

# Operációs rendszerek

ELTE IK.

Dr. Illés Zoltán

[zoltan.illes@elte.hu](mailto:zoltan.illes@elte.hu)

# Bevezetés

- ▶ Visszatekintés (Számítógépes alapismeretek)
- ▶ Számítógépek felépítése I.(HW)
- ▶ Számítógépek felépítése II. (SW)
- ▶ Operációs rendszer fogalma
- ▶ Operációs rendszerek fejlődése, története
  - Múlt, Jelen, Jövő?
- ▶ Operációs rendszerek fogalmai
- ▶ Rendszerhívások
- ▶ Operációs rendszerek struktúrája

# Visszatekintés

- ▶ Ahol a számítógépes alapismeretek befejeződött...
- ▶ Script programok
  - Rendszergazda legjobb barátja
    - Shell script
    - PowerShell
- ▶ Kliens–szerver gép
  - HW különbségek
- ▶ Kliens–szerver szolgáltatás
  - Adminisztráció
  - SW különbségek

# Számítógépek felépítése (HW)

## ► Számítógépek felépítése

### ◦ Hardveres oldal

- Tárolt program, utasítások, adatok azonos módon (binárisan, miért?) a memóriában helyezkednek el.
- Vezérlő egység (CPU), aritmetikai–logikai egység (ALU) az utasítások végrehajtását, alapvető aritmetikai műveleteket felügyelik.
- Szükség van be/kimenetek (I/O) kezelésére, mely a gép és a külvilág kapcsolatát biztosítja.
- Ezen jellemzőket gyakran a Neumann elv elemeiként is ismerjük.

### ◦ Alapvető elemek: Processzor, Memória, Perifériák, Háttértár

- Összekötő kapocs: Busz (sín, adat, cím, vezérlő)

# Processzor utasítások

- ▶ A rendszer gyakorlatilag minden eleme intelligens, de a kulcsszereplő: processzor
- ▶ Regiszterek: speciális memóriák, processzoron belül
  - Regiszter csoportok (általános, állapot jelző, stb)
- ▶ Utasításcsoportok
  - Adatmozgató utasítások (regiszter –memória )
  - Ugró utasítások, abszolút–relatív
  - I/O port kezelés,
  - Megszakítás kezelés stb.



# Processzor védelmi szintek

- ▶ Intel 80286 minden utasítás egyenlő
- ▶ Intel 80386 nem az, 4 védelmi szint
  - Ebből 2-t használ, kernel mód (védett,protected mód) és felhasználói mód
- ▶ Tipikusan védett módú utasítások
  - Megszakítás kezelés
  - I/O port kezelés
  - Bizonyos memória kezelés
- ▶ Szofveres megszakítás, csapda (trap) kezelése azonos a hardveres megszakítás kezeléssel
- ▶ Megszakítások maszkolhatóak.  
Kivéve az NMI .

# Processzor utasítások használata

- ▶ Adatok, utasítások a memóriában, ezeket a CPU végrehajtja
  - Mov al, 'F'
  - Mov ah, 'T'
  - Mov bl, 'C'
  - Stb.
- ▶ Hol van itt az élvezet?
  - Hát ott, ha látom is az eredményt (FTC)...
  - Ha egy perifériát (pl. képernyő) elérek és azon megjelenítem az adatokat

# Számítógépek felépítése (SW)

- ▶ **Végrehajtási, felépítési szintek**
  - Logikai áramkörök
  - CPU, mikroprogram, mikroarchitektúra szint
  - Számítógép, hardver elemek gépi kódja
  - **Operációs rendszer**
  - Rendszeralkalmazások
    - Alacsonyszintű, gépi kódú programok, meghajtók
    - Magas szintű nyelvek, programok
  - **Alkalmazások**
    - Felhasználói programok, Pasziánsz stb.



