Sisteme de operare
Laborator 11
Clienti TCP/UDP

1. Creati un nou subdirector *lab11/* 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/lab11/bin*.

2. Instalati pe calculator superserverul internet (**inetd** / **xinetd**). Activati serviciile TCP de **daytime** si **echo**, respectiv serviciul UDP **time (***Indicatie*: setati campul *disable* pe *no* in fisierele de configurare ale serviciilor si apoi folositi comanda **service** ca superuser). Cautati in fisierul **/etc/services** porturile pe care functioneaza aceste servicii si incercati sa vedeti cum functioneaza folosind urmatoarea comanda

$ telnet localhost <port>

Ce se intampla?

3. Folositi apelurile sistem *socket* si *connect* pentru a scrie programe client **daytime-tcp.c**, **echo-tcp.c** si respectiv **time-udp.c** care acceseaza serviciile corespunzatoare activate cf exercitiului anterior. La apelul *socket* puteti folosi ca protocol valoarea 0 (al treilea argument de apel). Pentru a putea efectua apelul *connect*, completati campurile structurii urmatoare in mod corespunzator:

```
struct sockaddr_in
{
    short  sin_family; // AF_INET
    unsigned short integer  sin_port; // portul setat cu htons
    struct  in_addr  sin_addr;
    char    sin_zero[8];
};

struct in_addr
{
      unsigned long integer  s_addr; // folositi inet_addr / inet_aton pt a seta adresa
};
```

Programele se vor compila si executa dupa cum urmeaza:

$ gcc -o daytime-tcp daytime_tcp.c

$ daytime-tcp 127.0.0.1 13

Obs: pentru acest laborator vom folosi numai clienti UDP conectati (adica programe care apeleaza *connect* pentru sockets UDP pentru a putea folosi primitivele uzuale de citire/scriere pentru fisiere, *read*/*write*).
Indicatie: timpul in Unix este reprezentat ca numar de secunde incepand de la 1 ian. 1970. Pentru a converti timpul obtinut cu apelul sistem *time* la timpul curent folositi urmatoarea secventa de cod:

```
#define UNIXEPOCH 2208988800
time_t now;
```

```
now = time(NULL);
now -= UNIXEPOCH;
```

Folositi functia de biblioteca *ctime* pentru a afisa o valoare de tip *time_t*.

4. Modificati clientul de **echo-tcp.c** a.i. sa verifice daca raspunsul serverului e intr-adevar copia string-ului trimis de client. Adaugati si facilitatea de a numara octetii trimisi respectiv primiti de la server pentru a completa testul de corectitudine.

5. Transformati clientul **echo-tcp.c** in **echo-udp.c** care sa foloseasca UDP ca protocol de transport. Trimiteti o datagrama UDP catre server si masurati RTT-ul (round trip time).

De cate ori se apeleaza functia *read* spre deosebire de clientul TCP?

6. Creati un fisier **connectsock.c** care contine o functie de biblioteca *connectsock* care conecteaza un client la serviciul unui server ca mai jos:

```
int connectsock(const char *host, const char *service, const char *transport )
```

unde primul argument este numele serverului care trebuie accesat, al doilea argument este port-ul serviciului iar ultimul specifica tipul de serviciu (“udp” sau “tcp”). Functia intoarce un file descriptor pentru comunicarea peste un socket conectat la serverul si portul specificate.

connectsock foloseste *gethostbyname* pentru a converti numele de host intr-o adresa de IP pe care o va stoca in campul *sin_addr* al unei structuri *sockaddr_in* ca mai sus. Similar, foloseste *getservbyname* pentru a cauta un serviciu de internet dupa nume si-l stocheaza in campul *sin_port*. Pentru a identifica protocolul necesar apelului *socket*, *connectsock* foloseste *getprotobyname*.

Odata identificate informatiile de mai sus, *connectsock* foloseste informatia furnizata de aceste functii pentru a apela *socket* si subsecvent *connect*. Descriptorul de fisier al socket-ului conectat este returnat ca rezultat al functiei *connectsock.*

Structurile de date pe care le veti folosi pentru apeluri arata in felul urmator:

```
struct hostent {
        char  *h_name; /* official host name */
        char  **h_aliases; /* other aliases */
        int h_addrtype; /* address type */
        int h_length; /* address length */
        char **h_addr_list;  /* list of addresses */
};
#define h_addr          h_addr_list[0]

struct servent {
        char *s_name; /* official service name */
        char **s_aliases; /* other aliases */
        int sort; /* port for this service */
        char *s_proto; /* protocol to use */
};

struct protoent {
        char *p_name; / * official protocol name * /
```

```
        char **p_aliases; /* list of aliases allowed */
        int p_proto; /* official protocol number */
};
```

Adaugati la **connectsock.c** doua functii wrapper pentru *connectsock* care creeaza sockets connectati la servicii de tip TCP respectiv UDP ca mai jos:

*int connectTCP(const char *host, const char *service);*

*int connectUDP(const char *host, const char *service);*

Rescrieti codul programelor de la punctul 3 pentru a folosi codul acestor functii de biblioteca:

Ex:

$ gcc -o daytime-tcp daytime-tcp.c connectsock.c

Acum ar trebui sa puteti apela programul folosind nume de host-uri si serviciu ca mai jos:

$ daytime-tcp localhost daytime