

# Az operációs rendszerek működése: fájl- és tárolórendszerek

*Mészáros Tamás*

<http://www.mit.bme.hu/~meszaros/>

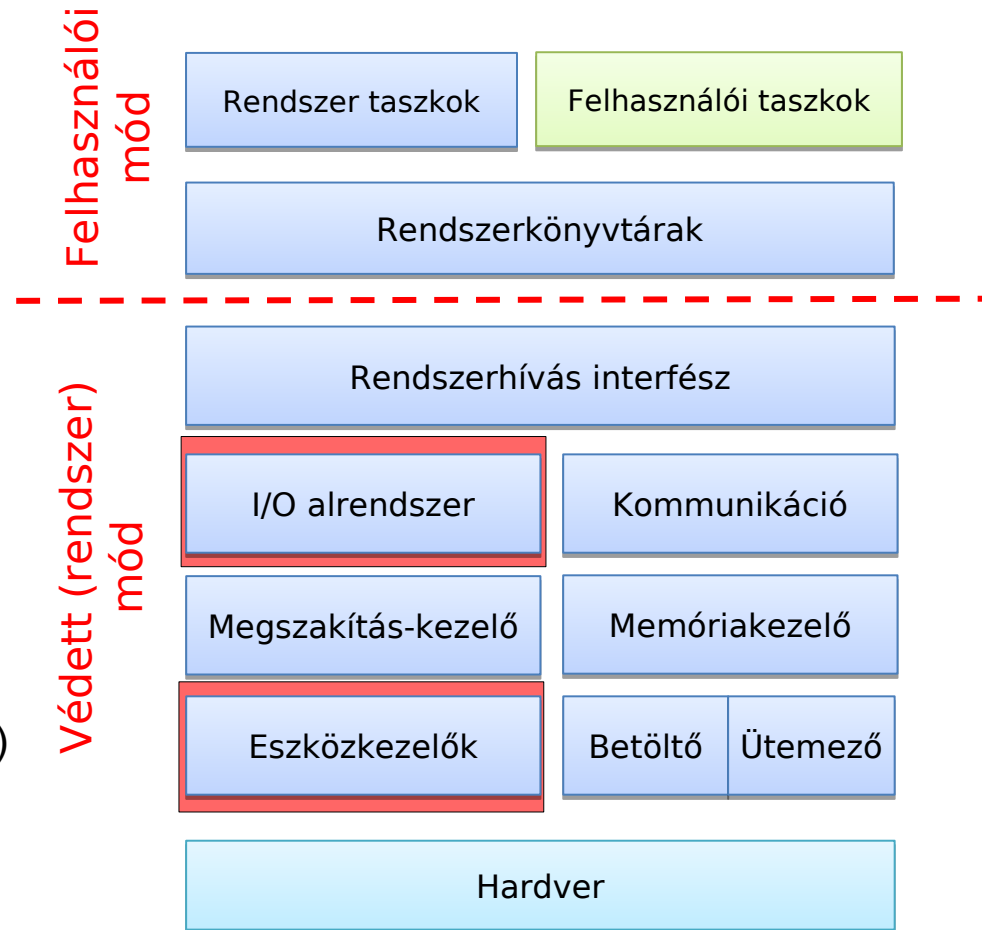
Budapesti Műszaki és Gazdaságtudományi Egyetem  
Méréstechnika és Információs Rendszerek Tanszék

Az előadásfóliák legfrissebb változata a tantárgy honlapján érhető el.  
Az előadásanyagok BME-n kívüli felhasználása és más rendszerekben történő közzététele előzetes engedélyhez kötött.

# Az eddigiekben történt...

- A taszkok...
  - jellemzően I/O-intenzívek
  - sok fájlműveletet végeznek
  - programkódjuk a fájlrendszerben
- Memóriakezelés...
  - a háttértárral bővül (cserehely)
- Kommunikáció
  - kommunikáció fájlon keresztül (mmap)
- Laborok
  - Linux: hálózati fájlrendszer (Samba)
  - Windows: fájlleírók, jogosultságok, megosztás, hálózati meghajtók

## Az OS felépítése



# Áttekintés

## Felhasználói szemmel...

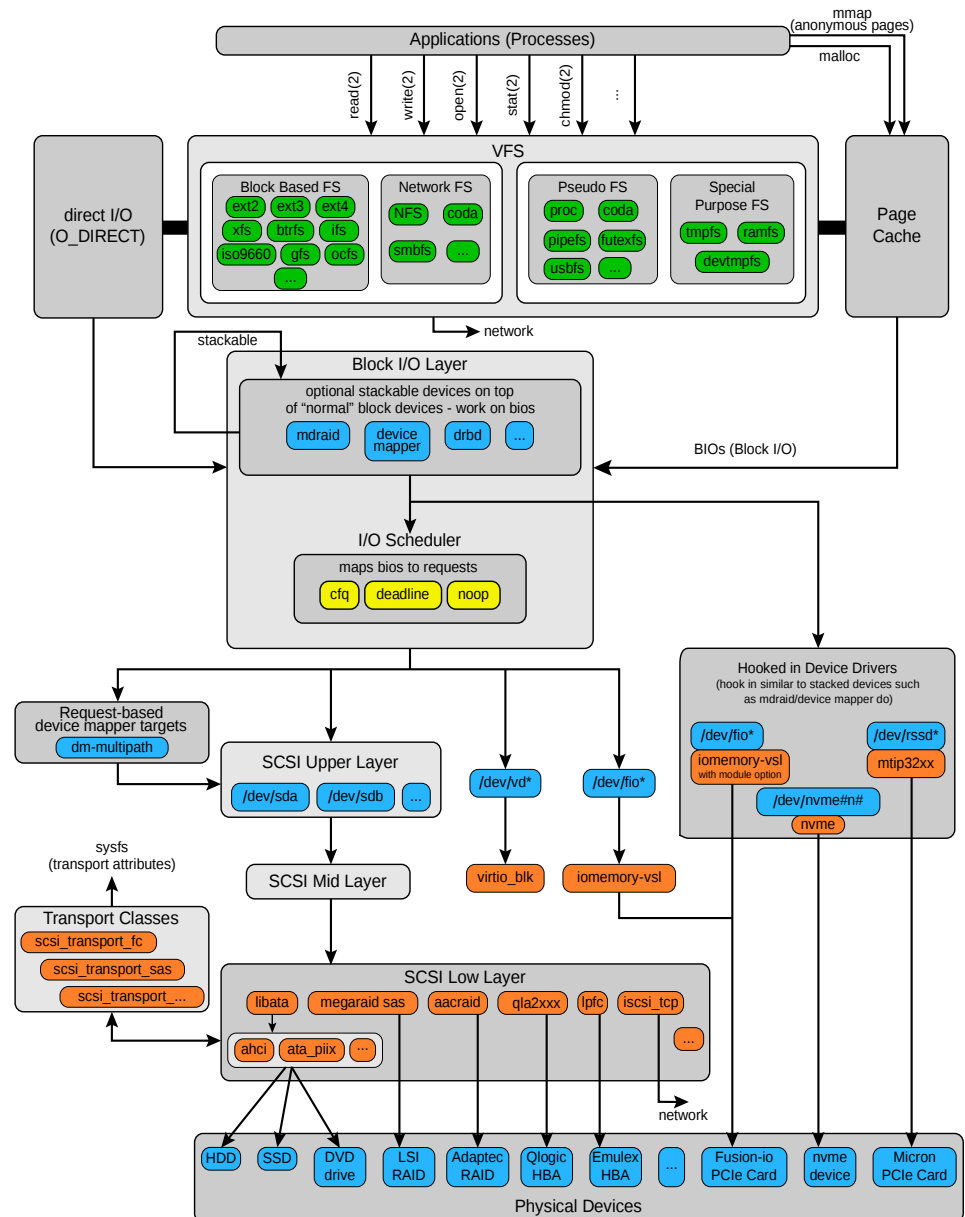
- végfelhasználó
- adminisztrátor
- programozó

## Belső működés

- fájlrendszer interfészek
- kernel adatstruktúrák
- a háttértár szervezése
- virtuális fájlrendszerek

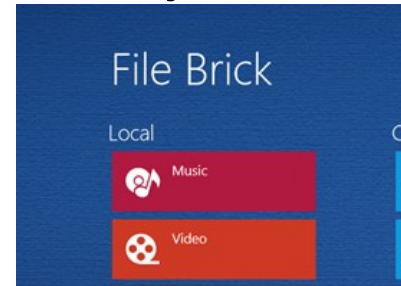
## Adattárolás

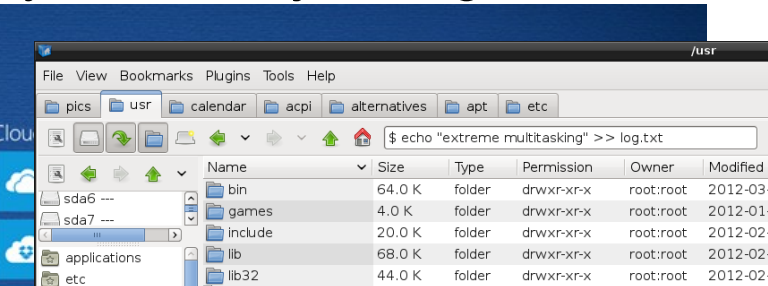
- fizikai tárolók (HDD, SSD)
- I/O ütemezés
- tárolórendszer-virtualizáció:
  - helyi (RAID, LVM)
  - hálózati (SAN, NAS)
- elosztott fájl- és tárolórendszerek



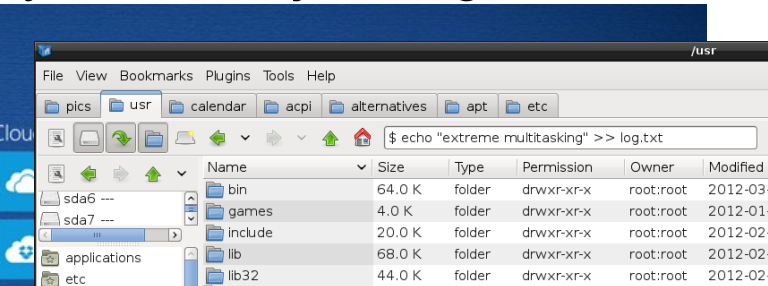
# A fájlrendszer felhasználói szemmel

- Parancssori és grafikus fájlkezelők
- A tárolási rendszer logikai felépítése (helyek)
- Fájlok és könyvtárak tulajdonságai





Left	File	Command	Options	Right
'n	Name	Size	Modify time	'n Name
/.xournal	4096	dec 9 2015		UP --DIR
/.yjp	4096	ápr 4 09:34		
/Android	4096	okt 6 2015		/00 Arch-sorok)
/Android~objects	4096	márc 11 2016		/00 Irod-aghoz)
/Backup	4096	febr 9 15:30		/00 Jegyzet
/DHmine	4096	máj 4 07:43		/00 Sablonok
/Documents	12288	ápr 27 08:29		/00 Tematika
/Downloads	4096	ápr 7 12:55		/01.Bevezetés
/Music	4096	jan 6 2016		/02.Felé-üködés
/Pictures	61440	ápr 27 10:02		/03.Felh-elület
/R	4096	máj 23 2016		/04.Fela-ezelés
/TODO	4096	márc 4 2016		/05.Ütemezés
/Videos	4096	márc 22 11:32		/06.Memó-ezelés
/abevjava	4096	máj 19 2016		/07.Komm-ikáció
/eclipse	4096	máj 11 2016		/08.Szin-izáció
				/09.Fájl-szerek
/DHmine				UP --DIR



# A fájlrendszer felhasználói szemmel (folyt.)

- Adminisztrátor
  - létrehozás, ellenőrzés, megszüntetés
  - helyi és távoli fájlrendszerek használatba vétele (csatolás)
  - teljesítményhangolás
  - helyfoglalás ellenőrzése és felszabadítása
  - biztonsági másolatok készítése
- Programozó (alkalmazásfejlesztő)
  - programozói interfészek
    - rendszerhívások
    - rendszerkönyvtárak
  - fájlleírók és fájlműveletek
    - megnyitás, létrehozás, írás, olvasás, pozicionálás, bezárás, törlés
  - fájlok zárolása kizárólagos használatra

# Alapfogalmak

- **Fájl** (file), állomány
  - az adattárolás logikai egysége
  - név (+ esetenként kiterjesztés)
- **Könyvtár** (directory)
  - a szervezés logikai egysége
  - fájlok és könyvtárak halmaza
- **Kötet** (volume), meghajtó
  - fájlok és könyvtárak tárolásának logikai egysége
  - fizikai tárolási egységhez (pl. partíció) rendelhető

Logikai

- 
- **Fájlrendszer** (file system)
    - fájlok és könyvtárak fizikai tárolása és szervezése
  - **Partíció** (partition)
    - a háttértár szervezési egysége
    - fájlrendszer tárolására képes

Fizikai

# Fájlrendszerek logikai szervezése

- Irányított fával reprezentálható
  - csomópontok: könyvtár, fájl, (tárolt adat)
  - élek: tartalmazás reláció
  - gyökér csomópont: a kötethez (meghajtóhoz) rendelt elem
- **Elérési út (path)**
  - egy csomópont elérési helye
  - **abszolút**: a fa gyökerétől kezdve
  - **relatív**: egy másik csomóponttól (pl. munkakönyvtár)
- A fa bővítése irányított gráffá
  - (rögzített) **link** (hard link)
    - több fájl ugyanarra az adatra hivatkozik
  - **szimbolikus link** (symbolic link, symlink, soft link)
    - egy másik fájlrendszeri elemre (fájl, könyvtár) mutat

*A link és az adat melyik esetben mikor és hogyan törölhető?*

*Mit okoz egy irányított kör a gráfban?*

# Példa: Windows 10

- Fizikai tárolók logikai meghajtókhoz rendelve
  - könyvtárakhoz rendelt kötetek is léteznek, de ritkák
- A boot meghajtó (jellemzően C:) a kiinduló pont ( `dir c:\` )
  - `\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 (D: E: stb.)
  - CD/DVD/USB fizikai tárolóeszközök
  - további partíciók a diszkeken
  - hálózati fájlrendszerek



# Példa: Unix / Linux

- Könyvtárakhoz rendelt fizikai tárolók

- egyetlen összefüggő gráfot alkot

- Gyökér: / avagy ROOT ( `ls /` )

