

Operációs rendszerek

ELTE IK.

Dr. Illés Zoltán

zoltan.illes@elte.hu

Mi történt a múlt héten...

- ▶ Operációs rendszerek kialakulása
 - Sz.gép – Op.rendszer generációk
- ▶ Op. Rendszer fogalma
- ▶ Fogalmak:
 - Fájlok, könyvtárak, processzek
- ▶ Rendszerhívások
- ▶ Rendszer struktúrák
 - Ma: Vegyes, tipikus kliens–szerver modell, rétegelt jellemzőkkel

Mi következik ma...

- ▶ Háttértárak
- ▶ Fájlok
 - Fájl típusok
- ▶ Könyvtárak
 - Könyvtárszerkezetek
- ▶ Fájlrendszerek
- ▶ Fájlrendszer kérés ütemezések
- ▶ Biztonsági rendszerek
- ▶ ...

Háttértár típusok ma

- ▶ **Mágneses elvű**
 - Mágnesszalagok
 - Mágneslemezek
 - Merevlemez
 - Floppy
- ▶ **Optikai elvű**
 - CD, DVD, Blu-Ray, lézer elv, kb. 5xDVD a kapacitás
- ▶ **Félvezető**
 - USB, memóriakártya
 - SSD(Solid State Disk) diszk

Háttértár típusok holnap

- ▶ Holografikus
 - GE 2011 bejelentés, 500GB, hologramok a bitek
- ▶ Biológiai
- ▶ Nano felépítésű
- ▶
- ▶ Moore törvény, ..."1–2 évenként duplázódik az integrált áramkörök összetettsége ..", nem kifejezetten a lemezekre vonatkozik, de...

Mágnesszalagok fizikai felépítése

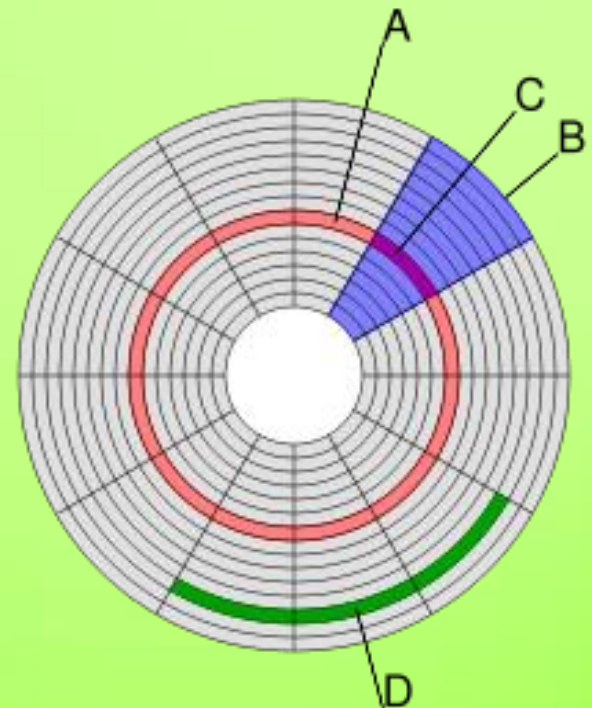
- ▶ Mágnesszalagok– sorrendi, lineáris felépítés
 - 9 bites keret (8 bit + paritás)
 - Keretek rekordokba szerveződnek
 - Rekordok között: rekord elválasztó (record gap)
 - Egymás utáni rekordok után, fájl elválasztó (file gap)
 - Szalag elején a könyvtárszerkezet
- ▶ Jellemző használat
 - Biztonsági mentés
 - Nagy mennyiségű adattárolásra
- ▶ Nem igazán olcsó
- ▶ Jellemző méret: DLT (Digital Linear Tape), LTO (Linear Tape–Open) 4 Ultrium 800/1600 GB

Mágnestlemezek felépítése I.

- ▶ FDD – Floppy Disk Drive
 - Jellemzően egy lemez
- ▶ HDD – Hard Disk Drive
 - Jellemzően több lemez
- ▶ Kör alakú lemez – sávok felosztás
- ▶ Sávok szektorokra oszthatók – blokk
 - Klaszter – több blokk
- ▶ Több lemez – egymás alatti sávok : cylinder
- ▶ Logikailag egy folytonos blokkosorozat
- ▶ A fizikai működést a meghajtó (firmware) eltakarja.

Mágnestlemez felépítése II.

