# 14. Tétel

Hogyan hozhatunk létre folyamatokat a Linux alkalmazásokban és milyen eszközeink vannak a folyamatok közötti kommunikációra?

## Processzek

OS-el szemben támasztott alapvető követelmény: **multitasking**

### Egy processz két egyforma processzé bontása:

* pid\_t fork(void); 🡪 ezután a definiált fv, akármi párhuzamosan létezik
* visszatérési érték: PID
* felbontja a processzt szülő, és gyermek processzre
  + szülő visszatérési értékként megkapja a gyermek PID-jét
  + gyermek 0-t kap
  + hiba esetén -1

### Processz megölése

int kill(pid\_t pid, SIGINT) (hagyományos mód)

### A processz saját azonosítójának lekérése

pid\_t getpid(void);

### Szülő PID-je

pid\_t getppid(void)

### Processz várakoztatása a gyermekei végéig

pid\_t wait(int \*status)

* visszatér, ha a hívó processz gyermek processzei közül bármelyik befejezte a működését
* ha a hívás pillanatában a gyermek már zombi 🡪a fv. azonnal visszatér

pid\_t waitpid(pid\_t pid, int \*status, int options) – flexibilisebb

* pid paraméterrel egy meghatározott PID-ű gyermek kilépésére várakozhatunk, az opciótól függően
* opciók
  + <-1 – vár bármely gyermek végére, aminek a csoportazonosítója megegyezik a PID paraméterben adottal
  + -1 – a függvény ugyanúgy működik, mint a sima wait
  + 0 – vár bármely gyerek végére, amelynek a GID-je megegyezik a hívó processzével
  + > 0 – vár bármely gyerek végére, amelynek a PID-je megeggyezik a pid-el
  + WNOHANG – felfüggesztés nélkül visszatér (no hang), ha még egy gyerekfolyamat sem ért véget (OR kapcsolatban valamelyik fentivel általában)

## Processzek közötti kommunikáció (IPC)

**IPC – Interprocess Communication**

ipcs program – kiírja a memóriában lévő olyan IPC objektumokat, amelyekhez a hívó processznek olvasási joga van

* csak az üzenetsorokat
* csak a szemaforokat
* csak az osztott memóriát

ipcrm – egyes IPC objektumok eltávolítása a kernelből – pl. ipcrm sem 81985536 – az adott ID-jű szemafor eltávolítása

### Szemfaorok

* Unsigned int – számláló – init egy meghatározott értékre – meghatározza hány processz foglalhatja le
* #include <sys/sem.h>
* módosításának atominak kell lennie
* erőforrás lockolása – szemafor dekrementálása 🡪 ha már 0, több processz nem foglalhatja le
* létrehozása:
  + int semget(key\_t key, nit nsems, int semflg)
    - key – egyedi azonosító – ezzel lehet rá hiatkozni
      * ha már létezik ilyen, a fv. a létező kulcsával fog visszatérni
      * kulcskeresés: ftok fv.
    - nsems – létrehozandó szemaforok száma
    - semflg – hozzáférési jogosultság
* int semctl(int semid, int semnum, int cmd, …); - szemaforvezérlés
  + cmd lehet:
    - IPC\_STAT – szemaforinfó lekérdezése (olvasási jogosultság szükséges)
    - IPC\_SET - jogosultság uid, gid módosítása
    - IPC\_RMID – szemafortömb megszüntetése, felébresztve a várakozó processzeket
    - GETALL – szemafortömb elemeinek értékét adja vissza
    - GETNCNT – egy adott szemaforra várakozó processzek száma
    - GETPID – szemafortömb utolsó módosítójának PID-je
    - GETVAL – key
    - GETZCNT – egy szamfor nulla értékére várakozó processzek száma
    - SETALL – szemafortömb összes értékét állítja be
    - SETVAL – egy szemafor értékét állítja
* int semop(int semid, struct sembuf \*sops, unsigned nsops);
  + szemaforra várakozás
  + semid – szemafortömb azonosítója
  + sembuf – struktúrák egy tömbje – a végrehajtandó műveletet írják elő
    - sem\_num – szemafor indexe a tömbben
    - sem\_op – szemafor értékének változtatása
    - sem\_flg – művelet jelzőbitjei
      * IPC\_NOWAIT – a műveletet megkísérli azonnal végrehajtani, különben hiba
      * SEM\_UNDO – a művelet végrehajtódik, amikor a hívó processznek vége lesz

