Operációs rendszerek elmélet

Előadás bemutatója

A számítógépek felépítése

- **A hardver fogalma:** a számítógépet alkotó fizikai elemek összessége.

  A **személyi számítógépet** (Personal Computer, PC) hardverelemek alkotják, amelyeket modulárisan építenek össze.
- **A szoftver fogalma:** a számítógépen lévő programok, adatok és azok dokumentációja.

2

Neumann-elvek

- **Tárolt program:** Az utasításokat az adatokkal azonos módon kell tárolni, egy nagy kapacitású tárolóban.
- **Kettes számrendszer:** Az előbb említett utasításokat és adatokat numerikus formában kell tárolni. Erre legalkalmasabb a bináris (kettes) számrendszer.
- **Vezérlőegység**: Ha az adatokat és az utasításokat egy helyen tároljuk, akkor szükség van egy olyan vezérlőegységre, amely különbséget tud tenni közöttük, és az utasításokat önműködően végre tudja hajtani.

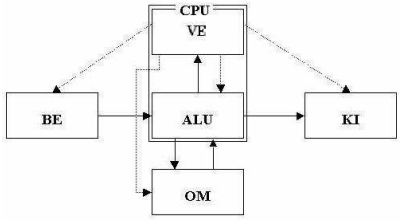
3

Neumann-elvek 2.

- **Aritmetikai-logikai egység**: Az előzőekből is kitűnik, hogy kell lennie a gépben egy olyan egységnek, amely a matematikai műveleteken túl képes elvégezni elemi logikai műveleteket is. (Aritmetical-Logical Unit, ALU)
  Napjainkban a központi vezérlő egység (Central Processing Unit, CPU) végzi el a vezérlő egység és az aritmetikai-logikai egység feladatát is.
- **Perifériák**: Szükség van olyan bemeneti/kimeneti (input/output, I/O) egységekre, amelyek segítségével létrejöhet a kapcsolat az ember és a gép között.

4

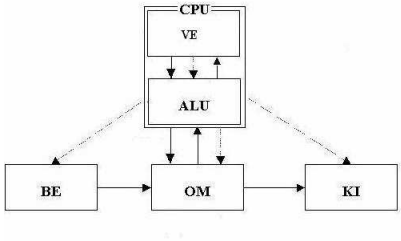
A kezdetek: processzorcentrikus felépítés



VE = Vezérlőegység
OM = Operatív Memória
Az adat jellegű információk útját folytonos, a vezérlő-jelek útját szaggatott nyíl jelzi.

5

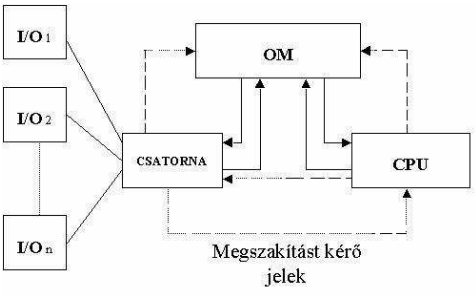
Memóriacentrikus felépítés



VE= Vezérlő egység
OM = Operatív memória
Az adat jellegű információk útját folytonos, a vezérlő-jelek útját szaggatott nyíl jelzi.

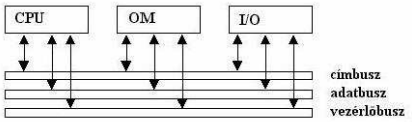
6

Autonóm I/O csatornával rendelkező szg.



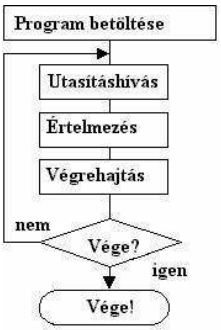
7

- A **busz** (vagy másik nevén a Sín) biztosítja az adatok cseréjét az egyes részegységek között.
- *Címbusz*, amelyet a processzor arra használ, hogy az adatátvitelhez egy meghatározott tárolócella vagy adatvezeték címét kijelölje;
- *Vezérlőbusz*, amely vezérlőjeleket továbbít;
- *Adatbusz*, amely a processzor és a számítógép más részei közötti adatforgalmat bonyolítja le.



8

A Neumann-ciklus:



9

A perifériákkal való kommunikáció

- Lekérdezéses átvitel (polling): a processzor folyamatosan kérdezi le a periféria állapotát, és ha érdemleges információt talál, akkor, beolvassa azt.
- Megszakításos átvitel (Interrupt ReQuest, IRQ): a periféria a számára kijelölt megszakításkérő vonalon jelzi a processzornak az adatátviteli igényét. A processzor megszakítja addigi munkáját és kiszolgálja a perifériát. Hátránya: mentés és újra betöltés.
- Közvetlen memóriaátvitel (Direct Memory Access, DMA): a memória és a periféria közötti átvitel a processzortól függetlenül megy végbe, ezért a processzort nem foglalja le. Az átvitel megkezdése előtt a kezdő címet és az átadandó blokk méretét kell közölnie az autonóm-csatorna vezérlővel.

10

Funkcionális megközelítés

- Az **operációs rendszer fogalma**: olyan program (programrendszer), amely a számítógépen futó összes folyamat végrehajtását vezérli (ISO szabvány szerinti megfogalmazás)!
- A számítógépes rendszer hardver és szoftverrétegekből felépülő, hierarchikusan strukturált rendszer.

11

Az operációs rendszerek alapvető feladatai:

- **Taszkok ütemezése, összehangolása**
- **Erőforrások kezelése**
- **I/O műveletek megszervezése**
- **Hibakezelés**
- **A helyes és optimális működés biztosítása**
- **Program- és adatvédelem biztosítása**
- **Naplózás**

12

Az operációs rendszerek alapvető feladatai 2:

- **Megszakításkezelés**
- **Monitoring**
- **Kapcsolattartás a felhasználókkal**
  - programindítás, kapcsolattartás a folyamatokkal;
  - a rendszermag szolgáltatásainak közvetlen elérése;
  - a rendszermag programozói felülete;
  - alapvető segédprogramok.

13

Az operációs rendszer felépítése

Az operációs rendszeren belül megkülönböztetjük a rendszer magját: a ***kernelt***, és a felhasználóval kapcsolatot teremtő részt a burkot, más szóval a ***shellt***.

Külső erőforrások

- Az úgynevezett nyílt rendszerek lehetőségeket kínálnak az egyes hardvergyártóknak, hogy a rendszerhez jól illeszkedő eszközvezérlő programokat készítsenek, amelyek beépülhetnek a rendszermagba.
- A Windows (95, 98) és fejlettebb verziói az automatikus beállítások támogatását tűzték ki célul. Az új technológia neve a ***Plug and play*** (PnP).