- ▶ A: sáv
- ▶ B: szektor
- ▶ C: blokk, 512 byte
- ▶ D: klaszter, a fájlrendszer által megválasztott logikai tárolási egység. $D = n \times C$, ahol $n = 1 \dots 128$.
- ▶ Cylinder: Az egymás alatti sávok (pirossal)



Mágnestape felépítése példa

- ▶ CHS címzés (Cylinder– Head– Sector)
 - Példa: 1.44 MB FD
 - Cylinder szám: 80 (0–79)
 - Fejek száma: 2 (0–1)
 - Szektorok száma egy sávon: 18 (1–18)
 - Össz. Méret: $80 \cdot 2 \cdot 18 = 2880$ szektor * 512byte
- ▶ LBA címzés (Logical Block Addressing)
 - Korábban 28 bites, kb 137GB-ig jó.
 - Jelenleg 48 bites, 144 PB (Petabájt), (144 000 000 GB)

$$A = (c \cdot N_{\text{heads}} \cdot N_{\text{sectors}}) + (h \cdot N_{\text{sectors}}) + s - 1$$

Optikai tárolók

- ▶ Tipikusan 8 vagy 12 cm átmérőjű optikai lemezek
 - CD – Compact Disc, DVD –Digital Versatile Disc
 - Méret: 650MB – 17 GB között
 - Sebesség: $1x = 150 \text{ KB/sec}$
- ▶ Működési elv: Fény visszaverődés idő különbség alapján.
 - Belső résztől spirális „hegyek – völgyek” (pit-land) sorozata
 - Írható lemezek: Írás a lemezfelület mágnesességét, fény törésmutatóját változtatja meg, így más lesz a fény terjedési sebessége.

Eszközmeghajtó–Device driver

- ▶ Az a program, amely a közvetlen kommunikációt végzi.
- ▶ A kernel, az operációs rendszer magjának része.
- ▶ A lemezek írása–olvasása során jellemzően DMA-t használnak (nagy adatmennyiség).
 - Megszakítás üzenet, tipikusan azt jelzi ha befejeződött az írás–olvasás művelet.
 - I/O portokon az írás, olvasási paraméterek beállítását végzik.
- ▶ Réteges felépítés

Mágnestape formázása

- ▶ Sáv-szektoros rendszer kialakítása
- ▶ Jellemzően egy szektor 512 byte
- ▶ Gyárilag a kápetek „elő vannak készítve”
- ▶ Quick format– Normal format
 - A normál hibás szektorokat (bad sector) keres
- ▶ Szektor= Szektorfej+adatblokk+lábléc
 - Szektorfej: sáv száma, fej száma, szektor száma
 - Lábléc: hibajavító blokk
- ▶ A szektorok kialakítását alacsony szintű formázásnak nevezzük.

Logikai formázás

- ▶ Partíciók kialakítása
 - Egy lemezen PC-s rendszeren maximum 4 logikai lemezrész kialakítható.
- ▶ 0. szektor– MBR (Master Boot Record)
 - 2 részből áll, mérete: 512 bájt
 - Rendszerindító kód (bootloader, 446 bájt)
 - Max. 4 partíció adatai (4x 16 bájt=64 bájt)
 - 2 bájt, mindig: 0x55 0xAA
 - Elsődleges partíció– erről tölthető be operációs rendszer
 - Kiterjesztett partíció– több logikai meghajtó lehet
 - Swap partíció
- ▶ A partíción a szükséges adatszerkezet (fájlrendszer) kialakítása

Az MBR szerkezete

MBR szerkezet					
Cím			Leírás		Méret (bájt)
Hex	Oct	Dec			
0000	0000	0	Betöltő programkód		440 (max. 446)
01B8	0670	440	Opcionális Disk kód		4
01BC	0674	444	Tipikusan: 0x0000		2
01BE	0676	446	Elsődleges partíciós tábla adatok (4 db 16-bájtos rész, IBM Partíció Tábla séma)		64
01FE	0776	510	55h	MBR zárás: 0xAA55	2
01FF	0777	511	AAh		
MBR, teljes méret: 446 + 64 + 2 =					512

Partíciós tábla bejegyzés

- ▶ 1. bájt: Partíció státusa (80=aktív, 0=nem boot)
- ▶ 2–3–4. bájt : Partíció kezdőblokk CHS címe
 - 0–5. bit: fej száma
 - 6–15. bit: cylinder száma
 - 16–23. bit: szektor száma
- ▶ 5. bájt: Partíció típusa
- ▶ 6–7–8. bájt : Partíció befejező szektor CHS címe
- ▶ 9–10–11–12. bájt: Partíció kezdőszektor LBA címe
- ▶ 13–14–15–16. bájt: Szektorok száma
 - 4 bájt: $4 \text{ GB} * 512 = 2 \text{ TB}$

Boot folyamat

- ▶ ROM–BIOS megvizsgálja, lehet–e operációs rendszert betölteni, ha igen betölti a lemez MBR programját a 7c00h címre.
- ▶ Egy elsődleges partíció lehet aktív, az MBR programja megvizsgálja melyik az.
- ▶ Az aktív partíció boot szektorát (1. szektor) betölti a memóriába.
- ▶ Ez már a partícióra installált operációs rendszer betöltő programja Pl. LILO
- ▶ A boot program tudja, hogy a partíció melyik fájljait kell a memóriába tölteni, majd elindít egy „rendszerstartot”
 - Többszintű folyamat, rendszerfüggő.

