Operációs rendszerek

A UNIX file-rendszer (UFS)

A UNIX file-rendszer

Kettő nagy csoport:

- lokális és
- távoli állományrendszerek.

A kezdeti rendszerek csak a lokális kezelésére voltak képesek.

A UNIX file-rendszer

UNIX File System (UFS) tulajdonságai:

- On-line file-rendszer.
- Változatlan felhasználói felület.
- Nyolcvanas évek: elosztott file-rendszerek megjelenése:
  - Felület marad, belső működés változik (pl. Virtual FS, VFS).

A UNIX file-rendszer

- Alapegység: a file, amelyet byte-folyamként kezel.
- Soros (szekvenciális) elérés.
- Transzparens (átlátszó) file-szerkezet.
- Link-ek (kapcsolatok) létrehozásának lehetősége:
  - egy file-ra több névvel is hivatkozhatunk.
- Blokkméret:
  - 512, majd 1024 byte,
  - azaz nő az adatátvitel sávszélessége, de nő a belső tördelődés is!

UFS felhasználói felületének jellemzői

- A könyvtárak szerkezete hierarchikus fa struktúra (irányított körmentes gráf).
- A file-ok elérése:
  - /könyvtár/alkönyvtár/file-név
- Egyetlen gyökérkönyvtár ("/").
- File-rendszerek összekapcsolása (mount).
- Aktuális könyvtár fogalmának támogatása.

A UNIX file-rendszerben használt adatszerkezetek

- Az i-node szám (*index node*) a file-t azonosító adat:
  - 1 file-hoz 1 i-node szám,
  - célszerűen cache-elve a memóriában,
  - 2 byte = 65.536 db file.
- Könyvtár leírás: a "." nevű file.

i-node-szám (2 byte)	file név (14 byte)
6	.
5388	..
45	adat.dat
677	.titok.titkos

A lemezen tárolt i-node tartalma I.

- File típusa, ami lehet:
  - speciális (karakteres vagy blokkos eszköz, I/O),
  - adatállomány,
  - könyvtár (katalógus) bejegyzés (. és ..),
  - PIPE/FIFO.
- Az adott i-node-ra mutató link-ek száma (a szimbolikus link-ek nélkül).
- Eszköz ID.

A lemezen tárolt i-node tartalma II.

- UID, GID (a file tulajdonosának felhasználói és csoportazonosítója).
  A UNIX védelmi rendszere használja.
  A file létrehozásakor a file a létrehozó felhasználó **hatásos** user-ID-jét és group-ID-jét örökli.
- Időcímkék:
  - utolsó elérés,
  - utolsó módosítás,
  - utolsó attribútum módosítás (i-node módosítás).

A lemezen tárolt i-node tartalma III.

- A file hozzáférési jogai (rwx).
- Címtábla, (13 mutató a file-hoz tartozó adatblokkokra):
  - 10 direkt és
  - 1 indirekt, 1 kétszeresen indirekt és 1 háromszorosan indirekt.
- A file mérete (a mezőszélesség 4 byte = 4 GB!).

A memóriában tárolt (cache-elt) i-node

- Megnevezés: in-core i-node.
- Tartalmazza a lemezen tárolt i-node összes információját.
- Továbbá:
  - a státusát:
    - zárolt,
    - folyamat vár rá,
    - a tartalma eltér a lemezen tárolt változattól:
      - attribútum írás miatt,
      - adat írás miatt,
  - az i-node-ra van-e egy másik file-rendszer illesztve, azaz file egy mount pont-e:
    - ha igen, akkor kell a mount-tábla azonosító,

A memóriában tárolt (cache-elt) i-node

- a file-t tartalmazó file-rendszer logikai berendezés azonosítója (melyik eszközről lett beolvasva),
- az i-node sorszáma (a lemezen meghatározott helyük van az i-node listában, ott ez felesleges),
- mutatók más in-core i-node-okra,
- hivatkozásszámláló (a file nyitott példányainak a száma, 0 esetén szabad).

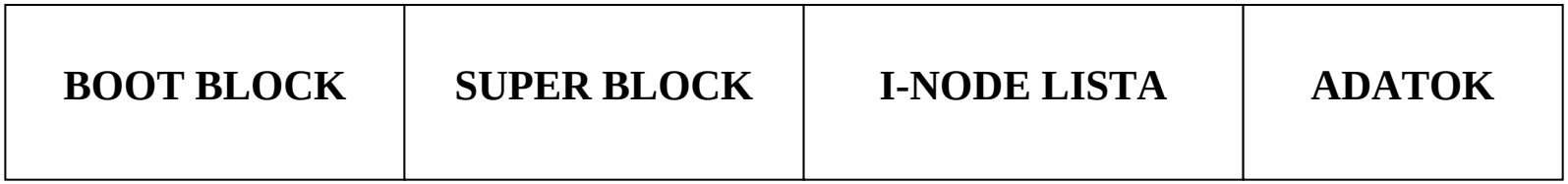
Tárolási korlátok

- A file-rendszer méretének korlátja:
  - a logikai diszk mérete nem lehet nagyobb a fizikai diszk méreténél.
- A file méretének korlátja:
  - a file mérete nem lehet nagyobb a logikai diszk méreténél.

Adatszerkezetek a lemezen

Adatszerkezetek és file-rendszer leírók a lemezen

File-rendszer a lemezen:



Adatszerkezetek és file-rendszer leírók a lemezen

- Boot block:
  - a rendszer indításához szükséges információkat tartalmazza,
  - bár csak 1 file-rendszernél szükséges, mindegyik tartalmazza, legfeljebb üres.
- Super block:
  - a file-rendszer metaadatait tartalmazza, leírva ezzel annak állapotát,
  - buffer cache szintű kezelés (állandó memóriában lévőség),
  - több file-rendszer esetén egy mount() rendszerhívással kerülnek a memóriába egészen egy umount() rendszerhívásig,
  - egy ún. mount-tábla tartja nyilván őket.

Adatszerkezetek és file-rendszer leírók a lemezen

- i-node list:
  - az i-node-ok nyilvántartásának a helye,
  - a file-rendszerben maximálisan létrehozható file-ok számát határozza meg (super user által).
- Data blocks:
  - az adatok fizikai tárolási helyei.

