Előadás_#08.

1. Állományrendszerek

   Rövid hardver áttekintés a lemez alapú háttértárakról:

   FDD 18 szektor, 80 sáv, 2 oldal, 1-1 fej

   HDD nagyszámú szektor és sáv, ZBR, lemezenként 1 oldal és 1 fej

   ODD spirál vagy ZBR, lemezenként 1-2 oldal, 1-2 réteg és 1 fej

   Leggyakoribb fájlrendszerek:

   - FAT, FAT12, FAT16, FAT32, exFAT

     FAT: File Allocation Table / Fájl Allokációs Tábla

     Az egyik legelső (a kódot 1977-ben készítette Marc McDonald), fix méretű blokkokból álló, láncolt lista szerkezetű fájlrendszer, az informatika nagy túlélője. Egyes változatai még ma is használatosak a FDD és HDD technológiákban.

     A FAT16 és a FAT32 összehasonlítása (Microsoft, angol nyelvű)

     Az exFAT és FAT32 összehasonlítása (NTFS.com, angol nyelvű)

   - HPFS

     HPFS: High Performace File System / Nagyteljesítményű Fájl Rendszer

     Az IBM által kifejlesztett rendszer, amely kb. 200MB lemezkapacitás felett mutat érdemi előnyöket, pl. évekig nem kell töredezettség mentesíteni (B+FA stuktúra), a főkönyvtár fizikailag nem a lemez elején, hanem a közepén helyezkedik el. Sajnos a Microsoft a HPFS-t csak rövid ideig támogatta, végül folyamatosan kiszorította a támogatott fájlrendszerek közül.

     A FAT, a HPFS és az NTFS fájlrendszer áttekintése (Microsoft)

   - NTFS

     NTFS: New Technology File System / Új Technológiájú Fájlrendszer

     Ezzel egy hamarosan következő előadás részletesen foglakozik.

   - UFS

     UFS: Unix File System / Unix Fájl Rendszer. Első verzióját 1984-ben fejlesztették ki a Berkley egyetemen. Paraméterei minden szempontból jobbak mint az akkori FAT rendszereké. Egyedi tulajdonsága például a szuperblokk, ami egy úgynevezett "magic number"-t (és még jó pár paramétert) tartalmazott a használt lemez geometriájával, statisztikáival és finombeállításával kapcsolatban. Arról sem szabad elfeledkezni, hogy a Unix mindet (beleértve a fizikai eszközöket, perifériákat is) fájlként kezel.

- NFS

  NFS: Network File System / Hálózati Fájlrendszer.

  A Sun Microsystem fejlesztése, ami hálózatos (pontosabban TCP/IP feletti) rendszerek között biztosít transzparens állomány elérést, első verziója 1984-ben jelent meg. Valójában nem fizikai fájlrendszer hanem egy, a megosztást biztosító szoftverréteg. Működése szerver-kliens alapú, és mára elmondható, hogy minden elosztott fájlrendszer alapja az NFS.
- VFS

  VFS: Virtual File System / Virtuális Fájlrendszer

  Valójában nem fizikai fájlrendszer hanem egy új absztrakciós réteget biztosító szoftveres virtualizációs megoldás, első verziója 1985-ben jelent meg. A VFS tulajdonképpen egy interfész a kernel és a konkrét fizikai fájlrendszer között, melynek segítségével egy virtuális gépen futó alkalmazások anélkül érik el a fizikai tárolókat, hogy azoknak a különböző fizikai fájlrendszereit ismernék.
- JFS

  JFS: Journal File System / Naplózó Fájlrendszer

  A JFS az IBM fejlesztése a HPFS továbbfejlesztésekét, első verziója 1990-ben jelent meg. A JFS a naplózó technológia nyílt forráskódú alapja. A naplózás segítségével akár a rendszerösszeomlás, illetve az áramkimaradás okozta adat konzisztencia problémák is szinte minden esetben automatikusan javíthatók.
- EXT, EXT2, EXT3, EXT4

  EXT: Extended File System / Kiterjesztett Fájlrendszer

  Az első EXT fájlrendszert 1992 áprilisában írta Rémy Card a Linux kerneléhez. A Linux kernelben már a 0.96c verzió óta meglévő VFS első érdemi, és széles körben elterjedt felhasználása az EXT fájlrendszer. Az EXT3 óta a JFS-ben megismert naplózási képesség is a fájlrendszer része.
- LTFS

  LTFS: Linear Tape File System / Szalagos Fájlrendszer

  Az LTO rendszerű mágnesszalagok jellemzően archiválásra használt fájlrendszere, amely a HDD-hez hasonló adatszerkezetet és adatkezelést biztosít. Első verziója 2000-ben jelent meg.
- CDFS, UDF

  CDFS: Compact Disc File System / Kompakt Lemez File Rendszer

  UDF: Universal Disc Format / Univerzális Diszk Formátum

  Az optikai lemezek lezárt illetve nem lezárt fájl rendszerei, lemezformátumai.