Címszámítás

- ▶ Blokkok sorszámainak meghatározása
 - Kell a fejek száma, szektorok száma
 - Tegyük fel adott 4 fej (2 vagy 4 lemez)
 - Egy sáv legyen felosztva 7 szektorra
- ▶ Lemezek forgási sebessége miatt a blokkok nem feltétlenül szomszédosak (interleave)
 - 1:2 interleave, párosával „szomszédosak”

	1 szektor	2 szektor	3 szektor	4 szektor	5 szektor	6 szektor	7 szektor
1 fej.	1	17	5	21	9	25	13
2 fej.	2	18	6	22	10	26	14
3 fej.	3	19	7	23	11	27	15
4 fej.	4	20	8	24	12	28	16

Lemez elérés fizikai jellemzői

- ▶ Forgási sebesség (ma tipikusan 5400, 7200, 10000 vagy 15000 percenként)
 - Egy sávon (cilindereken) belül mekkorát kell fordulni
- ▶ Fej mozgási sebesség
 - Egy cilindereken belül nem kell mozgatni a fejet.
- ▶ Az írás–olvasás ütemezés feladata a megfelelő (gyors, hatékony) kiszolgálási sorrend megválasztása
 - Hozzáférési idő csökkentése
 - Átviteli sávszélesség növelése

Írás–Olvasás műveletek

- ▶ Alacsonyszintű hívás során az alábbi adatok szükségesek:
 - Beolvasandó (kiírandó) blokk(ok) sorszáma
 - Memóriaterület címe, ahova be kell olvasni.
 - Bájtok száma
- ▶ Több folyamat használja
 - Melyiket hajtsuk végre először?

Írás–Olvasási műveletek ütemezése

- ▶ Alacsonyszintű (kernel) feladat paraméterek
 - Kérés típusa (írás–olvasás)
 - A blokk kezdőcíme, (sáv, szektor, fej száma)
 - DMA memóriacím
 - Mozgatandó bájtok száma
- ▶ Több folyamat is használná a lemezt
 - Kit szolgáljunk ki először.
 - Fejmozgás figyelembevétele (olvasandó blokk adataiból következik)

Sorrendi ütemezés (FCFS)

- ▶ First Come – First Service
- ▶ Legegyszerűbb „stratégia”, ahogy jönnek a kérések, úgy sorban kiszolgáljuk azokat.
- ▶ Biztosan minden kérés kiszolgálásra kerül.
 - Nincs kiéheztetés.
- ▶ Nem törődik a fej aktuális helyzetével.
- ▶ Nem igazán hatékony.
- ▶ Kicsi az adatátviteli sávszélesség.
- ▶ Átlagos kiszolgálási idő, kis szórással.

SSTF ütemezés

- ▶ Shortest Seek Time First – SSTF, leghamarabb elérhetőt először
- ▶ A legkisebb fejmozgást részesíti előnyben.
- ▶ Átlagos várakozási idő kicsi.
 - A várakozási idő szórása nagy
- ▶ Átviteli sáv szélesség nagy
- ▶ Fennáll a kiéheztetés veszélye

Pásztázó ütemezés

- ▶ SCAN (LOOK) módszer
- ▶ A fej állandó mozgásban van, és a mozgás útjába eső kéréseket kielégíti.
- ▶ A fej mozgás megfordul ha a mozgás irányában nincs kérés, vagy a fej szélső pozíciót ért el.
- ▶ Rossz ütemben érkező kérések kiszolgálása csak oda-vissza mozgás(írás-olvasás) után kerül kiszolgálásra.
 - Várakozási idő közepes, Szórás nagy
- ▶ Középső sávok elérés szórása kicsi

Egyirányú pásztázás

- ▶ Circural SCAN, C-SCAN
- ▶ A SCAN javítása, írás-olvasás, csak a fej egyik irányú mozgásakor történik.
- ▶ Gyorsabb fejmozgás
- ▶ Nagyobb sáv szélesség
- ▶ Az átlagos várakozási idő hasonló mint a SCAN esetén, viszont a szórás kicsi.
 - Nem fordulhat elő igazán rossz ütemű kérés

Ütemezés javítások

- ▶ FCFS módszernél, ha az aktuális sorrendi kérés kiszolgálás helyén van egy másik kérés blokkja (mozgás nélkül elérhető), akkor szolgáljuk ki azt is. (Pick up)
- ▶ Egy folyamat adatai jellemzően egymás után vannak, így egy kérés kiszolgálásnál „picit” várva, a folyamat az adatainak további részét is kéri a folyamat.
 - Előlegező ütemezésnek is nevezzük
- ▶ A lemez közepe általában hatékonyan elérhető.