14

Az operációs rendszer felépítése 2

Belső erőforrások: a számítógépes rendszer azon összetevői, amelyek szükségesek a folyamatok működéséhez. A két legfontosabb erőforrás: a rendelkezésre álló ***processzoridő*** és a ***memóriaterület.***

- Az éppen végrehajtás alatt álló folyamat számára a számítógép erőforrásai úgy jelennek meg, mintha egyedüli tevékenység lenne a rendszerben (**virtuális gép**).

15

Karakteres felhasználói felület

- A karakteres felületen a felhasználók parancsokat gépelnek be, amelyet az operációs rendszer értelmez, majd végrehajt. Ehhez kell egy parancsértelmező (command interpreter).
- A shell alapvető feladatai:
  - programindítás, programkezelés
  - egyéb, operációsrendszer-funkciók felhasználói szintű biztosítása (fájl kezelés)
- Programkezelés
  - A betölthető fájl kiválasztása (.EXE .COM .BAT)
  - A program számára megfelelő környezet biztosítása (AUTOEXEC.BAT CONFIG.SYS)

16

Grafikus felhasználói felület

- A processzorok műveleti sebességének növekedése maga után vonta a multitask rendszer létrejöttét. Igényként merült fel a különböző folyamatok nyomon követésére (egy-egy ablakban). Ez hozta maga után az ***ablakozó technikákat.*** Jellemző az ***egér*** használat.
- ***A felhasználóbarát felület jellemzői***
  - *Könnyű legyen megtanulni:*
  - *Jól kezelhető súgó és help (segítség) rendszere legyen.*
  - *Minden utasításra legyen válasz.*
  - *Meg lehessen szakítani elindított műveleteket.*
  - *Lehessen visszavonni a tévedésből kiadott műveleteket, minden következmény nélkül.*

17

Lemezkezelés

A mágneses jelrögzítés alapelvei

- **Írás:** Az elektromos áram mágneses mezőt hoz létre, és megváltoztatja az anyagok mágnesezettségét. Az írófej mágneses indukcióvonalai a mágneses hordozóanyagokon keresztül záródnak.
- **Olvasás:** Változó mágneses mezőben lévő vezetőben áram indukálódik.
- **Hordozóanyag:** Műanyag fóliára, alumíniumlemezre vagy más felületre felvitt kemény mágneses anyag (például króm-dioxid).

18

Állománykezelés

- **Állomány** (fájl):
  1. valamely háttértárolón elhelyezkedő adatok egyedi azonosítóval rendelkező együttese;
  2. az adattárolás egysége a felhasználó szempontjából.
- **Állományszervezés**: az operációs rendszer azon tevékenységeinek összessége, amely a felhasználói információk háttértárolón történő elhelyezésével, azonosításával és visszakeresésével kapcsolatos.

19

Kódolási eljárások

- NRZI: Az adatok impulzusok és szünetek segítségével kerülnek rögzítésre. Szalagos tárolóknál használatos a NRZI kódolás. Csak az 1 biteknél van fluxusváltás.
- FM 0,1 RLL: Ha egy szünetet egy impulzus követ, akkor az adatbit pl. 0, ha kettő, akkor az adatbit 1. A frekvenciával (jelek sűrűségével) kódoljuk az adatot. Egymás után következhet két impulzus (minimális futási hossz 0), de a szünet után mindig van órajel (maximális futási hossz 1).
- MFM 1,3 RLL: Ha szünet után impulzus következik, az jelenti az 1-et. A 0-t pedig, ha nem volt impulzus az előző adatbit végén, akkor az impulzus után egy szünet, vagy, ha az előző adatbit végén impulzus volt, akkor két szünet. Ez biztosítja, hogy legalább 1, de maximum 3 szünet legyen.

20

Kódolási eljárások 2.

- RLL 2,7 ARLL 3,9 RLL (Advanced RLL): A bejövő hét bitkombináció lefedi az összes lehetséges értéket. Az MFM-eljárással a tárolókapacitás 1,5-szeresét lehet így elérni, az ARLL-eljárással pedig a kétszeresét. A sávokban lévő szektorok száma így 17-ről 26-ra, illetve 34-re növelhető. A ma árusított merevlemezek kódolása RLL 2,7.

Adat	1		0		1		0		0		0		1		1	
FM	i	i	i		i	i	i		i		i		i	i	i	i
MFM		i				i			i		i			i		i
RLL 2,7 felbontás	10				10				0011							
RLL 2,7 kódolás	01		00		01		00		00		00		10		00	
RLL 2,7		i				i							i			

Kódolási eljárások összehasonlítása - ASCII „ú”

21

Soros elérésű tárak

- **Mágnesszalag (Kazetta)**
  Stremer, catridge, DAT
- **Mágneslemezek**
  - Az adatok tárolásának módja: formázás, sáv, szektor, cilinder
  - Lemezműveletek ütemezése:
    - FCFS (First Come First Served)
    - SSTF (Shortest Seek Time First, Először a legrövidebb keresési idő)

22

- Scan: A scanning módszer úgy rendezi sorba a kéréseket, hogy a fej mindig egy irányba mozogjon. (lift algoritmus).
- C-scan **(Circular scan):** A fejek az utolsó sávra érve visszatérnek az elejére.
- Look: a look algoritmus ugyanaz, mint a scan vagy a c-scan, de csak a legutolsó szükséges adatig halad egy adott irányba..
- C-look: Ugyanaz, mint a look algoritmus, de a fejek az utolsó sávra érve visszatérnek az elejére, így mindig csak egy irányban mozognak.

FCFS	50	45	110	47	112	14	23			305
SSTF	50	47	45	23	14	110	112			144
Scan	50	47	45	23	14	(0)	110	112		162
C-scan	50	110	112	(119)	(0)	14	23	45	47	245
Look	50	47	45	23	14	110	112			134
C-look	50	110	112	14	23	45	47			193

Az algoritmusok összehasonlítása és a fejmozgások összege

23

- A fordulási idő csökkenthető szektor-sorbarendezéssel.
- További gyorsítási lehetőségek:
  - A gyakran használt adatokat scan esetén a lemez közepére célszerű tenni;
  - redundáns (többszörös), elosztott tárolás;
  - tömörítés;
  - több példányban történő tárolás.
- A megbízhatóság növelhető:
  - rendszeres mentéssel;
  - egyszerre több blokk átvitelével;
  - a cache és a lemez szinkronizálásával.

