Operációs rendszerek 2.

Az OS erőforrás kezelés

IT alapok tantárgyhoz

Tibi V 2019

Erőforrás kezelés

1. Folyamatok - processzusok
2. Folyamat állapotok
3. PCB
4. Szálak
5. Ütemezés
6. A rendszerhívás
7. Folyamatok kommunikációja

1. A folyamatok - processzusok

- **A program a háttértárolón várakozó, végrehajtható fájl.**
- **A folyamat a program egy memóriában futó példánya és az adott példány végrehajtásával kapcsolatos OS tevékenység.**

Def: A folyamat (processz) fogalma: program végrehajtása + vele (azzal) kapcsolatos teendők összessége.

- Példa: *Főzés + Recept*

A folyamatok menedzselése a modern operációs rendszerek egyik legfontosabb tevékenysége.

Az operációs rendszer feladata, hogy biztosítson a processzek számára „saját logikai processzort”, ütemezze számukra a valódi processzor idejét, „kapcsolja” számukra a valódi processzort, gondoskodjon az elválasztott „logikai processzorok” állapotának megtartásáról.

Processzus (folyamat) létrehozása

- Egyszerű rendszer: kezdetben minden processzust létrehozunk
- Általános célú rendszer: Működés közben kell létrehozni

Négy esemény hatására jöhet létre processzus:

- A rendszer inicializálása
- A processzust létrehozó rendszerhívás létrehozása
- A felhasználó egy processzus létrehozását kéri
- Kötegelt feladat kezdeményezése

Processzus létrehozása

Rendszer induláskor több processzus keletkezik

Első elinduló folyamat az ütemező(PID 0 láthatatlan)

- **Előtérben futó**: Felhasználókhoz kapcsolódnak
- **Háttérben futó**: Nincs felhasználóhoz kötve

  Sajátos feladatuk van: démonok (daemon)

  PL.: nyomtatás kezelő, SSH szerver,

Processzus létrehozása a gyakorlatban

- Felhasználó bejelentkezik
- Szolgáltatás használata, pl. nyomtatás
- Program elindít egy másikat

Egyéb processzus műveletek:

- Processzus megszakítása (magasabb rangú folyamat indítása miatt, várakozás)
- Gyermek (child) processzus létrehozása

Processzus Hierarchia:

- Egy folyamatnak lehet gyermeke.
- Gyermek-folyamatnak lehet saját gyermeke.
- Minden gyermeknek csak egy szülő folyamata van.
- Együtt csoportot (UNIX/LINUX -group) alkotnak. Windowsban ilyen hierarchia nincsen

Processzus befejezése

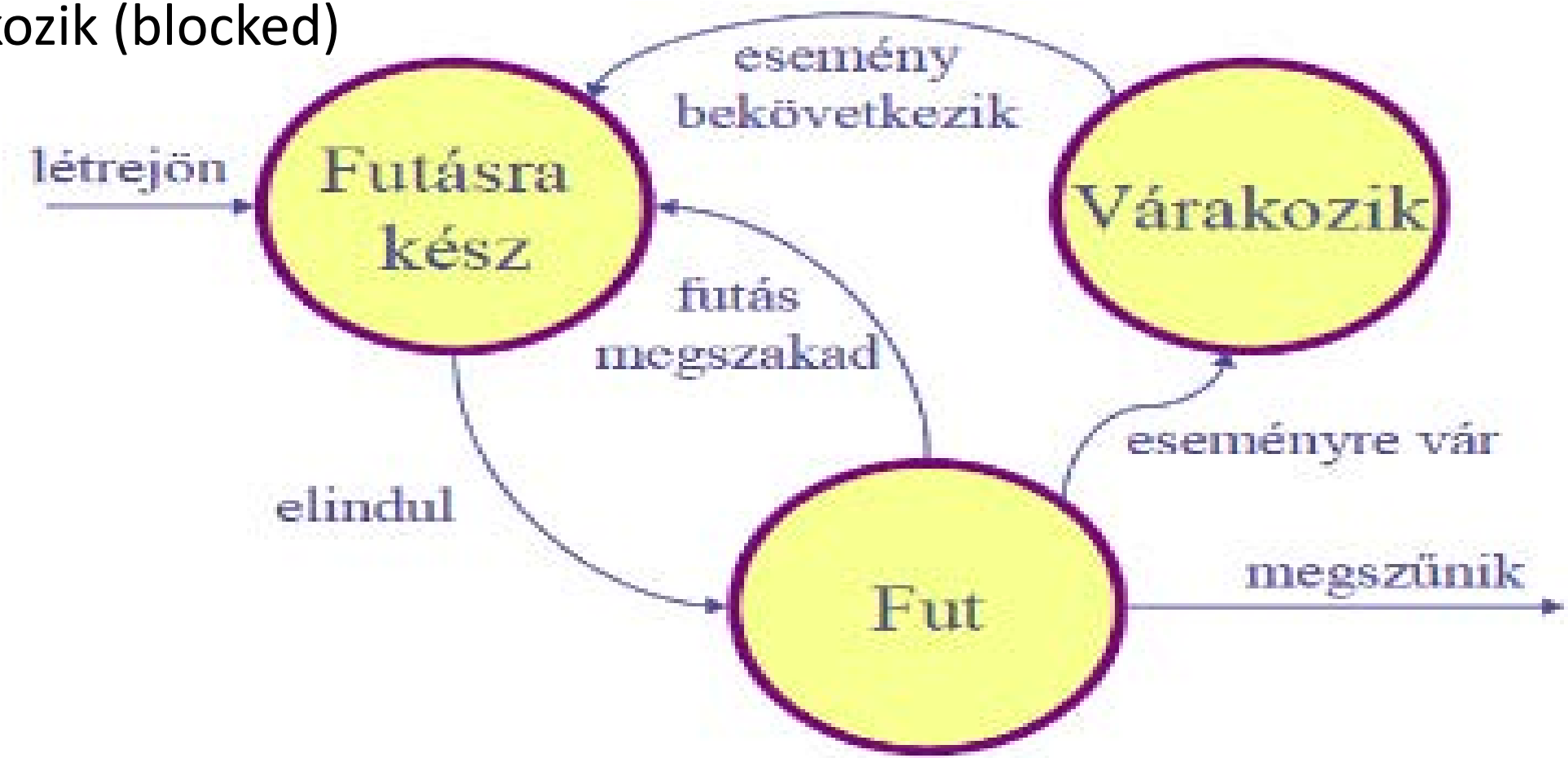
- Szabályos kilépés (önkéntes)
- Kilépés hiba miatt (önkéntes)
- Kilépés végzetes hiba miatt (önkéntelen)
- Egy másik processzus megsemmisíti (önkéntelen)

Processzus befejezése, gyakorlatban

- Időkorlát lejárt
- Nincs több memória
- Matematikai hiba Zérussal osztás
- Érvénytelen utasítás
- Operációs rendszer leállítja (holtpont elkerülés)
- Szülő leállítja

