

# Sisteme de Operare

## Administrarea perifericelor de stocare

**Cristian Vidrașcu**

<https://profs.info.uaic.ro/~vidrascu>

- Introducere
- Memoria secundară (discul magnetic)
  - Structura
  - Planificarea
  - Administrarea
- Gestiunea spațiului de swap
- Construirea unui disc mai bun
- Administrarea memoriei terțiare

- Clasificarea perifericelor (d.p.d.v. funcțional)
  - **Periferice de intrare/ieșire** – pentru schimbul de informații cu mediul extern  
(e.g. tastatură, ecran, imprimantă, cititor de cartele, ș.a.)
  - **Periferice de stocare** – pentru păstrarea nevolatilă (i.e. permanentă) a informațiilor  
(e.g. disc magnetic, bandă magnetică, dischetă, CD/DVD, ș.a.)

# Alte clasificări ale perifericelor /1

- Clasificarea perifericelor d.p.d.v. al modului de operare (de servire a cererilor)
  - **Periferice dedicate** – pot servi un singur proces la un moment dat  
(modelul tipic: imprimanta)
  - **Periferice partajabile** – pot servi mai multe procese simultan, în sensul concurenței aparente  
(modelul tipic: discul magnetic)

## Alte clasificări ale perifericelor /2

- Clasificarea perifericelor d.p.d.v. al modului de transfer și de memorare a informației
  - **Periferice bloc**
    - memorează informațiile în *blocuri* de lungime fixă, fiecare cu adresa sa
    - blocul este unitatea de transfer între periferic și memoria internă
    - fiecare bloc poate fi citit sau scris independent de celelalte blocuri
    - blocul conține informații propriu-zise și mai poate conține, eventual, informații de control de paritate, pentru verificarea corectitudinii informației memorate
    - e.g. disc magnetic, bandă magnetică, CD/DVD, memorii flash, ș.a.
  - **Periferice caracter**

## Alte clasificări ale perifericelor /3

- Clasificarea perifericelor d.p.d.v. al modului de transfer și de memorare a informației (cont.)
  - **Perifere bloc**
  - **Perifere caracter**
    - furnizează/acceptă un *flux* de octeți fără nici o structură de bloc
    - octeții din flux nu sunt adresabili
    - fiecare octet este disponibil ca și *character curent* numai până la apariția următorului octet
    - e.g. tastatură, mouse, ecran, imprimantă, difuzor, ș.a.

# Periferice de stocare

- Clasificarea perifericelor de stocare după variația timpului de acces ( $t_{ij}$  = timpul de acces de la informația  $i$  la informația  $j$ )
  - **Periferice cu acces secvențial** – timpul de acces  $t_{ij}$  are variații foarte mari (modelul tipic: banda magnetică)
  - **Periferice cu acces complet direct** – timpul de acces  $t_{ij}=k$ , este constant (modelul tipic: memoriile bidimensionale obișnuite – RAM)
  - **Periferice cu acces direct** – timpul de acces  $t_{ij}$  are variații foarte mici (modelul tipic: discul magnetic)

# Discul magnetic

## ➤ Memoria secundară: discul (intern)

Notă: tehnologia folosită a fost standardizată în ultimele aprox. 4 decenii: **discul magnetic**

- Structura discului
- Planificarea acceselor la disc
- Administrarea discului

Notă: în urmă cu aprox. un deceniu a apărut o tehnologie mai nouă: **discul SSD**, care se bazează pe diferite tipuri de memorii flash/NAND (această tehnologie încă nu a devenit standardizată, fiecare producător folosește propria implementare proprietară a tehnologiei NAND)



# Structura discului /1

- Discurile magnetice sunt adresate ca o matrice 1-dimensională (un vector) foarte mare de **blocuri logice**, unde blocul logic este cea mai mică unitate de informații ce se poate transfera (între disc și memorie)

Indexul unui bloc în acest vector este adresa sa LBA.

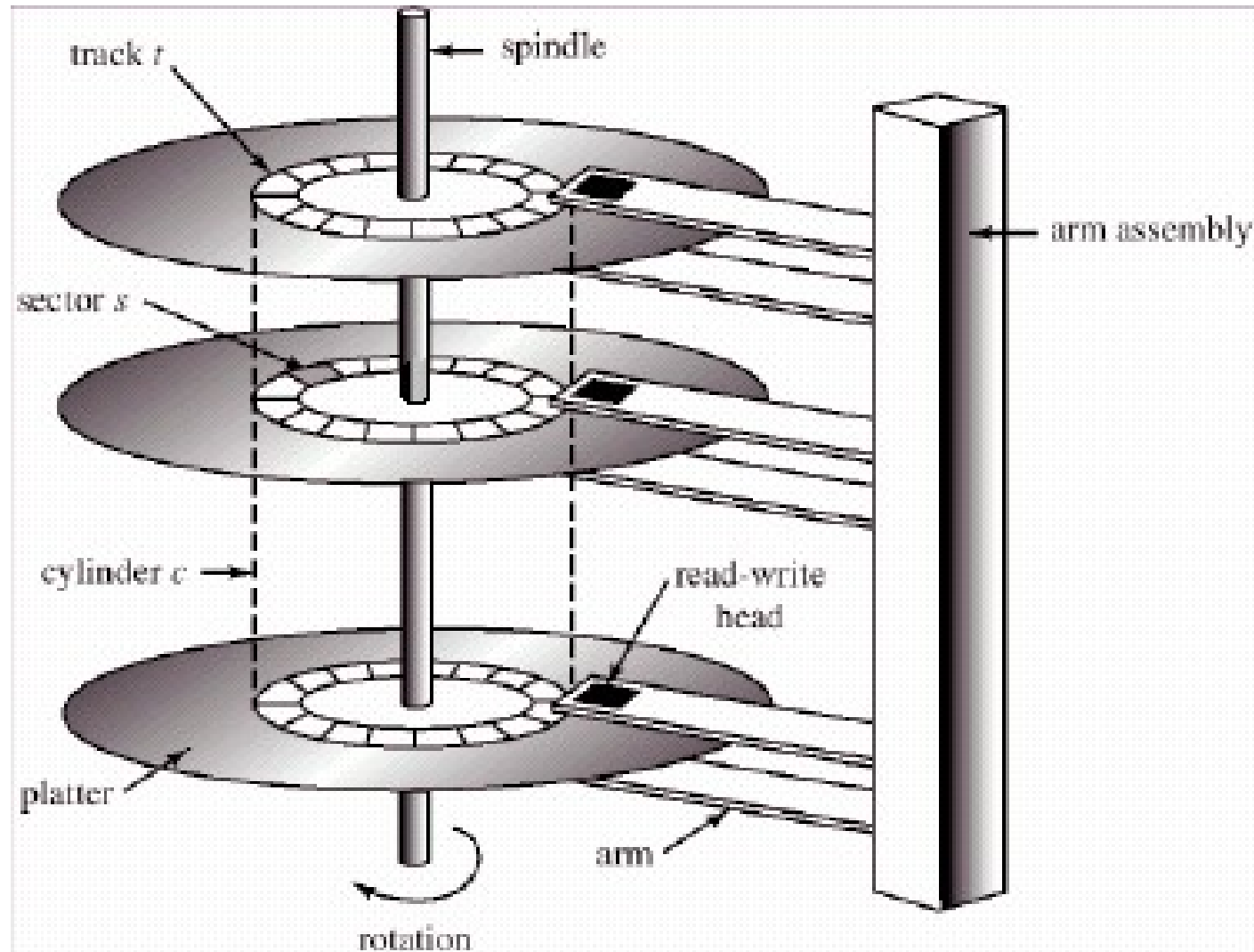
- Matricea 1-dimensională de blocuri logice este mapată secvențial pe sectoarele discului

Adresa LBA se convertește în adresa fizică (C,H,S), de către BIOS (în trecut) sau firmware-ul discului (în prezent).

