Dr. Varga Imre
Debreceni Egyetem, Informatikai Kar

Socket-programozás

C nyelven, Linux alatt, IPv4 alapon

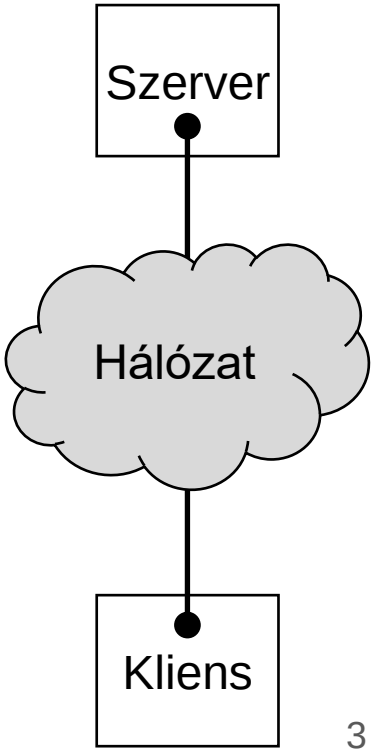
2022.03.31

Főbb pontok

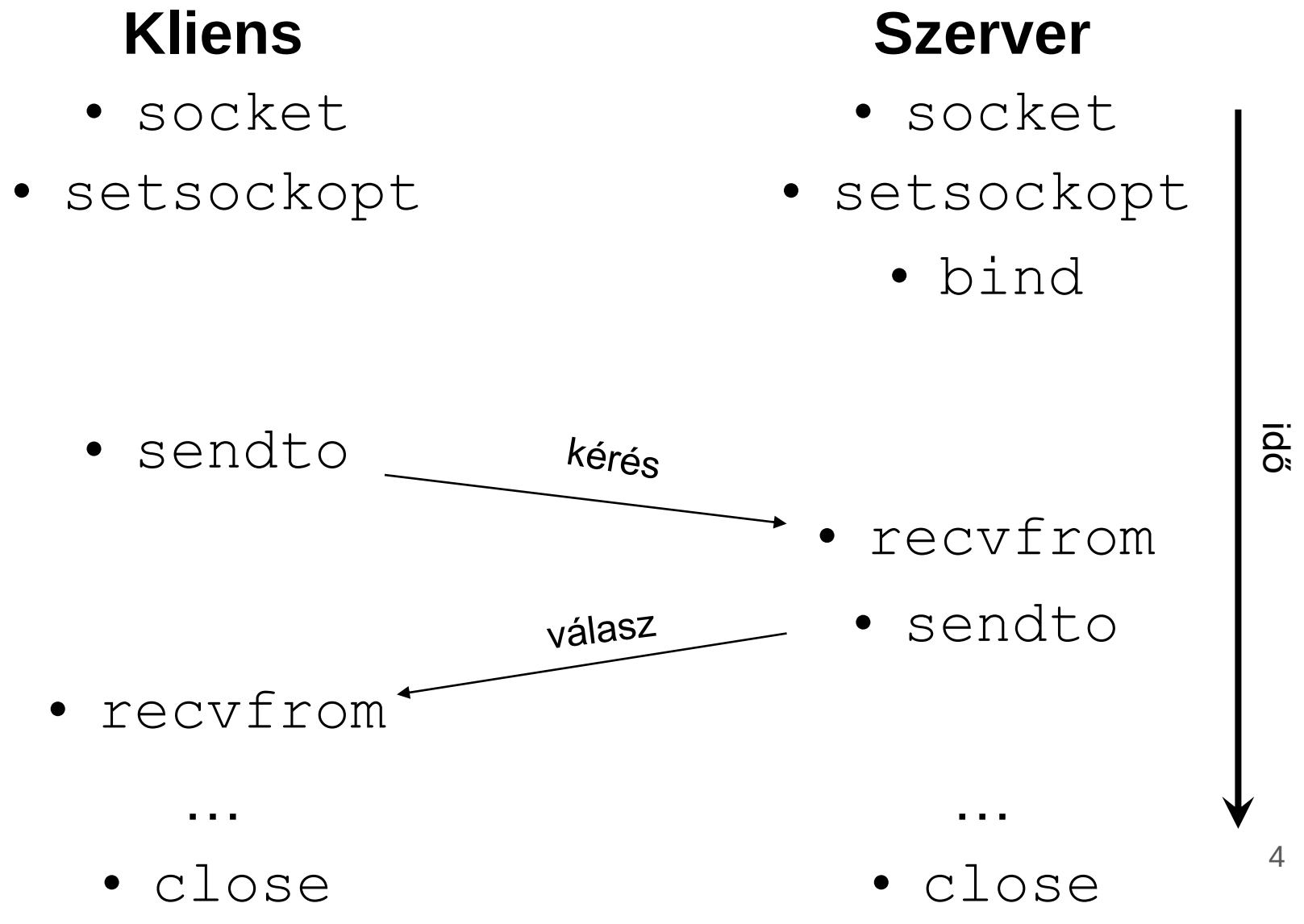
- A kommunikáció alapjai
- Adatstruktúrák és típusok
- Konvertáló függvények
- Rendszerhívások
- Információs függvények

