

Programarea in retea (III)

Lenuta Alboaie
adria@info.uaic.ro

Cuprins

- Primitive I/O - discutii
- Server concurrent UDP
- TCP sau UDP – aspecte
- Instrumente
- Trimiterea si receptarea datelor in regim *out-of-band*

Primitive I/O

- Citire de date
 - read() / recv() / **readv()** / recvfrom() / **recvmsg()**
- Trimitere de date
 - write() / send() / **writv()** / sendto() / **sendmsg()**

Alte primitive | I/O

```
#include <sys/uio.h>
```

```
ssize_t readv (int filedes, const struct iovec *iov, int iovcnt);
```

```
ssize_t writev (int filedes, const struct iovec *iov, int iovcnt);
```

```
    struct iovec
```

```
    {
```

```
        void *iov_base; /* adresa de start a bufferului */
```

```
        size_t iov_len; /* dimensiunea bufferului */
```

```
    };
```

Mai generale decit *read()/write()*, ofera posibilitatea de a transmite date aflate in zone necontigue de memorie

Cele 2 apeluri returneaza la executie normala lungimea transferului in octeti

Alte primitive | I/O

```
#include <sys/socket.h>
```

```
ssize_t recvmsg (int sockfd, struct msghdr *msg, int flags);
```

```
ssize_t sendmsg (int sockfd, struct msghdr *msg, int flags);
```

Ambele functii au majoritatea optiunilor incorporate in structura *msghdr*

Cele mai generale functii I/O; apelurile
read/readv/recv/recvfrom pot fi inlocuite de recvmsg

Cele 2 apeluri returnează la execuție normală lungimea transferului în octeți; -1 in caz de eroare

Alte primitive | I/O

Comparatie intre primitivele I/O:

Function	Orice descriptor	Doar descriptor de socket	Un singur read/write buffer	Scatter/gather read/write	Flag-uri optionale	Adresa nodului <i>peer</i>
read, write	●		●			
readv, writev	●			●		
recv, send		●	●		●	
recvfrom, sendto		●	●		●	●
recvmsg, sendmsg		●		●	●	●

Server UDP | situatii

Majoritatea serverelor UDP sunt iterative

- Server UDP care citește cererea clientului, procesează cererea, trimite răspunsul și termină cu acel client
- Dacă este nevoie de schimb de datagrame multiple cu clientul?

Server UDP concurrent

- dacă elaborarea răspunsului ia mult timp serverul poate crea (*fork()*) un proces copil care va rezolva cererea

