) va fi trimis automat intr-un fișier la care are acces procesul tată:

```
dup2(rezultat,2);
execlp(``gcc'',''-o'',''sol'',''sol.c'',0);
```

Iar evaluarea poate fi făcută prin creearea unui nou proces și apoi suprascrierea acestuia cu fișierul executabil generat de compilator:

```
dup2(aux,1);
dup2(test,0);
execlp(``sol'',''sol'',0);
```

După care fișierul aux va fi comparat cu rezultatul corect, iar rezultatul va fi cuantificat sub forma unui punctaj final cuprins între 0 și 100 care va fi trimis la final către toți clienții.

5 Concluzii

Această implementare face posibilă concurența prin folosirea primitivei fork care creează noi procese pentru fiecare client, de cele mai multe ori, utilizarea acesteri primitive este destul de costisitoare din punct de vedere al resursei timp, pentru eficientizare, se poate folosi implementarea concurentă prin folosirea firelor de execuție paralele ce beneficiază de memoria comună. Această imbunătățire ar aduce un timp de servire pentru server mai scăzut decît în implementarea cu procese.

Metoda de stocare folosită se bazează doar pe manipularea fișierelor de pe disc în manieră clasică. În cazul în care se vor aduna multe fișiere cu date despre probleme, acestea pot fi organizate mai bine folosind o bază de date relațională.

O altă posibilitate de îmbunătățire a acestui proiect este adăugarea mai multor funcționalități permise clientului: autentificarea cu un nume de utilizator și o parolă, reținerea și posibilitatea de afișare către client a datelor te tipul: numărul de probleme rezolvate, cîte puncte a acumulat în total, posibilitatea unui clasament ordonat după punctaje.

6 Bibliografie

https://sites.google.com/view/fii-rc/home
https://medium.com/@warren2lynch/use-case-learn-by-examples-5a63b67fa64d
https://app.diagrams.net/
https://artofproblemsolving.com/wiki/index.php/LaTeX:Symbols
https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings
https://www.guru99.com/linux-redirection.html
https://profs.info.uaic.ro/~computernetworks/
https://linux.die.net/
https://stackoverflow.com/questions/
https://www.overleaf.com/learn/latex