  Milyen CD- vagy DVD-formátumot érdemes használni? (Microsoft)

Fájl (File) – magyar megfelelője az állomány – alatt az összetartozó információk gyűjteményét értjük. Tartalma lehet program, adat, szöveg, ábra, vagy bármely kódolt információ. A fájlok perifériákon tárolódnak az operációs rendszer által használt fájlrendszernek megfelelően, jellemzően könyvtár struktúrában. A felhasználó egy fájlra a fájl egyedi azonosítójának (elérési út, név) segítségével hivatkozhat. Egy fálj használatához a felhasználónak - ha az elérési utat és a fáljnevet ismeri – nem szükséges ismernie a fájl tárolásának fizikai paramétereit: azaz hogy melyik fizikai eszközön, illetve az eszközön belül pontosan hol található a fájl; az eszköz milyen csatornán illeszkedik a rendszerhez; mi a tárolás adategysége.

A fájlrendszer tehát egy olyan szoftveres réteg, ami azt biztosítja, hogy a felhasználó a fizikai tárolót megfelelő jogosultsággal elérhesse, az abban használatos könyvtár struktúrában adatokat tudjon tárolni, illetve a tárolt adatokat vissza tudja nyerni. A fájlok a felhasználó egyéni céljai szerint, illetve az operációs rendszer szempontjai szerint könyvtárakba (Directory) csoportosíthatók. A könyvtárak és fájlok csoportosításának következő lépcsője a kötet (Volume).
A fájlkezelő (állomány kezelő) rendszer a következő feladatokat kell hogy ellássa:

- adatcsere, információátvitel a fájlok és a folyamatok tárterülete között
- műveletek fájlokon, könyvtárakon és köteteken
- osztott fájlkezelés, azaz a fájlok akár több folyamat egyidejű rendelkezésére is lehetnek bocsájtva
- a hozzáférések, jogosultságok szabályzása
- a tárolt információ illetéktelen hozzáférés elleni védelme, titkosítása
- a tárolt információ sérülések elleni védelme

A fájlkezelő rendszer az operációs rendszer I/O (be- és kimeneti) rendszerének szolgáltatásaira épül, gyakorlati megvalósítása további egymásra épülő szoftver rétegek segítségével történik. Az adatok az egyes rétegek szolgáltatásain keresztül, rétegenként kisebb-nagyobb átalakítások és konverziók során jutnak el a háttértárból az operatív tárba illetve az operatív tárból a háttértárba.

A fájlkezelő rendszer rétegei (a legalsó rétegtől elindulva):

- Készülék kezelő (Device Driver) réteg feladatai:
  - közvetlenül a tárolóeszköz (háttértár) vezérlése
  - az operatív tár és a háttértár közötti adatátvitel lebonyolítása
  - jellemző a megszakítások (IRQ) illetve a közvetlen memóriaelérés (DMA) használata
- Elemi átviteli műveletek réteg feladatai:
  - egy adott fájl leképzése optimális számú és méretű blokkra
  - konverzió a központi tár illetve a háttértár címzési rendszere között
  - az operatív tár és a háttértár sebesség különbségének kezelése gyorsítótárak (cache) igénybevételével
- Állományszervező réteg feladatai:
  - fizikai és logikai blokkok, fájlok, könyvtárak nyilvántartása
  - szabad területek nyilvántartása
  - blokkok logikai összefűzése
  - dinamikus helygazdálkodás
  - új fájlok létrehozásakor, vagy meglévő fájlok méretének növelésekor szükséges adminisztráció elvégzése, szabad hely biztosítása
  - fájl törlése után a hely felszabadítása
  - fálj elérés időbeli ütemezése, szinkronizálása, a hozzáférés szabályozása
  - jogosultságok kezelése

Az operációs rendszer által egységként kezelt lemezterület a blokk. A blokk jellemzően néhány szektort (ZBR szervezésű tárolás esetén természetesen zónát) tartalmaz. A fájlok jellemzően több blokkból állnak.
A szektorméret a tárolóeszköz fizikai paramétere, az viszont, hogy egy blokk hány szektort tartalmaz az paraméterezhető. A blokkméret meghatározását szervezési, tárolási és hatékonysági szempontok befolyásolják. Minél nagyobb blokkokat használunk, annál kevesebb az egységnyi információ átviteléhez szükséges többlet művelet illetve idő. Másrészt viszont, ha a nagyméretű blokkokat az információ nem tölti ki teljesen, akkor feleslegesen foglalunk területet mind a háttértárban, mind az operatív tárban, ami belső tördelődési veszteséghez vezet.