2. Folyamat állapotok (álltalános) - state

1. Futásra kész(ready)
2. Fut (running)
3. Várakozik (blocked)

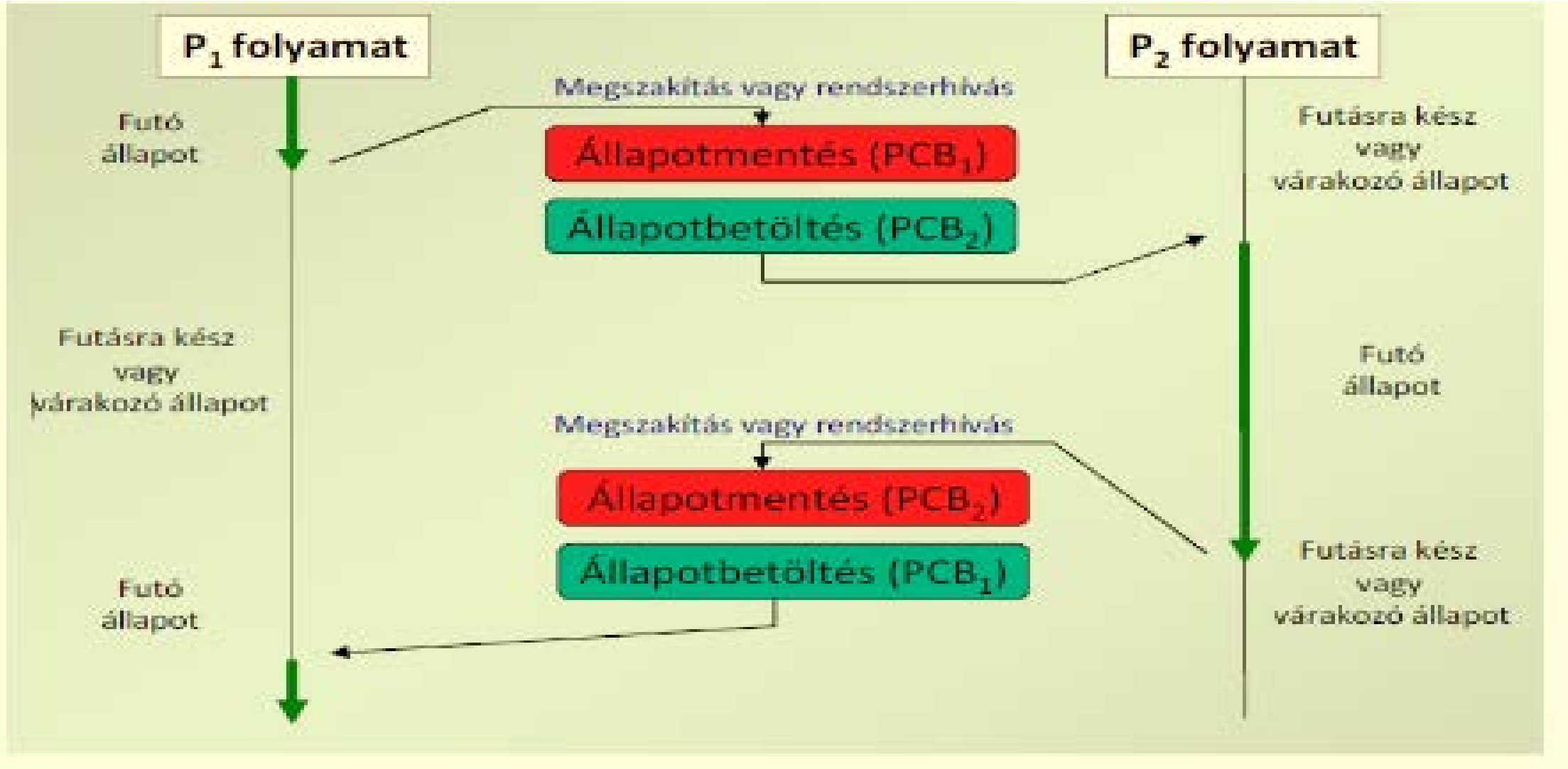


Folyamatok állapotainak változása

Az állapot átmenetek a következő események hatására következnek be:

1. A folyamat olyan rendszerhívást adott, amelynek perifériára kell várnia vagy a folyamat blokkolást kért.
2. Az ütemező úgy döntött, hogy épp elég régóta fut a folyamat, másnak adja a futás jogát (időmultiplexelés).
3. Az ütemező futási jogot ad a folyamatnak.
4. Bekövetkezett az esemény, amelyre a folyamatnak várnia kellett, a folyamat képessé vált a futásra.

A teendők:



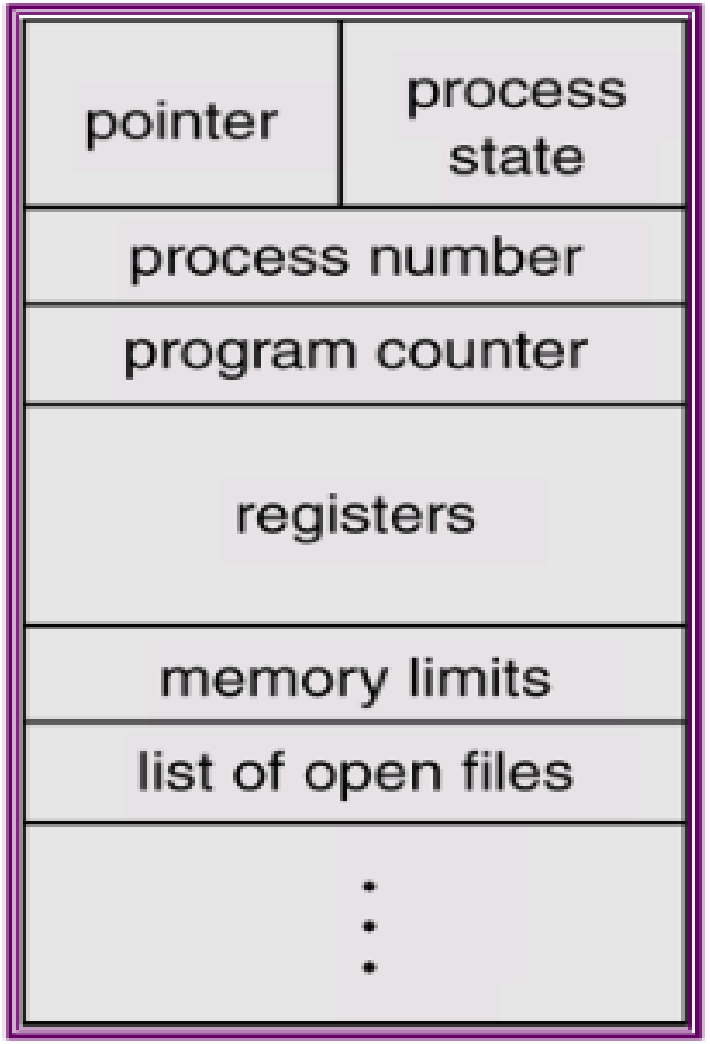
ahol PCB – Process Control Block – folyamat leíró blokk

3. A processz tábla

Az operációs rendszer nyilvántartja a futó folyamatokhoz tartozó információkat, ezt a folyamatosan változó nyilvántartást hívják processz táblának.

- Ebben táblázatszerű adathalmazban található minden, amit a folyamatok futásának adminisztrációjához kell:
- a futtatandó kód
- adatterület elhelyezkedése,
- regisztertartalmak,
- menedzsmentinformációk (pl. jogosultság)

Ezen adatok egyetlen folyamathoz tartozó összességét **folyamat tartalomnak (process context)** hívjuk.



PCB struktúrája

Megvalósítás: Folyamat tábla mezői (process table)

Process management	**Memory management**	**File management**
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Folyamat állapotok (Linux) részletesen

Egy multiprogramozású rendszerben (multitasking, multiprocessing) egy időben több folyamat él.

Folyamatok születnek, vetélkednek, együttműködnek, kommunikálnak, végül meg-szűnnek.

Egy új folyamat létrejötte a **nemlétező (Non-existent**) állapotból indul és bekerül a futásra kész, tehát futásra várakozó folyamatok listájára.