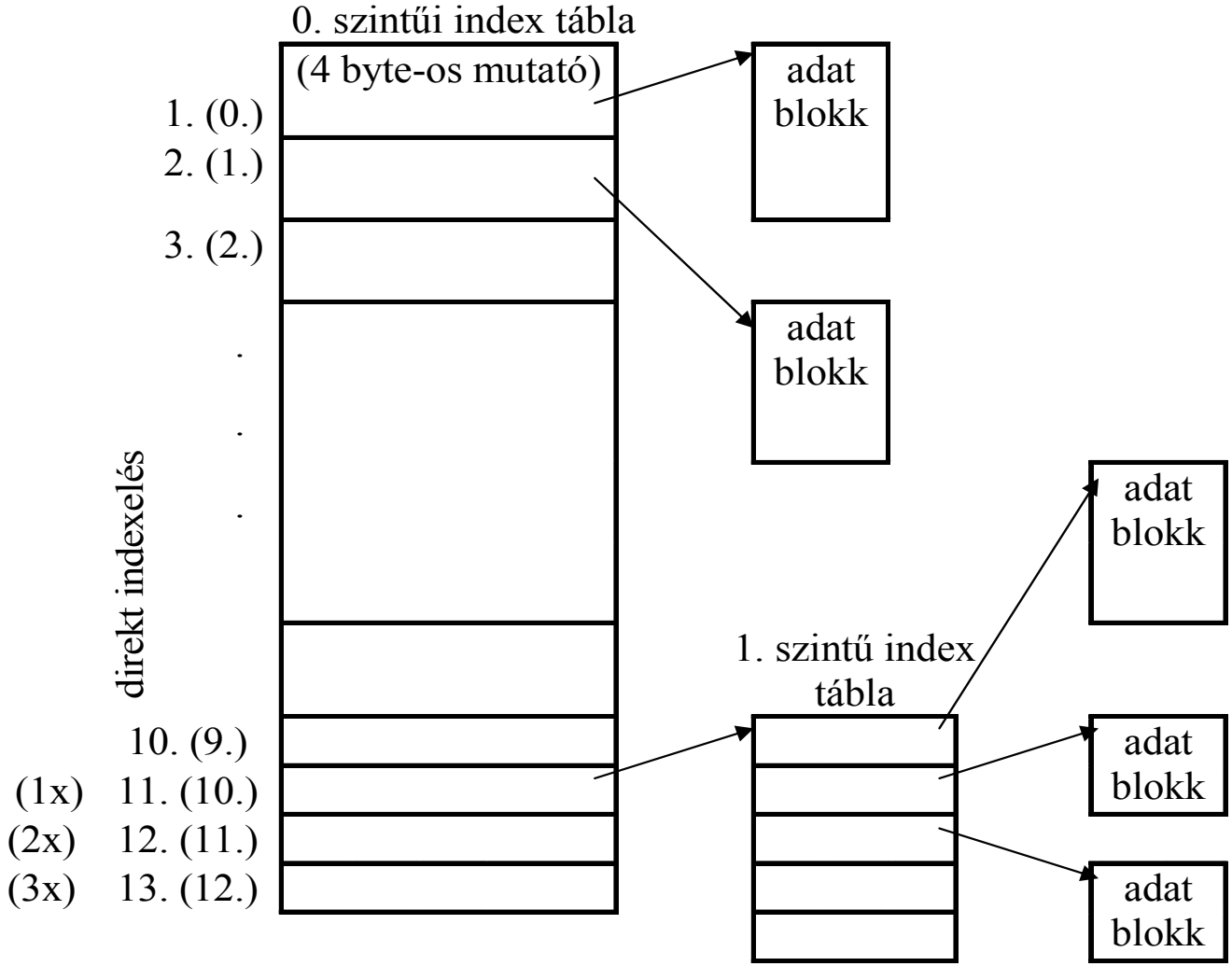
A file-hoz tartozó adatblokkok tárolása az i-node-ban

- Többszörös indexeléssel, 13 db mutatóval:
  - 1.-10. direkt index,
  - 11. egyszeres indirekt index,
  - 12. kétszeres indirekt index,
  - 13. háromszoros indirekt index.

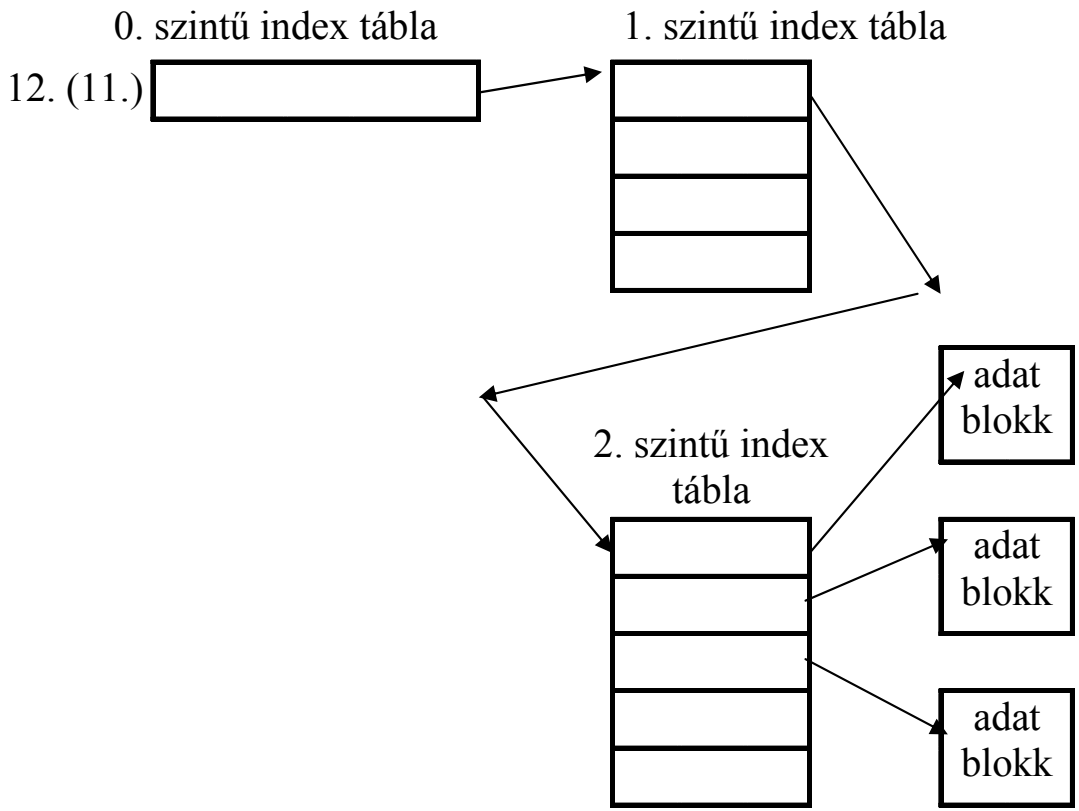
A file-hoz tartozó adatblokkok tárolása az i-node-ban