### Üzenetsorok (message queues)

* FIFO kommunikációs csatorna, amibe a programozó által meghatározott formátumú adatcsomagokat lehet belerakni
* #include <sys/msg.h>
* egy üzenetsorban több üzenetcsatornát lehet használni
* fizikailag: linked list a kernel címterében (struct msg)
  + struct msg \*msg\_next – a következő üzenet a sorban
  + long msg\_type – az üzenet típusa
  + char \*msg\_spot – magára az ünenetre mutató pointer(a kernel nem tud semmit a formátumról)
  + short msg\_ts – az üzi mérete
* az msgbuf struktúra az alapja annak, ahogy a rendszer leír egy üzenetet
  + long mtype – üzenettípus azonosítója
  + char mtext[1] – üzenet szöveg tartalma
* az msgbuf újradefiniálható, tartalmazhat komplex adatot (lehet definiálni egy saját struktúrát)
  + ekkor az msgbuf struktúra:
    - long mtype
    - struct definiált\_üzenet\_struktúra struktúranév
* int msgget(key\_t key, int msgflg)
  + üzenetsor létrehozása
  + key generálása: ftok fv.
  + jelzőbitek
    - IPC\_CREAT – létrehozás
    - IPC\_EXCL – IPC\_CREAT-el együtt használva: visszatér hamis értékkel, ha a létrehozandó szemafor már létezik
  + amennyiben a megadott kulcs létezik, akkor a függvény a már létező üzenetsor azonosítójával tér vissza. Egyébként az azonosító az újonnan létrehozotté
* int msgsnd(int msqid, struct msgbuf \*msgp, size\_t msgsz, int msgflg)
  + üzenet küldése: írása a sorba
  + msqid – üzenetsor azonosítója
  + \*msgp – saját üzenetünk pointere
  + ha a sor tele van 🡪 hiba
  + jelzőbit: IPC\_NOWAIT vagy 0
* int msgrcv(int msqid, struct msgbuf \*msgp, size\_t msgsz, long msgtype, int msgflg)
  + annak a memóriaterületnek a pointere, ahova az üzenetet kérjük
  + kiolvasás menete:
    - ha az msgtype 0 🡪a soron következő üzenetet olvassa ki
    - pozitív és az MSG\_EXCEPT bit nincs bekapcsolva 🡪 legelső üzenet, aminek a típusa msgtype
    - ha az MSG\_EXCEPT bit bekapcsolva 🡪 első üzenet, aminek a típusa nem msgtype
    - ha msgtype negatív 🡪 annak a kiolvasása, amelynek típusa kisebb vagy egyenlő, mint az msgtype abszolút értéke
* int msgctl(int msqid, int cmd, struct msqid\_ds \*buf)
  + msqid – üzenetsor azonosítója
    - cmd
    - IPC\_STAT – info másolása a buf-ba
    - IPC\_SET – a buf által mutatott sturktúra némely tagja alapján állítja az üzenetsor tulajdonságait
      * msg\_perm.uid
      * msg\_perm.gid
      * msg\_perm.mode (alsó 9 bit)
      * msg\_qbythes
    - IPC\_RMID – üzenetsor megsemmisítése
  + \*buf - tulajdonságokat rögzítő struktúra
    - struct ipc\_perm msg\_perm – hozzáférési jogosultságok
    - sturct msg \*msg\_first – az első üzenet az üzenetsor lácolt listájában
    - sturct msg \*msg\_last – az utolsó üznenet a láncolt listában
    - time\_t msg\_stime – legutolsó küldés ideje
    - time\_t msg\_rtime – legutolsó olvasás ideje
    - time\_t msg\_ctime – legutolsó változtatás ideje
    - struct wait\_queue wwait
    - struct wait queue\_rwait
    - ushort msg\_cbytes – az üzenetsorban levő bájtok száma, azaz az üzenetek száma
    - ushort msg\_qnum – az éppen az üzenetsorban lévő üzenetek száma
    - ushort msg\_qbytes – az üzenetsorban levő bájtok maximális száma
    - ushort msg\_lspid – a legutolsó küldő processz azonosítója
    - ushort msg\_lrpid – a legutolsó olvasó processz azonosítója

### Megosztott memória/Shared memory

* Közös memóriatartomány, amihez több processz is hozzáférhet 🡪 leghatékonyabb, leggyorsabb
* fork 🡪a gyermek processz örökli a szülőhöz csatolt összes megosztott memóriatartományt
* #include <sys/shm.h>
* int shmget(key\_t key, int\_size, int shmflg)
  + közös memóriatartomány létrehozása
  + key – ftok
  + shmflag – IPC\_CREAT, IPC\_EXCL
  + size – a kívánt memória mérete bájtokban
    - PAGE\_SIZE – spec érték – memória lap mérete 🡪 ajánlott ennek egész szánú többszörösét lefoglalni
* void \*shmat(int shmid, const void \*shmaddr, int shmflg)
  + közös memóriatartományhoz való hozzácsatlakozás (attach)
  + visszatérési érték: memóriatartomány pointere
  + shmflg
    - SHM\_RND – a visszaadott cím az shmaddr értéke lesz lekerekítve a legközelebbi laphatárra
    - gyakorlatban általában 0 – a rendszerre bízza a megfelelő címtartomány kiválasztását
    - SHM\_RDONLY – csak olvasásra csatolja a megosztott memóriát
* int shmdt(const void \*shmaddr) – megosztott memóriáról lecsatlakozás (detach)
* int shmctl(int shmid, int cmd, struct shmid\_ds \*buf)
  + shmid – shared memory id
  + cmd
    - IPC\_STAT – infó a megosztott memóriáról
    - IPC\_SET – hozzáférési jogosultságok és/vagy azonosítók megváltoztatása
    - IPC\_RMID – megosztott memória törlése
      * ha nincs olyan processz, ami csatolva tartaná az erőforrást 🡪 azonnal megszűnik
      * ha van 🡪 megjelöli megszüntetésre, hogy az utolsó lecsatolás után dögöljön meg
    - SHM\_UNLOCK – engedélyezi a swappinget
  + \*buf
    - struct ipc\_perm shm\_perm – hozzáférés és azonosítók beállítása
    - int shm\_segsz – memóriatartomány mérete
    - time\_t shm\_atime – legutolsó attach ideje
    - time\_t shm\_dtime – legutolsó detach ideje
    - time\_t shm\_ctime – utolsó változás ideje
    - unsigned short shm\_cpid – a létrehozó processz azonosítója
    - unsigned short shm\_lpid – az utolsó művelet végrehajtójának azonosítója
    - short shm\_nattch – aktuális felcsatolások száma
    - shm\_npages – tartomány mérete (nem írható)
    - unsigned long \*shm\_pages – (nem írható)
    - struct vm\_area\_struct \*attaches – felcsatolások leírója (nem írható)