

Dr. Varga Imre

Socket-programozás

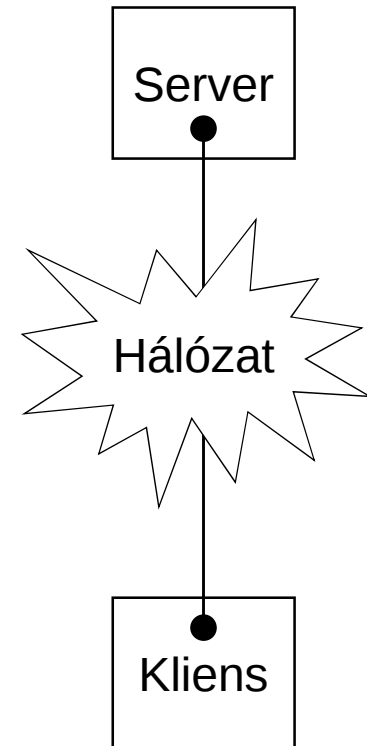
C nyelven

Főbb pontok

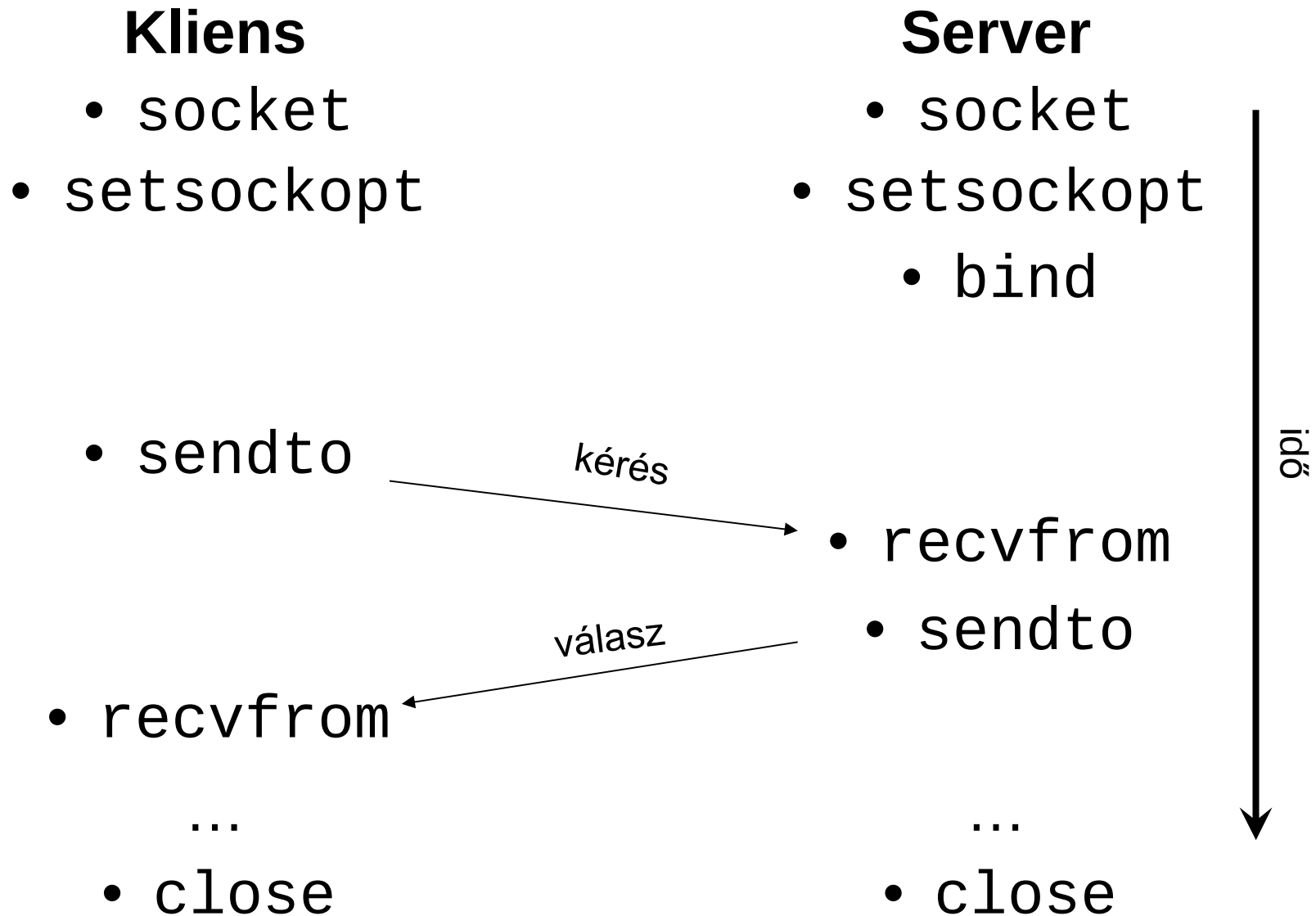
- A kommunikáció alapjai
- Adatstruktúrák és típusok
- Konvertáló függvények
- Rendszerhívások
- Információs függvények

