UNIX: folyamatok kommunikációja

kiegészítő fóliák az előadásokhoz

Mészáros Tamás

http://home.mit.bme.hu/~meszaros/

Budapesti Műszaki Egyetem

Méréstechnika és Információs Rendszerek Tanszék

A kommunikáció (alap)csatornái

- Jelzések
  - aszinkron események keltése és kezelése
- Csővezetékek
  - FIFO kommunikáció a „rokonságban”
- Szemaforok
  - a korábban megismert szinkronizációs megoldások
- Üzenetsorok
  - diszkrét, típusos üzenetek folyamatok között
- Osztott memória
  - azonos fizikai memóriaterület használata több folyamatban
- „hálózati” (socket) kommunikáció
  - címzéssel és protokollokkal támogatott kommunikáció

UNIX jelzések

- Cél (feladat)
  - értesítés a rendszer és más folyamatok eseményeiről
  - szinkronizációs primitívként is használható (nem ideális)
- Jelzés típusok (SIGINT, SIGHUP, SIGCHLD, SIGFPE, SIGKILL, ...)
  - rendszer: kivételek (pl. hibák), kvóta, riasztás, értesítés (gyerek leállt)
  - felhasználói: emberek (ctrl + c), folyamatok (tetszőleges céllal)
- A megvalósítás
  - rendszerhívással jelzést keltünk (értesítjük a kernelt)
  - a kernel értesíti a címzetteket a jelzésről
  - a címzett egy jelzéskezelő eljárásában fogadja a jelzést
- Problémák a megvalósítással
  - a létrehozás és a kézbesítés időben szétválik (akár elég messze is)
  - sokféle implementáció, némelyik nem túl megbízható (elveszhet jelzés)

UNIX csővezetékek: `pipe()`

- Cél
  - folyamatok közötti adatátvitel
- Jellemzők
  - csak szülő-gyerek viszonylatban
  - adatfolyam (nincs üzenethatár, tipizálás)
  - több író és olvasó is lehet, de egy adat csak egy helyre érkezhet meg
- A megvalósítás
  - egy folyamat létrehozhat egy csővezetéket (`pipe()`)
  - a kernel létrehozza az adatstruktúrákat és olvasási/írási leírókat ad vissza
  - a folyamat továbbadja a leírókat a gyerekének
  - a leírók segítségével kommunikálnak a folyamatok (`read()`, `write()`)
- Korlátok, problémák
  - nincs címzés, tipizálás
  - egymástól „független” folyamatok esetében nem működik

UNIX elnevezett csővezetékek

- Az egyszerű csővezetékek legkomolyabb problémájának megoldása
  - független folyamatok kommunikációja
- Jellemzők
  - nem csak szülő-gyerek viszonylatban
  - ugyanúgy működik, mint a csővezeték
  - a létrehozás a fájlrendszer segítségével történik
- Példa: kommunikáció az init folyamattal
  - a fájlrendszerben látható a csővezeték:

```
prw-------  1 root root 0 Jan  1 12:38 /dev/initctl
```

(a fájlrendszerrel kapcsolatos részletek később)

UNIX System V IPC

- Cél: folyamatok közötti „szabványos” kommunikáció
  - adatátvitel
  - szinkronizáció
- Közös alapok
  - erőforrás (fogalom): a kommunikáció eszköze
  - kulcs: azonosító az erőforrás eléréséhez
  - jogosultsági rendszer (fájlrendszerhez hasonlatos)
  - közös kezelőfüggvények: `*ctl(), *get()`
- Eszközök
  - szemaforok
  - üzenetsorok
  - osztott memória

UNIX System V IPC: szemaforok

- Cél: folyamatok közötti szinkronizáció
  - P() és V() operátorok
  - szemaforcsoportok kezelése

- Használat

```
sem_id = semget(kulcs, szám, opciók);
```

  - adott számú szemaforhoz nyújt hozzáférést (adott kulccsal és opciókkal)
  - a tényleges létrehozás és az egyszerű hivatkozás az opciókban válik szét

```
status = semop(sem_id, ops, ops_méret);
```

