CHƯƠNG 8_1: GIAO TIẾP VỚI HỆ THỐNG FILE

- Tổng quan về dữ liệu và file
- Các thuộc tính & thao tác trên file
- Các phương pháp truy cập file
- Tổ chức thư mục
- Mount hệ thống file
- Bảo vệ hệ thống file
- Sao lưu và phục hồi dữ liệu

TỔNG QUAN VỀ DỮ LIỆU & FILE

- **Yêu cầu lưu trữ của user**
  - Lưu trữ lâu dài
  - Truy cập nhanh
  - Lưu đựoc nhiêu dữ liệu
  - Chia xẻ và bảo vệ tốt
  - Dễ sử dụng

➔ **cần sự hỗ trợ của phần cứng và OS**

- **Khái niệm file (tập tin, tệp)**
  - Đơn vị lưu trữ luận lý của OS
  - Phân loại: chương trình hoặc dữ liệu
  - Có thể có/ không có cấu trúc:

CÁC THUỘC TÍNH & THAO TÁC TRÊN FILE

- **Thuộc tính file (*file attribute*)**
  - Tên, kiểu, vị trí lưu trũ , kích cỡ, thông tin bảo vệ…
- **Thao tác về dữ liệu trên file (*data operation*)**
  - create, write, read, seek, delete, truncate
  - open(F_i)
  - close (F_i)
- **Thao tác về đặt tên file (*naming operation*)**
  - Tạo hard link, soft link, rename,
  - Thiết lập thuộc tính, lấy thuộc tính

CẤU TRÚC DỮ LIỆU QUẢN LÝ FILE

- **Bảng thông tin về các file đang mở (*Open File Table*).**
  - Dành cho n quá trình dùng chung một file
  - Chứa: biến đếm sử dụng, thuộc tính file, vị trí file trên đĩa, con trỏ đến vị trí của file trong bộ nhớ.
- **Bảng thông tin về các file của từng quá trình (*Per-process File Table*):** Với mỗi file, bảng này chứa:
  - Con trỏ đến mục tương ứng trong Open File Table
  - Vị trí hiện tại trong file
  - Chế độ truy cập của quá trình với file (r, w, rw)
  - Con trỏ tới file buffer

TÁC VỤ FILE (1)

- **Tạo file:** Create(name)
  - Cấp không gian lưu trữ
  - Tạo file descriptor chứa thông tin quản lý file
  - Thêm file descriptor vào thư mục chứa file
- **Xoá file:** Delete(name)
  - Tìm thư mục chứa file
  - Giải phóng các khối đĩa dành cho file
  - Xoá file descriptor khỏi thư mục chứa file
- **Mở file:** file_id = Open(name, mode)
  - Kiểm tra file có mở hay chưa → chia xẻ file.
  - Kiểm tra quyền sử dụng file.
  - Tăng open count của file.
  - Tạo và thêm thông tin quản lý file đang mở vào bảng file của hệ thống và của quá trình.
- **Đóng file:** Close(file_id) ?

TÁC VỤ FILE (2)

- **Đọc file:**
  - Read(file_id, from, size, buf_addr) : đọc ngẫu nhiên
  - Read(file_id, size, buf_addr) : đọc tuần tự
- **Ghi file:**
  - Tương tự đọc file
  - Thực hiện copy dữ liệu từ buffer vào file
- **Seek:**
  - Cập nhật vị trí con trỏ file
- **Ánh xạ file vào bộ nhớ (*memory mapping a file*):**
  - Ánh xạ 1 vùng địa chỉ ảo vào nội dung file
  - Tác vụ đọc/ ghi lên vùng nhớ ⇔ đọc/ ghi file

CÁC PHƯƠNG PHÁP TRUY CẬP FILE

- **Theo quan điểm người lập trình**
  - **Tuần tự**: xử lý dữ liệu (byte, record…) theo trật tự
  - **Theo khoá**: tìm khối dữ liệu theo giá trị khóa
- **Theo quan điểm hệ điều hành**
  - **Truy cập tuần tự** (*sequential access*): giữ và cập nhật con trỏ đến vị trí truy cập kế tiêp trong file
  - **Truy cập trực tiếp** (*random access):* truy cập dữ liệu theo offset của khôi dữ liệu trong file.

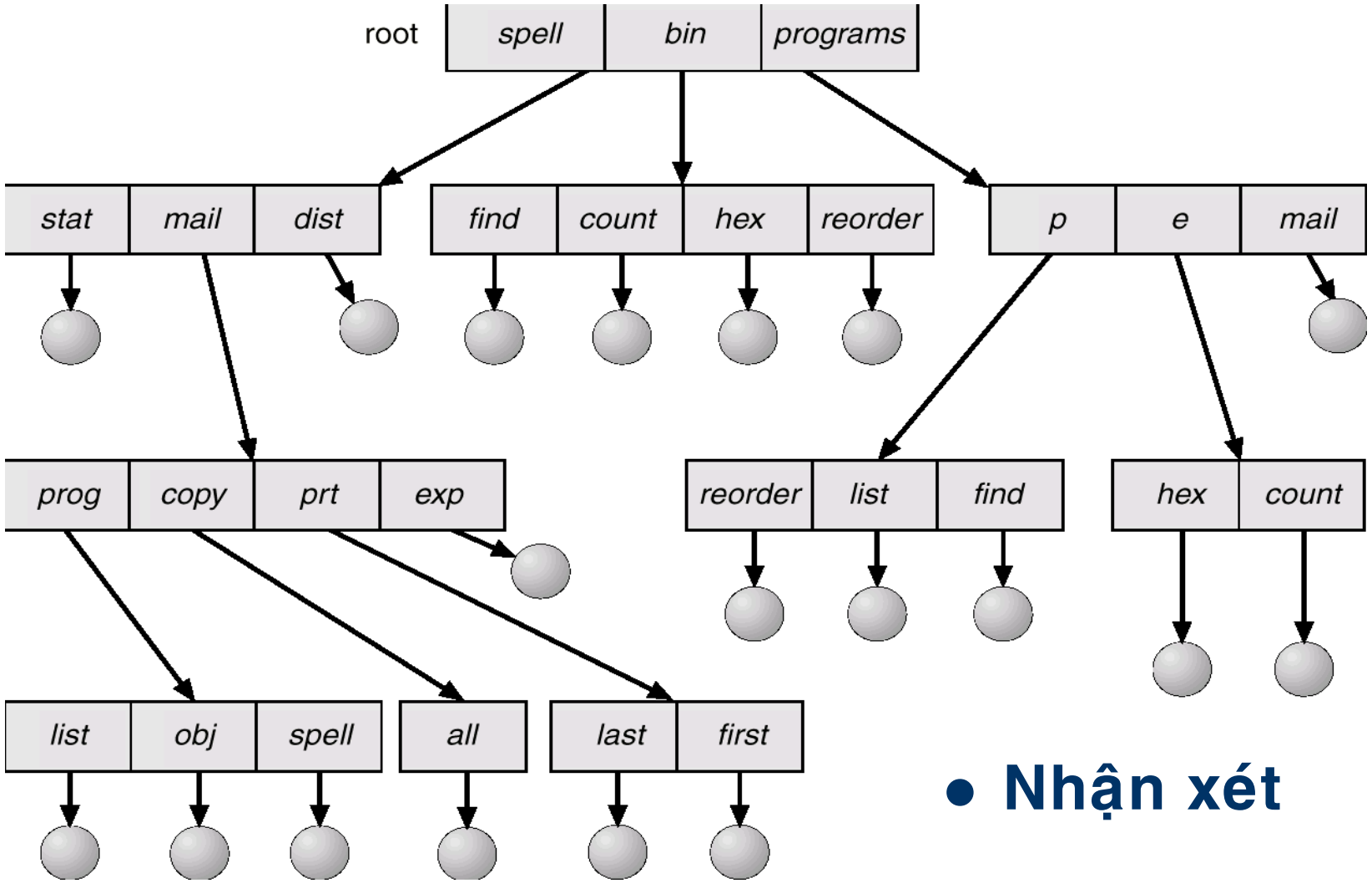
TỔ CHỨC THƯ MỤC

- **Thư mục (*directory*)**
  - Cấu trúc dữ liệu của HĐH để ánh xạ tên sang số nhận dạng file của HĐH
- **Tác vụ thực hiện trên thư mục**
  - Tìm file, tạo file, xoá file, liệt kê nội dung thư mục, đổi tên file, duyệt hệ thống file
- **Yêu cầu khi tổ chức hệ thống thư mục**
  - Hiệu quả
  - Tiện lợi cho người sử dụng
  - Có khả năng nhóm các file theo thuộc tính

