Operációs rendszerek

Tárkezelés

Témák

- I. Memória (központi tár) kezelés
  - 1. Programok fizikai tárigényének csökkentése
  - 2. Memória hézagmentes kitöltése.
  - 3. Háttértár használata memória kiváltására.
- II. Állományrendszerek
- Mágneslemezes háttértár kezelése

Adattároló eszközök hierarchiája

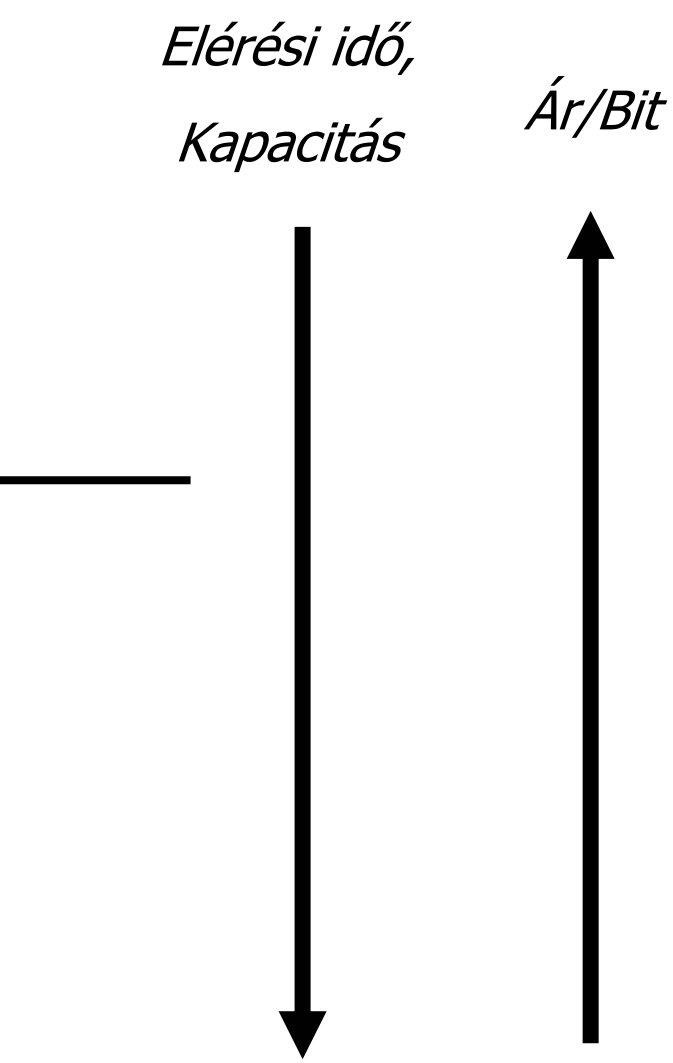
Számítógépek adattároló eszközeit hierarchiába rendezhetjük:

- CPU regiszterek
- operatív tár (memória)
- háttértár (másodlagos tár)
- külső tárak (harmadlagos tárak)

Számítógépek adattároló eszközeinek hierarchiája

- regiszterek
- gyorsmemória (cache)
- központi memória
- elektronikus diszk

- mágneses diszk
- optikai diszk
- magneto-optikai diszk
- mágnesszalag



Tárkezelés

- Program végrehajtás: tárolók közötti adatmozgatás.
- Háttértár - Memória - Processzor.
- Adatmozgatás meggyorsítása: gyorsító tárakat (cache) alkalmazása.
- Két szint közé épül be (alacsonyabb szinten –magasabb szintű adatrészek).

Gyorsító tárak

- Processzor:
  - utasítás és adat cache.
- Virtuális memória:

  fizikai memóriánál nagyobb címtartomány kezelése (teljes címtartomány a háttértáron).
- I/O kezelés:
  - buffer cache (file-ok részeinek tárolása).
- RAM diszk (több file teljes tárolása).

Tárkezelés

- A gyorsító tárak alkalmazásának, hatékonyságának alapjai:
  - Lokalítás (adat-utasítás környezet),
  - Szekvenciális működés.
- OPR hatókörei:
  - Memória, másod- és harmadlagos háttértárak file rendszerei.
  - Virtuális és buffer-cache.

I. Memória (központi tár) kezelés

Memória kezelési elvek

Logikai – Fizikai címtartományok.

- 1. Programok fizikai tárigényének csökkentése.
- 2. Memória hézagmentes kitöltése.
- 3. Háttértár használata memória kiváltására.

1. Programok fizikai tárigényének csökkentése

A tárkezelés az operációs rendszerekben

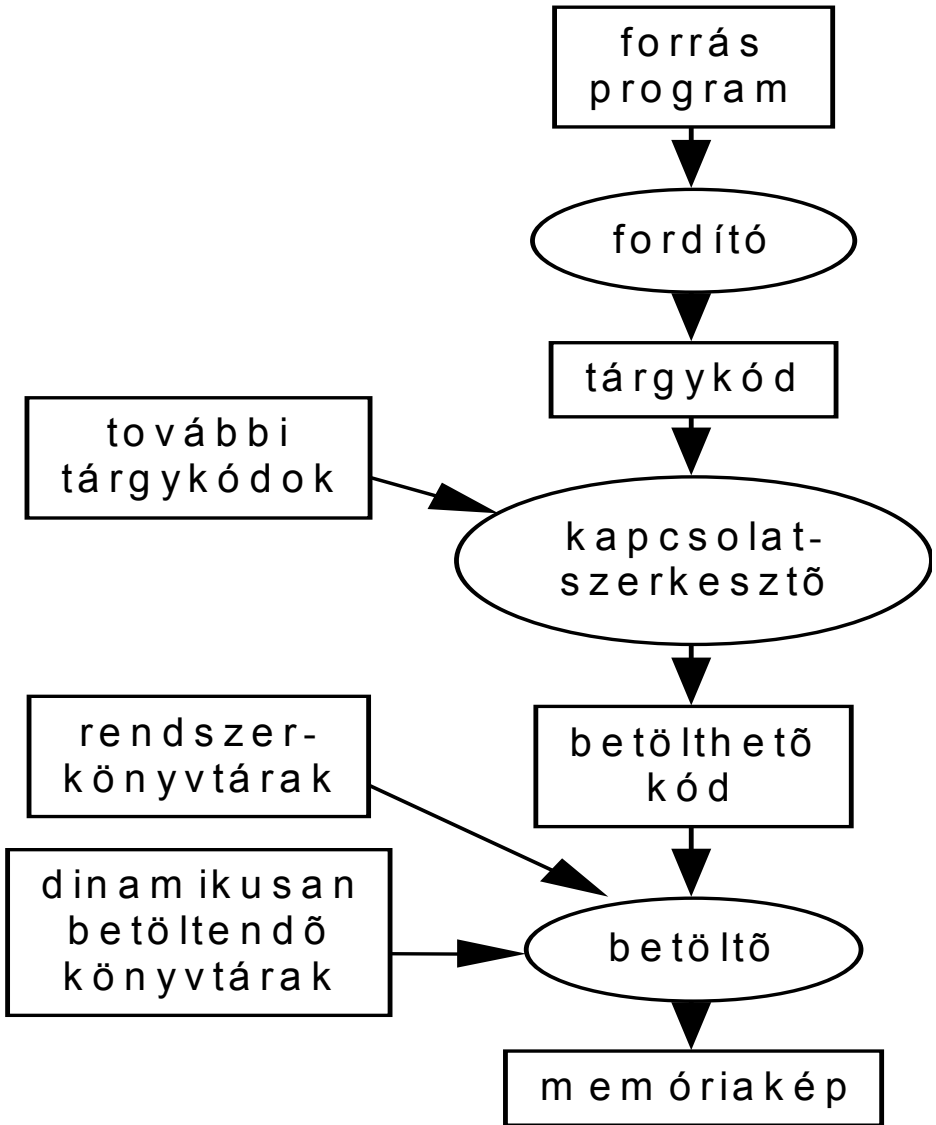
- Multiprogramozás:
  egyidejűleg több folyamat tartózkodik a központi tárban.
- Klasszikus tárkezelés fő kérdései:
  - Címek kötése (szimbolikus hivatkozás - fizikai memória cím).
  - Tár allokáció.

Programok címeinek kötése statikusan

A logikai-fizikai cím hozzárendelés történhet:

- Fordításkor:
  - abszolút kód (merev, csak ROM).
- Szerkesztéskor:
  - kapcsolatszerkesztő (linker) program,
  - áthelyező kód (0 és eltolás).
- Betöltéskor: (logikai cím)
  - áthelyező betöltő program (relocating loader),
  - áthelyező kód módosítása, az aktuális címkiosztás szerint.

Programok címeinek kötése



Programok címeinek kötése dinamikusan

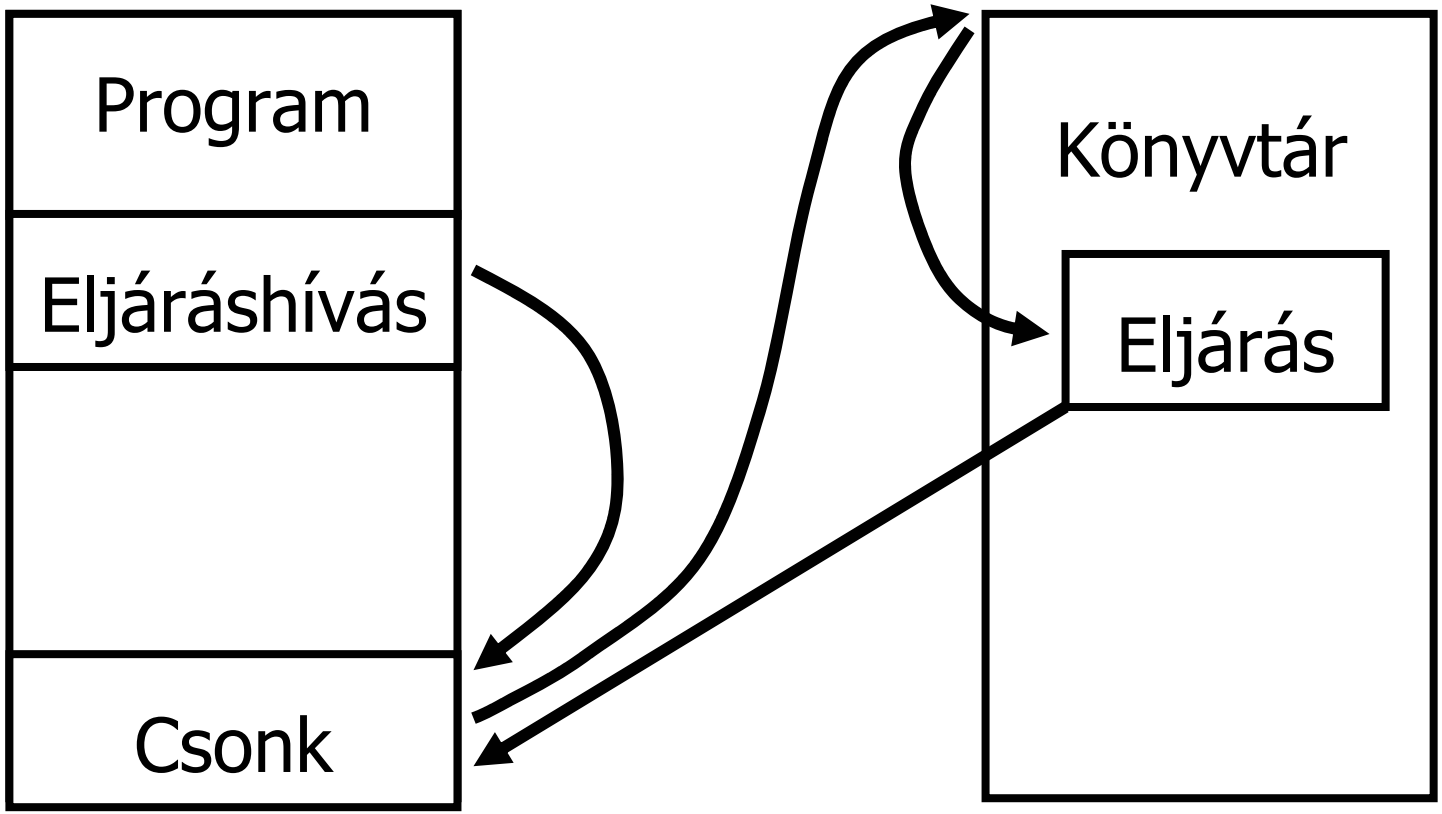
- Futás közben: (fizikai cím futáskor, dinamikus címleképezés, ha csak ilyen kódokat használ az adott program!)
  - Hardver támogatással (bázis-relatív címzés, a bázisregiszterben a betöltési kezdőcím + d, tetszőleges helyre betölthető!).
  - Egy változata az utasításszámláló-relatív címzés. Ez is pozíció-független kódot eredményez.
  - Újrahívhatósági feltétel (egy eljárást hív egy időben több program. Nincs saját változó!!!).