# Structura discului /2

- Organizarea discului

- platane
- piste
- cilindri
- sectoare
- data



# Viteza discului

- **Timpul de căutare** (*Seek time*)
  - timpul necesar pentru mișcarea (mecanică a) capului de citire-scriere până la pista specificată
- **Latența de rotație** (*rotational Latency*)
  - timpul de așteptare necesar pentru ca începutul sectorului specificat să ajungă prin rotație sub capul de citire-scriere
- **Timpul de transfer** (*Transfer time*)
  - timpul necesar pentru a citi datele de pe sectorul specificat
- **Timpul total de acces** =  $S + L + T$

# Probleme

- **Planificarea acceselor la disc**
  - ideea este de a reorganiza cererile de acces la disc pentru a minimiza timpii de căutare (*seek*-urile)
- **Plasarea datelor pe disc (*Layout*)**
  - un mod de plasare care să minimizeze *overhead*-ul operațiilor cu discul
- **Construirea unui **disc mai bun** (sau a unui substituent pentru discurile actuale)**
  - exemplu: RAID

# Planificarea discului /1

- Sistemul de operare este responsabil pentru utilizarea eficientă a hardware-ului
  - pentru discurile hard, aceasta înseamnă a avea un timp de acces rapid și o lățime de bandă mare
- Lățimea de bandă a discului (*disk bandwidth*) este numărul total de octeți transferați, împărțit la timpul total scurs între prima cerere de serviciu și sfârșitul ultimului transfer solicitat

# Planificarea discului /2

- Timpul de acces are două componente majore:
  - *seek time* – timpul necesar discului pentru a muta capul pe cilindrul ce conține sectorul dorit
  - *rotational latency* – timpul adițional de așteptare pentru ca discul să rotească sectorul dorit sub capul de citire-scriere
  - cea de-a treia componentă, *timpul efectiv de transfer*, este o constantă specifică perifericului respectiv
- Planificarea discului urmărește **minimizarea timpului de căutare**, eventual și a latenței de rotație
- Ideea: schimbarea ordinii de servire a cererilor venite de la procesele concurente (cu păstrarea ordinii cererilor fiecărui proces)

# Planificarea discului /3

- Algoritmi de planificare (a acceselor la disc)
  - FCFS (First Come, First Served)
  - SSTF (Shortest Seek Time First)
  - SCAN
  - C-SCAN (Circular SCAN)
  - LOOK
  - C-LOOK (Circular LOOK)

# Planificarea discului /4

- Pentru exemplificările următoare ale algoritmilor de planificare, considerăm următorul scenariu:
  - un disc cu 200 cilindri (numerotați cu 0-199)
  - coada cererilor de acces (doar cilindrul ce conține sectorul dorit): 98, 183, 37, 122, 14, 124, 65, 67 (fiecare cerere provenind de la un proces distinct)
  - poziția inițială a furcii cu capetele de citire-scriere a discului: cilindrul 53

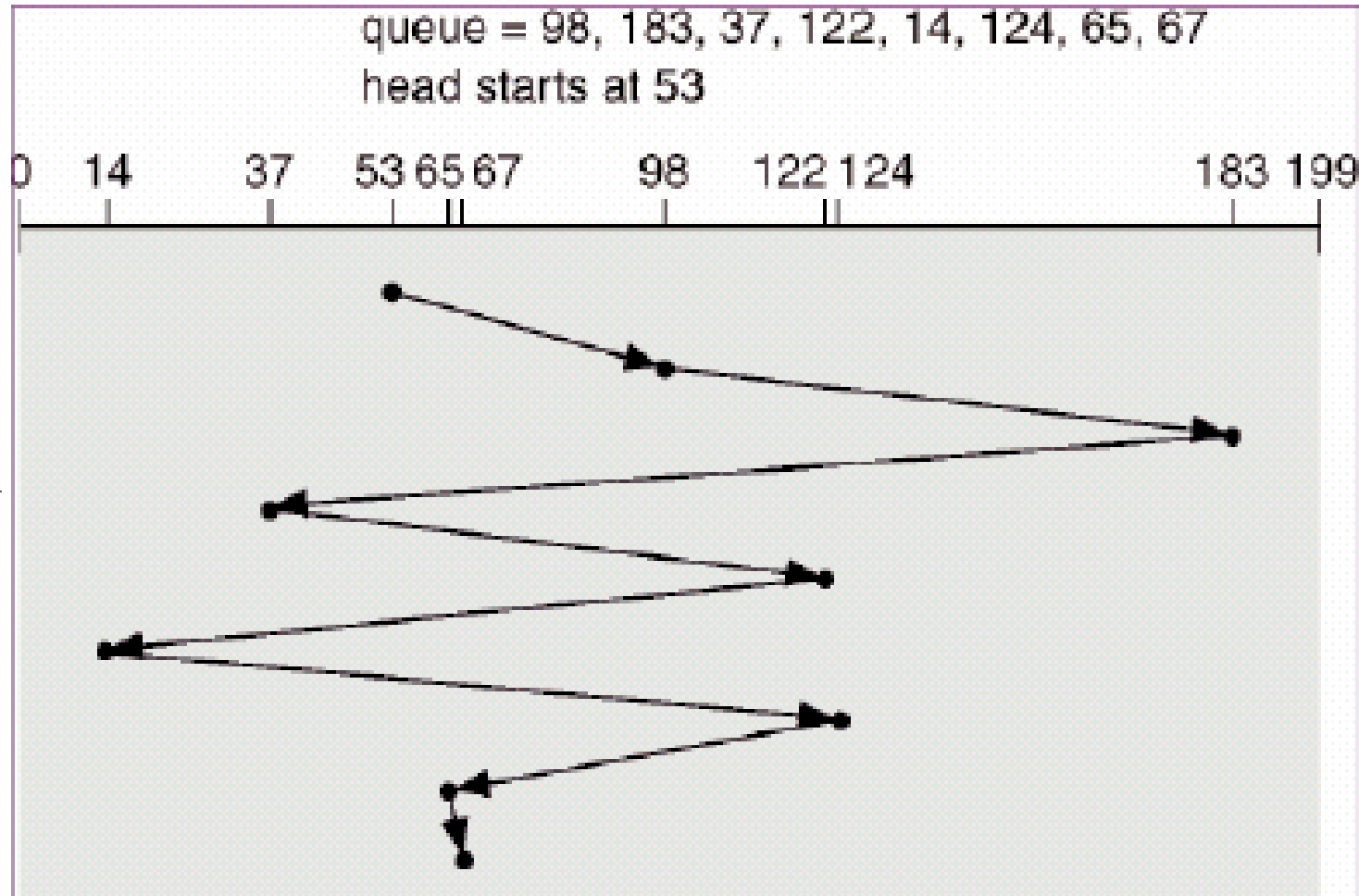


# Planificarea discului /5

- **FCFS** (First Come, First Served)