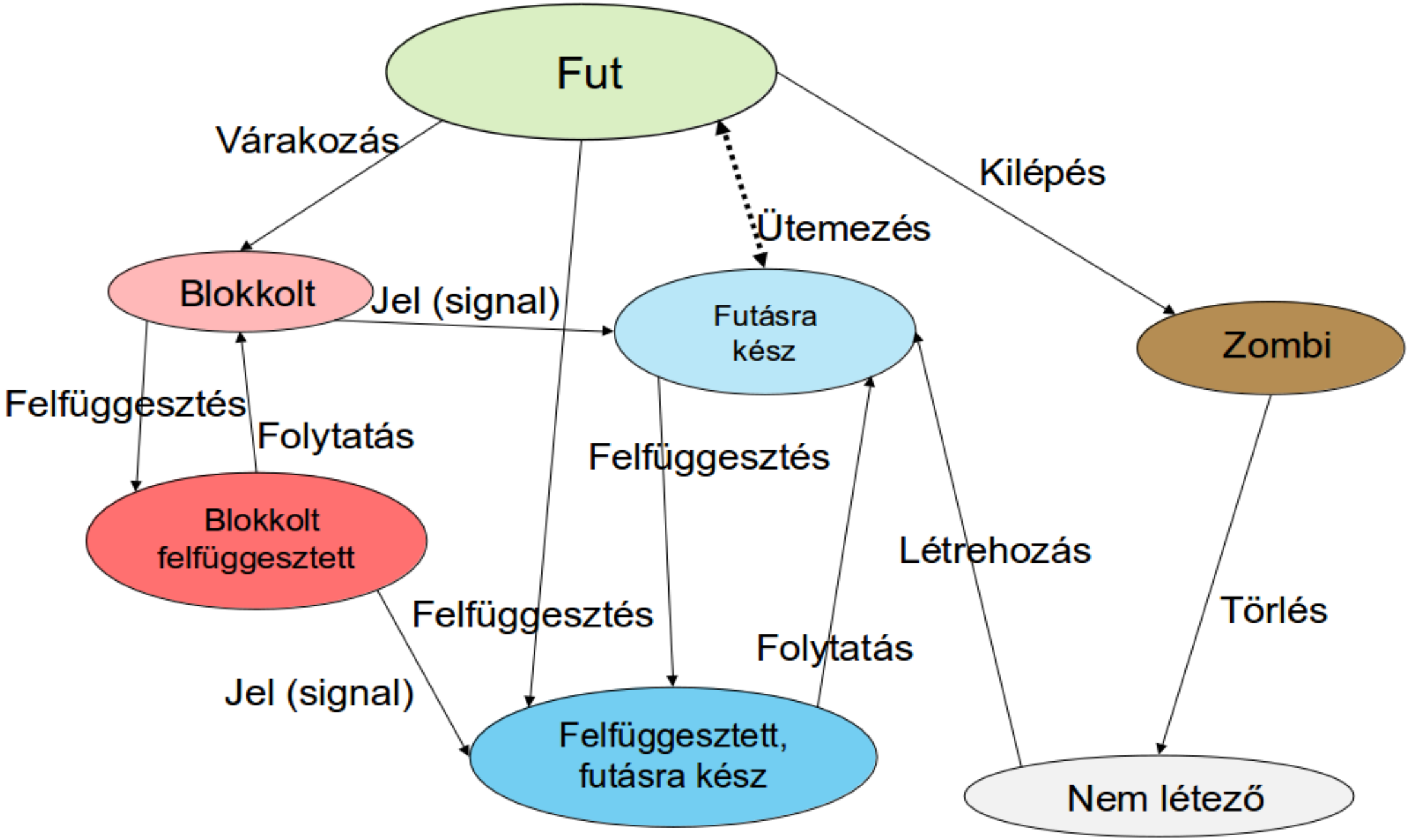
Az épp **futó (Running)** folyamatot vagy az ütemező teszi vissza egy idő után a futásra várakozók sorába,

Vagy „saját akaratából” kerül **blokkolt** állapotba, mert például egy számára szükséges inputra vár.

A folyamatokat futásukban fel lehet **függeszteni (Suspended**), ez természetesen a blokkolt, a futó és a futásra váró állapotban is megtörténhet.

Futó állapotból a felfüggesztés megszűntével **futásra váró állapotba** kerül a folyamat, a blokkolt állapotból pedig úgy kerülhet ki, hogy **egy jelzés (signal**) értesíti arról, hogy a blokkolást kiváltó körülmény megszűnt.

Egy folyamat befejezésekor annak futása **megszűnik (Zombie),–** ekkor már nem is beszélhetünk folyamatról – a nemlétező állapotba kerül.


Fut
Várakozás
Kilépés
Ütemezés
Blokkolt
Jel (signal)
Futásra kész
Zombi
Felfüggesztés
Folytatás
Felfüggesztés
Blokkolt felfüggesztett
Létrehozás
Törlés
Felfüggesztés
Folytatás
Jel (signal)
Felfüggesztett, futásra kész
Nem létező

Processzus állapot információk a gyakorlatban

UNIX/LINUX parancs:

- # ps

root@server: /home/tibi

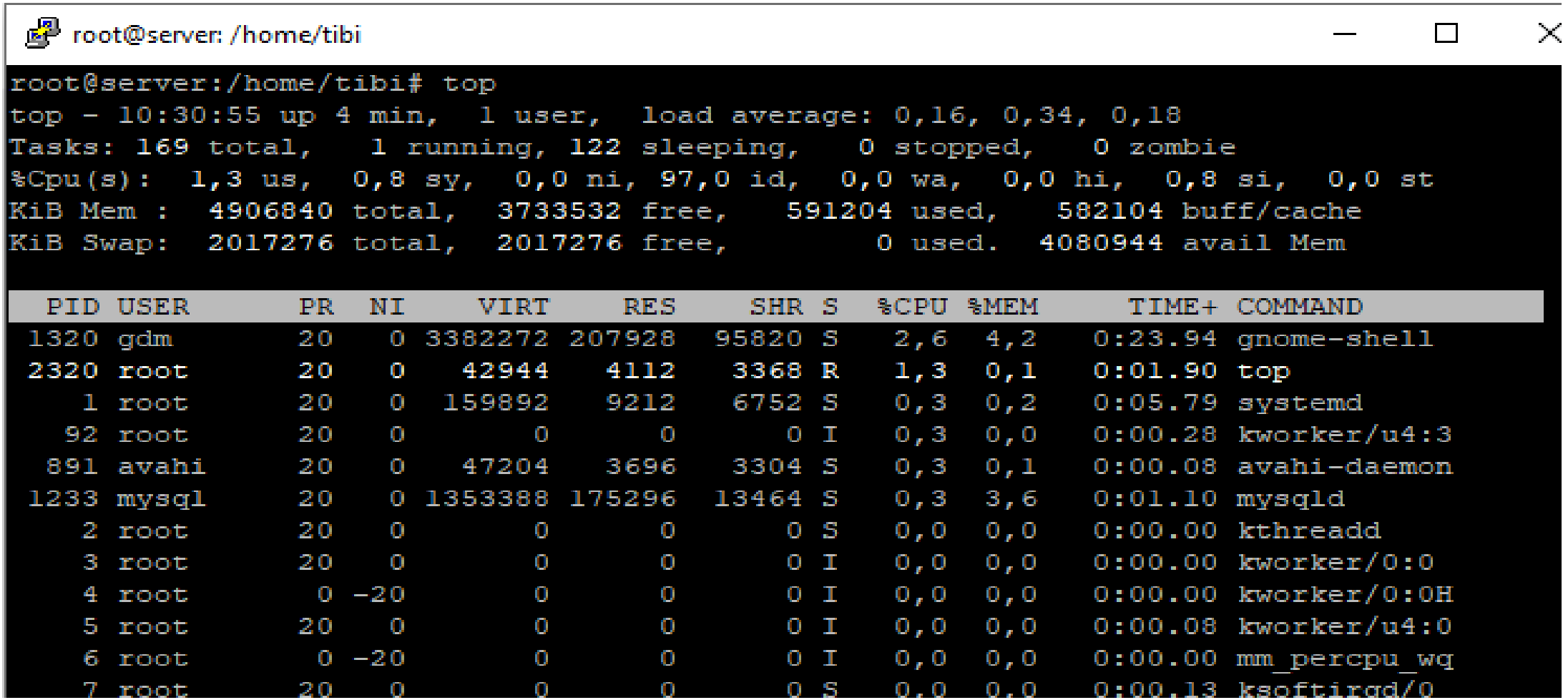
```
root@server:/home/tibi# ps -ef | more
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  1 10:26 ?        00:00:05 /sbin/init maybe-ubiquity
root           2       0  0 10:26 ?        00:00:00 [kthreadd]
root           4       2  0 10:26 ?        00:00:00 [kworker/0:0H]
root           5       2  0 10:26 ?        00:00:00 [kworker/u4:0]
root           6       2  0 10:26 ?        00:00:00 [mm_percpu_wq]
root           7       2  0 10:26 ?        00:00:00 [ksoftirqd/0]
root           8       2  0 10:26 ?        00:00:00 [rcu_sched]
root           9       2  0 10:26 ?        00:00:00 [rcu_bh]
root          10       2  0 10:26 ?        00:00:00 [migration/0]
root          11       2  0 10:26 ?        00:00:00 [watchdog/0]
root          12       2  0 10:26 ?        00:00:00 [cpuhp/0]
root          13       2  0 10:26 ?        00:00:00 [cpuhp/1]
root          14       2  0 10:26 ?        00:00:00 [watchdog/1]
root          15       2  0 10:26 ?        00:00:00 [migration/1]
```

