NTSF (Nu Te Supara Frate)

Andrei Stefan-Alexandru

Universitatea Alexandru Ioan Cuza, Facultatea de Informatica Iasi

1 Introducere

Nu te supara frate (NTSF) este un joc care poate fi jucat in minim 2 jucatori si in maxim 4.
Fiecare jucator trebuie sa dea cu un zarul, fiecare fata fiind numerotata de la 1 la 6.
In functie de numarul de pe fata de sus a zarului jucatorul isi va muta piesa pe tabla de joc pana la un spatiu predefinit.
Acest spatiu reprezinta sfarsitul si implicit obiectivul jocului, cine ajunge primul cu toate piesele in spatiul predefinit castiga jocul.
Sunt 4 piese pozitionate in spatiul de inceput si jucatorul trebuie sa ajunga cu toate cele 4 piese in spatiul de sfarsit.
O noua piesa poate fi adaugata pe tabla doar dupa ce piesa precedenta a ajuns la finish.

2 Tehnologii utilizate

Am ales un server de tip TCP concurent deoarece trebuie sa am cel putin 2 jucatori conectati in acelasi timp.
Clientul se conecteaza la server iar serverul asteapta pana cand cel putin 2 jucatori sunt conectati si pana cand toata lumea a dat "ready".

3 Arhitectura aplicatiei

Serverul contine o structura numita jucatori[] in care se retin date despre fiecare jucator in parte. Numarul maxim de jucatori este 4, fiecare avand campuri care pot fi modificate oricand.

```
typedef struct thData
{
    int idthread; //id-ul thread-ului
    int cl; //descriptorul intors de accept
    char piesa[4]; //culoarea piesei
    int pozitie; //pozitia piesei pe tabla
    char nume[20]; //numele jucatorului
    int rd; //0 daca jucatorul nu e ready, 1 altfel
    int wins; //maxim 4
```

```
    int qt; //1 daca jucatorul a dat quit
    int punct;
} thData;
struct thData jucatori[4];
```

Serverul foloseste arhitectura unui server tcp-concurent, utilizand functii precum: bind() pentru a-i da un port, listen() pentru a asculta potentialele conexiuni, si accept pentru a a accepta conexiunile. Serverulare un contor care verifica in permanenta cati clienti sunt conectati ca sa nu accepte conexiuni noi cand s-a atins numarul de 4 jucatori.

```
if (bind (sd, (struct sockaddr *) &server, sizeof (struct sockaddr)) == -1)
    {
      perror ("[server]Eroare la bind().\n");
      return errno;
    }

  printf ("[server]Asteptam la portul %d...\n",PORT);

if (listen (sd, 4) == -1)
    {
      perror ("[server]Eroare la listen().\n");
      return errno;
    }
```

Pentru fiecare client conectat serverul creeaza un thread care va fi responsabil de prelaurea, modificarea si trimiterea informatiei inapoi catre client.

```
    pthread_create(&th[count-1], NULL, &treat, jucatori);
```

In int main() vom avea un loop infint in care serverul accepta clienti, daca nu a inceput deja jocul sau nu sunt deja 4 conectati.

```
  while(1)
  {
    int client;
    int length = sizeof (from);
    fflush (stdout);
    if ((client = accept (sd, (struct sockaddr *) &from, &length)) < 0)
   {
     perror ("[server]Eroare la accept().\n");
     continue;
   }
    count++;
    if (count > 4 || start==1)
      {
        count--;
```

```
        send(client,"Nu mai sunt locuri sau deja a inceput un joc!", 45, 0);
        close(client);
        continue;
       }
     else {
      jucatori[count-1].cl=client;
      if(count==1)
                 strcpy(jucatori[count-1].piesa,"blue");
      else if(count==2)
                 strcpy(jucatori[count-1].piesa,"pink");
      else if(count==3)
                 strcpy(jucatori[count-1].piesa,"lime");
      else if(count==4)
                 strcpy(jucatori[count-1].piesa,"gray");
      jucatori[count-1].pozitie = 0;
      jucatori[count-1].idthread = count-1;
      tabla[count-1][0]=1;
      pthread_create(&th[count-1], NULL, &treat, jucatori);
      }
 }//while
```

Functia void treat() este threadul clientului. Ea accepta orice input, returnand clientului un raspuns in functie de inputul oferit. Serverul primeste comanda si asteapta pana se conecteaza cel putin 2 jucatori.

```
while(1)
    if(curent==jucatori[p].idthread)
      break;

  while(1)
  {
   memset(&comanda,0,100);

   while(count<2)
      sleep(1);
    ...
  }
```