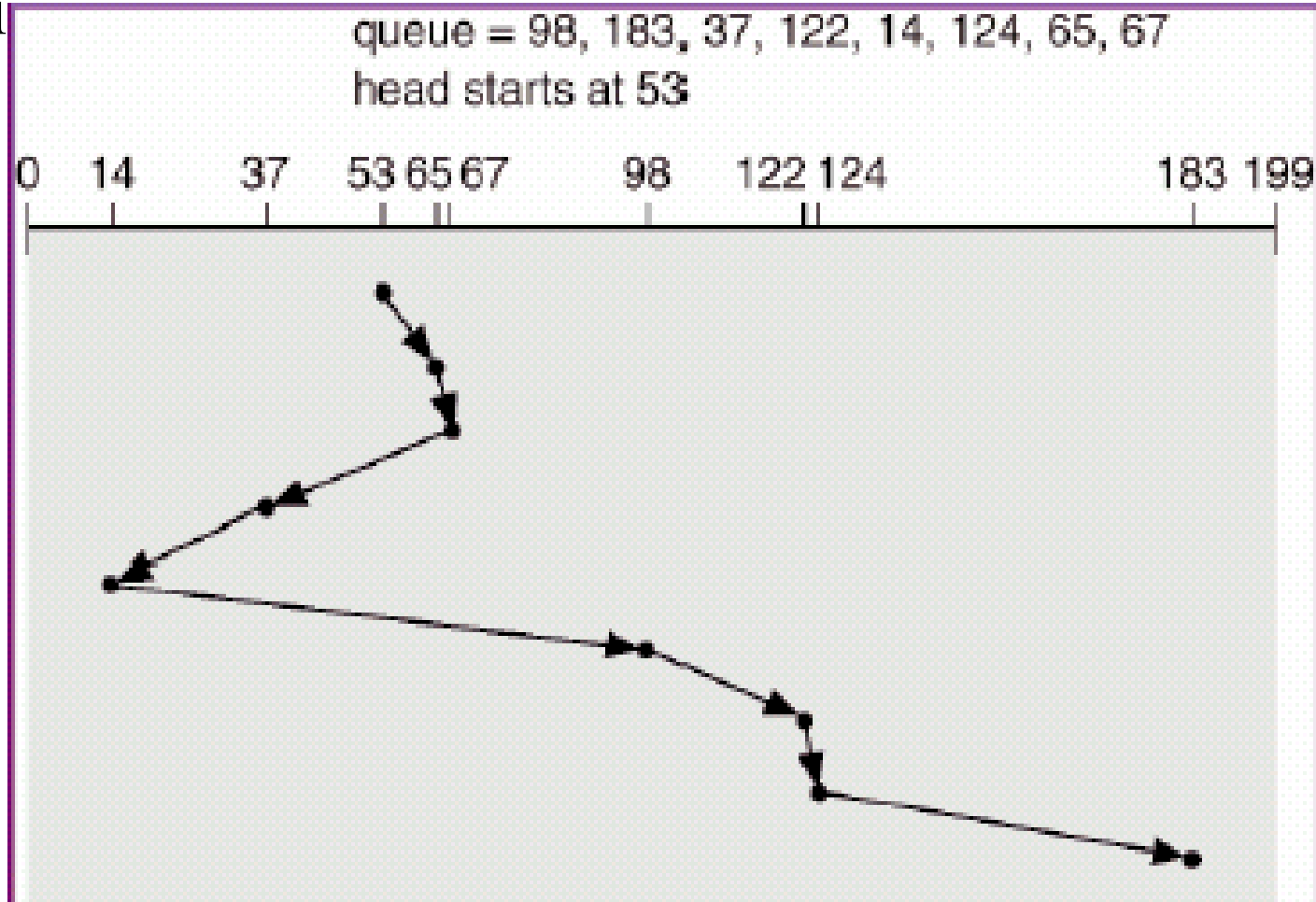
- Alg.: cererile sunt servite în ordinea sosirii
- Figura arată o mișcare totală a capului discului de 640 cilindri

Ipoteză de lucru:  
coada statică  
(i.e. **toate** cererile au ajuns în coadă la momentul  $t=0$ , în ordinea specificată)



# Planificarea discului /6

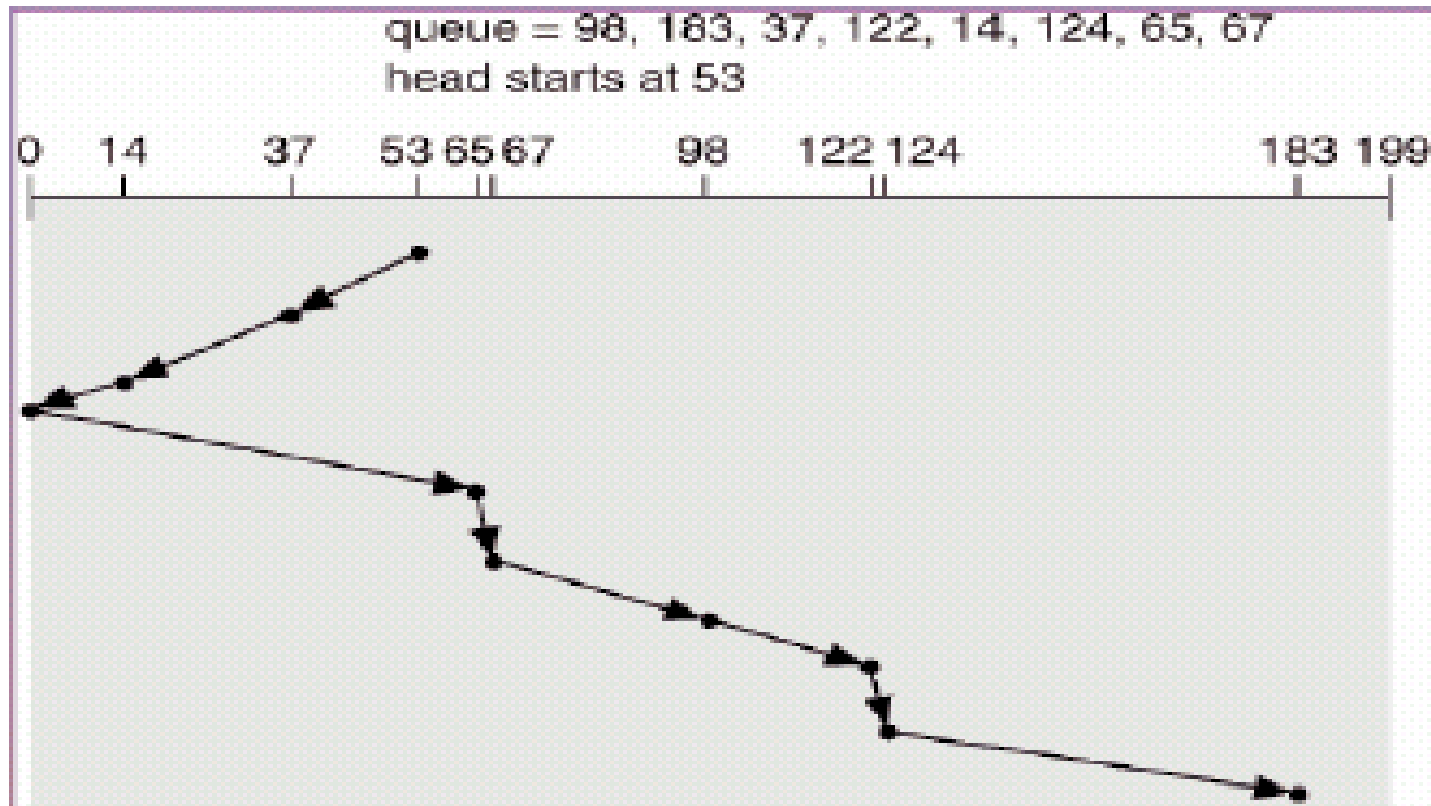
- **SSTF** (Shortest Seek Time First)
  - Alg.: se alege cererea cu timp de căutare minim de la poziția curentă a capului
  - Figura arată o mișcare totală a capului discului de 236 cilindri
  - Este mai eficient decât FCFS, dar nu este echitabil (poate favoriza fenomenul de *starvation*: întârzierea servirii unor cereri)



# Planificarea discului /7

- **SCAN** (algoritmul elevator)
  - Alg.: brațul cu capetele R/W începe la un capăt al discului și se deplasează spre celălalt capăt, rezolvând și cererile pe parcurs, iar când ajunge la celălalt capăt, se întoarce înapoi, continuând servirea cererilor

- Figura arată o mișcare totală a capului discului de 236 cilindri (în ipoteza că sensul inițial de deplasare era în jos. Altfel, mișcarea totală este de 331 cilindri).