Ütemezés javítása memória használattal

- ▶ A DMA maga is memória
- ▶ Memória puffer (átmeneti tár) használat
 - Kettős körszerű használat
 - Olvasás: Ütemező tölti, felhasználói folyamat üríti
 - Írás: Felhasználó folyamat tölti, ütemező üríti
- ▶ Disc cache– Lemez gyorsítótár
 - Előre dolgozik az ütemező, a memóriába tölti a kért adatok „környéki” lemezterületet is.
 - Operációs rendszernek jelent plusz feladatot
 - PL: Smartdrive

Milyen ütemezést válasszunk?

- ▶ A fenti algoritmusok csak a fejmozgás idejét vették figyelembe, az elfordulást nem.
- ▶ A sorrendi ütemezést tipikusan egy felhasználós rendszerben használt.
- ▶ SSTF, kiéheztetés veszélye nagy
- ▶ C-Scan , nagy IO átvitel, nincs kiéheztetés
- ▶ Beépített ütemező: PL. SCSI vezérlők
 - OS ömlesztve adja a kéréseket.

Ütemezés kulcsfeladata

- ▶ Gyorsan (minél gyorsabban) kiszolgálni a kéréseket.
- ▶ Ezt mi is (OS is) elősegíthetjük.
 - Összetartozó adatok együtt legyenek (töredezettség)
 - Sáv szélesség a lemez közepén a legnagyobb.
 - Leggyorsabban a lemez közepét érjük el (virtuális memória)
 - Lemez gyorsító tár a memóriában.
 - Esetleg adattömörítés (nagyobb CPU terhelés)

Lemezek megbízhatósága

- ▶ Jelentése: Az adatok redundáns tárolása, hogy lemezsérülés esetén se legyen adatvesztés
- ▶ Operációs rendszer szolgáltatás
 - Dinamikus kötet– több lemezre helyez egy logikai meghajtót. Méret összeadódik.
 - Tükrözés– két lemezre helyez egy meghajtót. Mérete az egyik (kisebb) lemez mérete lesz.
 - Nagy(obb) CPU igény.
- ▶ Hardware szolgáltatás
 - Intelligens meghajtó szolgáltatás
 - Az SCSI eszköz világban jelent meg először (RAID)

Megbízható lemezmeghajtók

- ▶ RAID – Redundant Array of Inexpensive Disks
- ▶ SCSI lemezegységeknél jelent meg először
 - Nem scsí...☺
 - Small Computer System Interface
 - Számítógépek és perifériák közti adatcsere egy ma is népszerű szabvány együttese.
 - Leggyakrabban lemezek körében használt, szerver gépek használják (ták)
 - Ennek egy újabb változata: SAS csatoló (Serial Attached SCSI)

RAID

- ▶ Ha operációs rendszer nyújtja, gyakran SoftRaid-nek nevezik.
- ▶ Ha intelligens (külső) vezérlőegység nyújtja, gyakran Hardver Raid-nek, vagy csak Raid diszkrendszernek nevezik.
- ▶ Bár nevében olcsó (Inexpensive), valójában inkább nem az.
- ▶ Több lemezt fog össze, és egy logikai egységként látja az operációs rendszer.
- ▶ Többféle „összefogási” elv létezik: RAID 0–6

RAID 0(striping)

- ▶ Ez az a Raid, ami nem is redundáns...
- ▶ Több lemez logikai összefűzésével egy meghajtót kapunk.
- ▶ A lemezkapacitások összege adja az új meghajtó kapacitását.
- ▶ A logikai meghajtó blokkjait szétrakja a lemezekre (striping), ezáltal egy fájl írása több lemezre kerül.
- ▶ Gyorsabb I/O műveletek.
- ▶ Nincs meghibásodás elleni védelem.

RAID 1 (tükrözés)

- ▶ Két független lemezből készít egy logikai egységet.
- ▶ Minden adatot párhuzamosan kiír mindkét lemezre.(Tükrözés,mirror)
- ▶ Tárolókapacitás felére csökken.
- ▶ Drága megoldás.
- ▶ Jelentős hibatűrő képesség.
 - Mindkét lemez egyszerre történő meghibásodása okoz adatvesztést.

