2020. tavasz

Operációs rendszerek

kidolgozott jegyzet

Összeállította: Csia Kitti

Tartalomjegyzék

Felhasznált irodalom:

Az operációs rendszerek nevű tantárgy hivatalos oldalán található aktuális előadások.

A tanszék által kiadott, felhasználható jegyzetek.

E jegyzet egy összefoglaló az előadásokról, az előadáson való megjelenés ettől függetlenül nagyon is ajánlott. Az előadáson előfroduló demókat, esetleges kódokat, linkeket a jegyzet nem tartalmazza.

BACK

1. előadás:
Bevezető

1. **Operációs rendszer**
   - Definíció
     - *szoftverrendszer*
     - programok összessége, amelyek vezérlik a számítógép hardverének működését, lehetővé teszik a felhasználói feladatok végrehajtását
     - **feladatai**:
       - felhasználói felület biztosítása
       - erőforrás-gazdálkodás
       - életciklus-menedzsment
       - kommunikációs és együttműködési protokollok
       - biztonsági, megbízhatósági alrendszerek
       - felhasználói feladatokat végrehajtó programok
       - fejlesztést támogató programok
       - programozót segítő funkciók
2. **Kezdetek**
   - ***batch computing***
     - *megelőzte*: tranzisztorok megjelenése
     - OS új funkció megjelenése:
       - állandóan működő program
       - feladatok egyenkénti végrehajtása → tovább lép másik köteg
   - ***multiprogramming***
     - *megelőzte*: IC, mágneses tárolóegység, DOS megjelenése
     - „üresjáratok” kitöltése más feladatokkal
       - OS egyszerre több feladatot kezelt
     - gépidő optimalizálás
     - adatátvitel támogatás háttértárakon

Commented [KC1]: Kötegelt számítási rendszerek.

Commented [KC2]: *Integrated Circuit* – Integrált Ármakör

- ***időosztásos***
  - hardver szintű védelem
  - programokhoz időszeleteket rendelve váltakozva futtatja azokat
- ***PC/Personal Computer***
  - *megelőzte:* LSI áramkörök, mikroprocesszorok, x86
  - felhasználóval való foglalkozás
- ***hálózati és elosztott rendszerek***
  - *megelőzte*: számítógép-hálózatok terjedése
  - együttműködő gépek → nagyobb feladatmegoldó erő
- ***beágyazott OS***
  - *megelőzte*: integrált funkcionalitás, komplex és olcsó chipek
  - „SMART" eszközök idejének kezdete

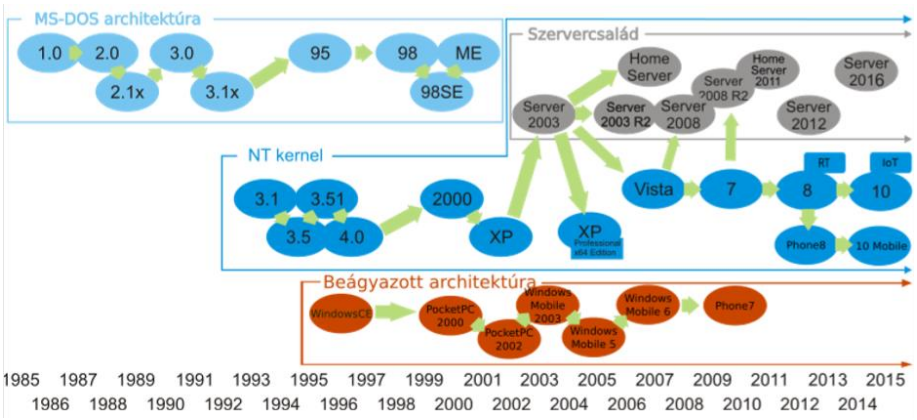
Commented [KC3]: Programozó és az admin is felhasználó.

3. UNIX disztribúciók

Commented [KC4]: Nem teljes lista.

- ***szervereken***
  - RedHat + klónjai: CentOS, Scientific Linux
  - SUSE Server
  - Debian
  - Ubuntu Server
  - OpenBSD, FreeBSD, NetBSD…
  - Oracle, Solaris
- ***klienseken***
  - Ubuntu
  - Linux Mint
  - Arch Linux
  - Fedora
  - SUSE Desktop
- ***beágyazott rendszerekben***
  - Android-alapú rendszerek
  - Tizen, OpenWrt, Embedded Debia, Arch Linux for ARM
  - OpenElec.tv

4. Windows változatok



5. Felhasználói feladatok elvárásai

- ***multitasking***
  - egymást nem zavaró folyamatok
- ***megbízhatóság***
  - ha hibázik, észlelje és javítsa ki
- ***biztonság***
  - védje meg az adatokat, rendszer erőforrásait, szolgáltatásait
- ***gyors folyamatvégzés***
  - inputra adott időn belül válaszoljon
- ***real-time system***
  - *hard real-time:* biztosan válaszol
  - *soft real-time:* nagy valószínűséggel időben válaszol

6. OS osztályzása

- ***futtatott felhasználói feladatok célja, jellege szerint***
  - **kliens**: felhasználói felület; kliens programok
  - **szerver**: monolitikus; mikrokernel; exokernel; multikernel
  - **beágyazott**: real-time; non-real-time
  - **más**: GRID rendszerek, nagy kapacitású tárolórendszerek

- ***OS és kernel tulajdonságai szerint***
  - **felhasználói felület**: GUI; karakteres
  - **kernel felépítése**: monolitikus; mikrokernel; exokernel; multikernel
  - **kernel működése**: real-time; non-real-time
  - **CPU**-támogatása: x86; arm; multi-arch; SMP; sokprocesszoros (100+)
  - **licence**: nyílt; zárt forráskódú
  - **kommunikáció**: hálózati; elosztott

Commented [KC5]: Graphical User Interface – Grafikus felhasználói felület

BACK

2. előadás:
Felépítés és alapműködés I.

1. **OS felépítése**

   - Definíció
     - *operációs rendszer*
       - azon programok összessége, amelyek vezérlik a számítógép hardverének működését
       - lehetővé teszik azon felhasználói feladatok végrehajtását
     - ***kernel***
       - *eseményvezérelt*
       - *védett mód*
         - OS egy része ebben fut
         - fennhatóságot minden program felett
         - szabályozza az életciklusokat
         - felügyeli a felhasználói módú programok működését
         - biztosítja a hozzáférésüket a rendszer erőforrásaihoz
       - *vezérlőprogram*
         - felügyeli más programok végrehajtását, szolgáltatást nyújt nekik
         - működési eseményeket kezeli, kézbesíti
         - életciklus menedzsment
       - *menedzseli az erőforrásokat*
         - eszközök előkészítése a felhasználásra
         - kezelésükkel kacsolatos közös funkciók biztosítása
         - működésükkel kapcsolatos események kezelése
         - párhuzamos kérések kiszolgálása, szeparációja, konfliktusok feloldása

Commented [KC6]: Operating System – Operációs Rendszer

Commented [KC7]: A kernel is egy ilyen védett módban fut. Minden más felhasználói módban működik. Ezeknek a hardverhez való hozzáférésük nagyon korlátozott.

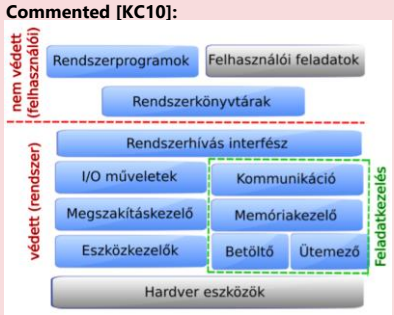
Commented [KC8]: Keletkezés, működés, megszűnés

Commented [KC9]: Létrehozás, működés, megszűnés.

- *megbízhatóság, biztonság*
  - erőforrások védelme a hibás vagy kártékony felhasználástól
  - a futó programok szeparációja, külső védelem
  - biztosnággal kapcsolatos közös funkciók biztosítása a programok számára

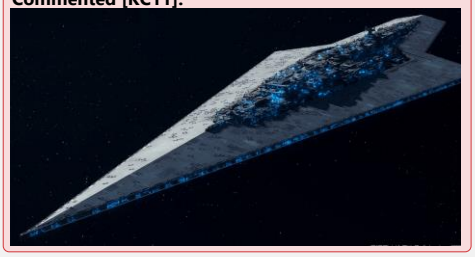
- ***OS további részei***
  - *rendszerkönyvtárak*
    - a programok felhasználhatják ezeket a könyvtárakat működésük során
  - *rendszerprogram*
    - operációs rendszer működésével kapcsolatos feladatokat oldja meg
  - *rendszerszolgáltatás*
    - operációs rendszer által kezelt, folyamatosan elérhető funkciókat nyújtó program

Commented [KC10]:



2. OS, mint végrehajtó

Commented [KC11]:



- ***feladatok***
  - *I/O intezív*
    - adatbetöltés, adatkiírás → sok várakozás
    - kevés processzoridőre van szükségük
  - Példa:
    - file server
    - web server
    - email client & server
  - *CPU-intezív*
    - kevés I/O műveletre van szükségük
    - nagy processzorigény
  - Példa:
    - titkosítás
    - matematikai műveletek
    - összetett adatfeldolgozás

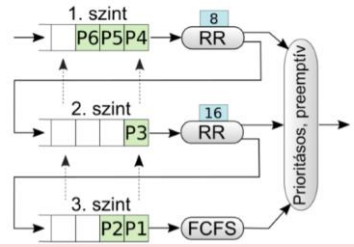
- *Memória-intezív*
  - ha sok memória van, akkor CPU-intenzívek, ha kevés, akkor I/O-intenzívek
  - egy időben nagy mennyiségű adat elérése
- Példa:
  - nagy mátrixok szorzása
  - keresési indexek építése, használata
- *Speciális igények*
  - valósidejű működés ...

- ***ütemezők mérőszámai***
  - **várakozási idő** *(waiting time)* csökkentése
    - task összes nem fut állapotában eltöltött ideje
  - **válaszidő** *(response time)* csökkentése
    - task külső kérésére (pl. kezelői parancsra) adott első válaszáig eltelt idő
  - **végrehajtási idő** *(execution time)*
    - task futó állapotában eltöltött ideje
  - **körülfordulási idő** *(turnaround time)*
    - task belépéstől kilépésig eltelt idő
    - *várakozási + futási idő = körülfordulási idő*
- ***hatékonyság***
  - **CPU-kihasználtság** *(CPU utilization)* minél nagyobb mértékben
  - **átbocsájtóképesség** *(throughput)*
  - **rezsiköltség** *(overhead)* minél kisebb
- ***jósolhatóság, determinisztikusság***

3. **OS, mint erőforrás-allokátor**

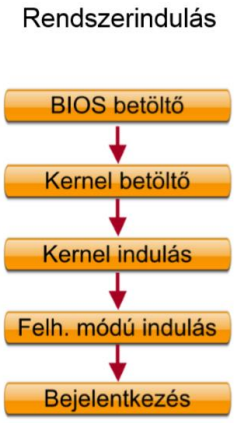
- ***többszintű visszacsatolt sorok ütemező***
  - taskok szintlépései
    - amelyik kihasználja az időszeletét, lentebb lép
    - várakozó task feljebb
- ***heterogén többprocesszoros rendszerek***

Commented [KC12]: *Multilevel feedback queue (MFQ)*



- ***memóriakezelés feladatai***
  - *kiosztja az erőforrást*
    - erőforrás: fizikai memória
    - igénylők: task, kernel
  - *elhelyezei a taskok adatait*
    - dinamikusan allokált
  - *elhelyezi a kernel adatait*
  - *biztosítja a védelmet*
  - *támogatja a kommunikációt*
    - adatcsere taskok között
- ***file-és tárolórendszerek***
  - **adattárolás**
    - fizikai tárolók (HDD, SSD)
    - I/O ütemezés
    - tárolórendszer-virtualizáció
      - helyi (RAID, LVM)
      - hálózati (SAN, NAS)
    - elosztott file- és tárolórendszerek

4. OS betöltése



- ***gép bekapcsolása***
  - elindul a rendszerórajel → processzor inicializációját indukálja
  - a processzor végrehajt egy fix címen kezdődő programot (ROM betöltő)
- ***ROM betöltése - BIOS***
  - hardverinicializálás (POST)
  - betöltőeszköz meghatározása
- ***RAM betöltése - MBR***
  - megkeresi és betölti a következő szintű betöltőt

Commented [KC13]: Read-Only Memory
Commented [KC14]: Basic Input/Output System
Commented [KC15]: Power-On Self-Test
Commented [KC16]: Random-Access Memory
Commented [KC17]: Master Boot Record

- ***OS betöltése – Boot loader, PBR\ VBR***
  - további programrészek betöltése (Windows: **Bootmgr**, Linux: **Grub** stage2)
  - felhasználói felület betöltése
  - kernel kódjának betöltése

Commented [KC18]: Partition Boot Record

Commented [KC19]: Volume Boot Record

Commented [KC20]:
Bootmgr
- 32 bites, nem védett módban
- megjeleníti a boot menüt
- 64 bites módba vált

Commented [KC21]: Windows: **Winload**
- 32/64 bites védett módban
- betölti az Ntoskrnl.exe-t és függőségeit, és az eszközkezelőket
- átadja a rendszerindulási paramétereket neki

BACK

3. előadás:
Felépítés és alapműködés II.

1. **Windows kernel indulása**
   - *SMSS*
     - a felhasználói módú működés alapbeállítása
       - fájlrendszerek ellenőrzése
       - környezeti változók
       - lapozófájlok
       - registry
   - *Wininit* – további felhasználói inicializálási lépések (Session 0)
     - pl. Service Control Manager (services.exe)
   - *CSRSS* – Win32 alrendszer indítása
     - további hardver-inicializálás (pl. teljes grafikus felbontás)
   - *Winlogon* – bejelentkezés-kezelő
     - bejelentkezési képernyő megjelenítése (LogonUI)
2. **Linux kernel indulása**
   - *inicializálás nem védett módban*
     - alap memóriakezelés, kernel verem, megszakítások...
     - indulási paraméterek átvétele
     - alapvető hardverek (pl. videókártya) beállítása
   - *védett módba váltás, inicializálás*
     - teljes memóriakezelés
     - megszakításvektorok kezelőfüggvényei
     - feladatkezelés adatstruktúrái
     - rendszerindulási eszközmeghajtók
     - ütemező
     - további architektúrafüggő feladatok

Commented [KC22]: Munkamenet-kezelő - *Session Manager*

Commented [KC23]: Client Server Runtime Process ~ munkamenet

3. UNIX felhasználói módú indulása

- ***indítás: init***
  - meghatározza, fenntartja a rendszer működési szintjét
  - elindítja/leállítja a szükséges OS szolgáltatásokat
- *futási szint (runlevel)*
  - számmal (0-6), vagy betűvel jelölik
    - 0: teljes leállás
    - 1 vagy S: single-user: egyfelhasználós (adminisztrátori) mód
    - 2-5: többfelhasználós üzemmódok (GUI, ha van)
    - 6: újraindítás
  - rendszergazda válthatja: telinit, init, shutdown, halt, reboot
- ***init működése***
  - konfiguráció: /etc/init.d/ és /etc/rc?.d/
  - parancsfájlok
    - nevüknek megfelelő sorrendben futnak
    - beállítják a rendszert
    - elindítják, illetve leállítják a szolgáltatásokat.
  - adminisztrátor meghatározhatja az aktív rendszerfeladatok körét
    - parancsfájlok manuálisan is meghívhatók
- ***sysinit alternatívái***
  - `systemd` (RedHat, CentOS, Ubuntu 15.04+, Debian...)
  - `upstart` (Debian, Ubuntu korábbi változatok)

4. Kernel belső felépítése

- ***felépítés alapelvei***
  - *réteges*
    - jól definiált interface-szekkel
  - *monolitikus*
    - a kernel részei egyetlen címtérben elérhetők
    - alapvető komponensek egyetlen modul részei

Commented [KC24]: Jellemzően az 5 az alapértelmezett teljes felhasználói mód.

Commented [KC25]: Lekérdezhető: who -r

Commented [KC26]: Pl. fájlrendszerek ellenőrzése és csatolása.

Commented [KC27]: Pl. felhasználói bejelentkezés, grafikus felület, adatbázis- és webszerver stb.

Commented [KC28]: Problémák az inittel:
egysíkú függőségek
lassan indítja a szolgáltatásokat
hibakezelés?

Commented [KC29]: deklaratív szolgáltatásleírások
párhuzamos/késleltetett indítás
működési hibák észlelése, kezelése

Commented [KC30]: A mai kernelek jellemzően moduláris, monolitikus szoftverek.