# Operációs rendszer fogalma

- ▶ **Operációs rendszer:** Olyan program ami egyszerű felhasználói felületet nyújt, eltakarva a számítógép(rendszer) eszközeit.
- ▶ **Op. Rendszer mint kiterjesztett (virtuális) gép**
  - Nem érdekel hogyan, csak át akarok másolni egy képet.
- ▶ **Op. Rendszer mint erőforrás menedzser**
  - Nyomtatási sor kezelő (időalapú megosztás)
  - Memória (tér, címtér alapú megosztás)
- ▶ **Kernel mód– Felügyelt mód**
- ▶ **Felhasználói mód**
  - Gyakran op.rendszer feladatok is itt helyezkednek el.
- ▶ **Speciális Felügyelt mód–Beágyazott rendszer**

# Operációs rendszer feladata

- ▶ Jól használható felhasználói felület biztosítása
  - 0. generációs felület: sajátos kapcsolótábla
  - Korai rendszerek felületei: Speciális terminálok
    - Már ekkor kialakul a mai rendszer szerkezete.
  - 80-as évek eleje: mikrogépek (ZX81 stb), Basic
  - PDP kompatibilis TPA1140, soros terminálok
  - MS DOS karakteres felület
  - Unix\_X Window rendszer, Xerox, MacOS
  - Windows 3.1, 95, 98, Mill, 2000, XP, Win7
- ▶ Ezek mennyire jó felhasználói felületek?

# Kommunikáció a perifériákkal

- ▶ **Lekérdezéses átvitel (polling)**
  - I/O port folyamatos lekérdezése.
  - Sok helyen alkalmazott technika, gyakran szinkron szoftver hívásoknál is alkalmazzák.
- ▶ **Megszakítás (Interrupt) használat**
  - Nem kérgezgetjük folyamatosan, hanem a kívánt esemény bekövetkezésekor a megadott programrész kerül végrehajtásra.
  - Aszinkron hívások (programesemények) megfelelő használata
- ▶ **DMA, közvetlen memória elérés**
  - Pl. közvetlen memória címezés: 0xb800:0

# Programkönyvtárak

- ▶ Az iménti (gépi kódú, stb.) utasítások szintjei
  - Gépi kód
    - Pl:intel x86, mov ax, 'F', mov eax, 'T', jmp cím
  - **Normál, felhasználói programkönyvtárak (API, Application Programming Interface)**
    - C64 ROM Basic
    - DOS (IBM, MS) , IO.sys, msdos.sys, interrupt tábla
    - Windows 98,...Windows 7, Win32 API
    - **Unix-Linux rendszerkönyvtárak, C nyelv**
  - **Script programozás (BASH, PowerShell)**
    - Ezt láttuk, megismertük az I. félévben

# Felhasználói programkönyvtárak

- ▶ Jellemzően réteges szerkezetű
- ▶ Alapvetően két rétegre oszthatjuk:
  - Rendszer szintű hívás
    - Kommunikáció a perifériákkal
  - Felhasználói hívás
    - Széleskörű könyvtár biztosítás
- ▶ Milyen nyelvhez illeszkednek a könyvtárak?
- ▶ Hát a C nyelvhez! És még? A C++-hoz...☺
  - Persze más nyelvhez is, pl, Delphi-hez is van...
- ▶ Kompatibilitás



# Mi a POSIX?

- ▶ **POSIX = Portable Operating System Interface for uniX**
- ▶ **Hivatalos neve: IEEE 1003 – ISO 9945**
- ▶ **A POSIX valójában egy minimális rendszerhívás (API) készlet, szabvány**
- ▶ **POSIX 1, 1a, 1b, 1c módosítások léteznek**
- ▶ **Szabvány ANSI C–vel azonos függvénykönyvtár**
- ▶ **Ma gyakorlatilag minden OS POSIX kompatibilis**
- ▶ **A Windows–nak is van POSIX felülete**
  - **Windows Services for Unix**

# Fontosabb POSIX API témakörök

- ▶ Fájl, könyvtárműveletek
- ▶ Folyamatok kezelése
- ▶ Szignálok
- ▶ Csövek
- ▶ Standard C függvénykönyvtár
- ▶ Órák, időzítők
- ▶ Szemaforok
- ▶ Szinkron, aszinkron I/O
- ▶ Szálak kezelése
- ▶ Stb.

# Függvénycsoport példák

- ▶ Matematikai függvények: pl. sin, cos, tan, atan, atan2, log, exp stb.
- ▶ Állománykezelő függvények: pl. creat, open, fopen, close, read, write, unlink stb.
- ▶ Könyvtárkezelő függvények: pl. opendir, closedir, mkdir, rmdir, readdir stb.
- ▶ Karakterfüzér-kezelő függvények: strcpy, strlen, strcmp, strcat, strchr, strstr stb.
- ▶ Memória-kezelők: malloc, free, memcpy stb.
- ▶ Belső kommunikációs függvények: msgsnd, msgrcv, shmat, semop, signal, kill, pipe stb.