# Planificarea discului /8

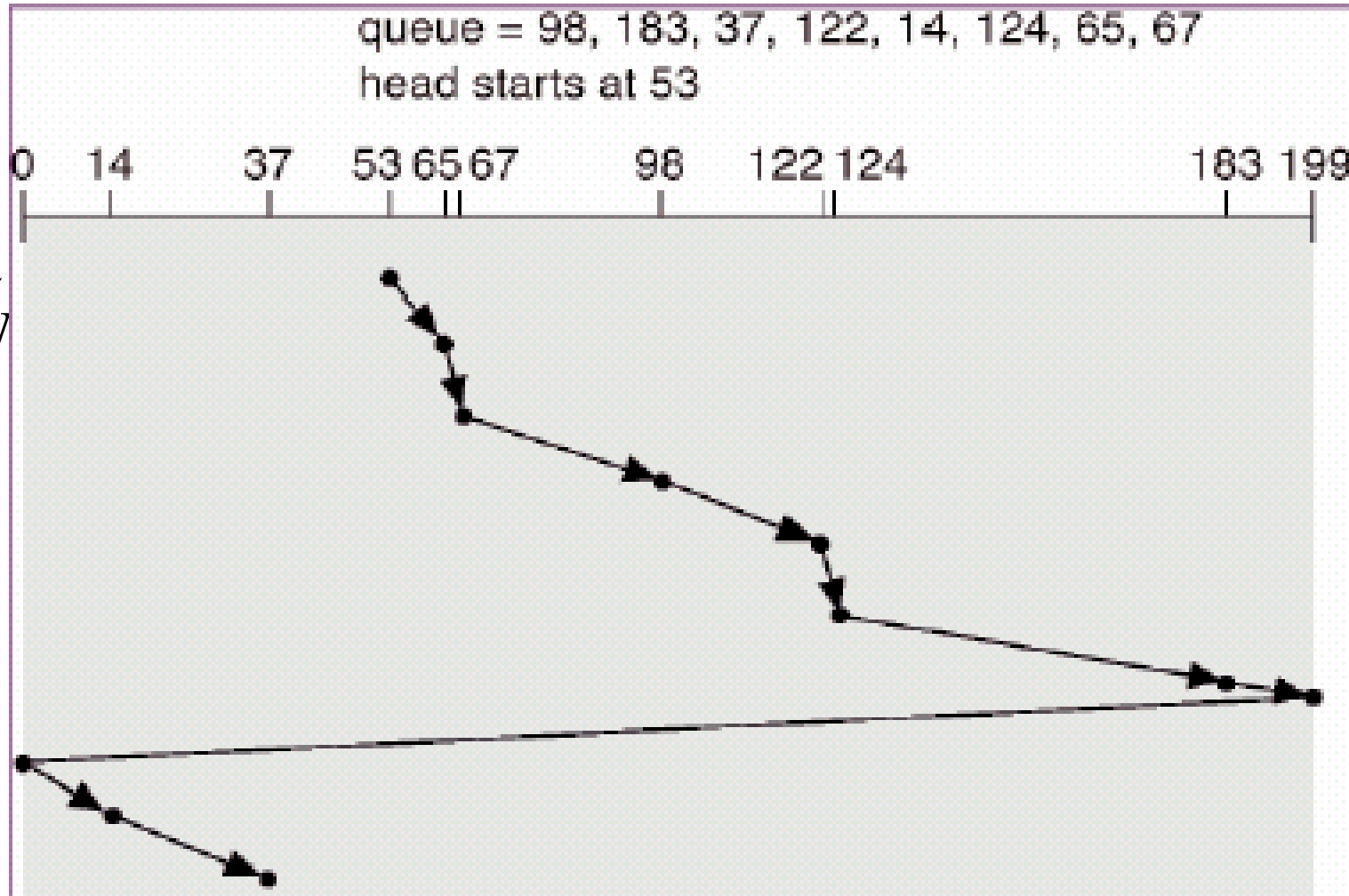
- **C-SCAN** (Circular SCAN)
  - Alg.: brațul cu capetele R/W începe la capătul 0 al discului și se deplasează spre celălalt capăt, rezolvând și cererile pe parcurs, iar când ajunge la celălalt capăt, se întoarce imediat (*foarte rapid*) înapoi la începutul discului, fără să servească nici o cerere pe drumul de întoarcere, și apoi reia lucrul
  - Practic, acest algoritm tratează cilindrii discului ca o listă circulară care “conectează” ultimul cilindru cu primul cilindru
  - Avantaj: furnizează un timp de așteptare (a rezolvării cererii) mai uniform decât algoritmul SCAN (la care este posibil ca o cerere să aștepte două parcurgeri ale discului până când este servită)

# Planificarea discului /9

- **C-SCAN** (Circular SCAN)

- Exemplu

- Figura arată o mișcare totală a capului R/W a discului de 382 cilindri



# Planificarea discului /10

- **LOOK și C-LOOK** (Circular LOOK)

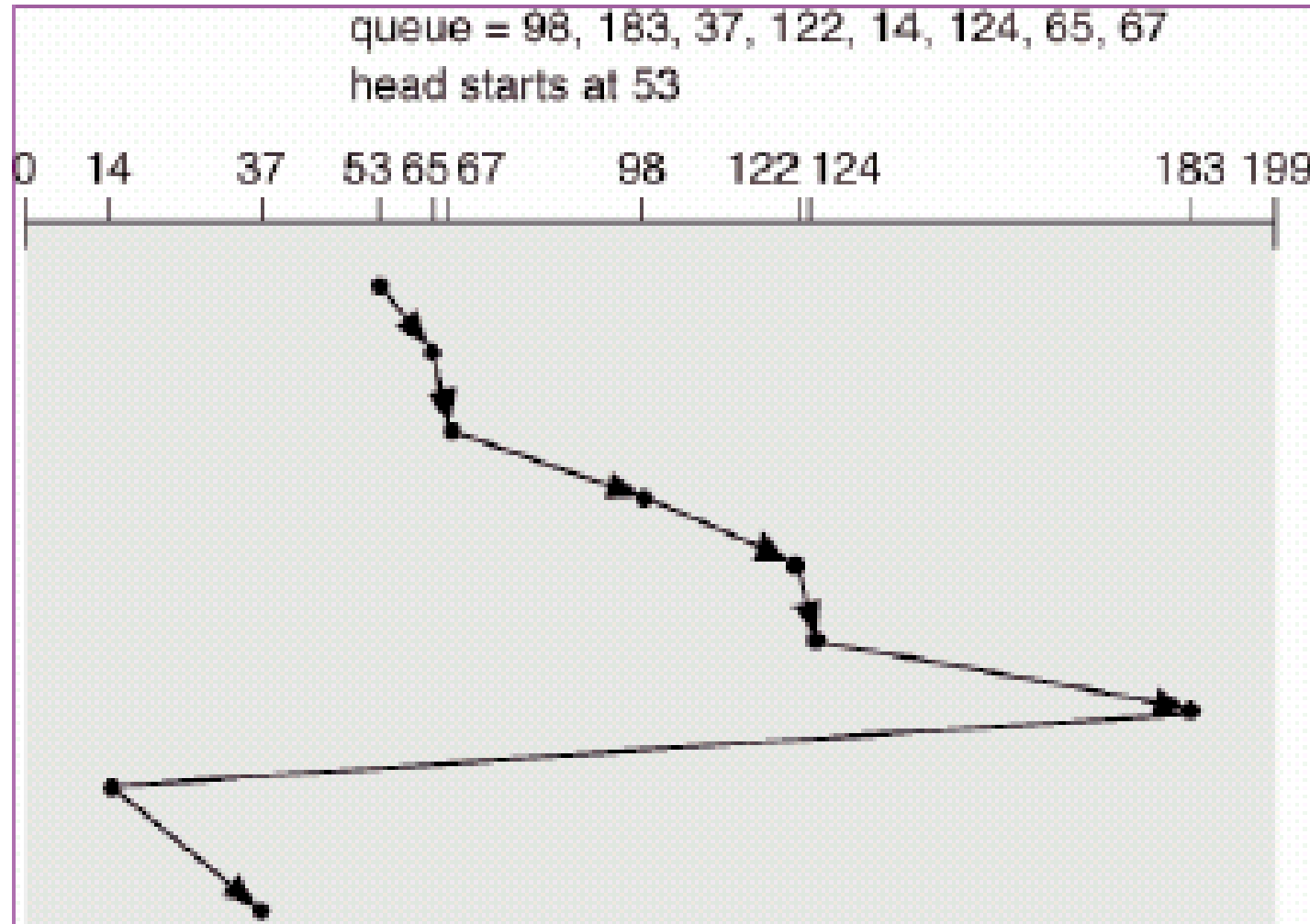
- Alg.: brațul cu capetele R/W se mișcă la fel ca la algoritmi SCAN și respectiv C-SCAN, cu deosebirea că aici se mișcă doar până la ultima cerere în fiecare sens, după care inversează direcția de deplasare imediat, fără să meargă mai întâi până la capătul discului
- Practic, LOOK/C-LOOK reprezintă o optimizare a algoritmilor SCAN/C-SCAN