CÁCH TỔ CHỨC THƯ MỤC

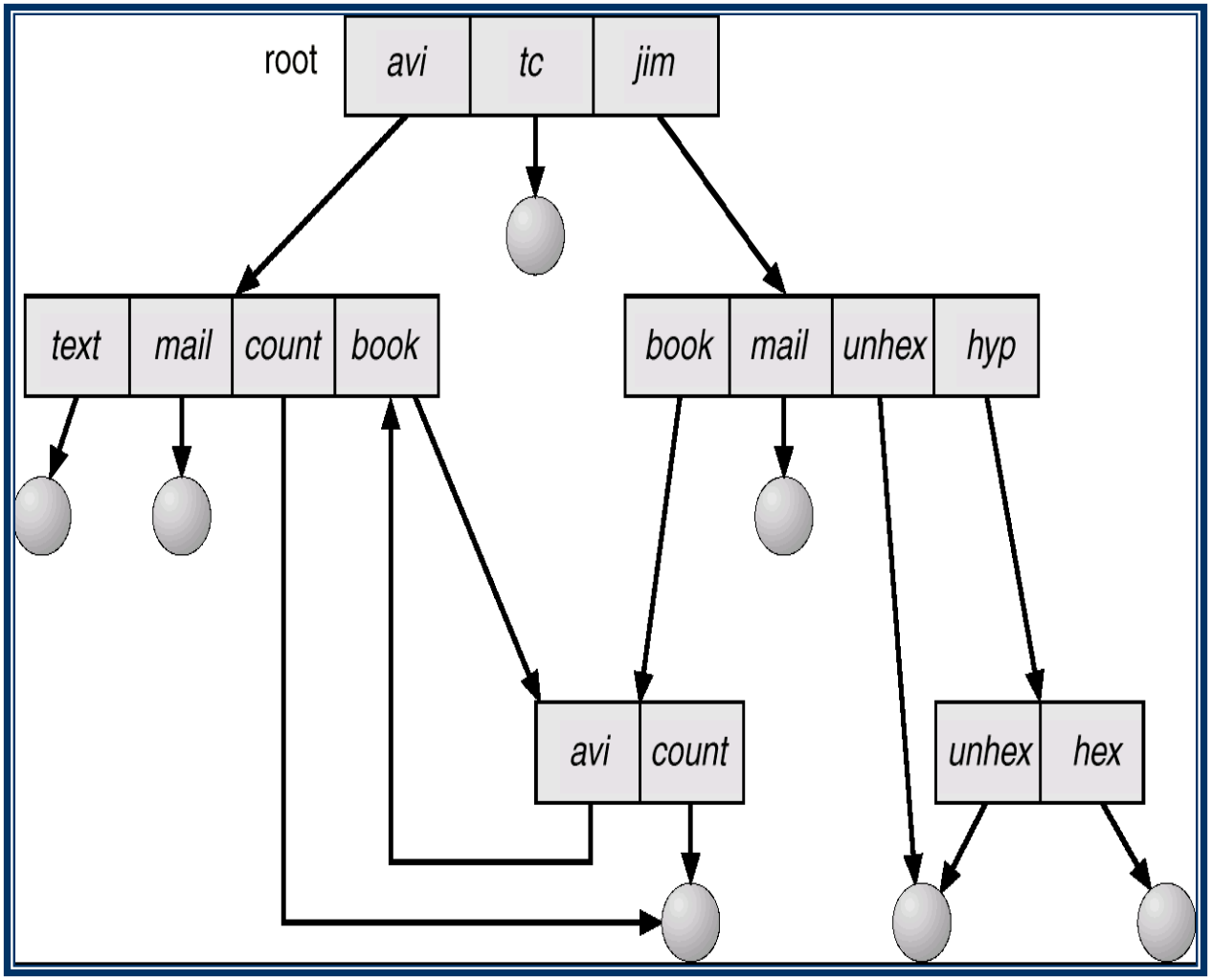
- **Tổ chức 1 cấp** *(Single-Level Directory)*
  - Sử dụng 1 không gian tên (thư mục) duy nhất cho mọi user
  - Việc đặt tên dễ đụng độ
  - Không có khả năng nhóm các file
- **Tổ chức 2 cấp** *(Two-Level Directory)*
  - 1 user có một thư mục riêng
  - Sử dụng đường dẫn để xác định nơi lưu file
  - Tìm kiếm nhanh
  - Vẫn có khả năng đụng độ khi đặt tên
  - Không có khả năng nhóm các file

TỔ CHỨC THƯ MỤC ĐA CẤP
(*Multilevel Directory*)



- **Nhận xét**

TỔ CHỨC THƯ MỤC DẠNG ĐỒ THỊ TỔNG QUÁT (*General Graph*)



- K/niệm link
  - Hard link
  - Soft link
- Vấn đề?
- Giải quyết?

MOUNT HỆ THỐNG FILE

- **Mount**
  - Gắn hệ thống file trên 1 thiết bị lưu trữ vào hệthống thư mục chính để truy cập
- **Mount point**
  - Thư mục nơi gắn hệ thống file ở ngoài vào
- **Unmount**
  - Tách hệ thống file của thiết bị lưu trữ ra khỏi mount point
- **Loại hệ thống file được mount:**
  - tùy thuộc sự hỗ trợ của hệ điều hành

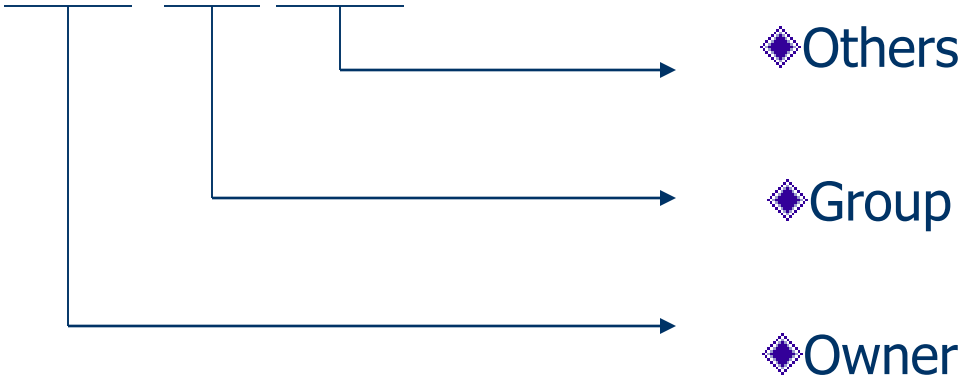
BẢO VỆ HỆ THỐNG FILE

- **Người tạo/ sở hữu file phải điều khiển được**
  - Các thao tác có thể thực hiện trên file
  - Ai có quyền thực hiện các thao tác trên
- **Các quyền thao tác trên file**
  - Read, Write, Execute, Append, Delete, List
- **Phương pháp bảo vệ**
  - Access list & group (Windows NT)
  - Access control bits (UNIX)
- **Điều khiển truy cập đồng thời**
  - Khóa toàn bộ file
  - Khóa từng phần file