# Hogy használjuk a gyakorlatban?

- ▶ **Operációs rendszer: Suse Linux Enterprise server**
  - [Oprendszerek.inf.elte.hu](http://Oprendszerek.inf.elte.hu)
- ▶ **Szövegszerkesztő: vi, mcedit**
  - Vagy helyi grafikus szerkesztés, majd ftp.
- ▶ **Segítség: man**
  - Pl: man exit, man strlen
- ▶ **Fordítás: cc -c elso elso.c**
  - Igyekezzünk a figyelmeztetéseket is orvosolni!

# Operációs rendszer API-k

- ▶ Ahány rendszer, annyi függvénykönyvtár
- ▶ Ma is jellemző API-k:
  - Open VMS
  - OS/400
  - System V, BSD , közös rész: POSIX
  - Win32 API
  - Mac OS API
  - Windows Mobile, CE API
  - Palm OS
  - Nokia S40, S60, S80 API
  - Beágyazott API:
    - Java, .NET



# Firmware – Middleware

- ▶ A két végletet láttuk: Hardware – Szoftvare
- ▶ Hardware alatt már egyáltalán nem csak a fizikai eszközt értjük.
  - Például: HDD, az operációs rendszer „logikai” kezelést végez, a valódi cilinderek elérése a HDD programjának feladata.
  - Például: BIOS, ....
- ▶ Firmware: Hardverbe a gyártó által épített szoftver
- ▶ Middleware: Op. Rendszer feletti réteg
  - PL: JVM

# Operációs rendszer generációk I.

- ▶ **Történelmi generáció: Charles Babbage (1792–1871)**
  - Tisztán mechanikus, nincs op.rendszer
    - Operátor alkalmazás
  - Később mint programozót alkalmazta Ada Lovelace-t (Lord Byron lánya) (Ada nyelv)
- ▶ **Első generáció, 1940–1955, kapcsolótábla, relé, vákumcső**
  - Neumann János, Institute for Advanced Studies, Princeton
  - Egyedi gépek
  - Gépi kód, egyszerű matematikai számítások
  - Lyukkártyák megjelenése

# Operációs rendszer generációk II.

- ▶ **Második generáció 1955–1965, tranzisztoros rendszerek**
  - Megbízhatóvá váltak az elemek
  - Géptermek (mainframe) kialakulása
  - Tervezés, gyártás, programozás, üzemeltetés fázisának elkülönülése
  - Lyukkártyás, szalagos egységek, kötegelt rendszer megjelenése
  - Fortran nyelv
  - Op. Rendszer
    - FMS, Fortran monitor system
    - IBM 7094 hármasa, 1401 beolvasó – 7094 feldolgozó– 1401 megjelenítő

# Operációs rendszer generációk III.

- ▶ Harmadik generáció, 1965–1980, integrált áramkörök megjelenése
  - IBM 1401 és 7094 egybeolvadása: System/360 gépcsalád
  - Azonos rendszerek, felépítések, kompatibilitás megjelenése
  - OS/360 megjelenése, ez minden gépre jó, eredmény nagy, bonyolult op. Rendszer.
  - Multiprogramozás, multitask megjelenése
    - Több feladat a memóriában egyidejűleg.
  - Spooling, időosztás megjelenése
  - Nincs közvetlen on-line munka

# Operációs rendszer generációk III.

- ▶ Első időosztásos rendszer: M.I.T-en CTSS (Compatible Time Sharing System)
- ▶ MULTICS, Multiplexed Information and Computing System
  - AT&T Bell labs, General Electric támogatás
  - PL/1 nyelven készült
- ▶ Bell Labs, Ken Thompson, Multics lecsupaszítás, PDP 7 → UNIX
- ▶ Két fő irány
  - Berkeley University – Berkeley Software Distribution
  - AT&T Bell Labs, System V Unix



# Operációs rendszer generációk IV.

- ▶ 1980–tól napjainkig, személyi számítógépek, MS Windows
- ▶ LSI (large scale integration) áramkörök, CPU fejlődés
- ▶ Z80– CP/M (Control Program for Microcomputers)
  - ZX-81, ZX-Spectrum– Basic
- ▶ Intel x86 család, IBM PC– DOS, MS DOS
  - Parancssoros felület
- ▶ GUI– X Window, Mac OS X, MS Windows
- ▶ Hálózati, osztott rendszerek