RAID 1+0, RAID 0+1

- ▶ RAID 1+0: Tükrös diszkekből vonjunk össze többet.
- ▶ RAID 0+1: Raid 0 összevont lemezcsoporthból vegyünk kettőt.
- ▶ A vezérlők gyakran nyújtják egyiket, másikat, mivel így is, úgy is tükrözés van, azaz drága, így ritkán használt.

RAID 2,3,4

- ▶ RAID 2: Adatbitek mellett hibajavító biteket is tartalmaz. (ECC–Error Correction Code)
Pl. 4 diszkhez 3 javító diszk
- ▶ RAID 3: Elég egy plusz „paritásdiszk”, $n+1$ diszk, Σn a kapacitás
- ▶ RAID 4: RAID0 kiegészítése paritásdiszkkal.
- ▶ Ma ezen megoldások nem gyakran használatosak.

RAID 5

- ▶ Nincs paritásdiszk, ez el van osztva a tömb összes elemére.(stripe set)
- ▶ Adatok is elosztva kerülnek tárolásra.
- ▶ Intenzív CPU igény (vezérlő CPU!!!)
- ▶ Redundáns tárolás, 1 lemez meghibásodása nem okoz adatvesztést.
 - 2 lemez egyidejű meghibásodása már igen
 - Hogy működik? (A paritásbitből meg a többiből az egy eltűnt kiszámítható!)
- ▶ N lemez RAID 5 tömbben($N \geq 3$), $n-1$ lemez méretű logikai meghajtót ad.

RAID 6

- ▶ A RAID 5 paritásblokkhoz, hibajavító kód kerül tárolásra.(+1 diszk)
- ▶ Még intenzívebb CPU igény.
- ▶ Két diszk egyidejű kiesése sem okoz adatvesztést!
- ▶ Relatív drága
- ▶ N diszk RAID 6-os tömbjének kapacitása, N-2 diszk kapacitással azonos.
- ▶ Elvileg általánosítható a módszer (3 diszk kiesése...)

RAID összegzés

- ▶ Ma leggyakrabban a RAID 1,5 verziókat használják.
- ▶ A RAID 6 vezérlők az utóbbi 1–2 évben jelentek meg.
 - Bár olcsó diszkekről szól a RAID, de valójában ezek nem mindig olcsók!
 - Itt már 2 lemez kiesik, így ez még inkább drága.
- ▶ Hot-Swap(forró csere) RAID vezérlő: működés közben a meghibásodott lemezt egyszerűen kicseréljük.

Adattárolás összefoglalása

- ▶ Adatok biztonságos tárolását biztosítja.
- ▶ Több szintű:
 1. Fizikai lemezek (HDD)
 2. Hardver RAID
 3. Partíciók
 4. Szoftver RAID
 5. Volume Manager az operációs rendszerben.
- ▶ Nem minden ellen véd
 - PL: Tápellátás elhal, emberi tévedés, stb.
 - Szoftveres támadások, vírusok.
- ▶ Hogy szerveződnek adataink a „volume”-on?

Fájlrendszer

- ▶ Fájl: adatok egy logikai csoportja, névvel egyéb paraméterekkel ellátva.
- ▶ Könyvtár: fájlok (könyvtárak) logikai csoportosítása.
- ▶ Fájlrendszer: módszer, a fizikai lemezünkön, kötetünkön a fájlok és könyvtárak elhelyezés rendszerének kialakítására.

Fájlok

- ▶ A fájl az információtárolás egysége.
- ▶ Névvel hivatkozunk rá.
- ▶ Jellemzően egy lemezen helyezkedik el.
 - De általánosan az adathalmaz, adatfolyam akár képernyőhöz, billentyűzethez is köthető.
- ▶ A lemezen általában 3 féle fájl, állomány található:
 - Rendes felhasználói állomány.
 - Ideiglenes állomány
 - Adminisztratív állomány. Ez a működéshez szükséges, általában rejtett.

Fájl jellemzők

- ▶ Fájlnev: Karaktersorozat
 - Operációs rendszer függvénye, hogy milyen a szerkezete(hossza, megengedett karakterek, kis-nagybetű különbözőség)
- ▶ Egyéb attribútumok (információ)
 - Mérete, tulajdonosa, utolsó módosítás ideje, rejtett (hidden) fájl-e, rendszer fájl-e, hozzáférési jogosítványok, tulajdonos,...
- ▶ Fizikai elhelyezkedés
 - Valódi fájl, link (hard), link (soft)