<code>/bin</code>	a rendszer működéséhez szükséges alapvető bináris állományok
<code>/sbin</code>	hasonló, de alapvetően a rendszergazda által futtatható programok
<code>/dev</code>	hardver eszközök
<code>/etc</code>	a rendszer konfigurációs beállításait tároló fájlok
<code>/home</code>	a felhasználók saját könyvtárai (jellemzően külön fizikai tárolóval)
<code>/lib</code>	alapvető (megosztott, shared) rendszerkönyvtárak
<code>/mnt</code>	alkalmilag felcsatolt partíciók helye (mount)
<code>/tmp</code>	átmeneti fájlok (programok és felhasználók számára)
<code>/usr</code>	felhasználói programok, programkönyvtárak, dokumentáció, stb.
<code>/var</code>	a rendszerműködés „dinamikus” fájljai, naplófájlok, adatbázisok

részletesebben lásd `man hier`

- Szabványok, változások

- jelentős eltérések lehetnek a részletekben
- FHS (Filesystem Hierarchy Standard): inkább csak ajánlás
- UsrMove: a `/bin`, `/sbin`, ... átkerül a `/usr` alatti helyére (Solaris11, Fedora)

# Példa: Android

- Unix-szerű, de eltérő könyvtárak
  - nem triviális megnézni a teljes gráfot (demo)
- Gyökér: / avagy ROOT ( `ls /` )
  - `/cache` gyorsítótár az alkalmazások számára
  - `/data` felhasználói **programok és adatok**
  - `/data/app` a felhasználó által telepített alkalmazások
  - `/data/data` az alkalmazások adatfájljai
  - `/data/anr` app-not-responding: alkalmazáshibák adatai
  - `/data/tombstones` hibával (pl. SIGSEGV) leállított alkalmazások memóriaképei
  - `/data/dalvik-cache` az alkalmazások optimalizált bináris állományai
  - `/data/misc` felhasználói konfigurációs fájlok (pl. wifi, bluetooth, vpn beállítások)
  - `/data/local` átmeneti fájlok
  - `/mnt` v. `/storage` további csatolt fájlrendszerek (pl. SD kártya) elérhetőségei
  - `/mnt/asec` az SD kártyára írt alkalmazások futásidejű (titkosítatlan) változatai  
**titkosítva** az `.android_secure` könyvtárban vannak
  - `/system` előtelepített alkalmazások, rendszerkönyvtárak, konfigurációk

# Fájlok tulajdonságai (Unix példákkal)

- Listázzuk ki fájlrendszeri bejegyzések adatait! `ls -la <fájlnév>`

```

-rw-r--r--  1 root root      2290 júl   5  2014 /etc/passwd
-rwxr-xr-x  1 root root   616920 nov  17  2015 /bin/bash
srwxr-xr-x  1 clamilt clamilt 0 ápr 22 10:16 clamav.sock
crw-rw----  1 root tty        4, 0 ápr 20  2007 /dev/tty0
---s--x--x. 1 root root   123832 Aug 13  2015 /usr/bin/sudo

```
- Mit látunk a listában?
  - a bejegyzés típusa: (- d p l b c s)
  - POSIX jogosultságok (lásd következő fólia)
  - linkek (hard) száma
  - tulajdonos és csoport
  - méret
  - időbélyeg (ctime: metaadatok változása, mtime: adatmódosítás, atime: olvasás)
  - a bejegyzés neve
- Amit fent nem látunk, de az OS tárolja (lásd később)
  - egyedi azonosító (belső használatra)
  - elhelyezkedés (hol vannak a fájl adatai)

# Unix hozzáférési jogosultságok

- **POSIX** jogosultságok (*alap*)

- 3 x 3 bit: { tulajdonos, csoport, mások } x { olvasás, írás, futtatás }
- könyvtárak használatához olvasás és „futtatás” is kell
- beállítás: **chmod** <jogosultság> <fájl v. könyvtár>

pl.: `chmod 750 /home/me`

`chmod u+rwx,g+rx,o-rwx /home/me`

- Speciális jogosultságok: SETUID, SETGID, StickyBit

- SETUID/GID: futási tulajdonos/csoport beállítása

`chmod u+s setuid_file`

`chmod g+s setgid_file`

**KOCKÁZATOS !**

- StickyBit: csak a tulajdonos törölhet

`drwxrwxrwt 44 root root`

`12288 máj`

`9 15:25 /var/tmp`

- POSIX ACL (access control list) (*kiterjesztett*)

- rugalmasabb, többféle jogosultság egyidőben

pl.: `setfacl -m u:student:r file`

- lásd `ls` parancs kimenetén + jel

# Adminisztrátori alapfeladatok

- Fájlrendszer létrehozása (formázás)
  - típus (l. köv. fólia)
  - jellemzők (alapértelmezett jó + esetleg titkosítás)
  - név (emberi), azonosító (gépi)
  - tárolási hely
- Csatlakoztatás (mount)
  - fizikai → logikai tárolási hely
  - **csatlakoztatási pont** (mount point)
  - **elfedés**
- Ellenőrzés, hangolás
  - állapotellenőrzés és hibajavítás (offline)
  - a méret megváltoztatása (online) a tárolórendszerrel összhangban
  - teljesítmény: tárolóhoz igazítás (alignment), tömörítés stb.
- Biztonsági mentés

# Széles körben elterjedt fájlrendszerek [áttekintése](#)

- FAT32
  - kompatibilis
  - eredetileg 8+3 karakteres fájlnev 255-re bővítve, 4GiB maximális fájl méret (!)
- NTFS
  - a Windows alapértelmezett fájlrendszere ([továbbiak áttekintése](#))
- UFS avagy Berkeley FFS (lásd KK. tankönyv)
  - tradicionális BSD Unix fájlrendszer
- ext2,3,4 (UFS-alapokra épült)
  - Linux
- XFS
  - eredetileg SGI, újabban pl. RedHat Linux 7
- HFS+, újabban APFS (iOS 10.3)
  - Apple
- Integrált fájl + tárolórendszerek (lásd még később)
  - **ZFS**: Solaris, később nyílt forrású, BSD-körökben is népszerű
  - Linux **btrfs**: újabb, aktív fejlesztés alatt álló Linux fájlrendszer
- [Ezernyi más](#) fájlrendszer, pl.:
  - CD/DVD (ISO 9660 és kiterjesztései)

# Demók (otthonra is!)

- Alapvető fájl- és könyvtárműveletek `cp mv cd pwd mkdir`

Hogyan lehet átnevezni egy fájlt?

- Fájlok attribútumai: `ls -la` `ls -laZ` `setfacl`

- Fájlrendszerek kezelése: `mount umount df mkfs fsck`

```
mount (/proc?) df umount /boot mount /boot (honnan?)
mount -bind ...
```

Hozzunk létre egy új fájlrendszer egy fájlban (`sudo su` – kiadása után)!

```
dd if=/dev/zero of=filesystem.img bs=1k count=1000
losetup /dev/loop0 filesystem.img
mke2fs /dev/loop0
mount /dev/loop0 /mnt
```

Az egyik tipikus, bosszantó hibajelzés

```
umount: /mnt: device is busy
```

Miért nem sikerül? Valaki foglal (nyitva tart) fájlt, könyvtárat.

Mit tehetünk? Megnézzük, ki mit tart fogva: `lsof /mnt` (esetleg `remount,ro?`)

- Mi történik a fájlrendszerben? `iostat sar dstat vmstat ...`

```
sudo sysctl vm.block_dump=1
tail -f /var/log/kern.log
```

- Tegyük tönkre a fájlrendszert, és próbáljuk helyreállítani!

# Fájlrendszerek hangolása (demók)

- Szabad hely növelése

- diszkhasználat elemző: `du xdu baobab kdiskstat filelight`
- töltsük tele a korábban létrehozott fájlrendszer-a-fájlban eszközt!  
`cp -r /bin /mnt` (ne root-ként futtassuk!)
- miért 0 a szabad hely, miközben nem foglalt minden blokk?  
súgó: `man tune2fs`
- futásidejű tömörítés bekapcsolása (tárolórendszerrel integrált fájlrendszerekben)  
pl.: `btrfs mount opció: compress = { zlib | lzo | snappy }`  
(A már létrehozott fájlrendszerre utólag is bekapcsolható.)

- Teljesítménynövelés – **Lassú diszk I/O: mi az oka, mi az elvárt IOPS?**

- a `noatime` opció hatása a teljesítményre  
A `/etc/fstab` fájlban módosítsuk az attribútumokat (lásd `man mount`)
- fájlrendszerszintű tömörítés  
Lényegesen kisebb adatmozgatás, CPU terhelés kismértékű növelése  
Lásd pl.: [www.phoronix.com/scan.php?page=article&item=btrfs\\_lzo\\_2638](http://www.phoronix.com/scan.php?page=article&item=btrfs_lzo_2638)
- `nr_requests`, `read_ahead_kb`, fájlrendszer naplózás és blokkméret
- fizikai és logikai blokkok összehangolása: **Partition Alignment**
- prioritás növelése (`ionice`) a kiemelt folyamatokra



# Biztonsági mentés és visszaállítása

- Adatvesztés oka
  - nem javítható meghibásodás
    - fizikai hiba
    - inkonzisztencia
  - felhasználó
  - kártevők
- Jellege
  - korlátozott
  - teljes (SSD „hirtelen halál”)
- Mentés (backup)
  - hogy: automatizált / kézi
  - mit: rész / teljes
  - hova: szalag, diszk, net
- Visszaállítás (restore)
  - „bare metal” / reinstall + restore

*Használat közben mi konzisztens?*

## Fájlok a programozó szemszögéből ...

# Programozói interfész

- Megnyitás (és létrehozás)

`open()`

- fájlleíró + nyitott fájl objektum (kernel)

- Írás, olvasás, pozicionálás

`read()` `write()` `fseek()`

- **soros elérés** (sequential access)

az adatokat tárolási sorrendben olvassuk illetve írjuk

- **közvetlen elérés** (direct access)

az adatok rögzített méretű részei tetszőleges sorrendben elérhetők

- Fájlok lezárása

`close()`

- Könyvtárak kezelése:

`opendir()` `readdir()` `rewinddir()` `closedir()`

# Mi történik egy fájl megnyitásakor?

- `open()`...
  - a cél lokalizálása (hol a fájl?)
  - metaadatok beolvasása
  - létrejön a **nyitott fájl objektum** (metaadatok a kernelben)
    - megnyitási mód
    - **fájlmutató (file pointer)**
    - fájl metaadatok
    - lehetséges műveletek
  - ezen objektum azonosítója a **fájlleírót (file descriptor)**
- A további műveletek során... (`read()`, `write()` stb.)
  - a fájlleíró azonosítja az objektumot
- Amikor lezárjuk (`close()`)
  - a kernel megszünteti a létrehozott adatstruktúrákat
- Miben más az `fread()` és a `fwrite()` pufferelt I/O? Hatékonyabb?
  - Otthoni gyakorlat: az `fread()` vagy a `read()` gyorsabb különféle terhelésekre?

# Fájlok zárolása

- Fájlok zárolása (= kölcsönös kizárás)
  - fájl = erőforrás *konzisztencia?*
  - szemaforokkal is lehetne,  
de a fájlműveletekkel egyszerűbb
  - **holtpont** itt is kialakulhat
- Ajánlott zárolás (advisory locking)
  - az OS csak eszközöket biztosít (rendszerkönyvtárakban), **nem kényszeríti ki**
  - a taszkok számára opcionális
  - példák: Java `FileLock()`, Unix `flock()`
- Kötelező zárolás (mandatory locking)
  - kernel mechanizmusok biztosítják (pl. fájlrendszer csatolásakor megadható)
  - a rendszerhívások **kikényszerítik** a betartását
  - példák: Windows általában, Unix / POSIX `fcntl()` `lockf()`
- Fájlok részleges (tartományi) zárolása
  - pl. Windows `LockFileEx()`, Unix `fcntl()`

# Fájlok megosztott elérése memórián keresztül (mmap)

- Egyszerűbb, mint a `read()`, `write()` és `fseek()`
- UNIX mmap (Windows: [CreateFileMapping](#))  
`mmap(addr, size, prot, flags, fd, offset)`
  - `addr`: ezt a címet rendeljük hozzá a fájl tartalmához (0: a kernel választ)
  - `size`: az elért adatmennyiség mérete
  - `prot`: a hozzáférés típusa (R, W, X), egyezik az `open()`-nél megadottal
  - `flags`: saját vagy megosztott fájl, stb.
  - `fd`: az `open()` rendszerhívás által visszaadott fájlleíró
  - `offset`: ettől a pozíciótól kezdődik a hozzárendelés
  - Visszatérési érték: az adatokhoz rendelt virtuális memóriacím (változóhoz köthető)
- A hozzárendelés megszüntetése: `munmap(addr, len)`
- Többszörös hozzáférés, konzisztencia és kölcsönös kizárás
  - a programozó dolga...
- **Fájlműveletek helyett is jó, ha sok** direkt elérésű olvasást végzünk.