Kliens & Server

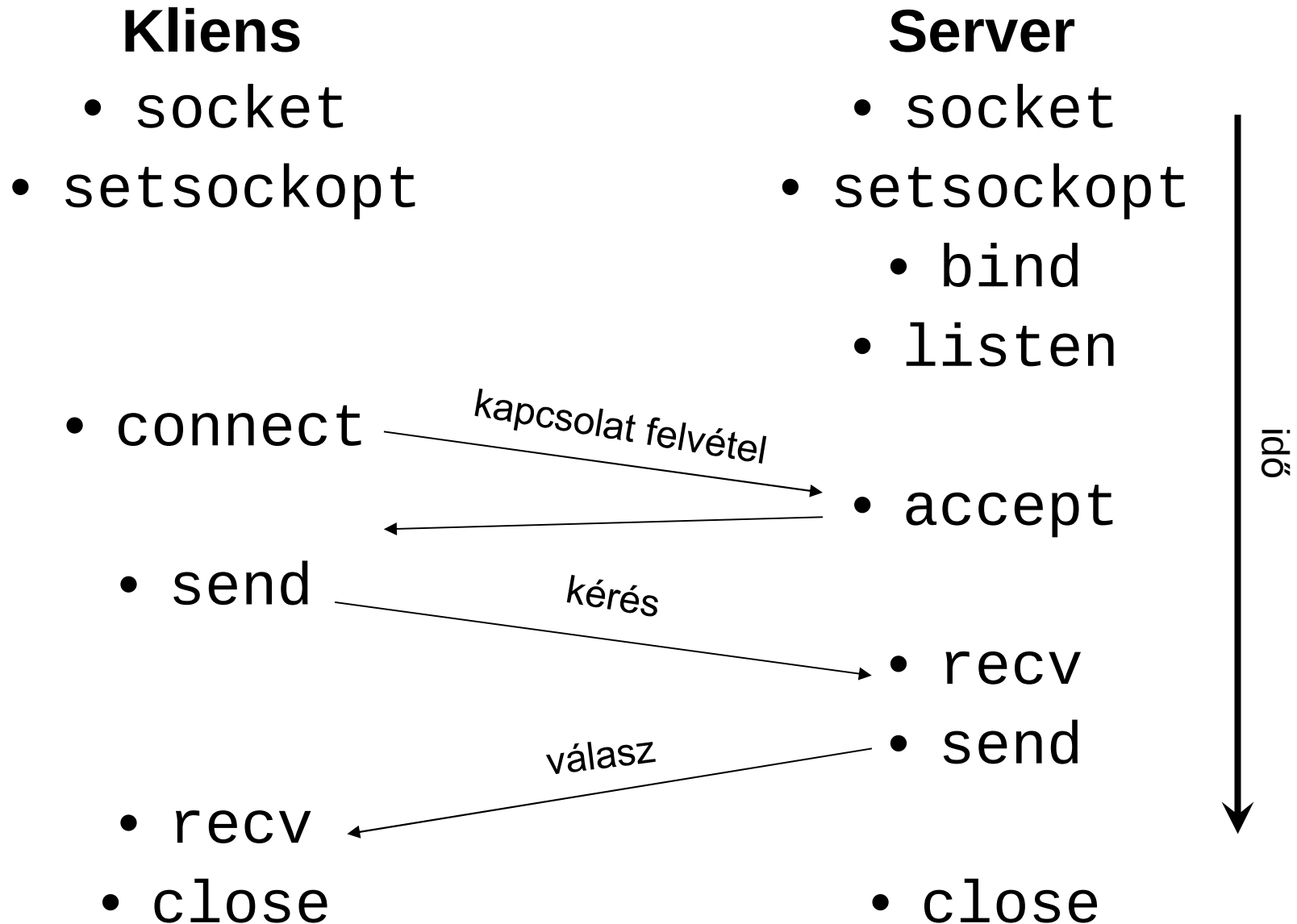
- Server szolgáltatást nyújt.
- Kliens igénybe veszi a szolgáltatást.
- Kommunikáció:
 - Kapcsolat-orientált
 - Kapcsolat nélküli
- Csatlakozó (socket) típusok:
 - SOCK_STREAM (TCP)
 - SOCK_DGRAM (UDP)



