Folyamatok ütemezése a UNIX-ban

Folyamatok ütemezése a UNIX-ban

- Ütemező algoritmus megválasztása meghatározza:
  - a rendszer teljesítményét,
  - a hardver kihasználtságát.
- Tradicionális UNIX ütemezési algoritmus:
  - A System V. R3, ill. 3.1 BSD UNIX,
  - Néhány megoldás már idejétmúlt.
- A mai UNIX rendszerek:
  - algoritmus valamilyen továbbfejlesztett változata.

Az ütemezési algoritmussal szemben támasztott követelmények

- A UNIX rendszer feltételezett felhasználása:
  - több felhasználós,
  - interaktív programok,
  - batch programok.
- Követelmények:
  - Alacsony válaszidő (interaktív folyamatok).
  - Nagy átbocsátó képesség.
- Éhezés elkerülése
  (háttérben futó pl. batch folyamatok).

A UNIX-ütemezés rövid jellemzése

- Prioritásos ütemezés, felhasználói, ill. kernel módban eltérő algoritmus.
  - **felhasználói** mód:
    preemptív ütemezés, időosztásos, időben változó prioritás.
  - **kernel** módban futó folyamatok ütemezése:
    nem preemptív ütemezés, fix prioritású folyamatok.

Ütemezés megvalósítása

- Ütemezéshez kapcsolódó tevékenységek (pl. prioritási szám számítása, legnagyobb prioritású folyamat kiválasztása stb.) önálló függvényekben van megvalósítva.
- Óra megszakításhoz kötődik az ütemezési rutinok végrehajtása:
  - ütemezési ciklusonként (egy v. néhány óra megszakításonként).
- Mechanizmus:
  - call-out függvények.

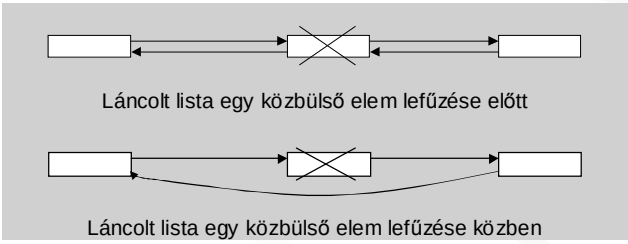
Felhasználói módú ütemezés

- Felhasználói módban az ütemezés *preemptív*:
- Újraütemezés:
  - Ha egy magasabb prioritású folyamat futásra késszé válik.
  - Ha több azonos prioritású folyamat van futásra készen és nincs náluk magasabb prioritású futásra kész folyamat: Round-Robin, FCFS.

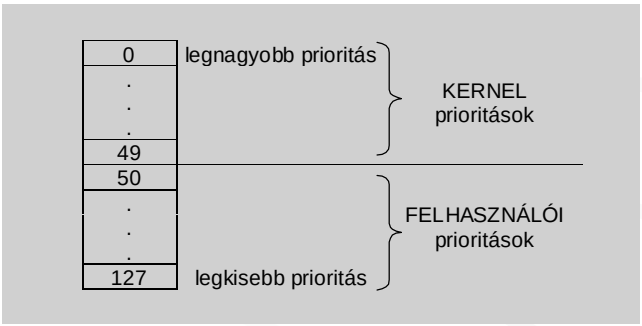
Kernel módú ütemezés

- A kernel módban az ütemezés szigorúan *nem preemptív*:
  - Kernel kódot végrehajtó folyamatot nem lehet kényszeríteni, hogy lemondjon a CPU használatról egy nagyobb prioritású folyamat javára.
  - Átütemezés kernel módban:
  - – önként lemond a futás jogáról (sleep).
  - – folyamat visszatér kernel módból felhasználói módba.
- A kernel kód korlátozottan reentrens!