Cand sun conectati cel putin 2 jucatori acestia vor trebui sa trimita comanda "ready" cand le vine randul fiecaruia.
Comanda "ready" va seta variabila din struct pe 1 pentru jucatorul respectiv.

```
if(strcmp(comanda,"ready")!=0 && strcmp(comanda,"quit")!=0
&& strcmp(comanda,"dice")!=0 && strcmp(comanda,"pause")!=0
&& strcmp(comanda,"scor")!=0 && strcmp(comanda,"unpause")!=0
&& strcmp("winner",comanda)!=0 && strcmp("help",comanda)!=0
&& strcmp(comanda,"pozitie")!=0)
```

```
   write(jucatori[p].cl,"Comanda invalida!",17);
  else{
       if((strcmp(comanda,"ready")==0 || strcmp(comanda,"ready\n")==0)
            && jucatori[curent].rd==0 && ps==0)
       {
         if(curent==jucatori[p].idthread)
          {
            write(jucatori[curent].cl,"Ai dat ready!",13);
                jucatori[curent].rd = 1;
                curent=(curent+1)%count;
              }
         else write(jucatori[p].cl, "Nu este randul tau sa dai ready!",32);
            }
       ...
        }
```

Comanda "pozitie" ofera pozitia piesei jucatorului pe tabla.

```
else if(strcmp(comanda,"pozitie")==0)
    {
        bzero(&rep,100);
        sprintf(rep,"Esti pe pozitia %d din 40!", jucatori[p].pozitie);
        write(jucatori[p].cl,rep,strlen(rep));
    }
```

Comanda "dice" reprezinta metoda prin care jucatorii muta piesa pe tabla. Pentru a muta prima piesa jucatorul trebuie sa primeasca numarul 6 de la rand(), dupa aceea piesa va fi mutata in functie de numarul returnat.

```
else if(check_ready()==1 && strcmp("dice",comanda)==0 && ps==0)
   {
   if(curent!=jucatori[p].idthread)
   write(jucatori[p].cl, "Nu este randul tau!",19);
   else {
   dice=rand()%6 + 1;
memset(&rep,0,100);
   if(jucatori[curent].pozitie==0)
   {
     if(dice==6)
   {
     tabla[curent][jucatori[curent].pozitie] = 0;
     jucatori[curent].pozitie += dice;
        tabla[curent][jucatori[curent].pozitie] = 1;
        sprintf(rep,"Ai dat cu zarul: %d, ai iesit din zona de start! ",dice);
     write(jucatori[curent].cl,rep,strlen(rep));
      }
      else {
```

```
     sprintf(rep,"Ai dat cu zarul: %d, din pacate nu poti inainta!",dice);
   write(jucatori[curent].cl,rep,strlen(rep));
       }
  }
     else {
     tabla[curent][jucatori[curent].pozitie] = 0;
     jucatori[curent].pozitie += dice;
       tabla[curent][jucatori[curent].pozitie] = 1;

          if(jucatori[curent].pozitie<=40)
       { sprintf(rep,"Ai dat cu zarul: %d, foloseste pozitie pentru a afla pe ce pozitie
        esti!",dice);
    write(jucatori[curent].cl,rep,strlen(rep));
             }
           ...
           }
       ...
       }
   ...
   }
```

Comanda "pause" opreste jocul si pune flagul "ps" pe 1. Numai jucatorul curent poate folosi comanda aceasta si comanda "unpause" pentru a recontinua jocul.

```
else if(strcmp(comanda,"pause")==0)
     {
     if(p==curent)
       {
         if(ps==0)
           {
             ps=1;
          write(jucatori[p].cl,"Ai dat pauza la joc!",20);
           }
         else write(jucatori[p].cl,"Ai dat deja pauza!",18);
       }
      else write(jucatori[p].cl,"Nu este randul tau!",19);
     }
else if(strcmp(comanda,"unpause")==0)
     {
       if(p==curent)
         {
           if(ps==1)
             {
               ps=0;
              write(jucatori[p].cl,"Jocul va continua!",18);
```

```
            }
           else write(jucatori[p].cl,"Nu s-a pus nicio pauza!",23);
         }
       else write(jucatori[p].cl,"Nu este randul tau!",19);
     }
```

Comanda "quit" elibereaza ultima pozitie (ultimul jucator). Cand un jucator da "quit" acesta este scos imedia fara a primi nimic, iar apoi jucatorii de dupa el vor fi mutati pe o pozitie anterioara. Spre exemplu jucatorul 2 vrea sa paraseasca jocul folosind comanda "quit", jucatorul 3 va fi considerat jucatorul 2, iar jucatorul 4 va fi considerat jucatorul 3, lasand locul jucatorului 4 liber.