BẢO VỆ FILE TRÊN UNIX

- Chế độ truy cập : read, write, execute
- 3 loại người dùng: owner, group, others
- Biểu diến quyền truy cập file bằng tổ hợp bit

rwx r-x r-x

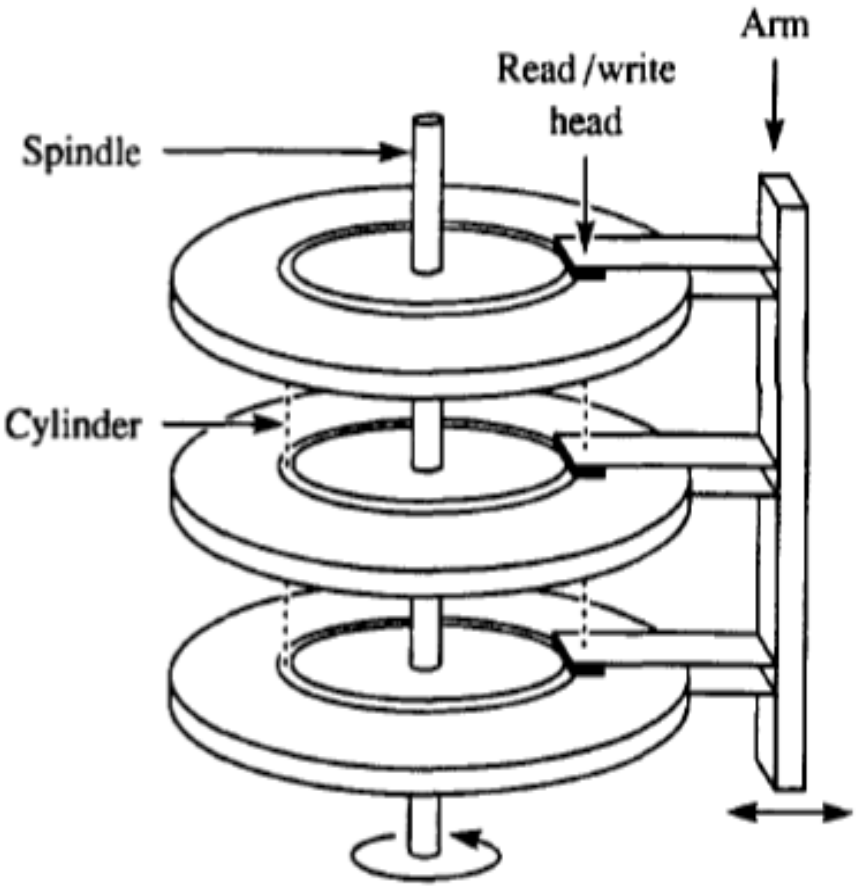


- Kiểm tra quyền sử dụng lần lượt theo owner, group rồi user

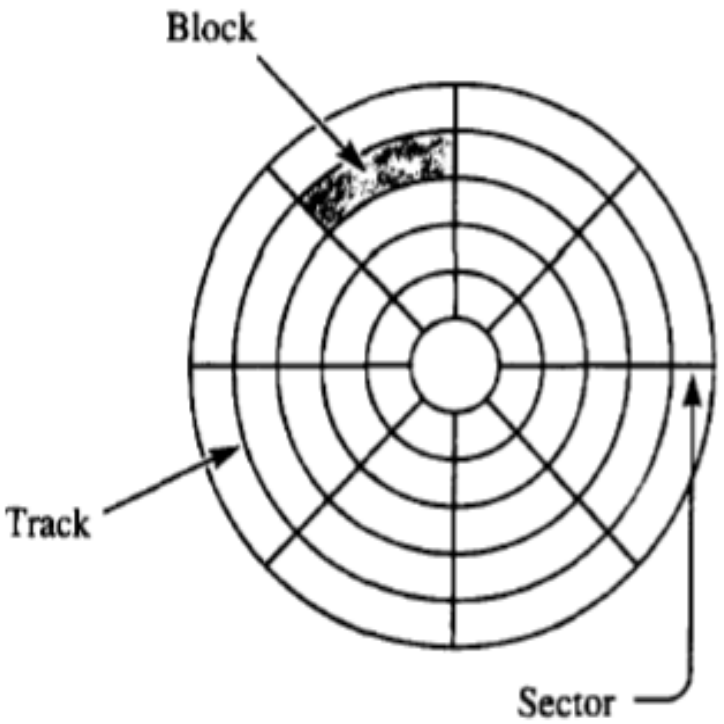
CHƯƠNG 8_2: HIỆN THỰC HỆ THỐNG FILE

- **Cấu trúc đĩa cứng**
- **Cấu trúc hệ thống file**
- **Hiện thực cấu trúc thư mục**
- **Cơ chế cấp phát vùng lưu trữ**
  - Cấp liên tục, theo liên kết, theo chỉ số
  - Hệ thống file của UNIX
- **Quản lý vùng trống**
- **Độ hiệu quả/ hiệu suất hệ thống file**
- **Sao lưu và phục hồi dữ liệu**
- **Bài tập**

CẤU TRÚC ĐĨA CỨNG



(a) A hard disk drive.

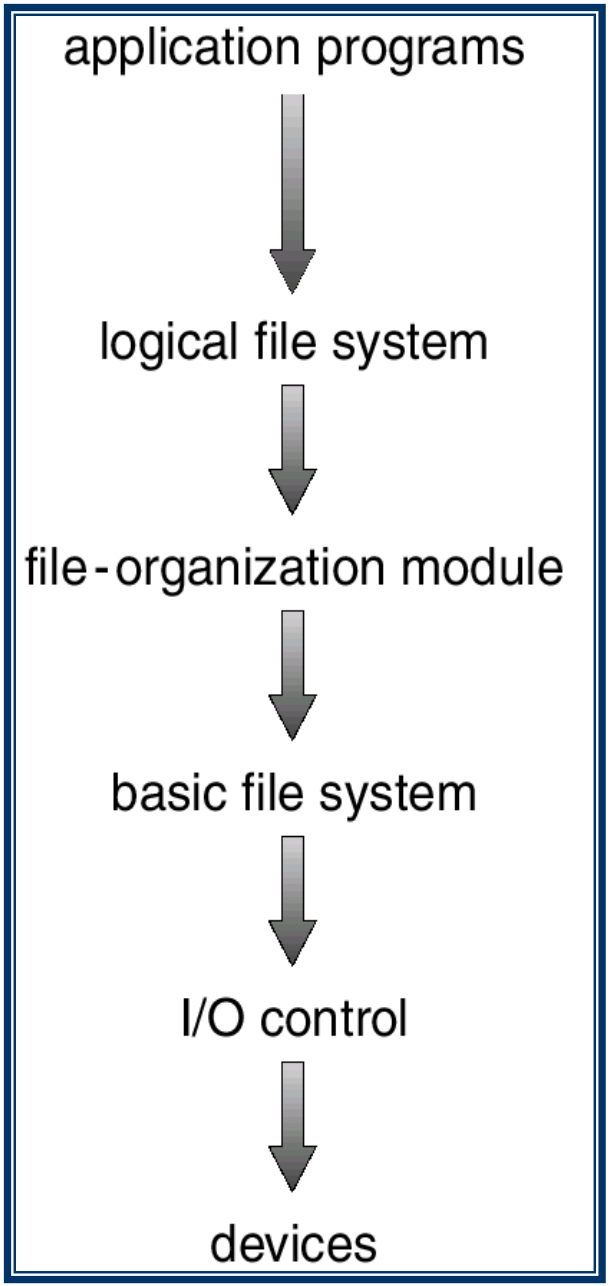


(b) A single disk.

