

# 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:** Felhasználókhöz 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

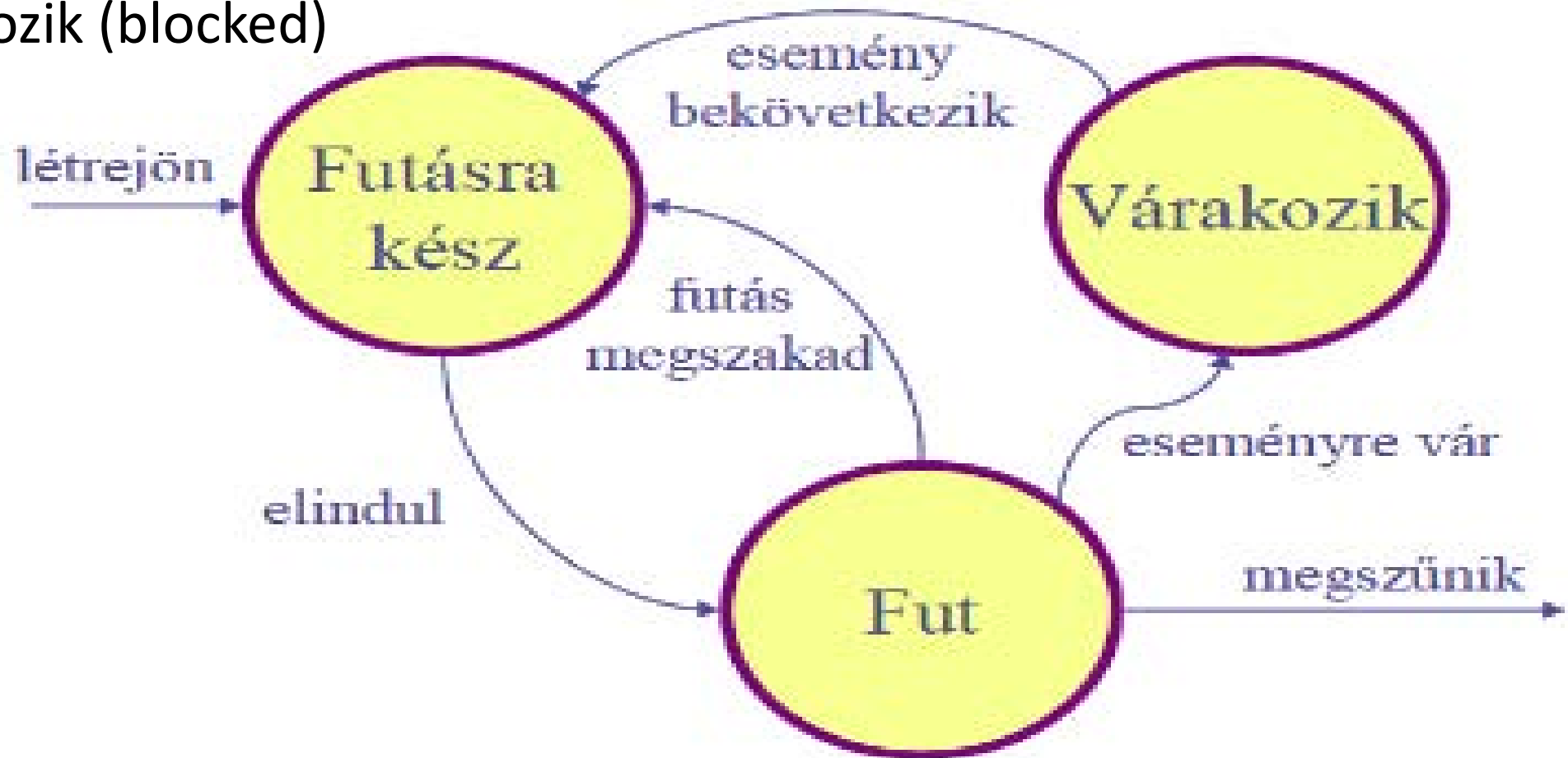
- Szabályos kilépés (önkéntes)
- Kilépés hiba miatt (önkéntes)
- Kilépés végzetes hiba miatt (önkéntelen)
- Egy másik processzus megsemmisíti (önkéntelen)