- **PID** : processz ID, **PPID** processz prioritás

Processzus állapot információk a gyakorlatban

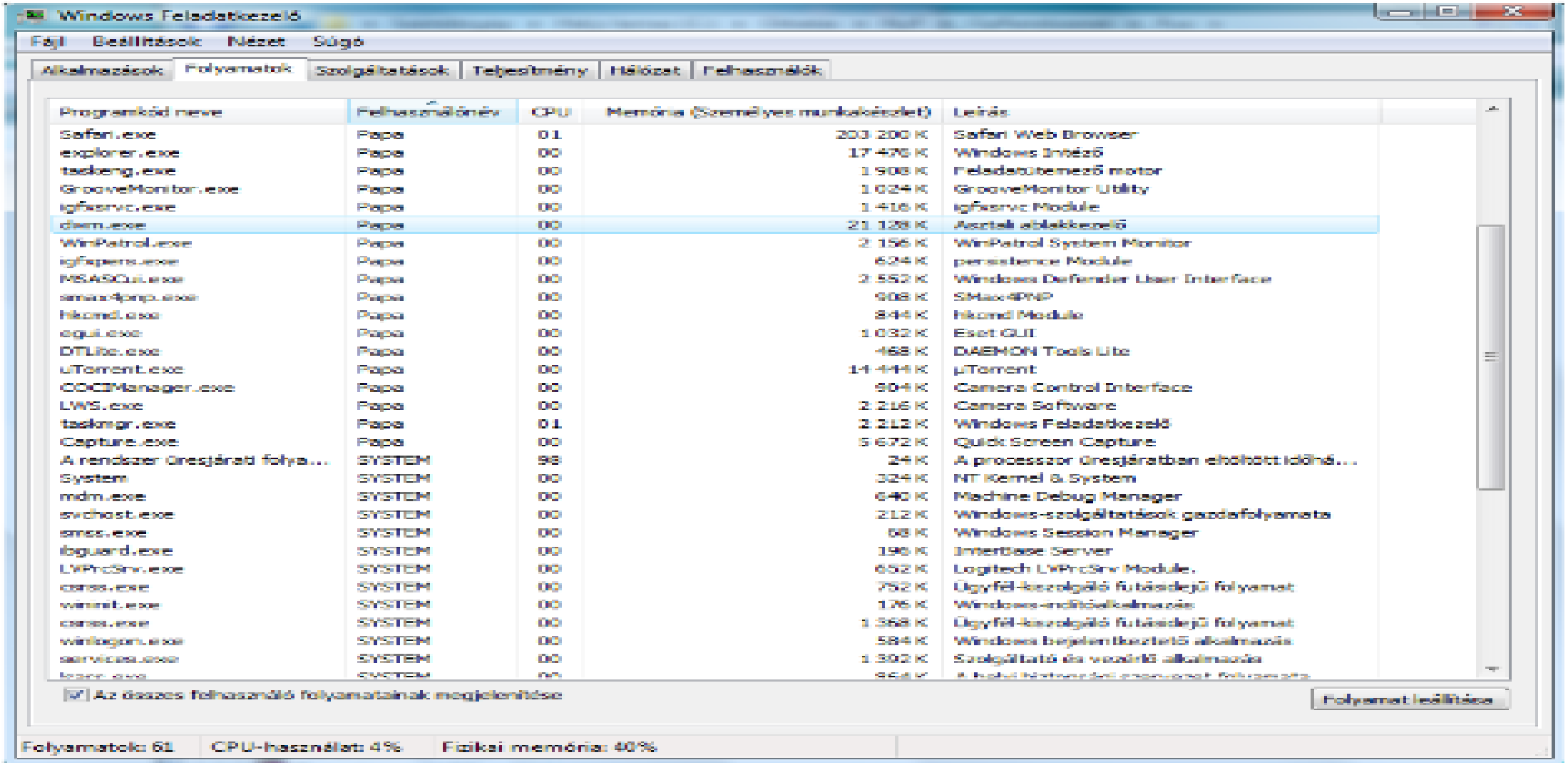
UNIX/LINUX parancs:

- # top



Processzus állapot információk a gyakorlatban

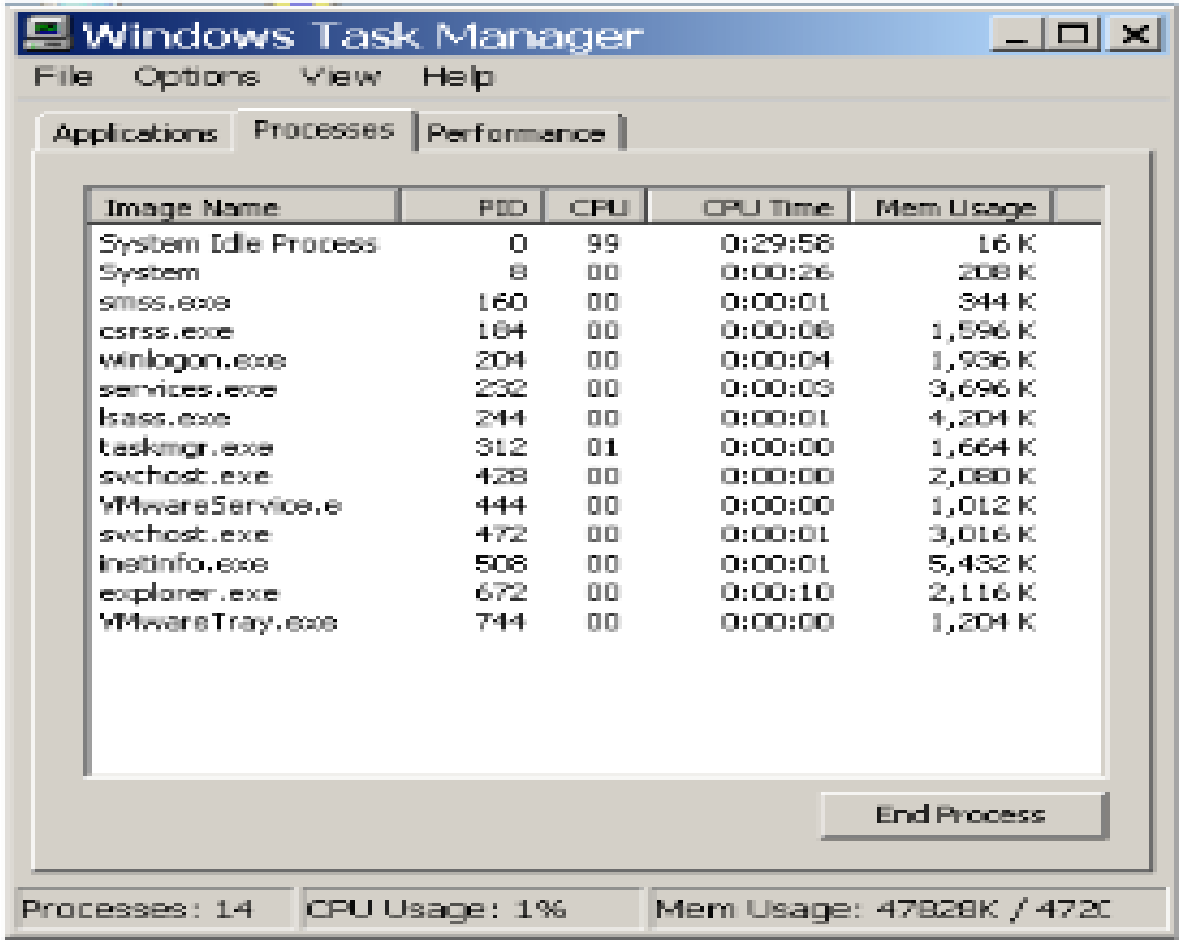
WINDOWS >> Feladatkezelő - Task manager (Ctrl+Alt+Del)