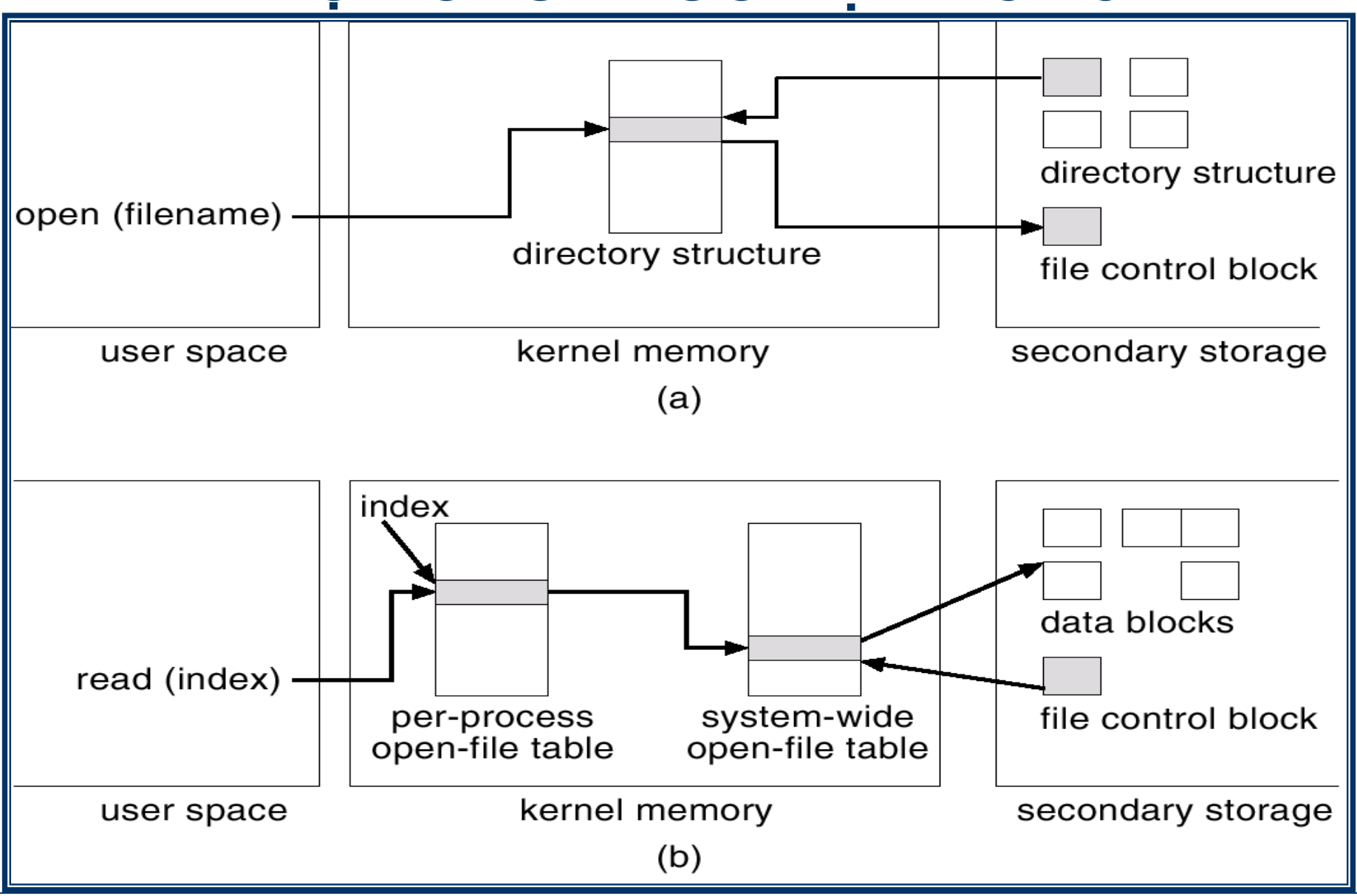
- **Hệ điều hành xem đĩa cứng như một chuỗi các block liên tiếp với kích thước cố định.**

CẤU TRÚC HỆ THỐNG FILE

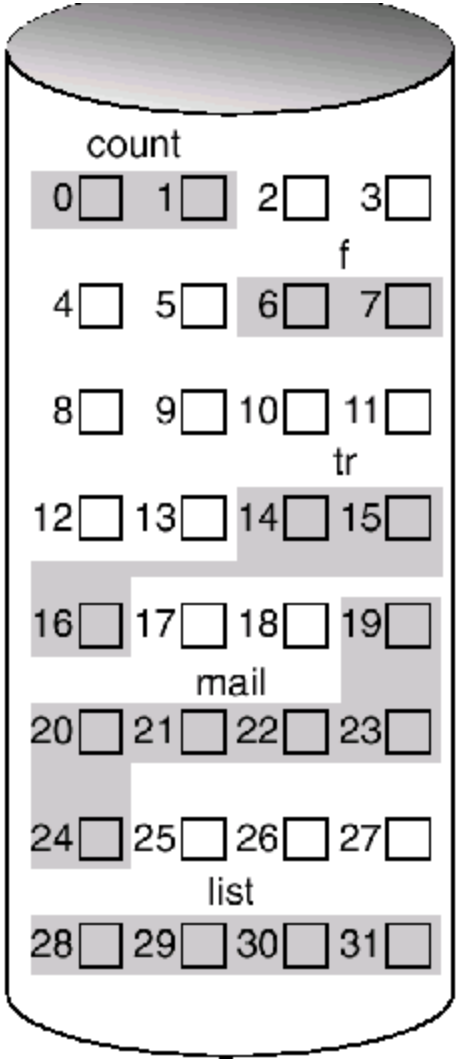
- **Tổ chức theo phân lớp**
- **File Control Block (FCB)**
  - Nằm trên đĩa cứng, chứa
    - Thông tin bảo mật file
    - Thông tin nơi lưu trữ file
- **Virtual File System (VFS)**
  - Cung cấp API chung đẻ truy xuất nhiều loại hệ thống file khác nhau
- **Cấu trúc thư mục**
  - Dùng danh sách liên kết
  - Dùng bảng băm