- *moduláris*
  - nem érhető el mindig minden rész
  - fordítási időben / konfiguráció során / futásidőben
- *elosztott*
  - önálló komponensek (külön címtérben)
  - üzenetalapú kommunikáció

- ***problémák kiküszöbölése***
  - *kernel sandboxing – amored OS*
    - kezelőfüggvények „védőréteggel” (hibadetektálás)
    - problémák felett felhasználói módú, helyreállító agent
  - *OS/app sandboxing*
    - kisebb felületű rétegek, erősebb szeparáció
      - **virtualizáció**: még egy felügyeleti szint
      - **konténerek**: ugyanazon a kernelen független OS-e
      - **unikernel**: mini kernel, alkalmazás + library OS egy címtérben
  - *monolitikus kernel elvetése*
    - elosztott rendszer
    - védett módban csak a legszükségesebb részek működnek

Commented [KC31]: KVM/VMware, Docker, MirageOS, Drawbridge.

5. Rendszerhívás

- Definíció
  - *rendszerhívás interface*
    - programozói felület
    - a kernel szolgáltatásai a felhasználói módban működő programok számára
    - üzemmódváltás szoftver megszakítással
- ***működése***
  - *kernel megszakításkezelője*
    - átveszi a paramétereket
    - végrehajtja a feladatokat
    - visszatér a megszakításból (`iret, sysexit`)

- ***működése UNIX alatt***
  - *kernel megszakításkezelője*
    - átveszi a paramétereket
- ***virtuális rendszerhívások***
  - megszakítás, üzemmódváltás költséges → virtuális verzió
    - speciális „kernel" memóriaterület a címtérben
    - biztonságosnak ítélt rendszerhívások érhetők el
    - ∾ megszakítás, módváltás
    - programok nem látnak különbséget

6. A mikrokernel

- Definíció
  - operációs rendszer kernel, mely csak az alapműködés feltétlenül szükséges kódrészleteket tartalmazza
  - minden más funkció felhasználói módban működtet
- ***elosztott rendszer***
  - működik, mint futtatórendszer
  - eszközkezelők hardver közeli részei
  - programok közötti kommunikációs infrastruktúra
  - minden más felhasználói módban
- ***előnyök***
  - rugalmas
  - megbízható
  - hibatűrőbb
  - helyes programozási szemlélet
- ***hátrány***
  - lassú, körülményesebb programozás

- ***új generációs mikrokernelek***
  - *L4 mikrokernel*
    - gyorsabb
    - CPU regiszterekbe is átvihető üzenet
    - kevés védett módú funkció
    - kicsi kernel
    - speciális ütemezés
    - hardverfüggő
  - *hibrid kernel*
    - monolitikus rendszerekkel vegyített mikrokernelek
    - OS X XNU

Commented [KC32]: Windowsban is van mikrokernel, de nem mikrokernel felépítésű.

BACK

4. előadás:
Felhasználói felület

1. **A paracsértelmező (shell)**
   - *a felhasználó parancsok*
     - *belső parancsok*
       - beépített utasítások
     - *külső parancsok*
       - shell meghatározza a program elérési helyét
       - elindítja a programot az argumentumokkal
       - munkamenet kezelése
       - végrehajtási eredmény visszaadása + hibák
   - *felhasználói felület típusok*
     - *karakteres/parancssori TTY*
       - stdin – stdout
       - minden rendszeren elérhető
       - korlátozott felhasználói „élmény”
       - hatékony, gyors, jól automatizálható
     - *Grafikus (GUI)*
       - WIMP: **W**indows, **I**cons, **M**enus, **P**ointer
       - ablakozó rendszer (*windowing system*)
       - ablakkezelő (*window manager*)
       - kijelzőszerver (*display server*)
       - erőforrásigényes
     - *a WIMP-en túl...*
       - hangalapú
       - gesztusokkal irányítható (mobil)

Commented [KC33]: TeleTYpewriter, text-only console.

Commented [KC34]: Graphical User Interface

- ***karakteres parancsértelmező***
  - UNIX: bash, csh, ksh, zsh
  - Windows: cmd.exe → PowerShell
  - *belső parancsok*
    - folyamatcsoport-vezérlés
    - bash builtins: `logut, alias, echo`...
    - PowerShell kulcsszavak
  - *programozható*
    - szövegkezeléssel kapcsolatos feladatokra is kiváló
    - alapvető programozási konstrukciók
    - külső parancsok használata

Commented [KC35]: Beépített súgó: man <parancs>, <parancs> -h vagy –help, Get-Help <parancs>, <parancs> /h vagy /?

Commented [KC36]: Elágazás, ciklus, eljáráshívás, makró...

2. A shell programozás alapjai

- ***beépített nyelvi elemek***
  - programozási konstrukciók
  - OS feladatok megoldása
- ***külső parancsok***
  - minden parancssori felülettel (is) rendelkező alkalmazás
  - PowerShell: .NET obejktumok manipulálása
  - elérési útvonaluk: `$PATH`
- folyamatcsoport vezérlés (***job-control***)

Commented [KC37]: Stop-Service -displayname „Bluetooth service”.

3. Grafikus felhasználói felületek

- ***WIMP***
  - ablakozó környezet + ablakkezelő + kijezőszerver
- ***GUI***: összetett szoftver
  - réteges, moduláris, sok, nyílt interface és protokoll
  - parancsértelmező + ablakkezelő + kijelzőszerver
  - pl.: Windows Shell, Gnome Shell, Ubuntu Unity...

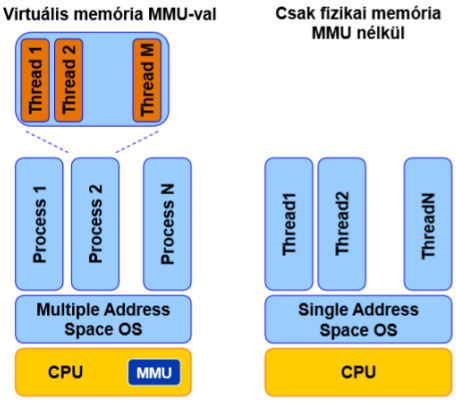
- ***ablakkezelő***
  - alkalmazásablakok megjelenítése + programozási felület, rendszerkönyvtár
  - testreszabható
  - pl.: Windows Desktop, Unix KDE
  - kompozíciós ablakkezelők (*compositing window manager*)
- ***kijelzőszerver***
  - *GUI és kiszolgáló harvder*
    - API és protokoll
    - továbbítás a harvder felé
    - eredmények az alkalmazás felé
    - működés felhasználó módban
  - *protokollok*
    - Unix X11
    - Unix Wayland
    - Android SurfaceFlinger
    - további :Microsfot RDP, Citrix HDX...

BACK

5-6. előadás:
Beágyazott real-time operációs rendszerek

1. **Beágyazott rendszerek**
   - Definíció
     - speciális számítógépes rendszerek, amelyek jól meghatározott feladatra találtak ki
     - külvilággal intenzív információs kapcsolat
     - adott eseményre adott időn belül válaszolnak
   - ***felépítés***
     - feldolgozó egység
     - szenzorok
     - beavatkozók
     - kommunikációs inteface
     - felhasználói felület
   - ***alkalmazási területek***
     - autóipar
     - szórakoztató elektronika
     - orvosi berendezések
     - háztartási berendezések
     - mérőberendezések
     - hálózati berendezések
     - űrkutatás
     - ipar PC-k
     - kártya méretű PC-k
     - okoskütyük?

	Klasszikus	High-end
Processzor órajel	~10 – x100 MHz	~ xGHz
Processzor védelmi szintek	Nincs	Van
MMU (virtuális tárkezelés)	Nincs	Van
Folyamatok, szálak?	Csak szálak	Folyamatok (és bennük szálak)
Linux futhat rajtuk?	Nem	Igen



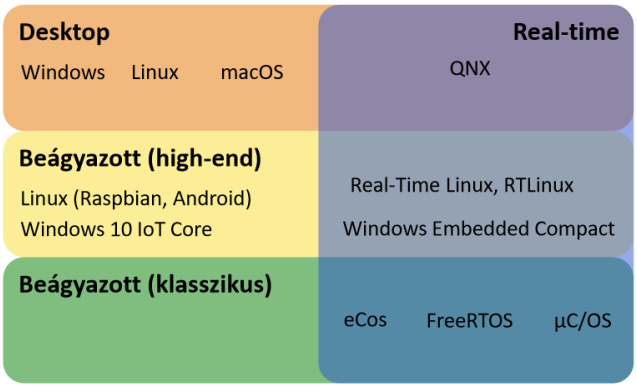
- ***belső perifériák***
  - CPU
  - memóriák: program (flash), adat (SRAM)
  - ADC, DAC
  - időzítő áramkörök
  - *kommunikációs interface-ek*
    - egyszerűbb: U(SART), I2C, SPI
    - bonyolultabb: USB (device, host), Ethernet (100M)
    - terület specifikus: CAN, LIN, FlexRay (autóipari buszoknál)
  - *általánosabb I/O*
    - egyszerűbb: LED kigyújtása, nyomógomb beolvasása
- ***csoportosításuk***
  - *szigorú módszer*
    - rendszer vagy real-time vagy nem

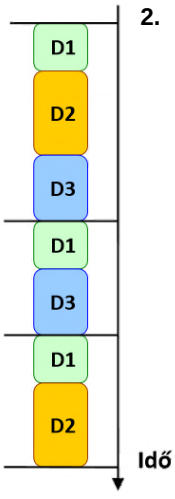
Commented [KC38]: Betartja a határidőket.

- *megengedőbb módszer*
  - **hard real-time**
    - rendszer a határiődket 1 valószínűséggel tudja tartani
    - határidő elmulasztása katasztrófa
  - **soft real-time**
    - rendszer a határiődket közel 1 valószínűséggel tudja tartani
    - határidő elmulasztása káros, de nem végzetes

Commented [KC39]: Pl. légiközlekedés.

Commented [KC40]: Pl. banki átutalás késik.





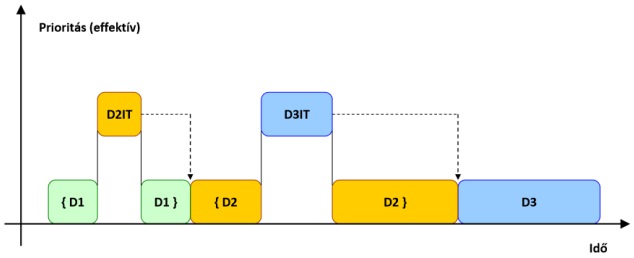
2. Real-time operációs rendszerek

- *ciklikus programtervezés*
  - legegyszerűbb
    - nincs megszakítás, közös erőforrás probléma
  - válaszidő nagy szórású
  - worst-case válaszidő
  - válaszidő javítható többszöri lekérdezéssel (főhurokban)
  - „törékeny” architektúra

Commented [KC41]: Feladatok válaszidejének összege.

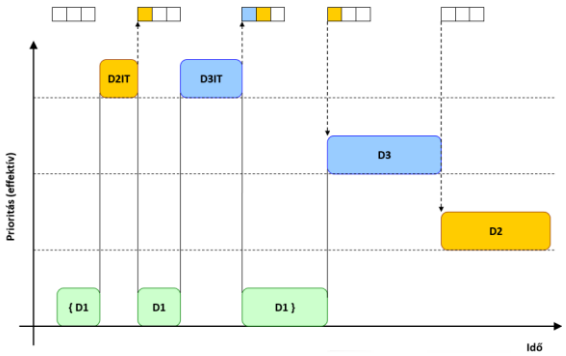
Commented [KC42]: Új task hozzáadása felboríthat mindent.

- ***ciklikus programtervezés IT-kkel***



  - jobban kezeli az időkritikus részeket
    - megszakítások magasabb effektív prioritáson futnak
    - közös erőforrás problémája
  - válaszidő szórása nagy
  - worst-case válaszidő
  - válaszidő javítható többszöri lekérdezéssel (főhurokban) egyes feladatoknál
  - „törékeny” architektúra
    - interruptokhoz prioritás rendelésével finomítható az architektúra

- ***függvény-sor alapú ütemezés***



  - prioritások kezelése (task, IT szinten)
    - közös erőforrás problémája IT és a főprogram között
  - válaszidő kevésbé szór
  - worst-case válaszidő nem nő lineárisan a taskok számával

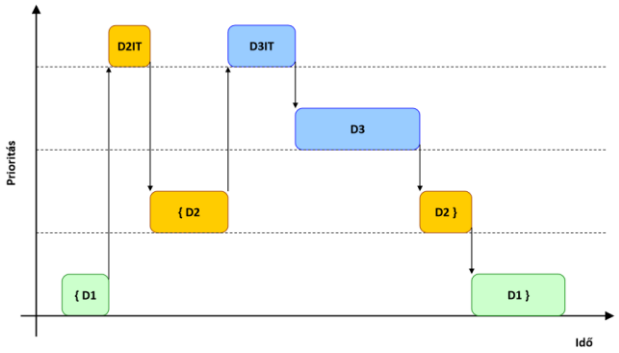
Commented [KC43]: Összes feladat válaszideje + megszakítások (arányos a taskok számával).

Commented [KC44]: Új task hozzáadása felboríthat mindent.

Commented [KC45]: Leghosszabb task válaszideje + IT.

- architektúra
  - új feladat nem borítja fel az eddigi időzítést
- nem preemptív

- ***RTOS architektúra (preemptív)***



- prioritások kezelése (task, IT szinten)
  - közös erőforrás problémája IT – főprogram, task – task között
- válaszidő szórása alacsony
  - nem nő új feladatok hozzáadásával
- worst-case válaszidő legmagasabb prioritású feladatra
- kód overhead
- tipikus képviselők
  - FreeRTOS
  - µC/OS-II, µC/OS-III
  - eCOS

Commented [KC46]: Task váltási ideje + IT (mindkettő kicsi).

3. FreeRTOS

- 2003: Richard Barry kezdte el fejleszteni, majd cége
- 2017-től AWS vette át + Richard
- egyéb szoftvercsomagok
  - kommunikáció (TCP/IP, MQTT)
  - titkosítás (TLS)

Commented [KC47]: Amazon Web Services

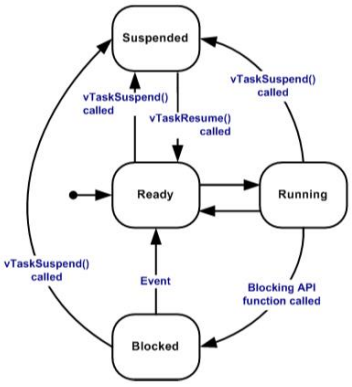
- cél:
  - beágyazott eszközök számára könnyű, biztonságos legyen egymás közti, felhő felé a kommunikáció
- forráskódja könnyen portolható
- kis méret, egyszerű használat
- ***főbb jellemzői***
  - *ütemezés egysége*
    - task (legelterjedtebb)
    - ko-rutin (tasknál egyszerűbb eszköz)
    - hibrid (task + ko-rutin)
  - *taskok ütemezése*
    - prioritásos
    - preemptív
    - RR időfelosztás segítségével az azonos prioritási szinteken
    -
- ***lincencelés***
  - ingyenes, nyílt forrású
  - kereskedelmi alkalmazásban használható
  - jogdíj mentes
  - technikai támogatás fizetős
  - nincs garancia, jogi védelem

- ***OPENRTOS***
  - Wittenstein High Integrity Systems áll mögötte
  - kereskedelmi lincencelésű változat
  - nem ingyenes → van garancia, IP védelem

- ***SAFERTOS***
  - Wittenstein High Integrity Systems áll mögötte
  - funkcionális modell hasonló
  - biztonság kritikus elvek alapján újraépítve
    - ⦰ dinamikus memória foglalás
    - alaposabb paraméter ellenőrzés Os hívásoknál

Commented [KC48]: Jelenleg 35 beágyazott architektúrával támogatott.