Példa: 1Kbyte-os lemez blokkok és 4 byte-os blokk címek esetén:

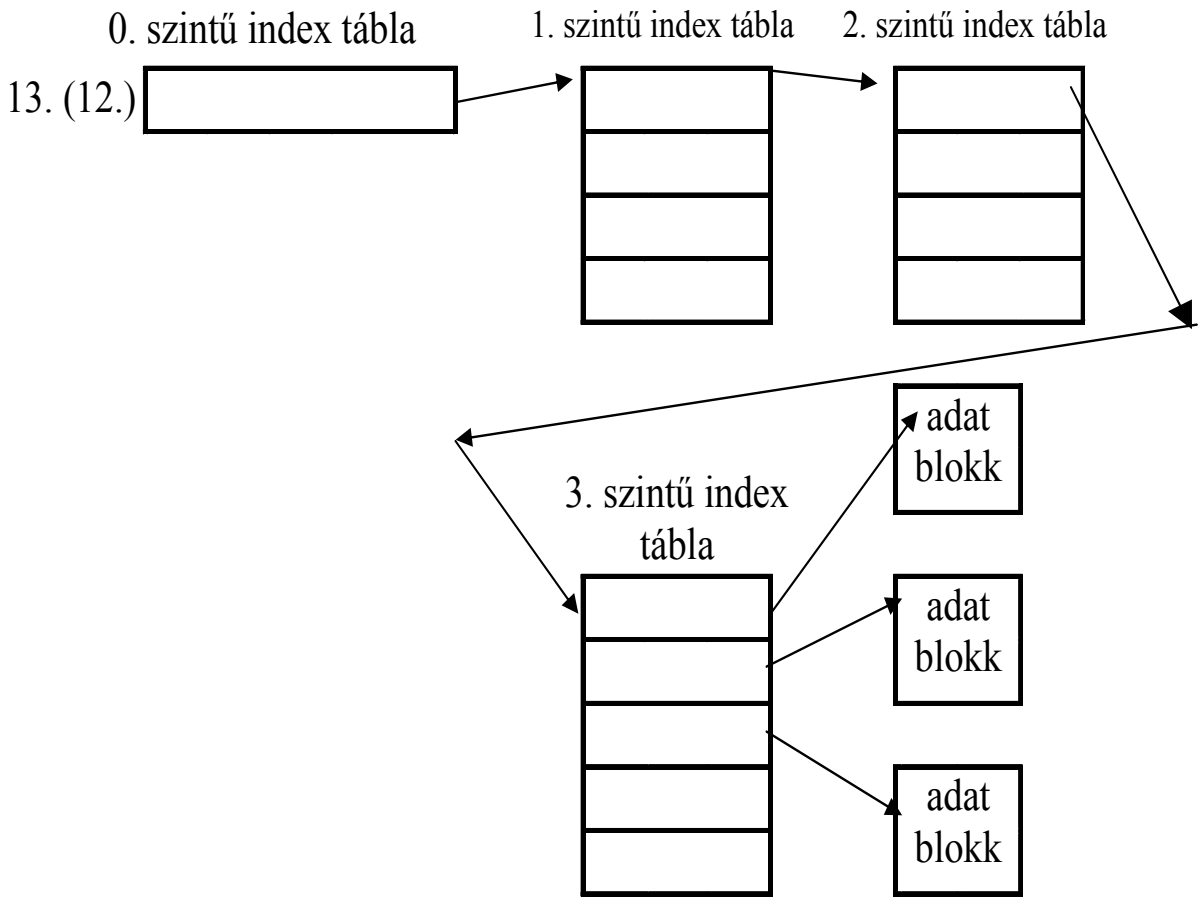
- 1.-10. direkt index esetén: 10 * 1.024 byte = **10 Kbyte-ig**,
- 11. egyszeres indirekt index esetén: 1.024 / 4 = 256 db cím, így: 256 * 1.024 byte = **256 Kbyte-ig**,
- 12. kétszeres indirekt index esetén: 256 * 256 * 1.024 = 65.536 Kbyte = **64 Mbyte-ig**,
- 13. háromszoros indirekt index esetén: 256 * 256 *256 * 1.024 = 16.777.216 Kbyte = **16 Gbyte-ig**.
  Ám a valóságban csak **4 Gbyte-ig**, mert az i-node állományhossz mezője is 4 byte-os!


0. szintűi index tábla
(4 byte-os mutató)
1. (0.)
2. (1.)
3. (2.)
direkt indexelés
10. (9.)
(1x) 11. (10.)
(2x) 12. (11.)
(3x) 13. (12.)
adat blokk
adat blokk
1. szintű index tábla
adat blokk
adat blokk
adat blokk

Kétszeres indexelés



Háromszoros indexelés



A file-hoz tartozó adatblokkok tárolása az i-node-ban

1. példa: 1Kbyte-os lemez blokkok és 4 byte-os blokk címek esetén, egy file 9.500-adik byte-ja kell:
   - 9. direkt blokk (9.216 byte-tól 10.240 byte-ig): 9 * 1.024 = 9.216 + 284 = 9.500 byte.
2. példa: 1Kbyte-os lemez blokkok és 4 byte-os blokk címek esetén, egy file 355.000-adik byte-ja kell:
   - 9. direkt blokk: 10.240 byte,
   - 10. egyszeresen indirekt blokk: 256 * 1.024 = 262.144 byte,
   - 10.240 + 262.144 = 272.384 byte (82.616 byte hiányzik),
   - 11. kétszeresen indirekt blokk első (0. indexű) mutatója által kijelölt, egyszeresen indirekt blokk 80. elemében található. Azaz: (80 * 1.024 = 81.920) + 696 = 82.616.