Kliens & Szerver

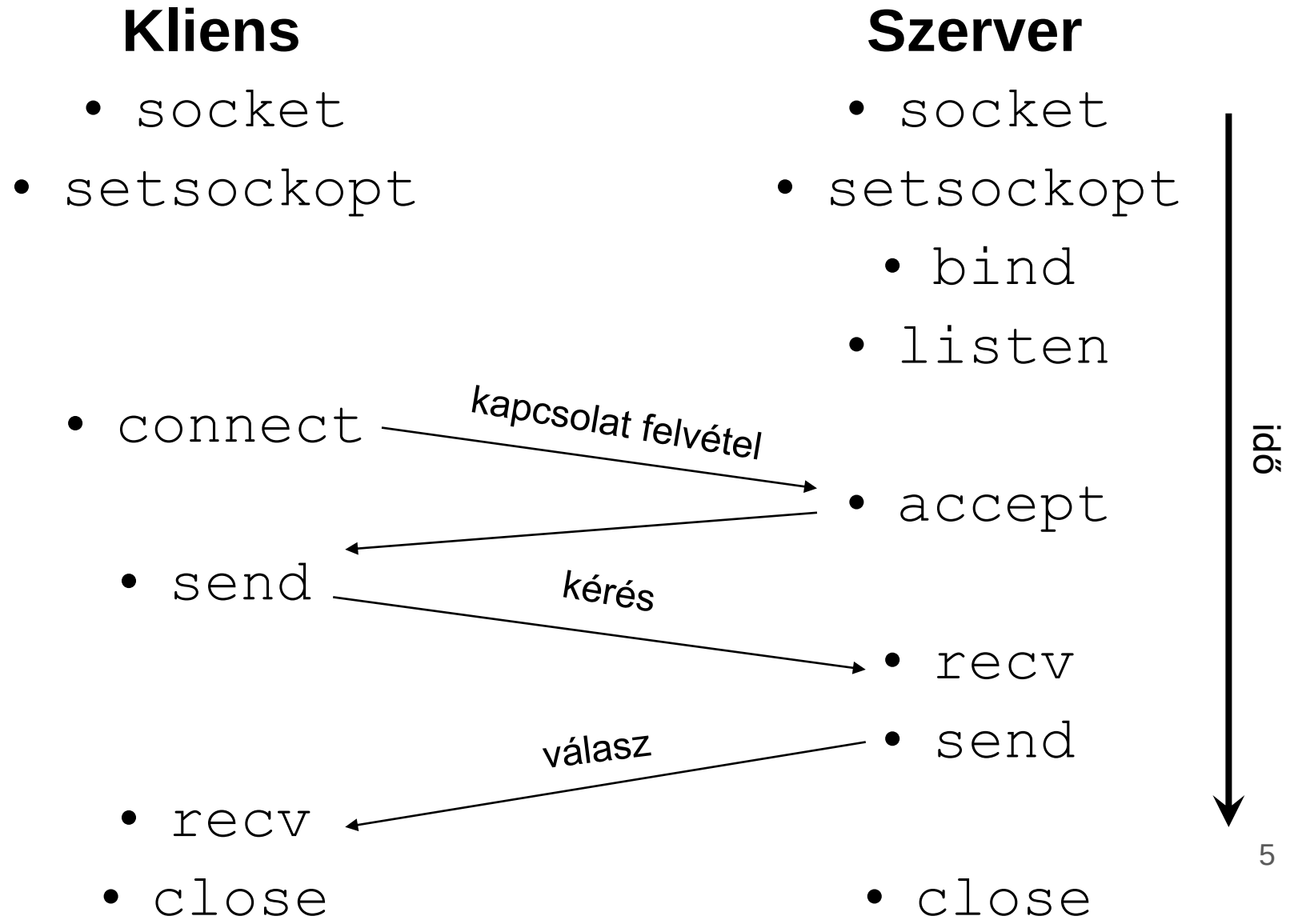
- Szerver szolgáltatást nyújt.
- Kliens igénybe veszi a szolgáltatást.
- Folyamatok közti kommunikáció:
  - Kapcsolat-orientált (TCP)
  - Kapcsolat nélküli (UDP)
- Csatlakozó (socket) típusok:
  - SOCK_STREAM (TCP)
  - SOCK_DGRAM (UDP)
- Alkalmazási réteg (API)