Könyvtárak

- ▶ Valójában egy speciális bejegyzésű állomány, tartalma a fájlok nevét tartalmazó rekordok listája.
- ▶ Könyvtár szerkezetek
 - Katalógus nélküli rendszer, szalagos egység
 - Egyszintű, kétszintű katalógus rendszer
 - Nem igazán használt
 - Többszintű, hierarchikus katalógus rendszer
 - Fa struktúra
 - Hatékony keresés
 - Ma ez a tipikusan használt.
- ▶ Abszolút, relatív hivatkozás
 - PATH környezeti változó

Hozzáférési jogok

- ▶ Nincs általános jogosítvány rendszer
- ▶ Jellemző jogosítványok:
 - Olvasás
 - Írás, létrehozás, törlés
 - Végrehajtás
 - Módosítás
 - Full control
- ▶ Jogok nyilvántartása
 - Attribútumként
 - ACL

Fájlok elhelyezése

- ▶ A partíció elején, az un. Szuperblokk (pl. FAT esetén a 0. blokk) leírja a rendszer jellemzőit.
- ▶ Általában következik a helynyilvántartás (FAT, láncolt listás nyilvántartás)
- ▶ Ezután a könyvtárszerkezet (inode), a könyvtár bejegyzésekkel, fájl adatokkal. (FAT16-nál a könyvtár előbb van, majd utána a fájl adatok.)
- ▶ Hova kerüljön az új fájl?
- ▶ Milyen módszert válasszunk?

Elhelyezési stratégiák

- ▶ Folytonos tárkiosztás
 - First Fit
 - Best Fit
 - Worst Fit (olyan memória szakaszba tesszük, hogy a lehető legnagyobb rész maradjon szabadon)
 - Veszteséges lemezkihasználás.
- ▶ Láncolt elhelyezkedés
 - Nincs veszteség (csak a blokkméretből adódóan)
 - Fájl adatok (blokkokra bontva) láncolt lista tábla
 - Az n. blokk olvasása lassú lesz.
 - Szabad– foglalt szektorok: File Allocation Table, FAT
 - Ez nagy lehet, a FAT-nak a memóriában kell lenni fájl műveletnél.
- ▶ Indextáblás elhelyezés
 - Katalógus tartalmazza a fájlhoz tartozó kis tábla (inode) címét
 - Egy inode címből elérhető a fájl.

Fájl, könyvtár műveletek

- ▶ Fájl
 - Megnyitás
 - Műveletek: Írás, olvasás, hozzáfűzés
 - Lezárás
- ▶ Adatok
 - Bináris– bájt sorozat
 - Szöveges– karakter sorozat
- ▶ Elérés módja, szekvenciális, random
- ▶ Könyvtár műveletek
 - Létrehozás, tartalom listázása, állomány törlés

Fájlrendszer típusok

- ▶ Merevlemezen alkalmazott fájlrendszer
 - FAT, NTFS, EXT2FS, XFS, stb
- ▶ Szalagos rendszereken (elsősorban backup) alkalmazott fájlrendszer
 - Tartalomjegyzék, majd a tartalom szekvenciálisan
- ▶ CD, DVD, Magneto-opto Disc fájlrendszere
 - CDFS, UDF (Universal Disc Format), kompatibilitás
- ▶ RAM lemezek (ma már kevésbé használtak)
- ▶ FLASH memória meghajtó (FAT32)
- ▶ Hálózati meghajtó
 - NFS
- ▶ Egyéb pszeudó fájlrendszerek
 - Zip, tar.gz, ISO

Naplózott fájlrendszerek

- ▶ Fájlrendszer sérülés, áramszünet stb. esetén inkonzisztens állapotba kerülhet.
- ▶ Gyakran nevezik: LFS-nek (Log-structured File System) vagy JFS-nek (Journaled)
- ▶ Adatbázis kezelők mintájára: művelet + log
 - Tranzakciós alap
 - Leállítás, hiba esetén a log alapján helyre lehet állítani.
 - Célszerűen a log másik lemez (másik partíció)
- ▶ Nagyobb erőforrás igény – nagyobb megbízhatóság

Fájrendszer támogatás

- ▶ Mai operációs rendszerek „rengeteg” típust támogatnak
 - PL: Linux 2.6 kernel több mint 50-et.
- ▶ Fájrendszer csatolása
 - Mount, eredményeképpen a fájlrendszer állományok elérhetők lesznek.
 - Automatikus csatolás (pl. USB drive)
 - Kézi csatolás (Linux, mount parancs)
- ▶ Külön névtérben való elérhetőség (Windows)
 - A,B,C,...
- ▶ Egységes névtér (UNIX)

Alkalmazás– Diszk kapcsolat

► Réteges felépítés

- Alkalmazói szint
 - Az alkalmazás, fejlesztői könyvtárak segítségével megoldja a lemezen tárolt adatok írását–olvasását.
 - Szöveges, bináris fájlműveletek
- Operációs rendszer szint
 - Fájlrendszer megvalósítás
 - Elérhetőség, jogosultságok
 - Kötetkezelő (Volume manager)
 - Eszközmeghajtó (device driver)
 - BIOS–ra alapozva
- Hardver eszköz szintje
 - I/O meghajtó, IDE, SATA stb.

