

# Operációs rendszerek I.

## 1. Bevezetés



Várkonyiné Kóczy Annamária

Professzor

Informatika Tanszék

[\(koczya@ujs.sk\)](mailto:koczya@ujs.sk)

[varkonyi-koczy@uni-obuda.hu](mailto:varkonyi-koczy@uni-obuda.hu)

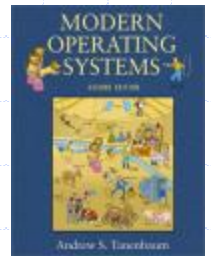
Felhasznált irodalom:

- Kóczy-Kondorosi (szerk.): Operációs rendszerek mérnöki megközelítésben
- Tanenbaum: Modern Operating Systems 2nd. Ed.
- Silberschatz, Galvin, Gagne: Operating System Concepts



# Irodalom

- **Kóczy-Kondorosi (szerk.): Operációs rendszerek mérnöki megközelítésben**  
Panem Könyvkiadó, 2000
- **Tanenbaum: Modern Operating Systems**  
Fordítása: **Operációs rendszerek**  
Műszaki Könyvkiadó, 1999
- **Silberschatz, Galvin, Gagne: Operating System Concepts**  
Wiley, 2005



# 1. Az operációs rendszerekről általában

- Az operációs rendszerek célja
- Az operációs rendszerek fogalma
- Az operációs rendszerek feladatai

# 1.1. Az operációs rendszerek célja

- felhasználók kényelme
  - egyszerű, kényelmes biztonságos használat (elsősorban kis gépeknél elsődleges, fontossága növekszik)
- hatékony gépkihasználás
  - adott idő alatt minél több program végrehajtása (nagy gépeknél elsődleges, fontossága csökken)

# 1.2. Az operációs rendszer fogalma

## DEF 1.

A számítógépen állandóan futó program

- Az a program, amely közvetlenül vezérli a gép működését (operációs rendszer magja, *kernel*).
- *Minden egyéb* alkalmazói program.

## DEF 2.

Az összes program, ami a szállítótól "operációs rendszer"-ként érkezik.

- A gyakorlati feladatokat ellátó programok.
- Minden ami a gép "általános" felhasználásához szükséges.
- Pl. grafikus felület, editor, számológép,...

## 1.3. Az operációs rendszerek feladatai

1. Végrehajtási környezet
2. Erőforrás kiosztás
3. Vezérlő program

# Operációs rendszer feladatai 1

- Végrehajtási környezet
  - Olyan környezet, ahol a felhasználók és programjaik hasznos munkát végezhetnek.
  - A számítógép hardver szolgáltatásainak bővítése
  - Elrejtí a „piszkos” részleteket, könnyű felhasználhatóságot biztosít

# Operációs rendszer feladatai 2

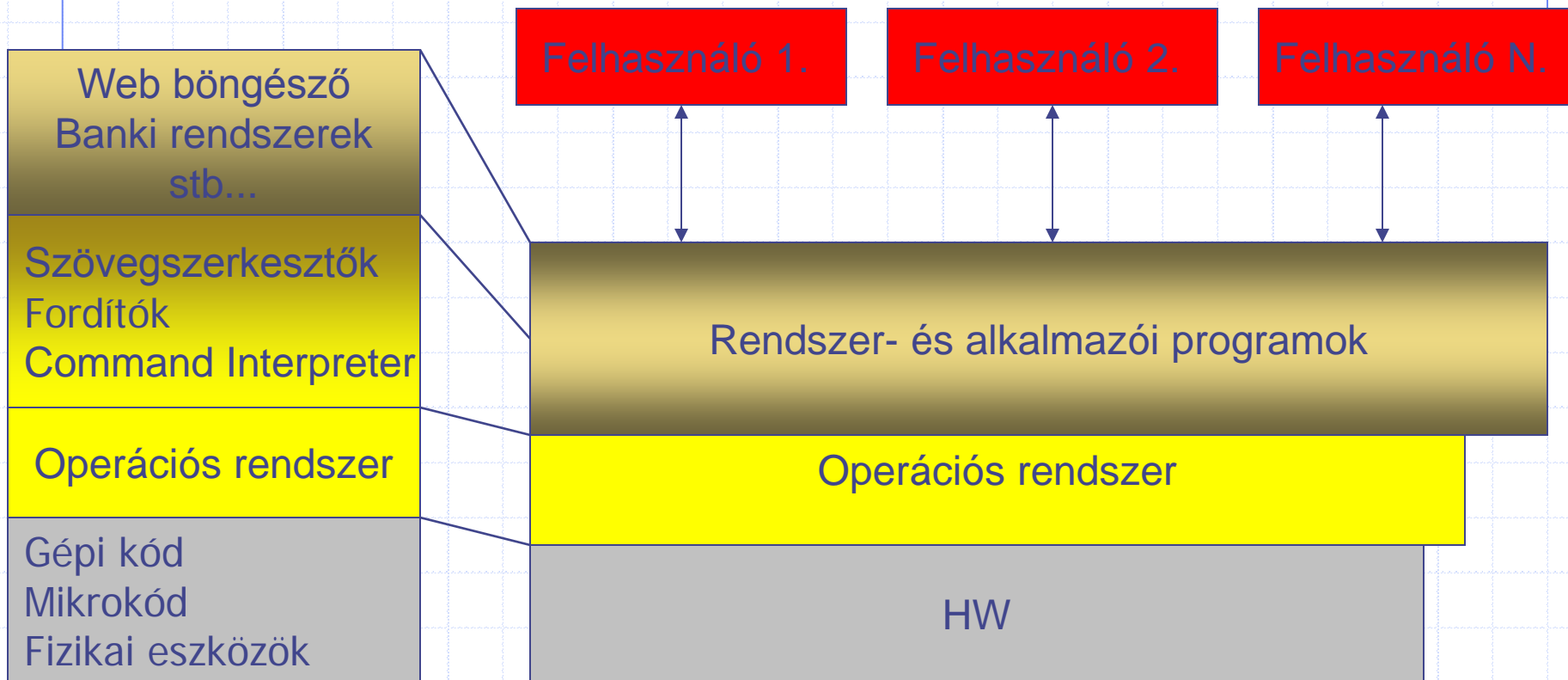
- Erőforrás kiosztás
  - Kezeli a rendszer erőforrásait
    - CPU,
    - központi tár,
    - merevlemez, stb.
  - Tulajdonságai:
    - hatékony
    - biztonságos
    - igazságos felhasználás.