MINH HỌA CẤU TRÚC HỆ THỐNG FILE



CẤP PHÁT VÙNG LƯU TRỮ LIÊN TỤC (*Contiguous Allocation*)`

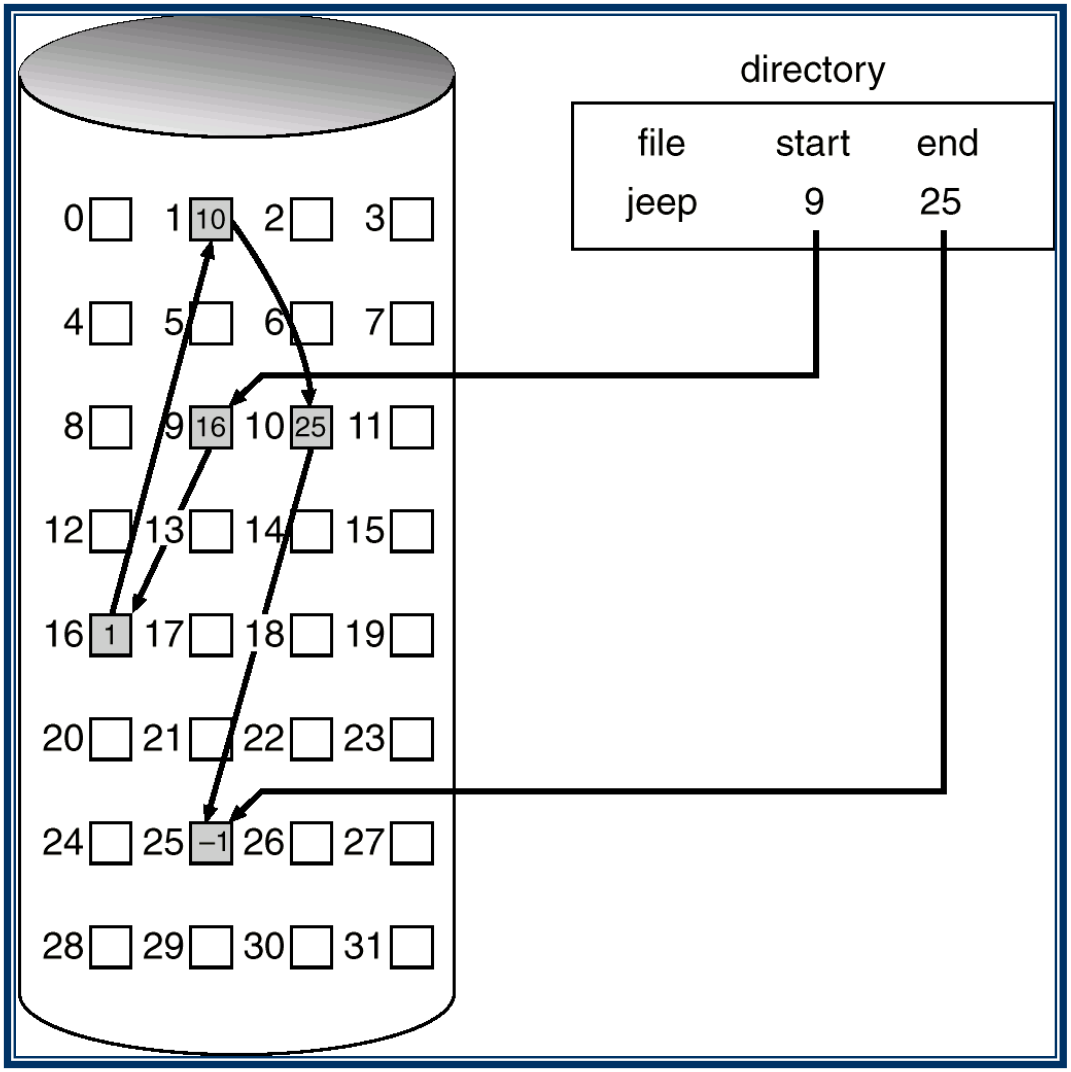


- File gồm n block liên tục
- Thông tin cấp phát:
  - Chỉ số block đầu, số block cấp
- Nhận xét ưu, nhược điểm.
- Khắc phục?

directory

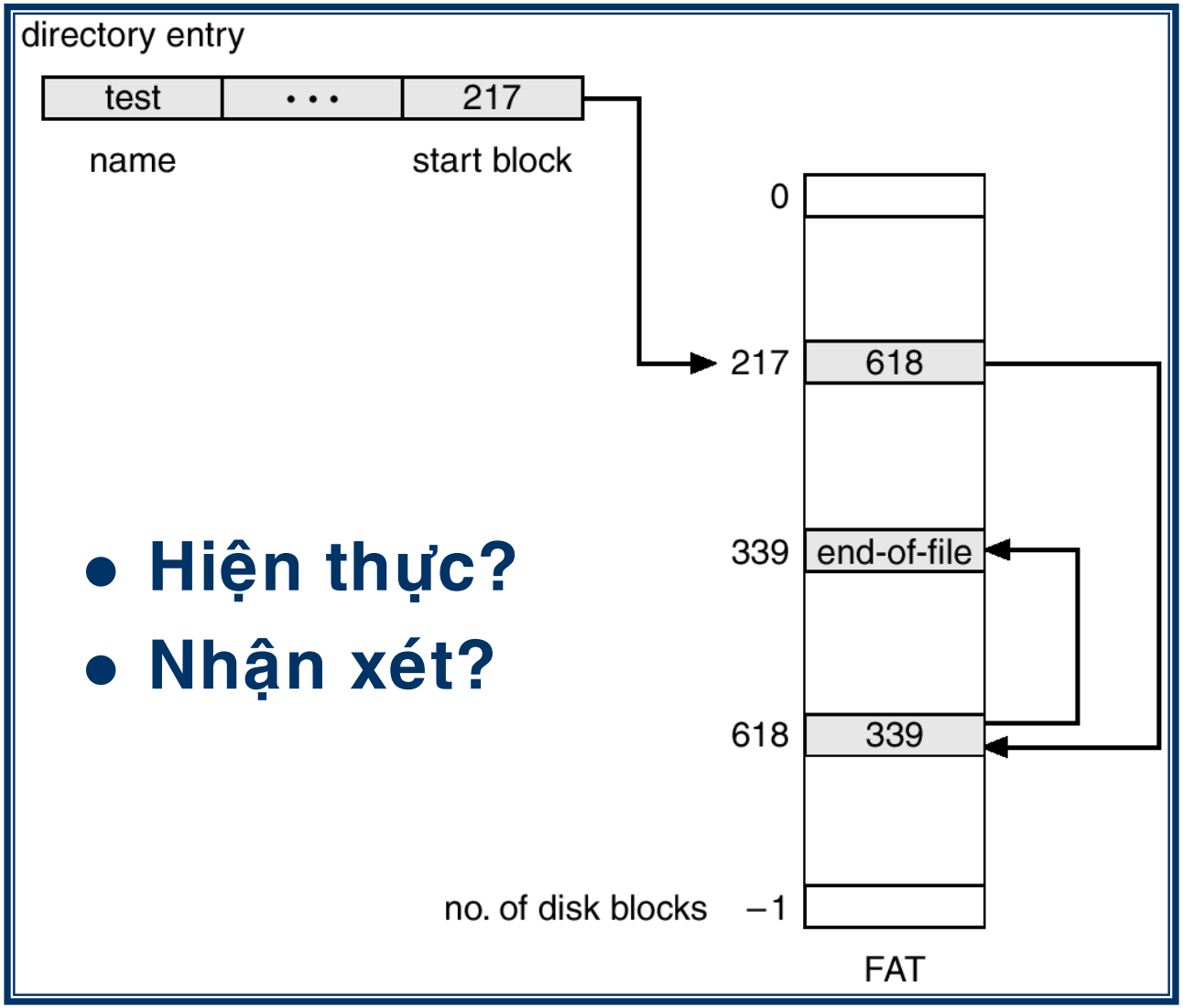
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

CẤP PHÁT VÙNG LƯU TRỮ THEO LIÊN KẾT (*Linked Allocation*)