# Planificarea discului /11

- **LOOK și C-LOOK** (Circular LOOK)

- Exemplu

- Figura arată o mișcare totală a capului discului de 322 cilindri pentru C-LOOK (pentru LOOK ar fi 276 cilindri)



# Planificarea discului /12

- **Selectarea unui alg. de planificare a discului**
  - SSTF este utilizat frecvent (dezavantaj: pericol de înfometare a unor cereri de acces la disc)
  - SCAN și C-SCAN se comportă mai bine pentru sisteme care au o încărcare mare a discului (i.e. multe operații I/O)
  - Algoritmul de planificare a discului este indicat să fie scris ca un modul separat al sistemului de operare, pentru a permite o înlocuire ușoară a sa cu un alt algoritm dacă se consideră necesar
  - Fie SSTF, fie LOOK este o alegere rezonabilă pentru algoritmul implicit de planificare a discului

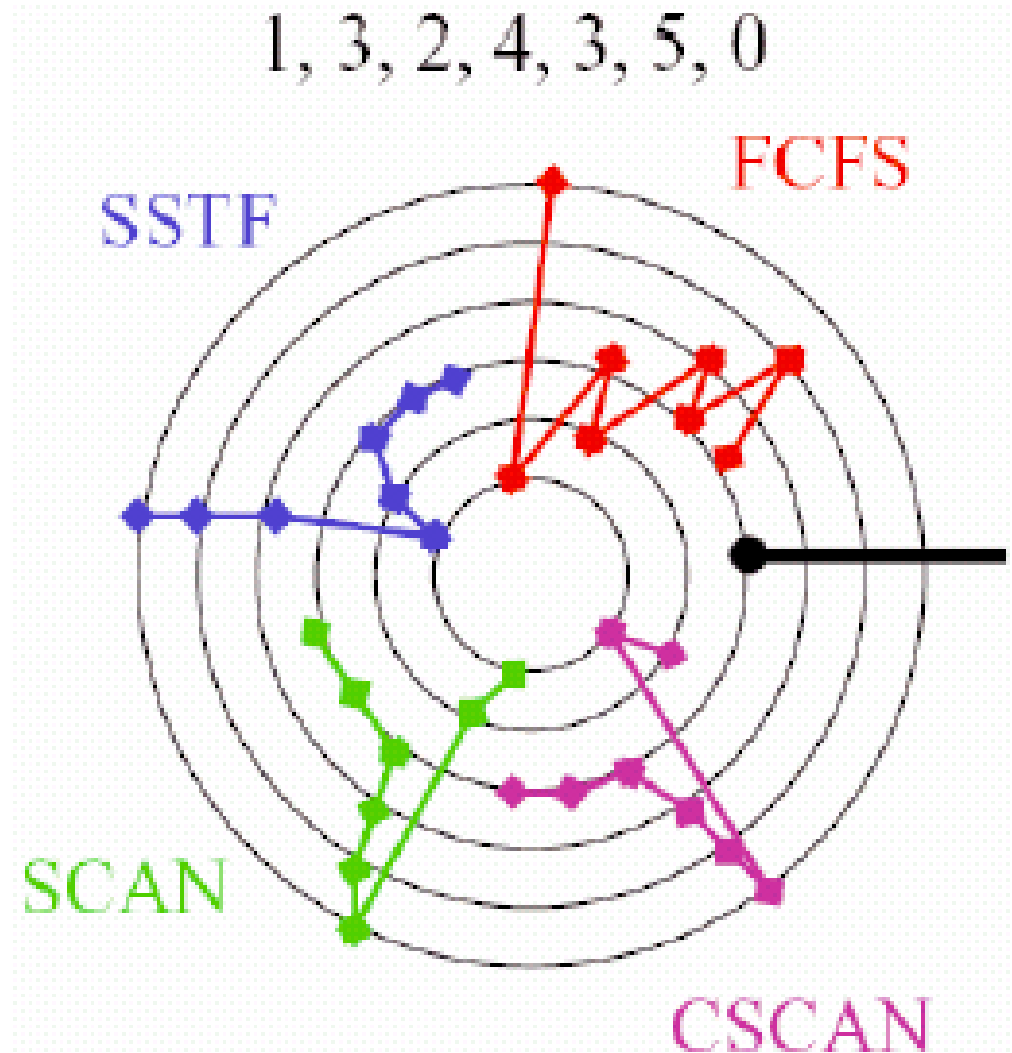


# Planificarea discului /13

- **Observații**

- Performanța depinde de numărul și tipurile cererilor de acces
- Cererile de acces pot fi influențate de metoda de alocare a fișierelor utilizată

Alt exemplu:



- **Observații** (cont.)

- De exemplu, la alg. SSTF cele mai favorizate d.p.d.v. al timpului de acces vor fi pistele din mijloc; ca urmare se pot aloca pe aceste piste fișierele cu frecvență ridicată de utilizare, sau structura de directoare (deoarece este folosită frecvent)
- La fel, la o recompactare (defragmentare) a discului, se pot aloca în aceste zone favorizate fișierele pentru care s-a constatat că au un grad de folosire mai ridicat

# Planificarea discului /15

## • Observații (cont.)

- în exemplele anterioare am simplificat expunerea, presupunând coada ca fiind statică (i.e. **toate** cererile ajungeau în coadă la momentul  $t=0$ )
- în realitate, coada este dinamică (i.e. **noi** cereri ajung în coadă pe parcursul trecerii timpului); iată un exemplu în acest sens:

Coada de cereri: 20, 45, 85 (toate ajung la  $t=0$ ) ; 120, 97, 72 (toate la  $t=100$  ms)

Se dau: *seek time* = direct proporțional cu distanța (2 ms/traversarea a 2 cilindri consecutivi); *latența de rotație*: în medie, 1 ms (pentru toate cererile); *timpul de transfer* = neglijabil.

Inițial, capul R/W este la cilindrul 50, iar sensul de deplasare este spre cilindrul cu nr. maxim.