# MINIX 3

- ▶ Kezdetben a UNIX forráskód az AT&T engedélye alapján felhasználható volt.
- ▶ UNIX – nem nyílt a forráskód, AT&T 7. verziótól
- ▶ MINIX – MINI Unix, nyílt forráskód
  - A.Tanenbaum, Vrije Univ. Amszterdam
  - C nyelven készült,
- ▶ Linus Torvalds, Tanenbaum hallgatója
  - MINIX módosítás, 1994, LINUs uniX->LINUX
  - Nyílt forráskód
  - LAMP–Linux–Apache–Mysql–Php

# Rendszerhívások

- ▶ Rendszerhívásoknak nevezzük azokat a szolgáltatásokat, melyek az operációs rendszer és a felhasználói programok közti kapcsolatot biztosítják.
- ▶ Két fő csoportba sorolhatók:
  - Folyamat vagy processz kezelő csoport
  - Fájlkezelő csoport
- ▶ Programozó legjobb barátja: man, ...

# Processz kezelés

- ▶ **Processz – egy végrehajtás alatt lévő program**
  - **Saját címtartomány**
  - **Processz táblázat**
    - Cím, regiszter, munkafájl adatok
  - **Processz indítás, megszüntetés**
    - Shell, gyerekfolyamatok
  - **Processz felfüggesztés**
    - memória térkép + táblázat mentés
  - **Processzek kommunikációja**
    - Szignálok

# Fájlkezelés

- ▶ Egy főkönyvtár, /
  - Fastruktúra
  - Bejegyzés kétféle: fájl, könyvtár
- ▶ Műveletek: másolás, létrehozás, törlés, megnyitás, olvasás, írás
- ▶ Jogosultságok: rwx, – adott jog hiánya
  - SETUID, SETGID, Sticky bit
- ▶ Fájlrendszer hozzácsolása, mount, leválasztása, unmount
- ▶ Specifikus fájlok:
  - Karakter, blokk fájlok, /dev könyvtár
- ▶ Speciális fájl: Adatcső, pipe



# Fontosabb folyamatkezelők

<code>pid = fork()</code>	A szülővel azonos gyermekprocesszus létrehozása
<code>pid = waitpid(pid, &amp;status, opts)</code>	Gyermek megszűnésére várakozás
<code>s = wait(&amp;status)</code>	A waitpid elavult változata
<code>s = execve(name, argv, envp)</code>	A processzus memóriatérképének felülírása
<code>exit(status)</code>	A processzus végrehajtásának befejezése és az exit státus beállítása
<code>size = brk(addr)</code>	Az adatszegmens méretének beállítása
<code>pid = getpid()</code>	A hívó processzus pid azonosítójának visszaadása
<code>pid = getpgrp()</code>	A hívó processzus csoportazonosítójának visszaadása
<code>pid = setsid()</code>	Új szekció létrehozása és processzuscsoport gid visszaadása
<code>l = ptrace(req, pid, addr, data)</code>	Tesztelésre használható



# Fontosabb szignálkezelők

<code>s = <b>sigaction</b>(sig, &amp;act, &amp;oldact)</code>	Szignálokon végrehajtandó akciót definiál
<code>s = <b>sigreturn</b>(&amp;context)</code>	A szignál eljárásból való kilépés
<code>s = <b>sigprocmask</b>(how, &amp;set, &amp;old)</code>	A szignál maszk vizsgálata vagy módosítása
<code>s = <b>sigpending</b>(set)</code>	A blokkolt szignálhalmaz megkérése
<code>s = <b>sigsuspend</b>(sigmask)</code>	A szignál maszk felülírása és a processzus felfüggesztése
<code>s = <b>kill</b>(pid, sig)</code>	Szignál küldése egy processzusnak
<code>residual = <b>alarm</b>(seconds)</code>	Az ébresztőóra beállítása
<code>s = <b>pause</b>()</code>	A hívó felfüggesztése a következő szignál érkezéséig