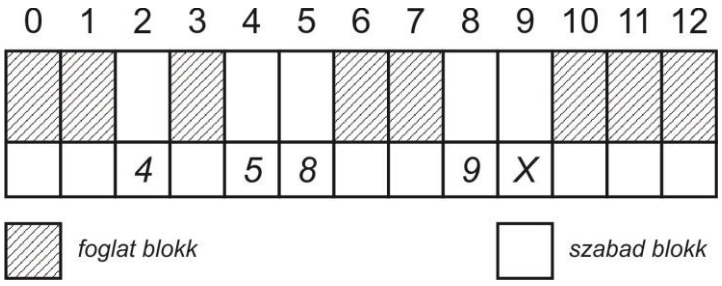
A szabad helyek nyilvántartása (ami a fentiek szerint alapvetően az állományszervező réteg egyik fontos feladata) többféleképpen is megvalósítható:

- Bittérképes ábrázolással

  Ebben az esetben a háttértár (jellemzően merevlemez) minden egyes logikai blokkjához egy bit tárolja, hogy az adott blokk szabad-e. Ezen a bitek segítségével foglaltsági vektor állítható elő, ami a lemez egy erre fenntartott helyén tárolódik. Ezzel a tárolási formával az egymás melletti szabad blokkok kiválasztása egyszerű (de sajnos csak ez). A hatékonyságot és a gyorsaságot növelendő, célszerű az egész vektort az operatív tárba letükrözni.

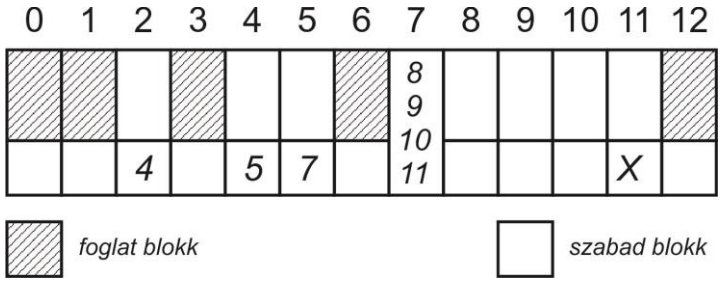
- A szabad blokkok láncolt listájával

  Ebben az esetben minden egyes szabad blokk utolsó néhány bájtját kiemeljük a klasszikus adattárolási funkcióból, ugyanis ezen a néhány bájton a következő szabad blokk sorszámát tároljuk. A módszer annak ellenére sem túlzottan hatékony, hogy a lista bejárásához elegendő az első szabad blokk ismerete, hiszen a lista bejárása lassú, sok lemezműveletet igényel.



- A szabad blokkok csoportjának láncolt listájával

  Ebben az esetben egy szabad blokk jellemzően nem csak egy, a soron következő blokk címét tartalmazhatja, hanem „n” darab blokk címét, melyek közül „n-1” darab ténylegesen szabad. A címgyűjtemény listáját tartalmazó blokk technikailag foglalt (de az operációs rendszer szempontjából természetesen szabad). Azaz ciklikusan egy teljes blokkot a nyilvántartás céljaira foglalunk le. Teljesítménye, hatékonysága jobb, mint a szabad blokkok láncolt listájának hasonló paraméterei.



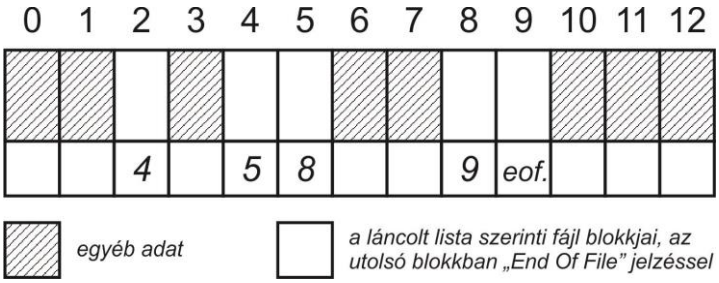
- Az egybefüggő szabad területek leírásával
  Ebben az esetben a szabad blokkokra nem egyesével hivatkozunk. A tárolás lényege, hogy az egybefüggő terület első blokkjának címe mellett a terület hosszát tároljuk el. Ezen tárolási forma nyilván akkor hatékonyabb, ha a szabad területek átlagos hossza minél nagyobb.