Ordinea de servire a celor 6 cereri, folosind algoritmul LOOK, este:

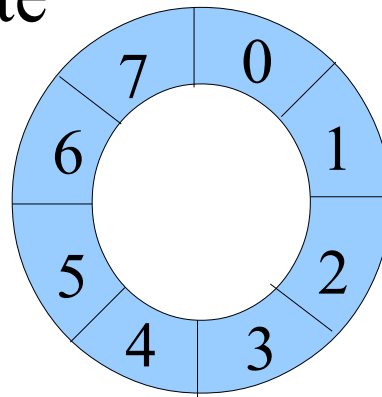
- 1) la momentul  $t=0$ , capul R/W este poziționat la cilindrul 50;
- 2) la momentul  $t=70$ ms, capul ajunge la cilindrul 85, iar după 1ms începe servirea cererii respective;
- 3) la momentul  $t=151$ ms, capul ajunge la cilindrul 45, iar după 1ms începe servirea cererii respective;
- 4) la momentul  $t=202$ ms, capul ajunge la cilindrul 20, iar după 1ms începe servirea cererii respective;
- 5) la momentul  $t=285$ ms, capul ajunge la cilindrul 72, iar după 1ms începe servirea cererii respective;
- 6) la momentul  $t=358$ ms, capul ajunge la cilindrul 97, iar după 1ms începe servirea cererii respective;
- 7) la momentul  $t=405$ ms, capul ajunge la cilindrul 120, iar după 1ms începe servirea cererii respective.

# Planificarea discului /16

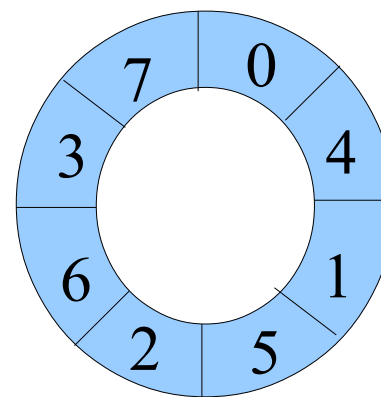
- **Reducerea așteptării rotației**

- În S.O.-urile mai pretențioase se urmărește și minimizarea latenței de rotație
- Ideea: reordonarea servirilor cererilor existente la un moment dat pentru același cilindru – alg. **SLTF (Shortest Latency Time First)**: *cel ce așteaptă cel mai puțin va fi servit primul*
- Altă metodă (statică): numerotarea *întreșesută* a sectoarelor în cadrul unei piste

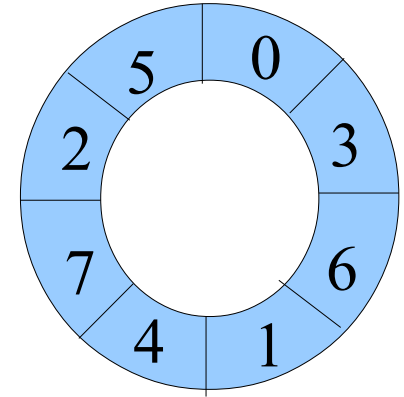
- a) numerotare normală
- b) numerotare întreșesută cu factorul 1
- c) numerotare întreșesută cu factorul 2



a)



b)



c)

# Administrarea discului /1

- **Formatarea fizică** (formatarea *low-level*)
  - Înseamnă operația de împărțire a discului în sectoare pe care controlerul de disc le poate citi și scrie
  - *Observație*: vreme de cca. 3 decenii (anii '80 – 200x) dimensiunea sectoarelor a fost de **512 B** (doar informația utilă), iar prin anii 200x au apărut discuri cu sectoare de **4096 B** (discuri numite *Advanced Format*) și au devenit standard pentru toți producătorii începând cu 2011
  - “Tranziția” de la discurile cu sectoare de 512B la cele cu sectoare de 4KB a durat aproape un deceniu, datorită nevoii de a rescrie multe componente software ce foloseau implicit sectoare de 512B
  - Primele modele de discuri AF permiteau *emularea* sectoarelor de 512B pentru a ușura tranziția, având însă și un dezavantaj: emularea era “costisitoare” (d.p.d.v. al performanței acceselor la disc pentru operații de scriere)

Referințe: [https://en.wikipedia.org/wiki/Advanced\\_Format](https://en.wikipedia.org/wiki/Advanced_Format)

și [Western Digital's Advanced Format: The 4K Sector Transition Begins](#)

# Administrarea discului /2

- Pentru a stoca fișiere pe disc, S.O.-ul trebuie să-și înregistreze propriile structuri de date pe disc
  - **Partiționarea** discului într-unul sau mai multe grupuri de cilindri, numite partiții, fiecare partiție putând stoca un sistem de fișiere
- Notă:* există mai multe scheme de partiționare a unui disc:
  - MBR: standardul folosit de PC-urile mai vechi (cu firmware BIOS)
  - GPT: standardul folosit de PC-urile mai noi (cu firmware UEFI)
  - ș.a.
- **Formatarea logică** a unei partiții = “crearea sistemului de fișiere” rezident pe acea partiție
- Blocul de boot folosit pentru inițializarea sistemului de calcul:
  - *Bootstrap*-ul este păstrat în memoria ROM
  - programul *bootstrap loader* este păstrat pe disc în **blocul de boot**

# Administrarea discului /3

- **Schema de partiționare MBR**

- este standardul de partiționare folosit de PC-urile cu BIOS
- un disc partiționat după această schemă va conține:
  - prima pistă este zonă rezervată (63 sectoare la discurile cu 512B/sector)
  - maxim 4 **partiții primare**, din care una poate fi **partiție extinsă** (i.e. poate conține un număr oarecare de **partiții logice**)
  - primul sector din cadrul primei piste se numește **sectorul MBR** (= Master Boot Record) și reprezintă blocul de boot, ce conține următoarele informații:
    - primii 446 octeți: conțin programul *bootstrap loader*
    - următorii 64 octeți: conțin tabela de partiții (cu informații despre poziția pe disc a celor 4 partiții primare și tipul sistemelor de fișiere stocate)
    - ultimii 2 octeți conțin întotdeauna valoarea 55AA (cu rol de semnătură)
  - restul de 62 sectoare din cadrul primei piste sunt rezervate (pentru programe de bootstrap mai mari de 446 octeți, e.g. **grub stage2**)

# Administrarea discului /4

- Alocarea blocurilor pe disc poate adresa ambele probleme, a timpului de căutare și a latenței de rotație
- Alocarea blocurilor urmărește scopuri competitive:
  - costul alocării
  - lățimea de bandă pentru transferul unor volume mari de date
  - eficiența operațiilor cu directoare
- **Scop:** reducerea mișcării brațului (cu capetele R/W ale) discului și a *overhead*-ului datorat *seek*-urilor
  - metrica utilizată: lățimea de bandă utilizată
- Metode de gestiune a blocurilor bad (e.g. [sector sparing](#))



# Administrarea discului /5

- O abordare posibilă pentru alocarea blocurilor:
  - Grupurile de cilindri utilizate de FFS (Fast File System)
    - FFS definește grupurile de cilindri drept unitatea de localitate a discului și factorizează localitatea în posibilități de alegere pentru alocare
    - **Strategia:** plasarea blocurilor de date “înrudite” în același grup de cilindri ori de câte ori acest lucru este posibil