# Várakozásmentes I/O: nem blokkoló és aszinkron

- Emlékeztető...
  - **ha van teendők, miért várakozunk?**
  - lassú I/O lassú → nagyon sok várakozás
- Nem blokkoló I/O műveletek
  - a rendszerhívás egyből visszatér
    - ha van adat, azzal
    - ha nincs adat, hibával
  - a programozó dolga időnként ellenőrizni
- Aszinkron I/O műveletek
  - beállítjuk az I/O műveletet és az adattároló puffert
  - elküldjük az aszinkron I/O kérést
    - a háttérben elindul az I/O művelet kiszolgálása
    - a rendszerhívás azonnal visszatér
  - a taszkunk tovább fut
  - amikor az I/O elkészült → esemény (jelzés)
    - az eseménykezelő kezeli az adatokat

lásd pl. [POSIX aio](#), Windows [I/O Completion Ports](#)

# Áttekintés

## • Felhasználói szemmel...

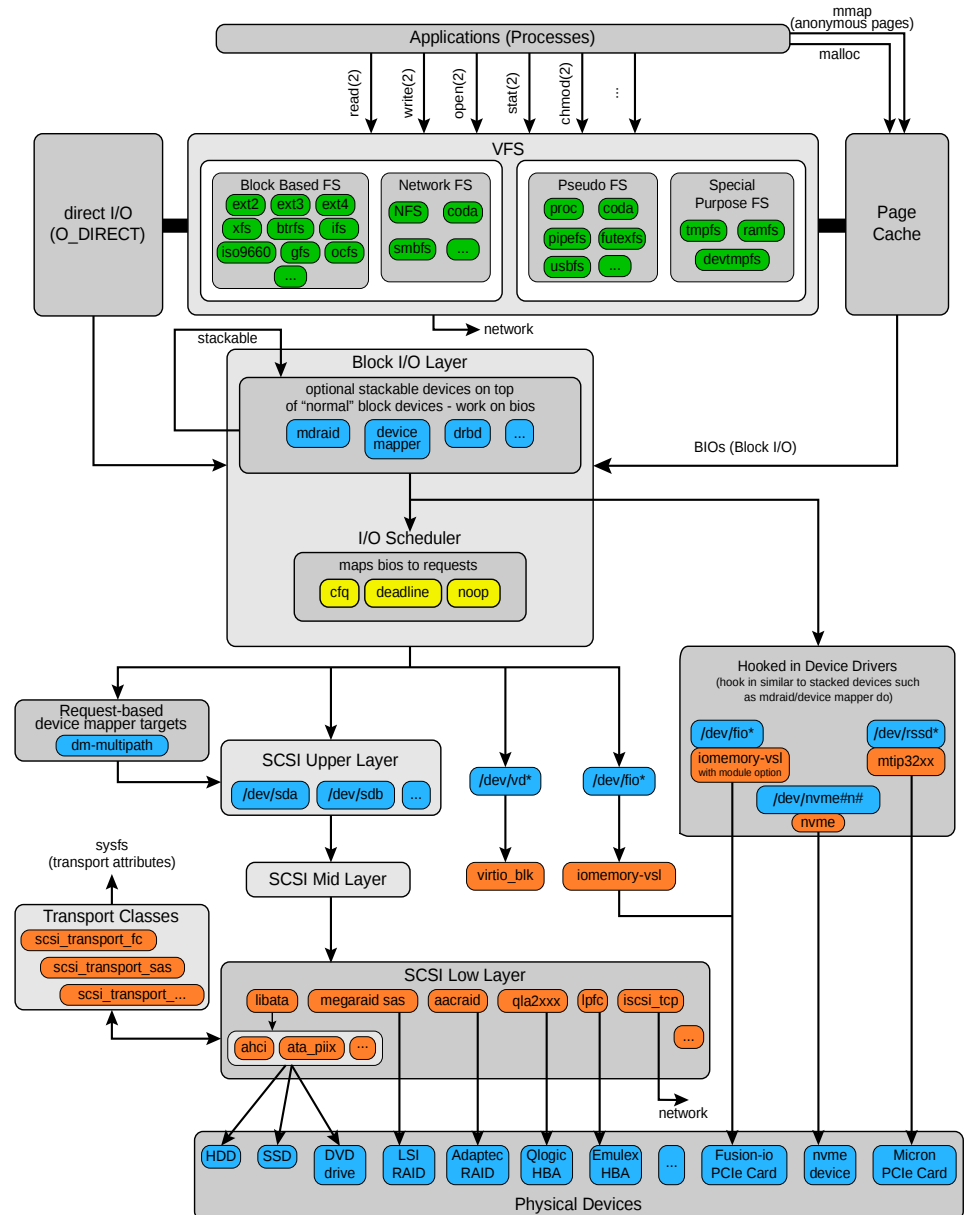
- végfelhasználó
- adminisztrátor
- programozó

## • Belső működés

- fájlrendszer interfészek
- kernel adatstruktúrák
- a háttértár szervezése
- virtuális fájlrendszerek

## • Adattárolás

- fizikai tárolók (HDD, SSD)
- I/O ütemezés
- tárolórendszer-virtualizáció:
  - helyi (RAID, LVM)
  - hálózati (SAN, NAS)
- elosztott fájl- és tárolórendszerek





# Fájlrendszerek megvalósítása (áttekintés)

- Felhasználói felület (rendszerhívások)
  - fájlok könyvtárak elhelyezése és kezelése
  - formázás, csatolás, lecsatolás stb.
  - ellenőrzés, javítás, paraméterek módosítása stb.
- Tárolás
  - logikai egységek → fizikai tárolók
  - **blokkos adattárolás**
  - adatok + metaadatok
  - szabad helyek nyilvántartása
- Belső működés
  - fájlrendszerek leírói (csatlakoztatott fájlrendszerek metaadatai)
    - csatlakoztatás nyilvántartása (elfedéssel)
  - fájlok leírói (metaadatok) a memóriában
    - kapcsolat a nyitott fájl objektumokhoz
  - beolvasott adatok elhelyezése a memóriában, puffereelés

# Tárolás: mit és hol?

- Metaadatok
  - partíciók típusai és elhelyezkedése
  - fájlrendszerek leírói (típus, méret, szabad helyek stb.)
  - fájlok (könyvtárbejegyzések) leírói (név, adatok elhelyezkedése stb.)
- Adatok
  - különféle rendszerindító programok (fájlrendszerekben és azokon kívül)
  - fájlok (és könyvtárbejegyzések) adatai (ezek tárolása a végső cél)
- Partíció
  - a tárolás legnagyobb fizikai egysége
  - fájlrendszer tárolására képes

# Tárolás a fájlrendszerekben

- A fájlrendszer (FS) felépítése

- FS metaadatok (szuperblokk, master file table, partition control block)
- (rendszerindulási adatok, ha ez egy boot partíció, boot control block)
- fájl metaadatok (inode, file control block, Windows: a master file table része)
- tárolt adatok



- A fájlrendszer metaadatai

## A háttértáron

- típus és méret
- szabad blokkok jegyzéke
- fájl metaadatok elhelyezkedése
- állapot
- módosítás információk
- ...

## A memóriában

- ami a háttértáron van
- csatlakozási információk
- „**dirty**” jelzőbit
- zárolási információk
- ...

- A fájlrendszer érzékeny a metaadatok elvesztésére (pl. blokkhiba)

- ezért másolatok készülnek, lásd `dumpe2fs /dev/sda1 | grep -i uperblo`
- demo: töröljük és állítsuk helyre a metaadatokat

# Fájlok metaadatainak elhelyezése

- **Diszken**

- hitelesítési információk (UID, GID)
- típus
- hozzáférési jogosultságok
- időbélyegek
- méret
- adatblokkok elhelyezkedése (lásd később)

Példa: UNIX inode (index node), Windows Master File Table bejegyzések

- **Memóriában** (nyitott fájl objektum) továbbá

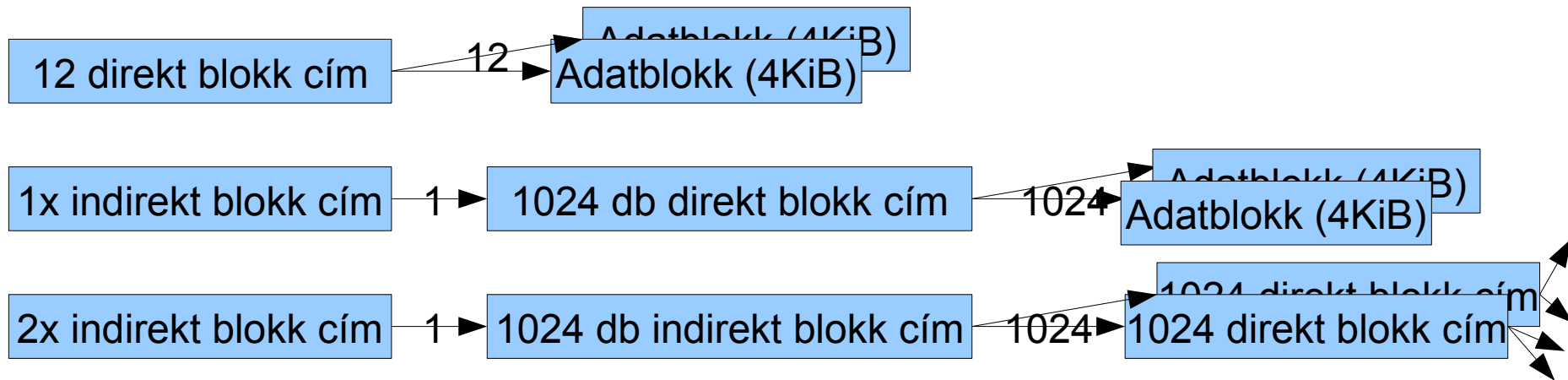
- státusz (zárolt, módosított, stb.)
- háttértár eszköz (fájlrendszer) azonosítója
- hivatkozás számláló (fájlleírók)
- csatlakoztatási pont leírója (elfedés)

# Adatblokkok allokációja

- Folytonos tárolás
  - törléssel egyre változatosabb méretű üres helyek keletkeznek
- Láncolt listás (soros hozzáférésű)
  - blokkokra bontott tartalom + hivatkozás további adatrészekre
  - pl. egyszeres láncolt lista
    - lassú a sokadik rész elérése
    - soros elérésre hatékony
    - érzékeny a hibákra (láncszakadás)
  - más variációk is léteznek, pl. a **FAT**, amely egy táblában épít listát blokkszámokból
- Indexelt (direkt elérésű)
  - blokkokra bontott tartalom + elhelyezkedési térkép (index)
  - ügyes elhelyezés: szekvenciális
    - szekvenciálisan (gyorsan) olvasható
    - az index segítségével direkt elérésű
  - gond lehet az index mérete
    - a túl nagy index nem fér el egy blokkban
    - láncolt listában tárolhatjuk

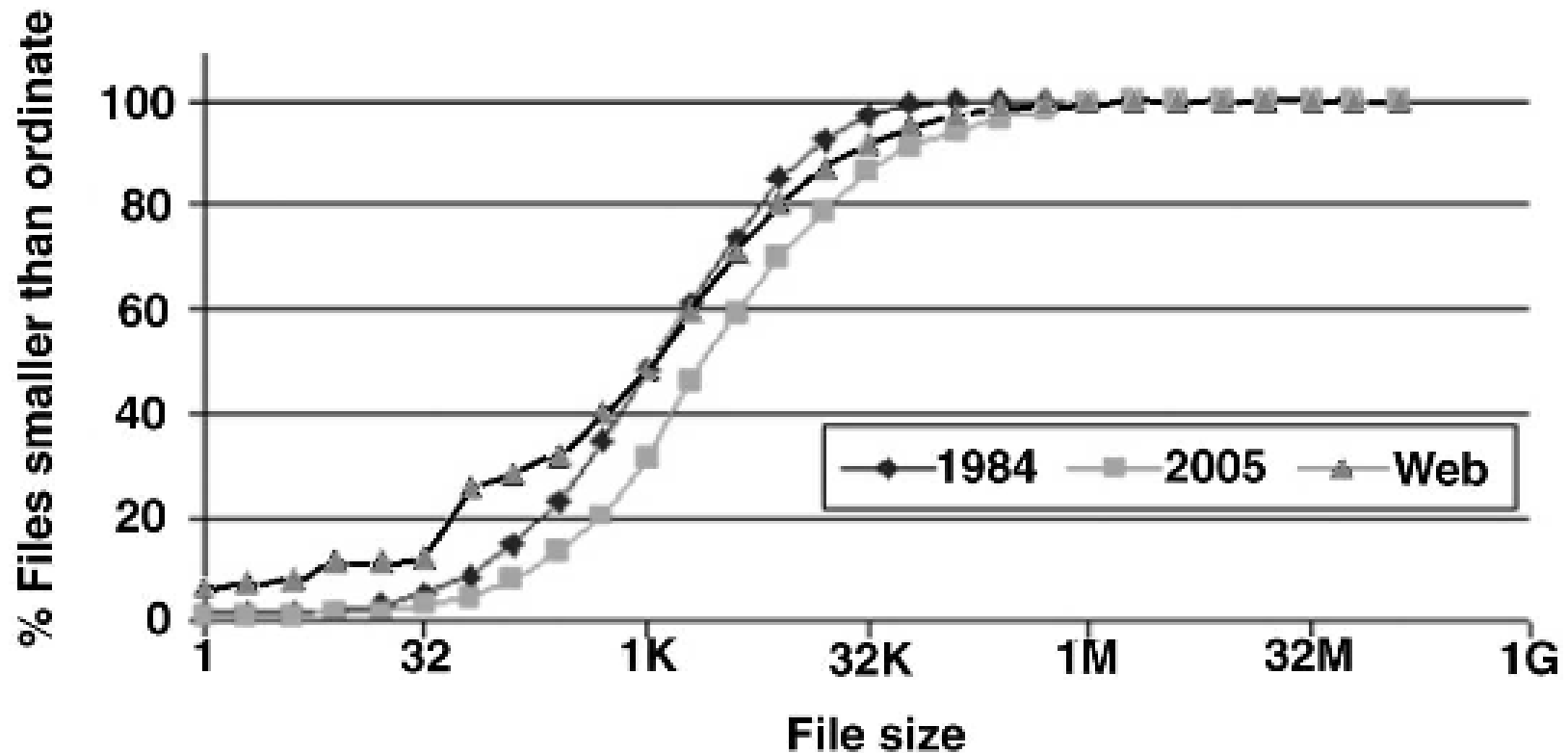
# Példa: többszörösen indirekt adatblokk címtábla

- A címtábla és az adattárolás jellemzői (példa)
  - 4 byte-os címek
  - 12 db direkt blokkcím
  - 1x és 2x indirekt blokkcímek
  - 4KiB-os blokkméret ( $4\text{KiB} / 4 = 1024$  cím tárolására képes)



„Mekkora a maximális fájl méret?”