A szabad helyek nyilvántartása mellett nyilván fontos feladat a fájlok tárolásának a megoldása, azaz a logikailag összefüggő foglalt helyek és azok sorrendjének nyilvántartása. Ez a nyilvántartás is többféle módon valósítható meg.
Allokációs módszerek:

- Folytonos terület használata
  Ebben az esetben az összetartozó információk csak egymást követő blokkokban tárolódhatnak. A nyilvántartáshoz elegendő ismerni az első blokk sorszámát, valamint a tároláshoz használt blokkok darabszámát.
  - Előnyei:
    Ez a módszer akkor célravezető, ha a szabad területek átlagos hossza jelentős számú blokkból áll. A tárolt információ elérése egyszerű, az összetartozó adatok sorban követik egymást. Ugyan a rendszer szemlátomást rugalmatlan, de például tárcsere esetén – mivel ismerjük az elmentendő címtartomány méretét, és a blokkok sorban követik egymást, így azok egyszerre, egy lépésben mozgathatók – hatékony és gyors megoldást biztosít.
  - Hátrányai:
    A külső tördelődés a módszer egyik kellemetlen velejárója, aminek a hatásait a változó méretű partícióknál megismert algoritmusokkal (Best Fit, First Fit, Next Fit, Worst Fit) lehet ellensúlyozni. Amennyiben a külső tördelődés közvetlenül kezelhetetlen szintet ér el, a megoldás az állományok átmásolása egy másik háttértárra, ezután a terület teljes felszabadítása, végül az állományok új struktúrában történő visszamásolása.
    További hátrány az, hogy a gyakorlatban a szükséges blokkok száma csak megbecsülhető, de pontosan előre nem ismert. Amennyiben a lefoglalt terület kicsinek bizonyul, és közvetlenül a terület után nem található szabad blokk, akkor az operációs rendszer kénytelen az egész területet egy nagyobb területre átmozgatni. Amennyiben ez nem lehetséges (és a tár más módon sem szervezhető át belátható időn belül), akkor legrosszabb esetben a programunk futása le fog állni.

- Láncolt lista

  Ebben az esetben (a szabad területek nyilvántartásánál már megismert módon) a tárolás szekvenciális és mindig a sor elejéről indul, oly módon, hogy maga a blokk tartalmazza a következő blokk számát, ami hasznos területet vesz el a blokkból.



  A nyilvántartáshoz a fájlok első és utolsó blokkjának sorszámát elegendő tárolni.

  - Előnyei:

    Ebben az esetben külső tördelődés nem lép fel. Az adatszerkezet dinamikus, a fájlméret, azaz a felhasznált blokkok száma tetszőlegesen (egyéb, a fájlrendszer szabta korlátok ismeretében) növelhető.

  - Hátrányai:

    A szekvenciális tárolásból fakadóan egy fájl eléréséhez a teljes listát lépésről lépésre végig kell járni. A blokkok tartalma két részre van osztva (adat illetve a következő blokk címe), ennek adminisztrálása erőforrást igényel. Egyetlen blokk meghibásodása a tárolt adatok részleges vagy teljes elvesztését okozhatja.

- Láncolt lista központi láncelem táblával

  Fájl allokációs tábla (File Allocation Table / FAT)

  Ez a módszer a láncolt tárolás olyan megvalósítása, ahol a fájlok adatait és a láncelemeket (a következő blokk sorszámát) elkülönítve tároljuk. A láncokat az úgynevezett fájl allokációs tábla tárolja. A táblában a tároló (lemez) minden egyes blokkjához tartozik egy bejegyzés (pointer), ami abban az esetben, ha a szóban forgó blokk egy fájlhoz tartozik a fájl következő blokkjára mutat. Az állományleíróban csak az egyes fájlok első blokkjának sorszámát kell tárolni, hiszen a többi sorszám a FAT-ban tárolódik.