## **Processzus befejezése, gyakorlatban**

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

## 2. Folyamat állapotok (általános) - state

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



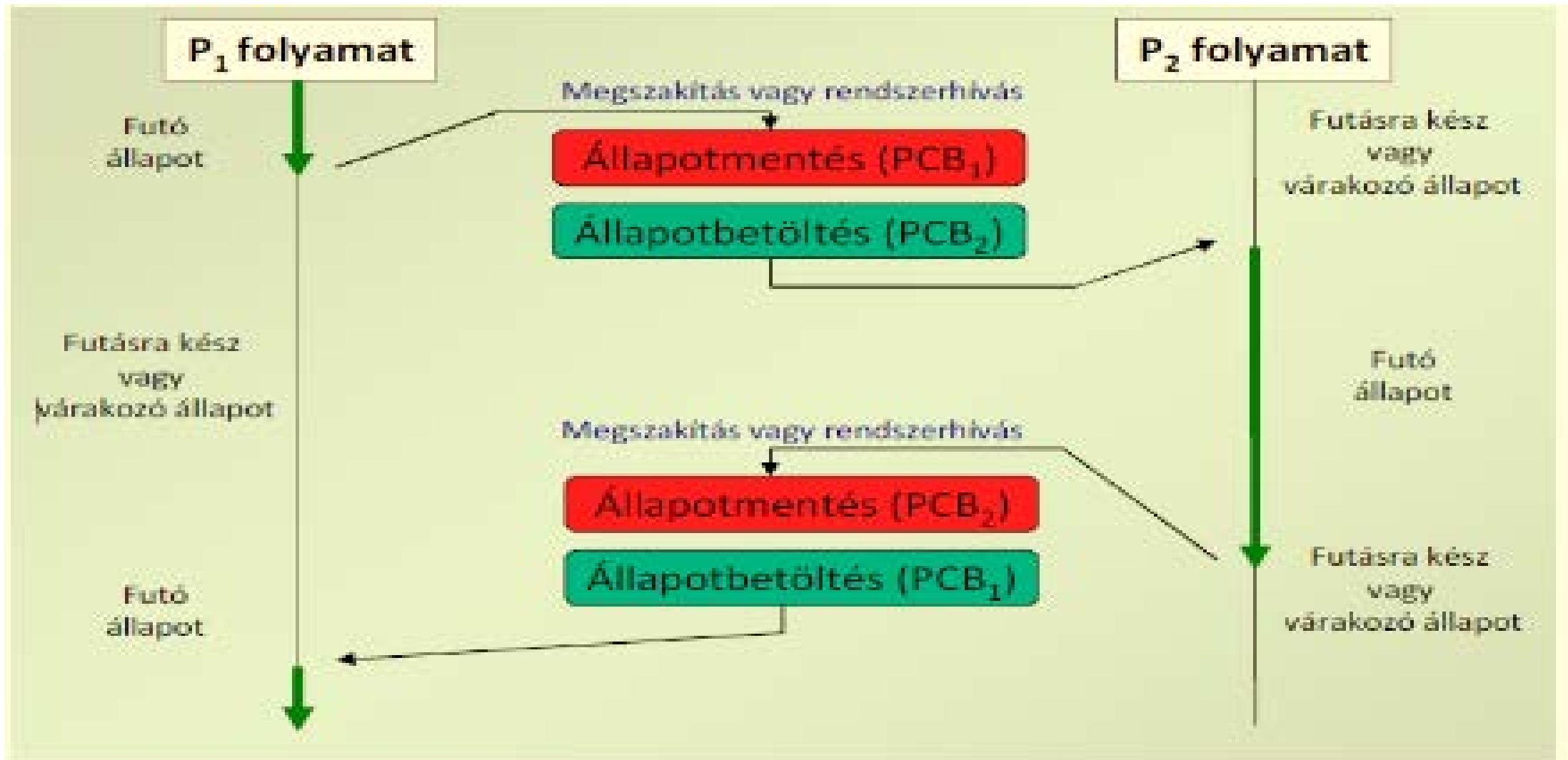


# Folyamatok állapotainak változása

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:



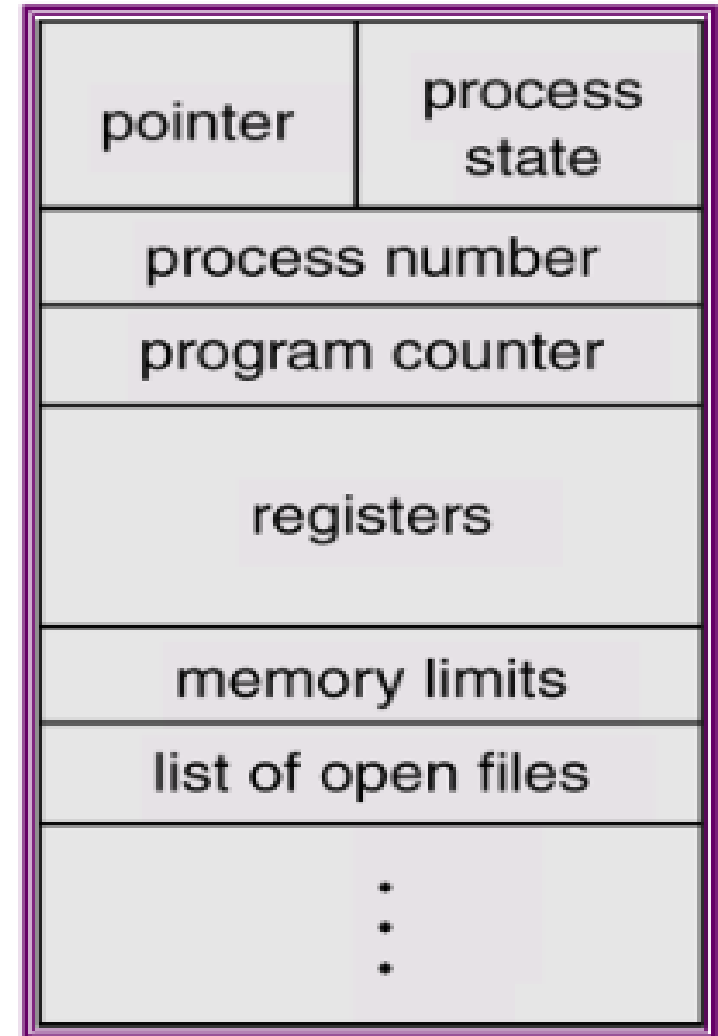
ahol PCB – Process Control Block – folyamat leíró blokk

### 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.

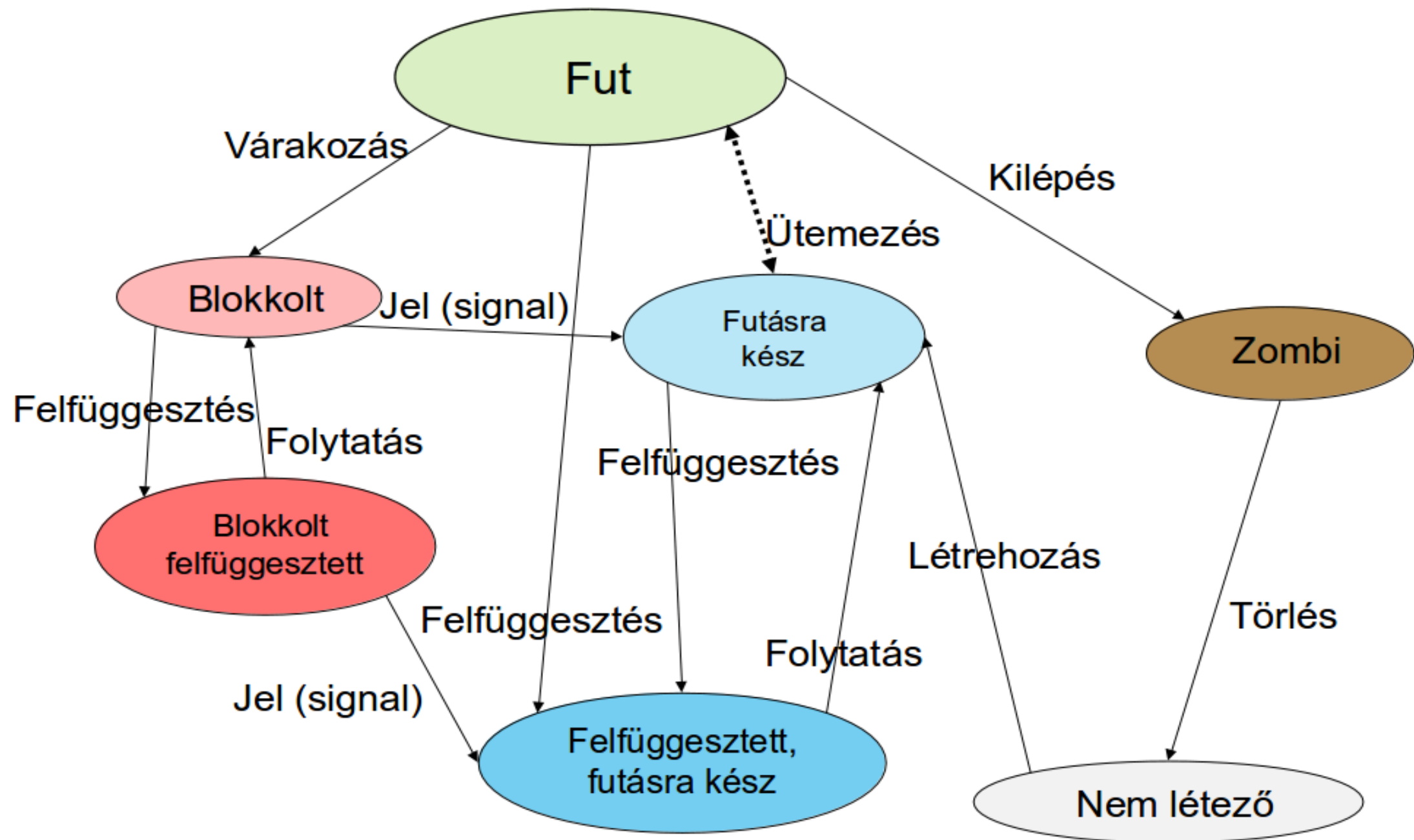
Az épp **futó (Running)** folyamatot vagy az ütemező teszi vissza egy idő után a futásra várakozók sorába,