# Hogyan válasszuk meg az adatblokkok méretét?



Forrás: Andrew S. Tanenbaum, Jorrit N. Herder, Herbert Bos

File size distribution on UNIX systems: then and now. Operating Systems Review 40(1): 100-104 (2006)

# Üres helyek menedzselése

- Bittérképes, bitvektoros
  - egy blokk egy bit (1 = szabad, 0 = foglalt)
  - egyszerű, és könnyű szabad blokkot találni
    - akár a memóriában is elérhet
    - egy CPU utasítás elég lehet
  - nagy fájlrendszereknél egyre kevésbé hatékony
- Láncolt listás
  - minden üres blokk egy következőre mutat
  - csak az első szabad blokk címét kell megjegyezni
  - egyszerű, bár nem a leghatékonyabb módszer (diszkműveletek)
  - összeolvasztható a láncolt listás (pl. FAT) adatblokk nyilvántartással
- Hierarchikus módszerek
  - üres helyek csoportjait kezelik (hasonlítanak a többszörös indexelt címtáblára)
  - csoportok létrehozhatók pl. a fájlrendszer mérete alapján és globálisan kezelhetők
  - csoporton belül egyszerű belső struktúra használható (pl. mind szabad, térkép stb.)



# Adatblokkok gyorsítárazása

- Diszkpufferelés (disk buffering)
  - a memóriát használja gyorsítótárként: **blokkgyorsítótár (buffer cache)**
  - növeli a hatékonyságot, írásnál csökkenti a megbízhatóságot
  
- A blokkgyorsítótár szervezése
  - ismételés: a SZGA „cache szervezés”
  - használhatjuk a virtuális tárkezelés mechanizmusait
    - egységes blokkgyorsítótár (unified buffer cache)** (Linux: page cache)
  - olvasás gyorsítása
    - előreolvasás (readahead)**, beállítható, lásd [posix\\_fadvise](#)
  - nem használt blokkok törlése: pl. LRU
  - írásműveletek kezelése (mikor írjuk ki a háttértárra)
    - **írástáteresztő gyorsítótár (write through cache)**
      - azonnal háttértárra ír
      - lassú, de megbízható
    - **pufferelt gyorsítótár**
      - csak időnként írjuk ki (flush, sync)
      - nagyobb teljesítményű, de kevésbé megbízható

# Metaadatok konzisztenciája

- Konzisztencia-problémák
  - módosított adatok gyorsítárazása
  - metaadatok változásai (fájlok, könyvtárak, fájlrendszerek) és a gyorsítárazás
- Metaadatok sérülése
  - komolyabb kiterjedésű lehet a hatás
  - pl. tárhelyelszivárgás (storage leak): inode törlés, összeomlás (adattörlés előtt)
  - akár a teljes fájlrendszer összeomlásához is vezethet
- Megoldási ötletek
  - adatok esetén: írásáteresztő gyorsítótár
    - lassítja a működést, de ésszerű, ahol nagy a kockázat
  - metaadatokra is működik?
    - önmagában nem, hiszen hosszabb tranzakciókról van szó

# Naplózó fájlrendszerek (journaling)

- **Napló (journal)**
  - szekvenciálisan írható körpuffer a **háttértáron**
  - az elvégzendő **műveleteket tartalmazza** (metaadat [+ adat])
  - pl. **NTFS LFS**, Linux ext3/4 stb.
- **Megvalósítás: tranzakcióalapú működés**
  - a tranzakció akkor zárul, amikor kiírt minden műveletet a naplóba
  - a naplóba írt tranzakciókat dolgozza fel és hajtja végre a fájlrendszeren
- **Mi történik, ha összeomlik a rendszer?**
  - Induláskor feldolgozza a naplót.
- **Log-structured fájlrendszer:** a napló a fájlrendszer (**cikk**, **példák**)
  - gyors szekvenciális írás (pufferelt), olvasás térképpel, szemétgyűjtés
- **Más megoldás is lehetséges: Copy-on-write fájlrendszer**
  - az írásműveleteket másolt adatokon hajtja végre, majd átírja a metaadatokat
  - pl. **ZFS**, **btrfs**

# Áttekintés

## • Felhasználói szemmel...

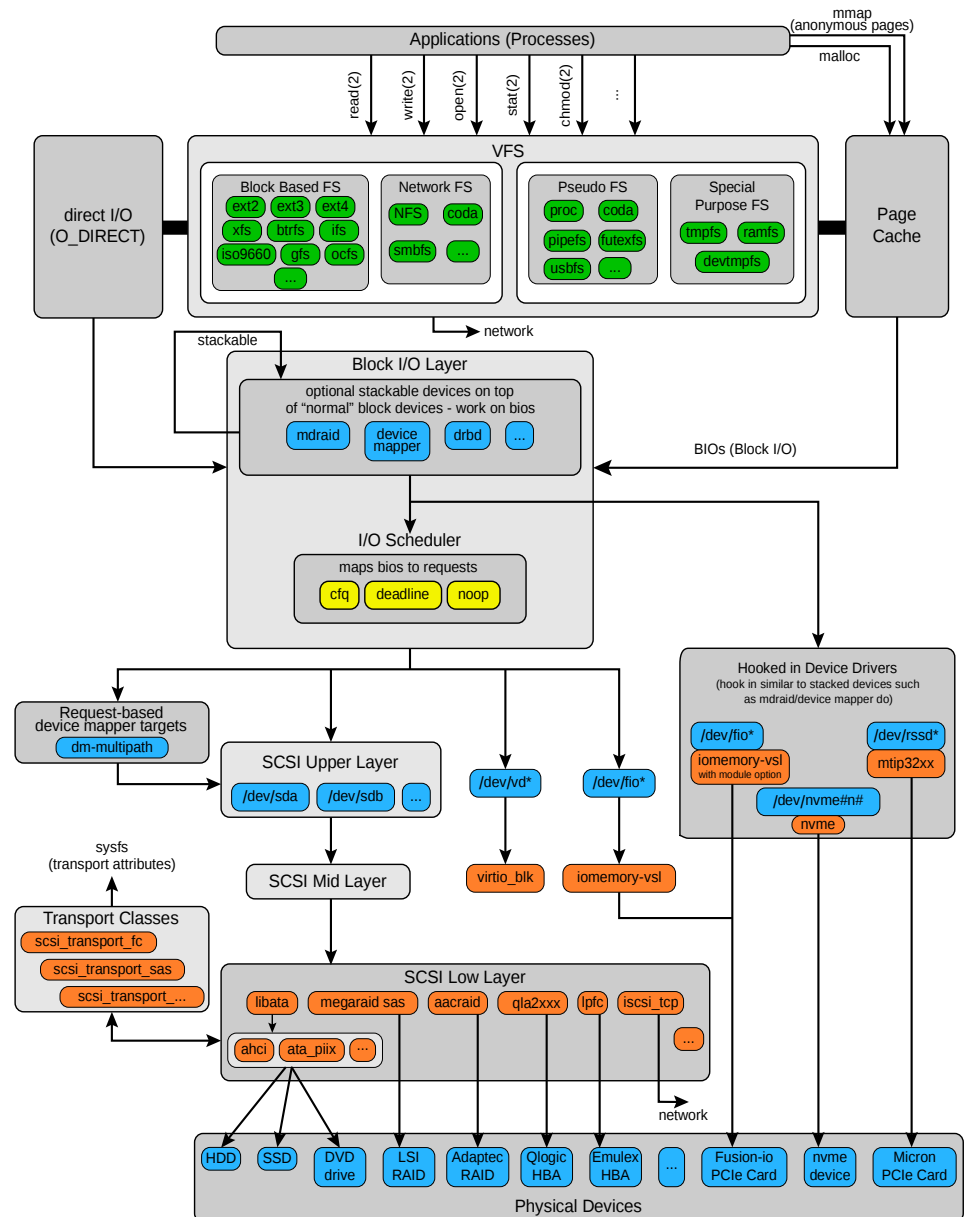
- végfelhasználó
- adminisztrátor
- programozó

## • Belső működés

- fájlrendszer interfészek
- kernel adatstruktúrák
- a háttértár szervezése
- virtuális fájlrendszerek

## • Adattárolás

- fizikai tárolók (HDD, SSD)
- I/O ütemezés
- tárolórendszer-virtualizáció:
  - helyi (RAID, LVM)
  - hálózati (SAN, NAS)
- elosztott fájl- és tárolórendszerek



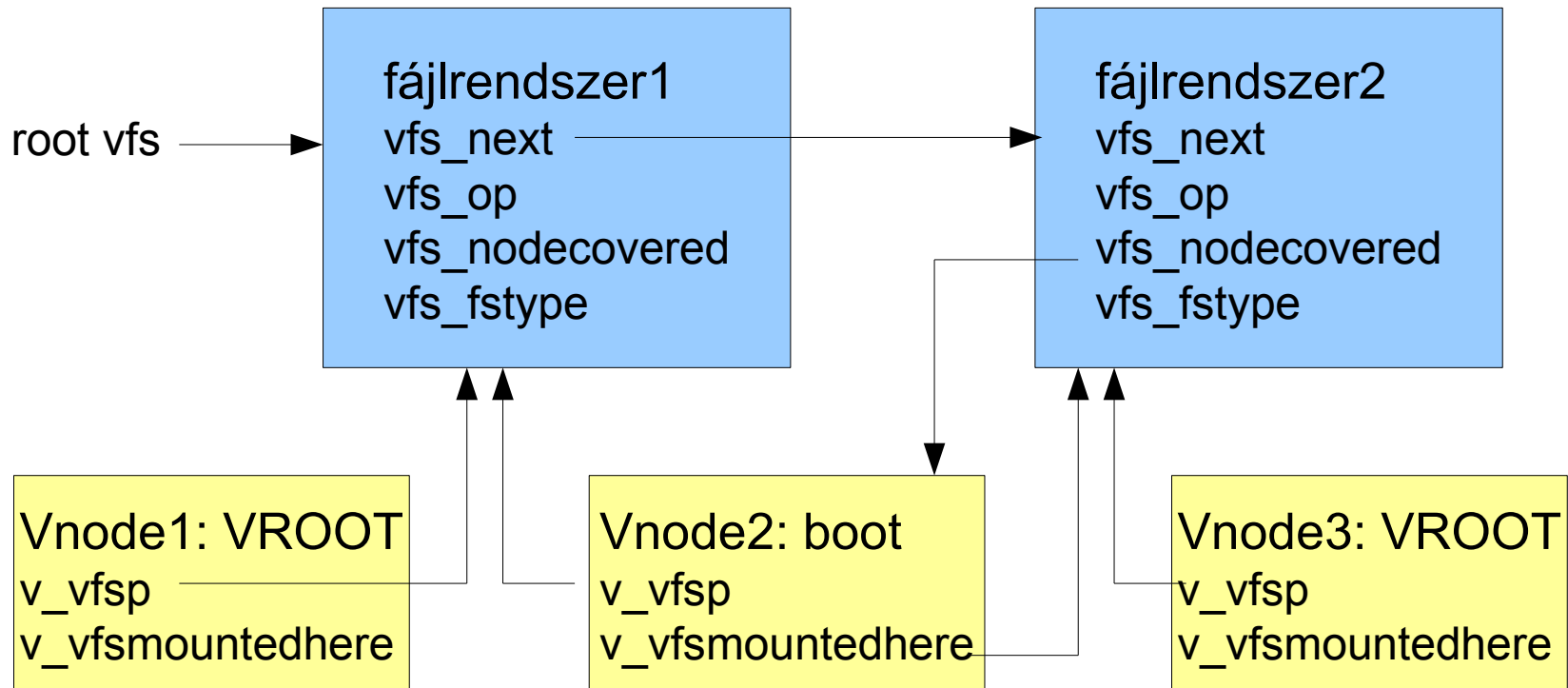
# A virtuális fájlrendszer (VFS)

- Nagyon sokféle fájlrendszer létezik
  - főleg UNIX alatt jellemző, hogy egy időben is sokfélét használ
  - a programozóktól (és kernelfejlesztőktől) nem várható el azok különálló kezelése
- A VFS egy implementáció-független fájlrendszer absztrakció
  - a modern UNIX fájlrendszerek alapja
- Célok:
  - többféle fájlrendszer egységes egyidejű támogatása
  - egységes kezelés a csatlakoztatás után (programozó IF)
  - speciális fájlrendszerek uniform megvalósítása (hálózati, /proc stb.)
  - modulárisan bővíthető rendszer
- Az absztrakció lényege
  - fs (fájlrendszer metaadatok) → vfs
  - inode (fájl metaadatok) → vnode

# A vnode és a vfs

- vnode adatmezők
  - közös adatok (típus, csatlakoztatás, hivatkozás száml.)
  - `v_data`: állományrendszertől függő adatok (inode)
  - `v_op`: az állományrendszer metódusainak táblája
- vfs adatmezők
  - közös adatok (fájlrendszer típus, csatlakoztatás, hivatkozás, `vfs_next`)
  - `vfs_data`: állományrendszertől függő adatok
  - `vfs_op`: az állományrendszer metódusainak táblája
- virtuális függvények
  - `vnode`: `vop_open()`, `vop_read()`, ...
  - `vfs`: `vfs_mount` `vfs_umount` `vfs_sync`
  - az állományrendszernek megfelelő hívásokra képződnek le
- segédrutinok, makrók

# A vfs és a vnode kapcsolata



# Speciális VFS fájlrendszerek (példák, demók)

- Milyen fájlrendszereket támogat a Linux?

```
cat /proc/filesystems
```

- devtmpfs és devfs
  - a hardvereszközök elérése fájlrendszeri interfészen keresztül
- **procfs**
  - taszkok adatainak és kernel adatstruktúrák elérésére
- **sysfs**
  - kernel alrendszerek elérése fájlműveletekkel
- **cgroup, cpuset**
  - folyamatcsoportok erőforrás-allokációinak beállítása

```
mount | egrep "cgroup|cpuset"
```



# Saját fájlrendszer készítése VFS alapon

- Otthoni gyakorlat (nem nehéz...)
- Dokumentáció

Ravi Kiran, „[Writing a Simple File System](#)”