FAT

- ▶ File Allocation Table
 - Talán a legrégebbi, ma is élő fájlrendszer!
- ▶ A FAT tábla a lemez foglaltsági térképe, annyi eleme van, ahány blokk a lemezen
 - Pl: Fat12, FDD, Cluster méret 12 bites. Ha értéke 0, szabad, ha nem foglalt.
 - Biztonság kedvéért 2 tábla van.
- ▶ Láncolt elhelyezés
 - A katalógusban a file adatok (név stb) mellett csak az első fájl blokk sorszáma van megadva.
 - A FAT blokk azonosító mutatja a következő blokk címét.
 - Ha nincs tovább, FFF az érték.
- ▶ Rögzített bejegyzés méret, 32 bájt (max. 8.3 név)
- ▶ System,Hidden,Archive,Read only, könyvtár attribútumok
- ▶ A fájl utolsó módosítás ideje is tárolva van.

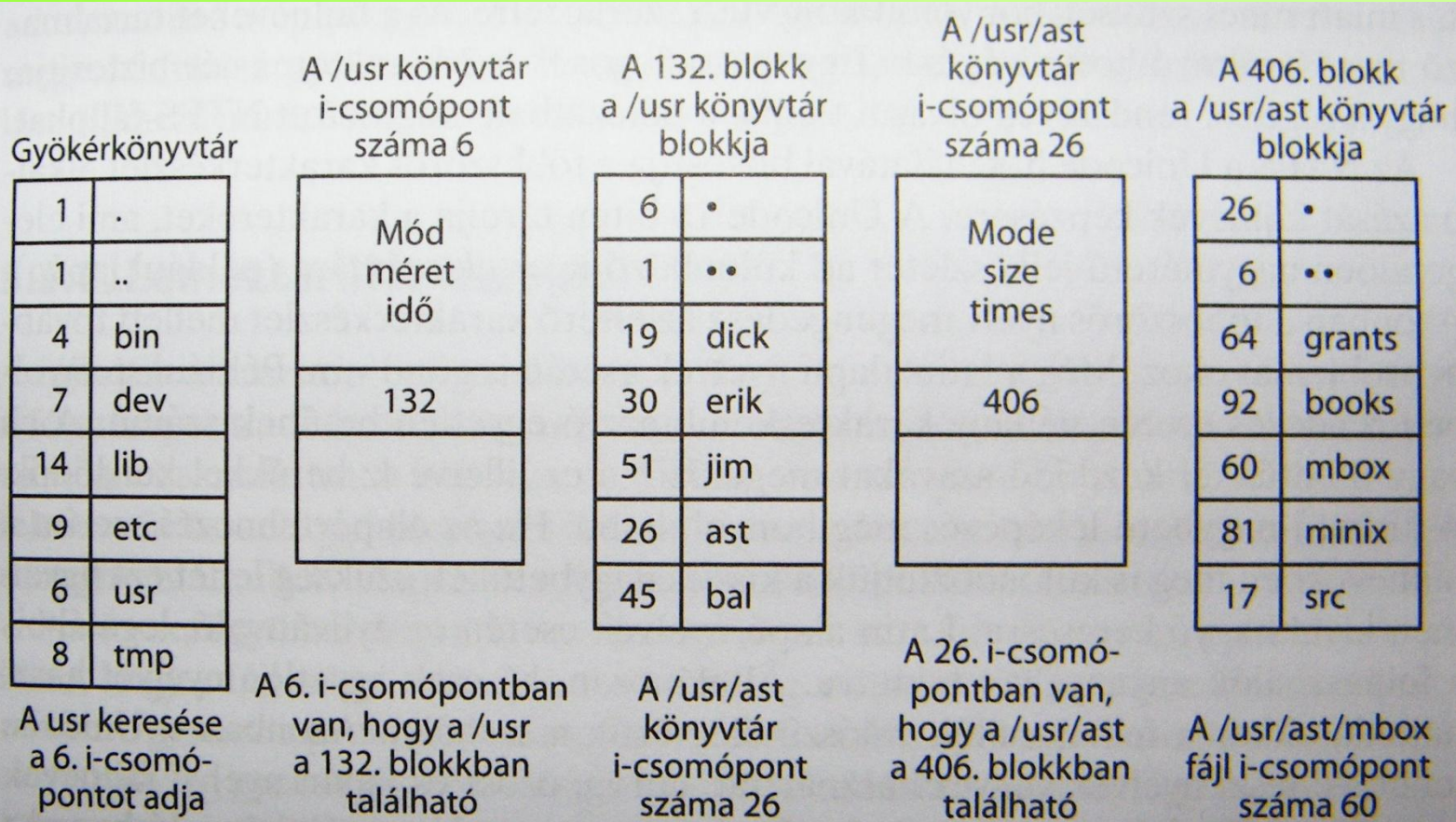
FAT jellemzők

- ▶ FAT16, 16 bites cluster leíró, 4 bájt (2x2) írja le a fájl kezdőblokkját
 - Max. 4 GB partíciós méret (64kb blokk méretnél), jellemzően 2 GB.
 - Fájl méret maximum is a 4 (2) GB.
 - Külön könyvtári terület (FDD-n ez a 0. sáv)
 - FDD-n 512 könyvtári bejegyzés
 - HDD-n 32736 könyvtári bejegyzés (16 bit előjelesen)
- ▶ FAT32 (1996-tól elérhető)
 - 28 bites cluster leíró
 - 2 TB partíciós méret (alap szektor mérettel)
- ▶ 32MB-ig, 1 blokk = 1 szektor(512bájt)
 - 64 MB, 1 blokk=1KB (2 szektor), 128MB, 1 blokk=2KB
 - 1 blokk max. 64 KB lehet.
- ▶ Támogatták már a hosszú fájl neveket is
 - Többszörös 8.3 részre fenntartott bejegyzésekkel.
- ▶ Töredezettség mentesítés szükséges.

UNIX könyvtárszerkezet

- ▶ Indextáblás megoldás
- ▶ Boot blokk után a partíció superblokkja (fájlrendszer paraméterek)
- ▶ Ezt követi a szabad terület leíró rész.
- ▶ i-node tábla, majd gyökérkönyvtár bejegyzéssel)
- ▶ Moduláris elhelyezés, gyorsan elérhető az információ, sok kicsi táblázat, ez alkotja a katalógust.
- ▶ Egy fájl egy i-node ír le!