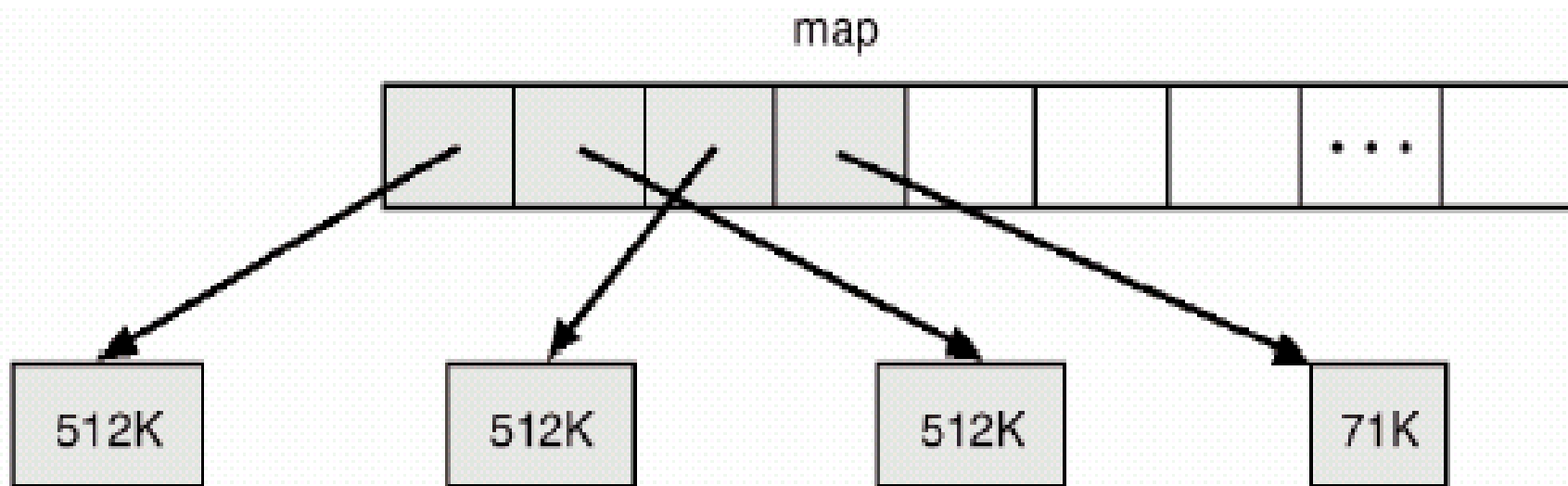
# Gestiunea spațiului de swap /1

- **Spațiul de swap**
  - Memoria virtuală utilizează spațiu pe disc drept o extensie a memoriei principale
- Spațiul de swap poate fi localizat:
  - într-un fișier (sau mai multe) din sistemul normal de fișiere (e.g. Windows)
  - pe o partiție (sau un disc) separată (e.g. UNIX/Linux)

# Gestiunea spațiului de swap /2

- **Exemplu:**

- UNIX BSD 4.3 îi alocă spațiu de swap la începutul execuției procesului
- spațiul alocat conține:  
segmentul de text (programul) și segmentul de date



# Construirea unui disc mai bun /1

- **De ce?**

- “Mai bun” a însemnat de obicei o densitate mai mare pentru producătorii de discuri – discurile mai mari sunt mai bune
- *I/O bottleneck* – discrepanța de viteză cauzată de faptul că procesoarele devin mai rapide mult mai repede decât discurile
- O idee este de a folosi paralelismul mai multor discuri
  - **Împrăștierea datelor** (*data striping*) pe mai multe discuri
  - Probleme de siguranță a păstrării datelor – introducerea tehnicilor de **redundanță** a datelor

# Construirea unui disc mai bun /2

- **Soluție**

- **RAID (Redundant Array of Independent Disks)**

- Discurile multiple asigură **siguranța** păstrării datelor prin **redundanța** datelor
- Discurile RAID se clasifică în 7 nivele RAID
- *Striping*-ul utilizează un grup de discuri ca o singură unitate de stocare
- Schemele RAID îmbunătățesc performanța și siguranța sistemului de stocare prin stocarea redundantă a datelor
  - prin tehnica oglindirii (*mirroring* sau *shadowing*) se păstrează un duplicat al fiecărui disc
  - tehnica *block interleaved parity* folosește mult mai puțină informație pentru redundanță

# Un disc mai bun /3

- **Nivelele RAID**



(a) RAID 0: non-redundant striping



(b) RAID 1: mirrored disks



(c) RAID 2: memory-style error-correcting codes



(d) RAID 3: bit-interleaved Parity



(e) RAID 4: block-interleaved parity



(f) RAID 5: block-interleaved distributed parity



(g) RAID 6: P + Q redundancy

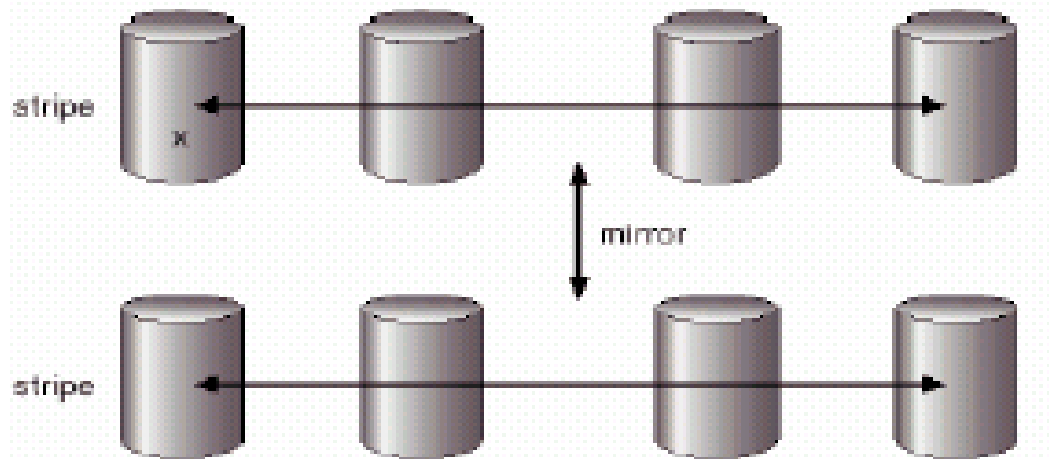
# Construirea unui disc mai bun /4

- **Nivelele RAID**

- Nivelul 0: fără redundanță, doar *striping*
- Nivelul 1: discuri oglindite
- Nivelul 2: coduri Hamming corectoare de erori
- Nivelul 3: un disc de paritate la fiecare grup, *bit-interleaved*
- Nivelul 4: citiri/scrieri independente, *block-interleaved*
- Nivelul 5: datele/informația de paritate sunt împrăștiate pe toate discurile (mărește accesul concurent)
- Nivelul 6: rezistă la mai mult de o singură eroare de disc