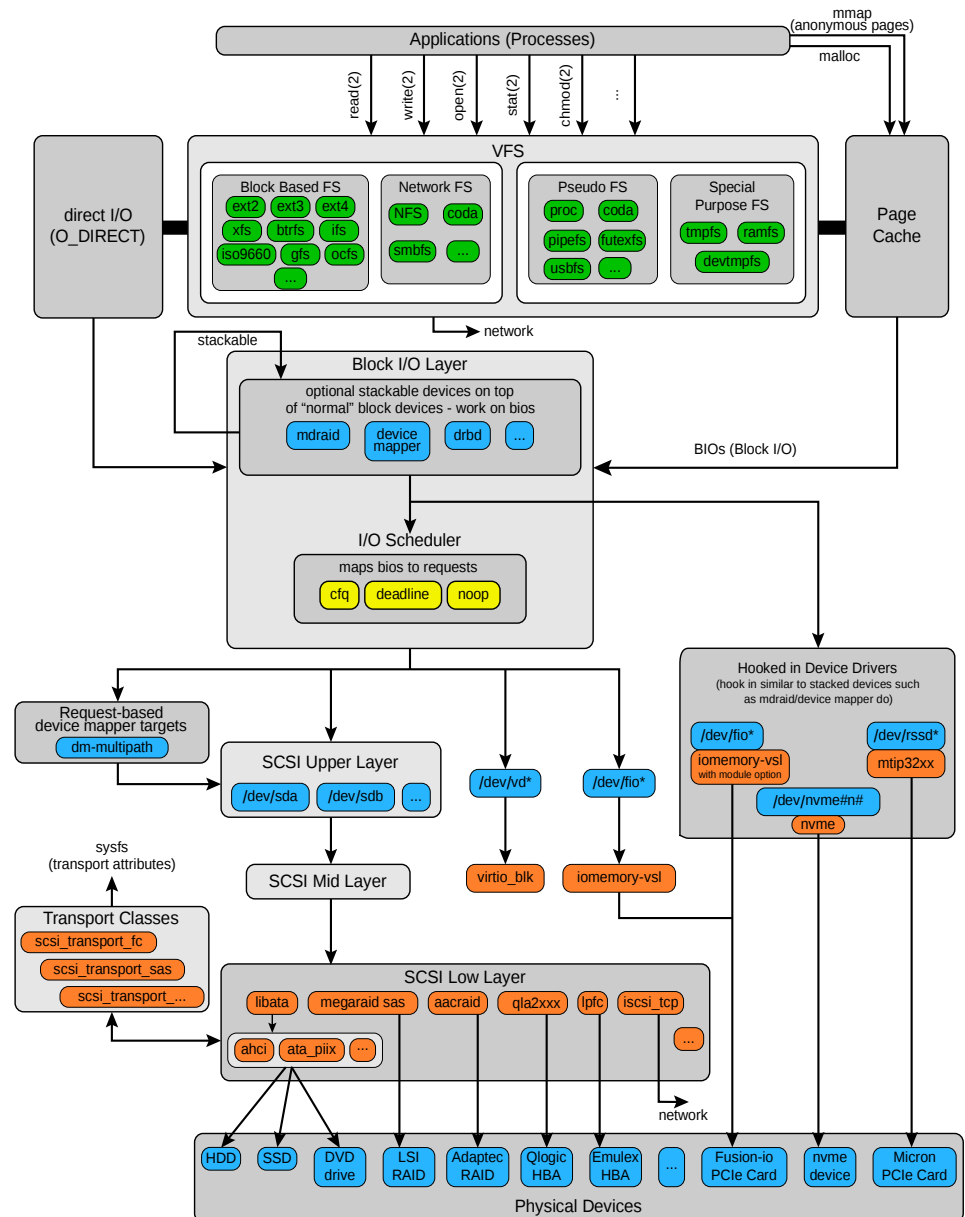
Steve French, „Linux Filesystems 45 minutes” [ODP PDF](#)

„A Step by Step Introduction to Writing (or Understanding) a Linux Filesystem”

Az IBM mérnöke, aki SAMBA fejlesztéssel foglalkozik

# Áttekintés

- Felhasználói szemmel...
  - végfelhasználó
  - adminisztrátor
  - programozó
- Belső működés
  - fájlrendszer interfészek
  - kernel adatstruktúrák
  - a háttértár szervezése
  - virtuális fájlrendszerek
- Adattárolás
  - fizikai tárolók (HDD, SSD)
  - I/O ütemezés
  - tárolórendszer-virtualizáció:
    - helyi (RAID, LVM)
    - hálózati (SAN, NAS)
  - elosztott fájl- és tárolórendszerek



# Tárolási megoldások a fájlrendszerek mögött

- Fizikai tárolóeszközök
  - mágneses elven működő
    - pl.: HDD és szalagos egységek (tape)
  - optikai elven működő
    - pl.: CD / DVD / Blu-ray
  - nemfelejtő memóriaalapú eszközök
    - pl.: SSD, USB flash diszk, SD kártya stb.
- Virtualizált tárolórendszerek
  - a fizikai tárolóeszközökre építenek egy (vagy több) további szolgáltatási réteget
    - összeolvasztanak tárolóeszközöket
      - megbízhatóság- és kapacitásnövelés érdekében
      - pl. RAID, LVM
    - hálózati elérést tesznek lehetővé
      - fájl vagy blokkszintű adatátvitellel
      - pl. NAS, SAN
    - elosztott tárolási rendszert valósítanak meg
      - megbízható és jól skálázható tárolórendszerek építéséhez
      - pl. Ceph, GlusterFS

# Fizikai tárolórendszerek legfontosabb jellemzői

## • Teljesítmény

- kapacitás: 32/64 bit → TB
- sebesség: 10 MiB/s → 200 GiB/s
- késleltetés: 0,5 ns → perc

## • Megbízhatóság

gyártói és tapasztalati adatok (lásd még [SMART](#))

- **éves hibaarány (annualized failure rate, AFR)**
  - jellemzően 2-4%, esetenként >10% tönkremegy

- **a meghibásodásig eltelt átlagos idő (mean time to failure, MTTF)**

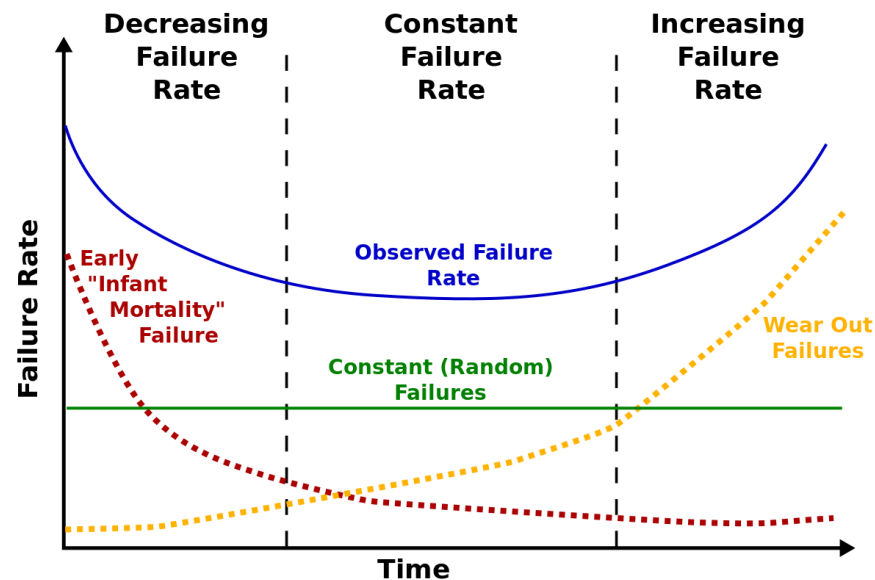
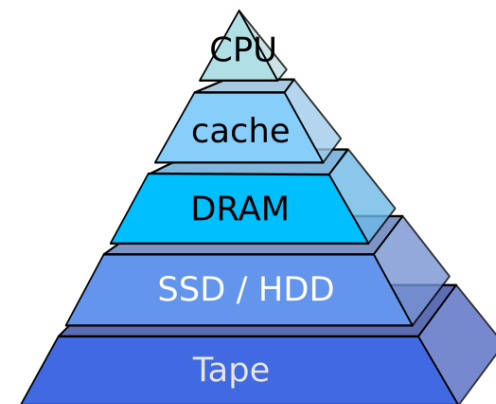
- gyártók: >100 év (eszközhalmazra)

[fürdőkád-görbe](#)

MTTF of 1,000,000 hours?

- **teljes írható adatmennyisége (SSD, tot**

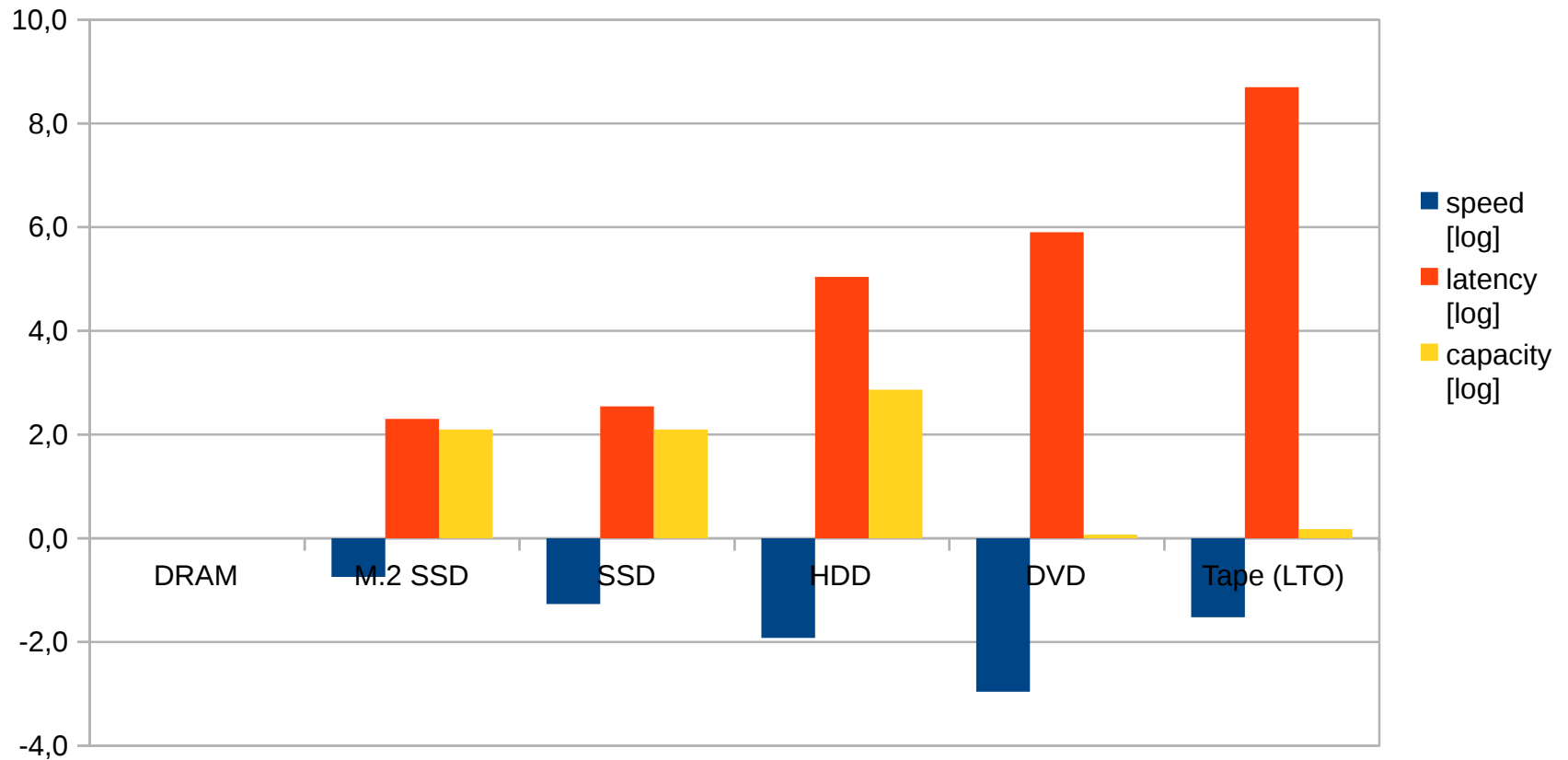
- a memóriacellák korlátos számmal í
- napi 50GB esetén is [évtizedekben mérhe](#)



# Tárolóeszközök teljesítménye

Fizikai tárolók teljesítménye a DRAM-hoz képest

A sebesség, a késleltetés és a tárolókapacitás logaritmikus összehasonlítása



# Trendek a fizikai tárolórendszerekben

- „A lassú I/O” évtizedei
  - a processzorok teljesítménye dinamikusabban nőtt
  - adatelérési késleltetésük jelentősen csökkent
  - a háttértárak a kapacitásra koncentráltak
- Változó teljesítményviszonyok
  - sok RAM → nagy puffer gyorsítótár
  - gyors CPU → I/O teljesítménynövelés
    - futásidejű adattömörítés (pl. btrfs, zfs)
    - deduplikáció
  - memóriaalapú háttértárak
    - növekvő sebesség, minimális késleltetés
    - „**storage class memory**”: DRAM-hoz közelítő tárolórendszer
- Következmények

Eltűnik az elsődleges – másodlagos határ.  
Az I/O-ra vár állapot időtartama csökken.