 - 15 rekeszből áll, első 12 a fájl blokkokra mutat.
 - Ha kevés, a 13. rekesz újabb i-node-ra, ami +15 rekesz.
 - Ha ez is kevés, a 14. rekesz újabb i-node-ra ami az első mintáját ismétli.

i-node láncolás példa



5.16. ábra. A /usr/ast/mbox keresésének lépései

NTFS

- ▶ New Technology File System
 - FAT–NTFS hatékonysági határ: kb. 400 MB.
- ▶ 255 karakteres fájl név, 8+3 másodlagos név
- ▶ Kifinomult biztonsági beállítások
- ▶ Ahogy a FAT esetén, itt is szükséges a töredezettség mentesítés.
- ▶ Titkosított fájlrendszer támogatása, naplózás
- ▶ POSIX támogatás
 - Hard link (fsutil parancs), időbélyegek, kis–nagybetűk különböznek
- ▶ Tömörített fájl, mappa, felhasználói kvóta kezelés
- ▶ Az NTFS csak klasztereket tart nyilván, szektort (512bájtt) nem

MFT

- ▶ LCN – Logical Cluster Number, egy köteten, a klaszterek sorszáma
- ▶ NTFS partíció az MFT (Master File Table) táblázattal kezdődik
 - 16 attribútum ad egy fájl bejegyzést.
 - Minden attribútum max. 1kb. Ha ez nem elég akkor egy attribútum mutat a folytatásra.
 - Az adat is egyfajta attribútum, így egy bejegyzés több adatsort tartalmazhat. (PL: Betekintő kép)
 - Egy adatsor 2^{64} bájt lehet
 - Ha a fájl < 1 kb, belefér az attribútumba, közvetlen fájl.
 - Nincs fájl méret maximum.

Az MFT felépítése

0	\$Mft – Master File Table
1	\$MftMirr – MFT Mirror
2	\$LogFile – Naplófájl
3	\$Volume – Kötetfájl
4	\$AttrDef – Attribútum definíciók
5	\ – Gyökérkönyvtár
6	\$BitMap – Cluster foglaltság
7	\$Boot – Bootszektor
8	\$BadClus – Hibás clusterek
9	\$Secure – Biztonsági leírók
10	\$UpCase – Unicode karaktertábla
11	\$Extend – Egyéb metadata
12	Nem használt
...	...
15	Nem használt
16	Felhasználói fájlok és mappák

Az NTFS metadata számára fenntartva

Köszönöm a figyelmet!

zoltan.illes@elte.hu