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

* cska 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
* 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
* 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 – maga az üzenet (a kernel nem tud semmit a formátumról, ez csak egy pointer)
  + short msg\_ts – az üzi mérete
* saját üzenetformátum megadása
  + default:
    - long mtype – üzenettípus
    - char mtext[1] – adat

Saját üzenet\_temlate definiálás

struct studentinfo {

char nev[40];

char cím[80];

char igszam[20];

char megj[80];

};

struct studentinfo\_msg {

long mtype;

struct studentinfo data;

}

Megosztott memória