  - az ops struktúrában leírt műveletek végrehajtása (részletek: man semop)
  - egyszerre több művelet, több szemaforon is végrehajtható
  - blokkoló és nem blokkoló P() operáció is lehetséges
  - egyszerű tranzakciókezelésre is van lehetőség

UNIX System V IPC: üzenetsorok

- Cél: folyamatok közötti adatátvitel
  - diszkrét, tipizált üzenetek
  - nincs címzés, üzenetszórás

- Használat

```
msgq_id = msgget(kulcs, opciók);
```

  - adott kulcsú üzenetsorhoz nyújt hozzáférést (adott opciókkal)
  - a tényleges létrehozás és az egyszerű hivatkozás az opciókban válik szét

  - üzenetküldés:

```
msgsnd(msgq_id, msg, méret, opciók);
```

  - vétel: `msgrcv(msgq_id, msg, méret, típus, opciók);`

  - a típus (egész szám) beállításával szűrést valósíthatunk meg
    - `= 0` a következő üzenet (tetszőleges típusú)
    - `> 0` a következő adott típusú üzenet
    - `< 0` a következő üzenet, amelynek a típusa kisebb vagy egyenlő

UNIX System V IPC: osztott memória

- Cél: folyamatok közötti egyszerű és gyors adatátvitel
  - a kernel helyett közvetlen adatátviteli csatorna
  - a fizikai memória elkülönített része, amely közösen használható

- Használat

```
shm_id = shmget(kulcs, méret, opciók);
```

  - adott kulcsú osztott nyújt hozzáférést (adott opciókkal)
  - a tényleges létrehozás és az egyszerű hivatkozás az opciókban válik szét

  - hozzárendelés saját virtuális címtartományhoz:

```
változó = (típus) shmat(...);
```

    az adott változót hozzákötjük a visszakapott címhez

  - lecsatolás: `shmdt(cím);`

  - a kölcsönös kizárást meg kell valósítani (pl. szemaforokkal)

UNIX: socket kommunikáció

- Cél: címzéssel és protokollokkal támogatott adatátvitel
  - tetszőleges folyamatok között kliens – szerver architektúrában
  - sokféle protokoll (TCP/IP család)
  - többféle címzés
  - gépen belül és gépek között is

- Fogalmak
  - socket: hálózati csatoló (azonosító)
  - IP cím és portszám (I. hálózatok)

- Használat

```
sfd = socket(domén, típus, protokoll);
szerver: bind(sfd, cím, ...);
kliens: connect(sfd, cím, ...);
szerver: listen(sfd, sor_mérete);
szerver: accept(sfd, cím, ...);
send(sfd, üzenet, ...);
recv(sfd, üzenet, ...);
shutdown(sfd);
```

Kliens és szerver programok váza

Kliens program

```
socket()


connect()


send()
recv()
close()
```

Szerver program

```
sfd1 = socket()
bind(sfd1)
listen(sfd1)
while
    sfd2 = accept(sfd1)
    fork()
```

szülő: vissza a ciklusba

gyerek:

```
recv(sfd2)
send(sfd2)
close(sfd2)
exit()
```

(Sun) RPC (távoli eljáráshívás)

- A socket kommunikációra épülő „elosztott rendszer” infrastruktúra
- Cél:
  - folyamatok közötti magas szintű kommunikáció
  - távoli eljárások meghívása (másik folyamatban, akár másik gépen)
  - programozói támogatás: interfész leírás + programgenerálás

- Fogalmak
  - RPC nyelv: a hívható eljárások és típusaik (interfész) leírása
  - azonosítók: a leírásban megadott egyedi számok (program, eljárás)
  - portmapper: a programazonosítók és a hálózati portok összerendelése
  - rpcgen: a leírásból C programkódot generáló program

- Ez a mai elosztott rendszerek (DCOM, CORBA, RMI, stb.) őse
  - interfész leírás
  - programgenerátor
  - kommunikációs infrastruktúra