  Az ábrán egyrészt az látszik, hogy az első két bejegyzés (000, 001) szabvány szerint nincs használatban. Az itt

FAT tábla		
blokk	foglalt	köv.
000	0	-
001	0	-
002	1	003
003	1	005
004	0	000
005	1	006
006	1	011
007	1	009
008	1	007
009	1	eof.
010	0	000
011	1	008

ábrázolt fájl első eleme a 002-es blokkban található (ez az infó kell, hogy az állományleíróban szerepeljen), a többi blokk sorrendje pedig: 003, 005, 006, 011, 008, 007 és végül az utolsó a 009.

A FAT tábla a foglalt és összetartozó, valamint a szabad blokkok tárolására egyaránt alkalmas. A FAT tábla jellemzően a tároló (lemez) elején helyezkedik el, ráadásul – adatbiztonsági megfontolások miatt – két példányban, esetenként jelentékeny tárolókapacitást elfoglalva.

- Előnyei:

  Az adatvesztés valószínűsége jelentősen csökkenthető, hiszen maga a FAT tábla két példányban tárolódik. A tárolt információ elérése egyszerűbb és gyorsabb, hiszen elegendő a FAT táblában keresni, nincs szükség a teljes adatblokkok beolvasására a kereséshez. FAT tábla memóriában történő tárolásával a keresés tovább gyorsítható.

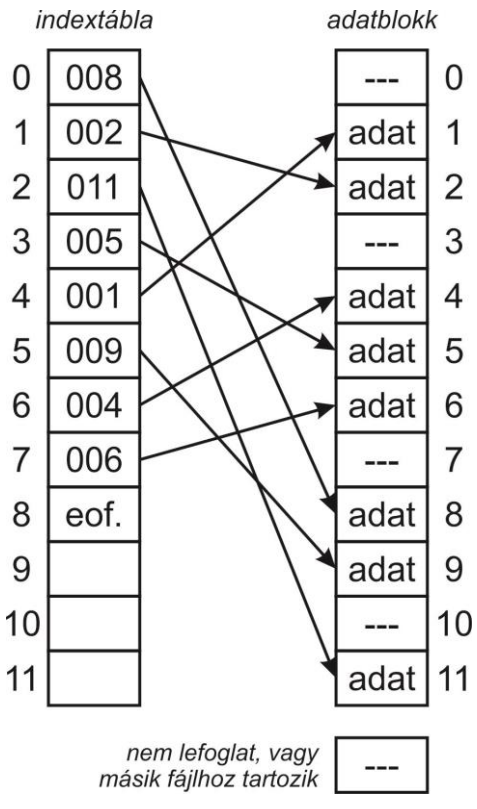
- Hátrányai:

  Maga a FAT tábla nyilvánvaló hogy területet hasít ki magának a rendelkezésre álló tároló kapacitásból. Mivel a FAT szekvenciális szerkezetű, a tárolókapacitás növekedésével a FAT tábla (illetve a FAT tábla második példánya) számára lefoglalt hely szintén növekszik, nyilván az adattárolásra felhasználható tárolókapacitás rovására.

- Indexelt tárolás

  Ebben az esetben az egy fájlhoz tartozó blokkok tárolása egy erre a célra fenntartott külön adatterületen, egy úgynevezett indextáblában történik. Egy fájl eléréséhez elég az állományleíróban eltárolt (első) indextáblát tartalmazó blokk címére hivatkozni.

  Az itt ábrázolt fájl első eleme a 008-as blokkban található, a többi blokk sorrendje pedig az indextáblában haladva: 008, 002, 011, 005, 001, 009, 004 és végül az utolsó a 006.



  - Előnyei:

    Mivel az indextábla segítségével minden blokk közvetlenül megtalálható, maga a közvetlen hozzáférés metodikája is egyszerű.

Ráadásul lehetőség van olyan állományok tárolására is amelyeknek nem minden blokkja tartalmaz információt, azaz az állományon belül üres blokkok fordulnak elő. Ez esetben nem szükséges ezen üres blokkokhoz fizikailag is lemez blokkot rendelni, elegendő ezeknek a helyét az indextáblában üresen hagyni.

- Hátrányai:

  Viszonylag pazarló helygazdálkodás jellemzi, hiszen az egy fájlhoz tartozó blokkok címeinek tárolásához akkor is legalább egy teljes indexblokkot kell lefoglalni, ha az állomány kis méretű és így csak kevés blokkot tartalmaz. Az indextábla mérete nem határozható meg előre, ezért mindenképpen biztosítani kell azt, hogy maga az indextábla mérete dinamikusan követhesse az igényeket. Ez a szempont megoldható az indexblokkok láncba fűzésével (például a szabad helyek csoportjainak tárolásához használt módon), vagy több szintű indextáblák használatával. Nyilván a két módszer kombinálva is használható, a kis fájlok esetében jellemzően egyszintű, a nagy fájlok esetében jellemzően többszintű indextáblák segítségével.