- ***fontosabb OS szolgáltatások***
  - tasks
  - binary semaphores (*bináris szemaforok*)
  - mutexes
  - queues *(sorok*)
  - event groups/flags
  - software timers
  - memory management
- ***fontosabb OS szolgáltatások***
  - co-routines
  - tickless idle mode (*óraütés nélküli üresjárás)*
  - counting semaphores
  - recursive mutexes
  - direct to task notifications (*közvetlen task értesítések)*
  - byte steam and stream/message buffers
  - blocking on multiple RTOS objects (*várakozás több RTOS objektumra)*
  - hook functions (*kicsatolások)*
  - TLS – Thread Local Storage pointers
  - stack overflow (*verem túlcsordulás)*
  - trace features (*nyomkövetés)*
  - run time statistics
  - memory protection support



- ***tipikus alkalmazás - taskok***
  - megvalósítás függvényekben
  - kétféle task szervezés lehetséges
    - *egyszeri lefutású*
      - futása során kiválthat olyan eseményeket, amelyek más taskok futását előidézik
      - futása végén törli magát → `vTaskDelete()`
    - *végtelen ciklusú*
      - elején opcionális inicializálós szakasz
      - végtelen ciklus
      - OS hívás elhelyezésekor várakozó állapot

4. **Egyszerű FreeRTOS demó**
   - lásd: előadás

BACK

7. előadás:
Feladatkezelés

1. **Fogalmak**
   - Definíció
     - ***task***
       - végrehajtás alatt álló program
     - ***állapot***
       - adminisztratív jellemzők összessége egy adott pillanatban
     - ***életciklus***
       - létrehozástól a befejeződésig terjedő állapotváltozások
     - ***szál***
       - szekvenciális működésű task
     - ***folyamat***
       - önálló memóriatartománnyal rendelkező task
     - ***verem***
       - átmeneti adattár, pl.függvényhívásoknak
     - ***halom***
       - futásidőben, dinamikusan tárolt adattár
     - ***kontextus***
       - állapotleíró
       - pl. utasításszámláló (PC), CPU, MMU...
     - ***task kontextusa***
       - végrehajtás állapot leírója
2. **Task**
   - ***1 feladat – 1 task***
     - `ps -ef`

- ***1 feladat – N task***
  - `ps -ef | wc -l`
  - jobb erőforrás kihasználtság
  - nagyobb teljesítmény
- taskok kommunikálhatnak egymással
  - adatcsere
  - részfeladat szétosztás
  - részeredmény egyesítés
  - közös vezérlési szerkezetek
- ***task szeparáció***
  - *absztrakt virtuális gép*
    - kernel által biztosított erőforrások együttese
    - virtuális CPU + virtuális memória
    - *multiprogramozott rendszer*
    - nehezíti a taskok kialakítását

3. Folyamatok és szálak

- ***szál***
  - **végrehajtó egység**
  - egy folyamaton belül vannak
    - egymással párhuzamos működés
    - más szálakkal közös memóriatartomány
    - tudnak egymással adatot cserélni, kooperálni
  - együttműködő taskok megvalósítására
  - *előnyök*
    - kisebb erőforrásigény
    - egyszerűbb létrehozás
    - egyszerűbb kommunikáció
  - *hátrányok*
    - nem mindenütt elérhető
    - gondosabb programozást igényel

- ***folyamat***
  - **védelmi egység**
  - több szál is futhat bennük
    - önmagukban párhuzamosan is működhetnek
    - nem látják más folyamatok memóriatartományát
    - egymással nehezebben működnek együtt
  - *előnyök*
    - kernelszintű védelem
    - elterjedtebb
  - *hátrányok*
    - nagyobb erőforrásigény
    - nehézkesebb kommunikáció
  - **UNIX folyamatok**
    - folyamatot csak másik folyamat hozhat létre
    - szülő-gyerek leszármaztatás
    - *szülő*
      - leálló folyamatok gyerekeit az init örökli
      - nyilvántarthatja a gyerekfolyamatait
      - értesítést kap a gyerek folyamat leállásról

Commented [KC49]: OS védelem.

Commented [KC50]: Nyugtáznia kell.

4. **Taskok adatai**

- ***saját***
  - programkód
  - statikusan allokált adatok
  - verem
  - halom
- ***adminisztratív*** *(kernel)*
  - *folyamat futása során szükségesek*
    - hozzáférés-szabályozás adatai
    - rendszerhívások állapotai és adatai
    - IO műveletek adatai
    - számlázási és statisztikai adatok

- *folyamat futása során szükségesek*
  - azonosítók (PID, TID)
  - futási állapot és ütemezési adatok
  - memóriakezelési adatok
- *tulajdonos és jogosultságok*

5. **Taskok állapotai**

Commented [KC51]:

- ***created*** *(létrejövés)*
  - task programja betölt, elindul
  - kernel létrehozza a szükséges adatstruktúrákat
- ***operating*** *(működés)*
  - futásra kész *(ready to run)*
  - várakozik *(waiting)*
  - fut *(running)*
  - blokkolt *(blocked)*
- ***terminated*** *(megszűnés)*
  - önszántából vagy végzetes hiba hatására
- ***interrupted*** *(megszakítás)*
  - rendszerhívások is megszakításnak számítanak
  - a kernelek megszakítás -, azaz eseményvezéreltek
  - kivétel hatására
    - CPU szól közbe
    - kernel megszakítás itt is
    - → kivételkezelő indul
- ***losted right to run*** *(futáshoz való jog elvesztése)*
  - lejár az időszelete
  - hibás működés

6. **Kontextusváltás**

- ***taskváltás → kontextusváltás (task → task)***
  - jelenlegi task kontextusának mentése
  - korábbi task kontextusának helyreállítása

- ***megszakítás → kontextusváltás (task → kernel)***
  - kontextus egy kis része hardver támogatással elmentődik
  - megszakításkezelés elindul
  - visszatérésnél visszaállítódik a korábbi végrehajtási kontextus

	felhasználói mód	kernel (védett) mód
taszk kontextus	fut a taszk saját programja	a taszk rendszerhívást hajt végre
kernel kontextus	(üres)	megszakítások, rendszerfeladatok kezelése

Commented [KC52]: Üzemmódváltás!)

BACK

8. előadás:
Ütemezők I.

Commented [KC53]: Egyszerű ütemező algoritmusok.

1. Ütemezés

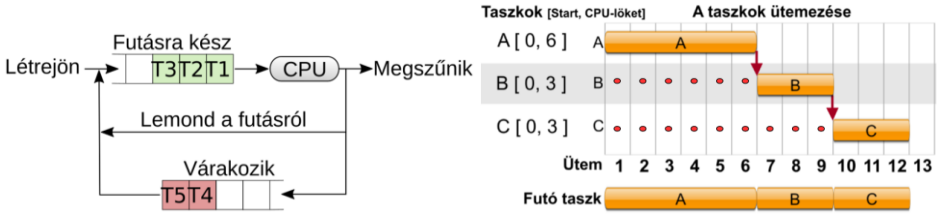
- ***feladata***
  - ütemezés feladat eldönteni, hogy melyik task fog futni
- ***ütemezés időskálái***
  - *rövid távú/CPU ütemezés*
    - futásra kész állapotú taskot választ
    - 1 – 100 ms
    - kernel alapfeladat
    - főbb típusai
      - *egyprocesszoros*
        - » egyszerű
        - » összetett
      - *többprocesszoros*
        - » homogén
        - » heterogén
    - fontos a rezsiköltség
  - *középtávú*
    - bármilyen állapotú taskot választ
    - új állapotok: felfüggesztve várakozik és futásra kész
    - percek – órák
    - felhasználó, kernel is kezdeményezheti
  - *hosszútávú*
    - feladatot választ, taskot indít
    - órák, napok, hetek...
    - nem a kernel hatásköre (*pl. Unix Cron, Windows Task Scheduler*)

2. Ütemezés adatstruktúrái

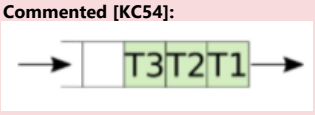
- *FIFO*
  - lista végéhez fűzünk és elejéről veszünk ki
  - művelet: $O(1)$ konstans (minimális)
- *rendezett lista*
  - beillesztés más ponton is lehetséges
  - művelet: $O(N)$ lineáris
- *bináris keresőfa*
  - beillesztés más ponton is lehetséges
  - művelet: $O(\log N)$ logaritmikus

3. Ütemezési algoritmusok

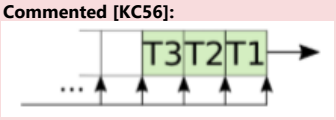
- *FCFS (First Come, First Served)*

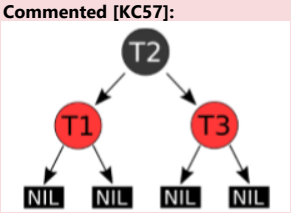


  - *tulajdonságai*
    - kooperatív
    - egyszerű (FIFO)
    - F → V nem hajtódik végre az ütemező által
  - *problémák*
    - **Konvoj hatás**
      - futásra kész task feltorlódik a hosszú CPU-löketű futó task mögött
      - feltorlódók nem kerülnek várakozó állapotba
      - algoritmus magas várakozási időt ad

Commented [KC54]:


Commented [KC55]: Vagy „rezsiköltség”.

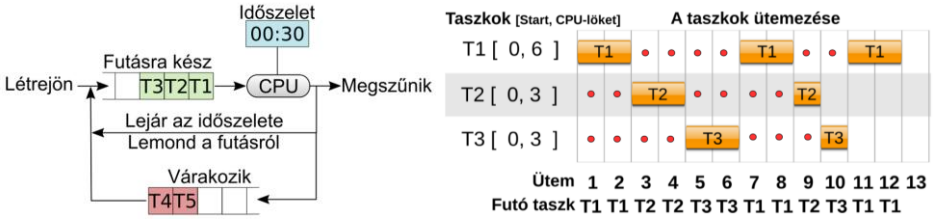
Commented [KC56]:


Commented [KC57]:


Commented [KC58]: Fut.

Commented [KC59]: Várakozik.

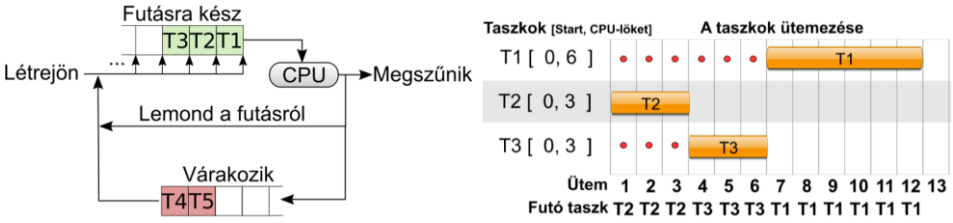
- ***körforgó ütemezés (Round-Robin, RR)***



  - *tulajdonságai*
    - preemptív
    - egyszerű (FIFO), minimális, ha jól megválasztott az időszelet
    - minden szinthez időszelet rendelése, fontosabb szint, nagyobb időszelet
  - *problémák*
    - nem jól megválasztott időszelet
      - túl nagy: FCFS algoritmus lesz
      - túl kicsi: túl sok kontextusváltás, rezsiköltség ↑

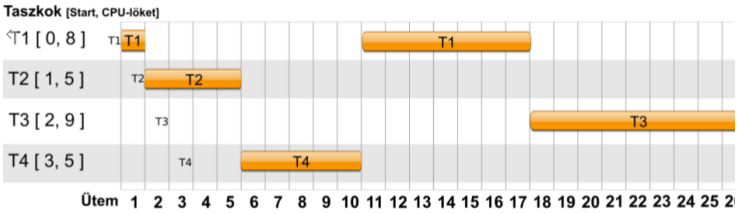
Commented [KC60]: Nő.

- ***legrövidebb löketidejű (Shortest Job First, SJF)***



  - *tulajdonságai*
    - kooperatív
    - bonyolultabb adatstruktúra
      - : $O(N)$ - *beszúrás*
    - futásra kész taskok CPU-löketidejük szerint növekvő sorrendben

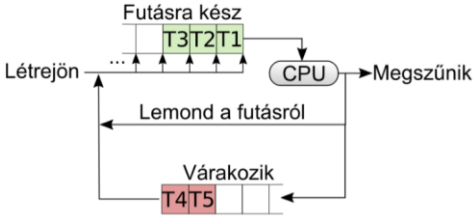
- *problémák*
  - optimális a várakozási, körülfordulási idő
  - taskok löketidejét becsülni lehet
  - előre ismert tasknál talán lehet számolni
- ***legrövidebb hátralevő idejű (Shortest Remaining Time First, SRTF)***



  - preemptív SJF
  - új task lép FK állapotba
  - összeveti a löketidejét a futó hátralevő idejével
  - ha az új task löketideje kisebb, akkor megszakítja a futó taskot
  - F → FK állapotba tud lépni

Commented [KC61]: Futásra Kész

- ***prioritás ütemező (PRI)***



- *tulajdonságai*
  - kooperatív és preemtív is lehet
  - kicsit bonyolultabb adatstruktúra - *rendezés*
    - $O(N)$ vagy $O(\log N)$ – *beillesztés*
    - $O(N^2)$ vagy $O(N * \log N)$ - *beillesztés*
  - futásra kész taskok prioritásuk szerint csökkenő sorrendben vannak

- *problémák*
  - rugalmas, de meg kell határozni a prioritás módját, számítási komplexitását
  - taskok nem egyenrangúak
  - *kiéheztetés (starvation)*
    - amikor egy taskot folyton megelőznek nála magasabb prioritásúak, így nem jut procihoz
  - elkerülés: *öregítés(aging)*
    - emeljük folyamatosan a prioritást, amíg futó állapotba nem kerül

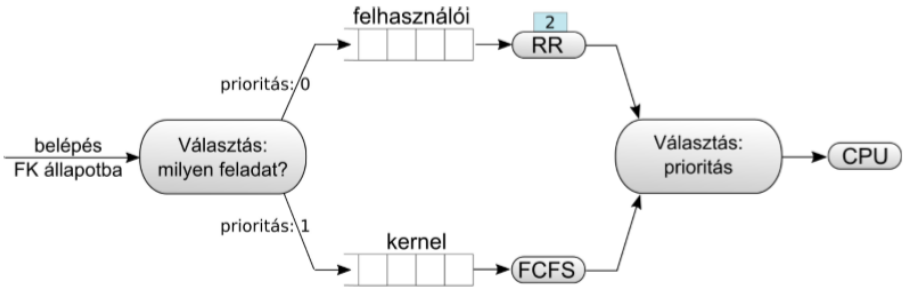
- ***külső prioritás***
  - task OS-en kívüli végrehajtási fontossága
  - függhet a feladat jellegétől, felhasználói szempontoktól
  - felhasználó korlátozottan befolyásolhatja
- ***belső prioritás***
  - task OS által meghatározott végrehajtási fontossága
- ***statikus prioritás***
  - task elindulása előtt rögzített, életciklusa alatt állandó
- ***dinamikus prioritás***
  - task életciklusa során időközönként kiszámított
  - újra kell rendezni az FK sort
  - → rezsiköltség, komplexitás ↑

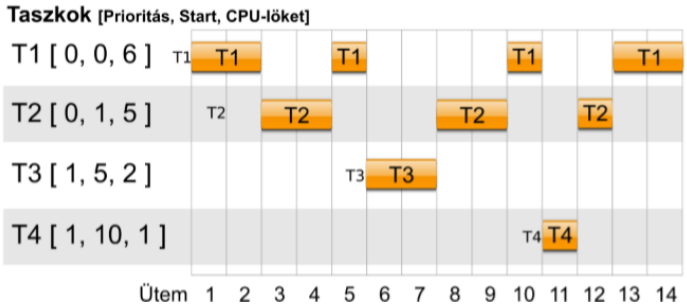
BACK

9. előadás:
Ütemezők II.

Commented [KC62]: Összetett ütemező algoritmusok.

1. Statikus többszintű sorok *(Static Multilevel Queues)*





- ***hozzárendelés***
  - taskok statikus módon való szinthez rendelés
  - nincs szintek közti lépés → statikus prioritás
  - adott ütemezési algoritmushoz hozzákötődés
- ***szintek kialakítása - algoritmusok***
  - valósidejű → FCFS, kooperatív
  - rendszerfeladatok → prioritás, kooperatív/preemptív
  - interaktív → RR, preemptív
  - kötegelt → SJF, kooperatív

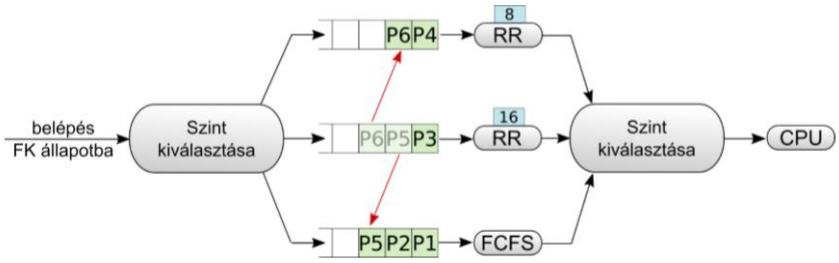
Commented [KC63]: Nagy CPU idejű, de nem időkritikus.

- ***szintek globális ütemezője***
  - preemtív

- ***adatstruktúra***
  - szintek ütemezőitől függ → FIFO, lista/fa
- ***algoritmikus komplexitás***
  - globálisan: $O(1)$ – konstans
  - többi szinttől függ
- ***rezsiköltség***
  - globálisan alacsony
  - többi szint ütemezőtől függ
- ***előny***
  - egyszerű
  - többféle szint – többféle algoritmus
- ***hátrány***
  - statikus
  - jelentkezhet kiéheztetés
  - taskok „jellemváltozása" nem kezelhető

Commented [KC64]: Kötegelt feladat időnként interaktív, rövid ideig elvárt, valósidejű működés stb.