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.



# 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

```
root@server: /home/tibi
root@server:/home/tibi# top
top - 10:30:55 up 4 min,  1 user,  load average: 0,16, 0,34, 0,18
Tasks: 169 total,  1 running, 122 sleeping,  0 stopped,  0 zombie
%Cpu(s):  1,3 us,  0,8 sy,  0,0 ni, 97,0 id,  0,0 wa,  0,0 hi,  0,8 si,  0,0 st
KiB Mem : 4906840 total, 3733532 free,  591204 used,  582104 buff/cache
KiB Swap: 2017276 total, 2017276 free,  0 used. 4080944 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 1320 gdm        20   0 3382272 207928 95820 S   2,6   4,2   0:23.94 gnome-shell
 2320 root       20   0  42944    4112   3368 R   1,3   0,1   0:01.90 top
    1 root       20   0 159892    9212   6752 S   0,3   0,2   0:05.79 systemd
   92 root       20   0      0         0      0 I   0,3   0,0   0:00.28 kworker/u4:3
  891 avahi      20   0  47204    3696   3304 S   0,3   0,1   0:00.08 avahi-daemon
 1233 mysql      20   0 1353388 175296 13464 S   0,3   3,6   0:01.10 mysqld
    2 root       20   0      0         0      0 S   0,0   0,0   0:00.00 kthreadd
    3 root       20   0      0         0      0 I   0,0   0,0   0:00.00 kworker/0:0
    4 root        0 -20      0         0      0 I   0,0   0,0   0:00.00 kworker/0:0H
    5 root       20   0      0         0      0 I   0,0   0,0   0:00.08 kworker/u4:0
    6 root        0 -20      0         0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
    7 root       20   0      0         0      0 S   0,0   0,0   0:00.13 ksoftirqd/0
```



# Processzus állapot információk a gyakorlatban

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

Windows Feladatkezelő

Fájl Beállítások Nézet Súgó

Alkalmazások Folyamatok Szolgáltatások Teljesítmény Hálózati Felhasználók

Programkód neve	Felhasználónév	CPU	Memória (Személyes munkakészlet)	Leírás
Safari.exe	Papa	0.1	200.200 K	Safari Web Browser
explorer.exe	Papa	0.0	17.476 K	Windows Intéző
taskeng.exe	Papa	0.0	1.908 K	Feladatkezelő motor
GrooveMonitor.exe	Papa	0.0	1.024 K	GrooveMonitor Utility
igfssvc.exe	Papa	0.0	1.416 K	igfssvc Module
daem.exe	Papa	0.0	21.128 K	Asztali ablakkezelő
WinPatrol.exe	Papa	0.0	2.156 K	WinPatrol System Monitor
igfssvc.exe	Papa	0.0	624 K	persistence Module
MSASQ.exe	Papa	0.0	2.552 K	Windows Defender User Interface
smss4pnp.exe	Papa	0.0	908 K	SMSS4PnP
hkcmd.exe	Papa	0.0	844 K	hkcmd Module
egui.exe	Papa	0.0	1.032 K	Eset GUI
DTLib.exe	Papa	0.0	468 K	DAEMON Tools Lite
uTorrent.exe	Papa	0.0	14.444 K	uTorrent
CCCMManager.exe	Papa	0.0	604 K	Camera Control Interface
LWS.exe	Papa	0.0	2.216 K	Camera Software
taskmgr.exe	Papa	0.1	2.212 K	Windows Feladatkezelő
Capture.exe	Papa	0.0	5.672 K	Quick Screen Capture
A rendszer órajáratát folya...	SYSTEM	98	24 K	A processzor órajáratában eltöltött időhá...
System	SYSTEM	0.0	324 K	NT Kernel & System
mdm.exe	SYSTEM	0.0	640 K	Machine Debug Manager
svchost.exe	SYSTEM	0.0	212 K	Windows-szolgáltatások gazdafolyamata
smss.exe	SYSTEM	0.0	68 K	Windows Session Manager
lgui.exe	SYSTEM	0.0	196 K	InterBase Server
LVPncSrv.exe	SYSTEM	0.0	652 K	Logitech LVPncSrv Module
csrss.exe	SYSTEM	0.0	752 K	Ügyfél-kezelő futásidejű folyamat
wininit.exe	SYSTEM	0.0	1.76 K	Windows-indítási alkalmazás
csrss.exe	SYSTEM	0.0	1.368 K	Ügyfél-kezelő futásidejű folyamat
winlogon.exe	SYSTEM	0.0	584 K	Windows bejelentkezési alkalmazás
services.exe	SYSTEM	0.0	1.302 K	Szolgáltatás és vezérlő alkalmazás
lsass.exe	SYSTEM	0.0	864 K	A hálózati hálózati szolgáltatás folyamatja