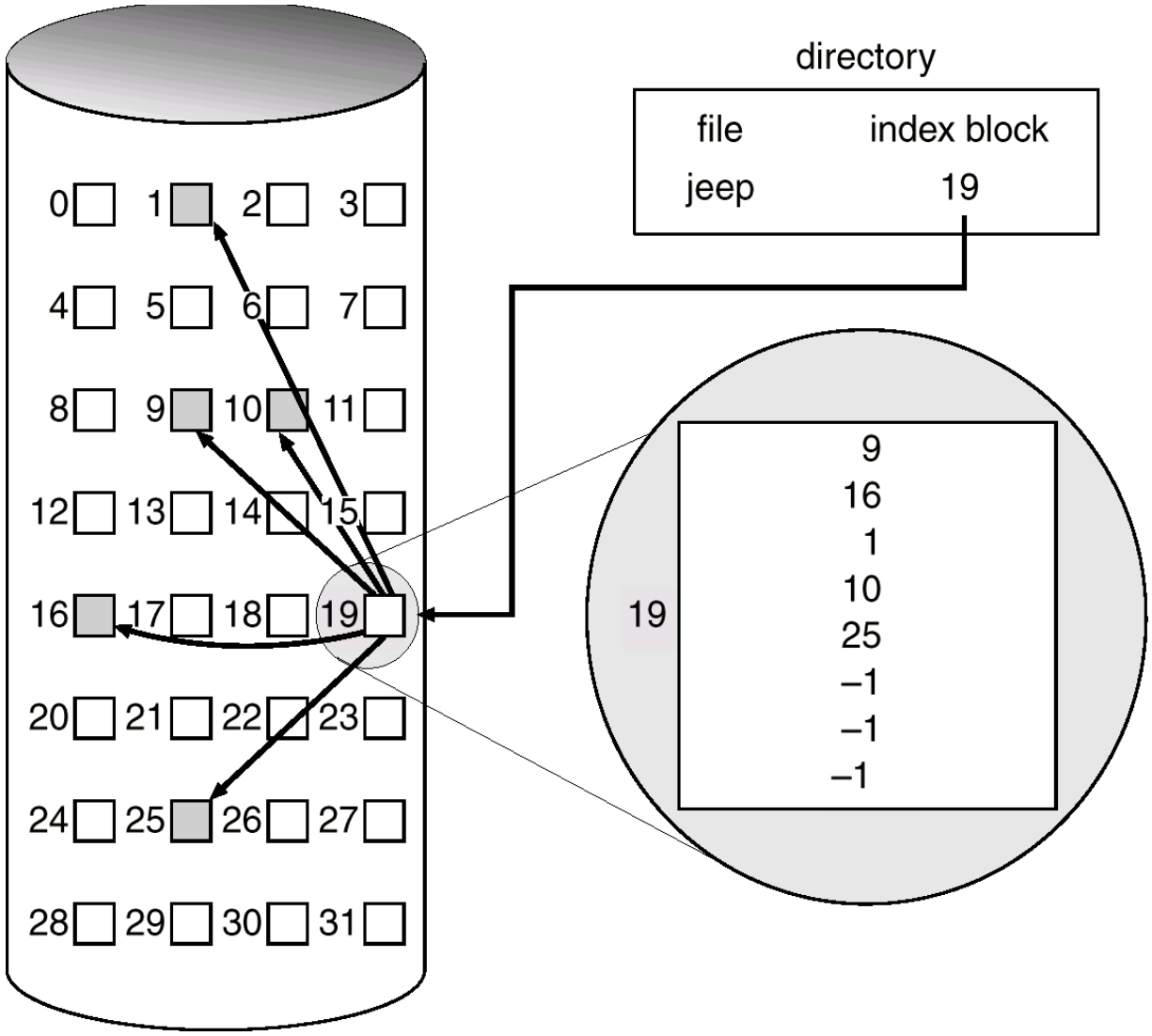
- File là danh sách liên kết của các block rải rác
- Hiện thực?
- Nhận xét?

FILE ALLOCATION TABLE (FAT)



- **Hiện thực?**
- **Nhận xét?**

CẤP PHÁT VÙNG LƯU TRỮ THEO CHỈ SỐ (*Indexed Allocation*)



- Dùng bảng các chỉ số để lưu các con trỏ đến các block dữ liệu của file
- Nhận xét?

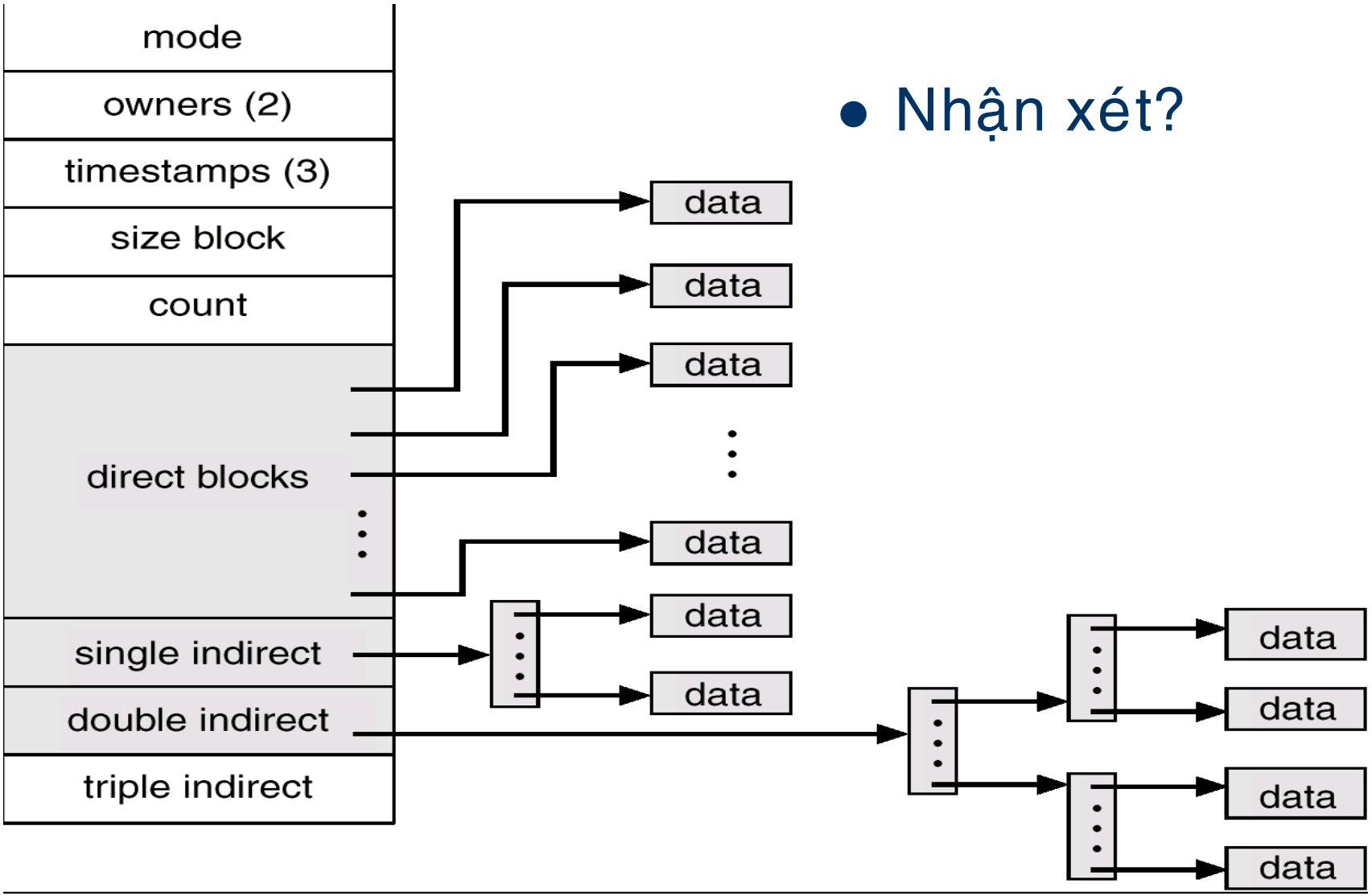
LƯU TRỮ BẢNG CHỈ SỐ CỦA FILE

- Lưu liên tục
  - Bảng chỉ số lưu trong 1 block của đĩa
- Lưu theo kiểu liên kết
  - Bảng chỉ số lưu trong n block của đĩa nối với nhau bằng danh sách liên kết
- Lưu bằng bảng chỉ số đa cấp
  - Dùng bảng chỉ số khác để lưu các con trỏ đến các index block của file
- Sử dụng cơ chế kết hợp