2. <u>Dinamikus többszintű sorok *(Dynamic Multilevel Queues)*</u>

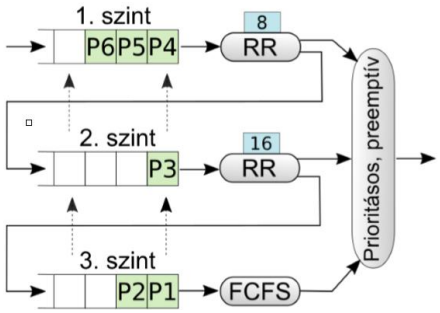


- ***hozzárendelés***
  - taskok dinamikus módon való szinthez rendelés
  - task prioritása dinamikusan változhat
- ***szintek kialakítása***
  - task mozoghat a sorok között
  - „fentebb" lépés *(upgrade)*: magasabb prioritási szintre lép
  - „lentebb" lépés *(downgrade)*: alacsonyabb prioritási szintre lép

- ***előny***
  - egyszerű
  - többféle szint – többféle algoritmus
  - használható öregítés
  - adaptálódhat a taskok változó viselkedéséhez
  - komplex, adaptív ütemezés
- ***hátrány***
  - upgrade/downgrade bonyolítja az ütemezőt
  - több számítás
  - számítás komplexitása, rezsiköltség ↑
  - gondosabb tervezés kell

Commented [KC65]: Dinamikus prioritások, beszúrás, átrendezés.

3. **Többszintű visszacsatolt sorok *(Multilevel Feedback Queue, MFQ)***

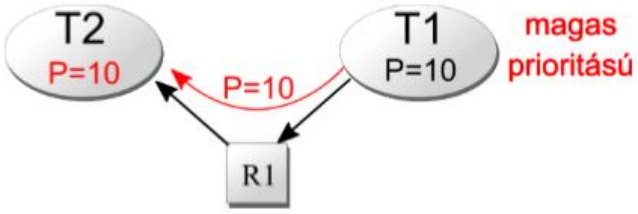


- ***hozzárendelés***
  - amelyik kihasználja az időszeletét, az lentebb lép
  - várakozó állapotba kerülő taskok fentebb lépnek
  - egyszerű megvalósítás, algoritmus
  - sok mai ütemező alapja
- ***szintek kialakítása***
  - CPU-intenzív taskok alacsonyabb prioritási szinten
  - I/O-intenzív taskok magasabb prioritási szinten

4. Ütemezési szintek

- ***kernel mód***
  - előre ismert feladatok *(statikus)*
    - kis CPU-löketek
    - hosszú I/O-löketek
    - → *kooperatív ütemező*
  - perifériakezelés
  - nincs konvoj-hatás
  - ha egyszerre több task FK, akkor nem mindegy a futási sorrend
  - minél kisebb rezsiköltség kell
    - *nem preemptív → statikus prioritás, egyszer kernel adatstrukútra védelem*
- ***felhasználói mód***
  - előre nem ismert feladatok *(dinamikus)*
    - lehetnek CPU-, I/O-intenzív
    - változó működésű taskok
  - konvoj hatás ellen kell védekezni
    - → *preemptív ütemező*
  - felhasználói preferenciák egy feladat fontosságára
    - → *prioritásos ütemező*
  - egyformán fontos feladatok hasonló eséllyel futtatás
    - → *RR ütemező*
  - ne legyen túl komplex

5. **Prioritásinverzió és kezelése**
   - ***prioritásinverzió***



   - ***kezelése → prioritásöröklés***
     - bonyolult függőség, nem triviális az öröklés
     - egy task több másiktól is függhet
       - túl messzire gyűrűzik a prioritásnövelés
     - más is várhat az erőforrásra
     - valósidejű működés esetén nem a legjobb megoldás

6. **Kernel preemptivitás**
   - ***előnyök***
     - jobb időkorlátok tarthatók → valósidejű működés
     - több task futhat egyszerre
   - magas delay-re hajlamos programágak feldarabolása
   - biztosítani kell a kernel adatstruktúrák konzisztenciáját a kernel módú taskváltáshoz
     - pl. Linux Low Latency kernel patch ~ Molnár Ingo
       - zenei alkalmazások problémái motiválták
   - átütemezési késleltetés: ***PDLT –*** *Process Dispatch Latency Time*
     - megszakítás és a kezelésére felébresztett folyamat első utasításának végrehajtása között eltelt idő
   - adatstruktúrát kell védeni
     - ha két kernel módú task ugyanazt módosítja, vagy az egyik olvassal, másik módosítja, akkor baj van

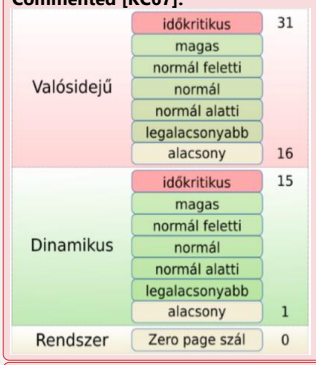
Commented [KC66]: Magyar Linux hacker, Red Hatnek dolgozik jelenleg. Javított a Linux teljesítményén, és biztonságán.

- megoldás: kritikus régiók védelme zárakkal (lock)
  - zárak és kulcsok
  - akinél a kulcs van, használhatja az adatokat
  - akinél nincs, az vár
    - pl. Linux 2.6+
    - Windows NT kerneltől kezdve
- *„kreempt" Linux patch by MontaVista*

7. Ütemezők a gyakorlatban

- ***Windows***
  - többszintű, prioritásos, időosztásos, preemptív, $O(1)$ szál-ütemező
  - kernel prioritási szintek
  - *Windows API*
    - folyamat prioritásosztályok
    - szál prioritásmódosítók
  - *prioritásemelés (Boost)*
    - eseménykezelés
    - I/O befejezés
    - UI esemény
    - éhezés elkerülése
    - prioritásinverzió kezelése
  - felhasználói prioritásemelés
- ***Linux***
  - *kernel v2.6*
    - $O(1)$ ütemező
    - prioritásos
    - visszacsatolt többszintű
    - preemptív, időosztásos
    - **szintek**
      - 0-99: valósidejű (statikus)
      - 100-139: időosztásos (dinamikus)

Commented [KC67]:



Commented [KC68]: A dinamikus tartományban.

- **időszeletek**
  - aktív *(active)*
  - lejárt *(expired)*
  - aktív → lejárt prioritás újraszámolása
  - ha aktív kiürült, csere a lejárttal

Commented [KC69]: Pointer művelet.

- *kernel 2.6.23-tól: CFS (Molnár Ingo)*
  - $O(1)$-et felváltó ütemező
    - szálak ütemez
    - kernel átütemezési pontok
    - teljesen preemptív kernel üzzemmód option
  - sor (lista) helyett piros-fekete fa
    - kisebb érték balra, nagyobb jobbra
    - $O(\log N)$ komplexitás
  - virtuális futási idő biztosítása
    - eszerint rendezi fába a taskokat
- *Solaris*
  - moduláris
  - szálalapú
  - teljesen preemptív
  - prioritásos
  - időosztásos

BACK

10. előadás:
Ütemezők III.

Commented [KC70]: Többprocesszoros ütemezők.

1. **Alapkérdések**
   - ***preemptivitás***
   - ***újrahívhatóság*** *(reentrancy)*
   - ***lokalitás***
     - F→FK→F állapotátmenetek (pl. megszakítás) maradványadatai
       - CPU regiszterek, Lx cache, RAM
   - ***erőforás-allokáció*** *(resource allocation)*
   - ***terheléselosztás*** *(load balancing)*
     - *egyenletes terhelés*
       - homogén végrehajtóegységek
     - *képességnek megfelelő terhelés*
       - heterogén rendszerek
   - ***erősen hardverfüggő feladatok***
2. **Hardver megoldások áttekintése**
   - ***egyprocesszoros, „többszálú” rendszerek***
     - 1 „szál” = 1 task
     - *többszörözött erőforrások*
       - utasításszámláló, tárolóregiszterek
     - *közös erőforrások*
       - TLB, utasítás cache, elágazásbecslő...
       - adataikat azonosítókkal szálakhoz kötik
   - ***párhuzamosított végrehajtás egyprocesszoros rendszerekben***
     - *finom felbontású (fine-grained) időosztásos*
       - minden órajelciklusban más task
       - kihasználja egy szál végrehajtásának kis szüneteit

Commented [KC71]: Translation Lookaside Buffer

- *durva felbontású (coarse-grained) időosztásos*
  - megakadás esetén vált taskot
  - kis időre megakad a végrehajtás
- *szimultán többszálú (**SMT**)*
  - szuperskalár CPU: több műveleti egység → több utasítás
  - szabad műveleti egységekre más taskok feladatai

Commented [KC72]: Pl. cache hiba

Commented [KC73]: Pipeline feltöltése.

- ***multiprocesszoros rendszerek***
  - több teljes értékű processzor
  - CPU-k közötti kommunikáció
    - közös memórián keresztül (monolitikus rendszer)
    - üzenetküldéssel (elosztott rendszer)
  - memóriaműveletek hatékonysága
    - **UMA** avagy SMP: azonos idővel használhatók
      - rosszul skálázódik
      - cache memória elérésben különbség
      - többmagos CPU
    - **NUMA**: nem mindegy hova írunk
      - memória egy részéhez direkt kapcsolat van
      - más részek kommunikációs hálózaton érhetők el
      - fontos a lokalitás
      - pl. multi-socket és több CPU-s rendszerek

Commented [KC74]: Uniform Memory Access

Commented [KC75]: Non-Uniform Memory Access

3. Lokalitástudatos ütemezés

- ***processzoraffinitás***
  - *laza affinitás (soft affinity)*
    - megpróbálja, nem garantált
    - mai OS-ekben jellemző
  - *kemény affinitás (hard affinity)*
    - garantált task – CPU párosítás
    - rendszerhívással és a felhasználói felületen állítható be
    - processzorhalmaz affinitás
      - task – CPU-halmaz

Commented [KC76]: Task és a végrehajtó processzor „kötődése”.

- programozó jobban ismerheti a feladatok, taskok jellemzőit, viszonyát → van, hogy jobban tud dönteni, mint a kernel

- ***memóriaaffinitás***
  - *NUMA architektúra*
    - CPU-memória elemek között hardver kapcsolat
  - kernel: transzparens módon kezeli
    - allokáció a végrehajtó CPU memóriatartományban történik
  - CPU-affinitással együtt állítható

4. <u>Terheléselosztás</u>

- OS-feladatok pl. ütemezés
- felhasználói feladatok nem ismertek, de lehetnek elvárások
- ***végrehajtóegység***
  - *egyforma*
    - feladatok tetszőlegesen szétoszthatók
  - *képességeikben különbözők (heterogén ISA)*
    - feladatok egy része csak adott egységen oldható meg
  - *teljesítményükben/fogyasztásukban különbözők*
    - taskok bárhol végrehajthatók
    - lehetnek preferenciák
      - felhasználó elvárásai
      - rendszer aktuális állapota
- ***asszimetrikus rendszerek***
  - összes kernelfeladat egy egységen, többin felhasználói feladatok
  - egyszerűen megvalósítható, programozható
  - kernel végrehajtója kihasználatlan lesz
  - heterogén CPU-architektúrákon jól jön

- ***szimetrikus rendszerek***
  - minden végrehajtóegység saját ütemezés
  - FK taskok lehetnek
    - egy közös sorban
    - egységenként külön sorban
  - jobb CPU kihasználtság
  - nehezebben programozható
  - mai OS-ekben
- ***FK taskok nyilvántartása***
  - *globális FK adatstruktúra*
    - globális döntések
    - processzoraffinitás megnehezül
  - *lokális ütemezési sorok*
    - processzoraffinitás OK
    - terhelés asszimmetrikus
      - taskok mozgatása FK halmazok között
      - affinitás sérülésből fakadó károk mérlegelése
    - `push`: kernel task
    - `pull`: CPU lokális ütemezője
  - *összetartozó/együttműködő taskok*
    - gang scheduler: taskok egy csoportja → végrehajtóegységek csoportjába
    - virtualizációs rendszereknél érdekes kérdés

5. <u>Heterogén többprocesszoros rendszerek</u>

- energiahatékonyság
- többféle funkció egy chipbe pakolás
- hardvergyorsító HPC alkalmazásoknál
- játékkonzolok rendszerei (CPU + GPU)

Commented [KC77]: High Performance Computing

- ***feladat-task-végrehajtóegység***
  - *homogén ISA*
    - szekvenciális → aszinkron
    - több taskból álló, konkurens rendszerek
    - egyre több végrehajtóegység
  - *heterogén ISA*
    - eltérő hatékonyság
    - nehéz, és adat-intenzív feladatok
    - több taskból álló, konkurens rendszerek
    - ilyen rendszerek pl
      - OpenMP
      - CUDA
      - PYNQ
    - a mai gépekben rengeteg dolog heterogén már
      - CPU
      - tárolórendszerek
      - hálózat
- ***példák heterogén rendszerekre***
  - a mai gépekben rengeteg dolog heterogén
  - → CPU
    - magok más-más órajelen
    - integrált GPU
  - → tárolórendszerek
    - CPU regiszterek
    - DRAM, SRAM
    - HDD, NAND/SSD
    - MRAM, STTRAM...
  - → hálózat
  - *AMD HSA + hUMA*
    - integrált GPU közös memóriával
    - pl. Play Station 4

Commented [KC78]: „Hitting the wall.”

Commented [KC79]: Heterogeneous System Architecture

Commented [KC80]: Heterogeneous Uniform Memory Access

- *Intel HD graphics*
- *AMD HSA + HUMA*
    - Intel MIC architektúra
        - Intel Xeon Phi saját memóriát használ, hálózaton érhető el
    - FPGA alapú
        - IBM CAPI
    - egyedi feladatra tervezett rendszerek
        - Google Tensor Processing Unit
    - GPGPU rendszerek
        - jellemzően dedikált memóriát használ

- ***feladatkezelés***
    - *CPU alapú*
        - taskok automatikus migrálása végrehajtóegységek között
        - OS itt csak órajelet skáláz
    - *OS kernel ütemező*
        - task migrálás, ha tudja a végrehajtók képességeit
        - heterogén, többprocesszoros ütemezés

Commented [KC81]: Heterogeneous System Architecture

Commented [KC82]: Heterogeneous Uniform Memory Access

Commented [KC83]: Many Integrated Core

Commented [KC84]: Pl. big.LITTLE végrehajtók között.

BACK

11. előadás:
Memóriakezelés

1. **Memóriakezelés**
    - ***feladata***
        - *kiosztja az erőforrást*
            - erőforrás: fizikai memória
            - igénylők: taskok, kernel
        - *elhelyezi a taskok adatait*
            - programkód + statikus adatok
            - dinamikusan allokált
        - *elhelyezei a kernel adatait*
            - programkód
            - adminisztratív adatok
        - *biztosítja a védelmet*
            - szeparáció hibák
        - *támogatja a kommunikációt*
            - adatcsere taskok között
    - ***kihívás***
        - *nem elég az erőforrás*
            - adatcsere taskok között
        - *hatékonyság*
            - minden CPU műveletet érint
        - *biztonság*
            - sok incidens forrása
    - ***megfigyelések***
        - *Neumann-architektúra*
        - *induláskor nincs szükség a teljes programra, adatkészletre*

- *dinamikus memóriafoglalás*
  - rendelkezásre álló fizikai memória méretével nem törődnek
  - az allokált memória „szellős”
- *lokalitási jellemzők*
  - időbeli, térbeli és algoritmikus
- *nem használt memóriarészek*
  - sokféle lehetséges lefutásból csak egy következő be
  - hibakezelés, ritkán használt funkciók
- *közösen használt*
  - pl. dinamikus rendszerkönyvtárak, folyamatklónozás

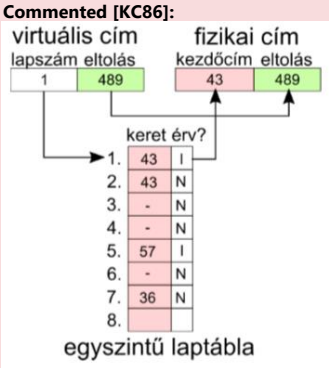
2. Virtuális tárkezelés

- Definíció
  - összefüggő, virtuális memóriatartomány a taskok számára
  - részekre bontva csak a használatban lévő részeit tároljuk
- ***működése***
  - *taskok lapokra bontása*
    - használatban lévő lapok memóriába, swapba
    - MMU beállítása: hardveres címképzés, védelmi funkciók (task szeparáció)
    - MMU által generált megszakítások kezelése (hiba esetében)
  - *taskok futása alatt*
    - TLB, MMU címfordít
    - hardver betartja a védelmi korlátokat
  - *megszakítások*
    - **védelmi hiba**: hibás címzés (érvénytelen, hozzáférhetetlen)
    - **laphiba**: a hivatkozott lap nincs a fizikai memóriában

- ***feladata (hardvertámogatással)***
  - *address translation*
    - taskok a CPU teljes címtartományát látják
      - virtuális címtartomány
      - pl. x86-64 eseténen 2^{84} byte = 256 terabyte
    - fizikai memória a fizikai címtartománnyal érhető el
      - gigabyte tartomány
    - **címleképzés lépései**
      - virtuális cím kettébontása
      - index → fizikai keret
      - fizikaicím előállítása
    - **címtér-elkülönítés**
      - taskonkénti laptábla, kontextus része
      - futó task esetén MMU támaszkodik a tartalmára
  - *paging*
    - fizikai memóriába nem fér el, átkerül a háttértárba
      - ameddig el nem shiftelődik a fizikai memóriáig, addig nincs használva
    - virtuális címtartomány → page
    - fizikai memória → frame
    - háttértár → block
    - laptábla: pages ↔ frames
    - TLB: címfordító gyorsítótár
  - *swap*
    - ami nem fér a kp-i memóriába, vagy felesleges ott tárolni
    - nagy kapacitású, lassú tároló
      - részekre bontott
      - lapok blokkokban
      - CPU közvetlen módon nem fér hozzá

Commented [KC85]: Címleképzés: virtuális, fizikai címek megfeleltetése (MMU),

Commented [KC86]:



Commented [KC87]: Lapszám index, eltolás.