# Szalagos tárolók (tape drive)

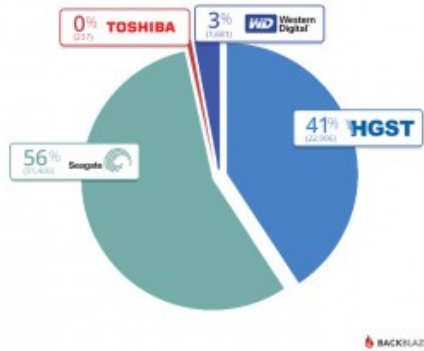
- Biztonsági mentésre
    - nagy kapacitás
    - hosszú élettartam
    - lassú
    - drága automatizálás
  - Trendek
    - szekvenciális olvasási sebesség:
      - Tape: 300 MB/s
      - SSD: 500 MB/s
    - HDD → SSD / Tape?
    - nagy puffer gyorsítótárak
      - a taszkok onnan dolgoznak
      - szekvenciális olvasással töltik
    - log-strukturált fájlrendszerek
      - szekvenciálisan ír/olvas
- adattárházakban újra feltűnhetnek



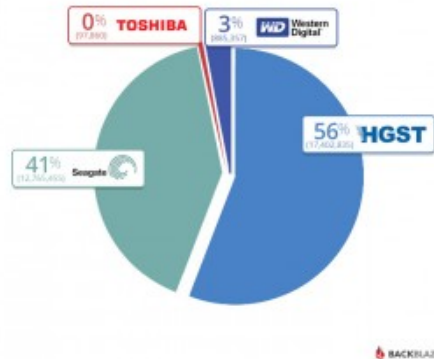
# Merevlemezés meghajtók megbízhatósága

## A Backblaze adatcenter kb. 56 ezer diszkjének statisztikái

Backblaze Datacenter Drive Count  
by Manufacturer  
as of 12/31/2015

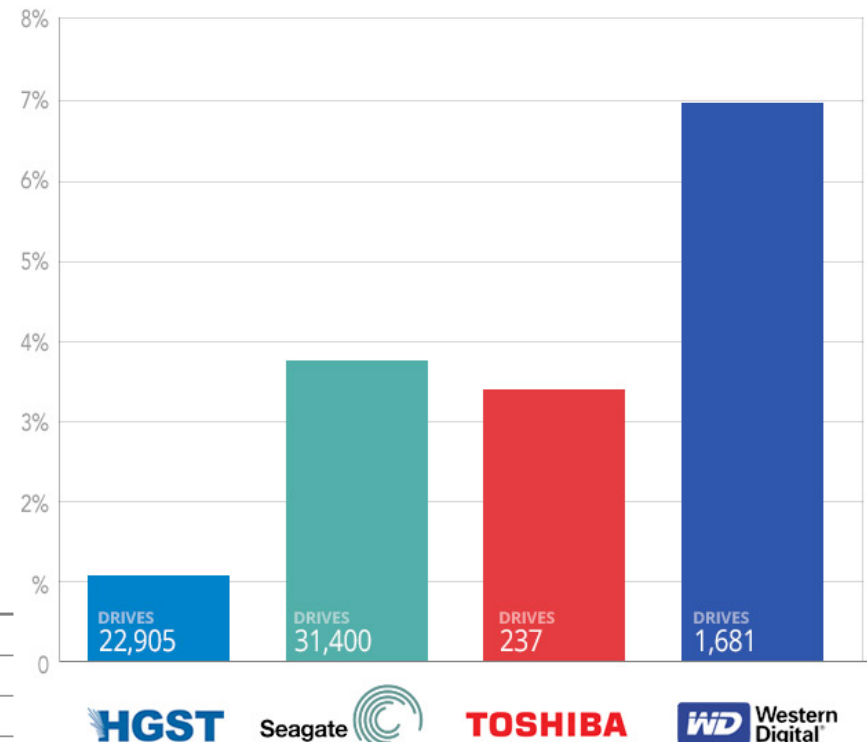


Backblaze Datacenter Drive Days  
in Service by Manufacturer  
as of 12/31/2015



Failure Rate by Manufacturer

Cumulative from 4/2013 to 12/2015



## Meglepő adatok is akadnak:

Cumulative Failure Rate through the Period Ending

MFG	Model #	Highest QTY	12/31/13	12/31/14	12/31/15
HGST	HDS5C3030ALA630	4,596	0.9%	0.7%	0.8%
HGST	HDS723030ALA640	1,022	0.9%	1.8%	1.8%
Seagate	ST3000DM001	4,074	9.8%	28.3%	28.3%
Seagate	ST33000651AS	325	7.3%	5.6%	5.1%
Toshiba	DT01ACA300	58	-	4.8%	3.8%
WDC	WD30EFRX	1,105	3.2%	6.5%	7.3%

(HGST korábban Hitachi Global Storage Technologies)

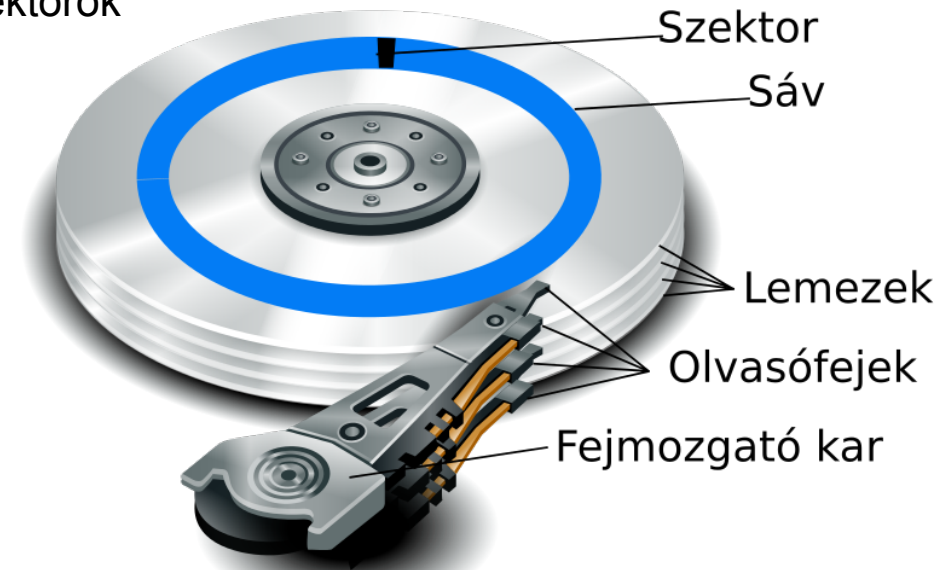


Forrás: <https://www.backblaze.com/blog/hard-drive-reliability-q4-2015/>



# Allokáció merevlemezes meghajtókon

- Szuperblokk, inode lista és adatblokkok elhelyezése
  - teljesítmény? megbízhatóság?
- Cilinder (blokk) csoport
  - azonos fejpozícióhoz tartozó sávok
  - a fej mozgatása nélkül olvasható szektorok
  - egyszerre sérülnek a fej miatt
- Allokációs elvek
  - szuperblokk másolása minden cilindercsoportba
  - inode lista és szabad blokkok csoportonként kezelve
  - egy könyvtár – egy csoport
  - kis fájlok egy csoportba
  - nagy fájlok „szétkenve” több csoportba
  - új könyvtárnak egy új, kevésbé foglalt csoportot keres



# Diszk feladatok ütemezése (Linux)

## Noop (FIFO): összevonhat szomszédos kéréseket

- minimális terhelést jelent
- ha a tároló maga is ütemez (pl. HW RAID, NCQ, virtualizált rendszerek stb.)
- ha nincs értelme ütemezni (pl. RAM diszk, SSD)
- ha nincs nagy I/O terhelés (CPU-intenzív rendszer)

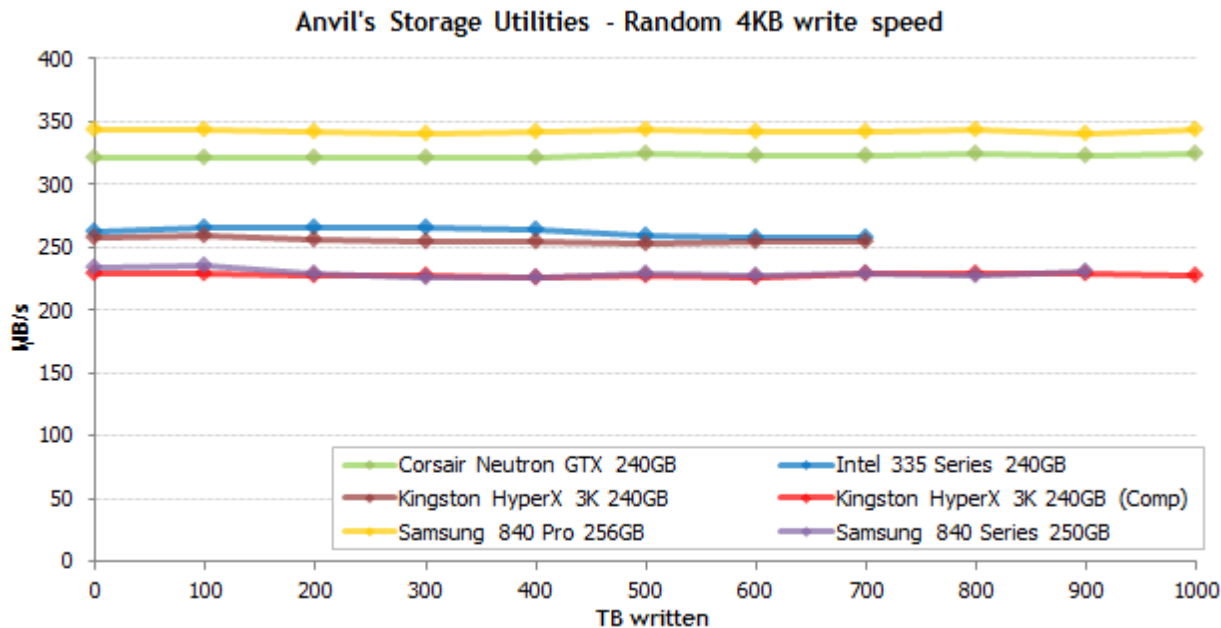
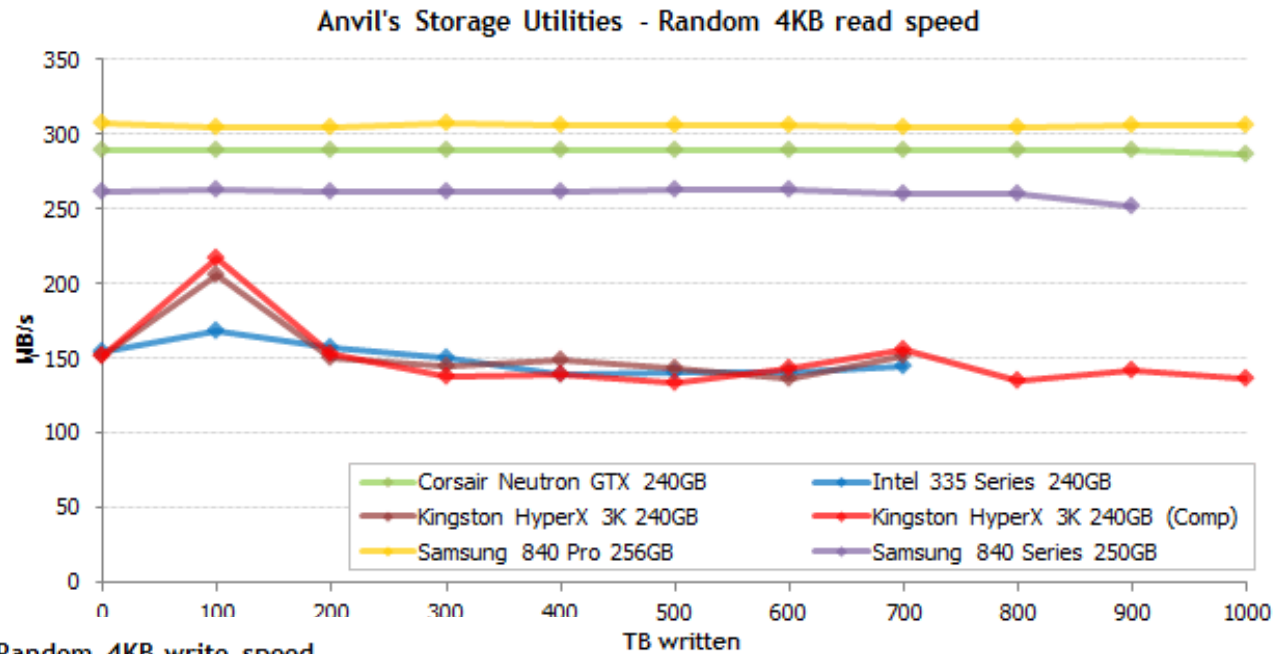
## Deadline: késleltetésre optimalizál

- blokkcím szerint rendezett írási vagy olvasási kötegek
- nagy I/O terhelésnél **globálisan** jó (egy taszkra nem feltétlenül előnyös)

## CFQ (Completely Fair Queuing): egyenletes kiszolgálás

- folyamatonkénti sorok, azokhoz rendelt I/O időszeletek
- a sorokhoz prediktív előrejelzés
- általános célra egy kiegyensúlyozott ütemező
- jellemzően ez az alapértelmezett
- konfigurálható: `man ionice`

# SSD megbízhatóság



Napi 50GB írás esetén is kb. 40 év a leggyengébb SSD élettartama.

Forrás: <http://techreport.com/review/24841/introducing-the-ssd-endurance-experiment>

# Áttekintés

## • Felhasználói szemmel...

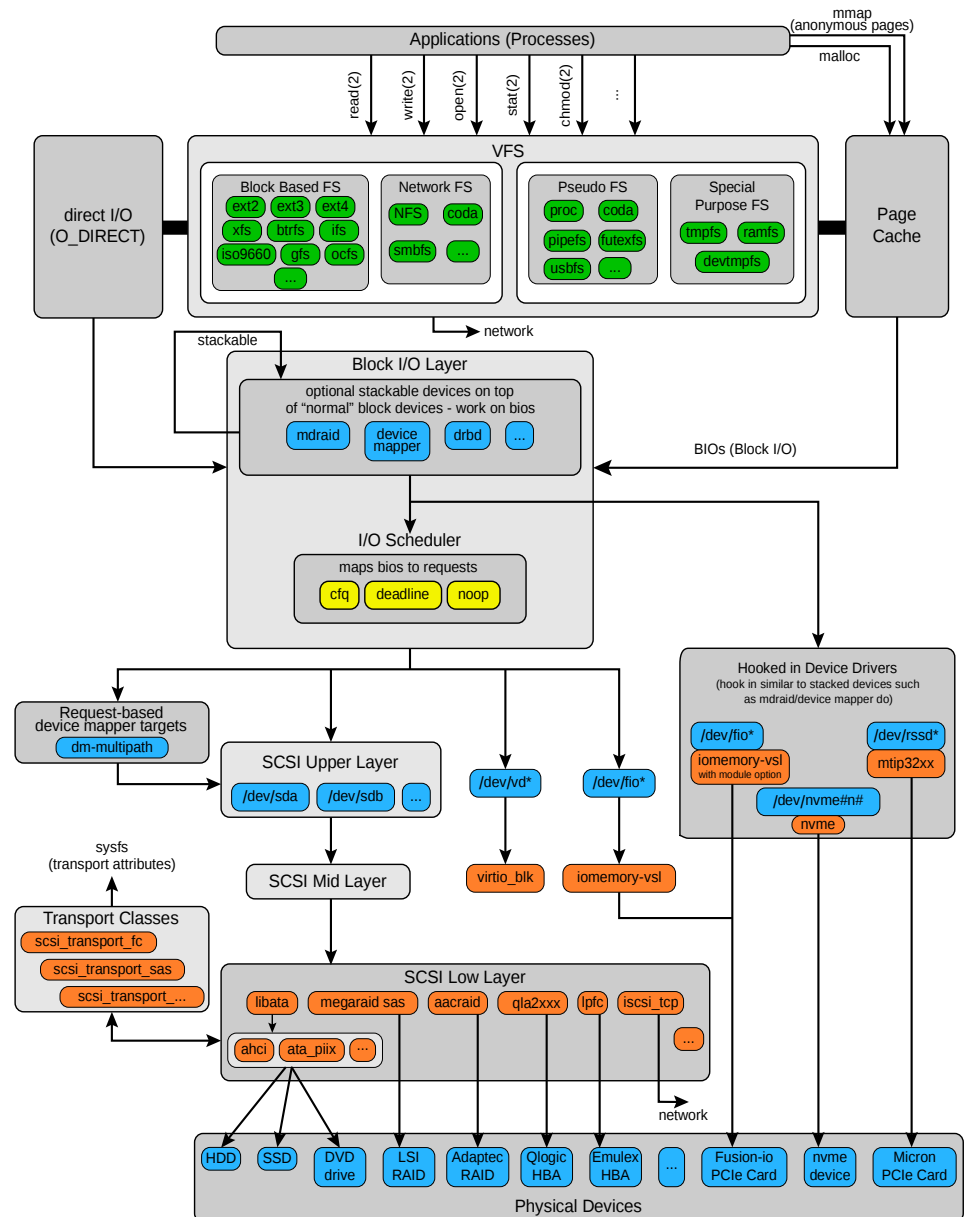
- végfelhasználó
- adminisztrátor
- programozó