Programok fizikai tárigényének csökkentése

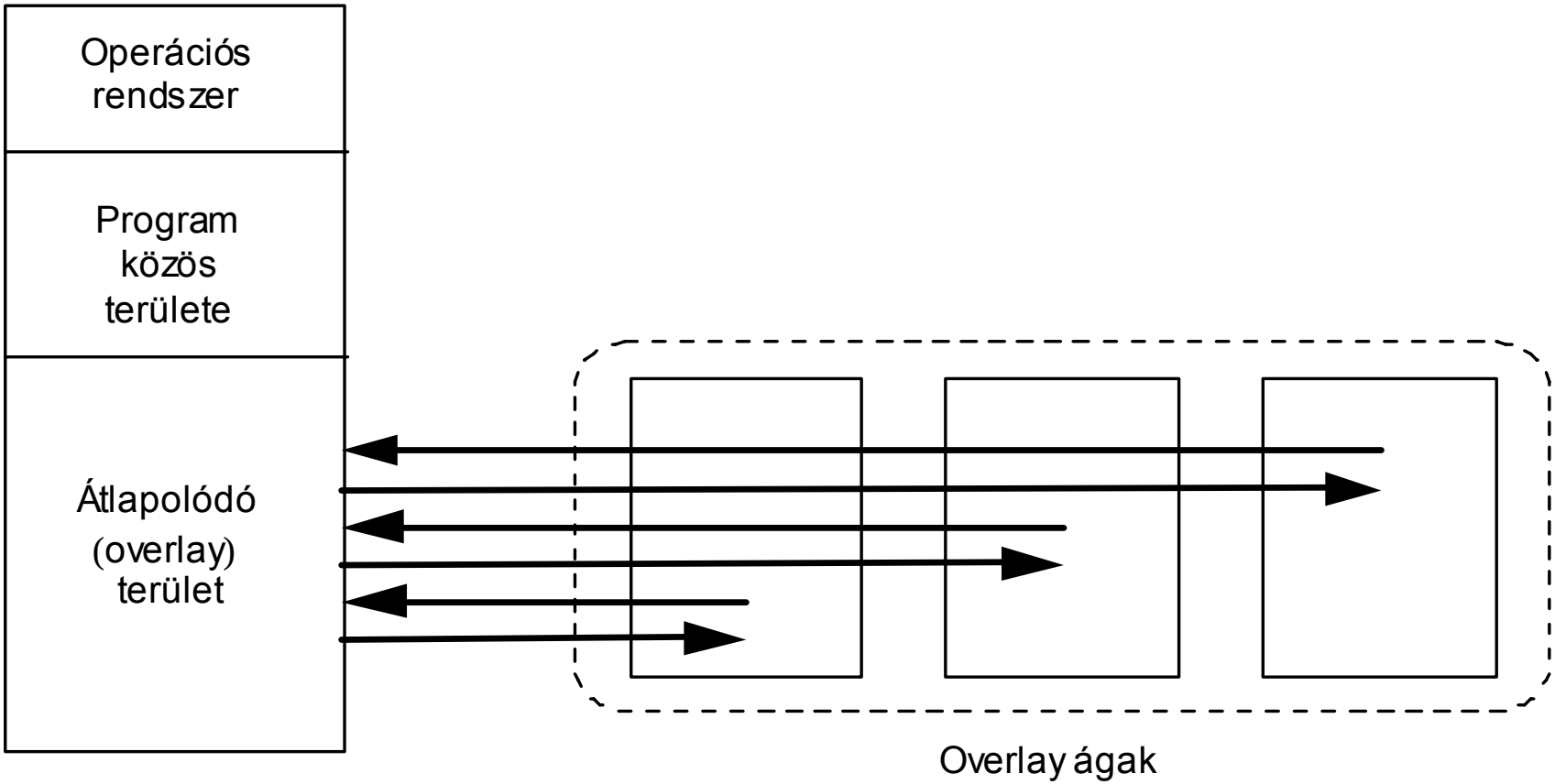
Gazdaságos memória-kihasználás megvalósítása.

- Dinamikus (késleltetett) betöltés (dynamic loading) - nincs OPR támogatás. Ritkán használt eljárásoknál.
- Dinamikus linkelhető könyvtárak (dynamicaly linked library, DLL):
  OPR támogatás, csonk – könyvtár - eljárás. Fizikai cím rendelése a logikaihoz. Újrahívható!
- Egymást átfedő programrészletek (overlay) - nincs OPR támogatás. Több részre bontás, de csak egy dolgozik.

Dinamikus könyvtárbetöltés



Overlay memóriaszervezés



2. Memória hézagmentes kitöltése

Tárkiosztási algoritmusok

Társzervezési elvek

- Egy partíciós rendszerek:
  - OPR és egyetlen alkalmazói folyamat.
  - OPR védelme határregiszterrel (legkisebb program cím).
  - Rendszer mód vagy felhasználói mód.
  - Probléma:
    OPR terület növekedése. (tár ”túl-vége”, vagy a program által nem használt terület.)

Társzervezési elvek

Több partíciós rendszerek:

- OPR és a többi folyamat védelme partíciónként alsó és felső határregiszterrel (kicímzés figyelés).
- Fix partíciók:
  belső tördelődés (internal fragmentation).
- Változó partíciók:
  külső tördelődés (external fragmentation),
  felszámolható: dinamikus áthelyezhetőség!
  Ám kicsit lassít! Érdemes-e?

Tárallokációs algoritmusok változó méretű partíciókná

- Első illeszkedő (first fit), (a tár elejétől, igen gyors, a memória ~30%-a marad kihasználatlan! {50%-os szabály}).
- Következő illeszkedő (next fit), (utoljára lefoglalt tartomány végétől, ~ azonos hatásfokú).
- Legjobban illeszkedő (best fit), (legkisebb még elegendő, rosszabb memória kihasználás!).
- Legkevésbé illeszkedő (worst fit) (legnagyobból és reménykedik, ~50%-os memória kihasználtság).

Memória-kihasználás változó méretű partíciók esetén

- 50%-os szabály:
  a folyamatok, az általuk lefoglalt partíció területeknek kb. a felét kihasználatlanul hagyják. Azaz a teljes tár ~ 30%-át.

Programok címeinek kötése futási időben

Hardver támogatás szükséges:

- címtranszformációs tábla használata.

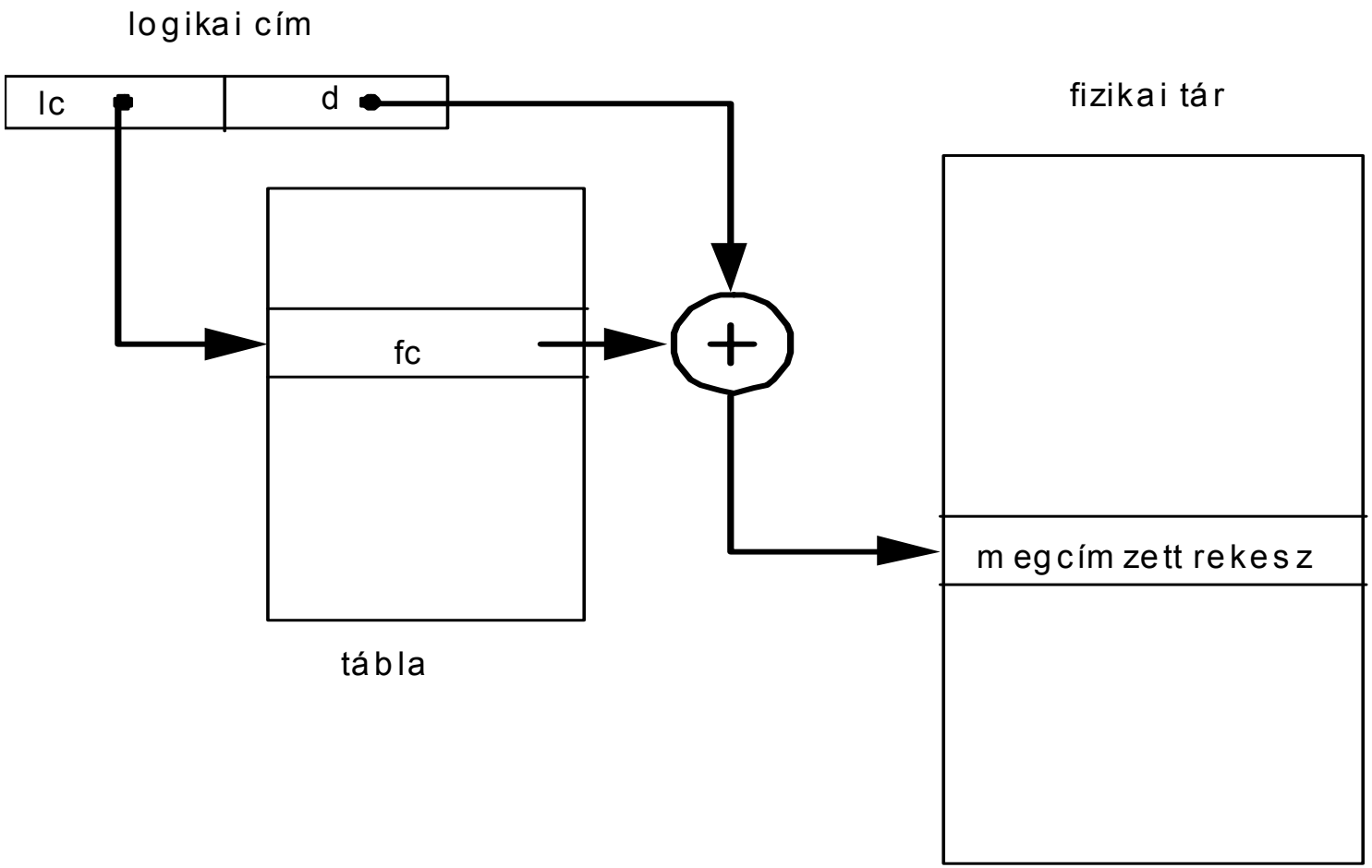
Mesterséges folytonosság:

- folytonos logikai címtartományt feleltetünk meg, nem folytonos fizikai címtartománynak,
- megszűnő külső, csökkenő belső tördelődés.