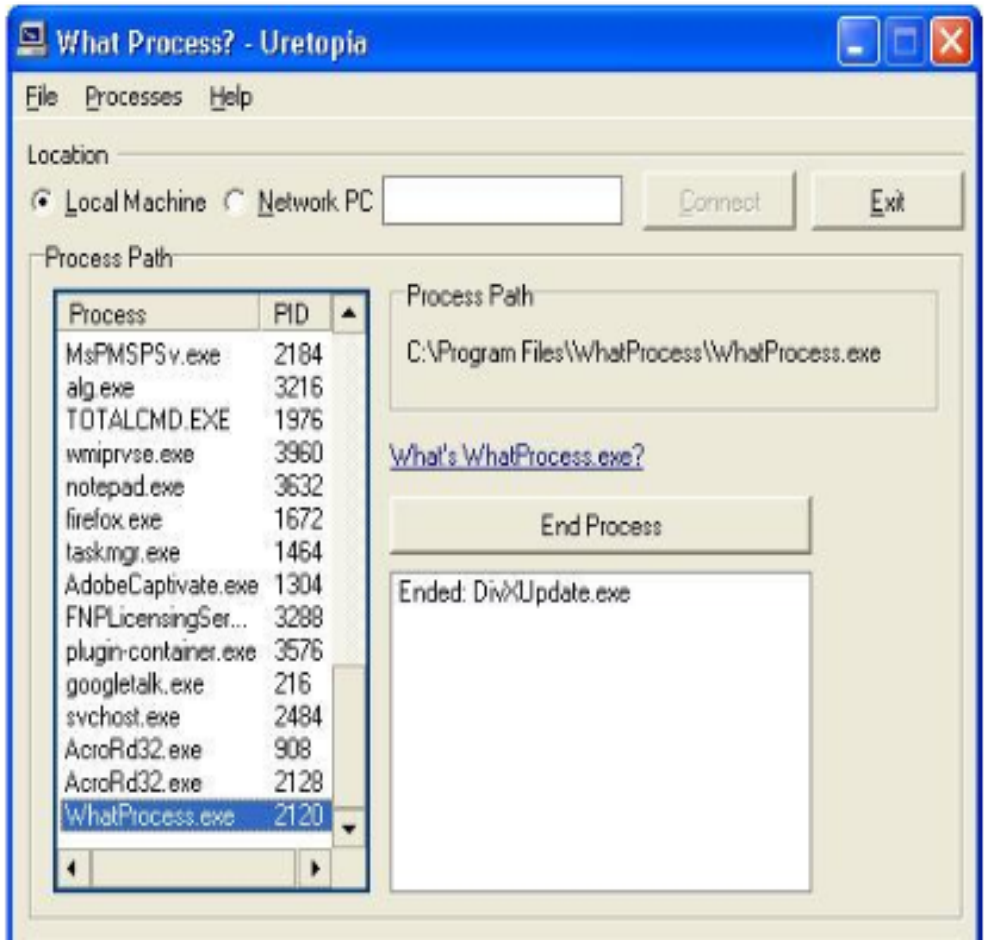


Processzus állapot információk a gyakorlatban

WINDOWS >> Feladatkezelő - Task manager (Ctrl+Alt+Del)

vagy Free process manager for Windows





4. Szálak - Thread

- **A szál a CPU kihasználtságának alapvető egysége, amely egy programszámlálóból, veremből és egy regiszterkészletből áll.** A végrehajtási szál a számítógépes program folyamatából kettő vagy több egyidejűleg futó feladatot eredményez. A szálak és a folyamatok megvalósítása operációs rendszerenként eltérő, **de a legtöbb esetben a szál egy folyamat belsejében található.**
- **Több szál létezhet ugyanazon a folyamaton belül, és megoszthatja az erőforrásokat, például a memóriát, míg a különböző folyamatok nem osztják meg ezeket az erőforrásokat.**
- Példa a szálak azonos folyamatára az automatikus helyesírás-ellenőrzés és a fájl automatikus mentése írás közben, vagy böngésző kép betöltés.
- **A szálak alapvetően olyan folyamatok, amelyek ugyanabban a memória-környezetben futnak. A szálak ugyanazokat az adatokat megoszthatják végrehajtás közben**

Szálak részei

Tehát a szálnak nincs saját memóriája, saját erőforrása csak a :

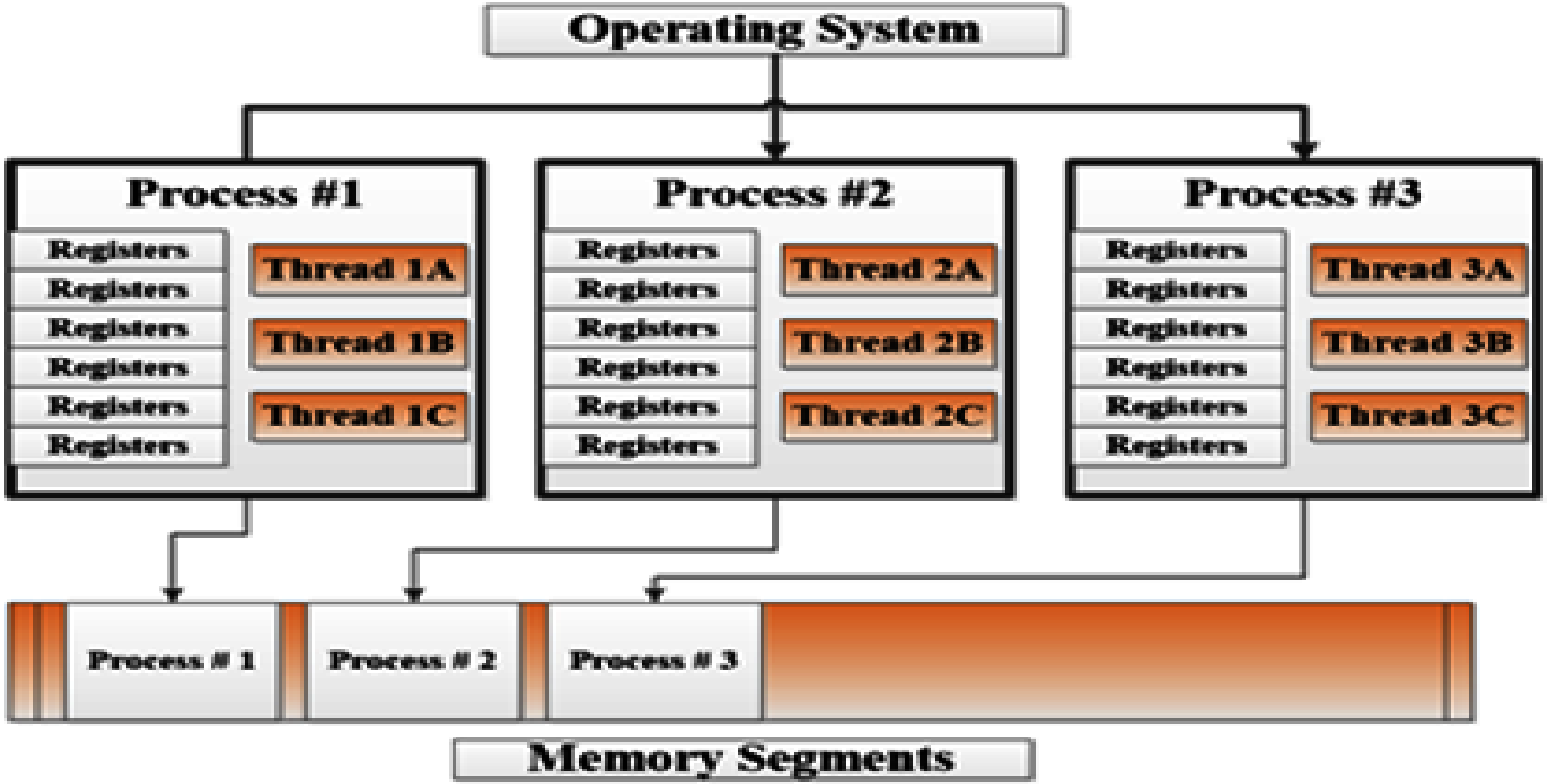
- regiszterek
- program címszámláló (PC–Program Counter)
- verem sajátja (+állapot)