# Fontosabb fájlkezelők

<code>fd = <b>creat</b>(name, mode)</code>	Új fájl létrehozásának elavult változata
<code>fd = <b>mknod</b>(name, mode, addr)</code>	Reguláris, specifikus vagy könyvtár i-csomópont létrehozása
<code>fd = <b>open</b>(file, how, ...)</code>	Fájl megnyitása olvasásra, írásra vagy mindkettőre
<code>s = <b>close</b>(fd)</code>	Nyitott fájl lezárása
<code>n = <b>read</b>(fd, buffer, nbytes)</code>	Adat olvasása fájl tárolóba
<code>n = <b>write</b>(fd, buffer, nbytes)</code>	Adat írás fájl tárolóból fájlba
<code>pos = <b>lseek</b>(fd, offset, whence)</code>	A fájlmutató mozgatása
<code>s = <b>stat</b>(name, &amp;buf)</code>	Fájl állapotinformációinak megkérése
<code>s = <b>fstat</b>(fd, &amp;buf)</code>	Fájl állapotinformációinak megkérése
<code>fd = <b>dup</b>(fd)</code>	Nyitott fájl leírójának átmásolása
<code>s = <b>pipe</b>(&amp;fd[0])</code>	Adatcső létrehozása
<code>s = <b>ioctl</b>(fd, request, argp)</code>	Fájlokra speciális műveletek végrehajtása
<code>s = <b>access</b>(name, amode)</code>	Fájl elérhetőségének vizsgálata
<code>s = <b>rename</b>(old, new)</code>	Fájl átnevezése
<code>s = <b>fcntl</b>(fd, cmd, ...)</code>	Fájl zárolása és egyéb műveletek



# Fontosabb könyvtárkezelők

<code>s = <b>mkdir</b>(name, mode)</code>	Új könyvtár létrehozása
<code>s = <b>rmdir</b>(name)</code>	Üres könyvtár megszüntetése
<code>s = <b>link</b>(name1, name2)</code>	Egy új, a name1-re mutató name2 bejegyzés
<code>s = <b>unlink</b>(name)</code>	Egy könyvtárbejegyzés megszüntetése
<code>s = <b>mount</b>(special, name, flag)</code>	Fájlrendszer felcsatolása
<code>s = <b>umount</b>(special)</code>	Fájlrendszer lecsatolása
<code>s = <b>sync</b>()</code>	A raktározott adatblokkok írása lemezre
<code>s = <b>chdir</b>(dirname)</code>	A munkakönyvtár változtatása
<code>s = <b>chroot</b>(dirname)</code>	A gyökérkönyvtár változtatása



# Fontosabb jogosítvány, idő kezelők

<code>s = <b>chmod</b>(name, mode)</code>	A fájl védelmi bitjeinek változtatása
<code>uid = <b>getuid</b>()</code>	A hívó uid azonosítójának megkérése
<code>gid = <b>getgid</b>()</code>	A hívó gid csoportazonosítójának megkérése
<code>s = <b>setuid</b>(uid)</code>	A hívó uid azonosítójának beállítása
<code>s = <b>setgid</b>(gid)</code>	A hívó gid csoportazonosítójának beállítása
<code>s = <b>chown</b>(name, owner, group)</code>	A fájl tulajdonosának és csoportjának változtatása
<code>oldmask = <b>umask</b>(complmode)</code>	A módmaszk változtatása
<code>seconds = <b>time</b>(&amp;seconds)</code>	Az 1970. jan.1-jétől eltelt idő megkérése
<code>s = <b>stime</b>(tp)</code>	Az 1970. jan.1-jétől eltelt idő beállítása
<code>s = <b>utime</b>(file, timep)</code>	A fájlok utolsó hozzáférési idejének beállítása
<code>s = <b>times</b>(buffer)</code>	Az elhasznált felhasználói és rendszeridő megkérése