Minden folyamathoz külön tábla.

Táblák cseréje pointer-ezéssel.

Címtranszformáció

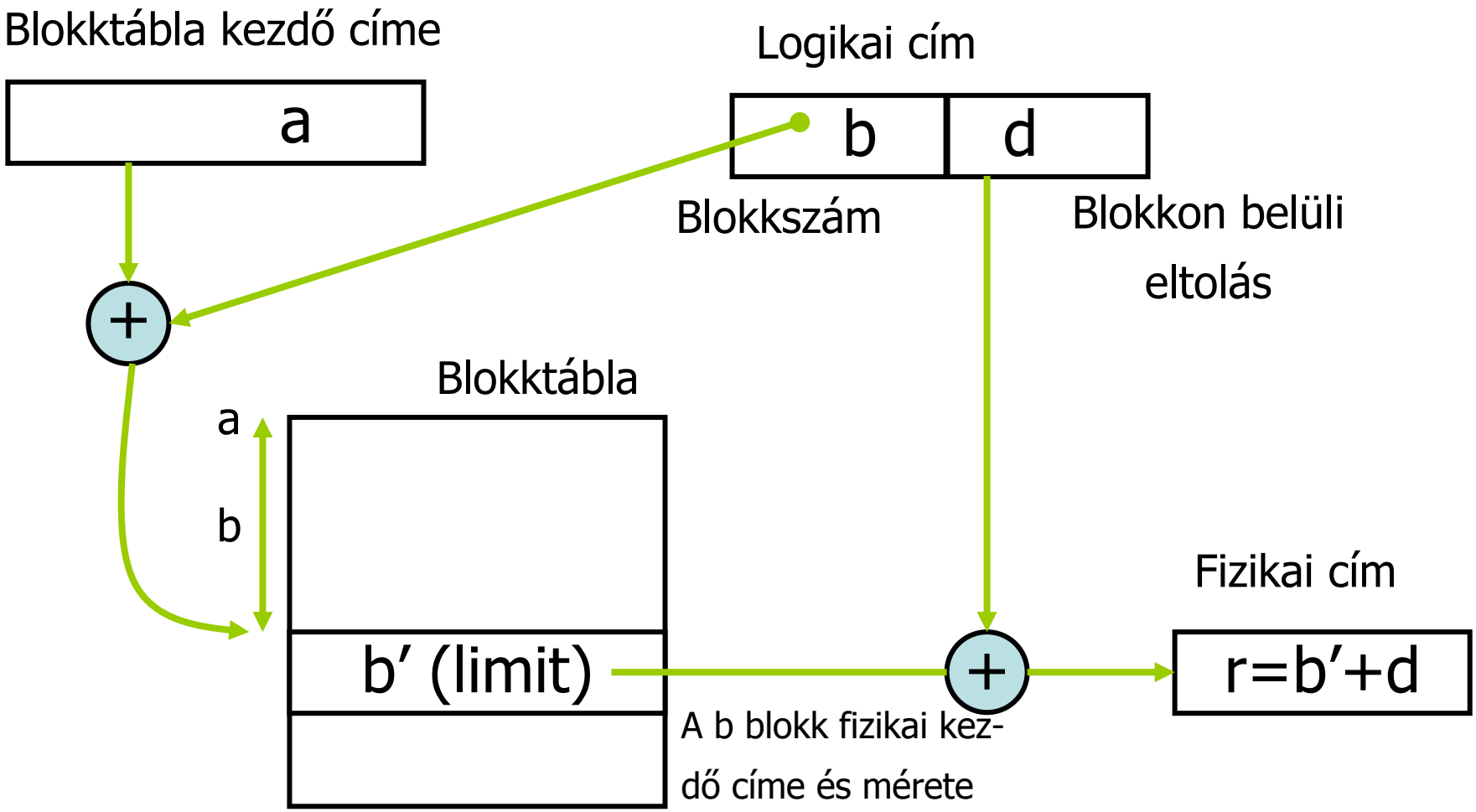


Szegmensszervezésű memória-kiosztás

Jellemzők:

- összefüggő logikai címtartomány - eltérő hosszú blokk – tetszőleges de folytonos fizikai címtartomány,
- szegmens hossz rögzítése (ne legyen kicímzés),
- memóriavédelemhez szegmensméret tárolása a címtranszformációs táblában,
- nincs belső tördelődés, de van külső,
- osztott szegmens használat (n folyamat - 1 eljárást használ, azonos fizikai címre mutatván), kölcsönös kizárás folyamat-szintű megvalósítása.

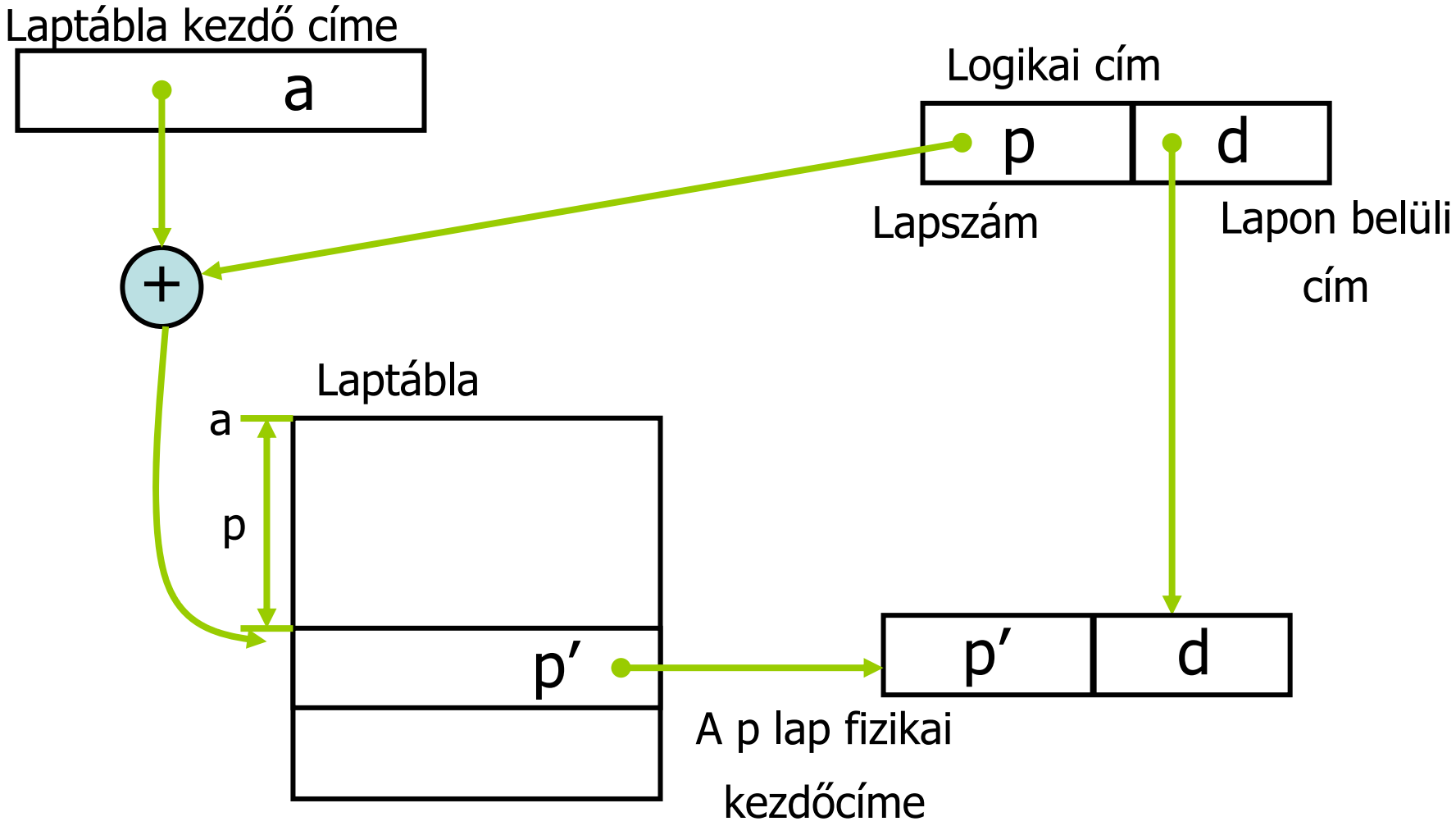
Címtranszformáció szegmensszervezés esetén



Lapszervezésű memória-kiosztás

- Azonos fix lapméret (a logikai és fizikai címtartományok, 2 hatványaiként valósítják meg).
- Egyszerű címtranszformáció:
  - a d mindig azonos bitszélességű (+logikai d!),
  - aktuális lapcím rögzítése, lapon belül csak d kell!
- Teljes lapok miatt nincs külső, de van belső tördelődés. (ha a lapméret csökken, nő a laptábla így a keresési idő is!)
- Egyszerű memóriavédelem:
  - az eltolás (d) értéke fix, így nincs kicímzés.
- Kettő- ill. háromszintű laptáblák a túl nagy laptábla méret elkerülése miatt.

Címtranszformáció lapszervezés esetén



Címtranszformáció a gyakorlatban

Logikai címek fizikai címmé való transz-formálásának módszerei:

- Állandó blokkméret: lapszervezés.
- Változó blokkméret: szegmensszervezés.
- Kombinált szegmens és lapszervezés (szegmens- és lapcím és d).
- Többszintű címtranszformáció:
  - 1. szegmenstábla ami laptábla címeket tartalmaz,
  - 2. laptábla ami fizikai lapcímeket tartalmaz,
  - nincs külső tördelődés, a belső szegmensenként jelentkezik.