A file-hoz tartozó adatblokkok tárolása az i-node-ban

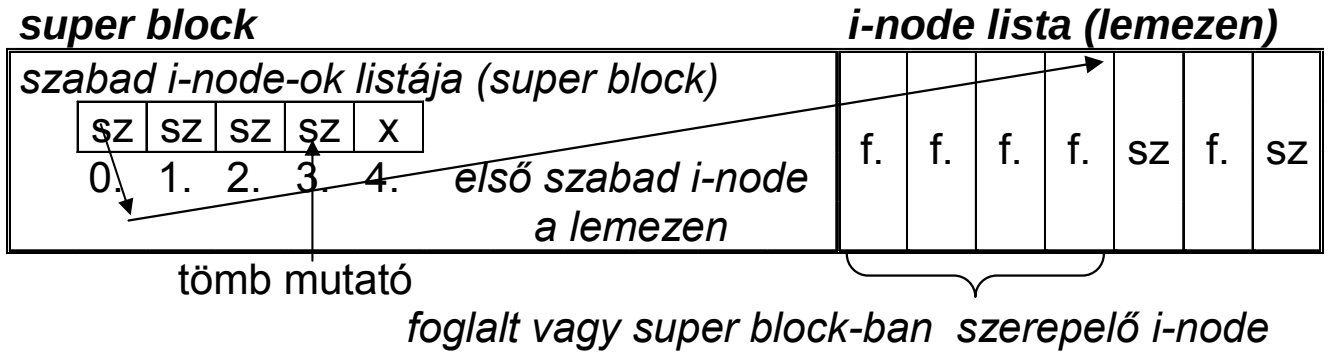
- Hátránya, hogy az indirekt hivatkozási láncok végigjárása időigényes.
- Megoldás lehet a buffer-cache:
  - a device driver (eszközmeghajtó) réteg és a kernel között elhelyezkedő szoftver réteg,
  - buffer objektumok halmaza, amelyek egy-egy lemezblokkot tartalmaznak,
  - a folyamatok ezen (és csak ezen!) a file-”darabkán” dolgozhatnak,
  - a visszaírás a lemezre késleltetett, ezért az áramkimaradás veszélyes lehet.

A SUPER BLOCK tartalma

- A file-rendszer mérete blokkokban.
- A szabad adatblokkok száma.
- A szabad adatblokkok listája.
- A következő szabad adatblokk indexe a szabad adatblokkok listájában.
- Az i-node lista mérete (a maximális file szám).
- A szabad i-node-ok száma.
- A szabad i-node-ok listája.
- Mutató az első szabad i-node-ra a lemezen tárolt i-node listában.
- Lock mezők (i-node, blokklista).
- Módosítás jelzőbit.

Az i-node-ok tárolása a rendszerben

- Cél: gyors kiszolgálása a kéréseknek.



Az i-node-ok tárolása a rendszerben

- Egy típus mező értéke dönti el, hogy az adott i-node szabad (az érték: 0) vagy sem.
- Tárolásuk egy lineáris listában. Ha közvetlenül innen történne a kiszolgálás, az lassú lenne.
- Ezért a hatékonyság növelése miatt, a super block tartalmaz egy fix tömböt, amelyben a szabad i-node-ok sorszámai vannak.
- A tömb feldolgozása mindig hátulról történik egy tömb mutató segítségével.
- A 0. tömb elem az ún. megjegyzett i-node, ez mutat a lemezen lévő első szabad i-node-ra.

Az i-node-ok tárolása a rendszerben

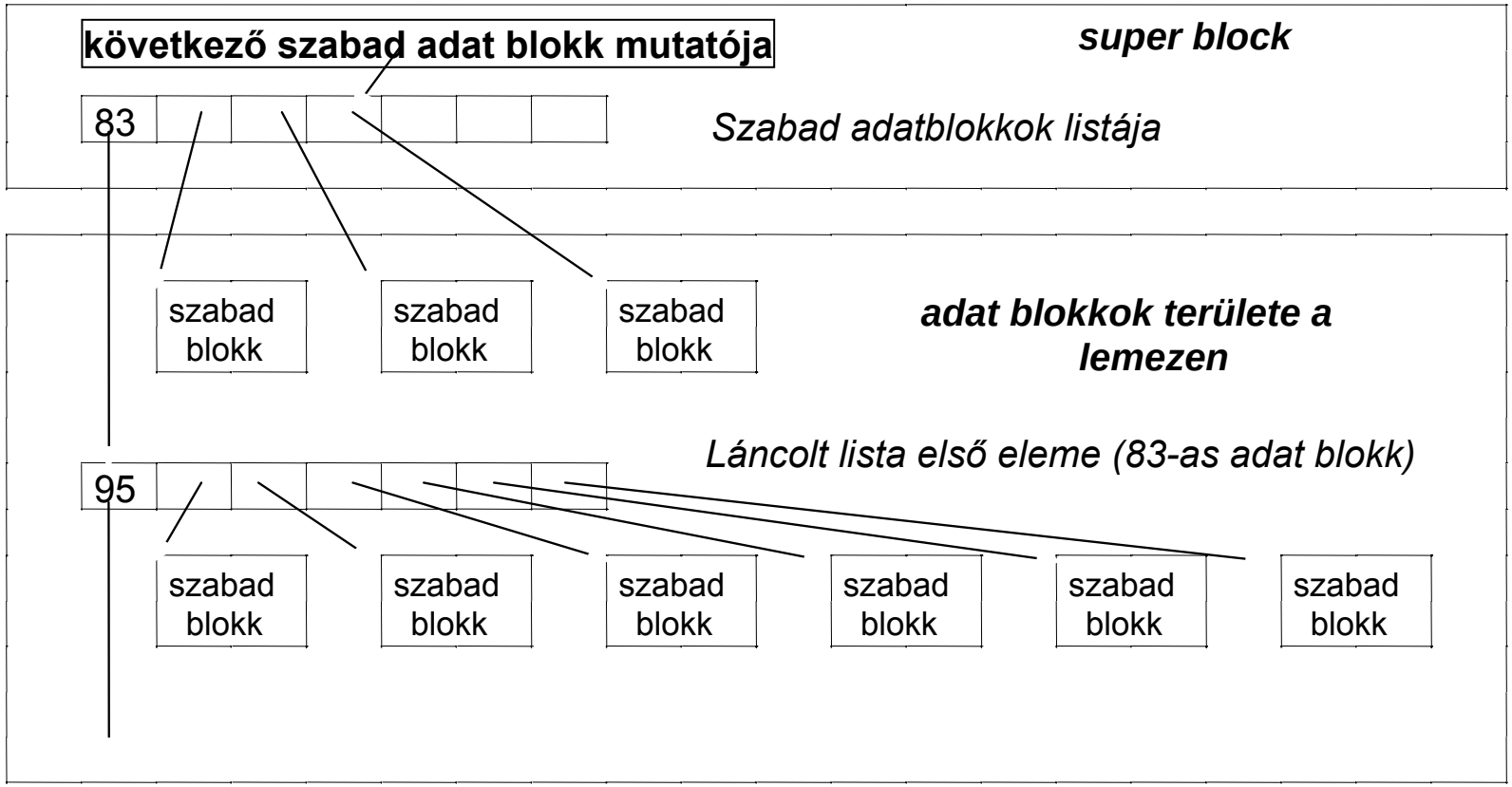
- Egy i-node felszabadulásakor:
  - ha a tömb nincs tele (ugyanis kiszolgáláskor felszabadítja az adott tömb-indexű helyet), akkor beteszi az első szabad helyre, tömbindex növeléssel,
  - ha a tömb tele van és a felszabadított sorszám kisebb mint a megjegyzett i-node, akkor az felülírásra kerül,
  - ha a tömb tele van és a felszabadított sorszám nagyobb mint a megjegyzett i-node, akkor nincs teendő.