# Operációs rendszer feladatai 3

- Vezérlő program
  - vezérli a felhasználói programok működését,
  - a felhasználói programok helyett vezérli a perifériák működését.

# Számítógépes rendszerek



# Statisztika

2017:

- Asztali operációs rendszerek



> 80%



~ 5%



~ 10%

- Mobil operációs rendszerek



~ 60%



~ 30%



< 2%

## 2. Az operációs rendszerek története

- Korai rendszerek
- Rendszerek a perifériás műveletek gyorsítására
- Multiprogramozott rendszerek
- Napjaink rendszerei

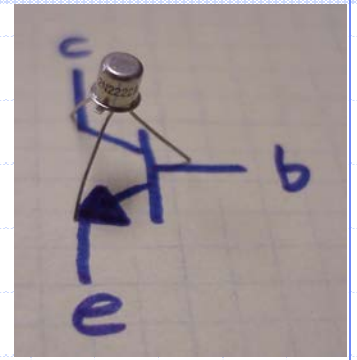
## 2.1. Korai rendszerek 1



### Nincs operációs rendszer (1945-1955)

- Hardver: nagy drága, konzolról vezérelt CPU, egyszerű perifériák (lyukkártya, nyomtató)
- Használat: programozó = operátor, kézi vezérlés, gépidő foglalás
- Mindent mindenki maga csinál, sok hiba, hibakeresés, ebből adódóan rossz kihasználtság.

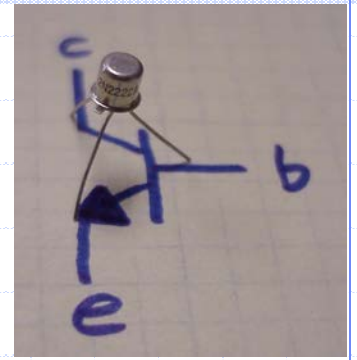
# Korai rendszerek 2



## Kötegetelt feldolgozás (1955-1965)

- Képzett operátor már van. Gyorsabb gépkezelés, de hibakeresés még nincs (core dump kiíratása).
- A programozó utasításokat mellékel a programhoz (munka, *job* leírása).
- Az operátor csoportosítja a munkákat, illetve egyes fázisait = kötegetelt feldolgozás (*batch*-ek).

# Korai rendszerek 3



## Egyszerű monitor

- A gép vezérlését egy állandóan a memóriában lévő program (a *monitor*) végzi, az operátor a perifériákat kezeli (ezek lassúak).
- a munkához rendelt vezérlő információkat a monitor értelmezi, végrehajtja.
- egy tevékenység befejezése után újra a monitor kapja meg a vezérlést, amely beolvassa a következő munkát.