# Operációs rendszer struktúrák

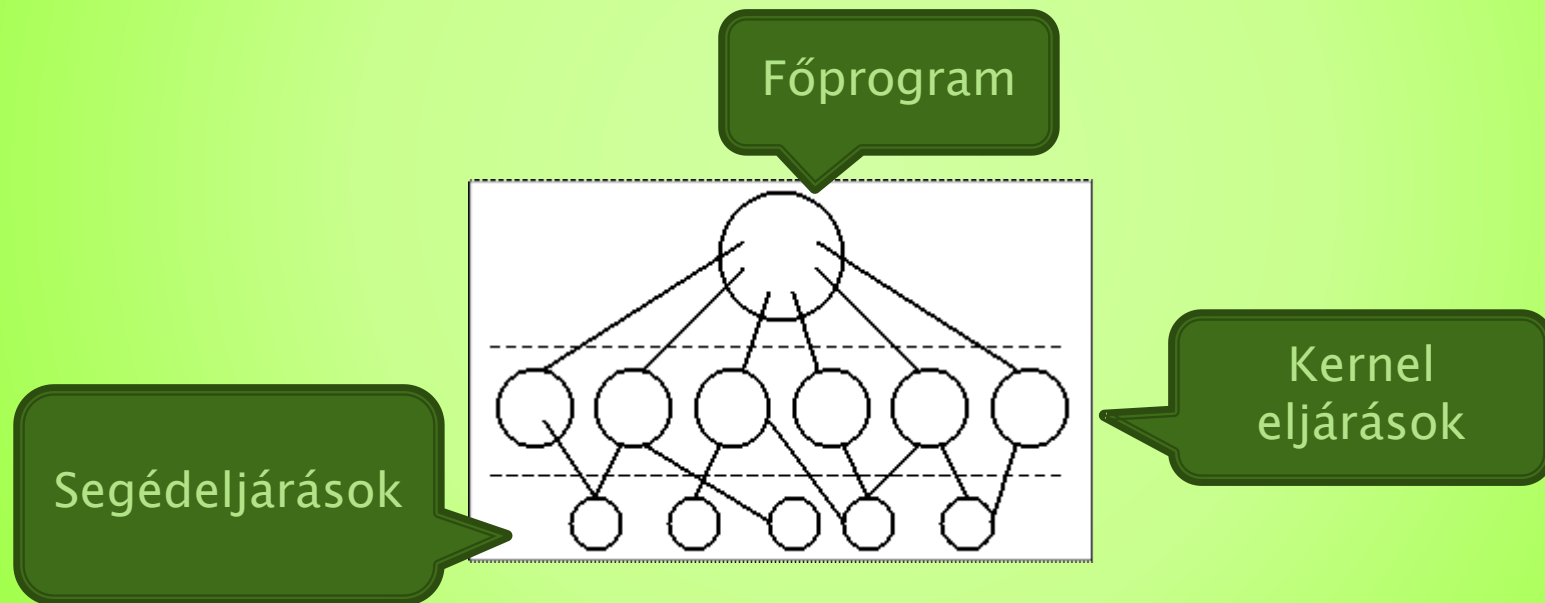
- ▶ Monolitikus rendszerek
- ▶ Rétegelt rendszerek
- ▶ Virtuális gépek
  - Exokernelek
- ▶ Kliens – Szerver modell

# Monolitikus rendszerek

- ▶ Általában igaz: nincs különösebb struktúrája, de...
- ▶ Rendszerkönyvtár egyetlen rendszer, így mindenki mindenkit láthat.
  - Információelrejtés nem igazán van.
- ▶ Létezik modul, modulcsoportos tervezés
  - Csak az előre tervezett belépési pontok hívhatók
- ▶ Rendszerhívás során gyakran felügyelt módba (kernel mód) kapcsolja a CPU-t
  - Paraméterek jellemzően regiszterekben
  - Trap, csapdázás

