Sisteme de operare
Laborator 12
Servere iterative TCP/UDP

1. Creati un nou subdirector *lab12/* in structura de directoare a laboratorului creata anterior (*SO/laborator*) si subdirectoarele aferente *doc src* si *bin*. Nu uitati sa actualizati variabila de mediu PATH pentru a include directorul *SO/laborator/lab12/bin*.

2. Scrieti un server UDP **timed-udp.c** pentru serviciul de **time** care sa raspunda la cererile clientului UDP din laboratorul trecut. Pentru receptia datagramelor UDP folositi apelul sistem *recvfrom*, iar pentru transmisia datagramelor UDP folositi apelul sistem *sendto*.

Indicatie: pt a afla timpul, serverul poate folosi apelul sistem *time*.

De ce nu putem folosi apelurile sistem *read*/*write*?

3. Scrieti un server **daytimed-tcp.c** pentru serviciul de **daytime** care sa raspunda la cererile clientului TCP din laboratorul trecut.

4. Scrieti un server **echod-tcp.c** pentru serviciul de **echo** care sa raspunda la cererile clientului TCP din laboratorul trecut.

Daca ar fi sa scrieti un server concurent, in care fiecare cerere client este servita de un alt proces, creat de server folosind apelul *fork*, cum ati proceda?

5. Folositi un shell script pentru a verifica functionalitatea iterativa a serverului **echod-tcp** de mai sus. In acest scop trebuie sa modificati apelul clientului TCP **echo-tcp** din laboratorul anterior a.i. sa nu mai citeasca datele de la tastatura ci dintr-un fisier text aflat in directorul curent (de pilda codul sursa al clientului sau al serverului, **echo-tcp.c** sau **echod-tcp.c**).

Indicatii: Nu trebuie sa modificati codul clientului din laboratorul trecut, e suficient sa redirectati din shell file descriptorul corespunzator pentru a putea cititi datele din fisier.

Pentru a verifica iterativitatea serviciului puteti lansa in executie mai multi clienti **echo-tcp** in paralel, de pilda lansand comenzile din script in background ca mai jos:

```
#!/bin/sh

echo-tcp … &
echo-tcp … &
…..
echo-tcp … &
```

Nu uitati sa dati drepturi de executie shell scriptului (sau alternativ sa apelati interpretorul de comenzi potrivit):

```
$ chmod u+x myscript
$ ./myscript
```

sau

```
$ /bin/sh myscript
```

6. Modificati serverul **echod-tcp.c** de mai sus a.i. sa foloseasca UDP ca protocol de transport. Simulati situatia cand serverul e *unreachable* precum si un protocol simplu de retransmisie pentru a simula pierderea de datagrame UDP in Internet. Folositi programul client din laboratorul anterior pentru a testa programele.

Indicatii: Daca raspunsul serverului nu vine in termen de *S* secunde, puteti declara hostul destinatie (serverul) *unreachable (*imposibil de atins/accesat).

Puteti folosi *alarm* pentru a simula notificarea expirarii unei perioade de *S* secunde pe care clientul a petrecut-o fara a primi un raspuns din partea serverului.

7. Functia *connectsock* pe care ati folosit-o in laboratorul trecut creaza un socket activ. Modificati aceasta functie si scrieti o functie *passivesock* care creeaza un socket pasiv, adica un socket care sa poata fi folosit de un server pentru a trata cereri de conexiuni, cu urmatoarea semnatura:

```
int passivesock(const char *service, const char *transport, int qlen)
/*
* Arguments:
* service
- service associated with the desired port
* transport - transport protocol to use ("tcp" or "udp")
* qlen
- maximum server request queue length (used by listen)
*/
```

In completare, dupa modelul *connectTCP*/*connectUDP* de la laboratorul trecut, scrieti variantele de creeare a unui socket pasiv TCP *passiveTCP*, respectiv UDP *passiveUDP,* folosind functia *passivesock* de mai sus.

Indicatie: inspirati-va din slide-ul nr 4 al prezentarii.