Előny:

- Gyors váltást tesz lehetővé
- Osztott erőforrások (memória!)
- Párhuzamosság
- Kevesebb idő kell
  - a létrehozáshoz
  - a lezáráshoz
  - a szálak közötti váltáshoz

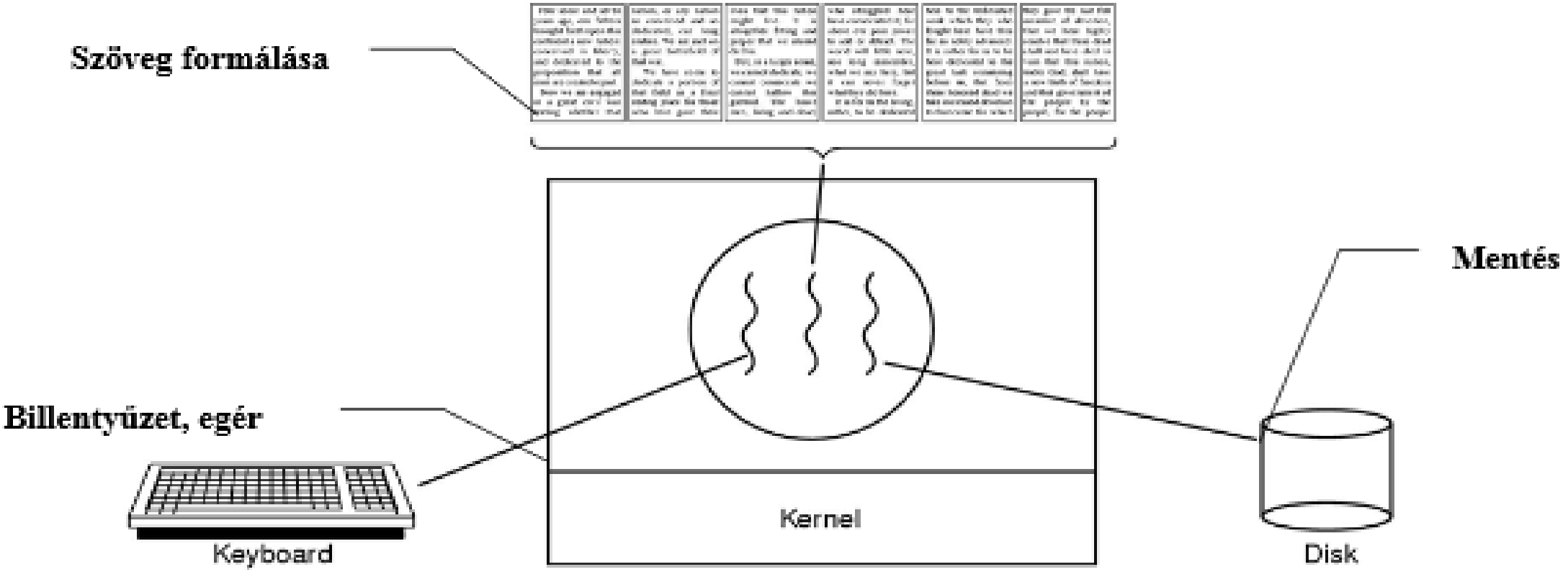
A szálak egy memóriaterületen belül vannak így könnyen kommunikálhatnak a kernel hívása nélkül

Szálak (Threads) a processzben



Példák a szálakra :

- Szövegszerkesztő, táblázatkezelő, Web-szerver, Nagy méretű adatok feldolgozása, stb.



Helyesírás ellenőrzés – még egy szál?

5. Ütemezés

Az ütemezés célja a multiprogramozott rendszerekben az azonos erőforrásokra igényt tartó folyamatok közötti választás, az erőforrás kiosztása.

Erőforrásként most elsősorban a processzorra gondolunk, így az operációs rendszer ütemezési tevékenységén azt a folyamatot értjük, amikor a processzorra várakozó („futni akaró”) folyamatok közül az operációs rendszer mindig választ egy folyamatot, amelyik a következő időszakban futhat.

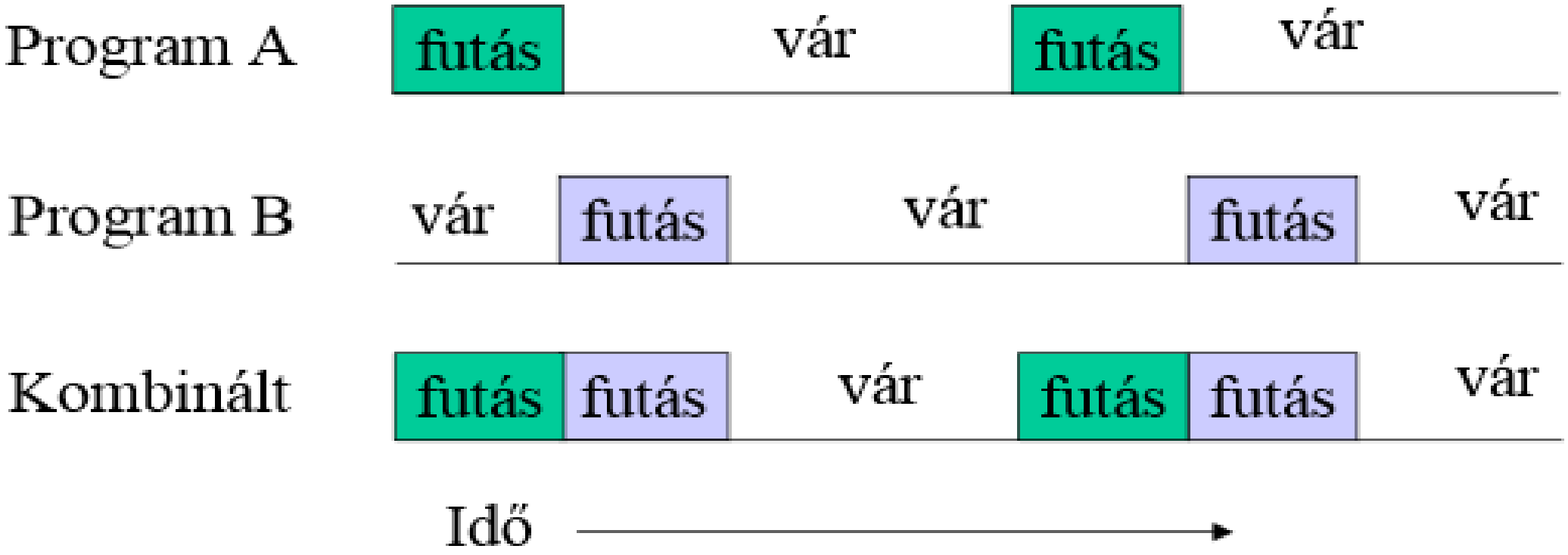
Az ütemezést a ütemező algoritmusok valósítják meg.