# Monolitikus szerkezeti modell

- ▶ Monolitikus rendszer: tipikusan 2 szintű támogatással



# Rétegelt szerkezet

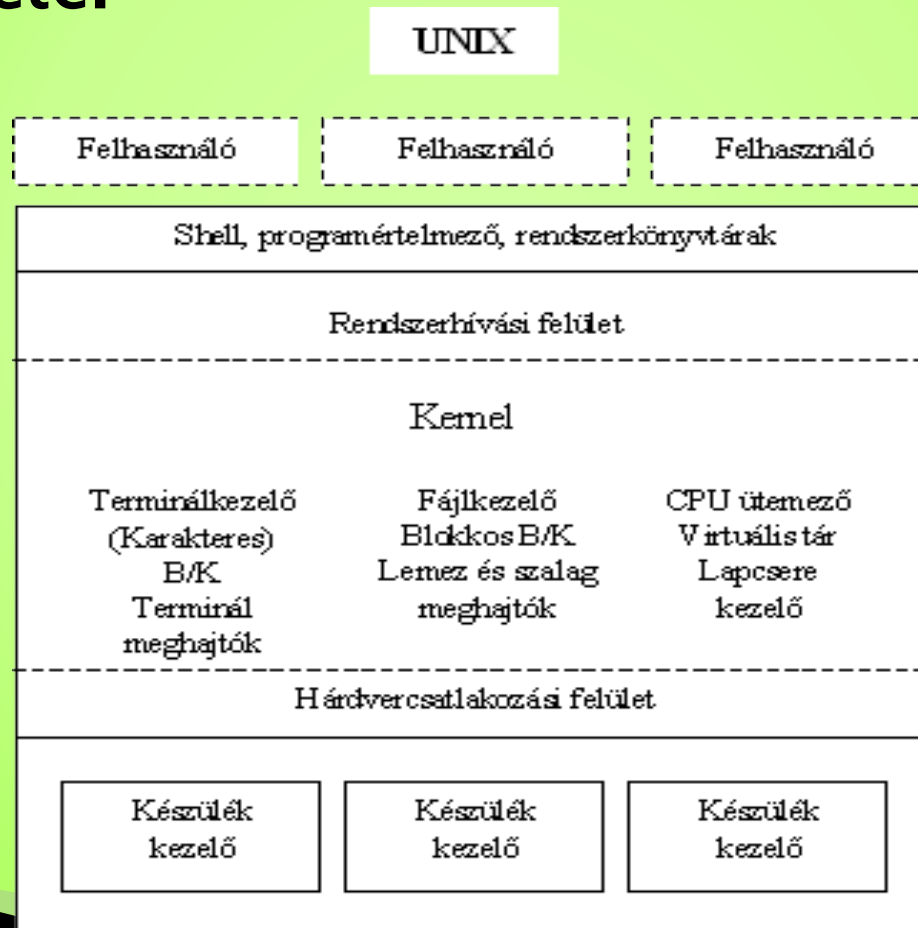
- ▶ E.W. Dijkstra tervezte, neve: THE (1968)

5.	A gépkezelő
4.	Felhasználói programok
3	Bemenet/Kimenet kezelése
2	Gépkezelő-folyamat
1	Memória és dobkezelés
0	Processzorhozzárendelés és multiprogramozás

- ▶ A MULTICS-ban tovább általánosították
  - Gyűrűs rendszer

# Tipikus rétegrendszer

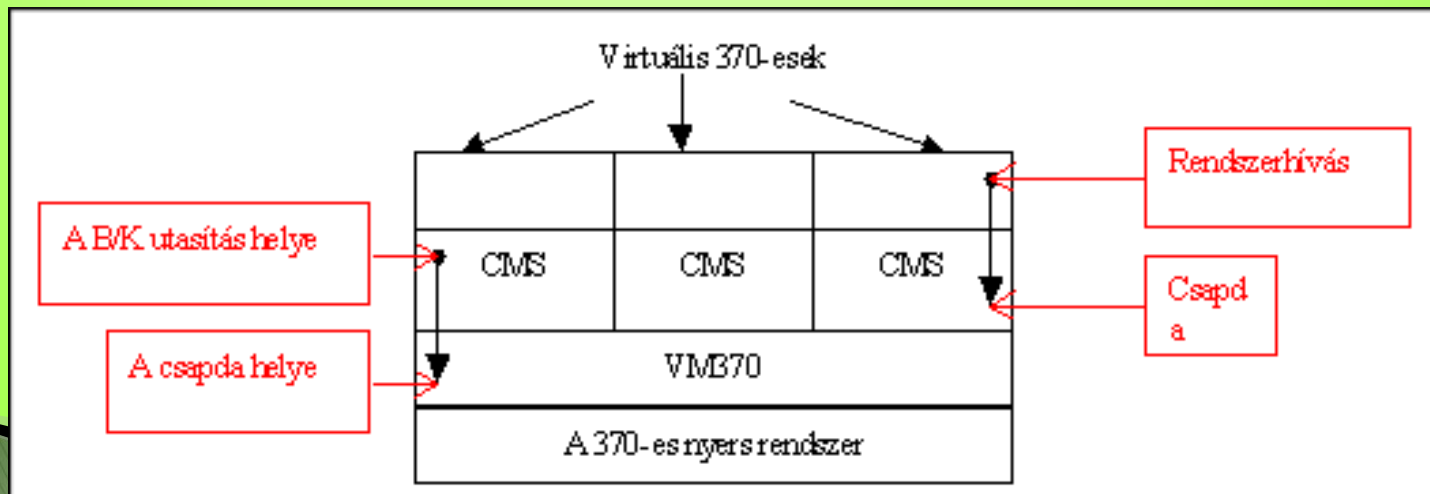
- ▶ A Multics utód UNIX jellemző réteges, gyűrűs szerkezete.