Kapcsolat nélküli időd iagram

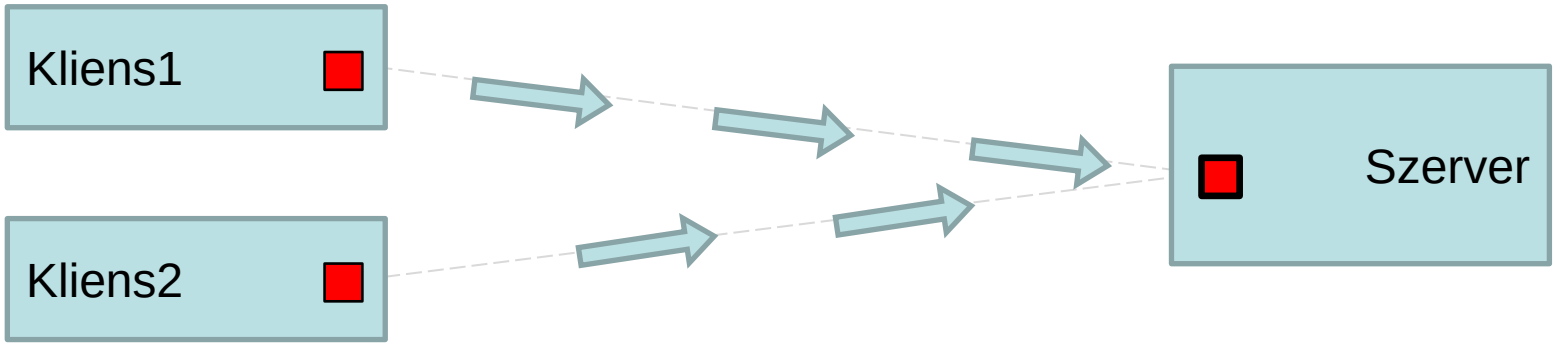


Kapcsolat-orientált időd iagram

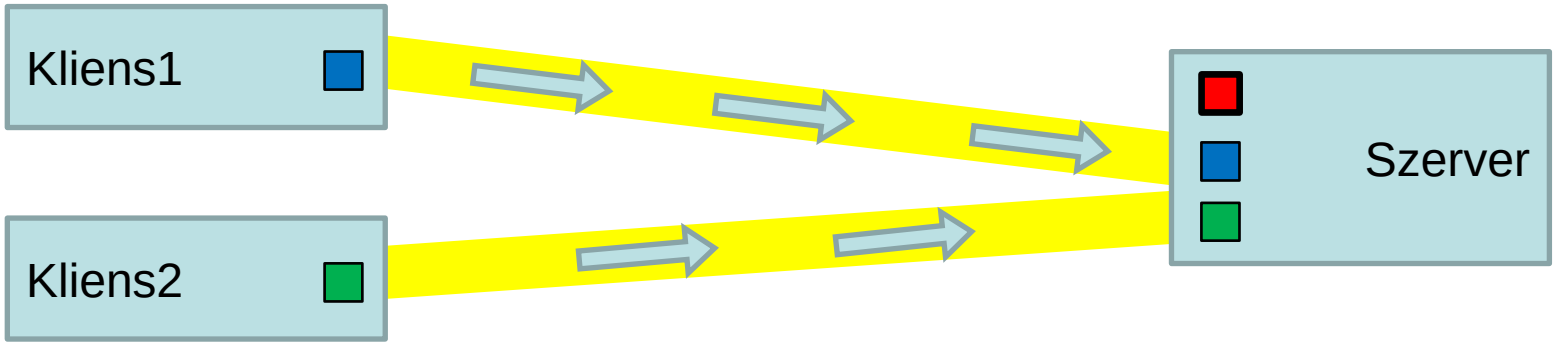


A kommunikáció sémája

Kapcsolat nélküli



Kapcsolat-orientált



Header állományok

- `sys/socket.h`
- `netinet/in.h`
- `arpa/inet.h`
- `netdb.h`
- `unistd.h`

Egyes header állományok inkludálhatnak másokat.

Adatstruktúrák és típusok

- `in_addr (netinet/in.h)`
- `sockaddr_in (netinet/in.h)`
- `sockaddr (sys/socket.h)`
- `hostent (netdb.h)`
- `netent (netdb.h)`
- `protoent (netdb.h)`
- `servent (netdb.h)`

in_addr

```
struct in_addr {
   uint32_t s_addr;
   };
```

- IPv4 címek tárolása 32 biten előjel nélküli egészként big-endian (hálózati) bájtsorrendben.
- `INADDR_ANY`: lokális címekre hivatkozás
- Példa a 127.0.0.1 cím tárolására:

```
struct in_addr IP;
IP.s_addr=16777343u; // 0x0100007f
```

sockaddr_in

```
struct sockaddr_in {
    short int  sin_family;
    unsigned short int sin_port;
    struct in_addr  sin_addr;
    unsigned char  sin_zero[8];
    };
```

- sin_family: cím család, pl. `AF_INET`.
- sin_port: port szám (2 byte) hálózati byte sorrendben.