Commented [KC88]: Fizikai keret kezdőcím, eltolás.

Commented [KC89]: Lapozás, lapszervezés: taszkok memóriatartományának részekre bontása.

Commented [KC90]: Gyors, fizikai memóriakapacitás kiterjesztése – cserehely avagy lapozófile

- használata (memóriakezelés feladatköre)
  - swapban tárolt adatok használata előtt fizikai memóriába betöltés
  - fizikai memóriában szabad helyre van szükség → adatok swapba mentése
- **swapping**
  - taskok teljes memóriatartományának háttértárra írása
  - memória és swap tördezettsége
  - töredezettség csökkentése
    - » kezelés: taskok áthelyezése (tördezettség-mentesítés)
    - » megelőzés: ügyesebb elhelyezési algoritmusok, lapozás
  - teljes lapozás tárcsere mellett
    - » overload esetén felfüggesztett taskok lapjai swapra

Commented [KC91]: Változó méretű lyukak, szabad helyek nehezen kitölthetőek.

- ***elvárásaink***
  - minél több task párhuzamosan
  - feleslegesen ne foglaljon erőforrást
  - a fizikai memóriát meghaladó igények kiszolgálása
  - szeparáció és együttműködés
  - alacsony rezsiköltség

3. Laphiba

- ***kezelés***
  - szoftveres címleképzés
- ***folyamat***
- 1) lap nincs a memóriában
- 2) laphiba
- 3) kernel észleli a laphibát → memóriakezelőt aktiválja (lap behozás)
- 4) lap benne van a swapban? → szabad keretbe tölti be
  - ha nincs szabad keret → fel kell szabadítani → lapcsere
- 5) task laptáblájába beállítja az új lap-keret összerendelést

- 6) laptábla frissít MMU számára
- 7) CPU ismét végrehajtja a műveletet sikeresen
- ***memóriakezelés feladatai***
  - szabad keretek biztosítása (ne laphiba alatt pls)
  - nem használt lapok háttértárra írása → keret felszabadít
  - nyilvántartás
    - taskok lapjai → laptábla
    - keretek → kerettábla
    - swap → diszk blokk leíró (*disk block descriptor*), swap térkép (*swap map*)
  - szükség esetén tárcsere → túlterhelés esetén teljes taskok kiírása
- ***memóriakezelés adatstruktúrái***
  - *kerettábla*
    - keret sorszámával indexelt
    - állapot
      - szabad
      - foglalt
      - módosult (dirty)
      - DMA alatt áll
    - hivatkozásszámláló (acc): hány task használja a keretet
  - *laptábla*
    - lap- és keretsorszám
    - jelzőbitek
      - valid
      - dirty
      - accessed
      - read-only
      - COW
    - jogosultságok

Commented [KC92]: MMU is módosíthatja.

Commented [KC93]: Dirty jelzőbitet akkor kap egy adat, mikor meg lett változtatva, de még nem lett kiírva egy külső helyre, ugyanis lehet, hogy utána is lesznek, még, akik változtatáson esnek át, és hasznosabb egyszerre kiírni őket, mint egyesével.

Commented [KC94]: Copy-On-Write

- állapot
  - memóriában
  - háttértáron
  - igény szerint kitöltendő
- task azonosítója
- másolás

▪ *diszk blokk leíró (kernel kontextus)*

- háttértár eszközazonosító
- blokk sorszám
- típus
  - swap
  - zero-fill
  - fill-from-text

○ ***teljesítménynövelő technikák***

▪ *fill-on-demand*

- malloc(): memória tartalma nem definiált → ⦰ keret, swapfoglalás
  - laptábla elemei fill-on-demand bejegyzést kaphatnak
- első hivatkozásnál MMU megszakít → címleképzés → allkoál és kitölt egy keretet → megtörétnik a memóriafoglalás
- programkód betöltésénél diszkblokk beállítás

▪ *COW*

- fork(): több task ua. a programkódot futtatja
- 1 keret – több lap, task
- olvasás nem gond, írás igen
- fork() és COW technika
  - laptábla duplikálás
  - hivatkozásszámláló növelés
  - read-only, COW jelzőbitek

- írásnál
  - » read-only miatt megszakítás
  - » megszakításkezelő látja a COW bitet
  - » duplikálja a lapot (allokálás)
  - » törli a lapok RO, COW bitjeit
  - » MMU újrapróbálja az írást

- ***PFF (Page Fault Frequency)***
  - task működését lassítja
    - futás megszakad → újraütemezés
    - memória-intenzív → I/O-intenzív
  - rezsiköltség, terhelés ↑ → újabb laphiba
  - vergődés (trashing)
    - gyakori laphibák miatt teljesítmény romlik
    - taskok számának kordában tartása (középtávú ütemezéssel) kezelhető → inkább elkerülés ajánlott

4. Lapozási stratégiák

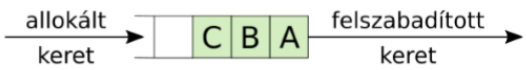
- OS max jósolni tud, hogy melyik lapot töltse be a memóriába, erre van két lehetősége
- ***igény szerinti lapozás (demand paging)***
  - *működés*
    - csak laphiba esetén fut → szükséges lapot hozza be
  - *értékelés*
    - egyszerű
    - korábban nem használt lapokra hivatkozás **mindig** laphibát generál
    - lassítja a task futását
- ***előretekintő lapozás (anticipatory paging)***
  - *működés*
    - „előre dolgozik” → több lapot hoz be
    - → megpróbálja kitalálni, mely lapokra lesz szükség
      - lokalitási jellemzőkből
      - laphibák a múltból

- *értékelés*
  - **jó becslés → kevesebb laphiba**
    - korábban nem hivatkozott lapok fizikai memóriában
    - kevesebb megszakítás, I/O átütemezés
  - **rossz becslés → több laphiba**
    - nem megfelelő lapokat tölt be → laphibák száma ↑
    - sok felesleges lap fizikai memóriában
    - kevesebb szabad keret
- **keretek felszabadítása - lépések**
  - laphiba-megszakítás
  - felszabadítandó keret kiválasztása lapcsere algoritmussal
  - keret tartalmának mentése swapre
  - kerethez tartozó laptábla-bejegyzés módosítása és a keret felszabadítása
  - keret új tartalmának betöltése a swapről (vagy előállítása)
  - új laptábla-bejegyzés készítése
  - MMU beállít
  - visszatérés a megszakításból

5. Lapcsere algoritmusok

Commented [KC95]: Mikor allokálták a lapot?
Milyen sorrendbe?
Használták-e a lapot mostanában?
Módosult-e a jelzőbit/keret tartalma? Ha igen, hosszabb lehet a felszabadítás
Választás jellege?
- aktuális task címteréből: lokális
- összes lap közül: globális

- melyik keret tartalma íródik ki a cserehelyre...
- ***FIFO lapcsere***



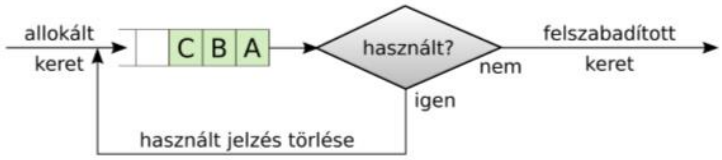
Feladatmegoldás: FIFO lapcsere 4 kerettel

Lapkérés:	1	2	3	4	1	2	5	1	2	3	4	5
Keretek A	1				1		5				4	
Keretek B		2				2		1				5
Keretek C			3						2			
Keretek D				4						3		
Foglalás:	A1	B2	C3	D4	-	-	A5	B1	C2	D3	A4	B5
FIFO	A	A	A	A	A	A	B	C	D	A	B	C
		B	B	B	B	B	C	D	A	B	C	D
			C	C	C	C	D	A	B	C	D	A
				D	D	D	A	B	C	D	A	B
Ütem	1	2	3	4	5	6	7	8	9	10	11	12

- *tulajdonságai*
  - FIFO algoritmus, adatstruktúra
  - előrenéző
- *komplexitás*
  - $O(1)$
- *rezsiköltség*
  - minimális
- *előnyök*
  - könnyű megvalósítani
- *hátrányok*
  - lapok jövőbeli használatát gyengén becsli
  - nem figyeli a módosítást
  - ha ↑ a rendelkezésre álló keretszámot
    - laphibák száma ↑
    - **Bélády-féle anomália**

Commented [KC96]: Bélády László, IBM

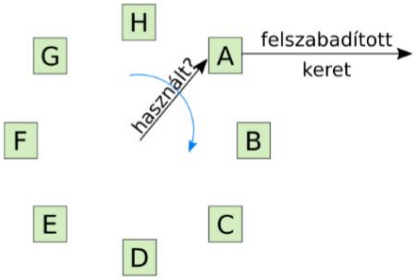
- ***SC lapcsere (Second Chance)***



- *tulajdonságai*
  - FIFO algoritmus, adatstruktúra
  - hátranéző
  - használt lapokat nem szabadítja fel
- *komplexitás*
  - $O(1)$
- *rezsiköltség*
  - minimális
- *előnyök*
  - könnyű megvalósítani
  - jobban becsül a FIFO-nál

- *hátrányok*
  - állandóan mozgatja az adatokat a FIFO-ban
  - nem figyeli a módosítást

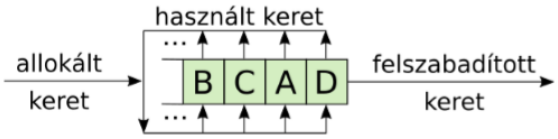
- ***Clock lapcsere***



Óra (clock) lapcsere

- *tulajdonságai*
  - nem bonyolult algoritmus, adatstruktúra
  - hátranéző
  - használt lapokat nem szabadítja fel
- *komplexitás*
  - $O(1)$
- *rezsiköltség*
  - minimális
- *előnyök*
  - könnyű megvalósítani
  - jobban becsül a FIFO-nál
  - megspórolja az adatok mozgatását FIFO-ban
- *hátrányok*
  - nem figyeli a módosítást

- ***LRU lapcsere (Least Recently Used)***



  - *tulajdonságai*
    - láncolt lista
      - lapok hatékonyságuk szerint sorba
      - használati idő, hivatkozási szám szerinti
    - hátranéző
    - használt lapokat nem szabadítja fel
  - *komplexitás*
    - $O(N)$
  - *rezsiköltség*
    - hardvertámogatással kicsi, anélkül nagy
  - *előnyök*
    - legjobban becsli a lapok jövőbeli használatát
    - nyilvántartja a lapok múltbeli használatát
  - *hátrányok*
    - frissen behozott lapot nagy eséllyel lecseréli
    - nagy hardvertámogatás kell
    - nem figyeli a módosítást
- ***LFU lapcsere (Least Frequently Used)***
  - *tulajdonságai*
    - LRU egyszerűsített változata
    - számláló alapján választ felszabadítandó keretet
  - *rezsiköltség*
    - közepes
    - periodikusan igényli a számlálók növelését

Commented [KC97]: Legrégebben nem használt.

Commented [KC98]: Nyilvántartja a lapok múltbeli használatát.

Commented [KC99]: Legkevésbé használt.

- *előnyök*
  - laplopó taskkal jól kombinálható
  - jól becsli a lapok jövőbeli használatát
- *hátrányok*
  - nagy hardvertámogatás kell
  - nem figyeli a módosítást
  - frissen behozott lapot nagy eséllyel lecseréli, ami nem jó
  - számláló túlcsordulhat → öregítés
  - nem tud különbséget tenni a módosított és változatlan lapok között → lapcsere diszk → lassú

- ***NRU lapcsere (Not Recently Used)***

  Commented [KC100]: Mostanában nem használt.

  - *tulajdonságai*
    - SC finomított változata
    - `referenced` + `dirty` jelzőbit módosulás figyelése
    -

      Commented [KC101]:
      ```
      ref=0 dirty=0 → a prioritás
      ref=0 dirty=1 → a prioritás
      ref=1 dirty=0 → a prioritás
      ref=1 dirty=1 → a prioritás
      ```
  - *rezsiköltség*
    - kicsi
    - jól kihasználja a hardvertámogatást
  - *előnyök*
    - SC-nél jobban becsli a lapok jövőbeli használatát
      - hivatkozások mellett a módosításokat is figyeli
    - különbséget tesz a módosított és változatlan lapok között
  - *hátrányok*
    - nagy hardvertámogatás kell
    - nem figyeli a módosítást
    - frissen behozott lapot nagy eséllyel lecseréli, ami nem jó
    - számláló túlcsordulhat → öregítés
    - nem tud különbséget tenni a módosított és változatlan lapok között → lapcsere diszk → lassú

- **Page locking – lapok tárba fagyasztása**
  - frissen behozott lap jövője nem becsülhető
  - könnyen kiírásra választhatja őket a lapcsere

- *I/O művelete alatt álló lapot ne szabadítsunk fel*
  - fizikai címeket használ
  - CPU-t megkerülve, DMA vezérlő segítségével is módosíthatják a memóriát
- **megoldás: lapok tárba fagyasztása**
- page lock bit jelzi a zárolt (fagyasztott) állapotot
- nem lapcserézhetők
- I/O művelet esetében végig zárolás
- első hivatkozás feloldja a zárolást

Commented [KC102]: Lapcserének erre is figyelnie kell.

- **laplopó task**
  - nevei
    - Page deamon
    - kswapd
    - Working Set Manager
  - **feladata: üres keretek biztosítása**
    - időközönként felébred/felébresztik
    - szabad keretek száma határérték között legyen
      - minimum szint alatt, elkezd kereteket „lopni”
      - maximum szintnél alvó állapotba lép
    - referenced bit törlése
    - használati számlálók öregítése
  - végezheti az összes lapcserét, ha nem elég ügyes

Commented [KC103]: Elfogytak a szabad memóriakeretek.