3. Háttértár használata memória kiváltására

Virtuális memóriakezelés

Tárcsere

Tárcsere (*swap*)

- Fizikai tár időbeni szétosztása a folyamatok között.
- Folyamat hosszabb várakozási ideje alatt ill. kevés szabad központi tár esetén.
- Folyamat **teljes** tárterületének ill. kisebb részeinek (dinamikus leképzés) háttértárra mentése és visszaállítása.
- Átlapolt tárcsere.

Virtuális memóriakezelés (VM)

- Tapasztalat:
  folyamatok csak egy kis részét használják aktívan az általuk lefoglalt memóriának.
- A **hardver támogatta címtranszformáció** lehetővé teszi az időlegesen nem használt memórialapok háttértárra történő mentését.
- Ám bármelyik folyamat szabadon hivatkozhat bármilyen tartományban szereplő logikai címre! (Így a határ a CSÉ ☺, Winyó méret.)
- A folyamatok VM-t látnak, a valósban csak a futáshoz szükséges rész van!

A VM működése

Laphiba esete:

- háttértáron tárolt memórialapra történő hivatkozás,
- laphiba megszakítás (speciális, mert utasítás végrehajtása közben is működnie kell!),
- folyamat leállítása (visszagörgetése),
- hivatkozott lap beemelése a memóriába,
- megszakított folyamat folytatása (a kezdőponttól, leggyakrabban).

A VM működése

Logikai lapcím ”értékei” (valid bit):

- érvényes (a lap a memóriában van),
- érvénytelen:
  - laphiba megszakítás (lsd. előbb),
  - kicímzés (”hihetetlen” cím adása pl. boot-szektor).

OR feladata VM esetén

- Három alapvető kérdés:
  - melyik lapot hozzuk be a tárba?,
  - melyik lapot cseréljük le? (nincs szabad hely a memóriában),
  - melyik folyamat, mennyi lapot kapjon?.
- Laphibák számának minimalizálása. (A rendszer egyensúlyban tartásához a CPU nem lehet tétlen! {Laphiba kezelés közbeni laphiba.})

Betöltendő lap kiválasztása

- Igény szerinti lapozás:
  - Csak a laphibánál hivatkozott lapot hozzuk be (egyszerű – új lap mindig hibát generál).
- Előretekintő lapozás:
  - ”Jóslás”, lehet, hogy kell ez a lap.
  - A folyamat felgyorsul (csökkenő laphibák), vagy felesleges lapok foglalják a valós memóriát! (A memória egyre olcsóbb ezért egyre népszerűbb ez a módszer.)

Lapcsere stratégiák

- Optimális algoritmus (OPT):
  - azt a lapot kell kimenteni, (ha változott) amelyre legkésőbb lesz szükség,
  - elméleti lehetőség - gyakorlatban csak közelítő algoritmusok.
  - a programok lokális működésének elve: a közelmúltban használt lapra lesz hivatkozás a közeljövőben,
  - legkevesebb laphiba szám.

Legrégebben nem használt lap (LRU)

- Legrégebben nem használt lapot kiírja a háttértárra.
- Megvalósítások:
  - számlálóval (hivatkozási idő):
    - használati idő tárolása, ill. legrégebbi idő,
  - láncolt lista:
    - használatkor lista végére fűzés,
  - hátrányuk: bonyolult HW támogatás.

FIFO tárolóra alapuló lapcsere

- Legrégebbi lap (FIFO)
  - Gyakran használt lapok is kikerülhetnek.
- Újabb esély (FIFO + használat)
  - Egy használat tényét rögzítő jelzőbitet (R) tárol minden laphoz.
  - Hivatkozáskor bebillenti.
  - Ha a sor elején levő lap hivatkozás jelzője be van billentve, nem teszi ki, hanem a sor végére állítja.
  - Jó teljesítményű, így a legnépszerűbb.

VM a gyakorlatban

Lapok mentésének általános gyorsítása:

- Módosított / nem módosított (M) bit használata.
- Módosított lapok mentése tétlen időben.

Így az áldozati sorrend:

- nem_használt – nem_módosított,
- nem_használt – módosított,
- használt – nem_módosított,
- használt – módosított,

Gazdálkodás a fizikai memóriával

- Globális memória gazdálkodás (más folyamat lapjai az áldozatok, egy sokfelé hivatkozó kiszorítja a ”kicsiket”).
- Lokális memória gazdálkodás (laphibázó folyamat lapjai az áldozatok, nagy terhelésingadozások).
- Multiprogramozás előtérbe helyezése, de így csökken az egy folyamatra jutó lapok száma.
  Így nő a laphibák gyakorisága, CPU tétlenség, teljesítménycsökkenés (ún. vergődés) alakul ki!
  Cél az optimum elérése!

Gazdálkodás a fizikai memóriával

- Megállapítható, hogy:
  - min. egy folyamatnak annyi lapra van szüksége, hogy a legbonyolultabb utasítása is le tudjon futni!
  - max. annyi lapra lehet igénye, amennyi a logikai címtartománya.
- Megfogalmazható, hogy:
  - egy program legyen egyensúlyban, azaz célszerűen annyi lapja legyen, amennyire a laphiba-kiszolgálás átlagos ideje alatt hivatkozik.
  - ez az ún. működő lapkészlet (ennek mérése nehézkes, ezért a laphibák gyakoriságát mérik. Az OPR egy AH- és egy FH érték között vizsgálódik.)

Gazdálkodás a fizikai memóriával

- Dinamikus lokális memória gazdálkodás:
  - laphibák kezelése lokálisan,
  - folyamat figyelés és lap átcsoportosítás a folyamatok között.

II. Állományrendszerek

Állományrendszer

Alapfogalmak:

- File (állomány):
  - ”tartós adatok”, név + elérési útvonal, kötet.
- Directory:
  - Katalógus file, gyökér~, ”alkönyvtár”.

Állománykezelő feladatai:

- Információátvitel az állományok és a folyamatok tárterülete között.
- Műveletek állományokon és könyvtárakon.
- Osztott állománykezelés vezérlése.
- Állományokhoz való hozzáférés szabályozása.
- Tárolt információ védelme.

Állományrendszer

- Az állományok kezelését, egymásra épülő rétegek valósítják meg. A file-kezelő rendszer az I/O rendszer szolgáltatásaira épül rá!
- Készülékkezelő réteg (központi tár-periféria).
- Elemi átviteli műveletek réteg (file tartalmának leképzése optimális számú blokkokra {sebesség probléma}).
- Állományszervező réteg:
  - HDD szabad-foglalt területeinek nyilvántartása,
  - blokkok logikai összefűzése,
  - dinamikus helygazdálkodás,
  - állomány-nyilvántartások (pl.: név alapján keres, ”hozzáfértett”).

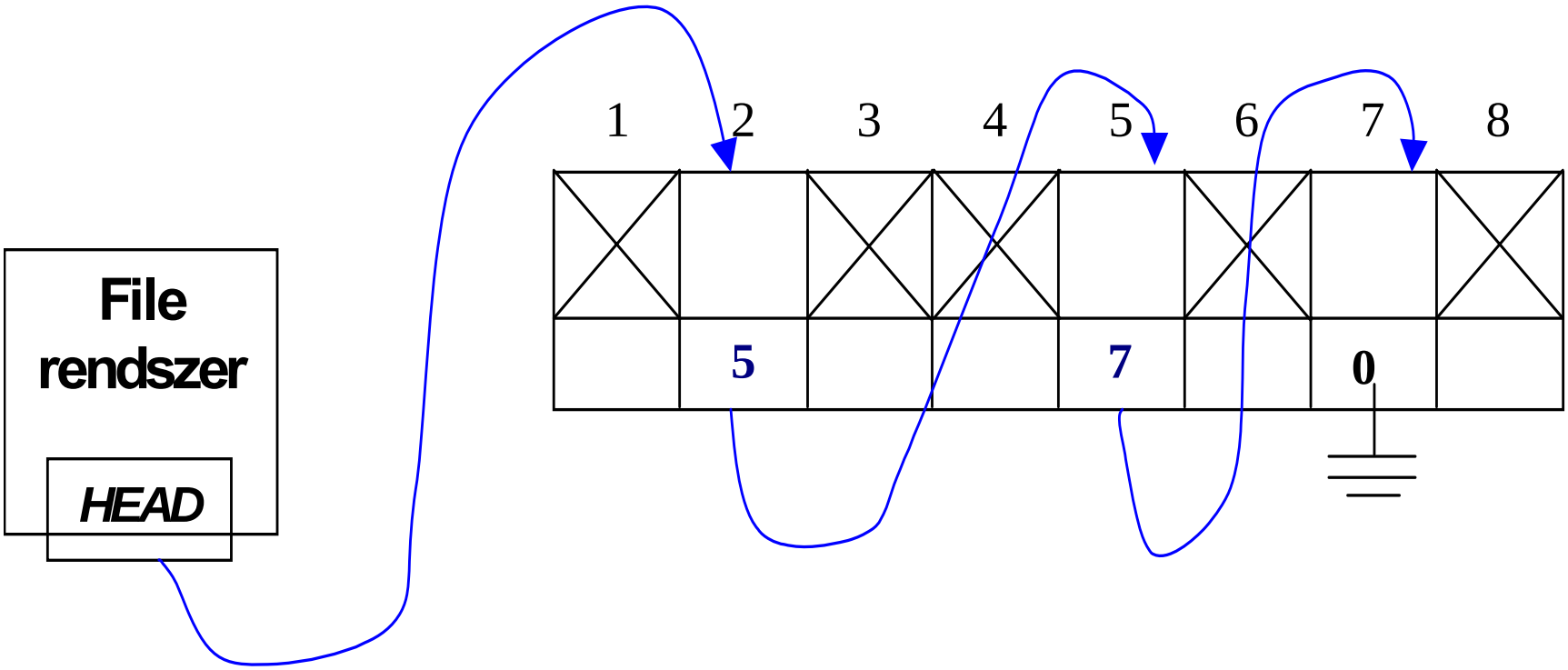
Adatszerkezetek a lemezen

- Adat blokkok. (Logikai - Fizikai)
- Kötet (volume, file-rendszer) leírás.
- Szabad helyek nyilvántartása.
- Állományokhoz tartozó blokkok nyilvántartása.
- Katalógusok ábrázolása.