- sin_addr: IP cím (4 byte) hálózati byte sorrendben.
- sin_zero: kitöltő, hogy `sockaddr` méretű legyen.

sockaddr

```
struct sockaddr {
   unsigned short  sa_family;
   char sa_data[14];
  };
```

- sa_family: cím család, pl. `AF_INET`.
- sa_data: protokoll cím.
- Általános címleíró a rendszerhívások esetén.
- kompatibilitás: `sockaddr_in`

hostent

```
struct hostent {
  char *h_name;        //hivatalos nev
  char **h_aliases;    //tovabbi nevek
  int h_addrtype;      //cim csalad
  int h_length;        //cím hossz
  char **h_addr_list;//in_addr cim lista
};

#define h_addr h_addr_list[0]
```

- Host leíró információk.
- A `h_aliases` és `h_addr_list` tömb utolsó eleme `NULL`.

Konvertáló függvények

- `inet_addr(…)`
- `inet_aton(…)`
- `inet_ntoa(…)`
- `inet_pton(…)`
- `inet_ntop(…)`
  - `htonl(…)`
  - `htons(…)`
  - `ntohl(…)`
  - `ntohs(…)`

IP cím kezelés

```
#include<sys/socket.h>
#include<arpa/inet.h>
struct sockaddr_in address;
```

- inet_addr(): char* → uint32_t

```
address.sin_addr.s_addr=inet_addr("127.0.0.1");
```

- inet_aton(): char* → in_addr

```
inet_aton("127.0.0.1",&(address.sin_addr));
```