Példa ami indokolja, hogy a kernel nem preemptív



Folyamatok ütemezési prioritása



Prioritás meghatározása kernel módban

- P:prioritás A(km^2) W:Munka s:út t: idő

$$P = \Psi * \oiint_A \sqrt[3]{\frac{f \pm g / l(.)(o,t,r)}{\langle e,r \rangle}} dA \approx Q$$

Prioritás meghatározása kernel módban

- Kernel módú prioritás meghatározása:
  - A rendszer milyen ok miatt hajtott végre sleep rendszerhívást, vagyis
  - **milyen eseményre várakozik**.
  - Kernel prioritást szokták **alvási prioritásnak** is nevezni.

Prioritást meghatározó tényezők felhasználói módban

- Két fő tényező:
  - *nice szám* (kedvezési szám):
    - felhasználó beavatkozása a prioritás számításába
  - CPU használat:
    - öregítés
    - egyenletes CPU használat

Prioritás számításához használt paraméterek

- A prioritás számításához négy paramétert használ a UNIX:
  - **p_pri**: aktuális ütemezési prioritás
  - **p_usrpri**: felhasználói módban érvényes prioritás
  - **p_cpu**: a CPU használatra vonatkozó szám
  - **p_nice**: a felhasználó által futás adott nice szám
- A paramétereket minden folyamat esetén külön-külön számon tartja a UNIX.

Prioritás számításához használt konstansok

- Konstans: P_USER
  - Legmagasabb user módú prioritás
- Korrekciós faktor (KF):
  - rendszer terheltségének figyelembevétele
  - sok folyamat: lassú öregítés
  - kevés folyamat: gyors öregítés
  - várakozó folyamatok számával fordítottan arányos érték

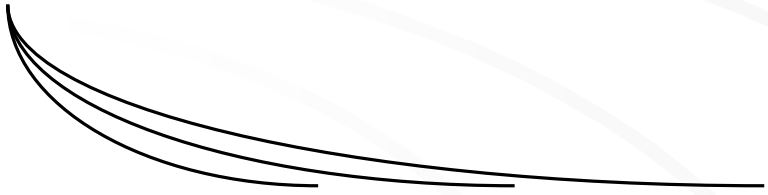
Prioritás számítása felhasználói módban

	futó folyamat	minden folyamat
minden interrupt	p_cpu := p_cpu +1	Megvizsgálja, van-e a futónál magasabb prioritású folyamat. Ha van, átütemez.
minden 10. interrupt	Round-Robin algoritmus: ha több azonos prioritású folyamat van a legmagasabb prioritású pozícióban, 10 óra-interrupt-onként váltja a futó folyamatot	
minden 100. interrupt		1. korrekciós faktor számítása 2. p_cpu = p_cpu * KF 3. p_usrpri = P_USER+p_cpu/4+2*p_nice

Korrekciós faktor

KF=várakozó folyamatok száma/
várakozó folyamatok száma+1

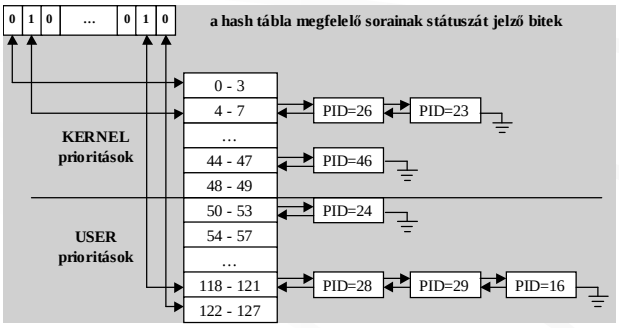
½, 2/3,3/4…



Környezetváltás ütemezéskor

- Nem preemptív ütemezés:
  - *várakozik:* sleep *rendszerhívás.*
  - *kilép:* exit *rendszerhívást hajt végre.*
- Preemptív ütemezés:
  - *A 100-adik óraciklusban a prioritások újraszámításakor*
  - *A 10-edik óraciklus esetén a Round-Robin algoritmus (azonos prioritás)*
  - Felébred (ready to run állapotba jut) egy, az aktuálisan futónál magasabb prioritású folyamat