24

A lemez logikai felépítése

- Alapegysége: klaszter
- logikai partíciók: FAT 16, FAT 32
- A FAT (Fájl Allocation Table)-tábla tulajdonképpen egy táblázat, amely 12 bites tárolás (hajlékonylemez) esetén 2↑12. hatványon=4096 bejegyzést,
  16 bites tárolás esetén 2↑16. hatványon=65536 bejegyzést tartalmazhat, amelyek közvetlenül hivatkoznak az adott sorszámú klaszterra.
- A FAT táblát egymás mögött két példányban tároljuk és ezután következnek az állományok fastruktúrájú elérésére szolgáló bejegyzések.

25

Merevlemez-csatoló típusok

- **IDE (Integrated Drive Electronics):** Adatátviteli sebesség: 1-4 Mbyte/s.
- **EIDE (Enhanced IDE - továbbfejlesztett IDE):** 255 fejet tudott kezelni, ezáltal a kezelhető lemezméret 255*1024*63*512=7,44 Gbyte.
- **UDMA (Ultra Direct Memory Access -** közvetlen memóriahozzáférés**)**
- **SCSI**

26

Hajlékonylemez (floppy disc)

Kapacitás	Átmérő (inch)	Sávok száma	Szektor/ sáv	Sávsűrűség (tpi)	Átviteli sebesség (kbyte/s)	Típus
360 Kbyte	5,25	40	9	48	250	DS,DD
1200 Kbyte	5,25	80	15	96	500	DS,HD
720 Kbyte	3,5	80	9	135	250	DS,DD
1440 Kbyte	3,5	80	18	135	500	DS,HD

- Lemeztípusra vonatkozóan az SS egyoldalast, a DS kétoldalast jelöl, a sávsűrűségnél az SD szimpla sűrűséget, a DD dupla sűrűséget, a HD nagy sűrűséget jelöl. A tpi (Track Per Inch)a hüvelykenkénti sávok számát jelöli. A lemez kapacitása pld. 80sáv*18szektor*512byte*2oldal=1473560byte=1440Kbyte= ~1,4 Mbyte.

27

Egyéb eszközök

- LS-120: a 120 Mbyte kapacitást a lézer-vezérlésű technika teszi lehetővé.
- ZIP-drive
- Flash drive

28

RAID rendszerek

- RAID (Redundand Array of Independent/Inexpensive Discs, olcsó diszkekből készült tároló tömb)
- A RAID rendszerek legújabb típusai képesek arra, hogy a meghibásodott diszket menet közben cserélhessük, illetve szerepét átvegye egy eleve beépített tartalék diszk, a rossz tápegységek is melegen (üzem közben) cserélhetők, illetve belép helyettük egy tartalék, a tömb méretét is növelhetjük, és átkonvertálhatjuk másik RAID rendszerbe.

29

RAID típusai

- **RAID0** nem redundáns (meghibásodás esetén nincs mód az adatok visszaállítására), de mind az olvasási, mind az írási műveletek párhuzamosan zajlanak. A RAID alapelve a striping (csíkozás), az adatokat nem folytonosan tároljuk a lemezeken, hanem az egymást követő mezők mindig más diszkre kerülnek a táblázatnak megfelelően.

1. diszk	2. diszk	3. diszk
1	2	3
4	5	6
7	8	9
...		

Csíkozás (striping)

- Az 1-2-3. mező egy csíkhoz tartozik, és a csíkok annyi mezőből állnak, amennyi diszket tartalmaz a rendszer. Amennyiben adatokhoz akarunk hozzáférni, a tömb valamennyi diszkjéhez egy időben hozzáférhetünk, ami jelentős sebességnövekedést eredményez, mert az író/olvasó fej mozgatása lassítja leginkább a műveletvégzést.

30

RAID típusai 2

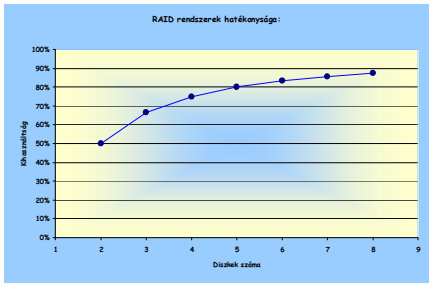
- **RAID1:** Az adatokat egyszerűen megkétszerezi, azaz mindkét diszken tárolja, és ez meghibásodás esetén jó védelmet biztosít.
- **RAID2:** Minden csík tartalmaz ellenőrző hibajavító biteket (ECC, Error Checking and Correction), amelyet külön diszkek tárolnak. Manapság nincs gyakorlati jelentősége, mert a diszkek eleve ECC-ellenőrzöttek.
- **RAID3:** A csíkok szélessége 1 byte. Egy diszket használunk a többiből képzett paritásbit tárolására. Meghibásodás esetén a működő diszkek adataiból kizáró-vagy művelettel állítható elő a hiányzó adat.
- **RAID4:** Egy paritásdiszk+RAID0, illetve szélesebb csíkozású RAID3. Nem elterjedt a paritásdiszk okozta szűk keresztmetszet miatt (a paritásdiszket minden íráskor frissíteni kell).
- **RAID5:**A paritásbitet az összes diszken elosztva tároljuk (rotating parity). A csíkok szélessége változtatható: keskeny csíkok esetén RAID3-ra, széles csíkok esetén RAID4-re hasonlít.
- **RAID6:** Soronként és oszloponként is tárolja a paritást. Hasonlít a RAID5-re, de kétszeres meghibásodást is kezel.

Jellemzően a RAID 0; 1; 5; 1+0; 5+0 megoldások terjedtek el.

31

A RAID rendszerek hatékonysága

A megvalósítás lehet szoftveres, amely nem igényel jelentős beruházást, de terheli a rendszert, illetve RAID-vezérlő (hardveres megoldás).



32

Optikai tárolók (CD-technológia)

- **A CD-DA (Compact Disc- Digital Audio) alapvető tulajdonságai („Vörös könyv”)**
- **A CD-ROM (Compact Disk-Read Only Memory)**
- **CD-ROM/XA (CD-I CD-Extra Video CD Photo CD)**
- **CD-R (Recordable CD)**
- **CD-RW (Rewritable CD)**
- **DVD**

33

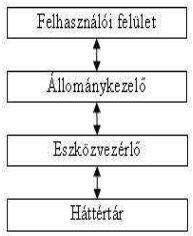
Állomány kezelés

- **Állomány**:
  - valamely háttértárolón elhelyezkedő adatok egyedi azonosítóval rendelkező együttese;
  - az adattárolás egysége a felhasználó szempontjából.
- **Állományszervezés**: az operációs rendszer azon tevékenységeinek összessége, amely a felhasználói információk háttértárolón történő elhelyezésével, azonosításával és visszakeresésével kapcsolatos.