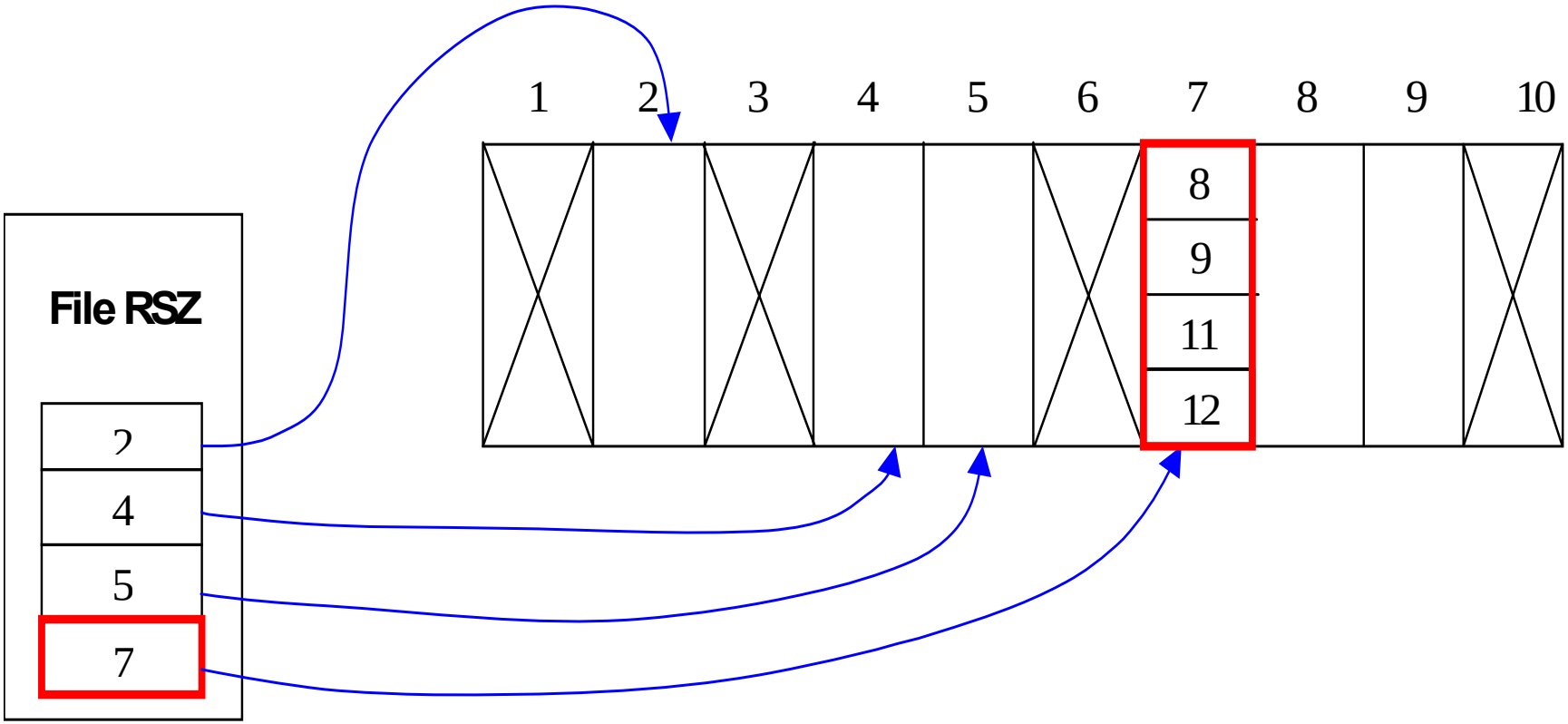
Szabad helyek nyilvántartása

- Bittérkép (logikai blokk - 1 bit - bit vektor, tárolása központi tárban).
- Szabad blokkok láncolt listája (csak a lánc elejét tároljuk majd hivatkozás a következőre, lassú, több lemezművelet).
- Szabad blokkok csoportjának láncolt listája (1 blokkban n-1 címnyi blokk cím, az n.-ben egy újabb m-1 címet tartalmazó blokk cím).
- Egybefüggő szabad területek leírása:
  - táblázatban az 1. szabad blokk címe és hossza,
  - táblázat elemei kezdő blokk szerint rendezettek, akkor az egymás melletti szabad területek össze-vonása egyszerű,
  - átlagos hossz jóval nagyobb mint 1.

Szabad blokkok láncolt listája



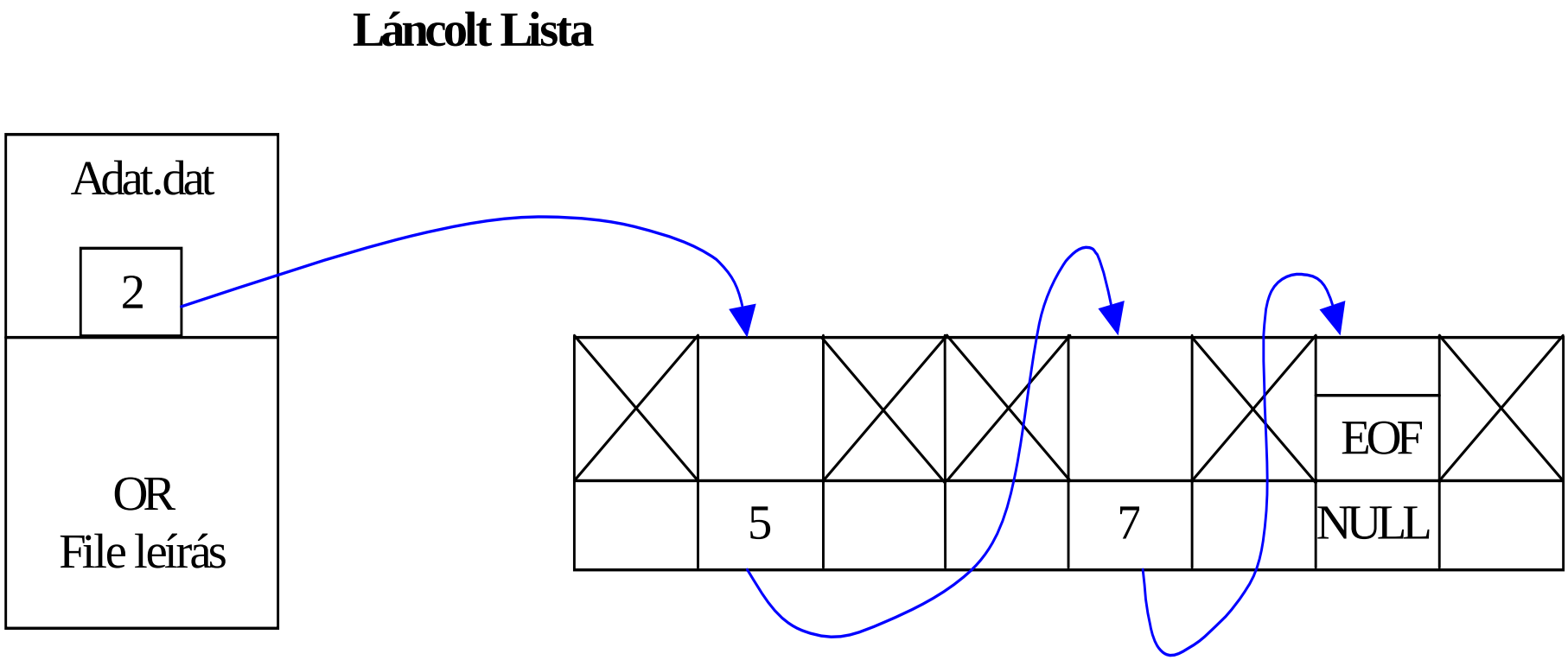
Szabad blokkok csoportjának láncolt listája



Állományokhoz tartozó blokkok nyilvántartása

- Egybefüggő terület használata:
  - állományleíróban: kezdő blokk és blokk szám,
  - egymás melletti szabad blokkok, így ismerni kellene a szükséges blokk számot, külső tördelődés fellépése (*worst-fit* használata),
  - előnyei: tárcsere alkalmával jól használható és az adatok soros és közvetlen elérése egyszerű, gyors.
- Láncolt lista:
  - állományleíróban: kezdő (esetleg utolsó) blokk,
  - dinamikus adatszerkezet, tördelődés mentes,
  - hátrány: a soros elérés, sérülékeny, címek kihagyása a memóriába másoláskor.

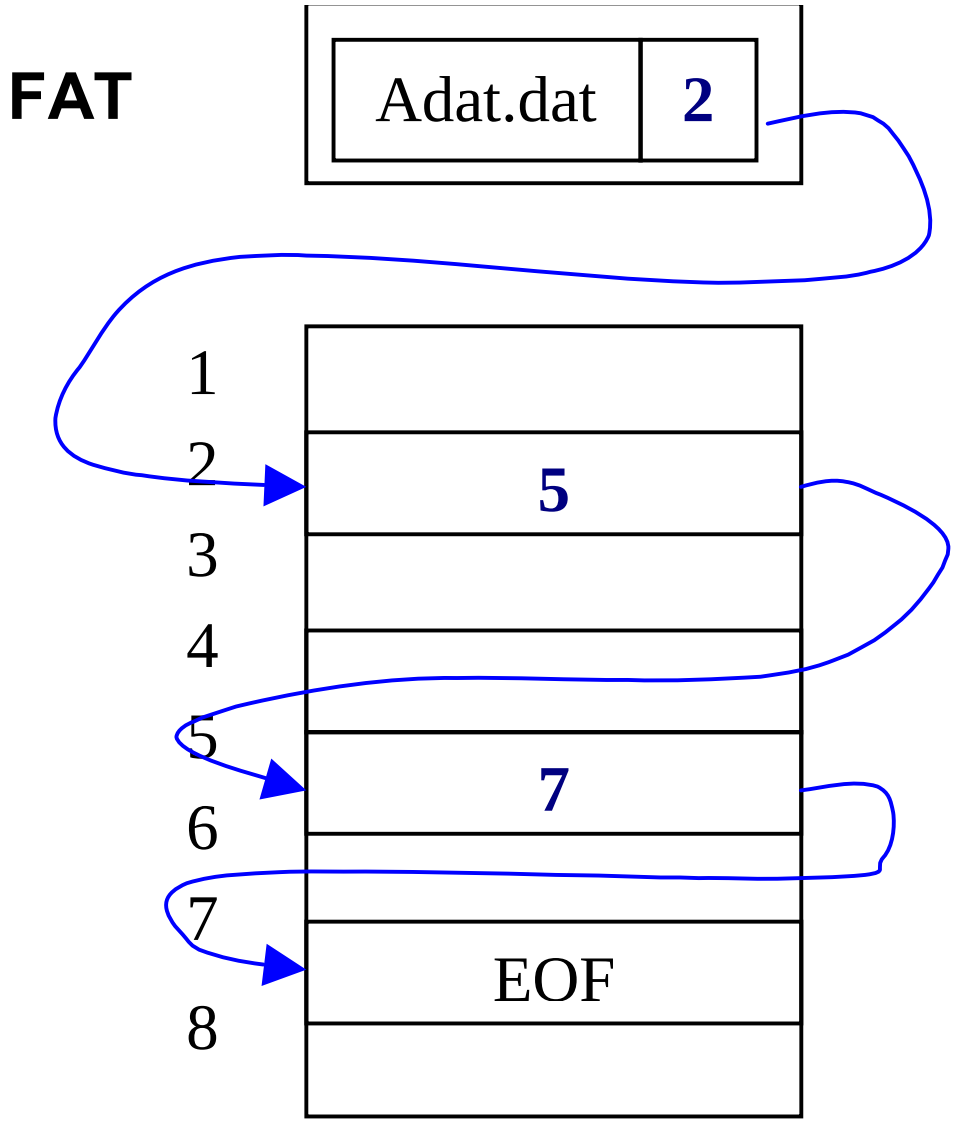
Láncolt lista



Állományokhoz tartozó blokkok nyilvántartása

- Láncolt lista központi láncelem-táblával (File Allocation Table, FAT, kettőzött tárolás):
  - állományleíró: kezdő blokk sorszáma,
  - a lánc külön tárolása,
  - adatok közvetlen elérése (a FAT-ban ”lépkedek”),
  - szabad blokkok tárolása.