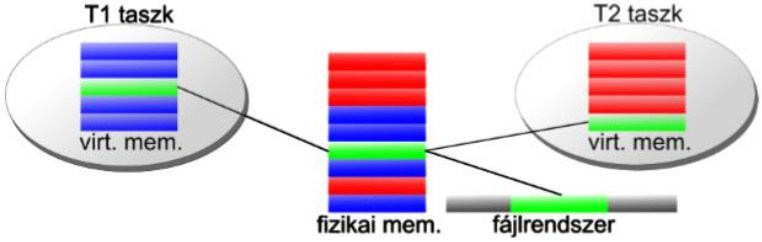
BACK

12-13. előadás:
Taskok kommunikációja

1. **Kommunikáció alapvető sémái**
    - *közös memória*
        - PRAM modell
        - versenyhelyzet
        - megvalósítási példák
            - folyamaton belüli szálak
            - POSIX osztott memória
    - *üzenetváltásos*
        - adatátviteli rendszer
        - megvalósítási példák
            - hálózati kommunikáció
            - távoli eljáráshívás
            - elosztott rendszerek
            - mikrokernel
2. **PRAM: Pipelined RAM**
    - taskok párhuzamosan közös memóriaterületen
        - egymástól függetlenek
        - ütközhetnek
    - *ütközés szabályai*
        - *olvasás – olvasás – mindkettő a memória tartalmát adja vissza*
        - *olvasás – írás – olvasás régi/új értéket adja vissza*
        - *írás – írás – két érték valamelyike kerül memóriába*
    - *szabályok hatása*
        - műveletek nem hatnak egymásra
        - (pipelined: sorba rendezett)
        - párhuzamos kérések sorrendje nem definiált
            - ezek összehangolva: **szinkronizáció**

- ***folyamaton belüli szálak***
  - *adatcsere globális változókkal*
    - OS közös memóriát biztosít a szálaknak
    - kommunikáció nem OS fennhatóság alatt → programozóé
- ***SHM (Shared Memory)***

  - *értékelés*
    - ☹ rezsiköltség, rendszerhívás
    - zero copy, szuper gyors kommunikáció
    - korlátos kapacitás
  - *megvalósítás*
    - szabvány: POSIX Shared Memory
    - UNIX, Windows felhasználói címtérben pl. C++ lib
- ***memóriába ágyazott file-elérés***



  - *értékelés*
    - lehet ilyen az SHM, ahol az OS nem támogatja
    - klasszikus fájlrendszer interfész helyett is jó
  - *megvalósítás*
    - széles körben: nmap()
    - OS virtuális memóriakezelője végzi a leképezést
    - sokféle programozási környezetben elérhető

- ***teljesítmény***
  - virtuális memóriakezelésre támaszkodik
    - ⦸ felesleges kernel réteg, adatmozgatás, extra rezis, +késleltetés
    - → nagy adatátviteli sebesség

3. <u>Üzenetváltásos kommunikáció</u>

- ***alapvető címzési módszerek***
  - *direkt címzés*

  - *indirekt címzés*

  - *aszimmetrikus címzés*

  - *többszörös (multicast, broadcast)*
- ***szinkronitás***
  - *szinkron adatátvitel*
    - egyszerűen programozható
    - eredmény, esetleges mellékhatások beérkeznek
    - megszakad a task futása, átütemezés
    - DOS támadások, időtúllépés

- *aszinkron adatátvitel*
  - műveletek nem blokkolnak
  - task tovább futhat → műveletek eredménye nem érhető el, ellenőrizni kell, nem kézbesített üzeneteket átmenetileg tárolni

- ***adatátviteli szemantika***
  - *copy semantics*
    - küldő, fogadó saját példánnyal
    - módosítások hatása lokális
  - *share semantics*
    - ugyanazt használják, jogosultság lehet különböző
    - módosítás hatása globális (szinkró)
    - adatmozgatás ↓
  - *move semantics*
    - küldő elveszíti a hozzáférését az adatokhoz, amikor elküldi őket
    - megvalósítható megosztással, jogosultságok elvételével
- ***adatbirtoklás***
  - szinkronizáció
  - munkamegosztás
- ***direkt és aszimmetrikus kommunikációs megoldások***
  - *socket communication*
    - értesítés eseményekről: task → task, kernel → task
  - *remote procedure call*
    - másik task programjában levő eljárás meghívása
    - aszimmetrikus (kliens-server)
    - adatkonverzió is
- ***jelzések***
  - *jelzés*
    - értesítés eseményekről: task → task, kernel → task
  - *cél*
    - értesítés eseményekről

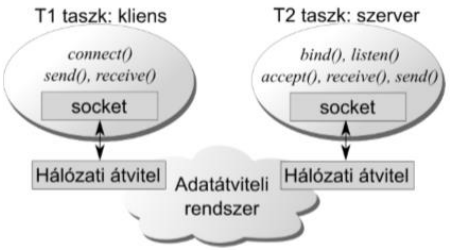
Commented [KC104]: Távoli eljáráshívás.

- *kétfázisú működés*
  - keltés
  - kézbesítés
- *típus*
  - felhasználói
  - rendszer
    - kivételek, értesítések, riasztások
- *kezelés*
  - kezelőfüggvény
  - figyelmen kívül hagyás

- ***teljesítmény***
  - infrastruktúrára épít
    - mozgatja, konvertálja, átmenetileg tárolja az adatokat
  - késleltetést okoz → adatátviteli sebesség ↓
    - *probléma mikrokernelben, HPC-ben*

4. <u>Hálózati kommunikáció</u>

- *címzés és hálózati protokollok*
  - gépen belül (`localhost, 127.0.0.1 ill. ::1`) és gépek között
  - többféle címzés (*cast) és protokoll
- ***csatorna leírója: socket*** *(hálózati csatoló)*
  - kommunikációs végpont (logikai) azonosítója a taskokban



- ***távoli eljáráshívás (RPC – Remote Procedure Call)***

Commented [KC105]: Pl. Open Network Computing (korábban Sun) RPC

- ***indirekt kommunikációs megoldások***
  - *mailbox*
    - véges számú üzenet
    - méret korlátos
    - postaláda címezhető, nem a fogadó (indirekt)
  - *message queue/passing*
    - méret korlátos
    - indirekt
    - sokféle implementáció
      - *RabbitMQ*
      - *Java MS*
      - *MSMQ*
    - többféle szabvány: *POSIX üzenetsorok, AMQP, MQTT, HPC, MPI* stb.
  - *pipe*
    - végtelen adattovábbítás
    - folytonos adatküldés, fogadás
    - csővezeték címezhető, nem a fogadó
    - egyszerre több vevő is

5. Teljesítménynövelés

- ***késleltetés és lassulás oka***
  - *rendszerhívások*
    - megszakítás
    - kontextusváltás
    - átütemezés
  - OS védelmi mechanizmusai miatt hatékonyság ↓

- ***modern mikrokernelek***
  - *kritikus a problémák megoldása*
    - minél kevesebb adatmásolás, kontextusváltás, átütemezés
  - *kontextusváltás, ütemezés rezsiköltség ↓*
    - *direkt kontextusváltás*
      - nem az ütemező dönt a következő futtatandó taskról
      - Küld() – Fogad() séma határozza meg az átütemezést
    - *lazy queueing*
      - Küld() – Fogad() párbeszédek kezelése
      - átmenetileg felfüggeszti a taskok állapotváltozásának adminisztrációját
- ***adatátvitelek***
  - *nagyon kevés adat (< ~ 16 byte)*
    - processzor regisztereiben (nincs sok ilyen)
    - teljes kommunikáció gyorsulása
  - *közepes adathalmaz (kb. 16-64 byte)*
    - virtuális regiszter tároló
      - elférhet a regiszterekben
      - erre a célra allokált memóriaterületen PRAM
    - hardverfüggő módon implementálja a kernel
  - *nagyobb adathalmaz*
    - **SHM átmeneti tárolással**
      - Küldő → SHM → Fogadó
      - korlátozott méret
      - 2 másolás (*Copy-in* / *Copy-out*)
    - **Single-copy**
      - task-task memóriamásolás (pl. KMEM)
      - Küldő → Fogadó direkt másolás rendszerhívással

Commented [KC106]: Kernel üzenetalapú kommunikációja miatt.

Commented [KC107]: Nem fut az ütemező (nincs rezsiköltsége).
Kicsi lesz a késleltetés a küldés és vétel között.

Commented [KC108]: Taskok nem mozognak a Fut, FK és Vár sorok között.
ha véget ér a párbeszéd, helyreállítja a sorok tartalmát.

Commented [KC109]: ARM11: 10%, CortexA9: 4%, x86-on ronthat is.

Commented [KC110]: Pl. ARM

- **Zero-copy**
  - lapmegosztás (pl. XPMEM)
    - task megoszthat memóriatartományt másokkal
  - távoli memóriaelérés (pl. CMA)
    - task elérheti egy másik memóriatartományát
- **hardvertámogatással (*kernel DMA Engine, RDMA)***
  - számítási csomópontok között is működhet
  - pl. HPC, SMP környezet

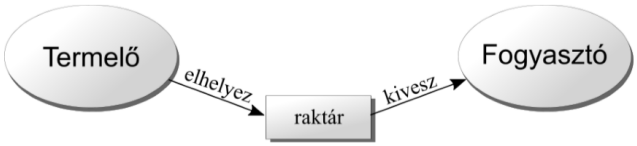
6. <u>Taskok közötti kommunikáció a gyakorlatban</u>

- lásd: előadás

BACK

14. előadás:
Szinkronizáció

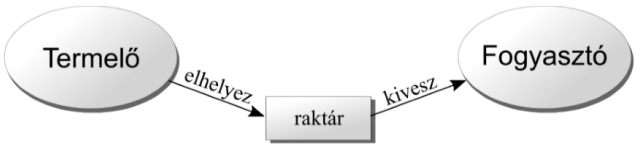
1. **Taskok együttműködése, versengése**
   - ***együttműködő taskok***
     - részekre bontott feladat
     - kooperálnak
       - adatcsere, végrehajtási függőségek
       - versenyhelyzet
     - OS védelmi mechanizmusa nehezíti
   - ***független taskok***
     - aszinkron végrehajtás
     - multiprogramozott rendszer
       - közös erőforrás-használat
       - versenhelyzet
       - végrehajtási függőségek
     - OS erőforrás mechanizmusa könnyíti
     - Példa:



       - **párhuzamosan működnek**
         - különböző ütem
         - eltérő sebesség
       - **megoldandó feladatok**
         - raktár konzisztencia (PRAM modell)
         - fogasztó blokkolás üres raktár esetén
         - termelő blokkolás tele raktár esetén

2. Taskok szinkronizációja

- Definíció
  - taskok működésének összehangolása a művelet-végrehajtás időbeli korlátozásával
- ***célja***
  - versenyhelyzetekben: konzisztencia
    - közös memória védelme
  - együttműködésben: műveleti sorrend
    - előidejűség (precedencia) biztosítása
- ***ára***
  - teljesítmény ↓
- ***formái***
  - *kölcsönös kizárás (mutual exclusion)*
    - taskok egymást kizáró működése
    - **cél:** erőforrás-védelem, versenyhelyzetek kezelése
    - **pl.:** osztott memória védelme, közös erőforrások használata
  - *egyidejűség (randevú)*
    - taskok megadott műveletei egyszerre kezdődjenek el
    - **cél**: összehangolt működés
    - **pl**.: blokkoló, nem pufferelt üzenetküldés
  - *előírt végrehajtási sorrend (precedencia)*
    - taskok adott műveletei meghatározott sorrendben hajtódjanak végre
    - **cél**: műveleti sorrend betartása
    - **pl**. termelő-fogyasztó együttműködés
  - Példa:



- **kölcsönös kizárás**
  - raktár konzisztenciája
  - egyszerre csak 1 módosíthajta
- **precedencia**
  - termék elkészít → felhasznál
    - » fogyasztó vár, amíg a raktár üres
  - termékkivétel teli raktárból → termék elhelyez raktárba
    - » termelő vár, amíg a raktár tele

3. Szinkronizáció megvalósítása

Ha $|T_C| \leq N - 1$
$|T_C|$++

$|T_C| \leq N$

$|T_C|$--

- Definíció
  - *kritikus szakasz*
  - utasítások egy olyan sorozata, amelyet egy időben a taskoknak csak egy korlátozott halmaza (TC) hajthat végre, azaz $|TC| \leq N$
- ***működési szabályok***
  - ha $|TC| < N$ akkor a Belépésre váró (**BV**) taskok közül egy beléphet a kritikus szakaszba
  - BV taskot csak véges számú másik előzhet meg a belépésben
  - kritikus szakaszban levő task csak véges számú utasítást hajthat végre
- ***hardver támogatása***
  - *egyszerű megoldás: megszakítások letiltása a kritikus szakaszban*
    - nincs átütemezés, taskváltás
    - egyprocesszoros esetben
    - fontos eseményekről lemarad a rendszer
  - *jó megoldás: atomi adatműveletek*
    - **test-and-set lock (TSL)**
      - beállítja egy bit (zár) új értéket IGAZ-ra és visszaadja a régit

Commented [KC111]: Atomi = nem megszakító

- **compare-and-swap (CAS)**
  - egy változó (zár) meghatározott értékű (a), akkor módosítja (b-re), régi értékével (a) tér vissza
  - *„spinning lock”:* a zárra várva végtelen ciklusban „teker” a program, task: „busy waiting”
  - → jobb lenne egy blokkoló művelet, hogy a task várakozó állapotba kerüljön

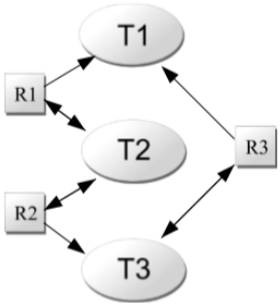
- ***zárolási eszközök áttekintése***
  - *lock bit*
    - egybites
    - oszthatatlan művelettel rendelkező zárolási eszköz
    - pl. TSL
  - *mutex (mutual exclusion lock)*
    - kritikus szakasz védelmére alkalmazott zárolási eszköz
    - pl. lock bit
    - blokkolja a taskot → kontextusváltás
  - *szemafor*
    - atomi műveletekkel rendelkező változó
    - várakozás: **P**
    - továbbengedés: **V**
  - *spinlock (spinning lock)*
    - lock, mutex vagy szemafor, amely aktívan várakozik („busy waiting”)
    - rövid kritikus szakaszok esetén kiváló (kontextusváltás spórol)
  - *ReaderWriterLock*
    - tetszőleges számú olvasó beléphet a kritikus szakaszba (reader lock)
    - ha író lép be (writer lock) → blokkolódik, amíg az összes olvasó ki nem lép
  - *RecursiveLock*
    - zárat birtokolja, blokkolás nélkül
    - rekurzív programoknál hasznos

4. Szemafor

- Definíció
  - két atomi művelettel támogatott adattípus
- ***értékkészlete***
  - **bináris** (mutex): {0, 1}
  - **számláló típusú**: nemnegatív egész számok {0, 1, …}
- ***műveletei***
  - **P(S)**: vár, amíg a szemafor értéke eggyel csökkenthető lesz (>0)
    - lehet blokkoló
  - **V(S)**: eggyel növeli a szemafor értékét
    - nem blokkoló művelet
- ***probléma***
  - *író-olvasó*
    - közösen használt változó (V) konzisztenciájának biztosítása (PRAM)
    - egypéldányos erőforrás védelme
  - *többszörös olvasó*
    - sok olvasó kevés író → szemafor túl sok várakozás
    - felesleges blokkolni az olvasókat, változó értéke nem módosul
  - *összetettebb kölcsönös kizárási példa*
    - több erőforrás együttes védelme

5. Holtpont (deadlock)

- Definíció
  - taskok egy *H* halmazában található valamennyi task olyan eseményre vár, amelyet csak *H*-n belüli taskok idézhetnek elő




```
T1          T2          T3
P(R3);      P(R1);      P(R2);
P(R1);      P(R2);      P(R3);
V(R1);      V(R2);      V(R3);
V(R3);      V(R1);      V(R2);
```

  - holtpont kialakulás
    - pl.: először lefut T1, T2 és T3 később indul
    - pl.: T1 fordítva foglalja le és szabadítja fel az erőforrásokat
- ***kialakulás feltételei***
  - *kölcsönös kizárás*
    - legyenek kizárólagosan használható erőforrások
  - *foglalva várakozás*
    - valamelyik task egy erőforrást foglalva másikra várakozik
  - *nincs erőszakos erőforrás-elvétel*
    - a taskok önszántunkból mondanak le erőforrásról, nem veszik el tőlük
  - *körkörös várakozás*
    - létezik taskoknak egy olyan T_N a T_1 sorozata, amelyre igaz az, hogy T_i a T_{i+1}által birtokolt erőforrásra vár
    - $(1 \leq i \leq N)$, és T_N a T_1 által foglaltra vár
    - rendszer állapotát az erőforrásfoglalási-gráffal modellezhetjük

- ***kezelése***
  - *„strucc algoritmus”*
    - holtpont esélye kicsi
    - nem okoz kritikus problémát
  - *megpróbáljuk kezelni*
    - detektálás
    - feloldás
      - taskok leállítása
      - erőforrás elvétel
  - *védekezés*
    - holtpontmentesre tervezzük
    - futásidőben ellenőrizzük a foglalásokat
      - kialakulása előtt detektáljuk
      - biztonságos állapot: holtpont nélkül erőforrást allokálhatunk
      - ⦸ erőforrás-foglalás → ⦸ holtpont
    - **egypéldányos erőforrásoknál**
      - kört keresünk az erőforrás-allokációs gráfban $O(N^2)$
      - előrejelzés: jövőbeli igényeket is vizsgáljuk
      - ha kör alakul ki → nem biztonságos → igény elutasítva
    - **többpéldányos erőforrásoknál**
      - bankár algoritmus

6. <u>Zárolás</u>

- ***problémái***
  - *prioritásinverzió*
    - alacsony prioritású task birtokol egy erőforrást
    - magasabb prioritású várakozik
    - **feloldása**: örökölt prioritásokkal
  - *kiéheztetés foglalással*
    - task folyamatosan foglal egy erőforrást

- erőforrásra várók blokkolódnak
- **feloldása**: a hibás működés javítása

▪ *kiéheztetés várakoztatással*

- nem FIFO várakozás esetén
- várakozási sorban „ragad” egy task
- **feloldása**: pl. öregítéssel

○ ***pesszimista zárolás***

▪ mindig véd

○ ***optimista zárolás***

▪ nem zárol, de detektálja → korrigálja a hibát

▪ *tranzakció-alapú megvalósítása*

- BEGIN: feljegyzi a kiinduló állapotot
- MODIFY: végrehajtja a műveleteket
- VALIDATE: ellenőrzi, hogy valami meghiúsítja-e a műveletek konzisztenciáját
- COMMIT: ha nincs gond, akkor zárja a műveleteket
- ROLLBACK: ha problémát észlel, akkor visszalép a kiinduló állapotba

▪ *értékelése*

- kevés konfliktus esetén javul a teljesítmény
- sok hiba esetén jelentősen romlik

▪ *megvalósítása*

- tranzakciós memória
- adatbáziskezelők, programozási nyelvek

○ ***zárolásmentes algoritmus*** *(lock-free)*

▪ az erőforráson mindig történik valamilyen „hasznos” művelet

▪ garantálja az erőforrás teljes kihasználtságát

▪ ⦸ teljesítményvesztés

○ ***várakozásmentes algoritmus*** *(wait-free)*

▪ zárolásmentes + minden művelet véges időn belül végrehajtható

▪ sokkal nehezebb megvalósítani

BACK

15. előadás:
File- és tárolórendszerek I.