Szabad adatblokkok tárolása

Tárolás módja:

- Szabad adatblokkokra mutató tömbök láncolt listája, a super block-ban illetve a lemez adatterületén.
- Cél: gyors kiszolgálása a kéréseknek.
- Adatblokk mérete:
  - 512 byte * 2^n nagyságú, ahol az n tetszőleges egész szám.
  - Általános: az 1 Kbyte-os blokkméret.

Szabad adatblokkok tárolása



Szabad adatblokkok tárolása

- Egy blokk felszabadulásakor:
  - ha a super block tömb nincs tele (ugyanis kiszolgáláskor felszabadítja az adott tömb-indexű helyet), akkor beteszi az első szabad helyre, tömbindex növeléssel,
  - ha a super block tömb tele van, akkor a felszabadult üres blokkba bemásolja a super block tömb elemeit és az így megtelt blokk sorszámát beírja a super block tömb első helyére.

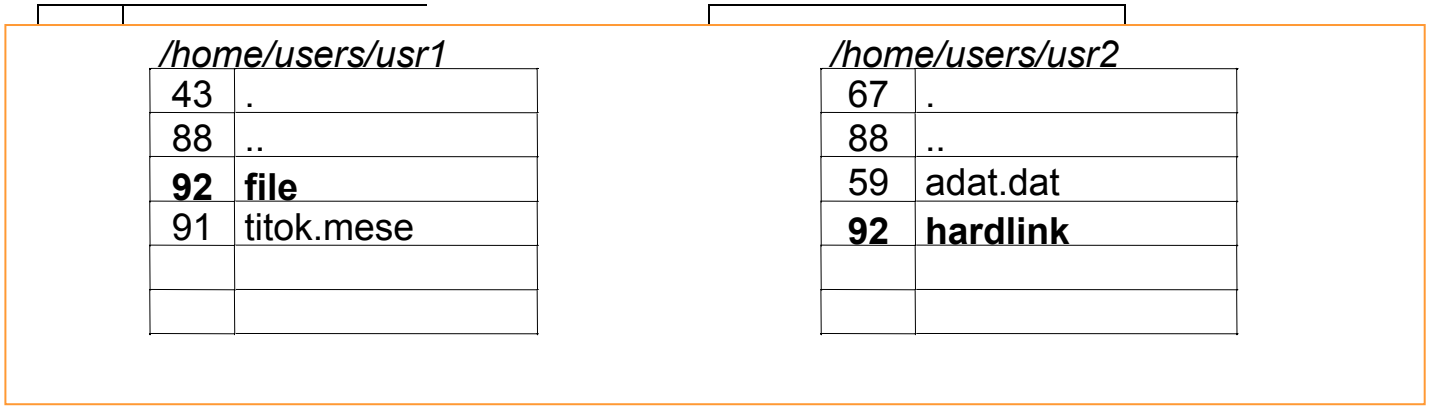
A UNIX file-rendszer felhasználói interface-e

A UNIX file-rendszer felhasználói felületének lehetőségei:

- standard input/output átirányítás,
- mount: két file-rendszer logikai összefűzése,
- link-ek létrehozása:
  - hard link,
  - szimbolikus link (csak FFS-ben),
- pipe (csatorna).

Hard link

- Egy file különböző nevekkel (könyvtár-bejegyzésekkel) történő elérése.
- **`ln /home/users/usr1/file /home/users/usr2/hardlink`**
- Hátrány: Felhasználó elveszítheti a felügyeletet a file-jai felett.



Szimbolikus link

- Hard link hibáinak kiküszöbölése (4.2 BSD).
- Csak az elérési utat tartalmazza.
- Az i-node típus mezője azonosítja.
- **`ln -s /home/users/usr1/file /home/users/usr2/szimb_link`**

/home/users/usr1

43	.
88	..
92	**file**
91	titok.mese

/home/users/usr2

67	.
88	..
59	adat.dat
(24)	**szimb_link:** ***/home/users/usr1/file***

Pipe

A pipe (csatorna, csővezeték):

- FIFO tároló,
- egy vagy több processz írhat bele,
- egy vagy több processz olvashatja,
- automatikus szinkronizáció,
- mérete korlátos, hossza 10 Kbyte,
- megvalósítás:
  - file-alrendszerben (általában),
  - speciális file descriptor párral.

Pipe típusok I.

- Elnevezett pipe, *mknode*(csatorna_név) speciális paranccsal:
  - speciális file, a nevével mindenki hivatkozhat rá,
  - a folyamatoktól függetlenül létezik, explicit törlő utasítással szűnik csak meg. Ha nem töröljük, akkor erőforrásokat köt le feleslegesen!

Pipe típusok II.

- Név nélküli pipe, *pipe*() rendszerhívással:
  - a rendszerhívás kettő file descriptor-ral tér visza,
  - i-node szám van (a gyökér file-rendszerben), hivatkozás könyvtárból nincs,
  - a szülő és a leszármazottai is hivatkozhatnak rá a file descriptor-ral,
  - automatikusan megszűnik, ha az utolsó descriptor is törlődik,
  - kétirányú adatforgalom is lehetséges (valójában kettő független csővezeték keletkezik). Ekkor a file descriptor-rok olvasára és írásra is használhatóak,
  - méretét a direkt módon elérhető adatblokkok össz-mérete határozza meg.

File-kezelés a folyamatok futása során

File-ok kezelése

- A folyamatok egy file-ra névvel, csak annak megnyitásakor - open() - hivatkoznak.
- Műveletek előtt, file nyitás, open():
  - a visszatérési érték a file descriptor (file leíró avagy logikai periféria szám),
  - így a további műveletek a file descriptor-t használják,
  - egy folyamat többször is megnyithat egy file-t, illetve több folyamat megnyithatja ugyanazt, de minden nyitás után egy önálló logikai periféria jön létre, amelyek ugyanarra a fizikai file-ra mutatnak, (rekord szintű kölcsönös kizárás megvalósítása),
  - a gyerek folyamat örökli a szülő logikai perifériáit.