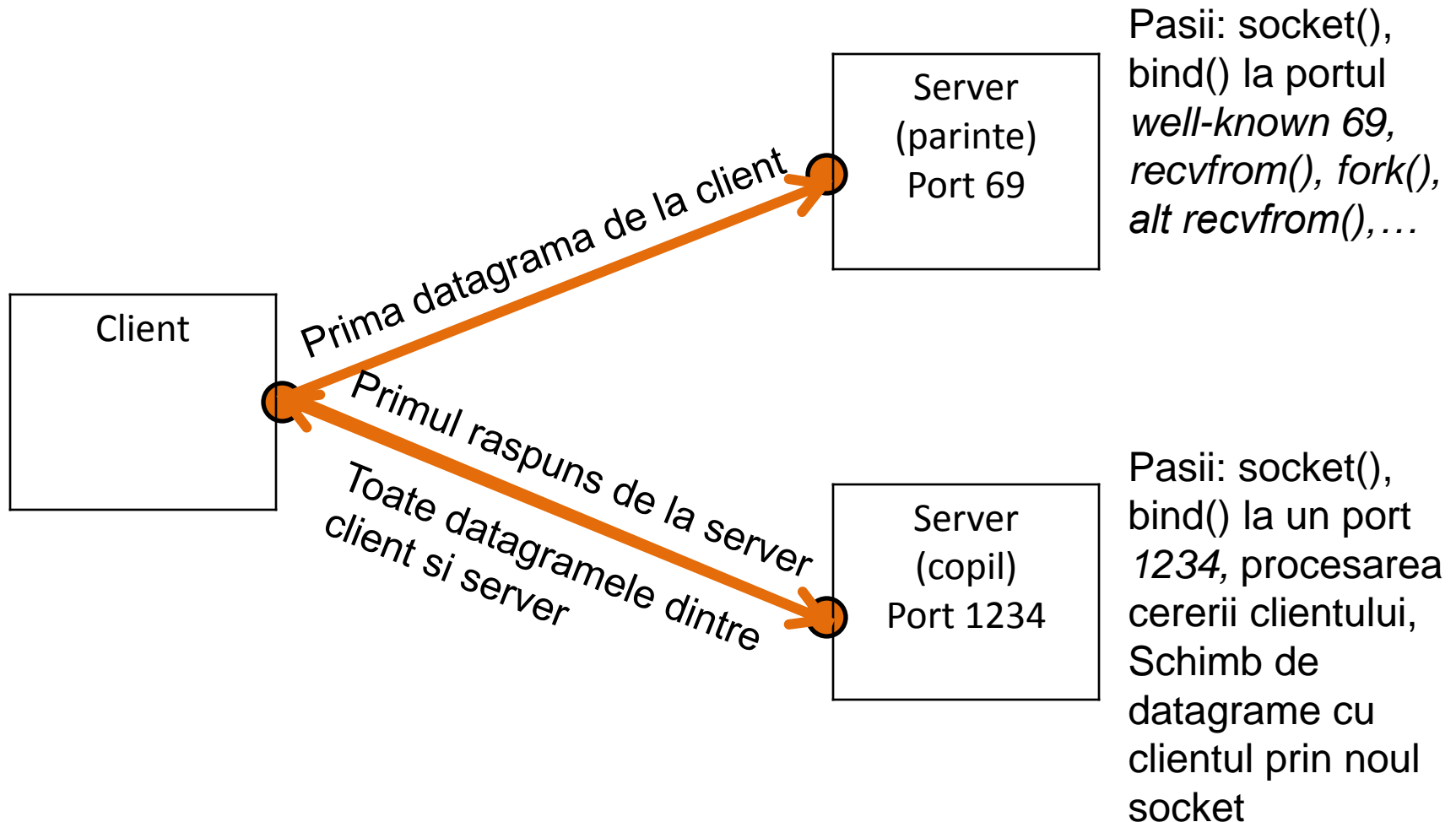
Server UDP | situatii

Server UDP concurent

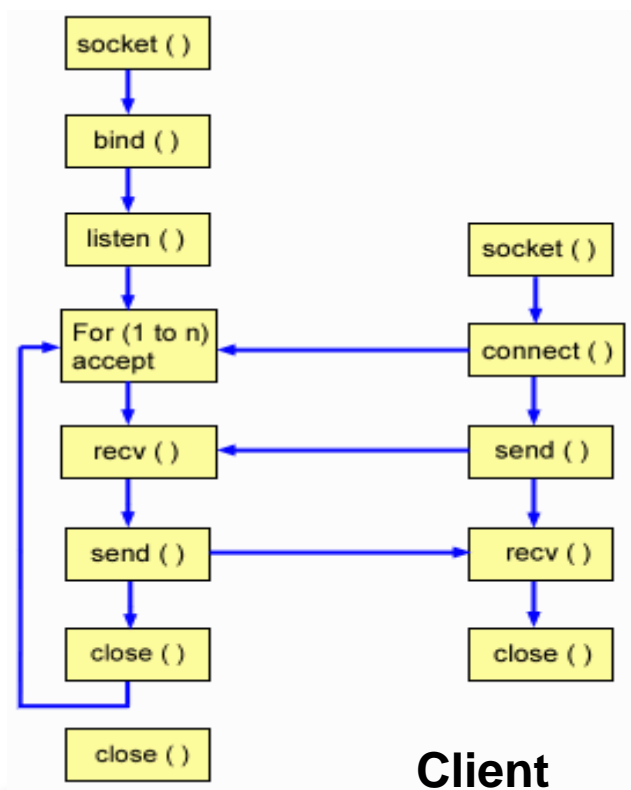
- Server UDP care schimba datagrame multiple cu un client
 - Problema: Doar un numar de port este cunoscut de client ca fiind un port “*wellknown*”
 - Solutia: serverul creaza un socket nou pentru fiecare client, si il ataseaza la un port “*efemer*”, si utilizeaza acest socket pentru toate raspunsurile.
 - Obligatoriul clientului trebuie sa preia din primul raspuns al serverului noul numar de port si sa faca urmatoarele cereri la acel port
 - Exemplu: TFTP - Trivial File Transfer Protocol