## • Belső működés

- fájlrendszer interfészek
- kernel adatstruktúrák
- a háttértár szervezése
- virtuális fájlrendszerek

## • Adattárolás

- fizikai tárolók (HDD, SSD)
- I/O ütemezés
- tárolórendszer-virtualizáció:
  - helyi (RAID, LVM)
  - hálózati (SAN, NAS)
- elosztott fájl- és tárolórendszerek



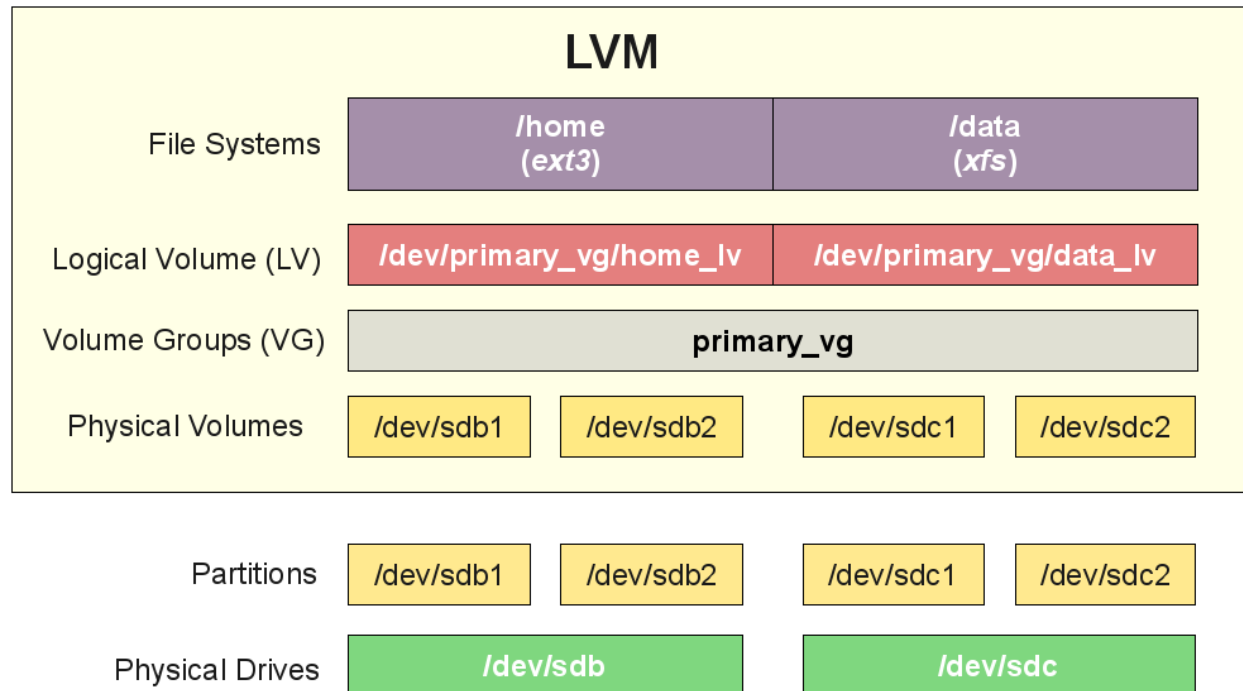
# Virtuális tárolórendszerek

- Fizikai tárolórendszerek korlátai
  - kapacitás, teljesítmény, megbízhatóság
  - menedzsment (rugalmatlan)
  - hibaelhárítás
- Virtualizáció
  - a fizikai eszközökre épít egy (vagy több) további szolgáltatási réteget
    - összeolvasztás (kapacitásbővítés)
    - szolgáltatásbővítés
    - jobb menedzsment
- Virtuális tárolórendszer
  - a fizikai eszközök határain átnyúló tárolórendszer
  - példák: LVM, RAID stb.
  - építőelemei
    - fizikai tárolók: diszk, partíció
    - más virtuális tárolók, pl. RAID diszkek

# Logikai kötetkezelés (logical volume management)

Windows: Logical Disk Manager    Linux: Logical Volume Manager

- **fizikai kötet** (physical volume, PV): diszk, partíció stb.  
részei: **physical extent (PE)**
- **logikai kötet** (logical volume, LV): virtuális diszk partíció ← **fájlrendszer**  
részei: **logical extent (LE)**, amelyek PE-kre képződnek le
- (logikai) **kötetcsoport** (logical volume group, VG): LV-k halmaza, **a virtuális tároló**

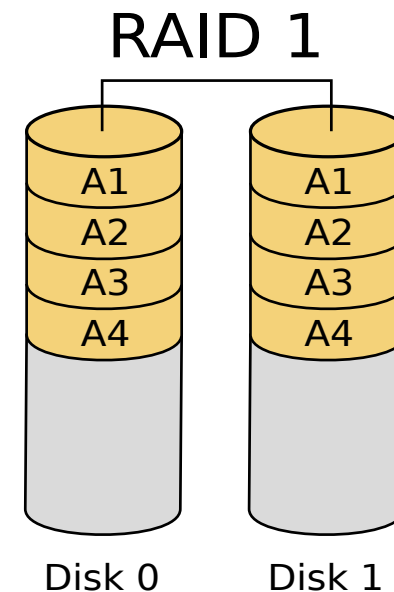
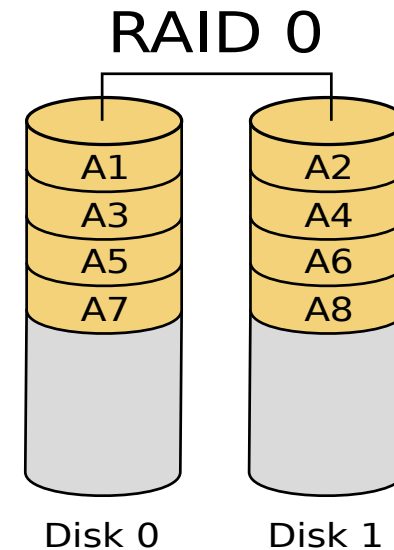


# RAID

- Tárolórendszerek megbízhatósága
  - a fizikai eszközök számának növekedésével **nő** a hiba esélye is
  - pl.: 1 diszk MTTF 100 000 óra, 100 diszk MTTF 1000 óra (41 nap)
  - több eszköz → kisebb **nagyobb** megbízhatóság
- Adatredundancia beépítésével elérhető
  - **tükrözés** (mirroring)
    - költséges (a kapacitás jelentősen csökken)
  - **paritás**: hibajelzés mellett javítására is szolgálhat
    - pl. N egyforma blokk mellé 1 blokk paritást rendelünk
- Redundant Array of Inexpensive Disks
  - virtuális tárolórendszer
  - „olcsó” („kis”) merevlemezek egybeolvasztásával
    - I = Independent, a RAID diszkek nem olcsók
  - cél: **redundancia** (megbízhatóság) és a **teljesítmény**
  - hardveres és szoftveres megvalósítása is létezik
    - az alaplap RAID szoftveres
    - hardveres RAID nem olcsó

# RAID szintek: 0 - 1

- **RAID szint (RAID level)**
  - az egybeolvasztás módja
- **RAID 0 (stripe, „csíkozás”)**
  - N diszken egyenletesen terít
  - cél: a teljesítmény növelése
  - a diszkek kapacitása összeadódik
  - diszkhiba esetén az adat elvész  
→ átgondoltan alkalmazandó
- **RAID 1 (mirror, **tükrözés**)**
  - az adatokat többszörözve tárolja
  - cél: megbízhatóság
  - mérete egy diszk kapacitása  
(a többi másolatot tárol)
  - az írás lassul (másolatok)
  - az olvasás mérsékelten gyorsabb

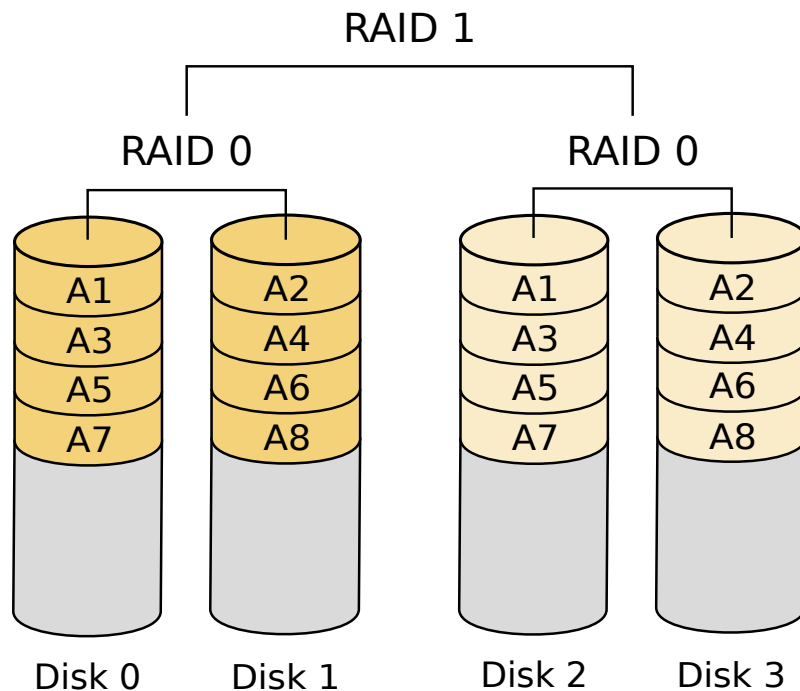




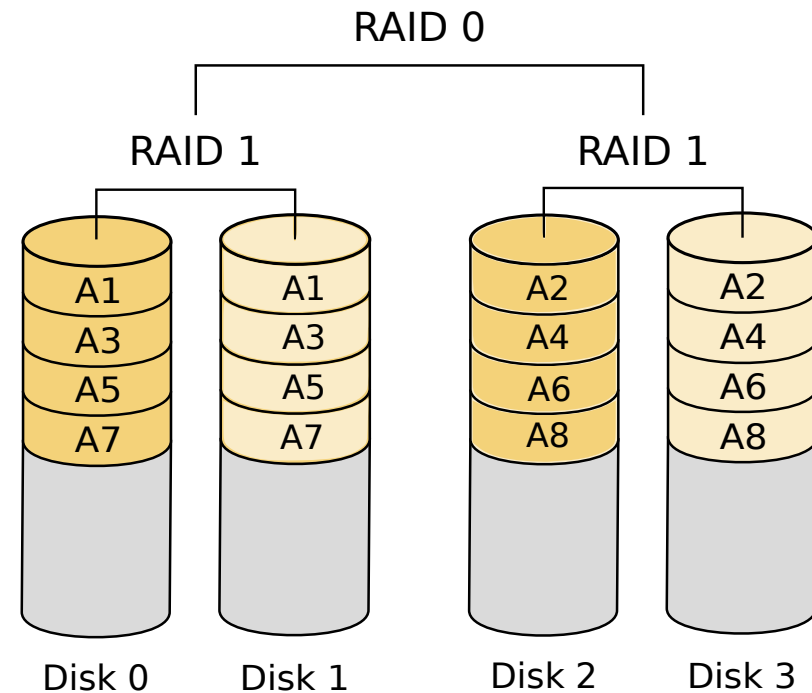
# RAID szintek: 01 és 10

- A két alap RAID szint ötvözhető is:
  - RAID 01** (0+1): „mirror of stripes”
    - elvi felépítés, a gyakorlatban nem használt
  - RAID 10** (1+0): „stripe of mirrors”
    - egyszerűbb I/O-intenzív rendszerekben ajánlott