File-ok kezelése

- Processz:
  - File Descriptor Table.
- Rendszer:
  - Global File Descriptor Table,
  - i-node Table (in-core i-node Table).

Adatszerkezetek

- File Descriptor Table, avagy folyamatonkénti állományleíró tábla:
  - minden egyes folyamathoz tartozik 1 ilyen tábla,
  - logikai perifériákat nyilvántartó rekordok sorozata,
  - mindegyik megnyitott állományhoz egy mutató is tartozik, amelyik a Global File Descriptor Table megfelelő struktúrájára mutat,
  - az első három rekord a standard: input, output és error. Ezeket a folyamatok az indulásukkor megnyitva kapják meg.

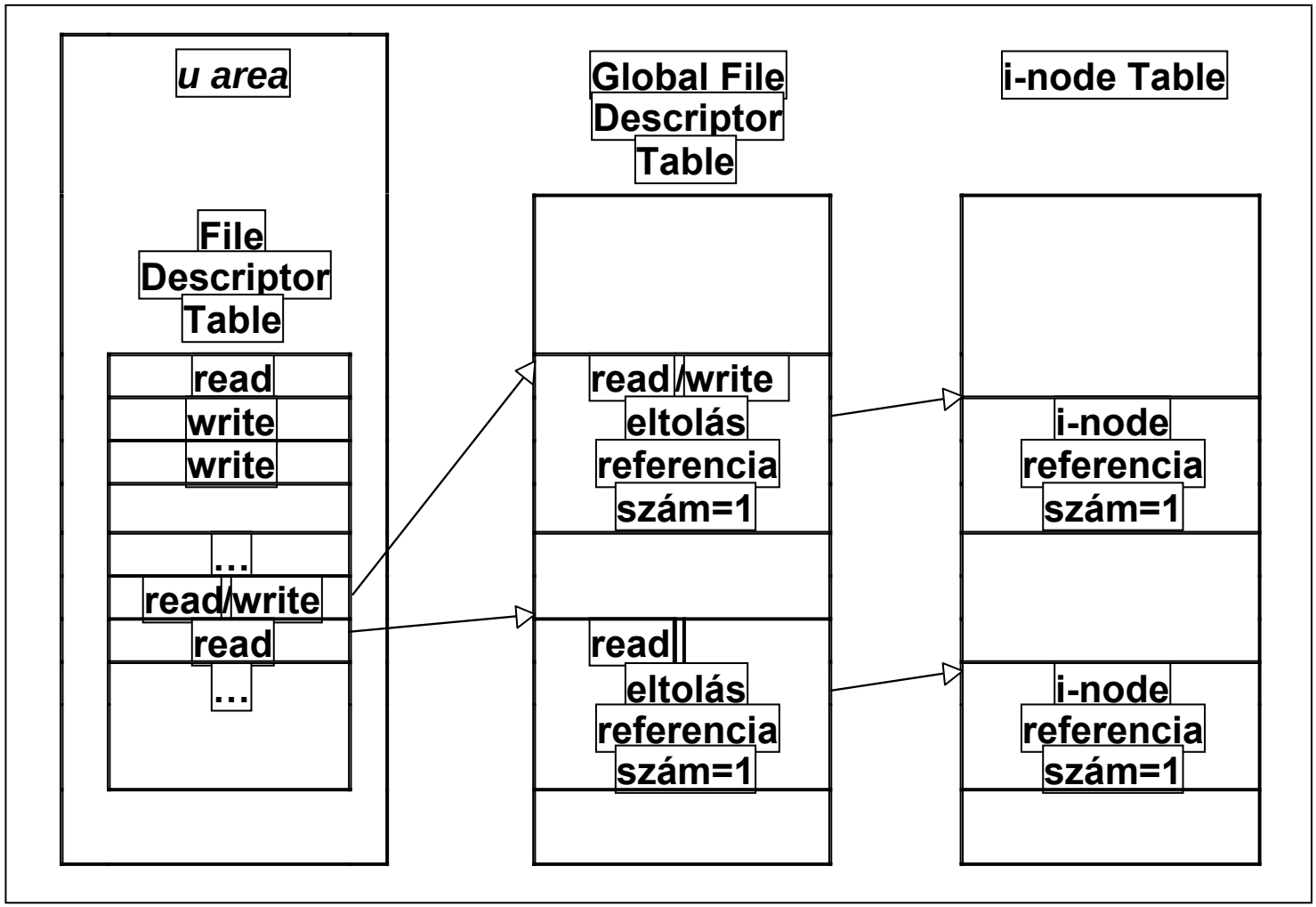
Adatszerkezetek

- Global File Descriptor Table, avagy globális állományleíró tábla:
  - minden open() rendszerhívás esetén létrejön egy struktúra, (1 munkamenet) amely az adott fizikai file-hoz rendelődik,
  - a struktúra jellemző elemei:
    - elérési útvonal,
    - megnyitási mód,
    - jogosultságok,
    - egy file-on belüli pozíció mutató (eltolás),
    - hivatkozás számláló, amely megmutatja, hogy hány darab File Descriptor Table bejegyzés mutat rá,
  - biztosítja az osztott file használatot.

Adatszerkezetek

- i-node Table (in-core i-node Table), avagy
  i-node memória tábla minden nyitott file-hoz:
  - hivatkozásszámláló, amely megmutatja, hogy az adott fizikai file-t hány példányban használják azaz, hogy hány darab Global File Descriptor Table bejegyzés mutat rá,
  - biztosítja a fizikai file elérését.