# Virtuális gépek

- ▶ Eredetileg az IBM-től származik az ötlet
- ▶ VM/370 rendszeren valósul meg először
- ▶ Virtuális gép monitor: a hardvert pontosan másolja
- ▶ Ezt tetszőleges példányban képes volt sokszorozni

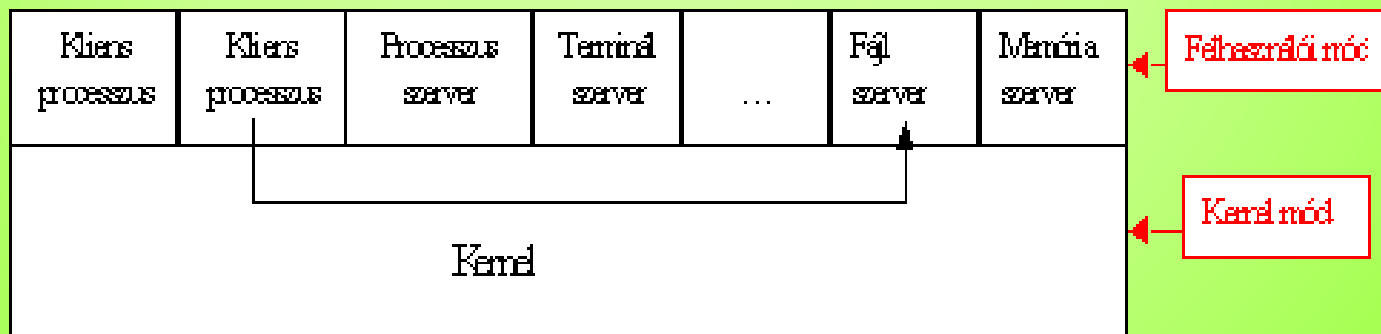


# Mai virtuális gépek

- ▶ **VMWare – Unix– Linux platformon**
  - Fut Windows-on is
- ▶ **MS Virtual Server, Virtual PC**
  - Létezik a Pentium utáni processzorokban 8086 virtuális üzemmód.
  - A Windows ebben futtatja a régi DOS programokat
  - Ez nem az igazi virtuális mód!!!
- ▶ **Hyper–V – XEN–KVM**
- ▶ **Exokernel: virtuális gép számára az erőforrások biztosítása**
- ▶ **Más rendszerű virtuális gépek:**
  - JVM
  - .NET

# Kliens–Szerver modell

- ▶ A vm/370 ötlet továbbfejlesztése
  - Még jobban szét kell választani a feladatokat.
- ▶ Felhasználói program: kliens program
- ▶ Kiszolgáló program: szerver program
- ▶ Mindegyik felhasználói módban fut
- ▶ Egyre kevesebb funkció marad a kernelben



# Operációs rendszer elvárások I.

- ▶ **Hatékonyság, a meglévő erőforrásokat a leghatékonyabban továbbítja a felhasználók felé.**
  - Efficiency
- ▶ **Megbízhatóság, a hibátlan működés biztosítása.(Reliability)**
  - Adatok megőrzése
  - Rendelkezésre állás (3–4 kilences...)
  - Megbízhatóság kiterjesztése: hibatűrés
    - Redundáns rendszerek (SW szinten is), Server Cluster

# Operációs rendszer elvárások II.

- ▶ **Biztonság (Security)**
  - Külső rendszerekkel szemben
  - Adatbiztonság
- ▶ **Kompatibilitás, hordozhatóság (Compatibility)**
  - Két rendszer közti adat, programcsere lehetősége.
  - Szabványok szerepe (POSIX)
- ▶ **Alacsony energia felhasználás**
  - Nem csak mobil gépek esetén.



# Operációs rendszer elvárások III.

- ▶ **Rugalmasság, skálázhatóság (Flexibility)**
  - Erőforrások rugalmas kiosztása (memória, processzor)
- ▶ **Kezelhetőség (Manageability)**
  - Üzemeltetési, felhasználói szinten
- ▶ **Megvalósítható mindez egyszerre?**
  - A gyártók szerint igen....☺
- ▶ **A félév végén meg fogjuk látni!**

# Köszönöm a figyelmet!

[zoltan.illes@elte.hu](mailto:zoltan.illes@elte.hu)