- inet_ntoa(): struct in_addr → char*

```
printf("IP: %s\n",inet_ntoa(address.sin_addr));
```

Byte sorrend konverzió

```
#include <arpa/inet.h>
```

- gazdagép → hálózati

```
uint16_t htons(uint16_t hostshort)
uint32_t htonl(uint32_t hostlong)
```

- hálózati → gazdagép

```
uint16_t ntohs(uint16_t netshort)
uint32_t ntohl(uint32_t netlong)
```

Socket rendszerhívások

- `socket(…)`
- `setsockopt(…)`
- `bind(…)`
- `listen(…)`
- `connect(…)`
- `accept(…)`
- `close(…)`
- `shutdown(…)`
- `select(…)`
- `send(…)`
- `sendto(…)`
- `sendmsg(…)`
- `write(…)`
- `recv(…)`
- `recvfrom(…)`
- `recvmsg(…)`
- `read(…)`

socket

```
int socket(int family, int type,
  int protokcol);
```

- Socket létrehozása.
- Visszatérési érték: OK: file leíró; hiba: `-1`
- family: `AF_INET`
- type: `SOCK_STREAM`, `SOCK_DGRAM`
- protocol: `0` (default a type és a family alapján)
- `#include<sys/socket.h>`

setsockopt

```
int setsockopt(int fd, int level,
  int cmd, char *arg, int len);
```

- Opciók beállítása.
- fd: file leíró, amit a `socket` ad.
- level: `SOL_SOCKET`
- cmd: `SO_REUSEADDR`, `SO_KEEPALIVE`
- arg: mutató a kívánt opciót tartalmazó bufferre, ahol a tárolt érték: `1`.
- len: arg mérete.
- `#include<sys/socket.h>`

bind

```
int bind(int fd,
  struct sockaddr *addrp, int alen);
```

- Socket hozzárendelése hálózati címhez szerver oldalon.
- Visszatérési érték: OK: `0`; hiba: `-1`
- fd: file leíró, amit a `socket` ad.
- addrp: a saját hálózati címet leíró struktúra címe.
- alen: a címleíró struktúra mérete
- `#include<sys/socket.h>`

listen

```
int listen(int fd, int backlog);
```

- Kapcsolatelfogadási szándék és queue méret beállítás szerveren.
- Visszatérési érték: OK: `0`; hiba: `-1`.
- fd: file leíró, amit a `socket` ad.
- backlog: hány feldolgozatlan `connect` kérést tárol.
- `#include<sys/socket.h>`

connect

```
int connect(int fd,
  struct sockaddr *addrp, int alen);
```

- Kapcsolati kérelem küldés (kliens oldalon).
- Visszatérési érték: OK: `0`; hiba: `-1`.
- fd: file leíró, amit a `socket` ad.
- addrp: cél (server) cím.
- alen: a címleíró struktúra mérete.
- `#include<sys/socket.h>`

accept

```
int accept(int fd,
  struct sockaddr *addrp, int *alenp);
```

- Kapcsolat elfogadása szerveren (lásd `tcpd`).
- Visszatérési érték: hiba: `-1`;
  OK: új file leíró fd tulajdonságaival.
- fd: file leíró, amit a `socket` ad.
- addrp: kliens címe ide kerül.
- alenp: híváskor addrp hossza, visszatéréskor kapott cím hossza.
- `#include<sys/socket.h>`

send

```
int send(int fd, char *buff, int len,
  int flags);
```

- Kapcsolat-orientált adat küldés.
- Visszatérési érték: OK: átvitt byte szám; hiba: `-1`.
- fd: file leíró, amit a `socket` ad.
- buff: az üzenet (bájtsorozat).
- len: az üzenet hossza.
- flags: `0`; `MSG_OOB`: nagy prioritás.
- `#include<sys/socket.h>`

sendto

```
int sendto(int fd, char *buff,
  int len, int flags,
  struct sockaddr *addrp, int alen);
```