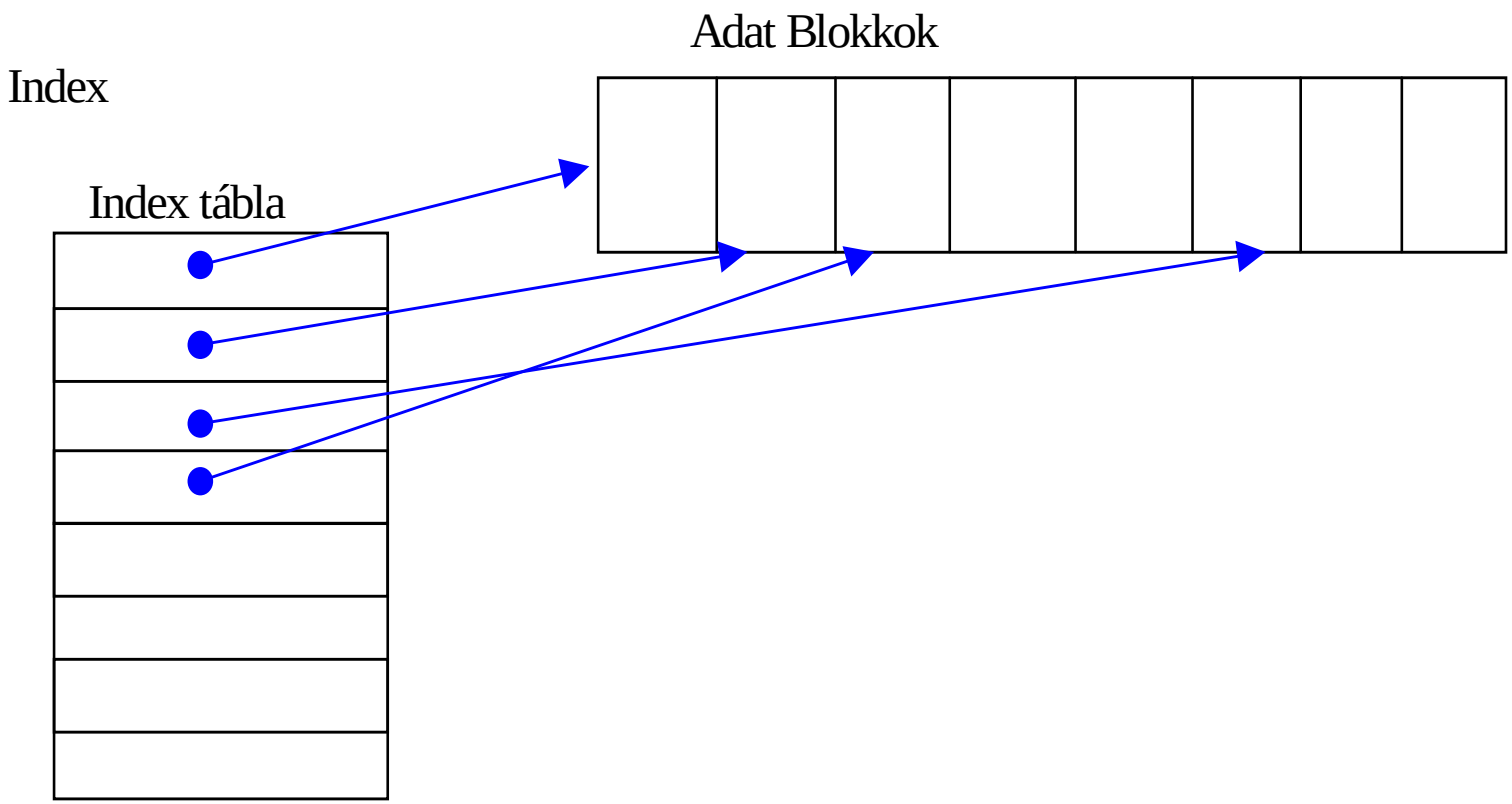
Láncolt lista központi láncelem-táblával, FAT-tel



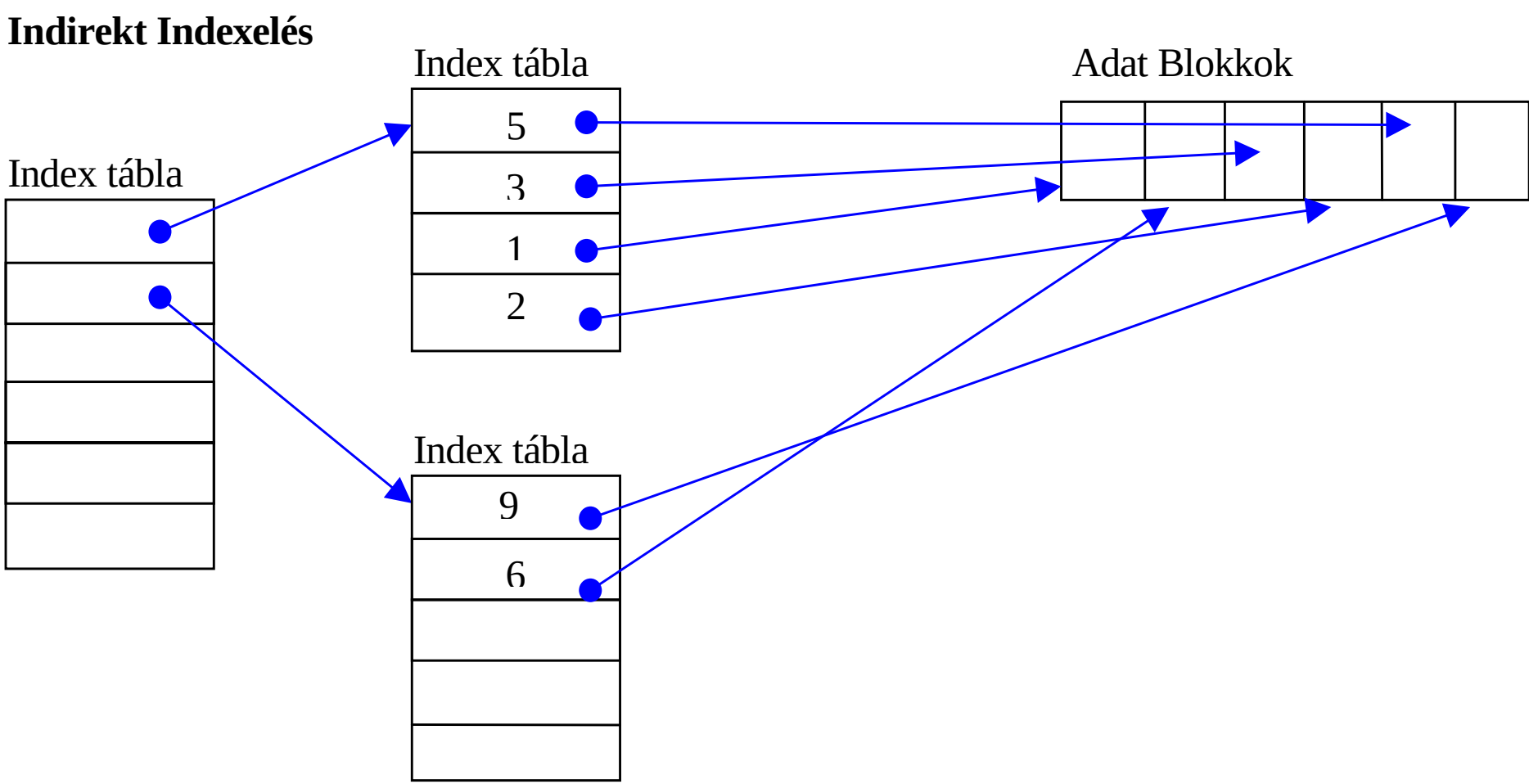
Állományokhoz tartozó blokkok nyilvántartása

- Indexelt tárolás:
  - állományleíró: indextáblát tartalmazó blokk címe,
  - közvetlen elérés,
  - pazarló, a teljes index-blokk lefoglalás miatt (kicsi állományok esetén is!).
- A **láncolt indexblokkok** és a **többszintű indexelés** a táblázat dinamikus növekedése miatt válhat szükségessé.

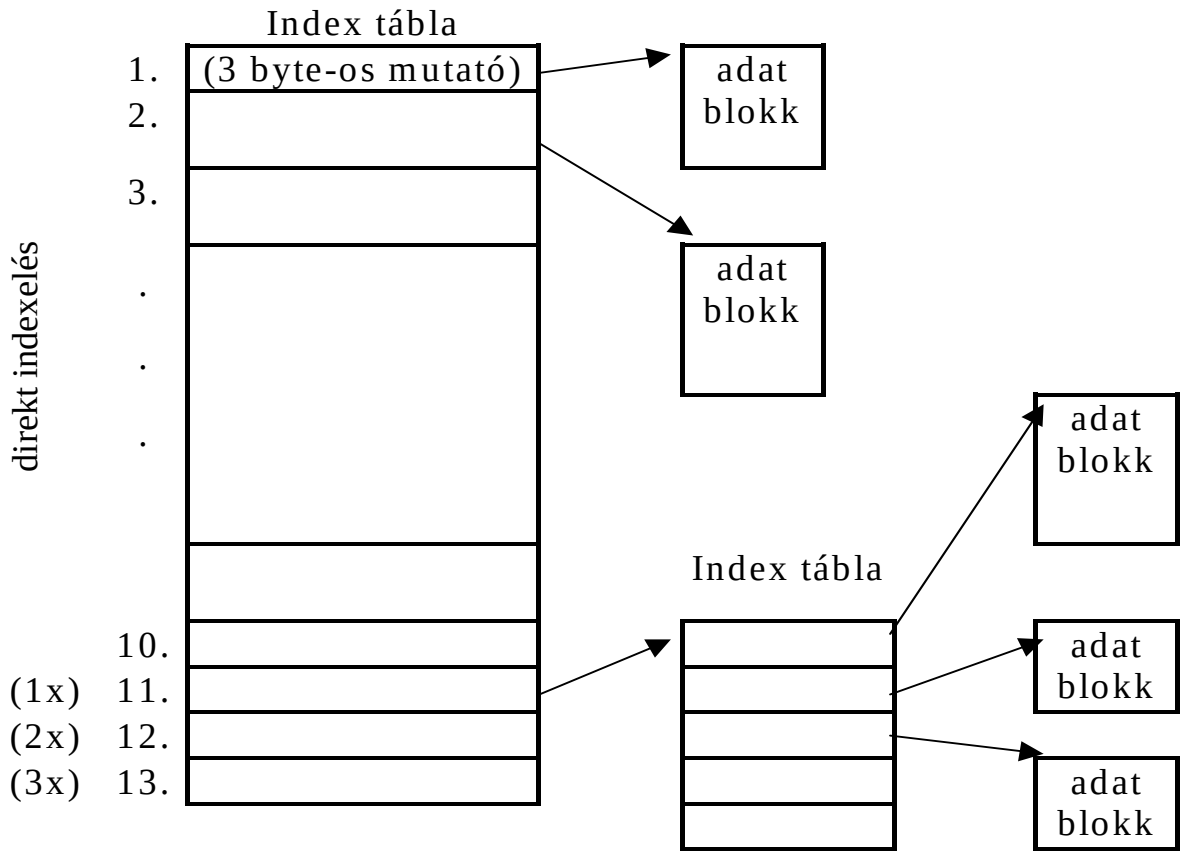
Indexelt tárolás



Indirekt Indexelés



Többszörös indexelés



Állományokhoz tartozó blokkok nyilvántartása

Kombinált módszerek:

- az állomány hozzáférési módja szerint:
  - soros esetén: láncolt,
  - közvetlen esetén: folytonos,
- az állomány mérete szerint:
  - kicsi esetén: folytonos,
  - nagy esetén: indexelt.