# Job Control language

\$JOB - munka kezdete

\$END - munka vége

\$FORTRAN - FORTRAN fordító

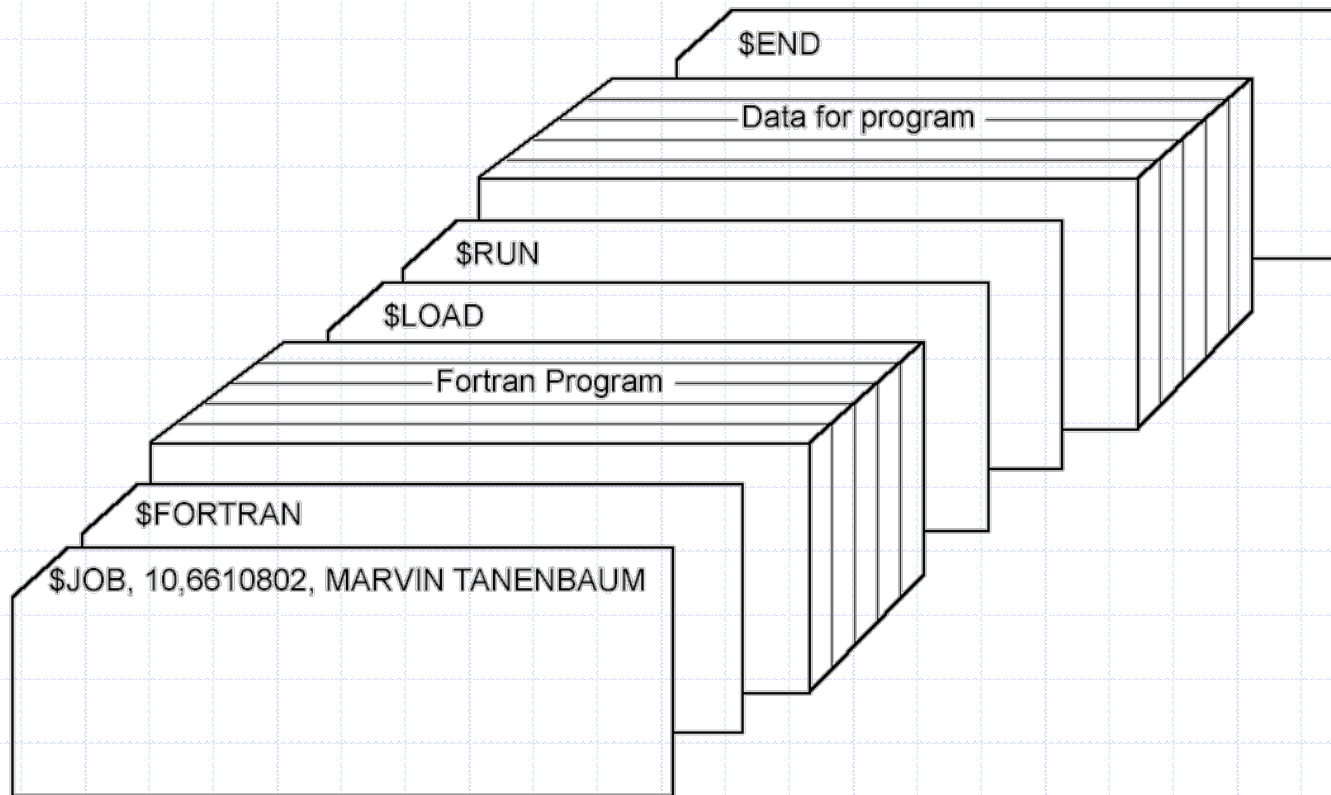
\$ASM - az assembler hívása

\$LOAD - program betöltése

\$RUN - program futtatása



# Job Control language



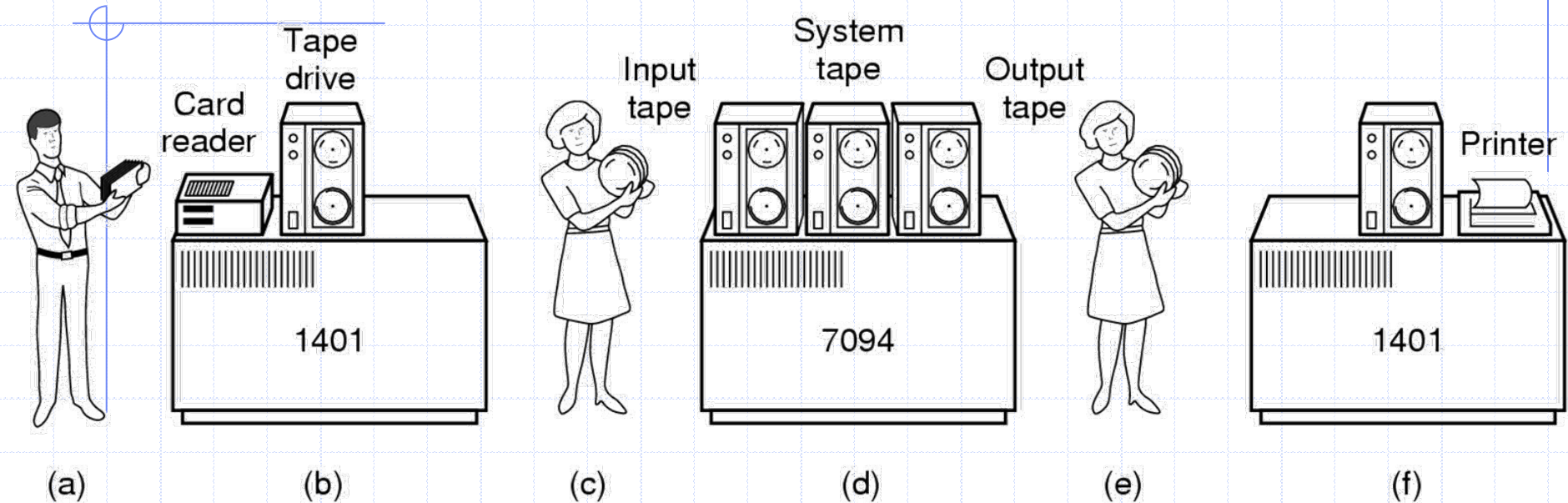
## 2.2. A perifériás műveletek gyorsítása

- A munkák között automatikus az átkapcsolás (GYORS)
- DE: a perifériák sebessége a CPU-hoz képest LASSÚ.
- *Megoldások:*
  - Off-line I/O műveletek
  - Pufferelés
  - Spooling (Simultaneous Peripheral Operations On-line)

# Off-line I/O műveletek

- Bemenet szalagról (lényegesen gyorsabb mint a kártya). Egy másik gép végzi a másolást kártyáról a szalagra. Kimenet hasonló. (+2 gép)
- Az I/O műveletek felgyorsulnak, a processzor kihasználtsága nő.
- A három számítógép párhuzamosan működik.
- Periféria-független programok, szabványos felületek.

# Off-line I/O műveletek



- a) Kártya a 1401-be
- b) Kártya másolása szalagra
- c) Szalag a 7094-hez
- d) 7094 elvégzi a számítást
- e) Szalag a 1401-be
- f) 1401 kinyomtatja az eredményt

# Pufferelés

- CPU és a periféria vezérlő egyidejűleg működnek. A beolvasás pufferbe megy. Az adatfeldolgozás és az I/O művelet átlapoltan történik.
- Pufferek mérete növekszik (blokkos átvitel).
- Csak akkor jelent hosszútávú sebesség növekedést, ha az I/O és a CPU műveletek kb. azonos időt vesznek igénybe (ami általában nem igaz)