[X] Az összes felhasználói folyamatnak megjelenítése

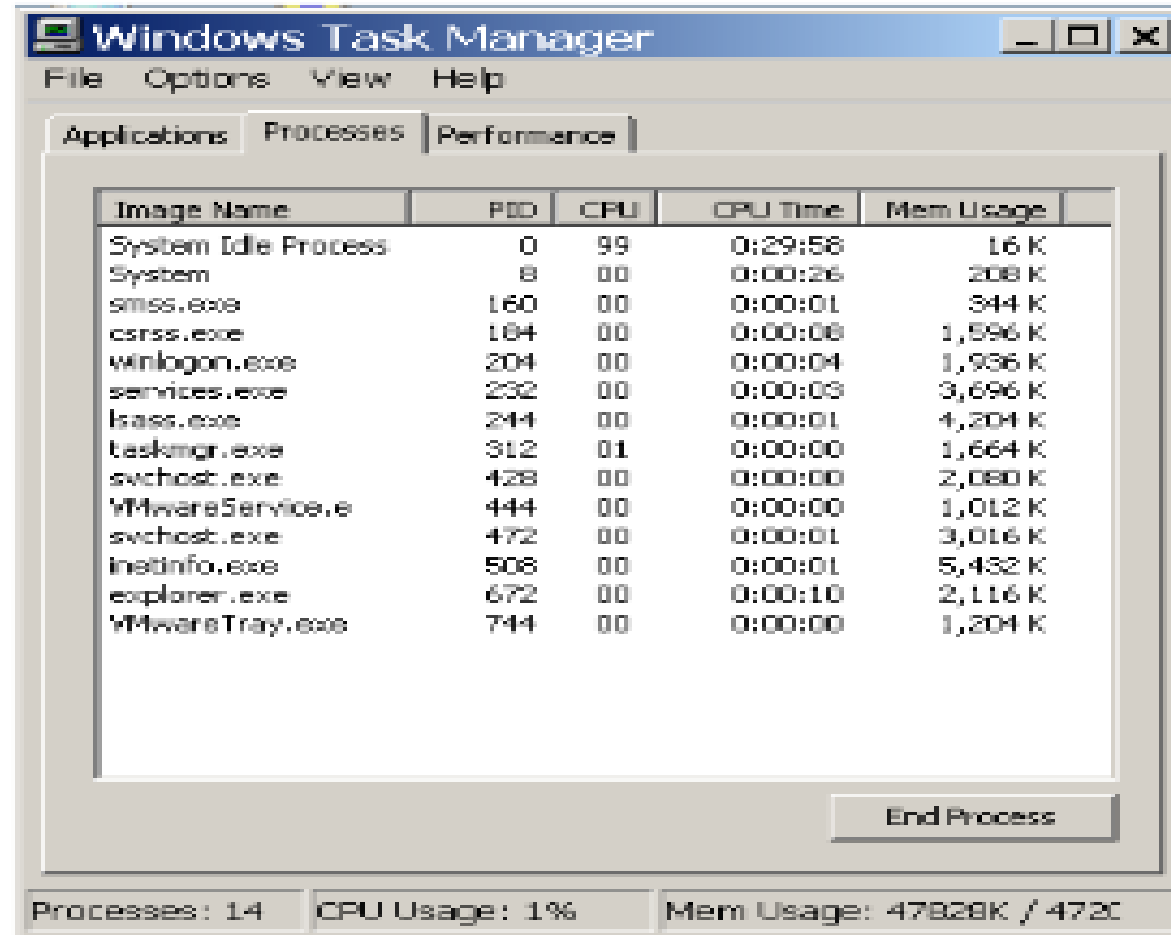
Folyamatok: 61 CPU-használat: 4% Fizikai memória: 40%

Folyamat leállítása

# Processzus állapot információk a gyakorlatban

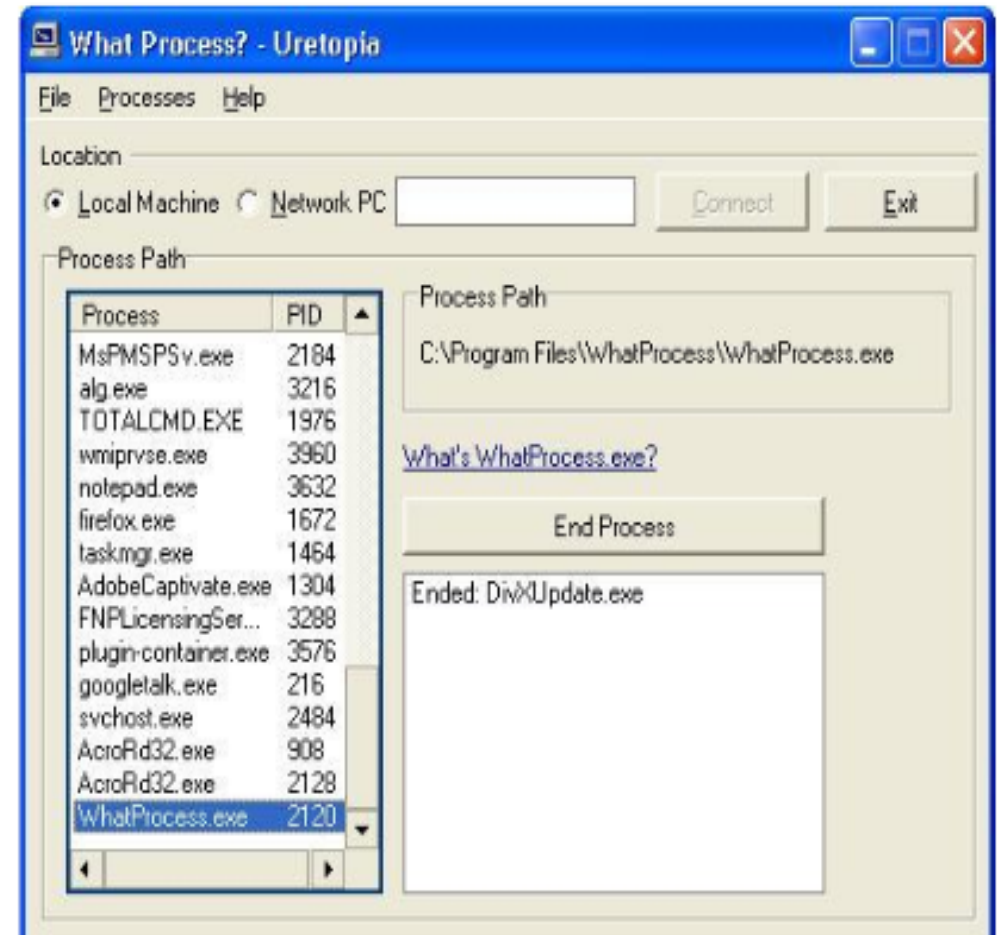
WINDOWS >> Feladatkezelő - Task manager (Ctrl+Alt+Del)

vagy Free process manager for Windows



The screenshot shows the Windows Task Manager window with the 'Processes' tab selected. It displays a list of running processes with columns for Image Name, PID, CPU, CPU Time, and Mem Usage. At the bottom, it shows system statistics: Processes: 14, CPU Usage: 1%, and Mem Usage: 47828K / 472C.

