# 17-18. Tétel

Hogyan valósítunk meg Linux alkalmazásunkban összeköttetés orientált / nélküli kommunikációt

## Linux hálózatkezelés

* #include <sys/socket.h>
* socketek
  + mint a Linux többi erőforrásai 🡪 egy-egy hálózati kapcsolat is állományként van reprezentálva
  + kapcsolat létrehozása 🡪 létrejön egy socket állomány 🡪read() write() 🡪 nem ajánlott
  + ajtó az alkalmazás és a hálózat között
    - IP cím
    - portszám
* hálózati kapcsolat felépítése
  + aszimmetrikus művelet

|  |  |
| --- | --- |
| **Kliens** | **Szerver** |
| Socket létrehozása | Socket létrehozása |
|  | Kapcsolódás portjának megadása (bind()) |
|  | Hallgatózás (listen()) |
| csatlakozási kérelem küldése: (connect()) |  |
|  | csatlakozás elfogadása (accept()) |
|  | csatlakozás létrejött, létrehoz egy kliens-socket-et 🡪 a kommunikáció ezek után ezen keresztül történik |

## Összeköttetés alapú kommunikáció (connection-oriented)

* folyamatos kapcsolatban van a két oldal között
* send, receive
* TCP
  + Transmission Control Protocol
  + megbízható átvitel
  + sorrendhelyes, hibamentes szállítást nyújt
  + ára: késleltetés (kapcsolat felépítése, bontása)
  + e-mail (smtp), ssh, web (http), ftp
* küldés: int send(int s, const void \*msg, size\_t len, int flags
  + s – socket leírója
  + \*msg – elküldendő adat buffere
  + len – elküldendő adat mérete
  + flags
    - MSG\_OOB – soron kívüli sürgős adatcsomag (out-of-band)
    - MSG\_DONTROUTE – ne a routeren keresztül, csak közvetlen hálózatra
    - MSG\_DONTWAIT – engedélyezi a nem blokkoló I/O-t
    - MSG\_NOSIGNAL – adatfolyam alapú kapcsolat esetén
* fogadás: int recv(int s, void \*buf, size\_t len, int flags)
  + jelzőbit eltérő:
    - MSG\_OOB – soronkívüli adat fogadása
    - MSG\_PEEK – adat beolvasása anélkül, hogy a beolvasott adatot eltávolítaná a bufferből
    - MSG\_WAITALL – addig nem tér vissza, amíg a puffer meg nem tellik, vagy rendhagyó dolog történik (pl. jelzés)
    - MSG\_NOSIGNAL

## Összeköttetés nélküli kommunikáció (connectionless)

* a kommunikációra „megbízhatóként tekint” 🡪 nem foglalkozik azzal, hogy valóban célba ért-e a csomag (ha hiba van, vagy megszakadt a csomag, akkor úgyis dob hibát a kommunikációs interface)
* olyan alkalmazásoknál, ahol nem probléma, ha egy-egy csomag elveszik
* UDP csomagok TCP helyett
  + UDP
    - User Datagram Protocol
    - korlátozott méretű adatcsomagok átvitelére képes, nem megbízható kommunikációs útvonalat biztosító protokoll
    - nem ganadtál célba jutást, csak hibakezelst nyújt
    - gyors
    - Hálózat menedzsment, Voip, média streaming, névfeloldás
    - blokkolás elkerülése: többszállúság, multiplexing
* küldő oldalon: socket létrehozása 🡪 adat küldése
  + küldés: int sendto(int sd, const void \*msg, size\_t len, int flags, const struct sockaddr \*to, socklen\_t tolen)
    - sd – socket descriptor
    - \*msg – message címe
    - len – message mérete
    - flags
    - \*to – socket pointere
    - tolen – socket mérete
* vevő oldalon:
  + megadjuk milyen porton figyelünk
  + honnan fogadjuk a csomagot (bárhonnan)
  + bind (socket hozzárendelése a helyi porthoz)
  + buffer törlése
  + üzenet fogadása (várakozás): int recvfrom(int sd, void \*buf, size\_t buffsize, int flags, struct sockadds \*from, socklen\_t \*fromlen)
    - ha a megadott puffer mérete kisebb, mint kéne, az érkező üzenet vége automatikusan levágódik