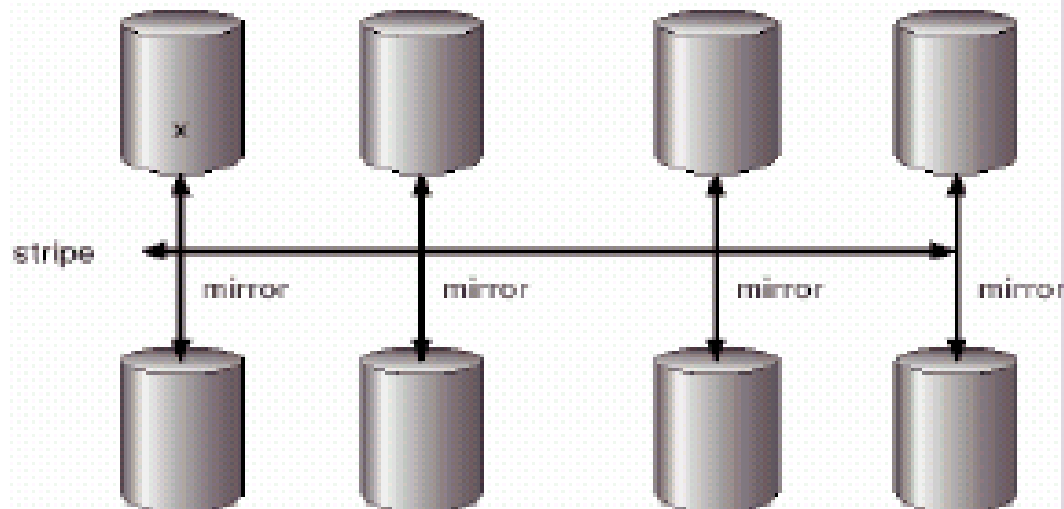
# Construirea unui disc mai bun /5

- **RAID (0+1)**



a) RAID 0 + 1 with a single disk failure

- **RAID (1+0)**

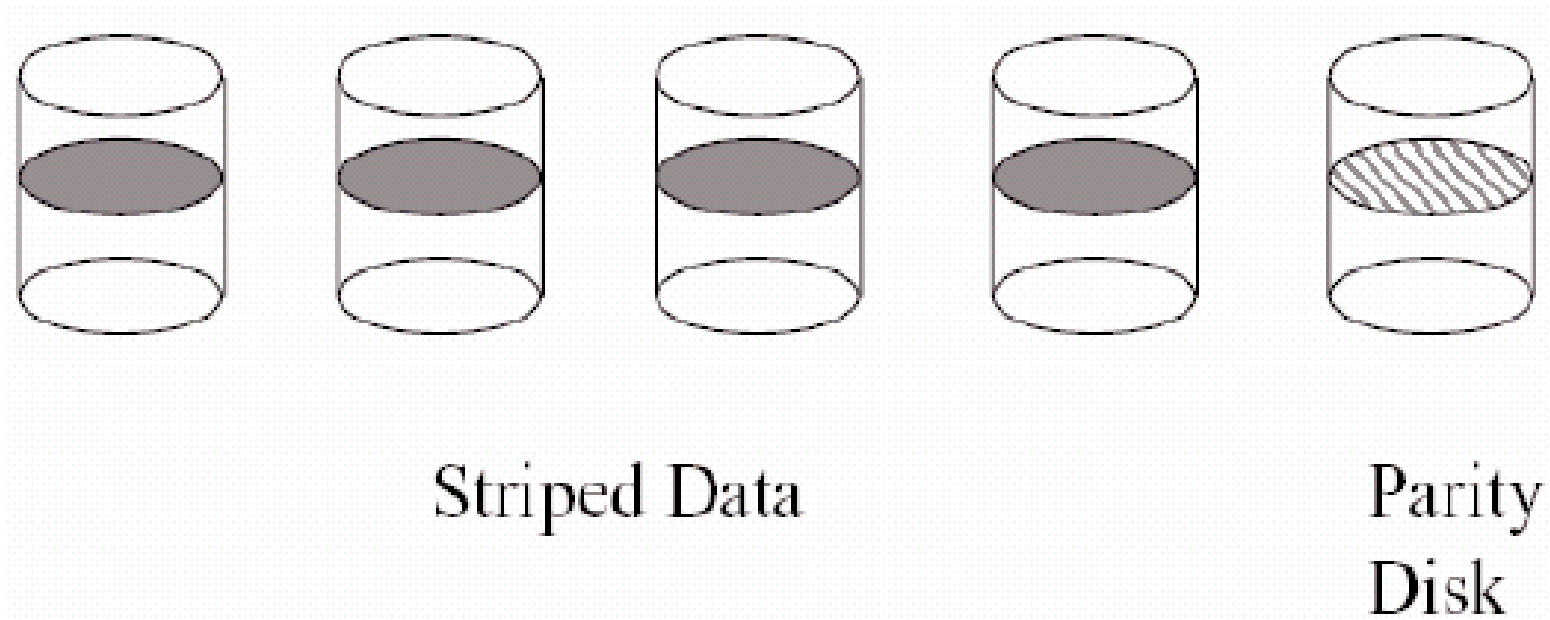


b) RAID 1 + 0 with a single disk failure



# Construirea unui disc mai bun /6

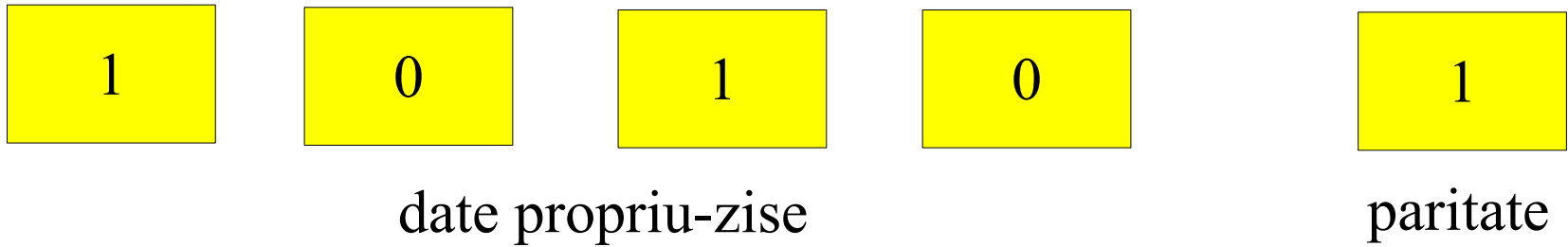
- **RAID – nivelele 2 și 3**



# Construirea unui disc mai bun /7

- **RAID : Paritatea**

- paritate bit/block-interleaved pentru toleranță la erori
- exemplu: stocarea valorii binare 1010



- Tolerază eroarea unui singur disc, întrucât eroarea este fie pe discul de paritate, fie pe unul dintre discurile de date
- Citește data, iar în caz de eroare, citește informația de paritate și corectează data
- Scrierea necesită în plus și actualizarea informației de paritate

# Construirea unui disc mai bun /8

- **RAID – implementare hardware:**

- controlerul RAID este o placă hardware (fie discretă, fie este integrată în northbridge, la unele modele de plăci de bază)
- plus un set de discuri, cu caracteristici specifice nivelului RAID dorit

- **RAID – implementare software:**

- controlerul RAID este implementat prin software
- plus un set de discuri, cu caracteristici specifice nivelului RAID dorit

Exemple de implementări software:

- un *device* virtual, la nivelul nucleului SO: md (Linux), softraid (OpenBSD)
- some logical *volume managers*: LVM (Linux), discuri dinamice create cu unealta Disk Management (Windows), sau tehnologia Storage Spaces (Win8/Win10)
- o componentă la nivelul *file-system*-ului: ZFS, btrfs (vezi cursul precedent)
- un nivel/o aplicație deasupra *file-system*-ului: SnapRAID, FlexRAID, etc.

Implementări la nivel de firmware+drivere specializate: e.g. Intel Matrix RAID

Referință: <https://en.wikipedia.org/wiki/RAID#Implementations>

# Administrarea memoriei terțiare /1

- Caracteristica definitorie a memoriei terțiare: **costul scăzut**
- În general, memoria terțiară este construită folosind medii de stocare de tip *removable* (e.g. dischete, CD/DVD-uri, memorii flash, discuri clasice (i.e. magnetice) externe)
- Sunt disponibile și alte tipuri ... (e.g. discuri SSD externe)

# Administrarea memoriei terțiare /2

- **Discuri *removable*:**
  - **Discul floppy** (discheta) – un disc subțire și flexibil, acoperit cu un strat de material magnetic și închis într-o carcasă protectivă de plastic
  - **Discul magneto-optic** – înregistrează datele pe un platan rigid acoperit cu un strat de material magnetic
  - **Discul optic** – nu utilizează magnetismul; se folosesc materiale speciale ce sunt modificate de raza laser
- **Medii WORM** (Write Once, Read Many times)
  - Exemple: **CD-uri**, **DVD-uri**
- **Benzi magnetice**
- **Memorii flash** (diverse tehnologii de tip NAND)
  - Exemple: **stick-uri USB**, **disk-uri SSD**, **card-uri de memorie** (în diverse formate: **CF**, **SD**, **MMC**, ș.a.)

- **Bibliografie obligatorie**

capitolele despre *administrarea perifericelor de stocare* din

- Silberschatz : “*Operating System Concepts*”

(cap.10 din [OSC9])

sau

- Tanenbaum : “*Modern Operating Systems*”

(cap.5, §5.4 din [MOS3])

# Sumar

- Introducere
- Memoria secundară (discul)
  - Structura
  - Planificarea
  - Administrarea
- Gestiunea spațiului de swap
- Construirea unui disc mai bun
- Administrarea memoriei terțiare

Întrebări ?