Server concurrent UDP

- TFTP utilizeaza UDP si portul 69

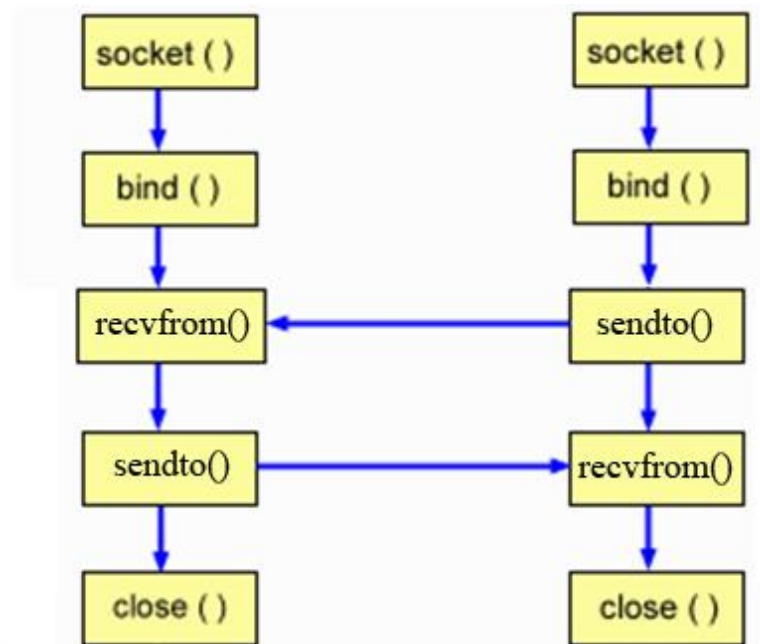


TCP sau UDP - discutii



Server

Client



Server UDP

Client UDP

Model server/client TCP

Model client/server UDP

TCP sau UDP – discutii

Aspecte privind utilizarea UDP:

- UDP suporta broadcasting si multicasting
- UDP nu are nevoie de un mecanism de stabilire a conexiunii
- Minimul de timp necesar unei tranzactii UDP cerere-raspuns este: $RRT(Round Trip Time) + SPT(server processing time)$

Aspecte privind utilizarea TCP:

- TCP suporta point-to-point
- TCP este orientat conexiune
- Oferă siguranță și asigură transmiterea în ordine a datelor;
- Oferă mecanisme de control al fluxului și control al congestiei
- Minimul de timp necesar unei tranzactii TCP cerere-raspuns dacă se creează o nouă conexiune este: $2 * RRT + SPT$

TCP sau UDP – discutii



[<http://www.skullbox.net>]

TCP sau UDP – discutii

Folosirea UDP , respectiv TCP – recomandari

- UDP *trebuie* folosit pentru aplicatii multicast sau broadcast

Controlul erorilor trebuie (eventual) adaugat la nivelul serverului sau clientului

- UDP *poate* fi folosit pentru operatii de cerere-raspuns simple; erorile trebuie tratate la nivelul aplicatiei