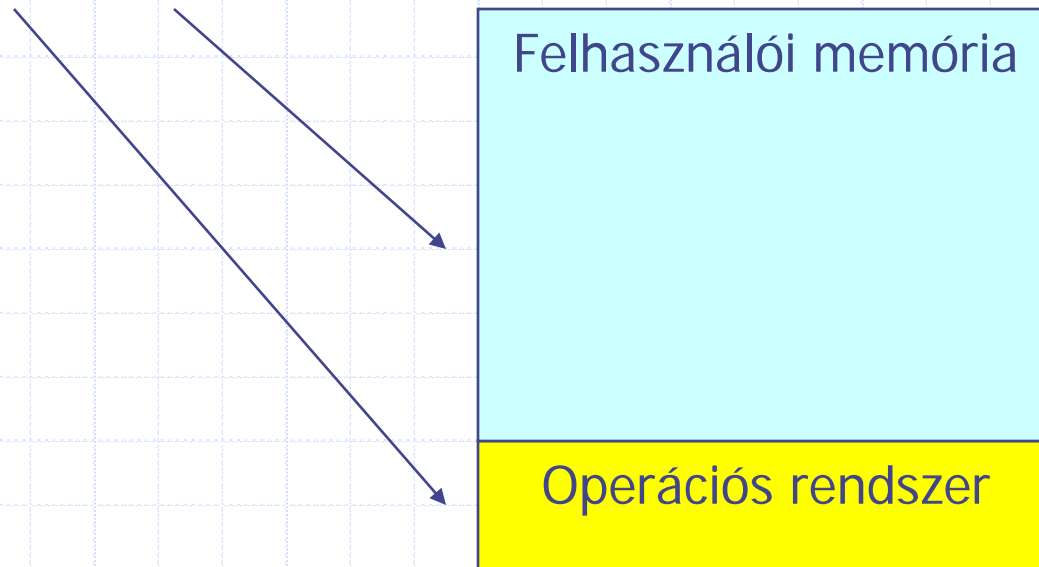
# Spooling

Nagy kapacitású, gyors, véletlen hozzáférésű perifériák kialakulása. → Spooling  
(Simultaneous Peripheral Operations On-line)

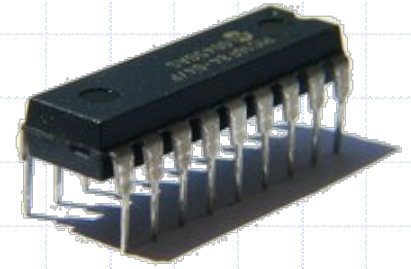
- A lassú perifériák adatait a feldolgozásig puffer helyett mágneslemezen tárolják.
- Különböző munkák perifériás és feldolgozási műveletei átlapolódhatnak.
- A lemez véletlen hozzáférése lehetővé teszi a munkák sorrendjének megválasztását.

# Kötegeltek rendszerek memória- kiosztása

Memória partíciók



## 2.3. Multiprogramozás



- 1960-1980
- A CPU sebessége megnőtt (IC-k)  
↓
- érdemes kihasználni az időt, amíg egy munka a lemezről olvasás eredményére vár  
↓
- erre az időre másik program működhet.  
↓

*Az OS egyszerre több munkát futtat.*



# A multiprogramozás lépései

- A rendszer nyilvántartja és tárolja a futtatandó munkákat.
- A kiválasztott munka addig fut, amíg várakozni nem kényszerül.
- Az OS feljegyzi a várakozás okát, majd kiválaszt egy másik futni képes munkát és azt elindítja.
- Ha a félbehagyott munka várakozási feltételei teljesülnek, akkor azt alkalmasint elindítja.

Napjainkban a legtöbb rendszer a *multiprogramozás* elvét alkalmazza.

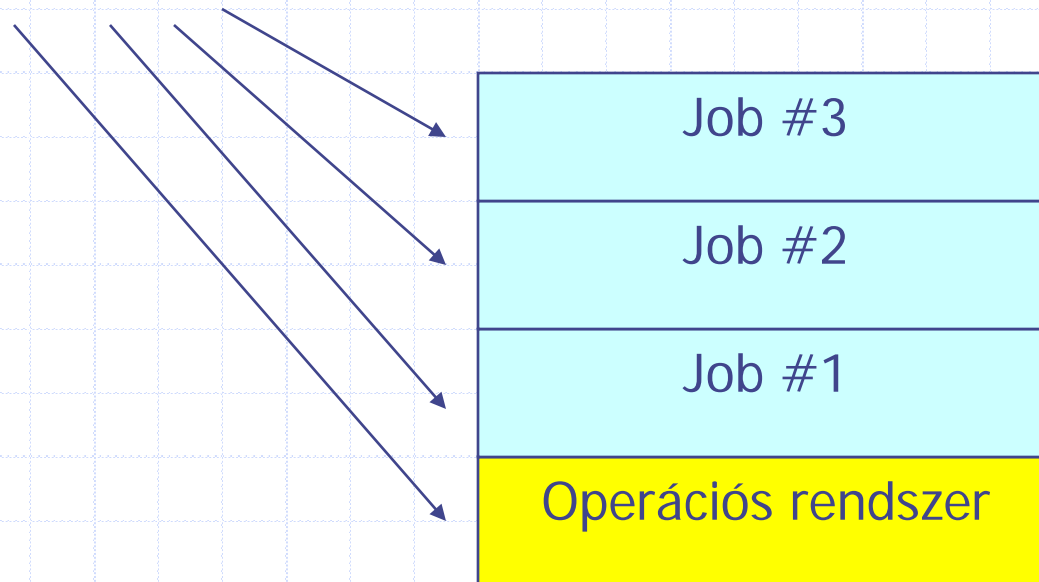
# Multiprogramozás

Felvetődő problémák:

- Az átkapcsoláshoz több program van a tárban: *tárgazdálkodás*.
- Egy időben több futásra kész program: *CPU ütemezés*
- A gépi erőforrások felhasználásának koordinációja. *Allokáció, holtpont kezelése*.
- *Védelmi mechanizmusok*, hogy a programok ne zavarják egymást és az OS-t.