Image Name	PID	CPU	CPU Time	Mem Usage
System Idle Process	0	99	0:29:58	16 K
System	8	00	0:00:26	208 K
smss.exe	160	00	0:00:01	344 K
csrss.exe	184	00	0:00:08	1,596 K
winlogon.exe	204	00	0:00:04	1,936 K
services.exe	232	00	0:00:03	3,696 K
lsass.exe	244	00	0:00:01	4,204 K
taskmgr.exe	312	01	0:00:00	1,664 K
svchost.exe	428	00	0:00:00	2,080 K
VMwareService.e	444	00	0:00:00	1,012 K
svchost.exe	472	00	0:00:01	3,016 K
instinfo.exe	508	00	0:00:01	5,432 K
explorer.exe	672	00	0:00:10	2,116 K
VMwareTray.exe	744	00	0:00:00	1,204 K



The screenshot shows the 'What Process? - Uretopia' window. It has a 'Location' section with 'Local Machine' selected. Below it is a 'Process Path' list showing various running processes. The 'What's WhatProcess.exe?' link is highlighted. The 'End Process' button is visible, and the 'Ended: DivXUpdate.exe' message is shown in the bottom right.

Process	PID
MsFMSPSv.exe	2184
alg.exe	3216
TOTALCMD.EXE	1976
wmiprvse.exe	3960
notepad.exe	3632
firefox.exe	1672
taskmgr.exe	1464
AdobeCaptivate.exe	1304
FNPLicensingSer...	3288
plugin-container.exe	3576
googletalk.exe	216
svchost.exe	2484
AcroRd32.exe	908
AcroRd32.exe	2128
WhatProcess.exe	2120

## 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 :**

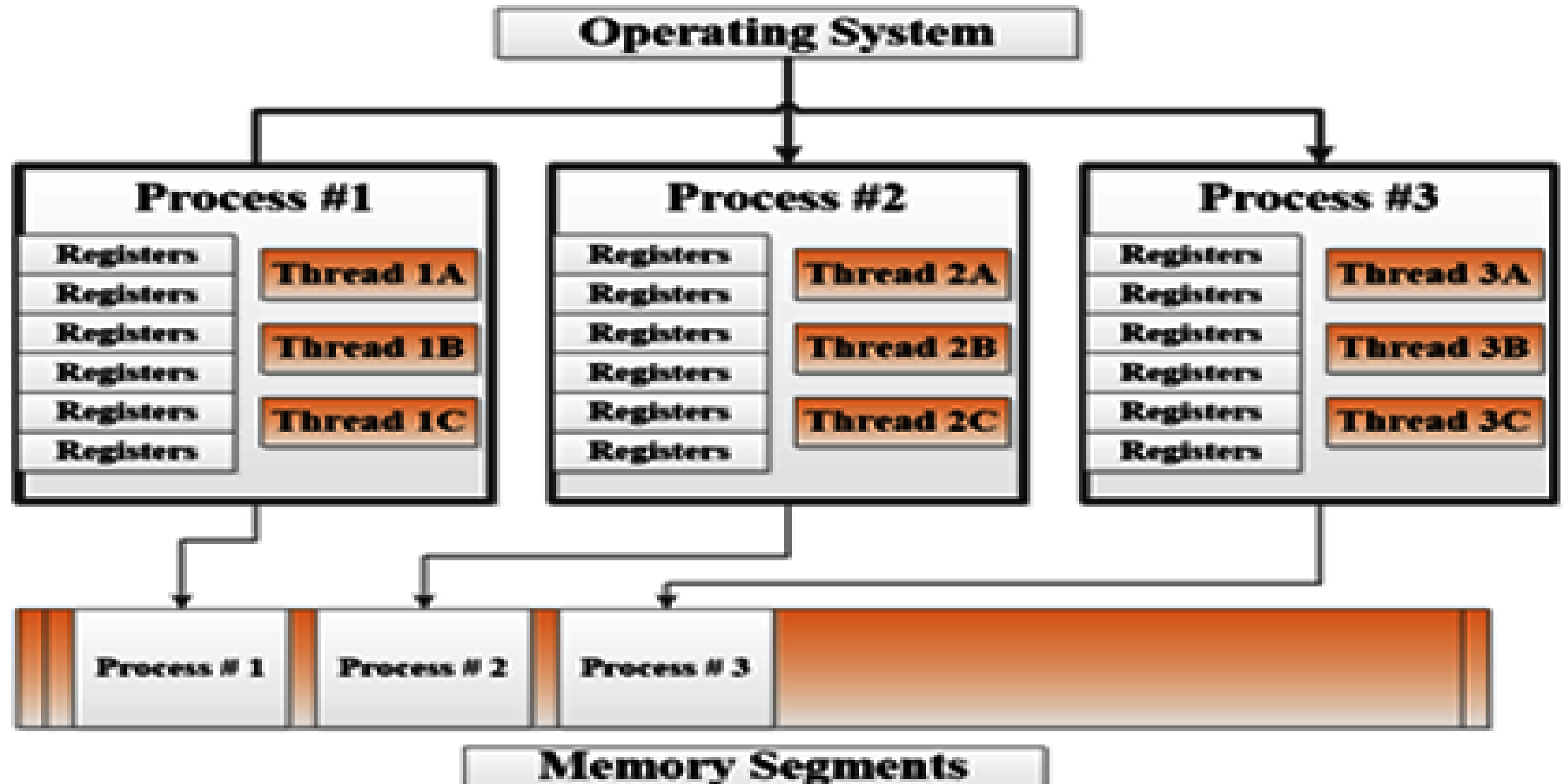
- regiszterek
- program címszámláló (PC–Program Counter)
- verem sajátja (+állapot)

