Socket stream — citire și scriere

`ssize_t write(int` *sd*`, const void*` *buf*`, size_t` *count*)

Efect: trimite *count* octeți începând de la adresa *buf*.

Exemple:

```
int i;
write(sd, &i, sizeof(i));

char buf[1000];
// fara terminator nul:
write(sd, buf, strlen(buf));
// incluzand terminatorul nul
write(sd, buf, strlen(buf)+1);
```

Socket stream — citire și scriere (continuare)

`ssize_t read(int` *sd*`, void*` *buf*`, size_t` *count*`)`

Efect: recepționează cel mult *count* octeți începând de la adresa *buf*. Returnează:

- -1 : eroare
- 0 : s-a închis conexiunea și restul datelor au fost citite
- 1–*count* : octeți citiți; **Atenție:** e posibil să citească mai puțin de *count* octeți.

Notă: rețeaua nu păstrează limitele între mesaje.

Închiderea conexiunii

`int shutdown(int` *sd*`, int` *how*`)`

how=

- `SHUT_RD`: oprește citirea
- `SHUT_WR`: oprește scrierea
- `SHUT_RDWR`: oprește tot

`int close(int` *sd*`)`

Socket pentru datagrame

creare: `SOCK_DGRAM`

Operaţii:

`ssize_t sendto(int` *sd*`, const void *`*buf*`, size_t` *len*`,`
`int` *flags*`, const struct sockaddr *`*to*`, socklen_t` *tolen*`)`

`ssize_t recvfrom(int` *sd*`, void *`*buf*`, size_t` *len*`,`
`int` *flags*`, struct sockaddr *`*from*`, socklen_t*` *fromlen*`)`

Observaţii:

- adresa se fixează prin `bind()` sau este atribuită la primul `sendto()`
- un server va trimite răspunsul la adresa de la care a sosit cererea (adresa returnată de `recvfrom()`
- o datagrama prea mare nu se transmite cu succes

Modelul transmisiei

Memoria:

adresa	biți	hexa	zecimal
123	01010111	57	87
124	10111010	BA	186
125	00001010	0A	10
126	11111110	FE	254
127	01010101	55	85

`char` = întreg pe un octet (8 biți)

Dacă `buf` este de tip `char*`, atunci `buf[0]`, `buf[1]`, ..., sunt octeți consecutivi din memorie.

Exemplu: `buf[0]`=87, `buf[1]`=186, etc.

`send()` + `recv()` ≈ `memcpy()`

Reprezentarea binară a întregilor

Exemplu: numărul 300:

Reprezentare „pe hârtie“: 100101100_2

Reprezentare pe 16 biţi (lungime fixă): 0000-0001-0010-1100

Reprezentarea în memorie (`short k=300`):

	big endian	little endian
adr.	biţi	biţi
124	00000001	00101100
125	00101100	00000001

Reprezentarea binară a întregilor (continuare)

Transmiterea întregilor, în programe portabile:

Întregi de lungime fixă: `uint16_t`, `uint32_t`.

Funcţii de conversie format local $\leftrightarrow$ reţea: `htons()`, `htonl()`, `ntohs()`, `ntohl()`.

Exemplu:

```
uint16_t k;
k=htons(expr);
write(sd, &k, 2);
```

Reprezentarea structurilor de date

Problemă: pe unele platforme se introduc octeţi neutilizaţi între variabile, în vederea alinierii la multiplu de x octeţi.

Exemplu:

`struct {uint16_t a; uint32_t b} s;`

a_0	a_1	b_0	b_1	b_2	b_3

a_0	a_1	0	0	b_0	b_1	b_2	b_3

Neportabil: `write(sd, &x, sizeof(x))`

Portabil: `write(sd, &x.a, 2);write(sd, &x.b, 4);`

Reprezentarea structurilor de date (continuare)

sau:

```
char buf[6];
memcpy(buf, &x.a, 2);
memcpy(buf+2, &x.b, 4);
write(sd, buf, 6);
```

Reprezentarea șirurilor de caractere

Reprezentarea lungimii:

- ca întreg, în fața șirului:

04	41	42	43	44	03	58	59	5A

- prin adāugarea unui terminator:

41	42	43	44	00	58	59	5A	00

Reprezentarea text vs. binar

Text: separatori/terminatori caractere albe (spațiu, tab, sfârșit de linie); numerele se reprezintă ca șiruri de cifre:

Exemplu: 300 ABC

33	30	30	20	41	42	43	0A

Binar: numerele se reprezintă binar; șirurile se dau de obicei ca lungime urmată de șir

01	2C	00	03	41	42	43