# Multiprogramozott rendszer memória-kiosztása

Memória partíciók



## 2.4. Napjaink rendszerei



- 1980- (*főleg személyi számítógépek*)
- Korszerű kötegelt rendszerek
- Időosztásos rendszerek
- Elosztott operációs rendszerek
- Valós idejű rendszerek
- Beágyazott operációs rendszerek

# Korszerű kötegelt rendszerek

*Csak nagy számítógépeken !!*

- A gyors véletlen-hozzáférésű mágneslemezek miatt megszűnt a kötegelt rendszerek szükségessége.
- A jelenlegi rendszerekben a programok futásába nem lehet interaktívan beavatkozni, csak előre összeállított munkák futnak.
- Hátrányok:
  - átfutási idő hosszú. (Egy munkának a rendszerbe adása és az eredmény közti idő.)
  - A programok csak statikusan, az eredmények alapján fejleszthetők, használhatók. Az átfutási idő megnehezíti a programokkal történő kísérletezést.

# Időosztásos rendszerek

- A időosztásos rendszerek (timesharing, multitask systems) *közvetlen interaktív kommunikációt* biztosít a felhasználó és a programja, ill. az OS között.
- Adatok tárolására közvetlen (on-line) állományrendszerben.
- A felhasználó interakciója nagyon lassú, közben az OS más tevékenységet tud végrehajtani.
- Gyors reakció a parancsokra, válaszidő (response time) kicsi. Sűrűn kell a programok között kapcsolgatni.
- Felhasználók függetlenül használják a gépet, mintha mindenki egy saját gépen dolgozna.

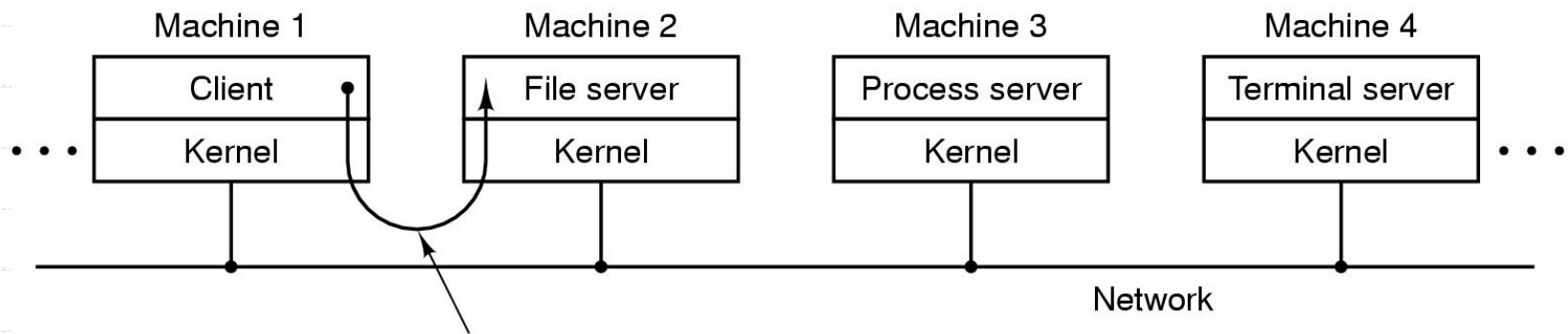
# Elosztott operációs rendszerek

- A számításokat több központi egység között osztják meg egyenletesen (multiprocessing)
- A felhasználó ezt nem veszi észre (transzparens)
- Szervezési elvek:
  - Azonosak-e a processzorok
    - homogén ↔ inhomogén
  - Van-e kitüntetett processzor:
    - szimmetrikus ↔ aszimmetrikus
  - Csatolás
    - szorosan csatolt (tár egy részét közösen használják)
    - nem szorosan csatolt (a CPU-kat kommunikációs csatorna köti össze) [elosztott rendszerek általában ilyenek]

# Elosztott operációs rendszerek kommunikációja

- Hálózati infrastruktúra kell (LAN vagy WAN)
- Architektúra: kliens-szerver, vagy peer-to-peer

Példa: egy kliens-szerver architektúra:



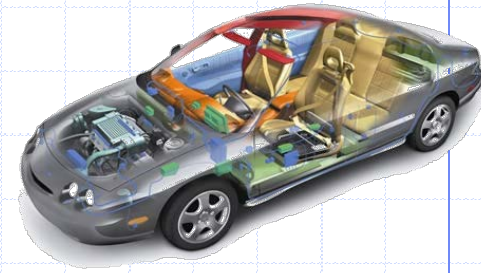
Üzenet egy klienstől egy szerver számára



# Elosztott operációs rendszerek előnyei

- Erőforrások megosztása
  - A felhasználó a másik gép erőforrásait (spec. hardver, perifériák, állományok adatbázisok) használhatja
- Terhelés megosztás
  - párhuzamosan végrehajtható részeket szét lehet osztani
  - megnövelt teljesítmény
- Megbízhatóság
  - Egy elem kiesése esetén a többi elem átveheti a szerepét.
    - Homogén esetén könnyebb, inhomogén esetében nehezebb
    - „graceful degradation”
- Kommunikáció
  - Programok, felhasználók közötti információ csere.