34

Az állománykezelésben részt vevő komponensek

- A felhasználó által megadott azonosító alapján az **állománykezelő** meghatározza a keresett információ fizikai helyét, és (szükség esetén az **eszközvezérlő** segítségével) utasítja a **háttértár** író/olvasó mechanizmusát a megfelelő művelet elvégzésére.



35

Állomány tulajdonságok

- **típus**: az állomány nevét kiegészítik egy azonosítóval, ami segít annak meghatározásában, hogy milyen műveletek értelmezettek az adott adatcsoporttal. A DOS alapú operációs rendszerekben szokás ezt **kiterjesztésnek** is nevezni;
- **időbélyegek**: feljegyzése részben adminisztratív, részben biztonsági célokat szolgál.
  - **keletkezésének időpontja**,
    - **utolsó megnyitás**, **utolsó módosítás**

36

Állomány tulajdonságok 2.

- **méret**: az állományban tárolt információ mennyisége. (nem azonos az elfoglalat hellyel)
- **jellemzők** (attribútumok): operációs rendszerenként változó kiegészítő információk
- **jogosultságok**: amennyiben az operációs rendszer támogatja, megadható (korlátozható) az egyes állományokon végezhető műveletek (pl. módosítás, törlés, megnyitás, tulajdonságok megváltoztatása, stb.) köre.

37

Állományműveletek:

létrehozás: az állománykezelő a háttértár alkalmas (méretű) területét lefoglalja és a háttértár foglaltságát tartalmazó táblázatban rögzíti az állomány azonosítására szolgáló adatokat;

megnyitás során az operációs rendszer *azonosítja az állományt* (logikai név – fizikai elhelyezkedés összerendelése), *ellenőrzi a kijelölt művelet végrehajtásához szükséges jogosultságok meglétét* és egy ***fájlleíró táblát*** (FCB – File Controll Block) hoz létre, *ami tartalmazza az állomány elhelyezkedésével és kezelésével kapcsolatos információkat* – lényegében ezen a struktúrán keresztül kezelhető az állomány tartalma.
A ***megnyitás módja***i: - olvasásra,
- írásra, mely lehet: módosítás, hozzáfűzés, felülírás

38

Állományműveletek 2.

pozicionálás: a következő írási vagy olvasási művelet elvégzésének helyének meghatározása. Az operációs rendszer egy mutató segítségével tartja nyilván, hogy a feldolgozási folyamat az állomány tartalmának mely részénél jár (ez a mutató is az FCB-ben található), ennek a mutatónak a beállításával történik az állománynak a következő műveletben érintett részének a kijelölése;

lezárás: az FCB megszüntetése, az állomány tartalmának rögzítése a háttértáron. Lezárt állománnyal állományszintű művelet nem végezhető.

39

Állományműveletek 3.

Az állományok kezelése során az állomány tartalmának feldolgozási módja szerint megkülönböztethetünk:

- **Szöveges:** a tárolt információ értelmezését a feldolgozó programnak kell elvégeznie
- **bináris módú**: az adatok feldolgozását az állományban tárolt vezérlőjelek – tipikusan ilyen a **„sor vége”-jel** – segítik

pozicionálás szempontjából pedig

- **Soros:** a pozicionálás csak folyamatosan növekvő módon történhet, azaz az állomány egy adott bájtjának feldolgozásához az összes megelőző bájton végig kell haladnunk
- **közvetlen elérésű:** van lehetőség az állomány tetszőleges pontjának sorszám alapján történő közvetlen elérésére

40

Könyvtár

1. az operációs rendszer által használt, adminisztratív célokat szolgáló **állomány**, amely a felhasználói információt tartalmazó állományok logikai csoportosítását a fizikai elhelyezkedésük tárolásával teszi lehetővé;
2. a (felhasználói) állomány elhelyezkedésének logikai struktúráját tükröző bejegyzés.

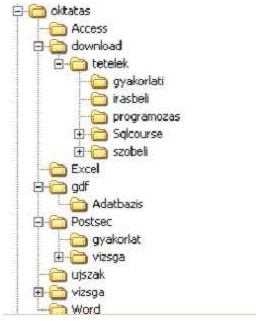
A könyvtár nem más, mint egy nyilvántartás

- rugalmasság: előre nem ismert, hogy a könyvtárban hány állomány információit fogjuk tárolni – sőt ez az érték a felhasználás közben folyamatosan változik.
- csoportosíthatóság: tetszőleges szempontok szerint.
- gyors visszakeresés: a könyvtárak szerepe abban áll, hogy segítsenek a nagyszámú állomány között megtalálni a számunkra fontosat

41

Könyvtár 2.

hierarchikus fastruktúra: Ez a struktúra egy *körmentes*, *irányított gráffal* írható le, amelynek pontosan egy kiinduló csúcspontja (**gyökér**) van, és minden más pontjába pontosan egy út vezet a gyökértől. Az egyes csúcsokból tetszés szerinti számú (irányított) él vezet más csúcsokhoz, azt a csúcsot, ahonnan egy él kiindul, **szülőnek**, ahová vezet, **gyereknek** (**alkönyvtár**) nevezzük.



42

Könyvtárakra vonatkozó műveletek

1. **létrehozás**: lévén a könyvtár csak egy virtuális állomány, létrehozása nem jár tényleges helyfoglalással, csupán a szülőkönyvtár nyilvántartása bővül egy újabb bejegyzéssel;
2. **megnyitás**: az állományoknál tárgyalttal azonos értelmű művelet, a könyvtárbejegyzéseket tartalmazó fizikai állomány tartalmának hozzáféréséhez szükséges, a nyitott bejegyzés-állományon értelmezett műveletek és hatásuk:
   - **olvasás**: a könyvtárban tárolt állományok (és/vagy könyvtárak) adatainak lekérdezése (listázása);
   - **írás**: a könyvtár egy adott állományára (könyvtárára) vonatkozó leíró részek módosítása. Ide tartozik:
   - **állomány létrehozása** (az állomány létrejöttekor készül róla egy feljegyzés a könyvtár nyilvántartásában);
   - **állományok tulajdonságainak** (név, méret, stb.) **megváltoztatása** (mivel ezek a információk nem az állományban tárolódnak);

Könyvtárakra vonatkozó műveletek 2.