- Kombinált módszerek

  Az optimális tárhely gazdálkodás jellemzően egy, kitüntetett módszer segítségével nem valósítható meg, hiszen a fájlok mérete, viselkedése, címzési módja több szempontból is eltérő. Az operációs rendszernek minden felmerülő esetet (közel) optimálisan és minimális tárolókapacitás veszteséggel kell tudnia kezelni.

  - A hozzáférés módja szerint

    A célszerűség a gyakorlati tapasztalatok alapján ez esetben az, hogy a soros hozzáférésű fájlok esetében a láncolt, míg a közvetlen hozzáférésű fájlok esetében az folytonos listák használata az optimális választás.

  - A fájlok mérete szerint

    A célszerűség a gyakorlati tapasztalatok alapján ez esetben az, hogy a kis méretű fájlokat folytonosan, a nagy méretű fájlokat pedig indexelt módon tároljuk és kezeljük.

Az állományokat a belső szerkezetük illetve a hozzáférési módok alapján is csoportosíthatjuk.

Tárolási szempontból egy fájl tulajdonképpen egy bitsorozat, pontosabban bájtsorozat, hiszen a jellemző fájlmanipulációs műveletek bájtonkénti és nem a

bitenkénti szervezésűek. A fájlok belső szerkezete főleg a fálj típusától. funkciójától függ. A futtatható fájlok és az adatfájlok szerkezetüket tekintve eltérőek. A futtatható fájlok utasításokat tartalmaznak, az adatfájlok pedig adatokat.
(Nyilván a tartalom a valóságban ennyire nem egyszerűsíthető le, hiszen például egy *.EXE állomány a végrehajtandó program utasításai mellett számos adatot és grafikus információt is tartalmazhat. A fájlok belső szerkezetének mostani, alapszintű tárgyalásakor csak a numerikus adatokat tartalmazó adatfájlokra helyezzük a hangsúlyt.)
A felhasználó a tárolandó adatokat a fájlokon belül önálló egységekbe csoportosíthatja. Az adatfájlok belső szerkezetük szerint tartalmazhatnak adatmezőket (Field), valamint az adatmezők összetartozó csoportjait, melyeket rekord-nak (Record) nevezünk. A mezők illetve a rekordok hossza lehet fix illetve változó is.
Az operációs rendszereknek jellemzően nem célja a fájlok belső szerkezetének közvetlen ismerete és annak közvetlen értelmezése. A gyakorlatban a különböző felépítésű (a fájl kiterjesztése jellemzően utal a fájl funkciójára) fájlok kezelése, manipulálása megfelelő cél-szoftverrel valósul meg. Egy-egy ilyen cél-szoftvert csak speciális esetben érdemes beleintegrálni az operációs rendszerbe, hiszen ez egyrészt növeli az operációs rendszer helyigényét, másrészt fix funkció lévén csökkenti az operációs rendszer rugalmasságát.

Egy fájl (állomány) a hozzáférési mód, az elérés szerint lehet:

- Soros azaz szekvenciális
  Ebben az esetben az adatokat csak a tárolt bájtok sorrendjében lehet olvasni illetve írni. Szükség van egy mutatóra (File Pointer) ami az aktuális elérési pozíciót tárolja.
- Közvetlen (Direct) azaz tetszőleges (Random)
  Ebben az esetben az elérni kívánt információ-elem, az állományon belüli címének (sorszámának) ismeretében, közvetlenül elérhető.
- Indexelt, index-szekvenciális (Index Sequental Access Method / ISAM)
  Ebben az esetben a rekordok közül indexállományok segítségével igény (tartalom) szerint tudunk választani. Az ehhez szükséges kulcsok egy-egy index állományban rendezett módon vannak tárolva.
- Partícionált
  Ebben az esetben egy fájlt soros rész-állományok alkotják, és maga a fájl tartalmaz egy nyilvántartást arról, hogy ezek a partíciók hol helyezkednek el a fájlban.