Adatszerkezetek

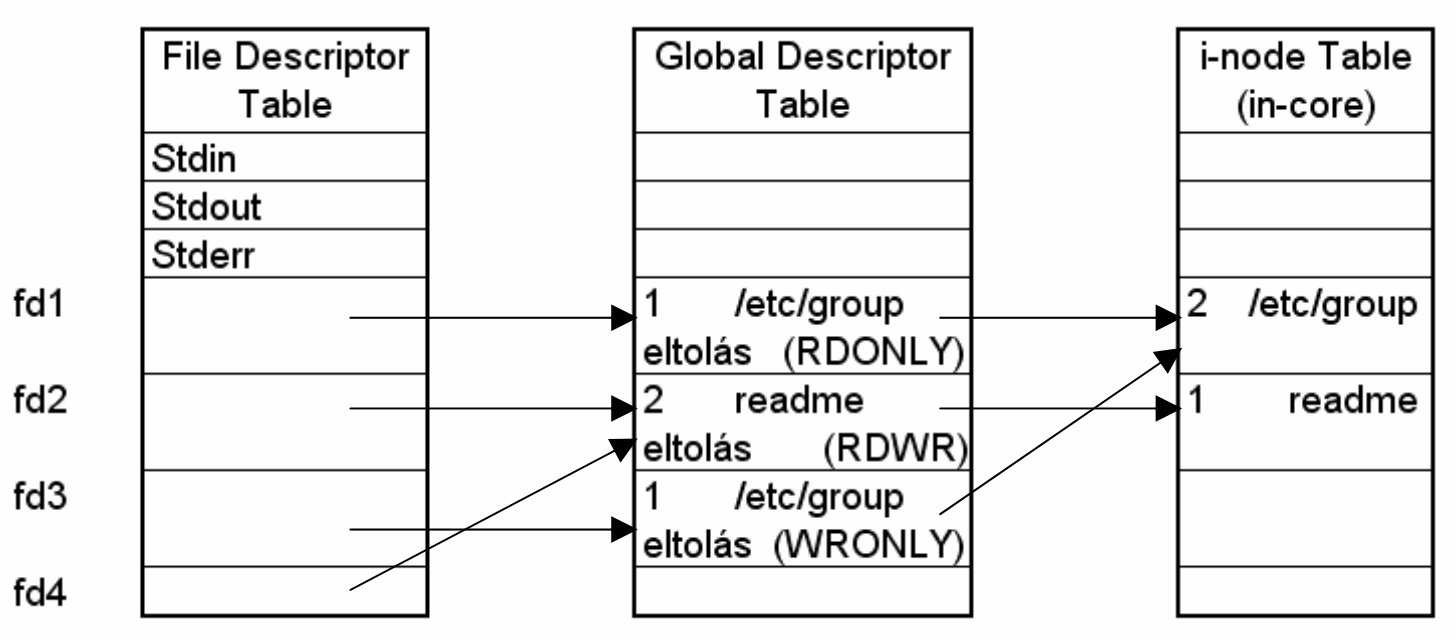


A file-kezelő UNIX rendszerhívások

- fd=open(elérési út/file_név, nyitás módja):
  - a visszatérési érték (fd) egy egész szám,
  - nyitáskor az eltolási érték 0 lesz,
  - a jogok ellenőrzése megtörténik,
  - új bejegyzés (az fd értéke) a processz FDT-ben.
- read/write(olvasandó/írandó byte_szám, file_descriptor).
- chown(file_descriptor): hatásos UID megváltoztatása.
- utimes(file_descriptor): időpontok módosítása.
- fd=dup(file_descriptor): file_descriptor duplikálása:
  - az FDT első szabad helyére íródik be.

Adatszerkezetek példa

- fd1=open(”/etc/group”, O_RDONLY);
- fd2=open(”readme”, O_RDWR);
- fd3=open(”/etc/group”, O_WRONLY);
- fd4=dup(fd2);



Adatszerkezetek példa

A file nyitások egymás után történnek, majd ezek után következnek a műveletek, így:

- fd1: a group file-t olvassa az elejétől (0 eltolástól),
- fd2: a readme file-t olvassa/írja az elejétől (0 eltolástól),
- fd3: a group file-t írja az elejétől (0 eltolástól),
- fd4: a readme file-t olvassa/írja az aktuális eltolástól.

A dup() rendszerhívás

- Tipikusan a standard átirányításoknál használatos.
- Egy lehetséges, de nem hibatűrő séma:
  - a FDT-ben bezárul az Stdout (átirányítás kezdeményezése),
  - open()-nel file nyit,
  - ha a nyitás során hiba lép fel (pl. nincs hely az újabb file-nak) elveszítettük a kimeneti eszközt!
- Egy lehetséges hibatűrő séma:
  - fdout=open()-nel egy file-t nyit,
  - ha a nyitás során hiba lép fel akkor, ezt jelezni tudjuk a kimeneten,
  - ha nincs hiba akkor, a FDT-ben bezárul az Stdout,
  - fdd=dup(fdout) hívása, így az Stdout helyére íródik be a FDT-ben az fdd file leíró,
  - tehát a kimenetre szánt információ, az fdout azonosítójú file-ba íródik!

Standard csatornák I.

- Input: STDIN, file-descriptora: 0 (szabványos bemeneti csatorna).
- Output: STDOUT, file-descriptora: 1 (szabványos kimeneti csatorna).
- Error: STDERR, file-descriptora: 2 (szabványos hiba csatorna).

Standard csatornák II.

- Új file nyitása:
  - első szabad file descriptor.
- Standard csatornák átirányítása:
  - file-descriptor-t duplikálunk (átmásolunk a dup() rendszerhívással).

Az UFS hátrányai

- A szuperblokk csak 1 példányban létezik, sérülése esetén elszállhat az állományrendszer!
- Az i-node-ok az állományrendszer elején tárolódnak, majd az adatok következnek. Így a fejmozgási idő jelentősen megnövelheti a válaszidőt!
- Az i-node-ok kiosztása véletlenszerű (pl. azonos könyvtárban lévő állományok).
- A szabad blokkok kiosztása is véletlenszerű (lassul az elérés)!
- Az 512 byte-os blokk méret miatt, a belső tördelődés ugyan alacsony, de az egy lemezművelettel beolvasható adat is ”csekély”.

Az UFS hátrányai

- Maximum 16 byte-os állomány-bejegyzés (2: az i-node számnak, 14: az azonosítónak).
- A 16 bit-es inode szám csak 2^{16}-1 = 65.535 db állományt enged meg.
- Csak lokális állományrendszereket tud kezelni.
- Továbbfejlesztés vált szükségessé!

Az FFS tulajdonságai

- Az egyik leggyakrabban használt, továbbfejlesztett file-rendszer az FFS (Fast File System).
- A legjelentősebb változtatás a fizikai lemez használatában történt.
- A szuperblokk nem 1 kizárólagos példányban létezik, hanem cilinder-csoportonként.
- Cilinder csoport:
  - a fizikai lemez egymás után következő sávjai (track-jei), két vagy több oldal esetén cilinderei,
  - mérete általában 8 Mbyte,
  - szerkezeteik megegyeznek.
- Az i-node-ok és a hozzájuk tartozó adatok egy cilinder-csoportban vannak. Így a fejmozgási idő optimalizálható.
- Az i-node-ok kiosztása irányított (pl. azonos könyvtárban lévő állományok).
- A szabad blokkok kiosztása is irányított (gyorsabb az elérés).