- Kapcsolat nélküli adat küldés.
- Visszatérési érték: OK: átvitt byte szám; hiba: `-1`.
- fd, buff, len, flags: mint a `send` esetén.
- addrp, alen: mint `connect` esetén.
- `#include<sys/socket.h>`

recv

```
int recv(int fd, char  *buff,
  int maxlen, int flags);
```

- Kapcsolat-orientált adat fogadás.
- Visszatérési érték: OK: kapott byte szám; hiba: `-1`.
- fd: file leíró, amit a `socket` ad.
- buff: az üzenet (bájtsorozat).
- maxlen: a buffer hossza.
- flags: pl. `0`; `MSG_OOB` csak az így küldött adatot veszi.
- `#include<sys/socket.h>`

recvfrom

```
int recvfrom(int fd, char  *buff,
  int maxlen, int flags,
  struct sockaddr *addrp, int *alenp);
```

- Kapcsolat nélküli adat fogadás.
- Visszatérési érték: OK: kapott byte szám; hiba: `-1`.
- fd, buff, maxlen, flags: mint `recv` esetén.
- addrp, alenp: mint `accept` esetén.
- `#include<sys/socket.h>`

write, read

```
int write(int fd, char *buff, int len);
int read(int fd, char *buff, int mlen);
```

- Kapcsolat-orientált esetben használható küldésre, fogadásra.
- Visszatérési érték: OK: byte szám; hiba: `-1`.
- fd: file leíró, amit a `socket` ad.
- buff: üzenet (bájtsorozat).
- mlen, len: (max) üzenet hossz.
- `#include<unistd.h>`

close

```
int close(int fd);
```

- Lezárja a socket-et.
- Visszetérési érték: OK: `0`; hiba: `-1`.
- fd: file leíró, amit a `socket` ad.
- `#include<unistd.h>`

shutdown

```
int shutdown(int fd, int how);
```

- Kapcsolat-orientált socket egyirányú lezárása.
- Visszatérési érték: OK: `0`; hiba: `-1`.
- fd: file leíró, amit a `socket` ad.
- how:
  `0`: nem lehet adatot átvenni tőle;
  `1` nem lehet adatot átadni neki;
  `2`: egyik sem (`close`).
- `#include<sys/socket.h>`

Információs függvények

- `gethostname(…)`
- `gethostbyname(…)`
- `gethostbyaddr(…)`
- `getpeername(…)`
- `getservbyname(…)`
- `getservbyport(…)`
- `getsockname(…)`

gethostname

```
int gethostname(char *hname,
  size_t len);
```

- Helyi gép neve.
- Visszatérési érték: hiba esetén `-1`.
- hname: ide kerül a helyi gép neve.
- len: név hossz.
- `#include<unistd.h>`

gethostbyname

```
struct hostent *gethostbyname(
  char *hname);
```

- Távoli fél azonosítás név alapján.
- Visszatérési érték: hiba esetén `NULL`.
- hname: a távoli gép neve.
- `#include<netdb.h>`

gethostbyaddr

```
struct hostent *gethostbyaddr(
  char *addrp, int len, int family);
```

- Távoli fél azonosítás cím alapján.
- Visszatérési érték: hiba esetén `NULL`.
- addrp: keresett cím (`in_addr`).
- len: cím hossz.
- family: cím család, pl. `AF_INET`.
- `#include<netdb.h>`

További hasznos irodalmak

- Socket mintaprogramok
  https://irh.inf.unideb.hu/~vargai/download/sysprog/Socket.zip
- Michael J. Donahoo, Kenneth L. Calvert:
  TCP/IP Sockets in C (Elsevier, 2009)
- Linux manual pages
  https://man7.org/linux/man-pages
- Socket Programming Tutorial In C For Beginners
  https://www.youtube.com/watch?v=LtXEMwSG5-8
- Socket Library Functions
  https://userpages.uni-koblenz.de/~ros/Rechnerorganisation/socketlibfun.pdf
- University of Crete: Introduction to Sockets Programming in C using TCP/IP
  https://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf
- Berkeley sockets

  https://en.wikipedia.org/wiki/Berkeley_sockets