## **Előny:**

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

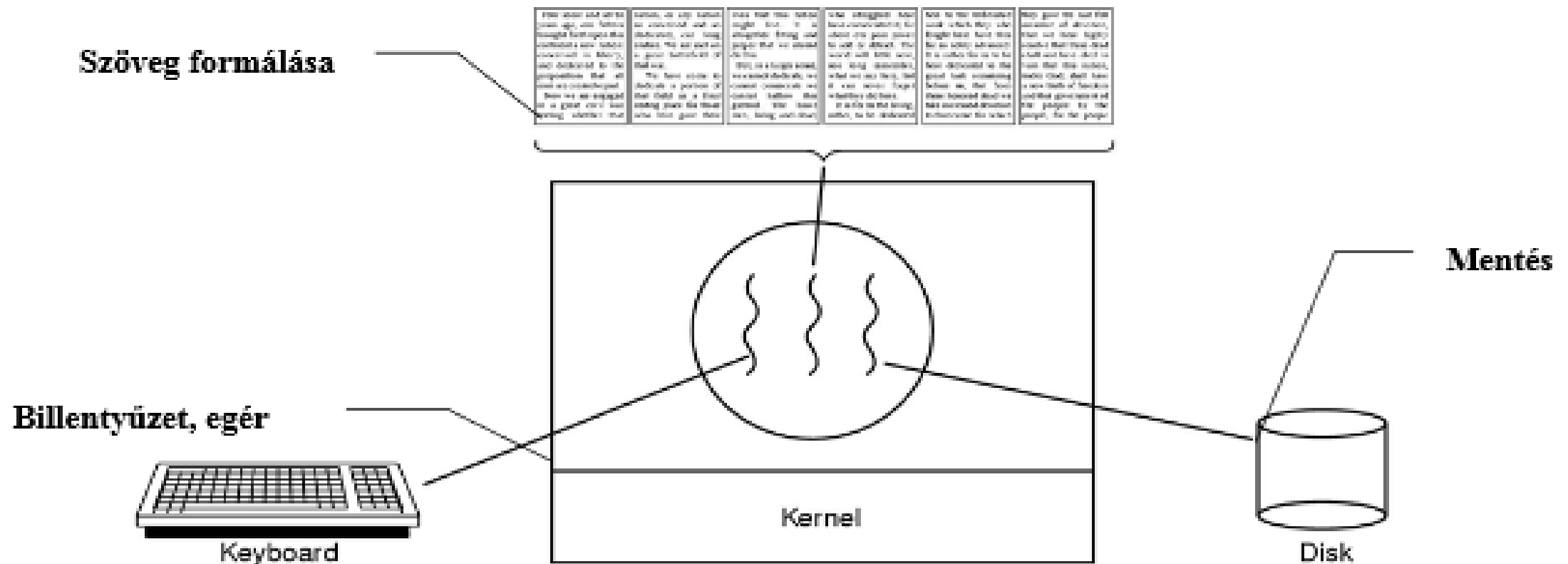
**A szálak egy memóriaterületen belül vannak így könnyen kommunikálhatnak a kernel hívása nélkül**