Exemple: streaming media, teleconferinte, DNS

TCP sau UDP – discutii

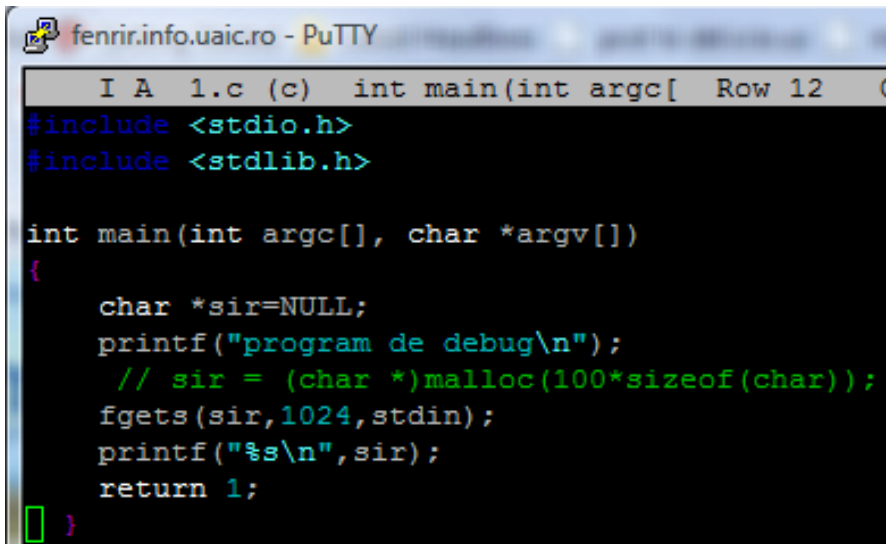
Folosirea UDP , respectiv TCP – recomandari

- TCP *trebuie* folosit pentru *bulk data transfer* (e.g. transfer de fisiere)
 - S-ar putea folosi UDP? → Reinventam TCP la nivelul aplicatiei!

Exemple: HTTP (Web), FTP (File Transfer Protocol),
Telnet, SMTP

Instrumente

- Multe sisteme UNIX ofera facilitatea de “*system call tracing*”

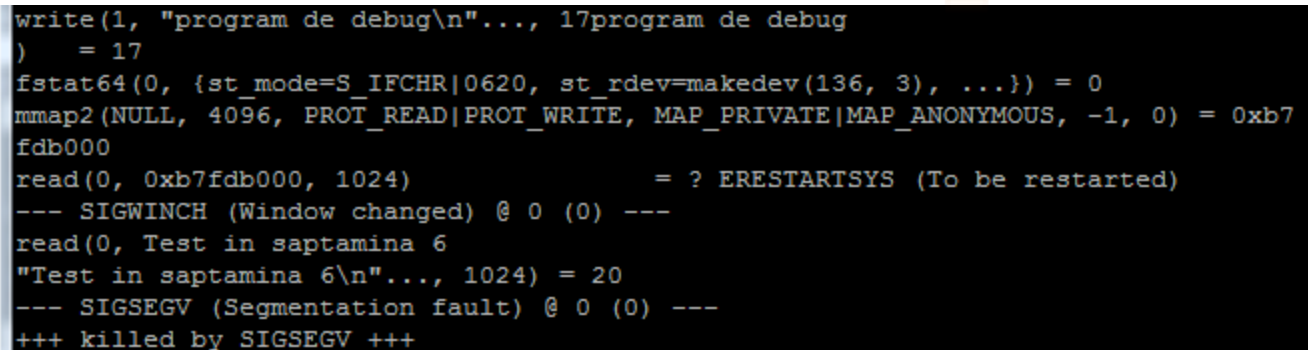


```
I A 1.c (c) int main(int argc[] Row 12 C
#include <stdio.h>
#include <stdlib.h>

int main(int argc[], char *argv[])
{
    char *sir=NULL;
    printf("program de debug\n");
    // sir = (char *)malloc(100*sizeof(char));
    fgets(sir,1024,stdin);
    printf("%s\n",sir);
    return 1;
}
```