Kapcsolat nélküli idődiagram



Kapcsolat-orientált idődiagram



Adatstruktúrák és típusok

- sockaddr
- sockaddr_in
 - in_addr
 - hostent
 - netent
 - protent
 - servent

sockaddr

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

- sa_family: cím család, pl. AF_INET.
- sa_data: protokoll cím.

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

in_addr

```
struct in_addr {  
    union {  
        struct {u_char s_b1, s_b2, s_b3, s_b4;}  
            S_un_b;  
        struct { u_short s_w1, s_w2; }  
            S_un_w;  
        u_long S_addr;}  
    S_un;  
};
```

- Csak a 32 bites long változó (S_addr) használt.

```
#define s_addr S_un.S_addr
```

in_addr

- Ekvivalencia:

```
struct sockaddr_in address;  
address.sin_addr.s_addr = IPCIM;
```

```
struct sockaddr_in address;  
address.sin_addr.S_un.S_addr = IPCIM;
```

- A címnek hálózati byte sorrendben kell lennie.

hostent

```
struct hostent {  
    char *h_name;           //hivatalos nev  
    char **h_aliases;       //tovabbi nevek  
    int h_addrtype;         //cim család  
    int h_length;           //cím hossz  
    char **h_addr_list;     //cimek listaja  
};
```

```
#define h_addr h_addr_list[0]
```

- Host leíró információk.

Konvertáló függvények

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

IP cím kezelés

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

```
struct sockaddr_in address;
```

- char* → long

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

- char* → struct sockaddr_in

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

- struct sockaddr_in → char*

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

Byte sorrend konverzió

```
#include <netinet/in.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 protocol);
```

- Socket létrehozása.
- Visszatérési érték: OK: file leíró; hiba: -1
- family: AF_INET, PF_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.
- len: arg mérete.
- `#include<sys/socket.h>`

bind

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

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

listen

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

- Kapcsolatelfogadási szándék és queue méret beállítás.
- 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);
```

- Kapcsolat létrehozása.
- 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.
- Visszatérési érték: OK: új file leíró fd tulajdonságaival; hiba: -1.
- 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 (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.
- len: az üzenet hossza.
- flags: 0; MSG_00B: nagy prioritás.
- `#include<sys/socket.h>`

sendto

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

- Nem kapcsolat-orientált 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 (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.
- maxlen: a buffer hossza.
- flags: pl. 0; MSG_OOB csak az így küldött adatot veszi.
- `#include<sys/socket.h>`

recvfrom

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

- Nem kapcsolat-orientált adat küldés.
- 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.
- 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

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

getpeername

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

- Partner socket cím lekérdezés.
- Visszatérési érték: hiba esetén -1.
- fd: ezen a csatlakozón érhető el.
- addrp: ide kerül a távoli gép címinformáció.
- alenp: cím hossz.
- `#include<sys/socket.h>`

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<sys/socket.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.
- len: cím hossz.
- family: cím család, pl. AF_INET.
- `#include<sys/socket.h>`