# 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.**

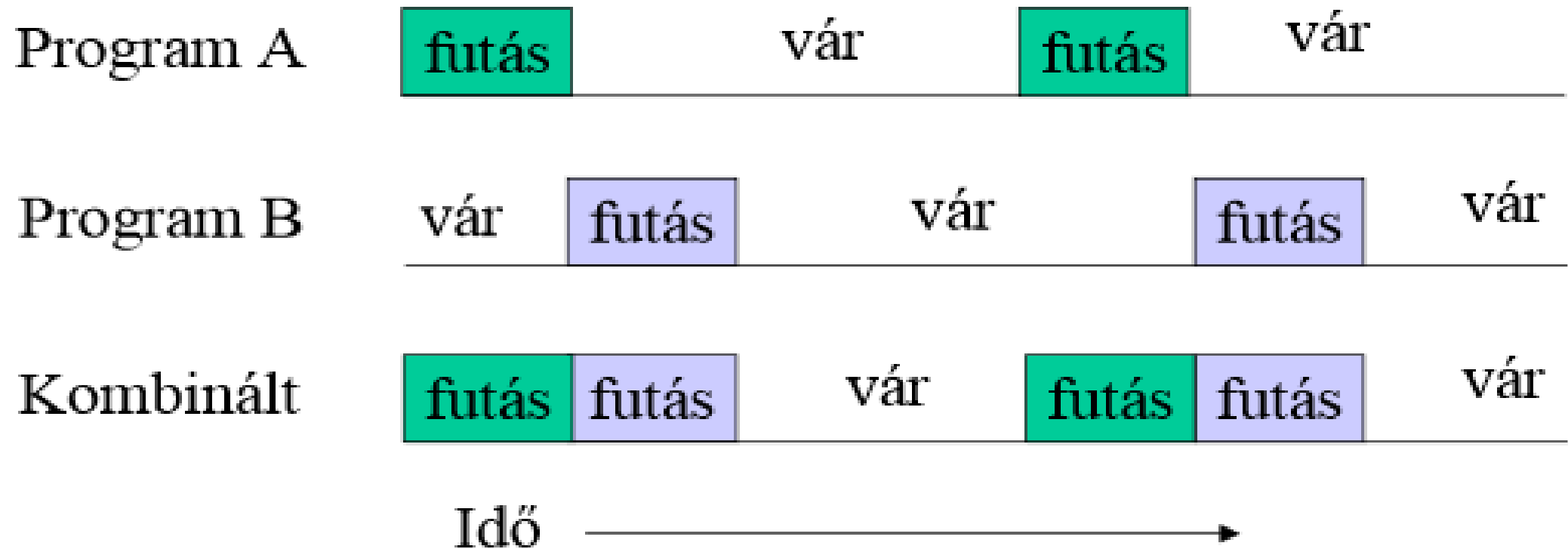
# Ütemezés célja

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

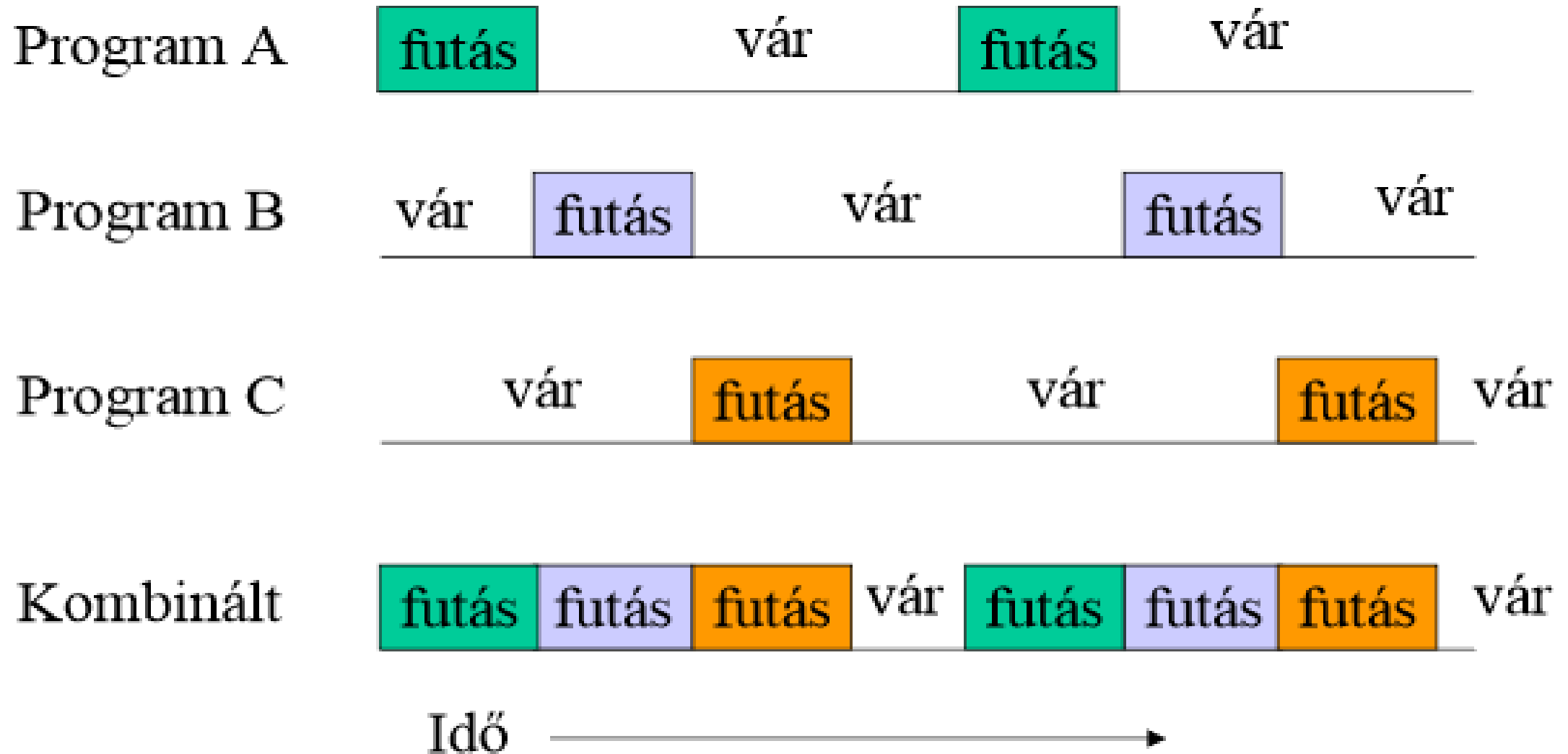


# Multiprogramozás



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

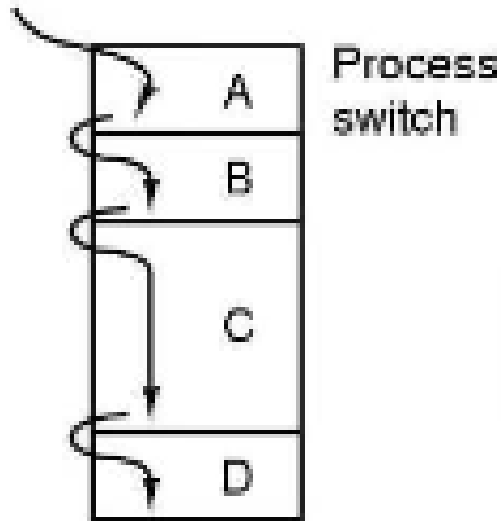
# Multiprogramozás



# Párhuzamosság

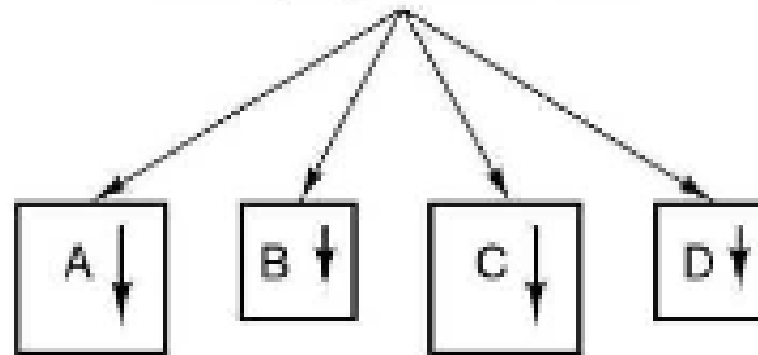
- Valós párhuzamossághoz minden folyamatnak saját processzora (prozessor magra) van lenne szüksége.
- A számítógép látszólag egyszerre, párhuzamosan futtatja a folyamatokat, míg valójában egymás után végzi el az egyes folyamatok parancsait (pseudoparallelism).

One program counter

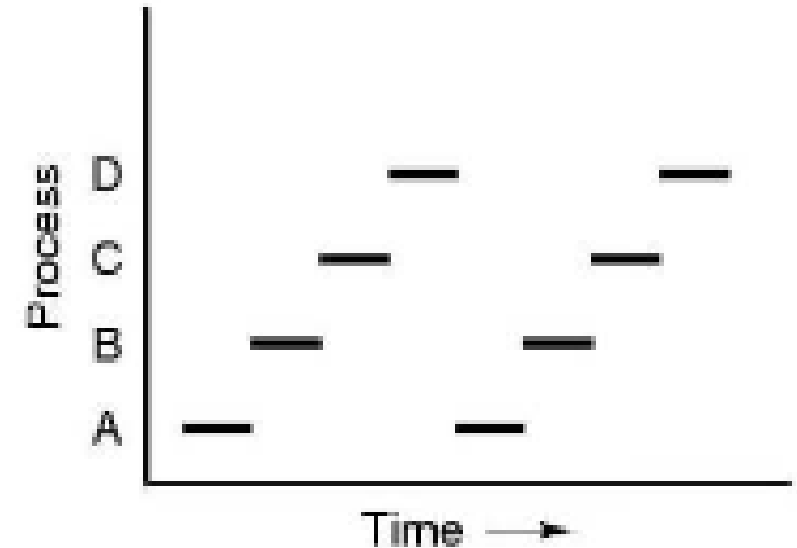


(a)

Four program counters



(b)