- **állományok törlése** (a tartalmazó könyvtár állományra vonatkozó bejegyzése módosul, nem történik meg az állomány tényleges eltávolítása a háttértárról)
- **állomány helyének megváltoztatása** (**másolás**: az állomány tartalma ismételten található meg a háttértár két különböző helyén; **áthelyezés**: az állomány logikai helyének megváltoztatása – általában nem jár az állomány fizikai helyének megváltozásával, csupán az érintett könyvtárak adminisztrációs bejegyzéseinek módosulásával);

logikai (felhasználói) értelemben ugyanígy nevezzük egy könyvtár kiválasztását, aktuálissá tételét is (szokás **könyvtárváltásnak** is nevezni). Hatására azok az állományok, amelyek ebben a könyvtárban találhatóak, hozzáférhetőek: megjeleníthetők (listázhatók), feldolgozhatók. Ez a művelet helyettesíthető az elérési út (ld. később) alkalmazásával.

Hivatkozások

- **Elérési út:** egy könyvtár helyének megadása a hierarchikus könyvtárstruktúra szerkezetének A megadás során az elérési út könyvtárait az operációs rendszer speciális szimbóluma (általában a / vagy a \) választja el egymástól.
  - **Abszolút elérési út:** a gyökérkönyvtártól induló és a hivatkozott könyvtárig az összes tartalmazó könyvtárat a hierarchiában elfoglalt helyének megfelelő sorrendben megadó útvonal-leírás.
  - **Relatív elérési** út:a hivatkozott könyvtár elérését megadó leírás, amely az aktuálisan használt (munka-) könyvtártól indul

Kötet

- **Partíció**: a háttértár fizikailag összefüggő (folytonosan sorszámozott szektorokat tartalmazó) önálló része, amely az operációs rendszer számára önálló adattároló egységként kezelhető.
- **Kötet**: a háttértároló egyedi azonosítóval rendelkező része. Szokás **meghajtónak** is nevezni

A kötet és a fizikai meghajtó közötti megfeleltetés egyáltalán nem egyértelmű: egy particionált winchester esetében az összefüggés: egy eszköz – több kötet több meghajtó, tárolási területének összevonására egyetlen logikai név alá pedig több eszköz – egy kötet (pl.: RAID).

Állományszervezés

Folytonos állomány-elhelyezési módszerek

A legjellemzőbb stratégiák:

- **legelső megfelelő** (First Fit): az első alkalmas méretű tartományt foglalja le. Leggyorsabb, de nem feltétlen hatékony;
- **legjobban illeszkedő** (Best Fit): azt a tartományt használja fel, amely a legkisebb mértékben tér el az állomány méretétől. Lassú, de hatékony: a legkevesebb veszteséget eredményezi;
- **legrosszabbul illeszkedő** (Worst Fit): az előző ellentéte, a legnagyobb eltérésű területen helyezi el az állományt – így biztosítja, hogy továbbra is (relatíve) nagy blokk-csoportok maradjanak felhasználhatóak.

Nem folytonos állomány-elhelyezési módszerek

- **Láncolt lista**: (File Allocation Table)

Fájl: valami.txt

Könyvtárbejegyzés

név	méret	dátum	jellemzők	#1blokk
valami.txt	22 845	2002-01-12	10110001	74

FAT

59	60	61	...	74	...	88	89	...	93	94	95
66	00	94		61		60	77		89	95	88

Háttértár

59	60	61	...	74	...	88	89	...	93	94	95
	#6	#2		#1		#5				#3	#4

Nem folytonos állomány-elhelyezési módszerek 2.

- **Láncolt lista**: (File Allocation Table)

Előnye:

– hogy minden blokk kihasználható,
– nincs szükség a helykereső stratégiák alkalmazására

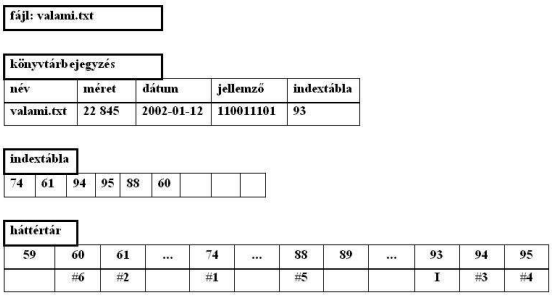
Hátránya:

– a FAT szerepének kizárólagossága,
– egyetlen állomány feldolgo-zásához is az egész táblázatot a memóriában kell tárolni,
– nem támogatja az állomány tartalmának közvetlen elérését

49

Nem folytonos állomány-elhelyezési módszerek 3.

- **Indextábla**: ebben az esetben minden állomány saját fájlelhelyezkedési táblával rendelkezik.



50

Nem folytonos állomány-elhelyezési módszerek 4.

- **Indextábla**: ebben az esetben minden állomány saját fájlelhelyezkedési táblával rendelkezik.

Előnye:

– hogy az indextábla esetleges sérülése csak az adott állomány elvesztését eredményezi,
– a mérete jelentősen kisebb a FAT-nál,
– lehetőséget ad az állomány tartalmának közvetlen elérésére

Hátránya:

– az állományok maximális méretét meghatározza az indextábla mezőinek a száma
– ennél a módszernél külön kell gondoskodni a szabad és foglalt blokkok megkülönböztetéséről,
– állomány tartalmának módosulása az indextábla újra-összeállítását igényli.

51

Alapvető kernelfunkciók

- A számítógép működése során állandóan több tevékenységet hajt végre, többé-kevésbé egyszerre.
- E tevékenységeknek összehangolása azonban felvet néhány problémát:
  1. A mai számítógépek működésüket tekintve Neumann-elvűek, és mint ilyenek, a műveleteket sorosan hajtják végre.
  2. A számítógép funkcionális egységeinek tárgyalásakor láttuk, hogy a vezérlőegység egyszerre csak egy feladat ellátására képes
  3. A számítógép egyes részegységei, az egyes feladatok ellátásához szükséges hardver (vagy szoftver) eszközök nem állnak korlátlan mennyiségben rendelkezésre – gondoskodni kell az egyes eszközök kiosztásáról.

52

Alapvető kernelfunkciók 2.

- **Algoritmus**: valamely feladat megoldását eredményező tevékenység-sorozat lépéseinek egyértelmű, meghatározott és véges megadása..
- **Utasítás**: az algoritmus egysége, egy elemi tevékenység.
- **Program**: a számítógép által értelmezhető algoritmus.
- **Folyamat**: a vizsgált számítógépes rendszerben az adott pillanatban végrehajtás alatt álló program.