## RAID 0+1



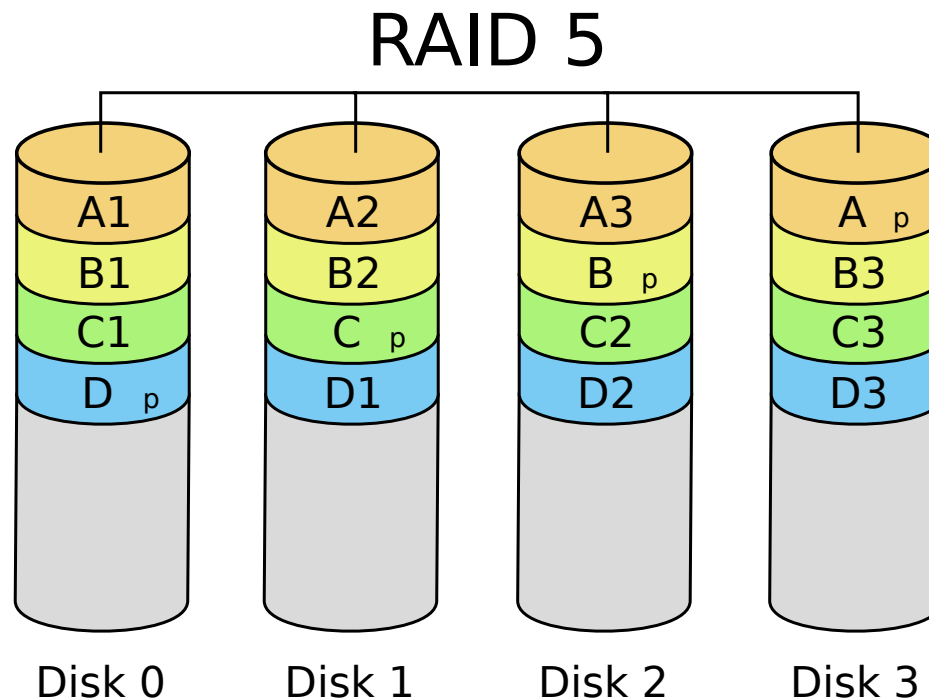
## RAID 1+0



# RAID 5: Blokkszintű csíkozás egy paritással

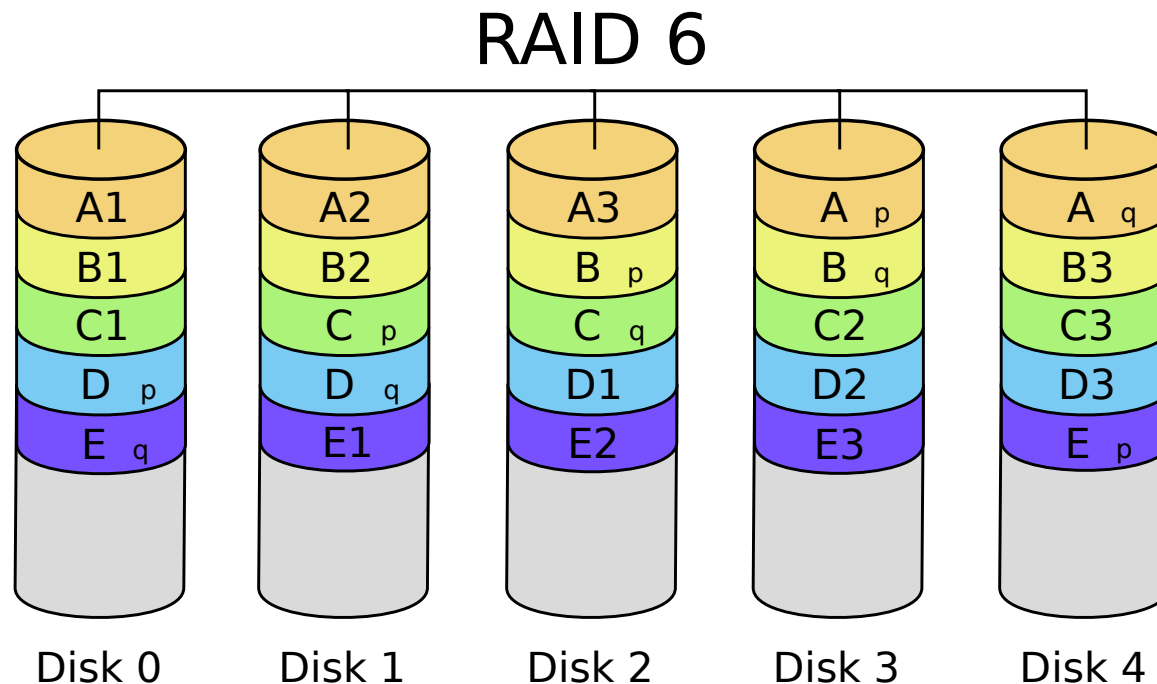
- Felépítés
  - N adatblokk + 1 paritásblokk (N+1 diszk)
  - a paritásblokkokat egyenletesen („csíkozva”) helyezi el a fizikai diszkeken
- Jellemzés
  - a teljesítménye a RAID0-hoz közeli
  - a kapacitás egy diszk méretével csökken
  - egy diszk meghibásodása ellen véd

„néma hiba” (silent error)



# RAID 6: blokk szintű csíkozás két paritással (N+2 diszk)

- Felépítés  
RAID5 + második paritásblokk
- Jellemzők
  - két diszk hibája ellen véd
  - jó teljesítmény, mérsékelt kapacitáscsökkenés
  - a helyreállításkor jelentkező néma hiba ellen is véd



# A RAID korlátai

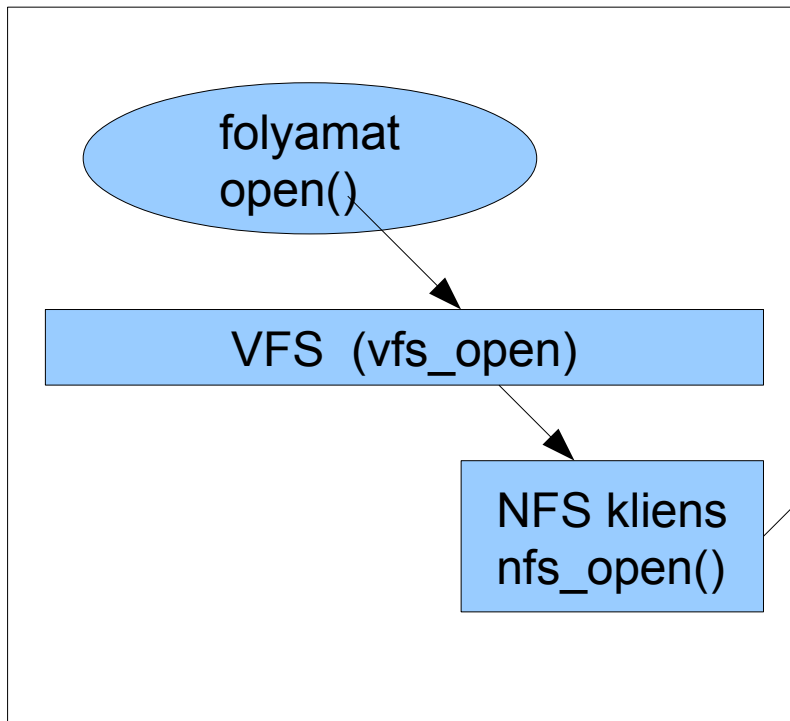
- Mennyi ideig tart egy RAID5 hiba javítása 4+1 diszk esetén?
  - 150GB-os diszkek: kb. 10 óra (egy hosszú éjszaka)
  - 6TB-os gyorsabb diszkek: kb. 80 óra (több nap)
  - meleg tartalék (hot spare) + RAID6 a legjobb, de nem elég
- Sok egyforma diszket kíván
  - egyre nehezebb a pótlás, a HW RAID nagyon érzékeny erre
- Kötött redundanciájú, nem rugalmas
  - RAID5 → RAID6 migráció futásidőben?
- A tárolókapacitás nem növelhető nagyra
  - 6-8 diszk / HW RAID kártya
  - a kiépítés fizikai korlátai
- Csak diszkhiba ellen véd
  - alaplap, CPU, memória, táp stb.?

# Hálózati és elosztott tárolórendszerek

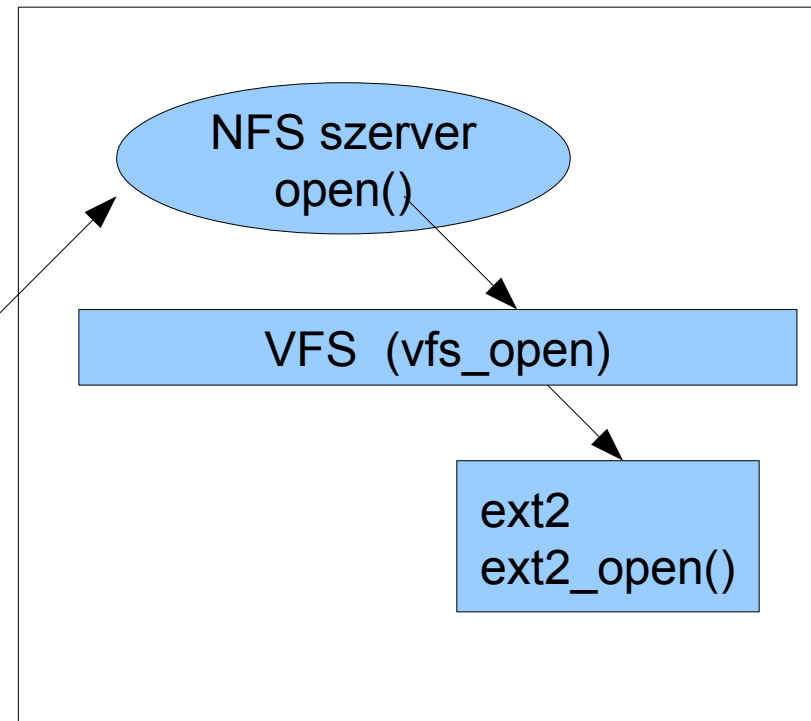
- Kliens-szerver modell
  - szerver: helyi tároló → hálózat → kliens
  - fájl tároló: **NAS (Network Attached Storage)**
    - NFS** (Network File System)
    - SMB/CIFS** (Common Internet File System)
  - blokk tároló: **SAN (Storage Area Network)**
    - iSCSI** (internet SCSI): SCSI parancsok IP-alon
    - FCP** (Fibre Channel Protocol) és **FCoE**: SCSI átvitel FC / ethernet-alapon
- Elosztott fájlrendszer (distributed file system)
  - elosztott rendszerként működő tárolórendszer
    - Ceph** (Inktank, Red Hat, SUSE), Google **GFS**, RedHat **GlusterFS**, Windows **DFS**, PVFS (parallel virtual FS) → **OrangeFS**
- Kérdések, problémák
  - késleltetés
  - hálózati hibák
  - konzisztencia

# Az NFS megvalósításának egyszerűsített felépítése

gép 1



gép 2



RPC

# Hálózati fájlrendszerek elméleti kérdései (ízeltő)

- Hogy/hol érjük el az adatainkat?
  - **elhelyezkedés-átlátszóság** (location transparency)
    - az adatok címezése (fájlok megnevezése) nem utal az elhelyezkedésükre
  - **elhelyezkedés-függetlenség** (location independence)
    - a címezés, elnevezés nem változik az adatok áthelyezésével
- Ki teljesíti a kéréseket?
  - távoli szolgáltatás (központi)
    - ahol az adat → egyszerű
    - kommunikációs gondok
      - késleltetés, csomagvesztés, sorrend változása
  - (részben) helyi átmeneti tárral
    - helyi másolaton → nehezebb
    - tárolható helyileg?
    - írás, konzisztencia (több másolat)?
- Hogyan működik a hálózati kiszolgáló?
  - **állapotot tároló** (stateful): a fájlműveletek előélettel rendelkeznek, gyorsabb
  - **állapotmentes** (stateless): örökifjú, megbízhatóbb

# Elosztott, skálázható tárolórendszerek: Ceph

- Virtualizált tárolórendszer (szoftveres) többféle eléréssel
  - blokkos tárolóeszköz (SAN)
  - fájl tárolás (NAS)
  - objektumtár (OSD, az adatok objektum-alapú tárolására, az alap)
- Skálázható és 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)
- További előnyei
  - PB (petabájt, 1000 TB,  $10^{15}$  bájt, 76 évnyi 720p H.264 videó) kapacitás
  - sokkal gyorsabban felépül a hardver hibákból, mint a RAID tömbök
  - nem igényel speciális hardvereket (lásd RAID kártya és diszk)
  - nem szükséges tartalék hardverek beépítése (lásd RAID spare disk)
  - együttműködik a virtualizációs rendszerekkel ([OpenStack](#), [Amazon S3](#))
  - nyílt forráskódú (a technológia mögötti céget [megvette](#) a [RedHat](#))



# A Ceph alapja a **RADOS** tárolási rendszer

- RADOS: Reliable, Autonomic Distributed Object Store
- OSD (object storage device): adattárolási csomópont
  - CPU + memória + lokális diszk (jellemzően diszkenként egy OSD)
- Tárolóegység (placement group, PG): az objektumok tárolóhelye
  - az OSD-kre épülő elosztott (logikai) tárolási hely (pár OSD sok PG)
  - meghatározza a replikáció mértékét
- Klasztertérkép: a tárolási rendszer leírása (minden csomópontban)
  - az OSD-k és PG-k listája (milyen építőelemekből épül fel a klaszter)
  - a tárolt adatok elhelyezkedése (mit és hol tárol a rendszer)
- Monitor: felügyelő, menedzsment komponens
  - kezeli és szükség szerint módosítja a klasztertérkép mesterpéldányát
  - jellemzően annyi példány van belőle, ahány fizikai gép alkotja a klasztert
- Az objektumok (adatok) elhelyezése (CRUSH algoritmus)
  - kvázi véletlenszerűen egyensúlyozva a tárolóegységek (PG) között
  - a tárolóegységen belül az ún. OSD térkép segítségével (replikáció)
  - automatikus áttelepítés kiesett vagy újonnan belépő eszközök esetén

# Merre tovább, fájl- és tárolórendszerek?

- Integrált fájl- és tárolórendszerek
  - a fájlrendszerek, LVM és RAID megoldások integrált megvalósítása
  - pl. zfs, btrfs
- Skálázhatóság
  - a tárolókapacitás dinamikus növekedése, különösen virtualizált rendszerek alatt
- Megbízhatóság
  - nagy tárolókapacitás, sok diszk → sok hiba jelentkezik
  - csökkenteni kell a hibajavítás (diszkcseré, adatmozgatás) okozta kiesést (nullára)
- Memóriaalapú tárolók
  - lásd bevezető előadás: a háttértárak és a fizikai memória sebessége közelít
- [Data deduplication](#) (pl. zfs, btrfs)
- További ajánlott olvasmányok érdeklődőknek:
  - Microsoft [ReFS](#) (Resilient File System), [ezen az oldalon](#) is
  - Solaris [ZFS](#) (Z File System, eredetileg Zettabyte...), [FreeBSD-n](#) és [Linuxon](#) is
  - Linux [Btrfs](#) (B-Tree File System, „butter F S”)
  - [F2FS](#) (Flash-Friendly File System, Samsung)
  - [GPUfs](#) (fájlrendszerek elérése GPU-n, lásd heterogén rendszerek)