HỆ THỐNG FILE CỦA UNIX

- Đĩa cứng chia thành nhiều block
  - Boot block
  - Super block
  - Các block chứa danh sách các i-node
  - Các block dữ liệu
- Thông tin lưu trong 1 i-node
  - Mode truy cập
  - Owner UID
  - Số link trỏ tới file
  - Thông tin về thời điểm truy cập , tạo file…
  - Kích thước file
  - Dãy các địa chỉ khối chứa dữ liệu
  - …

CẤU TRÚC I-NODE CỦA BSD UNIX



- Nhận xét?

I-NODE CỦA THƯ MỤC

- Thư mục /

Chỉ số i-node	Tên file / thư mục con
2	.
2	..
5	etc
10	home
12	usr

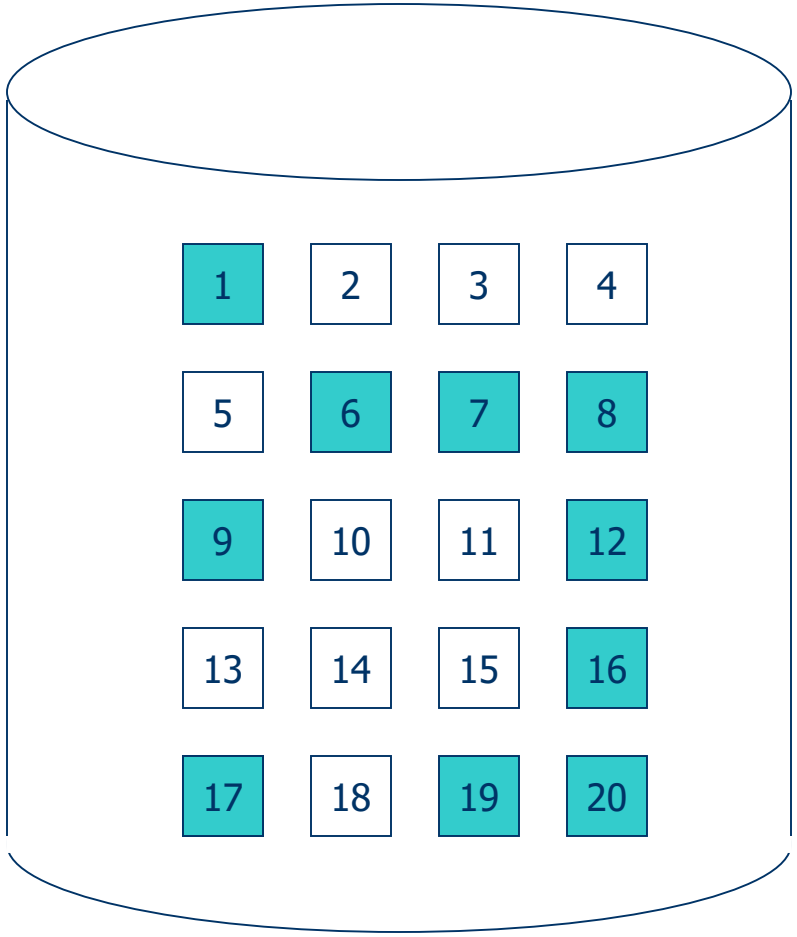
- Thư mục /home

Chỉ số i-node	Tên file / thư mục con
10	.
2	..
15	hung
20	Os01
21	Os02

QUẢN LÝ VÙNG TRỐNG (1/2)

- **Dùng bit vector: N bit quản lý N block data**
  - Bít =0 : block đã cấp
  - Bit=1: block còn trống
- **Dùng danh sách liên kết các block trống**
- **Nhóm các block trống (Grouping)**
  - Chứa địa chỉ N block trong 1 block trống đầu tiên
  - N-1 địa chỉ đầu trỏ đến các block trống thực sự
  - Địa chỉ cuối trỏ đến block chứa N địa chỉ block trống khác
- **Đếm khoảng trống (Counting)**
  - Mỗi block trống lưu trữ số khoảng trống liên tục tiếp theo nó & địa chỉ block trống không kê tiếp.

QUẢN LÝ VÙNG TRỐNG (2/2)



18 Vùng trống

17 Vùng đã cấp phát

- **P/p grouping**
  - Block 2: 3,4, 5,10
  - Block 10: 11, 13,14,15
  - Block 15: 18,-,-,-
- **P/p counting**
  - Block 2: 3, 10
  - Block 5: 0,10
  - Block 10: 1, 13
  - Block 13: 2, 18`

ĐỘ HIỆU QUẢ/ HIỆU SUẤT CỦA HỆ THỐNG FILE

- **Độ hiệu quả hệ thống file phụ thuộc**
  - Cách cấp phát đĩa, các giải thuật trên thư mục
  - Loại dữ liệu trong mục của bảng thư mục
- **Tăng hiệu suất hệ thống file**
  - Disk cache
  - Page cache
  - Free-behind & read-ahead
  - Virtual Disk/ RAM disk
  - Parallel I/O