Comanda "scor" afiseaza scorul fiecarui jucator, scorul fiin calculat in functie de cate piese au ajuns in spatiul de sfarsit.

```
 else if(strcmp(comanda,"scor")==0)
    {
        char msg[200];
        bzero(&msg,200);
        bzero(&rep,100);
        for(int i=0; i<count; i++)
            {
                sprintf(rep,"Jucatorul %s are scorul %d! ",
                    jucatori[i].nume,jucatori[i].wins);
                strcat(msg,rep);
            }
        write(jucatori[p].cl,msg,strlen(msg));
    }
```

Functia check_ready() verifica daca fiecare jucator a dat start sau daca sunt macar 2 jucatori conectati.

```
int check_ready()
{
  for(int i=0;i<count;i++)
   if(jucatori[i].rd==0)
   {
     puncte=4;
     return 0;
   }
  if(count>=2)
   {
     start=1;
     return 1;
   }
  puncte=4;
  start=0;
  return 0; }
```

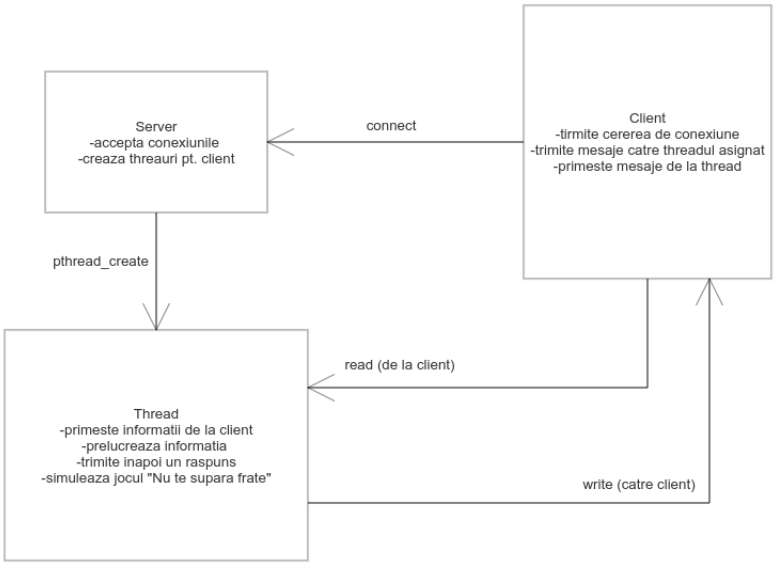
Piesele se muta pe matricea "tabla[4][46]" in care fiecare linie corespunde unui jucator, fiecare jucator avand "vectorul" lui.

```
int count, curent, tabla[4][46];
```

Castigatorul jocului este cel care ajunge cu toate cele 4 piese in spatiul de sfarsit, moment in care resetam tot jocul.