A fájlokon végrehajtandó műveletek – a fentiek szerint – vonatkozhatnak a fájl egészére, vagy csak a fájl egy részére. A legáltalánosabb fájl műveletek a következőek:

- Attribútumok módosítása
    - A fájlhoz rendelt logikai és numerikus információk, közvetlenül, azaz a fájl tartalmának megváltoztatása nélkül is felülírhatók.
- Adatátvitel: írás és olvasás
    - Közvetlen elérés esetében az információ címét, helyét kell ismerni.
    - Soros, szekvenciális elérés esetben a kezdő pozíciót kell ismerni, illetve az aktuális pozíciót kell tárolni.
- Hozzáírás, bővítés
    - Az állomány végéhez új információt írunk, aminek hatására az állomány mérete növekszik, esetlegesen új blokk lefoglalására kerül sor.
- Pozícionálás
    - Soros hozzáférés esetén megadhatjuk az aktuális pozíciót.
- Állomány megnyitása
    - az állomány megkeresése
    - jogosultságok ellenőrzése
    - az elvégezendő műveletek megadása
    - osztott állománykezelés szabályozása
    - soros hozzáférés pozíciójának beállítása
- Állomány lezárása
    - Puffer használata esetén a ki nem írt információ kiírása, osztott állománykezelés esetében az állomány felszabadítása.
- Programállomány végrehajtása
    - Az operációs rendszer létrehoz egy új folyamatot, a programállományt betölti a folyamat tárterületére és elindítja azt.
- Új állomány létrehozása
    - Szükséges az új allokációs bejegyzés(ek) létrehozása, valamint a szabad blokk(ok) lefoglalása.
- Állomány törlése
    - Szükséges a meglévő allokációs bejegyzés(ek) törlése, valamint a foglalt blokk(ok) felszabadítása.

Az osztott állománykezelés több folyamat esetén jellemzően a kölcsönös kizárás segítségével valósítható meg, ami vonatkozhat:

- a teljes állományra megvalósítható
  - a kizárólagos használat
  - több folyamat is csak olvashatja (azonnal vagy csak a lezárás után)
  - több folyamat is írhatja
    (a gyakorlatban sokkal inkább csak az állomány egy részére mód)
- az állomány egy részére megvalósítható
  - a rekord szintű kizárás
  - a holtpont és kiéhezés elkerülése

A hozzáférés szabályozásakor nem feledkezhetünk meg arról, hogy a fájlokat (állományokat) minden esetben valamilyen folyamat hozza létre, így a létrehozott fájlok jogosultságai a folyamat, illetve a folyamat felhasználójának (gazdájának) jogosultságaitól fog függeni.
A hozzáférési jogosultságok, fájlokhoz, könyvtárakhoz vagy elérési útvonalakhoz tartoznak. Ezeket a jogosultságokat felhasználónként illetve felhasználói csoportonként lehet definiálni.
Tipikus jogosultságok:

- fájl szinten
  átnevezés, írás, olvasás, végrehajtás, hozzáírás, törlés
- könyvtár szinten
  átnevezés, módosítás, listázás, keresés, új file létrehozás a könyvtáron belül, könyvtár törlés

A könyvtárak tulajdonképpen fájlok és más könyvtárak (alkönyvtárak) gyűjteménye, melynek tartalmát a könyvtár nyilvántartás írja le, nyilvántartás bejegyzések (Directory Entry) formájában.
A nyilvántartás bejegyzés a következő elemekből áll:

- az állomány neve
  A fájlnév vagy azonosító könyvtáranként egyedi, ugyanaz a név több könyvtárban is szerepelhet.
  Az fájlnév szerkezetileg lehet:
  - homogén karaktersorozat
    Például a UNIX esetében a fájlnév korlátozás nélkül bármilyen nyomtatható karaktert tartalmazhat.

    - önálló részekből álló karaktersorozat
      Például Windows rendszerek esetében megkülönböztetjük a fájlnevet és a kiterjesztést. Egyéb rendszerek esetében további önálló részek is lehetnek, például verziószám.
- 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 fájlhoz rendelt időpontok
      Leggyakoribb megvalósulásai: létrehozás, utolsó módosítás, utolsó hozzáférés, argumentumok utolsó módosítása, érvényességi idő.
    - hozzáférés jogosultságok
    - hivatkozás számláló
      Akkor használatos, ha egy fájlra több különböző néven, illetve több különböző elérési úton lehet hivatkozni.

A nyilvántartás bejegyzésben tárolt információkat a UNIX két részre bontja, ugyanis külön tárolja a kötet-nyilvántartást (Volume Directory) és a fájl nyilvántartást (File Directory).

A háttértáron tárolt információk mellett az operációs rendszerek az operatív tárban is tárolnak állománykezeléssel kapcsolatos információkat.