1. **File-rendszer (felhasználói szemmel)**
    - ***adminisztrátor***
        - helyi és távoli fájlrendszerek használatba vétele
        - teljesítményhangolás
        - helyfoglalás ellenőrzése és felszabadítása
        - biztonsági másolatok készítése
    - ***programozó (alkalmazásfejlesztő)***
        - programozói interfészek
            - rendszerhívások – és könyvtárak
        - fájlleírók – műveletek, zárolás
    - Definíció
        - ***file, állomány***
            - adattárolás logikai egysége
        - ***könyvtár*** *(directory)*
            - a szervezés logikai egysége
            - file-ok és könyvtárak halmaza
        - ***kötet*** *(volume)*
            - fájlok és könyvtárak tárolásának logikai egysége
            - fizikai tárolási egységhez (pl. partíció) rendelhető

Logikai

Fizikai

        - ***file system***
            - file-ok és könyvtárak fizikai tárolása és szervezése
        - ***partíció***
            - háttértár szervezési egysége
            - file system tárolására képes

- ***file system logikai szervezése***
  - *irányított fával reprezentálható*
    - csomópontok: file
    - élek: tartalmazás reláció
    - gyökér csomópont: kötethez rendelt elem
  - *elérési út (path)*
    - csomópont elérési helye
    - **abszolút**: a fa gyökerétől kezdve
    - **relatív**: egy másik csomóponttól
  - *fa bővítése irányított gráffá*
    - **rögzített link** - több fájl ugyanarra az adatra hivatkozik
    - **szimbolikus link** - másik fájlrendszeri elemre mutat
- Példa:
  - **Windows**
    - fizikai tárolók logikai meghajtókhoz rendelve
    - boot meghajtó (jellemzően `C:`) a kiinduló pont
    - `\Program Files` a telepített alkalmazások (x86: mindegyik, x64: 64 bites)
    - `\Program Files (x86)` a telepített 32-bites alkalmazások x86 esetben
    - `\ProgramData` az alkalmazások felhasználófüggetlen adatai
    - `\Users` felhasználói könyvtárak (adataik, fájljaik, programok egyedi adatai)
    - `\Windows` az operációs rendszer saját fájljai, könyvtárai
    - további meghajtók, partíciók
    - hálózati file systems
  - **UNIX/Linux**
    - könyvtárakhoz rendelt fizikai tárolók - egyetlen összefüggő gráf
    - gyökér: / avagy `ROOT`

- **Android**
  - Unix-szerű, de eltérő könyvtárak
  - nem triviális megnézni a teljes gráfot

- ***UNIX hozzáférési jogosultságok***
  - *POSIX*
    - 3 x 3 bit: { tulajdonos, csoport, mások } x { olvasás, írás, futtatás }
    - könyvtárak használatához olvasás és „futtatás” is kell
  - *speciális jogosultságok: SETUID, SETGID, StickyBit*
    - SETUID/GID: futási tulajdonos/csoport beállítása
    - StickyBit: csak a tulajdonos törölhet
  - *POSIX ACL (Access Control List)*
    - rugalmasabb, többféle jogosultság egyidőben
    - pl. `ls`

- ***adminisztrátori alapfeladatok***
  - *file system létrehozása (format)*
  - *csatlakoztatás (mount)*
  - *ellenőrzés, hangolás*
  - *biztonsági mentés*
    - **adatvesztés oka**
      - meghibásodás
      - felhasználó
      - kártevő
    - **jellege**
      - korlátozott
      - teljes
    - **mentés (backup)**
      - automatizált/kézi
      - részleges/teljes
      - szalag/disc/net
    - **visszaállítás**
      - „bare metal” / reinstall + restore

- **file rendszerek példa**
  - ***FAT32***
  - ***NTFS*** *(Windows)*
  - ***UFS, Berkely FFS*** *(BSD UNIX)*
  - ***ext2, 3, 4*** *(Linux)*
  - ***XFS*** *(RedHat Linux 7)*
  - ***HFS+, APFS*** *(Apple)*

2. File-rendszer (programozói szemmel)

- ***file megnyitás***
  - cél lokalizálás
  - metaadatok beolvasása
  - nyitott file objektum létrejötte
    - megnyitási mód
    - file pointer
    - file metaadatok
    - lehetséges műveletek
  - file descriptor
- ***file olvasás, írás***
  - fájlleíró azonosítja az objektumot
- ***file bezárás***
  - kernel megszünteti a létrehozott adatstruktúrákat
- ***file zárolása (= kölcsönös kizárás)***
  - fileműveletekkel
  - holtpont kialakulhat
  - *ajánlott zárolás (advisory locking)*
    - OS eszközöket biztosít, nem kényszeríti ki
    - taskoknak is opcionális
    - pl. Java `FileLock()`, Unix `flock()`

Commented [KC112]: Metaadatok kernelben

- *kötelező zárolás (mandatory locking)*
  - kernel mechanizmusok biztosítják
  - rendszerhívások kikényszerítik
  - pl. Windows
- *részleges (tartományi) zárolás*
  - pl. Windows `LockFileEx()`,

- ***file-ok megosztott elérés memórián keresztül (mmap)***
  - többszörös hozzáférés, konzisztencia és kölcsönös kizárás
  - fájlműveletek helyett is jó, ha sok direkt elérésű olvasást végzünk.
- ***I/O műveletek***
  - *nem blokkoló*
    - rendszerhívás visszatér adattal vagy hibával
  - *aszinkron*
    - beállítjuk a műveletet, adattároló puffert
    - elküldjük a kérést
      - háttérben elindul a kiszolgálás
      - rendszerhívás visszatér
    - task fut tovább
    - eredmény jelzése
    - eseménykezelő kezeli az adatokat

3. Belső működése

- Definíció
  - ***metaadat***
    - partíciók típusai és elhelyezkedése
    - fájlrendszerek, file-ok leírói (név, méret, location...)
  - ***adat***
    - különféle rendszerindító programok
    - file-ok (*és könyvtárbejegyzések*) adatai
  - ***partíció***
    - tárolás legnagyobb fizikai egysége
    - file-rendszerekben tárolására képes

- ***tárolás file-rendszerekben***
  - *felépítés*
    - FS metaadatok
    - file metaadatok
    - tárolt adatok
  - *file-rendszer*
    - háttértáron (méret, állapot, információk...)
    - memóriában („dirty” jelzőbit...)
  - *metaadatok elvesztése*
    - → másolatok készítése
- ***metaadatok elhelyezése***
  - *diszken*
    - hitelesítése információk
    - típus
    - méret
    - időbélyeg
  - *memóriában*
    - state (zárolt, módosított)
    - háttértár eszköz azonosítója
    - hivatkozási számláló
    - csatlakoztatási pont
- ***adatblokk allokáció***
  - *folytonos tárolás*
    - törléssel különböző méretű üres helyek
  - *láncolt lista*
    - blokkokra bontott tartalom + hivatkozás további adatrészekre
      - **egyszeres láncolt lista**
        - » lassú
        - » soros elérésre hatékony
        - » érzékeny viszont hibákra

Commented [KC113]: Más variáció pl. a FAT: táblában épít listát blokkszámokból.

- *indexelt*
  - blokkokra bontott tartalom + elhelyezkedési térkép (index)
  - **ügyes elhelyezés: szekvenciális**
    - gyors olvasás
  - direkt elérés
- index mérete gond → tárolás láncolt listában

- ***üres helyek menedzselése***
  - *bittérképes, bitvektoros*
    - könnyű szabad blokkot találni
    - memóriában is elférhet
    - CPU utasítás elég
    - nagy file-rendszereknél nem hatékony
  - *láncolt lista*
    - minden üres blokk egy következőre mutat
    - az első szabad blokk címét kell megjegyezni
    - egyszerű
    - nem a leghatékonyabb
  - *hierarchikus módszer*
    - üres helyek csoportjait kezeli
    - csoportok létrehozhatók
    - csoporton belül egyszerű belső struktúra
- ***adatblokk gyorstárazása***
  - *disk buffering*
    - memóriát használja gyorsítótárként: blokkgyorsítótár **(buffer cache)**
    - hatékonyság ↑
    - írásnál megbízhatóság ↓

- *gyorsítótár szervezés*
  - virtuális tárkezelés mechanizmusai használható
  - egységes blokkgyorsítótár (unified buffer cache)
    - pl. Linux
  - előreolvasás (readahead)
  - nem használt blokkok törlése
    - pl. LRU
  - írásműveletek kezelése
    - **írásáteresztő gyorsítótár (write through cache):** háttértárra azonnal, lassú, megbízható
    - **pufferelt gyorsítótár**: írás időnként, nagy teljesítmény, kevésbé megbízható

- ***metaadatok konzisztenciája***
  - *konzisztencia-problémák*
    - módosított adatok gyorstárazása
    - metaadatok változásai, gyorstárazás
  - *metaadatok sérülése*
    - pl. tárhelyelszivárgás (*storage leak*)
    - teljes file-rendszer összeomlásához is vezethet
  - *megoldási ötletek*
    - adatok esetén: írásáteresztő gyorsítótár
      - → lassítja a működést
    - metaadatokra nem jó

- ***naplózó file-rendszerek (journaling)***
  - *napló (journal)*
    - szekvenciálisan írható körpuffer a háttértáron
    - elvégzendő műveleteket tartalmazza
    - pl. NTFS LFS
  - *megvalósítás: tranzakció alapú működés*
    - tranzakció zárul, ha minden művelet naplóban van
    - ezeket feldolgozza, végrehajtja
    - ha összeomlik, induláskor feldolgozza

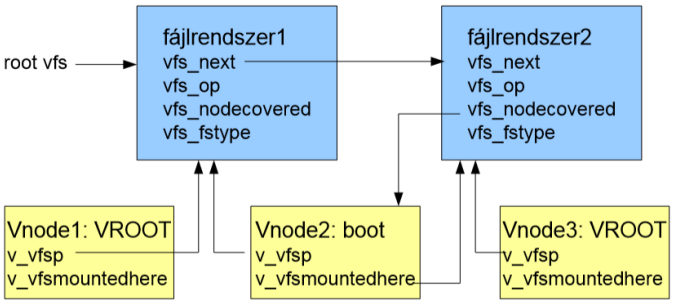
- *log-structured file-rendszer*
  - a napló a file-rendszer
- *COW file-rendszer*
  - írásműveletek másolt adatokon végrehajtva, majd átírja metaadatokra

BACK

16. előadás:
File- és tárolórendszerek II.

1. Virtuális file-rendszerek

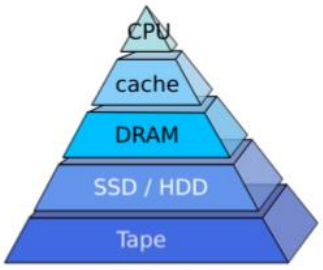
- *cél*
  - többféle fájlrendszer egységes támogatása, kezelése
  - uniform file-rendszer megvalósítás
  - modulárisan bővíthető
- *vnode és vfs*



  - *vnode/vfs*
    - közös adatok
    - v_data/ vfs_data: állományrendszertől függő adatok (vnode-inode)
    - v_op/ vfs_op: az állományrendszer metódusainak táblája
  - *virtuális függvények*
  - *segédrutinok, makrók*
    - közös adatok
    - vfs_data: állományrendszertől függő adatok
    - v_op: az állományrendszer metódusainak táblája

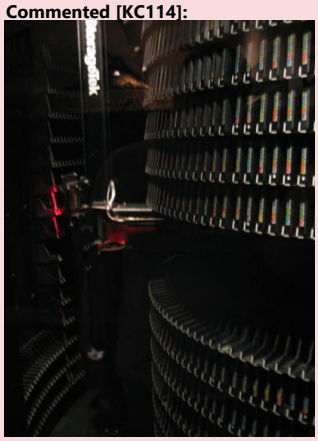
2. Adattárolás

- ***tárolási megoldások***
  - *fizikai tárolóeszközök*
    - mágneses elven működő (HDD)
    - optikai elven működő (CD)
    - nemfelejtő memóriaalapú eszközök (SSD)
  - *virtualizált tárolórendszerek*
    - összeolvasztanak tárolóeszközöket
      - megbízhatóság- és kapacitásnövelés
      - pl. RAID
    - hálózati elérést tesznek lehetővé
      - file vagy blokkszintű adatátvitel
    - elosztott tárolási rendszert valósítanak meg
      - megbízható és jól skálázható tárolórendszerekhez

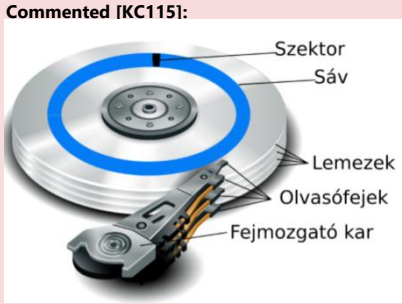


- ***változó teljesítményviszonyok***
  - sok RAM → nagy puffer gyorsítótár
  - gyors CPU → I/O teljesítménynövelés
    - futásidejű adattömörítés (pl zfs)
    - deduplikáció
- ***szalagos tárolók***
  - biztonsági mentésre
    - nagy kapacitás
    - hosszú élettartam
    - lassú
    - drága automatizálás
- ***allokáció merevlemezes meghajtón***
  - *cilinder (blokk) csoport*
    - azonos fejpozícióhoz tartozó sávok
    - fej mozgatása nélkül olvasható szektorok
    - egyszerre sérülnek a fej miatt

Commented [KC114]:



Commented [KC115]:



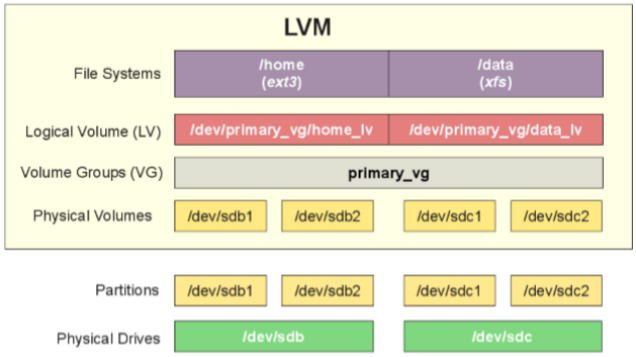
- *allokációs elvek*
  - szuperblokk másolása minden cilindercsoportba
  - inode lista és szabad blokkok csoportonként kezelve
  - egy könyvtár – egy csoport
  - kis file-ok egy csoportba
- ***diszk feladatok ütemezése***
  - *Noop (FIFO): összevonhat szomszédos kéréseket*
    - minimális terhelés
    - ha tároló is ütemez
    - ha nincs értelme ütemezni vagy nincs nagy I/O terhelés
  - *deadline: késleltetésre optimalizál*
    - blokkcím szerint rendezett írási vagy olvasási kötegek
    - nagy I/O terhelésnél globálisan jó
  - *CFQ (Completely Fair Queuing): egyenletes kiszolgálás, default*
    - folyamatonkénti sorok, azokhoz rendelt I/O időszeletek
    - sorokhoz prediktív előrejelzés
    - kiegyensúlyozott ütemező