# Valós idejű rendszerek



- Kemény valós idejű rendszerek (hard real-time)
  - Szigorúan definiált és betartott válaszidők
  - Kritikus rendszerek (pl. atomreaktor, járművek)
  - Adattárolás: RAM, ROM, másodlagos tárolást (diszk) nem támogatják
  - Általános célú operációs rendszerek nem támogatják
- Puha valós idejű rendszerek (soft real-time)
  - A válaszidők betartására törekednek, csúszás megengedett
  - Nem kritikus folyamatirányítási feladatok
  - Multimédia rendszerek, virtuális valóság

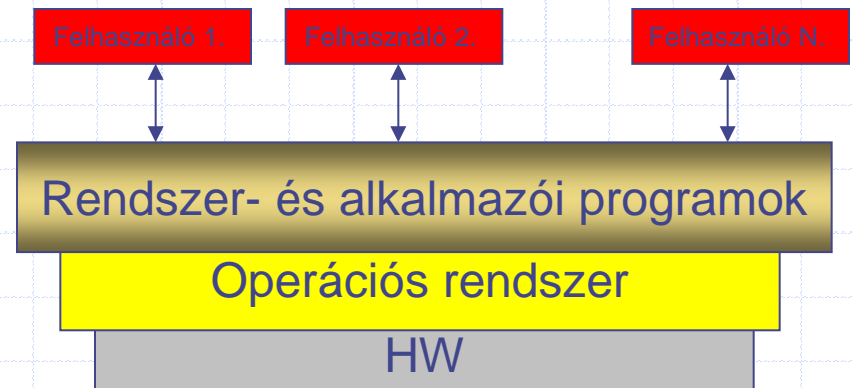


# Beágyazott operációs rendszerek

- Háztartási berendezések, mobiltelefonok, járművek, játékok
- Nem általános célú, nem független az alkalmazástól (összefordul)
- Korlátos erőforrások, korlátos szolgáltatások. OS-nek csak a szükséges része fordul be.

# 3. Az operációs rendszerek szerkezete

- OS:
  - nagyméretű
  - komplex program,
  - belső szerkezetük részekből áll.
- A felosztásnál *rétegeket* különböztetünk meg:
  - egymásra épülnek
  - az alattuk elhelyezkedő réteggel jól definiált felülettel érintkeznek
  - egyre bővülő funkcionalitás.
- Legalább 3 réteg:
  - hardver
  - operációs rendszer
  - felhasználói programok



# Rétegszerkezet

- Tovább bontott rétegek. pl:
  - 1. felhasználói programok
  - 2. periféria meghajtók
  - 3. virtuális tár kezelése
  - 4. I/O csatornák
  - 5. CPU ütemezése
  - 6. hardver

Tendencia a *kevés réteg*.

# Rétegszerkezet

- Tiszta rétegszerkezetű rendszerek: csak az alatta és felette levő réteggel kommunikál, a rétegek feladatainak egymásra épülését megnehezíti.
- Modulok, a rétegen belüli strukturálás: egy-egy jól meghatározott feladatcsoportra egy modul.
- A modulok egymás között kommunikálnak, egységes mechanizmus.

## 4. Az operációs rendszerek feladatai

- Rendszerhívások
- Fontosabb rendszermodulok
- Szolgáltatások
- Rendszerprogramok

# 4.1. Rendszerhívások

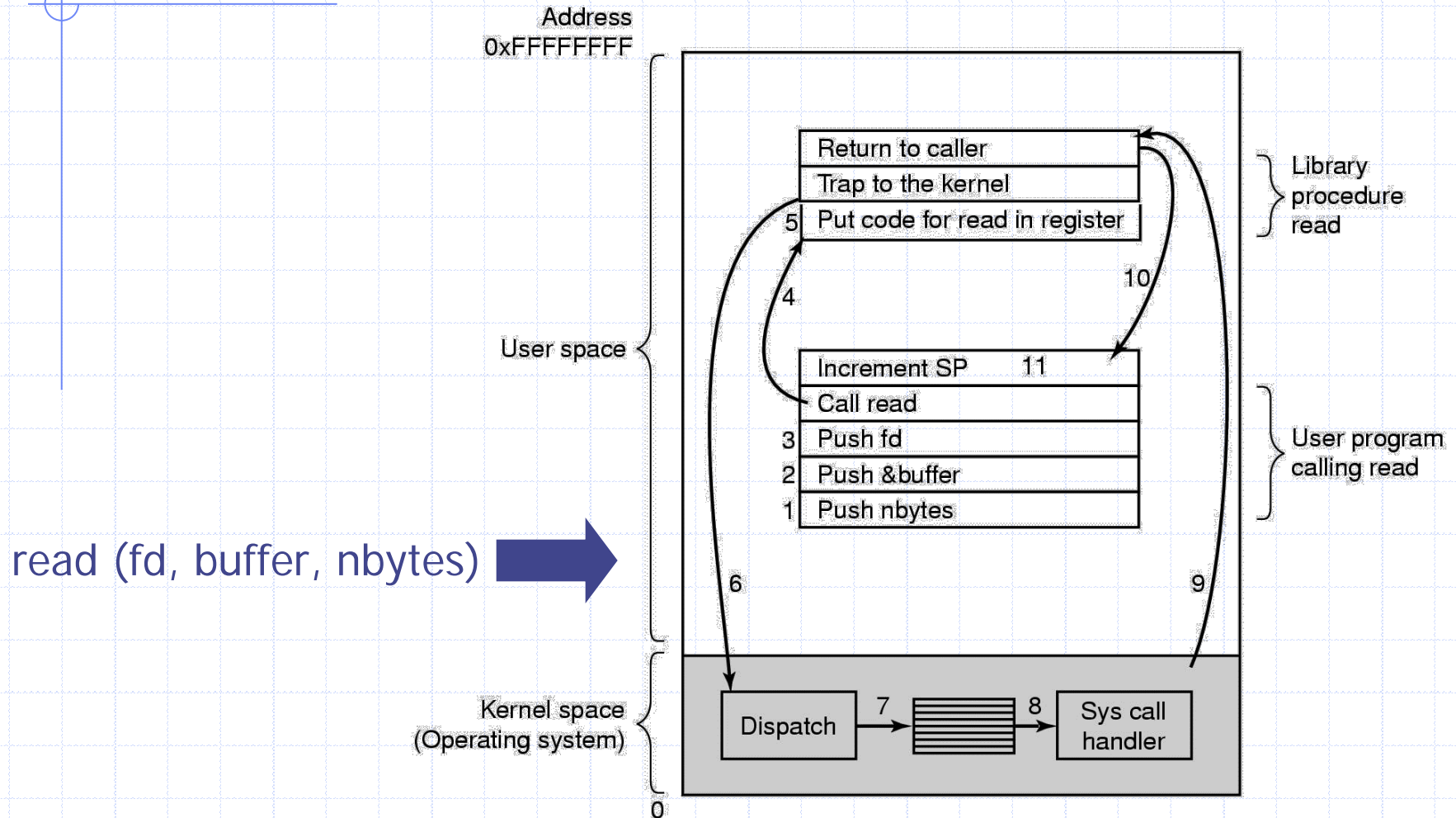
- Program - OS közötti felület szigorúan rögzített.
- A programok csak ezt látják.
- A rendszer szolgáltatásainak igénybevételeének eszközei a *rendszerhívások*.
- Rendszerhívás speciális gépi utasítás segítségével történik (trap). A vezérlést az OS programjának adja.
  - assembly-ből közvetlenül kiadhatók (pl. DOS int010 megszakítás)
  - magasszintű nyelveknél könyvtári eljárások.