Adatszerkezetek folyamatok prioritásának tárolására



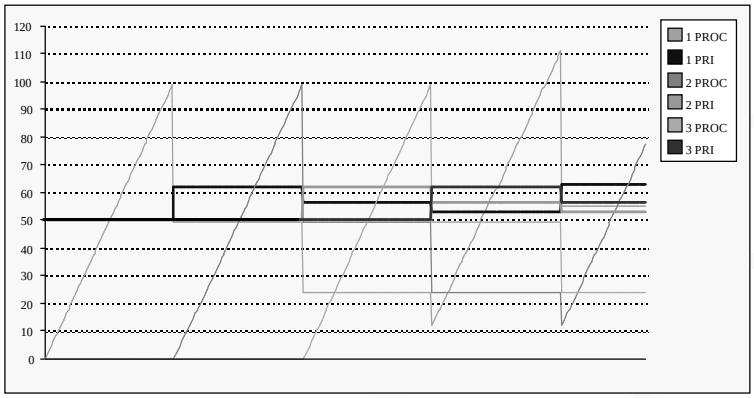
Példa az ütemezés számolására I.

- minden óra megszakításnál:
  p_cpu := p_cpu +1
- minden 100-adik óra-interruptnál :
  p_cpu = p_cpu * KF = p_cpu * ½
  p_usrpri = P_USER+p_cpu/4+2*p_nice
- P_USER=50, p_nice=0

UNIX ütemezési példa

A		B		C			
p_pri	***p_cpu***	***p_pri***	***p_cpu***	***p_pri***	***p_cpu***	***Lépés***	***Futó foly.***
50	*0*	50	0	50	0	1	A
50	*1*	50	0	50	0	2	**A**
…	…	…	…	…	…	…	**A**
50	*99*	50	0	50	0	99	**A**
50+50/4 63	***100/2 50***	**50**	**0**	**50**	**0**	**100**	**A**
63	50	50	*1*	50	0	101	**B**
…	…	….	…	…	…	…	**B**
63	50	50	*99*	50	0	199	**B**
50+25/4 56	**50/2 25**	**50+50/4 63**	***100/2 50***	**50**	***0***	**200**	**B**
56	25	63	50	50	*1*	201	**C**
…	…	….	…	…	…	…	**C**
56	25	63	50	50	*100*	299	**C**
50+13/4 53	**25/2 13**	**50+25/4 56**	**50/2 25**	**50+50/4 63**	***100/2 50***	**300**	**C**

Példa az ütemezés számolására II.



A UNIX-ütemezés értékelése

- Nem méretezhető megfelelően a terhelés függvényében:
  - A korrekciós faktor használata nem elég hatékony.
- Az algoritmussal nem lehet meghatározott CPU időt allokálni.
- Nem lehet a folyamatok fix válaszidejét garantálni.
- Ha egy kernel rutin sokáig fut, az feltartja az egész rendszert.
- A felhasználó a folyamatainak prioritását nem tudja megfelelő módon befolyásolni.

Call-out függvények

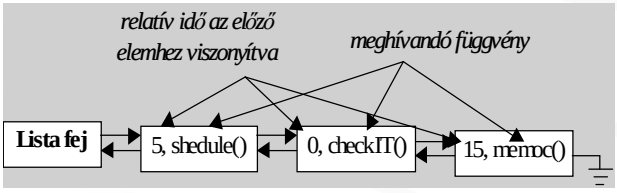
Call-out I.

- A *call-out* mechanizmus egy adott tevékenység későbbi időpontban történő végrehajtása:
  - timeout (függvény)
  - untimeout (függvény)
- Ismétlődő feladatok (kernel funkciók) végrehajtása:
  - hálózati csomagok ismételt elküldése,
  - ütemezési és memóriakezelő függvények hívása,
  - perifériák monitorozására,
  - perifériák lekérdezésére, amelyek nem támogatják a megszakításokat ...

Call-out II.

- A megszakítás-kezelő nem közvetlenül hívja meg a call-out függvényeket, hanem beállít egy erre a célra rendszeresített jelzőbitet.
- Dinamikus adatszerkezet a call-out függvények kezelésére:
  - láncolt listás
  - időkerekes.

Call-out függvények láncolt listás ábrázolása



Call-out függvények tárolása időkerékkel