53

Alapvető kernelfunkciók 3.

- **Erőforrás**: a számítógépes rendszer azon összetevői, amelyek szükségesek a folyamatok működéséhez. A két legfontosabb erőforrás a rendelkezésre álló **processzoridő** és a **memóriaterület**.

- Az éppen végrehajtás alatt álló folyamat számára a számítógép erőforrásai úgy jelennek meg, mintha egyedüli tevékenység lenne a rendszerben (**virtuális gép**).

54

Alapvető kernelfunkciók 4.

- Azonban a folyamatok száma meghaladja a processzorok számát, így szükség van a folyamatok közötti váltásra.

  Ebből következik, hogy az egyes folyamatok számára nemcsak biztosítani kell a megfelelő erőforrások elérhetőségét, de gondoskodni kell az „egyedüllét illúziójának” fenntartásáról is!

55

Alapvető kernelfunkciók 5.

- **Folyamatleíró blokk** (Process Controll Block): az operációs rendszer által az egyes folyamatok jellemzésére használt adatok csoportja, amely alkalmas a folyamat azonosítására és a folyamat által használt erőforrások nyilvántartására.
- **Rendszermag** (kernel): Az operációs rendszer folyamatainak egy olyan kitüntetett csoportja, amely a rendszer- és a felhasználó által indított folyamatok között elosztja a rendelkezésre álló erőforrásokat.

56

Alapvető kernelfunkciók 5.

- Ezek a folyamatok a számítógép bekapcsolásakor jönnek létre, folyamatosan jelen vannak a rendszer működése során, létrehozzák, adminisztrálják és vezérlik a felhasználói folyamatokat.
  A rendszermag és a felhasználói folyamatok közötti kommunikáció **rendszerhívásokon** (általában a processzor egy kitüntetett utasítása) keresztül valósul meg.

57

Az operációs rendszer a rendszerhívás hatására:

1. befejezi a jelenleg végrehajtás alatt álló utasítást
2. elmenti a jelenleg futó folyamat jellemzőit
3. a rendszerhívás típusának (értékének) megfelelő rendszerfolyamat elindul és nyilvántartásba veszi az igénylő folyamatot (elkészíti a folyamatleíró blokkot, megvizsgálja az igény jogosultságát és/vagy sürgősségét és dönt a kiszolgálásról vagy várakoztatásról)

58

Az operációs rendszer a rendszerhívás hatására (2):

- a rendszerfolyamat visszajelzést küld az igénylő folyamatnak a döntéséről (esetleg engedélyezi számára a végrehajtást)
- a félbeszakadt folyamat környezete helyreállításra kerül és az folytatódhat tovább.

59

A rendszerhívások keletkezésük szerint különböző típusúak lehetnek:

- **kivétel**: a végrehajtás alatt álló folyamat működésében bekövetkezett, a hibátlan működést befolyásoló esemény, (pl. 0-val osztás)
- **csapda**: a felhasználói folyamat által okozott olyan esemény, amelynek kezelése az operációs rendszer feladata
- **megszakítás**: a futó folyamattól független (külső) forrásból származó rendszerhívás.

60

A rendszerhívások keletkezésük szerint különböző típusúak lehetnek (2):

- Aszerint, hogy a folyamat által használt erőforrás valamilyen okból történő elvétele a folyamat működését hogyan befolyásolja, megkülönböztetünk
  - **elvehető** (rabolható, „preemptive") erőforrásokat, amelyekre az jellemző, hogy elvételük esetén a folyamat végrehajtása megszakad, de az erőforrás ismételt rendelkezésre állásakor a folyamat tevékenysége hiba nélkül folytatható
  - **nem elvehető** („non-preemptive") erőforrásokat, amelyek elvétele esetén a folyamat hibamentes folytatása nem lehetséges.

61

A rendszerhívások kezelése:

1. Rendszerhívás érkezik.
2. Az éppen végrehajtás alatt álló utasítás befejezése után a processzor megvizsgálja a kérést:
   - ha a jelenlegi maszkolási beállítások és prioritási szintek lehetővé teszik, végrehajtódhat,
   - ellenkező esetben várakoznia kell.
3. A végrehajtás alatt álló folyamat környezete elmentésre kerül.

62

A rendszerhívások kezelése:

- Beállításra kerülnek az adott típusú rendszerhíváshoz rendelt tiltások (maszkok) és a prioritási szint.
- A kérés helyének és típusának alapján meghatározásra kerül a kiszolgáló folyamat címe és átadódik számára a vezérlés.
- A kiszolgáló folyamat befejeződésekor a letiltott megszakítások és a prioritási szintek visszaállítódnak.
- A félbeszakadt folyamat környezete visszatöltődik és megkapja a vezérlést.

63

Folyamatok kezelése

- **Ütemezés**: a folyamatok működése során a rendelkezésre álló idő leghatékonyabb kihasználását célzó eljárások összessége

Egy program állapotai az elindításától a befejeződéséig:

1. A program a háttértárról a memóriába kerül (még nem folyamat!), és rendszerhívás segítségével jelzi igényét a processzorra és egyéb erőforrásokra.
2. A folyamatok kezeléséért felelős rendszerfolyamat érzékeli az igényt és nyilvántartásba veszi a programot (létrehozza a folyamatleíró blokkot).

64

Folyamatok kezelése (2)

3. Megfelelő feltételek (prioritás, szabad processzor stb.) teljesülése esetén elkezdődik a folyamat végrehajtása.
4. Mivel a rendszerben több folyamat is működik, a folyamat időnként kikerül a végrehajtás alól és várakozni kényszerül. Környezetváltás.
5. Befejeződésekor kikerül a folyamatok közül.

65

Folyamatok kezelése (3)

- Ezeknek a tevékenységeknek a megoldásához az operációs rendszer különböző, **ütemezőnek** nevezett komponenseket használ:
- **Főütemező** (magas szintű ütemező): feladata a program betöltése a háttértárról a memóriába és a programhoz tartozó folyamat(ok) számára a folyamatleíró blokk elkészítése.
- **Középtávú ütemező** (közbenső szintű ütemező): feladata a végrehajtásra várakozó folyamatok működésének ellenőrzése, az optimális végrehajtási időnek megfelelő végrehajtási sorrend kialakítása.
- **Rövid távú ütemező** (alacsony szintű ütemező): feladata a futásra kész folyamatok között a processzor működési idejének szétosztása.