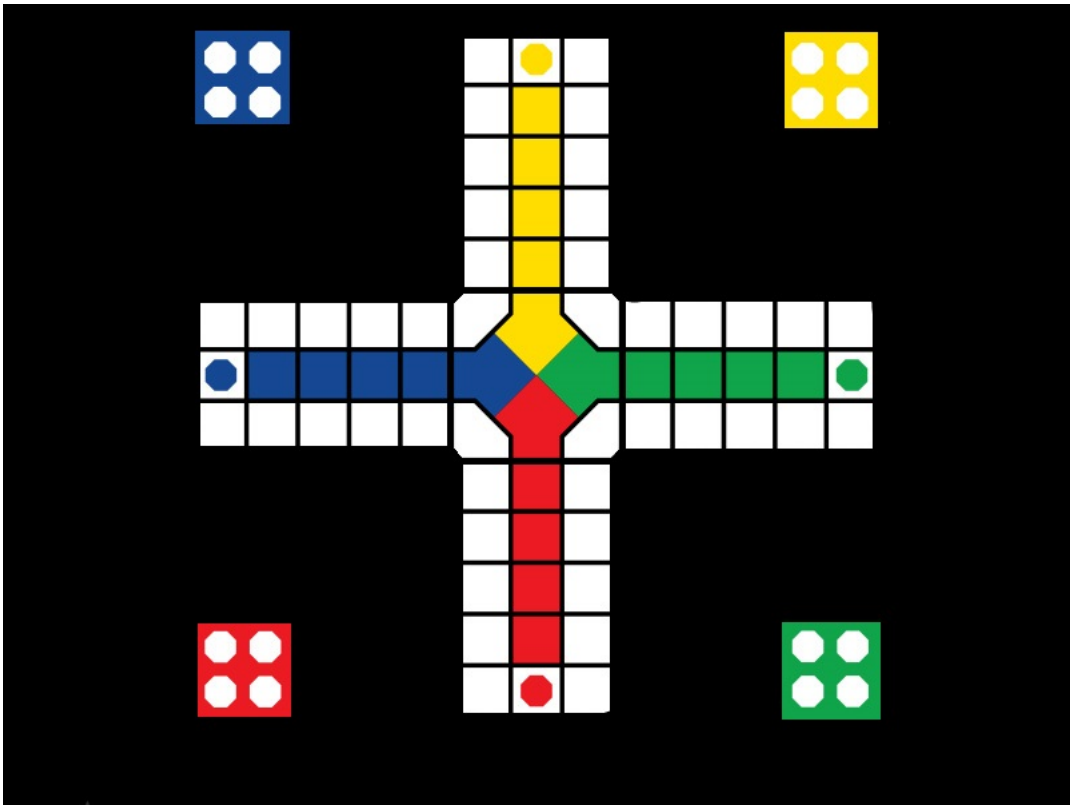
Cand cineva castiga jocul toti jucatorii isi vor reseta flagul rd, iar flagul start va fi pus pe 0 folosind functia check_ready() de la sfarsitul while(1).

```
else if(jucatori[curent].wins==3 && jucatori[curent].pozitie>40)
    {
        jucatori[curent].wins++;
        jucatori[curent].punct=jucatori[curent].wins;
        write(jucatori[curent].cl,"Bravo! Ai Castigat! Jocul s-a terminat!",40);
        strcpy(winner.nume,jucatori[curent].nume);
        for(int i=0; i<count; i++)
            {
                jucatori[i].wins=0;
                jucatori[i].rd=0;
                jucatori[i].punct=0;
                tabla[i][jucatori[i].pozitie]=0;
                jucatori[i].pozitie=0;
            }
    }
```



4 Detalii de implementare

Serverul verifica daca toata lumea a trimis comanda “ready” si porneste jocul.
La inceperea jocului primul jucator va trebui sa transmita comanda “dice” pentru a da cu zarul.
Acesta va primi apoi un numar aleatoriu intre 1 si 6. Daca nu are deja o piesa scoasa din spatiul de start atunci nu poate sa miste nicio piesa pana la valoarea 6 a zarului.
O data ce jucatorul a nimerit valoare 6 i se va adauga prima piesa pe pozitia 6 a ”vectorului” sau.
Fiecare mutare cu zarul va indica numarul de mutari pe care trebuie sa le faca piesa pana la spatiul de sfarsit.
Cand un jucator ajunge cu toate piesele in spatiul de sfarsit acesta castige si se termina jocul.
Cand jocul se termina clientii sunt trimisi intr-un ”meniu” de unde pot alege sa iasa din joc, sau sa dea ”ready” pentru alt joc.



5 Conlcuzii

O imbunatatire directa a solutiei propuse ar putea fi utilizarea obiectelor si claselor. Deasemenea, aplicatia poate fi inbunatatita astfel incat sa suporte mai mult de 4 jucatori, sa suporte echipe de cate 2,3 jucatori pana la un maxim de n echipe si alte modificari asupra regulilor/metodei de joc.
Codul poate suferi mici adjustari pentru eficienta folosind structuri de date mai complexe sau algoritmi mai eficienti.
Deasemenea, codul are cateva probleme nerezolvate care pot fi rezolvate in viitor.

6 Bibliografie

https://www.wikipedia.org/
https://stackoverflow.com/
https://man7.org/
https://www.educlass.ro/data/manuals/e16005c86fc1e28e76adf8f19af0a7ee.pdf
http://www.springer.com/lncs
https://profs.info.uaic.ro/ computernetworks/
https://profs.info.uaic.ro/ eonica/rc/