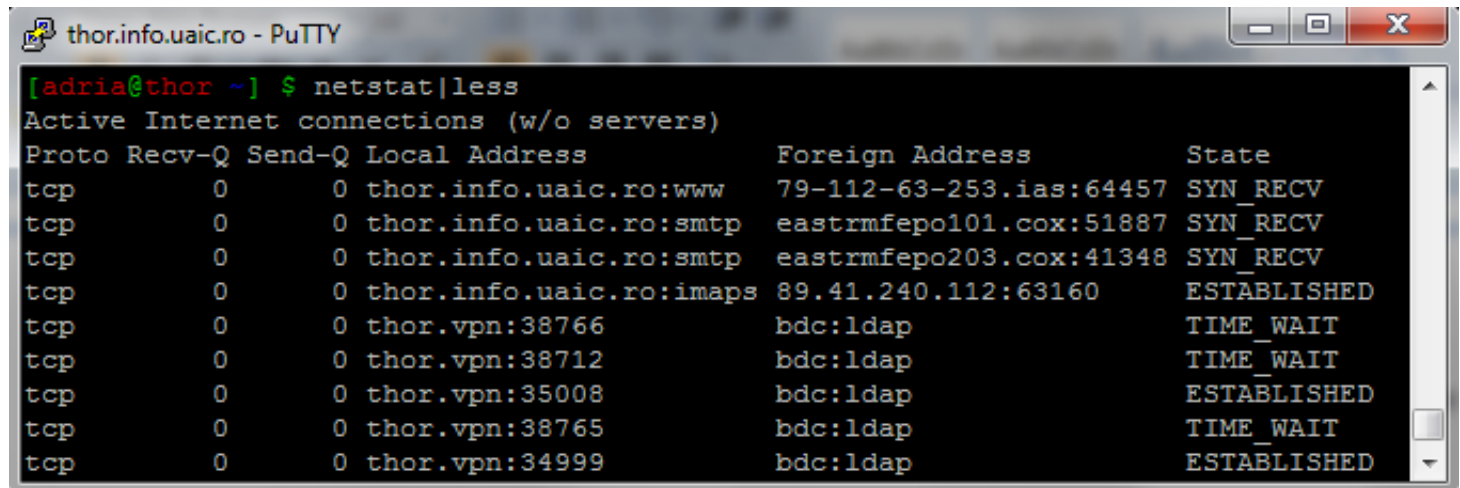
strace



```
write(1, "program de debug\n"... , 17program de debug
) = 17
fstat64(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 3), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7fdb000
read(0, 0xb7fdb000, 1024) = ? ERESTARTSYS (To be restarted)
--- SIGWINCH (Window changed) @ 0 (0) ---
read(0, Test in saptamina 6
"Test in saptamina 6\n"... , 1024) = 20
--- SIGSEGV (Segmentation fault) @ 0 (0) ---
+++ killed by SIGSEGV +++
```

Instrumente

- Programe de test de dimensiuni reduse
- Instrumente:
 - **tcpdump** – majoritatea versiunilor de Unix
 - Oferă informații asupra pachetelor din rețea
 - <http://www.tcpdump.org/>
 - **snoop** – Solaris 2.x
 - **lsof**
 - Identifică ce procese au un socket deschis la o adresă IP sau port specificat
 - **netstat**



```
thor.info.uaic.ro - PuTTY
[adria@thor ~] $ netstat | less
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 thor.info.uaic.ro:www   79-112-63-253.ias:64457 SYN_RECV
tcp        0      0 thor.info.uaic.ro:smtp  eastrmfepo101.cox:51887 SYN_RECV
tcp        0      0 thor.info.uaic.ro:smtp  eastrmfepo203.cox:41348 SYN_RECV
tcp        0      0 thor.info.uaic.ro:imaps 89.41.240.112:63160    ESTABLISHED
tcp        0      0 thor.vpn:38766          bdc:ldap                TIME_WAIT
tcp        0      0 thor.vpn:38712          bdc:ldap                TIME_WAIT
tcp        0      0 thor.vpn:35008          bdc:ldap                ESTABLISHED
tcp        0      0 thor.vpn:38765          bdc:ldap                TIME_WAIT
tcp        0      0 thor.vpn:34999          bdc:ldap                ESTABLISHED
```