66

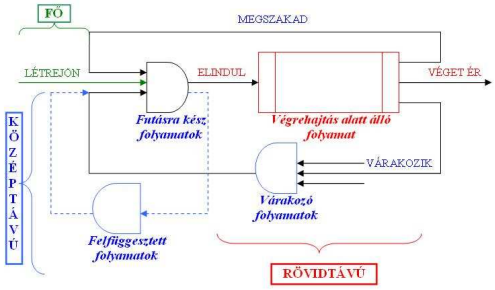
Folyamatok kezelése (4)

- **Várakozási sor** („queue”): A működésük azonos logikai állapotában levő folyamatok adatainak tárolására és nyilvántartására szolgáló listaszerkezetű tároló.

Milyen állapotai lehetnek egy folyamatnak?

1. létrejött,
2. futásra kész,
3. végrehajtás alatt,
4. felfüggesztett (várakozik, megszakadt),
5. befejeződött.

67



Folyamat-állapotok és ütemezők

68

Ütemezési stratégiák

- **A főütemező** stratégiájának a meghatározása gyakorlatilag lehetetlen feladat, hiszen az lenne az elvárás vele szemben, hogy olyan sorrendben hozza létre a végrehajtásra váró programokból a folyamatokat, hogy azok a rendszerben jelenleg működő folyamatokkal együttesen egyenlő arányban igényeljenek processzor-műveletet és külső forrásból származó adatot

69

Ütemezési stratégiák 2.

- **A középtávú ütemező** stratégiáját tekintve meglehetősen sokféle lehet, de a gyakorlatban ezek az ütemezők a prioritási szintek alapján döntenek az egyes folyamatok állapot-változtatásáról.
- **A rövidtávú ütemezővel** kapcsolatban a legfontosabb elvárás a környezet-váltás gyors elvégzése és a folyamatok igazságos kiszolgálása

70

Ütemezési stratégiák 3.

Ezeknek az (alkalmasint szubjektív) fogalmaknak a mérése elég nehéz, ezért olyan mérhető jellemzőkkel szokás elemezni a rövidtávú ütemező hatékonyságát, mint pl.:

- **várakozási idő**: a folyamat mennyi ideig nem fut;
- **átfutási idő**: a folyamat érkezése és befejezése közti idő;
- **válaszidő**: a folyamat rendszerbe-lépésétől az első futásig eltelt idő.

71

Ütemezési algoritmusok

Előbb jött, előbb fut („First Come First Served”): a folyamatok végrehajtási sorrendje megegyezik a várakozási sorba érkezésük sorrendjével.

Előnye: a lista tárolási szerkezetének megfelel a feldolgozás menete, így egyszerűen megvalósítható.

Hátránya: a nagyobb időigényű folyamatok feltarthatják a rövidebb folyamatokat (torlódás);

72

Ütemezési algoritmusok 2.

Legrövidebb fut („Shortest Job First”): a futásra várakozó folyamatok közül az kerül sorra, amelynek a legrövidebb a processzoridő-igénye.

Előnye: a leggyorsabb folyamat-végrehajtást teszi lehetővé.

Hátránya: előfordulhatnak kizárt folyamatok valamint hogy a processzoridő-igény előre ismerni kell (az esetek többségében ez nem meghatározható, csak becsülhető érték!);

73

Ütemezési algoritmusok 3

Körben járó („Round Robin”): minden folyamat rendelkezésére áll egy időszelet, ameddig működhet, ennek leteltével a várakozási sor végére kerül.

Előnye: minden folyamat azonos ideig működhet (nem fordulhat elő feltartás vagy kizárás).

Hátránya: a folyamatok újra-sorbaállítása miatt arányosan növekszik a környezetváltások (és adminisztrációs műveletek) száma, így nő a folyamatok végrehajtásának ideje is.

74

Memóriakezelési módszerek

Valóságos tárkezelés: a számítógépben rendelkezésre álló fizikai memórián belüli műveletek.

Virtuális memória: a háttértárolón elhelyezkedő, az operatív tárral azonos szerepet betöltő tárolóterület.

Mivel a számítógép csak azokkal az adatokkal és utasításokkal képes műveletet végezni, amelyek az operatív tárban találhatók, ezért a **virtuális tárkezelés** lényege abban áll, hogy a processzor folyamatváltásaihoz hasonlóan ezek az információk is folyamatosan cserélődnek az operatív és a virtuális memória között.

75

Milyen problémákat kell megoldania az operációs rendszernek a tárkezeléssel kapcsolatban?

1. Kernelfolyamatok védelme
2. Folyamatok védelme egymástól
3. Egyidejűleg több folyamat
4. Rendelkezésre álló memóriánál nagyobb memóriaigényű folyamatok kezelése
5. Végrehajtás közben kialakuló memóriahivatkozások (címek) kezelése

76

Az operatív tár kezelésére szolgáló módszerek:

Rögzített címzés: az operációs rendszer a memória rögzített részén helyezkedik el. Az operációs rendszer számára fenntartott terület végét egy határregiszter rögzítette, az ellenőrzés mindössze azt jelentette, hogy felhasználói program nem hivatkozhatott ennél a címnél kisebb címre. A programok rögzített címeket használtak (az utasítás vagy az adat a program minden futása során ugyanabban a memóriarekeszben helyezkedett el).

77

Az operatív tár kezelésére szolgáló módszerek 2.

Áthelyezhető címzés: lényege, hogy az operációs rendszer mérete nem állandó, így a folyamatok számára rendelkezésre álló memóriaterület kezdőcíme minden futásnál megváltozhatott.
A programok relatív címek alkalmazásával hivatkoznak a tárolt információra: a program kódja nem a keresett adat memóriában elfoglalt helyét tartalmazza, csupán azt, hogy az adott utasítás vagy adat a program kezdetéhez képest hol helyezkedik el.

Fizikai cím: Bázis reg. értéke + hivatkozási (relatív) cím

78

Az operatív tár kezelésére szolgáló módszerek 3.

Átlapoló („overlay”) **technika**: lényege, hogy a programokat logikai részekre (**blokk**) osztjuk, és nem a teljes program, csak a végrehajtás alatt álló kódrészt tartózkodik az operatív memóriában. Előnye: lehetőséget biztosít az operatív memória méreténél nagyobb programok futtatására is.

Az aktuálisan használni kívánt modul kiválasztása a programozó feladata. Ezért a programoknak van egy vezérlő modulja, ami állandóan a memóriában tartózkodik, és eldönti, hogy mely blokkot kell betölteni.