Adatveszteségi kérdések.

Állományok belső szerkezete

A legáltalánosabb esetben egy file bit-ek/byte-ok összetartozó sorozata.

A felhasználó szempontjából a logikai felépítése:

- mező: egy-egy adott típusú adat (pl.: int, string,),
- rekord: mezők összetartozó csoportja.

Az OPR szempontjából a felépítése:

- nem foglalkozik a jelentés szerinti adatcsoportosítással,
- csak a folyam manipulálására ad közvetlen eljárásokat (pl.: olvas n byte-ot).

Állomány hozzáférési módok

- Soros (sequential):
  - csak a tárolt byte-ok sorrendjében lehet írni/olvasni,
  - file-pointer, aktuális elérési pozíció tárolása.
- Közvetlen (direct, random access):
  - az információ-elem állományon belüli címével.
  - Indexelt, index-szekvenciális (index sequential access method, ISAM):
    - egy ún. kulcs (kijelölt mező) alapján rendezzük a rekordokat, és egy csatolt indextáblában tároljuk, a rekord, állományon belüli címét.

Műveletek az állományokon

A file egészére vagy részeire vonatkozik.

- Átvitel: írás vagy olvasás:
  - soros (aktuális pozíció növelése és tárolása),
  - közvetlen (az infó címe és pozíciója kell).
- Hozzáírás, bővítés (a végéhez az új infót).
- Pozicionálás (tetszőleges pontra állítás).
- Állomány megnyitása:
  - az állomány megkeresése, mutató generálása, adatszerkezet felépítése (műveletek célpontja),
  - hozzáférési jogosultságok ellenőrzése,
  - az állományon elvégezhető műveletek megadása,
  - osztott állománykezelés szabályozása,
  - file-mutató kezdeti beállítása.

Műveletek az állományokon

- Állomány lezárása.
- Programállomány végrehajtása:
  - az OPR egy új folyamat tárterületére betöltve elindítja.
- Állomány létrehozása (új katalógusbejegyzés, szabad blokk(ok) lefoglalása).
- Állomány törlése (katalógusbejegyzés törlése, blokk(ok) felszabadítása).

Osztott állománykezelés

- Több folyamat esetén a kölcsönös kizárás megvalósítása szükséges. Ez vonatkozhat:
  - a teljes állományra:
    - kizárólagos használat,
    - a többiek csak olvashatják (azonnal vagy csak a lezárás után),
    - többiek is írhatják (lsd. köv.).
  - az állomány részeire:
    - rekord szintű kizárás,
    - holtpont és kiéhezés elkerülése.

Hozzáférés szabályozása

Az állományokat folyamatok hozzák létre, így a jogosultság adója az a felhasználó, aki a folyamatot elindította.

Hozzáférési jogosultságok, állományokhoz, vagy elérési útvonalakhoz tartoznak. Ezeket felhasználónként ill. felhasználó-csoportonként adjuk meg.

Tipikus jogosultságok:

- file-okra (írás – olvasás – végrehajtás – hozzáírás - törlés),
- könyvtárakra (módosítás – listázás - keresés- új file létrehozása - könyvtár törlés).

Könyvtárak leírása

- A nyilvántartás-bejegyzés tartalma:
  - az állomány neve (azonosítója),
  - az állomány fizikai elhelyezkedését leíró információk:
    - hossza,
    - hozzá tartozó háttértár blokkok leírása,
    - a hozzáférés módja.
  - Az állománykezeléssel kapcsolatos logikai információk:
    - típusa (ha van ilyen),
    - tulajdonosának, létrehozójának azonosítója,
    - különböző időpontok, (létrehozás, utolsó módosítás ill. hozzáférés, érvényességi ideje),
    - hozzáférés jogosultságok,
    - hivatkozás számláló (link-ek).

Könyvtárak leírása

- **Állomány kezelési információk**, amelyek nem a háttértáron, hanem a központi memóriában, az OPR saját adatszerkezeteiben kerülnek tárolásra. Ezek:
  - az átvitel állapota (folyamatonként):
    - soros hozzáférés aktuális pozíciója,
    - megengedett műveletek listája,
  - osztott kezelés esetén:
    - mennyi folyamat kezeli egyidejűleg,
    - a kölcsönös kizárást milyem módon, mekkora tartományra kell biztosítani,
    - mely folyamatok várakoznak az állomány felszabadulására.

Könyvtár-rendszerek

- Kétszintű ~, (egyes felhasználók állományai kerülnek egy könyvtárba). Problémák:
  - más felhasználók állomány használata,
  - az OPR állományainak használata.
- Fa gráf, (nem csak file-ok, hanem könyvtárak is).
- Körmentes irányított gráf, (egyes állományokat a gyökértől több útvonalon is el lehet érni, hurkok nélkül). A link-ek lehetnek:
  - fizikaiak, az állományleíró infókat megismétlik,
  - szimbolikusak, csak az állomány elérési útvonalát tároljuk.

Könyvtárak leírása

Problémák:

- több elérési út, de változást okozó esetben, csak egy (pl.: másolás, osztott elérés, törlés).

- Általános gráf, veszélyes lehet a hurok miatt:
  - végtelen ciklus,
  - ág törléssel a hurok a ”levegőben” maradhat.

Ezért ”értelmes” OPR ilyet nem használ.

Könyvtárak hierarchiájának kezelése (user interface)

- Alapfogalmak:
  - Aktuális könyvtár (current directory).
  - Gyökér könyvtár (root directory).
  - Elérési út (path).
  - Keresési út (search path).

Műveletek a könyvtárakon

- Állomány attribútumainak módosítása (logikai információk megváltoztatása).
- Könyvtár létrehozása, törlése (ha üres, vagy minden állományt, vagy rekurzívan az ”alkönyvtárakat” is).
- Keresés (leggyakoribb művelet). Keresni lehet:
  - egy nyilvántartásban (szerkezet - sebesség):
    - rendezetlen keresés, teljes bejárással,
    - rendezett keresés, felező eljárással,
  - a könyvtárszerkezetben:
    - bejárva a megadott elérési útvonalat.

Műveletek a könyvtárakon

- Új bejegyzés létrehozása, törlése:
  - nyilvántartás méretének dinamikus változása,
  - blokkok lefoglalása,
  - rendezettség megtartása,
  - a törölt bejegyzés csak logikailag törlődik.

Mágneslemezes háttértár kezelése

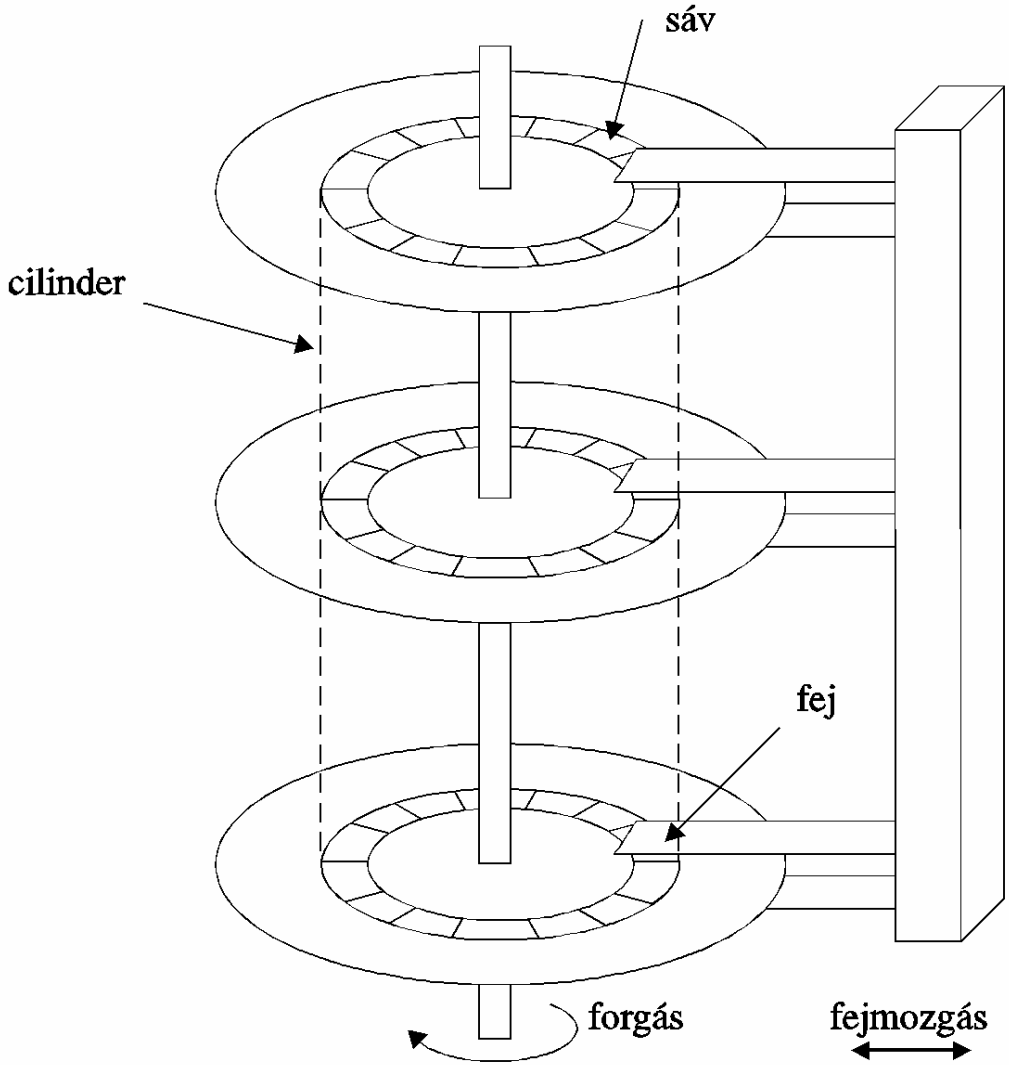
Háttértár tulajdonságai

- Kedvező ár/kapacitás arány.
- Nagy tárolókapacitás.
- Állandó (passzív) tárolás.