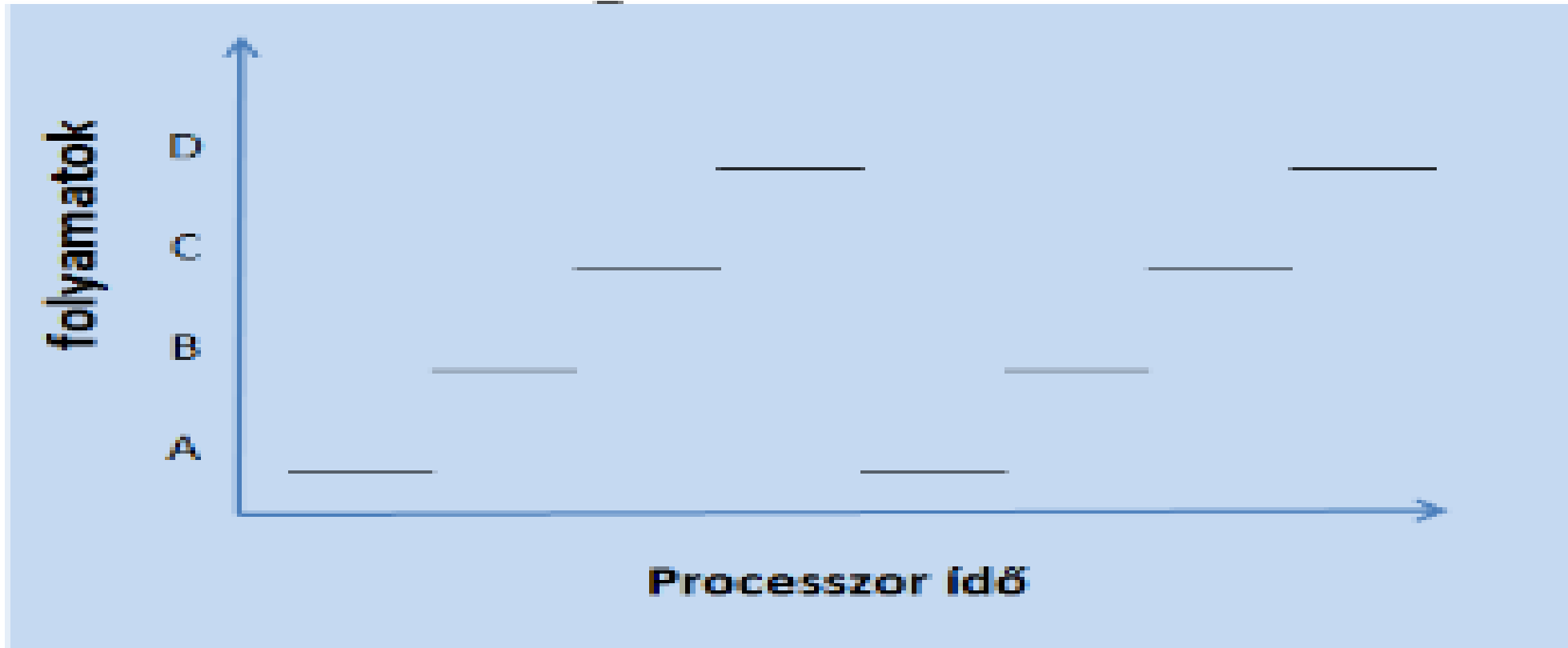
(c)

# 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.

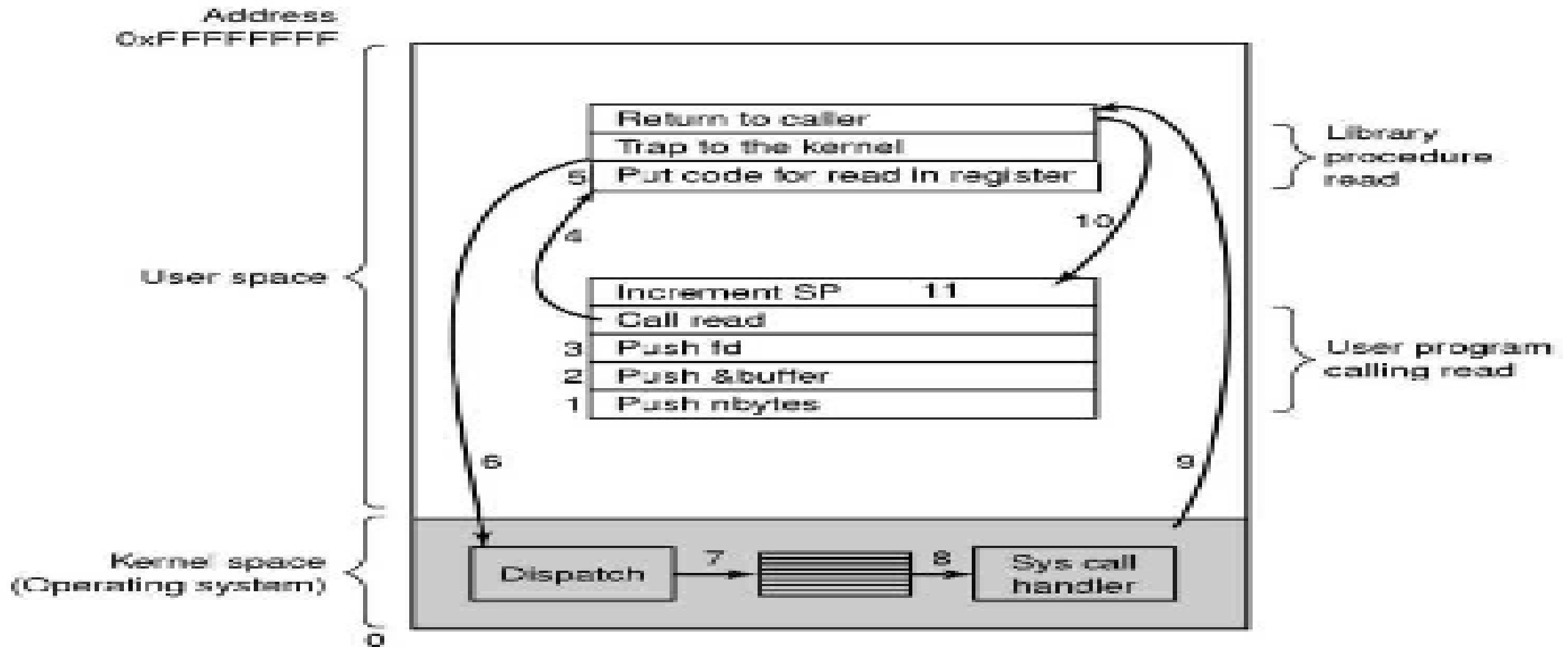
# Időosztás

- 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.