3. <u>Tárolórendszer virtualizáció</u>

- ***virtuális tárolórendszerek***
  - *fizikai tárolórendszerek korlátai*
    - kapacitás, teljesítmény, megbízhatóság, menedzsment, hibaelhárítás
  - *virtualizáció*
    - fizikai eszközökre épít szolgáltatási réteget
      - összeolvasztás
      - szolgáltatásbővítés
      - jobb menedzsment

- *virtuális tárolórendszer*
  - fizikai eszközök határain átnyúló tárolórendszer
  - építőelemei
    - fizikai tárolók: diszk, partíció
    - virtuális tárolók, pl. RAID

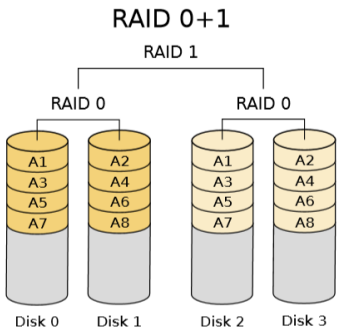
- ***logikai kötetkezelés (logical volume management)***
  - *fizikai kötet*
    - részei: physical extent (PE)
  - *logikai kötet*
    - részei: logical extent (LE)
    - kötetcsoport: LV-k halmaza, a virtuális tároló



4. RAID

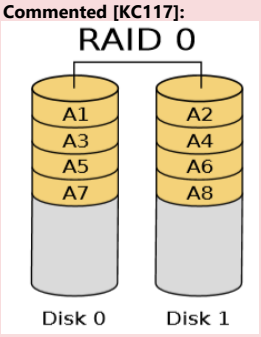
- ***tárolórendszerek megbízhatósága***
  - fizikai eszközök számának növekedésével nő a hiba esélye
  - több eszköz → **nagyobb** megbízhatóság
- ***adatredundancia beépítésével elérhető***
  - *tükrözés (mirroring)*
    - költséges
  - *paritás*
    - hibajelzés mellett javítás is

- ***Redundant Array of Inexpensive Disks***
  - virtuális tárolórendszer
  - olcsó, kis lemezek egybeolvasztása
  - cél: redundancia (megbízhatóság) és a teljesítmény
- ***RAID szintek***
  - *RAID 0 (**stripe** – csíkozás)*
    - N diszken egyenletesen terít
    - cél: a teljesítmény ↑
    - kapacitása összeadódik
    - hiba esetén az adat elvész
  - *RAID 1 (**mirror** - tükrözés)*
    - adatok többszörözve tárolj
    - cél: megbízhatóság
    - méret: egy diszk kapacitása
    - lassú írás, olvasás gyorsabb
  - *RAID 0+1 (**mirror of stripes**)*

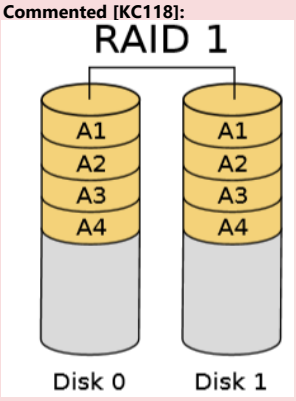


Commented [KC116]: Szint = egybeolvasztás módja

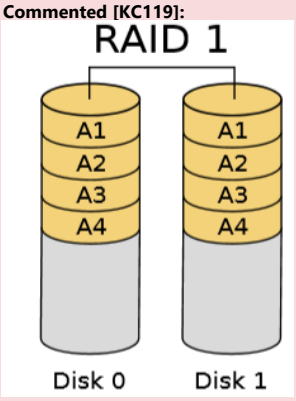
Commented [KC117]:



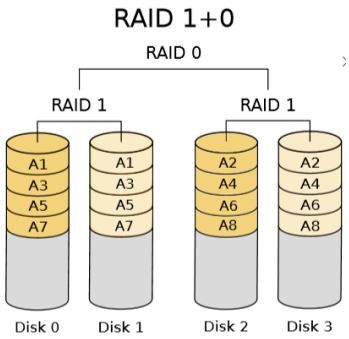
Commented [KC118]:



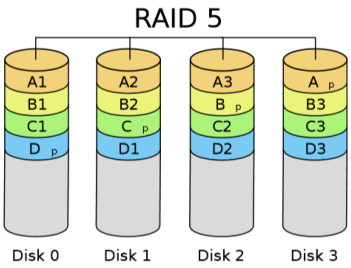
Commented [KC119]:



- *RAID 1+0 (**stripe of mirrors**)*

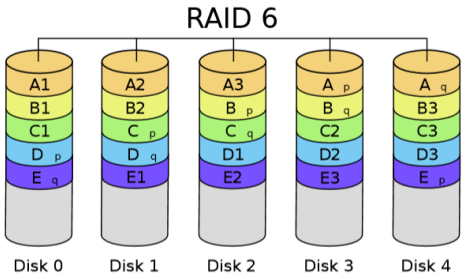


- *RAID 5 (**blokkszintű csíkozás paritással**)*



- **felépítés**
  - N adatblokk + 1 paritásblokk (N+1 diszk)
  - paritásblokkokat egyenletesen („csíkozva”) helyezi el a fizikai diszkeken
- **jellemzés**
  - teljesítménye a RAID0-hoz közeli
  - kapacitás egy diszk mérettel ↓
  - meghibásodás ellen véd („néma hiba”)

- *RAID 6 (**blokkszintű csíkozás két paritással N+2**)*



  - **felépítés**
    - RAID5 + második paritásblokk
  - **jellemzés**
    - jó teljesítmény, mérsékelt kapacitáscsökkenés
    - helyreállításkor jelentkező néma hiba ellen is véd
    - két diszk hibája ellen véd
- *problémák*
  - hibajavítás órákig is eltart
  - sok egyforma diszk kell → nehéz pótolni
  - kötött redundancia, nem rugalmas
  - tárolókapacitás nem növelhető nagyra
    - 6-8 diszk/HW RAID kártya
  - csak diszkhiba ellen véd

Commented [KC120]: Mi van akkor, ha alaplapi, CPU , memória vagy akár táp hiba keletkezik, velük ki foglalkozik?

5. Hálózati és elosztott tárolórendszerek

- ***kliens-szerver modell***
  - szerver: helyi tároló → hálózat → kliens
  - file-tároló: NAS
    - NFS
    - SMB/CIFS
  - blokktároló: SAN
    - iSCSI (internet SCSI): SCSI parancsok IP-alon
    - FCP és FCoE: SCSI átvitel FC / ethernet-alapon

Commented [KC121]: Network Attached Storage

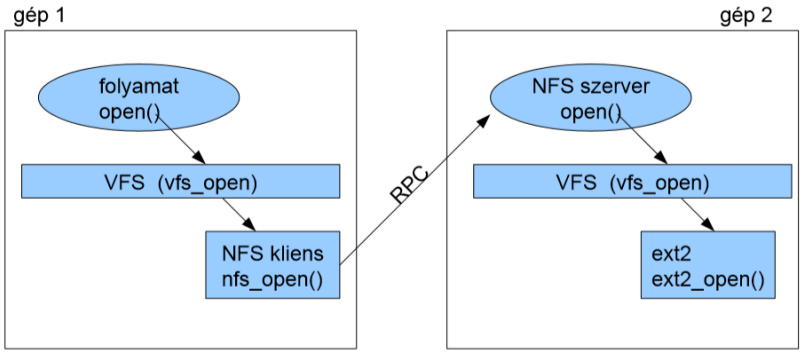
Commented [KC122]: Network File System

Commented [KC123]: Common Internet File System

Commented [KC124]: Storage Area Network

Commented [KC125]: Fibre Channel Protocol

- ***elosztott file-rendszer***
  - elosztott rendszerként működő tárolórendszer
    - Ceph (Inktank, Red Hat, SUSE), Google GFS, RedHat GlusterFS, Windows DFS, PVFS (parallel virtual FS)
- ***problémák***
  - delay
  - error
  - consistence



- ***adatok elérése***
  - *elhelyezkedés-átlátszóság (location transparency)*
    - adatok címzése (fájlok megnevezése) nem utal az elhelyezkedésükre
  - *elhelyezkedés-függetlenség (location independence)*
    - címzés, elnevezés nem változik az adatok áthelyezésével
- ***hálózati kiszolgáló***
  - *állapottároló (stateful)*
    - file-műveletek előélettel rendelkeznek
    - gyorsabb
  - *állapotmentes (stateless)*
    - örökifjú
    - megbízhatóbb

- ***elosztott, skálázható tárolórendszerek: Ceph***
  - *elérés*
    - SAN, NAS, OSD
  - *előnyök*
    - alapja RADOS
    - skálázható, hibatűrő (nincs leállás)
    - nincs sebezhető pontja (single point of failure)
    - minden komponense futásidőben cserélhető, bővíthető (új diszk, gép)
    - futásidőben változtatható a replikáció mértéke (hány másolat legyen)
    - gyorsabban felépül a hardver hibákból, mint a RAID tömbök
    - nem igényel speciális hardvereket, sem tartalék hardvereket
    - együttműködik a virtualizációs rendszerekkel (OpenStack, Amazon S3)
    - nyílt forráskódú

Commented [KC126]: Objektumtár. OSD (Object Storage Device), az adatok objektum-alapú tárolására, az alap.

Commented [KC127]: Reliable, Autonomic Distributed Object Store

BACK

17. előadás:
Virtualizáció

1. Virtualizáció

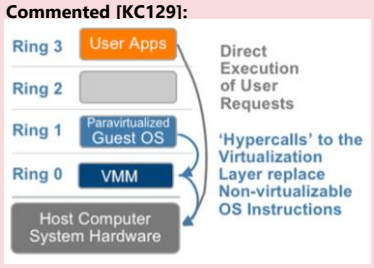
- Definíció
  - erőforrás virtuális (szoftveres) változatának létrehozása, amely az eredetire támaszkodva, ahhoz hasonlóan, de attól elválasztott módon működik
  - ***virtualizált erőforrás***:
    - számítógépes hardver vagy szoftver
  - ***host***:
    - virtualizált erőforrást biztosítj
  - ***guest***:
    - erőforrás felhasználója
- ***virtualizálhatók***
  - *hardver*
    - teljes számítógép
    - számítógépes hálózat
    - grafikus kártya
  - *szoftver*
    - szolgáltatáshalmaz (API)
    - rendszerkönyvtár (pl. GUI)
    - kernel, kernelrész
  - *adat*
    - formátum- és elhelyezkedésfüggetlen
    - hozzáférés és módosítás
  - *már virtualizált rendszer*

- ***virtualizáció haszna***
  - *konkurens erőforrás-használat*
    - egyszerre többen használják az erőforrást → kevesebb üresjárat
    - versenyhelyzetek kezelése
  - *összeolvasztás*
    - kapacitásbővítés
    - szolgáltatások fúziója
  - *szolgáltatásbővítés - szűkítés*
    - többféle összegyúrva
    - újfajta aggregált szolgáltatások megvalósítása
  - *felügyelet, mendzsment*
    - szabályozott, automatizált
    - szereplők, jogosultságok
  - *csökkennő...*
    - gyártófüggőség
    - erőforrásszám
    - hiba
    - költség
      - **TCO** (*Total Cost of Ownership*)
      - beruházás
      - fenntartás
  - *növekvő...*
    - rendelkezésre állás
      - kezelhetőbb hibák
      - megbízható rendszerek
    - flexibilitás
      - rugalmasabb specifikáció

- ***fajtái***
  - *rendszer (system/ platform/ full)*
    - teljes rendszer, környezet virtualizáció felépítése
    - feladatok: OS és taskok
    - részei
      - host: amelyen a virtuális gép fut
      - guest: gazdagépen futó virtuális gép
      - VMM: virtuális gépeket felügyelő program
    - pl. VirtualBox
  - *folyamat (process/ software)*
    - API/ ABI virtualizáció
    - task működéséhez felület
    - erőforrás működéséhez szükséges felület
    - pl. Java VM
  - *infrastruktúra (infrastructure)*
    - infrastruktúrális elemek virtualizációja
    - erőforrás egy hardware/software elem
    - pl. hálózat, adattároló
- ***hardver virtualizáció fajtái***
  - *bare meta*
    - hardvert VMM kezeli → hypervisor
    - hostban nincsenek alkalmazások
    - *transzparens megfeleltetés*: **natív virtualizáció**
      - hardver támogatás
      - futásidejű bináris átírással
    - *más hardver képében*: **paravirtualizáció**
      - fizikai hardverhez hasonló, nem virtuális hardver
  - *hosted*
    - VMM alkalmazás a gépben
    - gazdagépen más alkalmazások is futhatnak (több VMM)

Commented [KC128]: Virtual Machine Monitor.

Commented [KC129]:



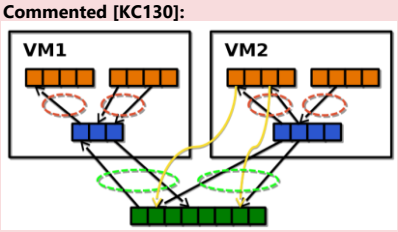
- *hybrid*
  - hypervisor + kernel
  - VMM funkciói kernelre építve

2. **Megvalósítás**

- ***elvárások***
  - *transzparencia*
    - vendég gép változtatás nélkül működjön
    - programokat ne kelljen kézzel átírni
    - automatikus és láthatatlan az utasítás-átírás
    - sok feladat a virtuális rendszerre
  - *védelem*
    - vendég ↔ gazda, vendég ↔ vendég
    - felügyelet, jogosultságok megvalósítása
  - *hatékonyság*
    - VM rezsiköltsége kicsi
    - átírás kevésbé csökkentse a teljesítményt
    - hardvertámogatás kihasználás
- ***CPU***
  - *tiszta emuláció*
    - utasítások VM-en, azokat leképzi
    - nem hatékony
  - *trap and emulate*
    - utasítások futásidőben
    - privilegizált: elkapja, átírja
    - nem védett: közvetlenül végrehajtja
    - hardvertámogatás
  - *bináris átírás*
    - utasításokat végrehajtás előtt átírja
    - CPU biztonságos utasításokat hajt végre
    - átírás ↓ hatékonyságot

- *forráskód-átírás (paravirtualizáció)*
  - vendég OS forráskód fejlesztési időben átalakul
  - privilegizált utasítások VMM hívásra
  - fejlesztői támogatás
- ***memória***
  - *teljesítményérzékelt terület*
  - *kétszeres címfordítás*
    - hardvertámogatás
    - beágyazott laptáblák, TLB címkézés
- ***I/O***
  - *szoftveres emuláció*
    - teljes kommunikáció, hardver emulálás
    - korlátozott képesség
    - transzparens
    - nem hatékony
  - *paravirtualizáció*
    - guest által kínált eszközt használja
    - hívások, mozgások egyszerűsödnek hardver felé
    - speciális eszközmeghajtó guesten
    - hatékonyabb, kevésbé transzparens
  - *hardveres virtualizáció*
    - I/O eszközök megosztása
    -

Commented [KC130]:



Commented [KC131]: AMD Rapid Virtualization Indexing, Intel Extended Page Tables

Commented [KC132]: Intel VT-d, AMD IOMMU, PCI IOV

3. **Termékek, szolgáltatások**
   - ***felhőalapú szolgáltatások***
     - *IaaS*
       - teljes hardver
       - OS-t telepíthetünk, sablonokkal
       - pl. Amazon EC2, RackSpace, Microsoft Azure, Linode, Digital Ocean

Commented [KC133]: Infrastructure-as-a-Service

- *PaaS*
  - teljes hardver
  - saját alkalmazások futtatása
  - pl. Amazon AWS, Microsoft Azure, Google AppEngine, Heroku
- *SaaS*
  - szoftverszolgáltatás
  - előre telepített alkalmazások
  - pl. Microsoft Office365, Google Docs és Gmail

Commented [KC134]: Platform-as-a-Service

Commented [KC135]: Software-as-a-Service

- ***virtualizáció kockázatai***
  - *támadások*
    - virtualizációs infrastruktúra lecserélésre (**hyperjacking**)
    - virtualizációs mechanizmusok ellen
      - egy-egy mechanizmus (pl. hálózat, migráció) megfigyelése, megváltoztatása
    - felügyelt rendszerek közötti adat- és kódszivárgás (**VM jumping**)
      - vendég kitörése (**guest breakout**)
  - *auditálás*
    - nehézkes, bonyolultabb rendszer → nagyobb dinamizmus
  - *menedzsment*
    - sokféle virtualizált erőforrás, összetett virtualizációs sémák
  - *szakemberhiány*
    - új és változó technológiák