- az átvitel állapota (folyamatonként)
    - soros hozzáférés aktuális pozíciója
    - megengedett műveletek listája
      Egy fájl saját jogosultságainál egy folyamatnak az adott fájllal kapcsolatos műveleti jogosultságai magasabb prioritással bírnak.
- osztott kezeléssel kapcsolatos információk
    - mennyi folyamat kezeli egyidejűleg
    - a kölcsönös kizárást milyen módon, mekkora tartományra kell biztosítani
    - mely folyamatok várakoznak az állomány felszabadulására

Az állományok nagy és esetenként folyamatosan változó száma indokolttá teszi a könyvtárak hierarchikus szervezését. A könyvtár rendszerek legismertebb megoldásai a következők.

- Kétszintű könyvtárszerkezet

  Az egyes felhasználók állományai kerülnek egy könyvtárba. Ez a megoldás nem problémamentes, hiszen a felhasználó más felhasználók állományait, illetve az operációs rendszer állományait is szándékozhat használni.

- Fa struktúra

  A fa struktúra gyakorlatilag egy irányított gráf, két pont között egyetlen lehetséges útvonallal. A könyvtárak a fájlok mellett további alkönyvtárakat is tartalmazhatnak. A fa struktúrával kapcsolatos fogalmak:

  - aktuális könyvtár (Current Directory) folyamatonként
  - gyökér könyvtár (Root Directory)

    A hierarchia kiindulási pontja, ebből nyílnak a további könyvtárak.
  - elérési út (Path)

    A gyökér könyvtárból az aktuális könyvtárig vezető útvonal leírása, a közbeeső könyvtárak segítségével. A fa struktúrából következően minden (al)könyvtárhoz csak egyetlen elérési út tartozhat.
  - keresési útvonalak (Search Path)

    Azoknak az elérési utaknak a gyűjteménye, amely elérési utakon egy adott fájl megtalálása érdekében a rendszer az keresést végez.

- Körmentes irányított gráf

  Ellentétben a fenti irányított gráffal, ez esetben két pont között több útvonal is lehetséges, és a gráf szigorúan hurokmentes. Az egyes állományokat a gyökérből így több útvonalon is elérhetjük. Az állományra történő hivatkozás (Link) lehet:

  - fizikai

    A nyilvántartás bejegyzésben az állományt leíró információt tároljuk.
  - szimbolikus

    A nyilvántartás bejegyzésben az állomány (egy vagy több) elérési útvonalát tároljuk.

  A körmentes irányított gráf használatakor egyes esetekben problémát jelent az, hogy egy adott állományra több elérési úton is lehet hivatkozni, hiszen például mentéskor, vagy használati statisztikák készítésekor csak egy útvonalra lehet hivatkozni. Hasonló helyzet áll elő egy állomány törlésekor is.

- Általános gráf

  Ez a struktúra a gyakorlatban nem használatos, hiszen az általános gráfban található hurkok az addig ismertetett problémák mellett végtelen ciklusokat is okozhatnak egy állomány keresésekor. Az ok, amiért egyáltalán megemlítésre került ez a struktúra az, hogy a körmentes gráfok sérülés esetén (szerencsétlen esetben) részben vagy egészben általános gráffá alakulhatnak UNIX rendszerekben.

A könyvárakon jellemzően a következő műveleteket hajthatjuk végre:

- Attribútumok módosítása

  A könyvtárhoz rendelt logikai és numerikus információk felülírása
- Könyvtár létrehozása
- Könyvtár törlése

  Három lehetséges megvalósítási megoldása létezik:
  - a könyvtár akkor törölhető, ha üres
  - csak a könyvtárban tárolt fájlok törlődnek, de a könyvtárban lévő (al)könyvtárak, és azok tartalma nem
  - az összes tárolt információ fájl és könyvtár törlése
- Keresés
  - egy nyilvántartásban
    - rendezetlen keresés, teljes bejárással,
    - rendezett keresés, felező eljárással,
  - a könyvtárszerkezetben, bejárva a megadott elérési útvonalat
- Új bejegyzés létrehozása illetve törlése
  - nyilvántartás méretének dinamikus változása
  - blokkok lefoglalása illetve felszabadítása
  - rendezettség megtartása

  Törlés esetén a törölt bejegyzés első lépésként csak logikailag törlődik, azaz az állományt tartalmazó blokk „foglalt” bejegyzése „törölt” bejegyzésre változik. Ez azt jelenti, hogy egészen a fizikai törlésig (amikor az állományt tartalmazó terület felülíródik új adatokkal) elvileg (és időnként gyakorlatilag is ☺) visszaállítható a törölt anyag.