Ütemezés célja

Az ütemezésnek több célja is van:

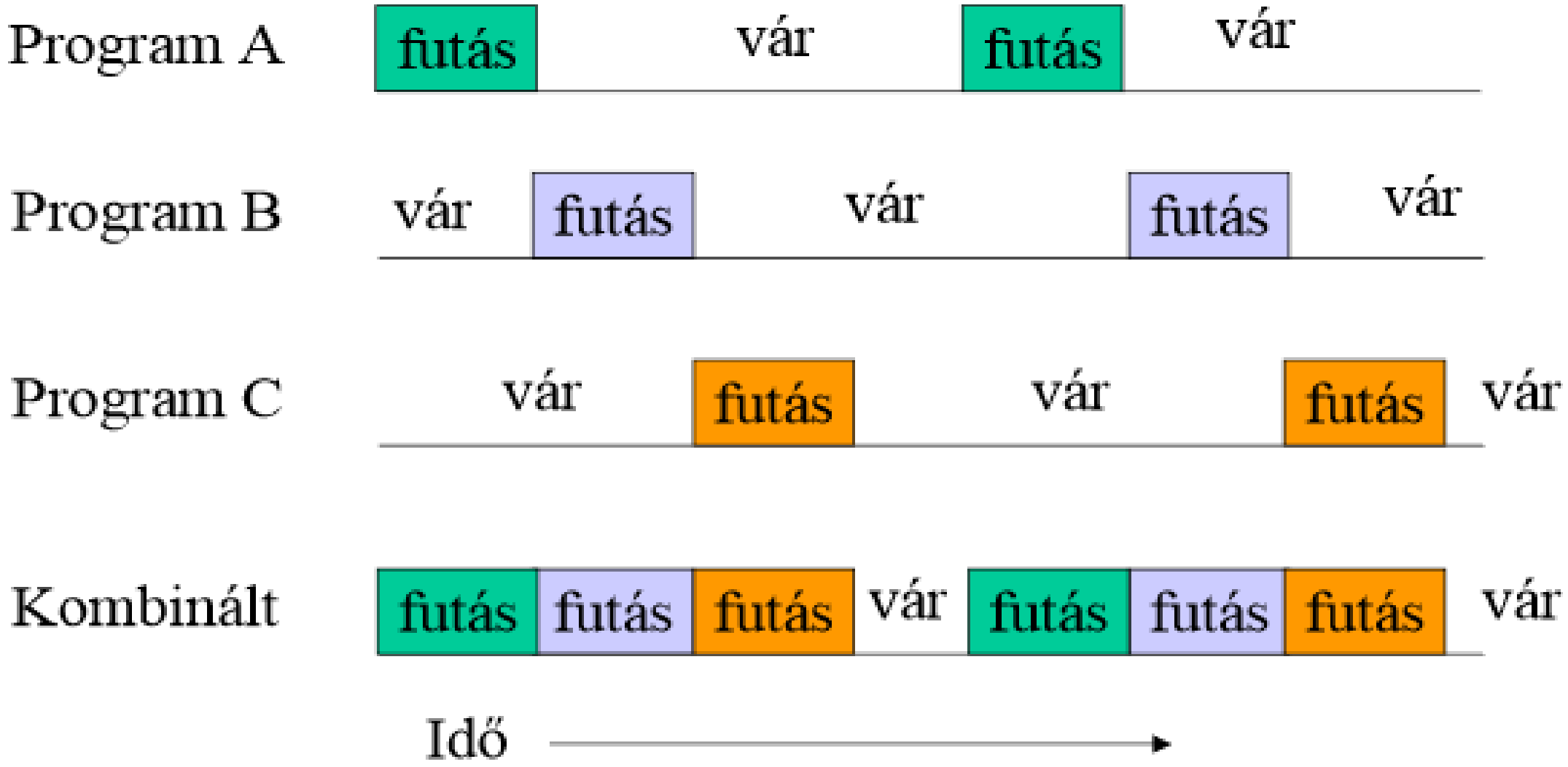
- Mindig legyen legalább egy processzus, amelyik képes és kész a processzort lefoglalni
- Minden folyamat lefusson
- Válaszidő csökkentése
- A processzor, a rendszer hatásfokának növelése

Multiprogramozás



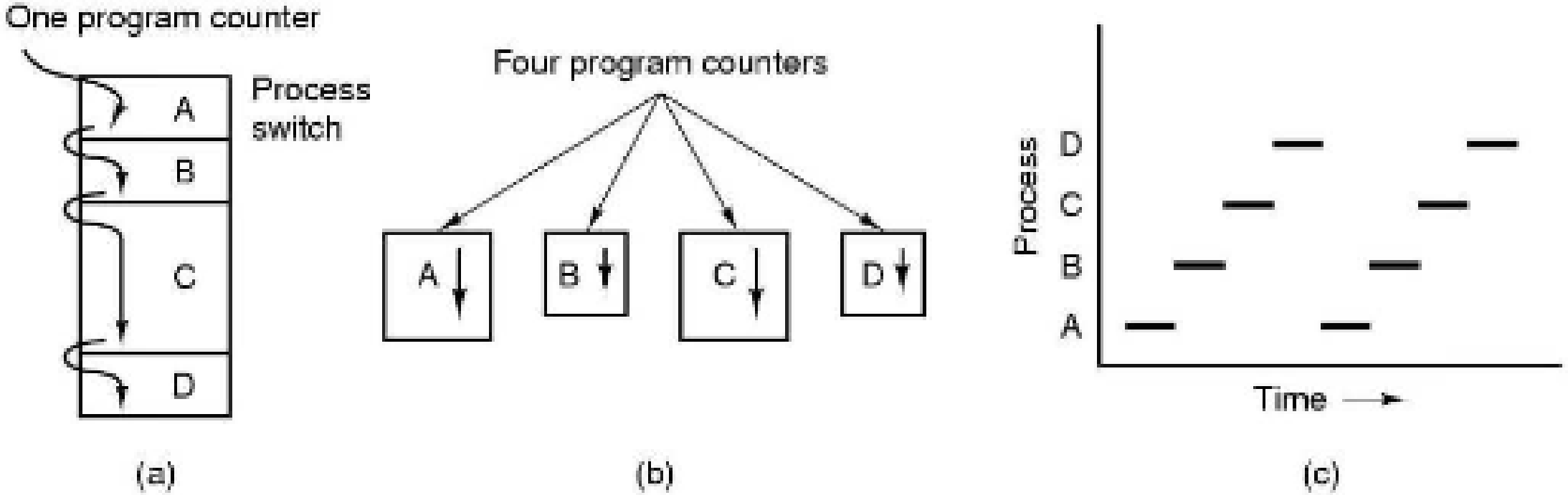
Amíg egy programnak várakoznia kell az I/O végrehajtására egy másik program futhat

Multiprogramozás



Párhuzamosság

- Valós párhuzamossághoz minden folyamatnak saját processzora (proszesszor magra) van lenne szüksége.
- A számítógép látszólag egyszerre, párhuzamosan futtatja a folyamatokat, míg valójában egymás után végzi el az egyes folyamatok parancsait (pseudoparallelism).