79

Az operatív tár kezelésére szolgáló módszerek 4.

Tárcsere („swapping”): az egyes folyamatok közti váltáskor a teljes memória kiírásra kerül a háttértárra („swap-file”).

Előnye az egyszerűsége

Hátránya: a nagy mennyiségű adat kiírására-visszaolvasására fordítandó (a futási időhöz képest) hosszú idő.

Optimalizálható, ha csak a legutolsó mentés óta megváltozott memóriaterületeket írjuk ki a cserefájlba, ennek az adminisztrálása viszont lényegesen bonyolultabb.

80

Az operatív tár kezelésére szolgáló módszerek 5.

Állandó partíciókaz operatív memóriát egymástól független részekre osztják (**partíció**), amelyek mindegyike úgy viselkedik az egyes folyamatok számára, mintha teljes memória lenne.

Előnye: egy időben több folyamat is tartózkodhat a memóriában, lényegesen gyorsabb.

Hátránya: a partíciók mérete nehezen meghatározható: ha túl kicsi, nem fér el benne a folyamat; ha túl nagy, sok kihasználatlan hely marad az egyes partíciókon belül – ezt nevezzük ***belső elaprózódásnak***.

81

Az operatív tár kezelésére szolgáló módszerek 6.

Rugalmas partíciók: a partíciók mérete a folyamatok memória-igényének megfelelően kerül meghatározásra Az operációs rendszer nyilvántartja az egyes partíciók foglaltságát, a távozó folyamatok után fennmaradt „lyukakból” választ helyet, ahová az érkező folyamat belefér.

ez a módszer gyakorlatilag kiküszöböli a belső elaprózódást, viszont előfordulhat, hogy sok kicsi „memórialyuk” keletkezik ez a ***külső elaprózódás***.

82

Az operatív tár kezelésére szolgáló módszerek 7.

Lapozás (paging): A memóriát viszonylag kis méretű, azonos nagyságú részekre (**lapok**) osztva az egyes folyamatok a szabad lapok közül annyit foglalnak le, amennyire szükségük van.

A címzési eljárás tovább bonyolódik: nem elég azt tudni, hogy a folyamat kezdeti címéhez (bázisregiszter) képest mennyivel eltolva található a keresett információ, de azt is tudni kell, hogy ez a cím hányadik logikai lapra mutat, és ez a fizikai memória mely címtartományának felel meg.

83

Az operatív tár kezelésére szolgáló módszerek 8.

Laptábla: a folyamat betöltésekor az operációs rendszer által létrehozott leíró tábla, amely a folyamat logikai lapjainak megfelelő fizikai memóriacímeket tartalmazza.

A keresett adat logikai címét 2 részre osztjuk: az első rész megadja a lapszámot, a második rész a lapon belüli eltolás mértékét. A lapszám alapján a laptáblából meghatározható a lap fizikai kezdőcíme, amihez a hagyományos módon hozzáadva a lapon belüli eltolást, meghatározható a keresett memóriarekesz sorszáma.

84

Az operatív tár kezelésére szolgáló módszerek 9.

Virtuális memóriakezelés: folyamatok számára az operatív memória és a virtuális memória azonos módon használható.

Elve a lapozásos memóriakezelés, azonban az egyes lapok leírását ki kell egészíteni egy jelzőbittel, amely azt mondja meg, hogy az adott lap az operatív vagy a virtuális memóriában található. Amennyiben olyan lapra érkezik hivatkozás, amely a virtuális memóriában található (ezt nevezik **laphibának** – ez nem jelent valódi hibát!), akkor az operatív memória valamely lapja kikerül a virtuális memóriába, a virtuális memória hivatkozott lapja pedig betöltődik az operatív tárba.

85

Lapcsere-stratégiák

- **Optimális stratégia**: azt a lapot kell kicserélni, amelyre a legtávolabbi jövőben történik hivatkozás – mivel ez nem dönthető el egy folyamat futása során, ez csak elvi stratégia.
- **Előbb jött, előbb megy** („First In First Out” - FIFO): azt a lapot kell kicserélni, amelyik a legrégebben tartózkodik az operatív tárban.
- **Legrégebben használt** („Last Recently Used” - LRU): azt a lapot kell kicserélni, amelyre a legrégebben történt hivatkozás.

86

Lapcsere-stratégiák 2.

- **Második esély** („Second Chance”): módosított FIFO, csak akkor cseréli ki a legrégebben bent levő lapot, ha a folyamat a legutolsó lapcsere óta nem használta. Ehhez az egyes lapokhoz (a laptáblában) hozzárendelünk egy újabb jelzőbitet, amelyet a lap használatakor (ha hivatkozás történik a lapra) 1-esre állítunk, és ha laphiba esetén ez a lap lenne a kicserélendő, akkor a jelzőbitet 0-ra állítva betesszük a FIFO végére (mintha most töltődött volna be), és megnézzük a következő lapot (amíg olyan lapot nem találunk, amelynek a „használt”-bitje 0).

87

Lapcsere-stratégiák 3.

- **Mostanában használt** („Recently Used”): módosított LRU, csak akkor cseréli ki a legrégebben használtat, ha azt a legutolsó lapcsere óta nem használták. Elve az előző módszerhez hasonlóan a lap használatának jelzésén alapszik, laphiba esetén olyan lapot választunk, amelynek „használt”-bitje 0, de lapcserekor minden operatív tárban levő lap jelzőbitjét kinullázzuk (Így előzhető meg, hogy valamely lap „örökké” a memóriában maradjon.)

88

Lapcsere-stratégiák 4

	Igényelt lap											
	0	**2**	**4**	**2**	**3**	**4**	**1**	**2**	**4**	**0**	**3**	**2**
Operatív memóriában levő laphelyek												
OPT - 1. keret	0!	0	0	0	3!	3	1!	1	1	0!	0	0
2. keret		2!	2	2	2	2	2	2	2	2	2	2
3. keret			4!	4	4	4	4	4	4	4	3!	3
FIFO – 1. keret	0!	0	0	0	3!	3	3	3	4!	4	4	2!
2. keret		2!	2	2	2	2	1!	1	1	0!	0	0
3. keret			4!	4	4	4	4	2!	2	2	3!	3
LRU – 1. keret	0!	0	0	0	3!	3	3	2!	2	2	3!	3
2. keret		2!	2	2	2	2	1	1	1	0!	0	0
3. keret			4!	4	4	4	4	4	4	4	4	2!

89