Háttértár típusok

- Mágnesszalag.
- Mágnesdob.
- Mágneses merev ill. hajlékony lemez.
- CD-ROM (700 MB).
- EEPROM kártya.
- DVD (4.7-5.4 GB).

Mágneslemez felépítése



A lemezek fizikai szerkezete

- Alapfogalmak:
  - Sáv (track, fej-elmozdulás nélküli, 1 körbefordulás alatt elért terület).
  - Cilinder (a fej adott pozíciójában leolvasható sávok).
  - Szektor (sávon belüli, azonos szögtartományt elfoglaló egyforma méretű rész, amely az infó-átvitel legkisebb egysége. A lemezvezérlő egyszerre olvassa/írja.).
  - Az átvitel kiszolgálásának az ideje áll:
    - fejmozgási idő (seek time, a fej a kívánt sávra áll, {ez a leghosszabb idő }),
    - elfordulási idő (rotation latency time, a kívánt szektor a fej alá fordul),
    - az infó átvitelének az ideje (transfer time).

A lemezek fizikai szerkezete

- Szektor címzés: b = s * (i * t + j) + k. Ahol:
  - b: a szektor 0-tól induló lineáris címe,
  - s: szektor / 1 sáv,
  - i: kijelölt cilinder,
  - t: sáv / 1 cilinder,
  - j: kijelölt fej (avagy felület),
  - k: kijelölt szektor a sávon belül.
- Az i, j és k értékei 0-tól indulnak.

Lemezműveletek ütemezése

Fejmozgás optimalizálása, vagy az elfordulási várakozás csökkentése.

Az algoritmusok értékelésének szempontjai:

- átbocsátó képesség (időegység alatt lebonyolított átvitelek száma),
- átlagos válaszidő (egy átvitel kérésétől a végrehajtásig eltelt átlagidő),
- válaszidő szórása.

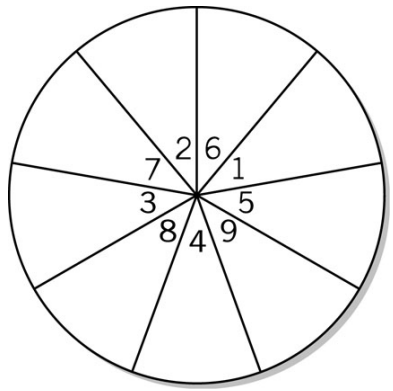
Ütemezési algoritmusok

- A fejmozgás optimalizálása:
  - Sorrendi kiszolgálás, FCFS (az átviteli kéréseket érkezésük sorrendjében szolgáljuk ki, így kicsi az átbocsátó képesség, nagy a válaszidő).
  - Legrövidebb fejmozgási idő, SSTF (az aktuálishoz legközelebbi cilinder hivatkozást szolgálja ki, kiéhezés veszély (új kérések), válaszidők szórása nagy).
  - Pásztázó, SCAN (az aktuális fejmozgású kéréseket szolgálja ki, ha elfogyott akkor vált csak irányt - új kérések kiszolgálása irányváltás után).
  - N lépéses pásztázó, N-SCAN (egy irányba mozogva, csak N - a pásztázás kezdetén meglévő - kérést szolgál ki. Új kérések kiszolgálása irányváltás után).

Ütemezési algoritmusok

- Egyirányú (körforgó) pásztázó, C-SCAN (a kérések kiszolgálása csak az egyik irányú fejmozgásnál, majd a másik irányban a legtávolabbi kérés cilinderére ugrik. Megvalósítható a menetközbeni illetve következő pásztázásra halasztott kérés kiszolgálás.)

- Elfordulási idő optimalizálása:
  - Szektor sorba rendezés (a kiszolgálás előtt, interleaving).



Egyéb gyorsítási lehetőségek

- Lemezterület tömörítése (blokkok fizikailag is egymás mellett, lokalitás).
- Ütemezési algoritmusok sajátosságainak figyelembe vétele:
  - infó többszörözése a lemez különböző területein,
  - középső sávokon való infó elhelyezés.
- Több blokk egyidejű átvitele.
- Átmeneti, gyorsító tár alkalmazása (lokalitás).
- Adattömörítés (perifériakezelő végzi).

Megbízhatóság növelése

- Rendszeres mentés (CD/DVD).
- Átmeneti- és háttértár tartalmának szinkronizálása.
- Lemezegységek többszörözése:
  - tükrözés (az írásokat mindkét lemezen elvégezzük),
  - többszörözés, elosztott tárolás (egy helyett több lemezegység, az infó hibajavító kódolással kerül a lemezekre).

Bemeneti – kimeneti (I/O) rendszer

Az I/O rendszer alapjai

A szervezett, rendezett adatcsere a perifériákkal, az OPR egyik legfontosabb feladata.

A perifériák jellegzetes feladatkörei:

- ember-gép kapcsolattartás,
- gép-gép kapcsolattartás,
- adattárolás.

Az I/O rendszer alapjai

A perifériák jellemzői:

- csatlakoztatási szabvány,
- saját vezérlőegység,
- lassabb működési sebesség a procihoz képest,
- szinkronizáció (a periféria által diktált ütemhez).

Fontos, hogy a perifériák egymással és a procival párhuzamosan is működhessenek!

A perifériák és hardver-illesztéseik általában nagyon eltérőek és sokszínűek.

Az OPR feladatai az I/O kezelésben

Egységes alkalmazói felület (API) kialakítása:

- műveletek egységesítése (periféria = file),
- logikai periféria kezelés (perifériafüggetlen programozás) ,
- műveletek átirányíthatósága.

Egységes csatlakozó felület (DI) kialakítása:

- a vezérlők csatlakozási pontjaira.

Eszközmeghajtók illesztése:

- alapértelmezettek,
- speciálisak (device driver).

Az OPR feladatai az I/O kezelésben

A perifériák hatékony működtetése:

- ütemezés,
- átvitelek párhuzamos lebonyolítása.

Egyszerű adatmozgatás

Adatblokk mozgatás a memória és egy periféria között (alacsony szintű I/O kezelés):

- indító (START) művelet,
- várakozó/tesztelő (WAIT) művelet,
- hiba leállító (STOP) művelet,
- paraméterezési adatstruktúra (IOCB).

Az adatmozgatás paraméterei:

- forrás és cél: az egyik egy memória cím, a másik egy logikai periféria név.
- mennyiség: átviendő byte szám.

Egyszerű adatmozgatás

A műveletek argumentuma:

- logikai periféria neve,
- egy mutató az IOCB-re.

Az IOCB tartalma:

- parancs (pl.: írás, olvasás),
- állapotjelző (átvitel OK ill. ERROR),
- memóriacím,
- mennyiség,
- egyéb (pl.: HDD esetén blokkcím).

A folyamat

START:

- logikai perifériához fizikait rendel,
- a fizikai várakozási sorába fűzi az IOCB-t,
- ha a periféria szabad, hívás a perifériakezelő *Start_Device* műveletének,
- ha szabad, ha nem, visszatér. Azaz nem várja meg az I/O művelet végét (aszinkron hívás).

A folyamat

WAIT:

- IOCB állapotjelzőjének ellenőrzése,
- ha az I/O művelet még nem fejeződött be akkor, a hívó folyamatleíróját ráfűzi az adott IOCB-re, és várakozó állapotba helyezi, majd CPU ütemezést indít,
- ha az I/O művelet befejeződött, akkor visszatér a hívóhoz.

A folyamat

STOP:

- ha a készülék még nem kezdte meg a végrehajtást, akkor kifűzi az IOCB-t,
- ha már megkezdte, akkor meghívásra kerül a perifériakezelő *Stop_Device* eljárása.

Az egyszerű I/O műveletek lehetővé teszik a blokkolt (szinkron) vagy a nem blokkolt (aszinkron) hívást.

A szinkron mód a ”kedveltebb”, mert ilyenkor a vezérlési szál nem ágazik el. Így a START után rögtön WAIT jön, így a folyamat várakozási állapotba kerül.

File-műveletek

Az eszközökre a szekvenciális file-modell jól illeszthető, hiszen az adatáramlás byte jellegű.

Így a műveletek (read, write, seek) egy, a megnyitáskor kapott azonosítóval elvégezhetőek. Ezek vagy blokkonkénti vagy karakterenkénti I/O-rendszerhívást okoznak. Értelemszerűen, egyes eszközökre csak olvasás (pl.: billentyűzet), másokra csak írás (pl.: nyomtató) értelmezhető.

Grafikus eszközök kezelése

- Általában egy grafikus könyvtár eljárásaival.
- Ezek; raszter- vagy vektorgrafikai modellt valósítanak meg.
- Grafikus kártyán elhelyezett célproci és memória, felgyorsíthatja a műveletek végrehajtását.

Terminál kezelés

- Egy karakteres I/O egység, ami képernyőt és billentyűzetet tartalmaz, távol magától a géptől.
- Adatátviteli vonal vagy hálózat kell.
- Karakterenkénti adat áramlás, vezérlőkarakterekkel, azaz kódrendszert használ.

Hálózati kapcsolatok kezelése

- Logikai csatlakozók (*socket*) bevezetése.
- Az alkalmazások létrehozhatnak ilyeneket és becsatlakozhatnak távoli címekre, majd figyelik, hogy más rákapcsolódott-e.
- A kapcsolat létrejötte után indirekt kommunikációs objektumként működik.
- Egyéb ilyen kommunikációs modellek pl. a postaláda, üzenetsor.

A kernel I/O alrendszere

Feladatai:

- perifériák ütemezése (átviteli kérelmek sorbaállítása, a hatékonyság miatt),
- átmeneti- és gyorsítótár működtetése:
  - két eltérő sebességű adatfolyam összekapcsolásakor,
  - a periféria számára blokkosítani kell, ám az adott alkalmazás nem blokkokkal vagy más méretűekkel dolgozik,
  - az átmenetitáron: az egyetlen létező kópia van,
  - a gyorsítótáron: a HDD-n lévő adat egy másolata van.

A kernel I/O alrendszere

- perifériahasználat koordinálása:
  - az egyszerű I/O műveletek sorbaállítása,
  - egyetlen I/O művelettel el nem végezhető oszthatatlan műveletek (pl. nyomtató, HDD) hatékony kezelése (specko puffer, ún. spool file).
- hibakezelés:
  - az OPR vagy a kezelőprogramok észlelik,
  - kijavítása művelet vagy részművelet ismétléssel,
  - hiba jelzése a hívó felé.