# A rendszerhívás lépései

- Paraméter átadás (1 , 2 , 3)
  - Az OS által megadott helyre kell elhelyezni.
- Hívás, hardver működési mód váltás (4 , 5 , 6)
  - felhasználói mód (korlátozott, bizonyos utasítások nem) → rendszer mód.
  - A rendszerhívó gépi utasítás végzi el.
- Paraméterek kiolvasása, elágazás a hívott szolgáltatáshoz (a rendszerhívást megvalósító kód végrehajtása) (7)
- Visszaadott paraméterek másolása (8)
- Visszatérés a hívóhoz (módváltás) (9)
- Hívó tovább fut, eredmények felhasználása (10,11)

# A rendszerhívás lépései



# Példák rendszerhívásokra

## Miscellaneous

Call	Description
<code>s = chdir(dirname)</code>	Change the working directory
<code>s = chmod(name, mode)</code>	Change a file's protection bits
<code>s = kill(pid, signal)</code>	Send a signal to a process
<code>seconds = time(&amp;seconds)</code>	Get the elapsed time since Jan. 1, 1970

## 4.2. Fontosabb rendszermódulok

- Folyamatok kezelése
- Központi tár kezelése
- Perifériák kezelése
- Állományok kezelése
- Védelmi mechanizmusok
- Hálózatok kezelése
- Kezelői felület

# Folyamatok kezelése 1.

- Folyamat = végrehajtás alatt álló program (dinamikus)
- A program (statikus) egyszerre több folyamatból áll.
- Felhasználói folyamatok
  - egy munkafázis elindulásakor jönnek létre
- Rendszer folyamatok
  - a rendszer indulásakor keletkeznek.
- Folyamatok más folyamatokat hozhatnak létre.

# Folyamatok kezelése 2.

## Problémák

- Folyamatokhoz erőforrások biztosítása
  - Folyamat indulásakor allokálódnak, vagy amikor szükség van rá.
- Holtpont a folyamatok között (kölsönös várakozás).
- CPU mint erőforrás a folyamatok közötti megosztása (CPU ütemezés).

# Folyamatok kezelése 3.

## Folyamatokat kezelő rendszerhívások:

### Process management

Call	Description
<code>pid = fork( )</code>	Create a child process identical to the parent
<code>pid = waitpid(pid, &amp;statloc, options)</code>	Wait for a child to terminate
<code>s = execve(name, argv, environp)</code>	Replace a process' core image
<code>exit(status)</code>	Terminate process execution and return status

# Központi tár kezelése

- Folyamathoz annak indulásánál (esetleg később is) memóriát kell rendelni.
- Virtuális tárkezelés
  - virtuális cím - közvetlen cím hozzárendelés
- Központi tárban több folyamat utasításai, adatai vannak. Ezért biztosítani kell, hogy
  - a futáshoz szükséges részek az operatív tárban,
  - a nem szükségesek pedig a háttértáron legyenek.