Instrumente

- Instrumente:
 - **tcptrack**



Client	Server	State	Idle	A	Speed
172.23.195.11:48328	67.39.222.44:22	ESTABLISHED	0s		38 KB/s
172.23.195.11:48646	196.30.80.10:80	ESTABLISHED	1s		30 KB/s
172.23.195.11:48661	64.37.246.17:80	ESTABLISHED	0s		387 B/s
172.23.195.11:48620	216.239.39.99:80	RESET	2s		0 B/s
128.230.225.95:3531	172.23.195.10:1220	ESTABLISHED	5s		0 B/s
172.23.195.11:48621	216.239.39.99:80	ESTABLISHED	7s		0 B/s
172.23.195.11:48606	64.233.167.99:80	ESTABLISHED	10s		0 B/s
172.23.195.11:48014	67.39.222.44:22	ESTABLISHED	16s		0 B/s
172.23.195.11:47988	67.39.222.44:22	ESTABLISHED	18s		0 B/s
TOTAL					69 KB/s
Connections 1-9 of 9					Unpaused Sorted

Trimiterea si receptarea datelor in regim out-of-band

- Ideea: in timpul unei conexiuni cind sunt transmise date (“**in-band data**”), si daca la un capat se intimpla “ceva” acesta va dori sa transmita rapid celuilalt *peer* in regim de prioritate, o notificare (“**out-of-band data**”)
- Mecanismul de realizare
 - Se utilizeaza bitul URG setat in antetul TCP
 - Antetul TCP contine un cimp indicind locatia datelor urgente ce trebuie trimise
 - **Trimiterea datelor OOB:**
 - Pentru a expedia un octet urgent intr-un flux de date putem utiliza **send()**:
send (sd, buff, 1, **MSG_OOB**);

Trimiterea si receptarea datelor in regim out-of-band

Trimiterea unui octet OOB

Buffer-ul
de trimitere
TCP



Primul byte trimis

Ultimul byte trimis

`send (sd, "a", 1, MSG_OOB)`

Transmiterea unui
byte OOB

Buffer-ul
de trimitere
TCP



Ultimul byte trimis

Pointer de urgenta TCP
(Urgent Pointer)

Trimiterea si receptarea datelor in regim out-of-band

Receptionarea datelor OOB:

- Se genereaza semnalul **SIGURG**
- Apelul **select()** va modifica lista descriptorilor de exceptie

Citirea datelor OOB:

- Daca socketul nu are asociata optiunea **SO_OOBINLINE**, mesajul OOB este plasat intr-un buffer special (*out-of-band buffer*); citirea bufferului se poate realiza cu **recv()** sau **recvmsg()** setind **MSG_OOB**
- Daca socket-ul are asociata optiunea **SO_OOBINLINE**, mesajul OOB este plasat in *buffer*-ul normal de primire;
- Procesul va sti ca a ajuns la *acel octet* in functie de valoarea *out of-band-mark* asociata conexiunii

Trimiterea si receptarea datelor in regim out-of-band

Primitiva `socketmark()`

- Cind se primesc date OOB, se face asocierea cu *out-of-band-mark*, reprezentind pozitia datelor OOB in streamul de date trimis de emitator
- `socketmark()` asigura ca procesul receptor sa determine daca conexiunea are sau nu asociata marca *out-of-band*

```
#include <sys/socket.h>
```

```
int socketmark (int sockfd) ;
```

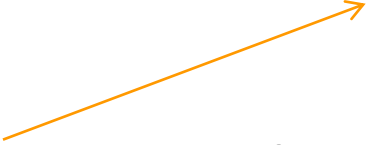
Returneaza: 1 – este *out-of-band mark*,