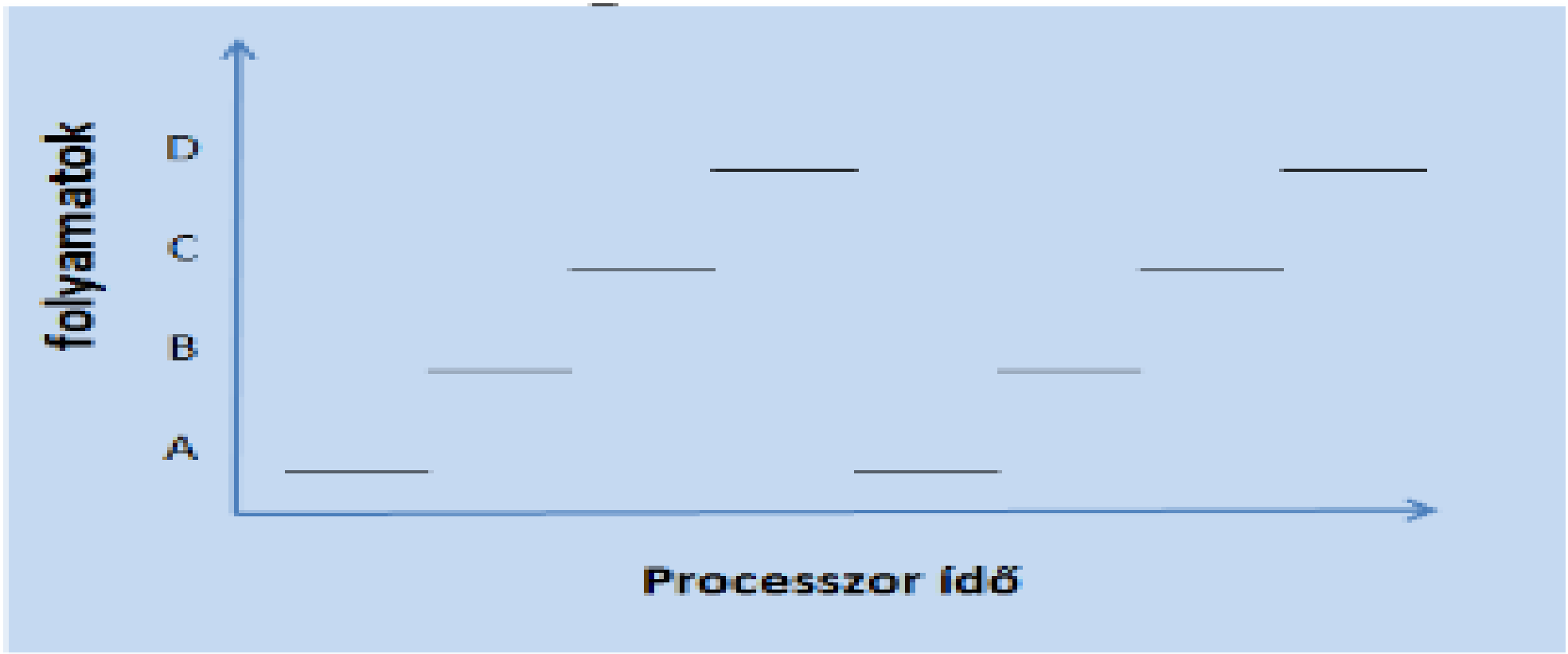


Idő osztásos rendszerek

- Olyan rendszerekben, ahol a processzor átkapcsolásokat végez a folyamatok közt, nem tehetünk előfeltételezéseket az időzítésre.
- Multiprogramozott környezetben a folyamat lefutása általában nem megjósolható vagy megismételhető időzítés szerint történik.
- A program nem feltételezheti, hogy az egyes műveletek milyen időzítés szerint történnek.
- Az ütemező **hely** (több processzor) és **idő** multiplexeléssel osztja szét a processzort vagy processzorokat a folyamatok számára.

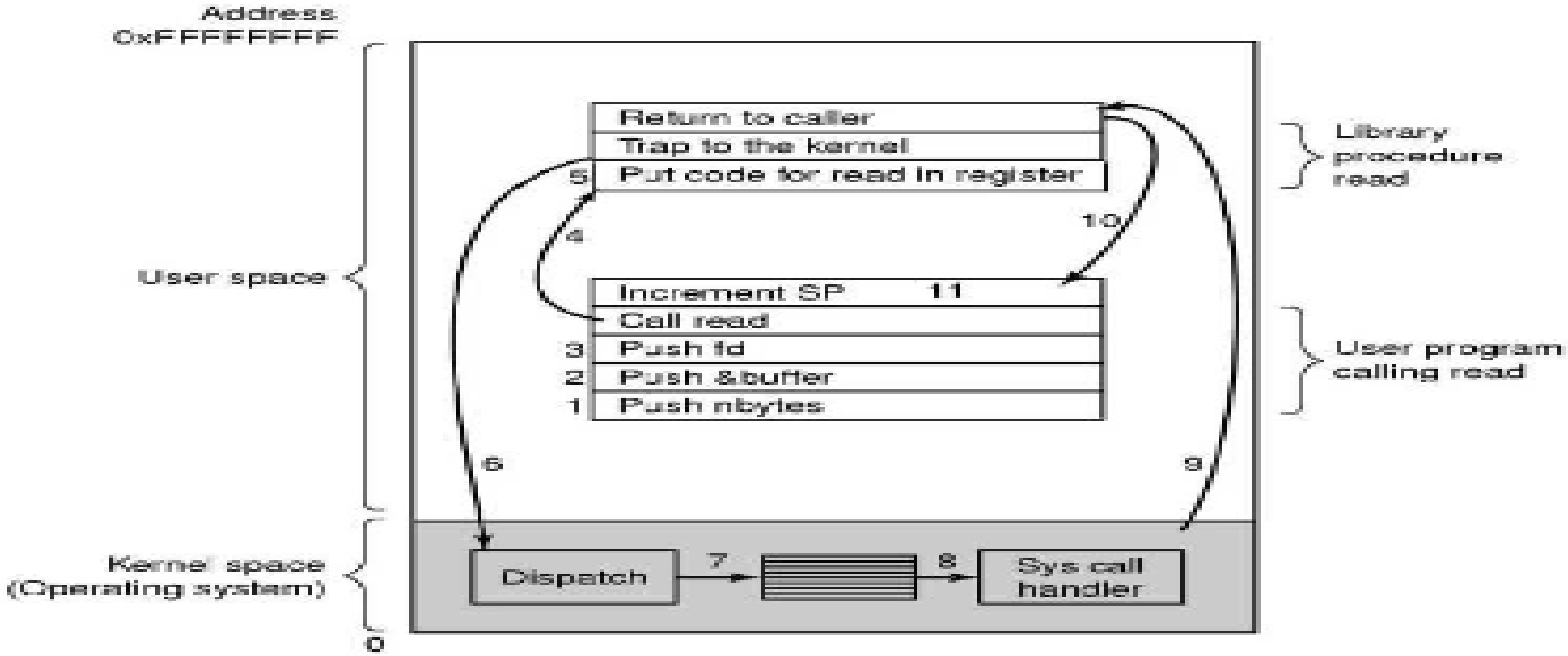
Időosztás

- Multiprogramozás használata esetén, hogy egyszerre több akár interaktív programot is kezelni lehessen
- Processzor ideje megosztott több felhasználó között



6. A rendszerhívás

A **rendszerhívás (system call**) kérés a **rendszermag felé**, a folyamat, amelynek során a program a megfelelő módon **kéri a hardver vezérlését vagy más változtatást.**



7. Folyamatok közötti kommunikáció

Kommunikáció szükséges a következő szituációkban:

1. Adatok átadása az egyik folyamatból a másiknak (**Pipelining)**
2. Közös erőforrások használata (memória, nyomtató, stb.)

Példa: Az eredmény: az a.txt fájl a nyomtatási sorba nem kerül.