# Perifériák kezelése

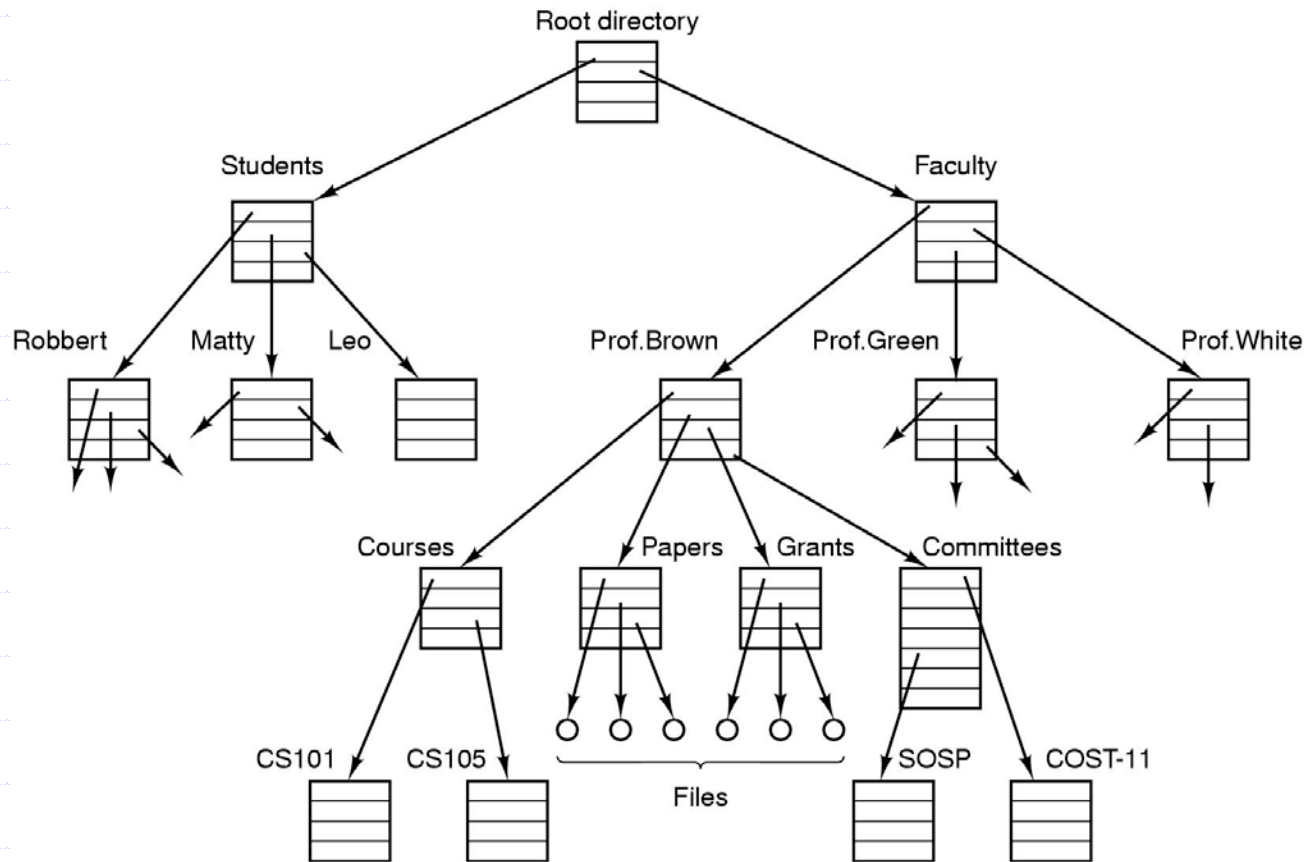
- Perifériák elrejtése a programok előtt.
- Átvitel pufferelése
  - CPU jobb kihasználása érdekében.
- Periféria (HD) jobb kihasználása érdekében több egyidejű igény ütemezése.

# Állományok kezelése

Fájl (file):

- egymáshoz tartozó információk gyűjteménye
- elrejt a tárolás részleteit.
- Tárolja a felhasználói programokat és adatokat.
- állományok csoportosítása könyvtárakban (hierarchia)
- állományok létrehozása, megszüntetése, olvasása, írása, stb., valamint a szabad hely nyilvántartása.
- Megbízható mentés (backup).

# Állományok felépítése



# Állománykezelő rendszerhívások

## File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

## Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

# Védelmi mechanizmusok

- OS-t és erőforrásait védeni kell a hibás vagy illetéktelen felhasználásoktól.
- Lehet
  - hardver, vagy
  - OS-ben megvalósított védelmi mechanizmus

# Hálózatok kezelése

- Távoli felhasználókkal, gépekkel való *kommunikáció*,
- távoli erőforrások elérése.

# Kezelői felület

- *OS és a felhasználó kommunikációja.*
  - Programok indítása, interaktív információcsere
- Parancsértelmező, parancsnyelv (karakteres kommunikáció)
- Grafikus, speciális beviteli eszközökkel megvalósított kommunikáció.
- Felhasználói programokra egységes eszközök, rendszerhívások, ezek használatát szabályozó ajánlások.

## 4.3. Az operációs rendszerek szolgáltatásai

- Folyamatok vezérlése
- Állományok kezelése
- Perifériás eszközök kezelése
- Rendszerinformációk kezelése
- Kommunikáció



# Folyamatok vezérlése

- program betöltése, végrehajtása
- folyamatok létrehozása, megszüntetése; attribútumainak beállítása;
- központi tár igénylése, felszabadítása;
- folyamatok közötti kommunikáció, folyamatok szinkronizálása (egymáshoz, órához, külső eseményhez);
- nyomkövetés hibakereséshez

# Állományok kezelése

- állomány létrehozása, törlése, attribútumok beállítása;
- könyvtárak létrehozása, módosítása;
- állományok megnyitása, lezárása, szekvenciális és véletlen átvitel .

# Perifériás eszközök kezelése

- Perifériák
  - igénylése;
  - lefoglalása;
  - felszabadítása;
- Adatátvitel a program és a periféria között

# Rendszerinformációk kezelése

- A rendszerkomponensek
  - folyamatok
  - állományok
  - perifériákállapotának lekérdezése, módosítása;
- rendszer statisztikák, naplózás

# Kommunikáció

- kommunikációs csatornák
  - létrehozása
  - megszüntetése;
- üzenetek
  - küldése
  - fogadása;
- műveletek távoli erőforrásokon.

## 4.4. Rendszerprogramok

Az OS-hez adott, annak általános célú felhasználását támogató programok.

- parancsértelmező
- állományrendszer kezelő
- szövegszerkesztő
- fordító program, linker, loader, könyvtár kezelő
- kommunikációs programok (pl. levelezés)