0 – nu este *out-of-band mark*

–1 on error

Trimiterea si receptarea datelor in regim out-of-band

Primitiva `socketmark()` – observatii

- *out-of-band mark* se aplica indiferent daca procesul receptor primeste datele in mod *out-of-band inline* (*socket*-ul cu optiunea `SO_OOBINLINE`) sau in mod *out-of-band* (flagul `MSG_OOB`)
 - Este implementata folosindu-se `ioctl()` si `SIOCATMARK`
`int value;`
`error = ioctl(tcp_socket, ioctl_type, &value);`
- 

Trimiterea si receptarea datelor in regim out-of-band

Primitiva `socketmark()` – discutii & exemplu

- Daca s-a primit date in mod *OOB inline* `socketmark()` intoarce `true` daca urmatorul octet ce poate fi citit a fost trimis cu flagul `MSG_OOB`
- Daca *socket*-ul nu are asociata optiunea `SO_OOBINLINE`, `socketmark()` intoarce *true* daca urmatorul octet ce poate fi citit este primul octet care a fost trimis dupa data OOB
- Operatia de citire se opreste in functie de *out of-band-mark*
 - Exemplu:

Daca sunt 100 de octeti in buffer dar doar 5 octeti pina la *out of-band-mark*, chiar daca procesul citeste 100 de octeti el va primi initial doar primii 5 octeti

Trimiterea si receptarea datelor in regim out-of-band

Erori posibile

- Se asteapta citirea de date OOB, dar ele nu au fost inca trimise – se returneaza **EINVAL**
- Procesul a fost notificat ca va primi date OOB (via **select()** sau **SIGURG**), el incearca sa le citeasca dar ele inca nu au ajuns – se returneaza **EWOULDBLOCK**
- Se incearca sa se citeasca un acelasi octet **OOB** de mai multe ori – se returneaza **EINVAL**
- Daca procesul are setat optiunea **SO_OOBINLINE**, dar incearca sa citeasca cu flagul **MSG_OOB** – se returneaza **EINVAL**

Trimiterea si receptarea datelor in regim out-of-band

- Utilizari:
 - Modalitate de comunicare a celuiilalt punct terminal a unei conditii de exceptie chiar si in cazul cind controlul fluxului a oprit emitatorul
 - Pentru a detecta timpuriu erori de comunicare intre client si server (*heart-beat*)

Trimiterea si receptarea datelor in regim out-of-band

Tratare **OOB** prin **SIGURG**:

```
if (listen (sd, 5) == -1)  { ... }
...
while (1)
{
    ... client = accept (sd, (struct sockaddr *) &from, &len);
    signal(SIGURG, urgHandler);
    fcntl(client, F_SETOWN, getpid()); /*setarea proprietarului socketului conectat */
    for(;;)  {
        if((n=read(client,msg,sizeof(msg)-1))==0)
        {
            printf("Am primit EOF\n");
            break;
        }
        else
        {
            msg[n]=0;
            printf("Am citit %d octeti: %s\n",n, msg);
        }
    } ... } //while
void urgHandler(int signnr)
{
    int n;          char buff[100];
    printf("SIGURG e primit\n");
    n=recv(client,buff, sizeof(buff)-1, MSG_OOB);
    buff[n]='\0';
    printf("Am citit %d octet OOB %s\n",n, buff);
}
```

Trimiterea si receptarea datelor in regim out-of-band

Tratare **OOB** folosind primitiva **sokatmark()**

DEMO

- **Emitatorul** trimite 4 octeti de date normale, apoi un octet OOB si inca un octet de date normale
- **Receptorul** nu foloseste SIGURG sau select(); el apeleaza primitiva **sokatmark()**

Rezumat

- Primitive I/O - discutii
- Server concurrent UDP
- TCP sau UDP – aspecte
- Instrumente
- Trimiterea si receptarea datelor in regim out-of-band

Bibliografie

- UNIX Network Programming: The sockets networking API, W. Richard Stevens, Bill Fenner, Andrew M. Rudoff
- The Illustrated Network: How TCP/IP Works in a Modern Network (The Morgan Kaufmann Series in Networking), Walter Goralski



Intrebari?

Intrebari?