Az FFS tulajdonságai

- Minden könyvtárat másik cilinder csoportra helyez, az egyenletes lemez használat miatt.
- A nagy file-okat szétteríti a cilinder csoportok között, ha azok méretei egy bizonyos határt meghaladnak (a direkt címzésű blokkok nagyságát).
  Így akadályozza meg, hogy 1 file felhasználja az adott cilinder csoport területének nagy részét.
- Interleaving technika alkalmazása a blokkok allokációja során.

Az FFS tulajdonságai

- Általános a 4 illetve a 8 Kbyte-os blokk méret:
  - Nő az egy lemezművelettel beolvasható adat mennyiség.
  - Pl.: 4 Kbyte-os blokk méret és 4 byte-os cím esetén:
    - 1.-10. direkt index esetén: 10 * 4.096 byte = **40 Kbyte-ig**,
    - 11. egyszeres indirekt index esetén: 4.096 / 4 = 1.024 db cím, így: 1.024 * 4.096 byte = **4 Mbyte-ig**,
    - 12. kétszeres indirekt index esetén: 1.024 * 1.024 * 4.096 = **4 Gbyte-ig**,

    lehet címezni és nincs szükség a háromszoros indirekt címzésre!
  - A belső tördelődés kiküszöbölése a töredék (fragment) blokkokkal.
- Maximum 255 karakter hosszú file-azonosító.
- 4 byte-os i-node szám.
- A 32 bit-es i-node számmal 2^{32}-1 = 4.294.967.295 db állományt lehet létrehozni.

Az FFS tulajdonságai

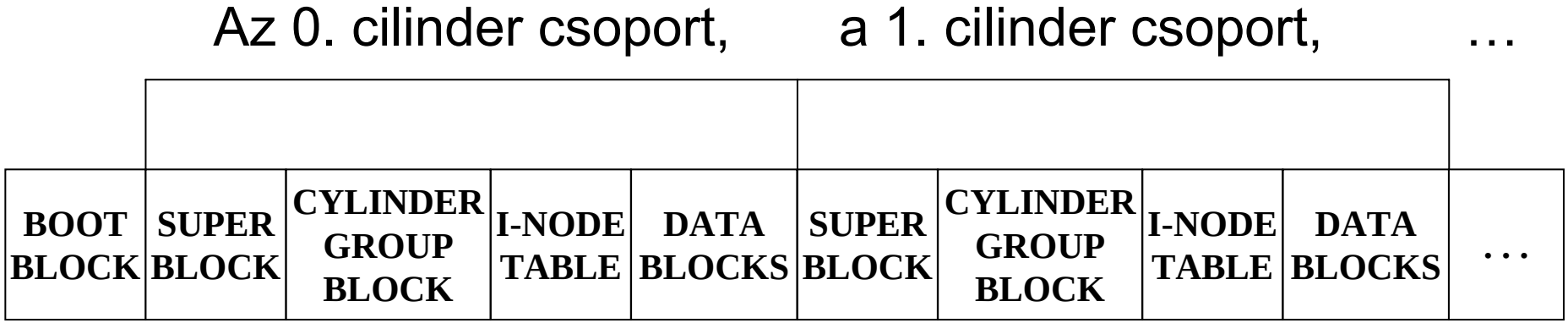
- A könyvtárbejegyzés két részből áll:
  - rögzített rész:
    - a 4 byte-os i-node szám,
    - file azonosítónak lefoglalt terület nagysága byte-ban,
    - a file azonosító hossza byte-ban,
  - változó hosszúságú rész:
    - 4 byte-os foglalási egységekben lehet növelni,
    - a ki nem használt részek 0-val töltődnek fel.

Az FFS tulajdonságai

- A szimbolikus link tulajdonságai:
  - adott elérési útvonalon egy file-azonosítóra mutat,
  - az eredeti file törléskor a ”semmibe” mutat,
  - beillesztett file-rendszer esetén:
    - a mount táblában tárolt super block alapján betöltődik a beillesztett file-rendszer gyökér könyvtára, és keresés ott folytatódik.

Az FFS tulajdonságai

File-rendszer a lemezen:



A SUPER BLOCK tartalma

- A file-rendszer mérete blokkokban.
- 1 blokk mérete byte-okban (4096 - 8192).
- A blokknál kisebb foglalási egység (fragment) mérete.
- A maximális i-node-ok száma.
- A szabad i-node-ok száma a rendszerben (bit-térképpel).
- Az i-node-ok száma a cilinder csoporton belül.
- Módosítás jelzőbit.

A töredék blokk

- Csak az utolsó adatblokk tartalmazhat 1 vagy több ilyen külön címezhető blokkot.
- A maximális számuk meghatározása a file-rendszer létrehozásakor történik. Így lehet:
  - 1: ekkor a teljes adatblokkot,
  - 2: ekkor csak az adatblokk felét,
  - 4: ekkor csak az adatblokk negyedét,
  - 8: ekkor csak az adatblokk nyolcadát,

  foglalja le a töredék.

A töredék blokk

- Pl.: 4 Kbyte-os adatblokk és 8 töredékblokk esetén, a megcímezhető legkisebb fragment 512 byte, azaz a szektorméret.
- Hátrányai:
  - többlet adminisztráció,
  - file bővülés esetén, a több file által használt blokk töredékeinek a kimozgatása egy másik töredék blokkba.
- Ezért ha csak lehet, az alkalmazások teljes blokkal dolgoznak.

A CYLINDER GROUP BLOCK tartalma

- Az adatblokkok száma és,
- az i-node-ok száma,

  az adott cilinder csoportban.
- A szabad adatblokkok száma,
- a szabad töredékek száma és,
- a szabad i-node-ok száma,

  és elhelyezkedésük az adott cilinder csoportban.

Az elhelyezkedést bit-térképpes ábrázolással oldják meg a listás ábrázolás helyett, mert az utóbbi nem elég hatékony.

Az I-NODE TABLE tartalma

- Ez nem változott, lásd a 7. - 11. diákat.

A DATA BLOKS tartalma

- A file-rendszerben tárolt file-ok adatblokkjai.
- A file-ok elhelyezkedését leíró egyszeresen illetve kétszeresen indirekt címblokkok.