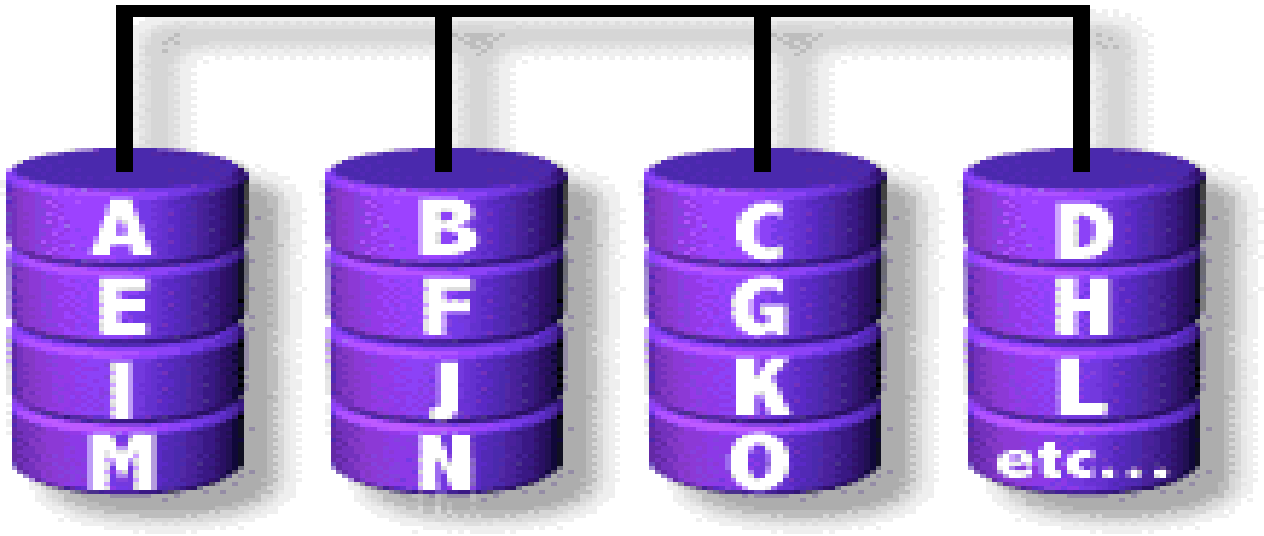
SAO LƯU VÀ PHỤC HỒI DỮ LIỆU

- **Kiểm tra sự nhất quán của dữ liệu**
  - So sánh thông tin trên block đĩa và trong thư mục
  - Sử dụng các tiện ích: ndd, fsck, scandisk,…
- **Sao lưu** (*backup*) **dữ liệu sang thiết bị lưu trữ khác**
  - Sao lưu toàn phần *(normal backup)*
  - Sao lưu tăng dần *(incremental backup)*
- **Phục hồi** (*restore*) **dữ liệu từ thiết bị sao lưu**
  - Khi có hỏng hóc hệ thống
  - Khi cần phục hồi hệ thống về trạng thái cũ
- **Hệ thống file có ghi log** *(Log Structured File System)*

RAID (*Redundant Array of Inexpensive Disks*)

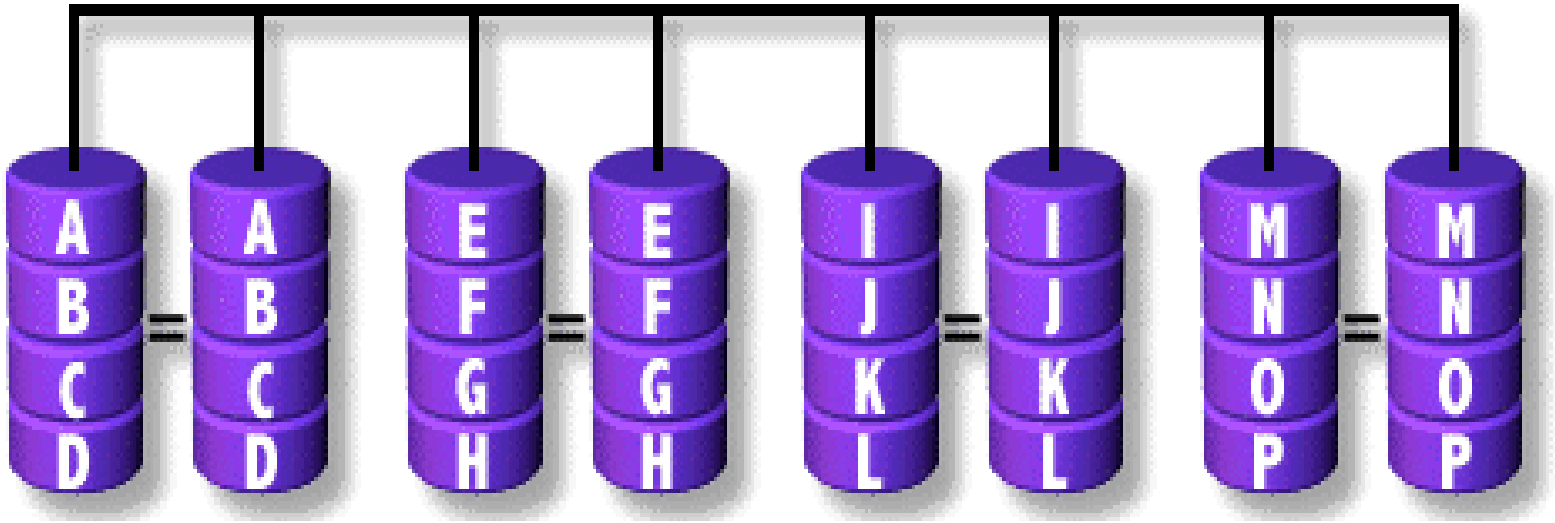
- Tập hợp các đĩa cứng được hệ điều hành xem như một thiết bị lưu trũ luận ly
- Dữ liệu được phân bố trên tất cả các đĩa
- Các mục tiêu chính
  - Tăng dung lượng lưu trữ
  - Tăng hiệu suất I/O
  - Tăng tính sẵn sàng cao
  - Tăng khả năng phục hồi hệ thống
- Các loại RAID
  - RAID 0 → RAID 10 (phổ biến RAID 0, 1, 3, 5)
  - Software RAID/ Hardware RAID

RAID-0



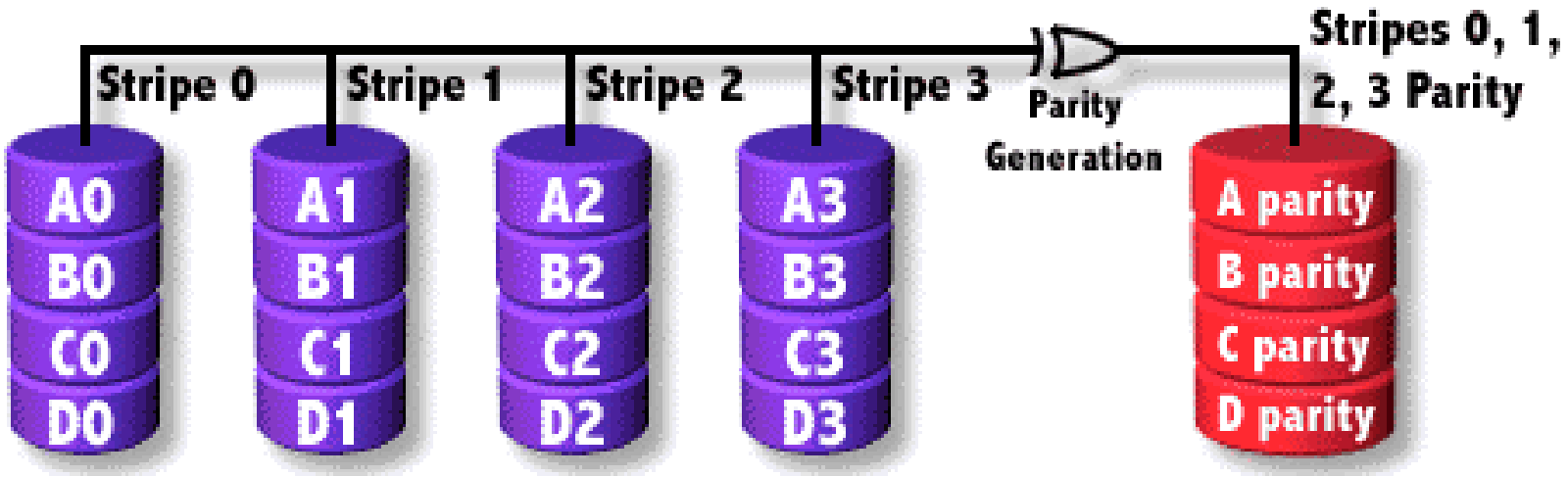
- Dữ liệu lưu trữ trải đều trên các đĩa
- Tăng không gian lưu trữ
- Tăng hiệu suất hệ thống
- Tính sẵn sàng của dữ liệu thấp

RAID-1



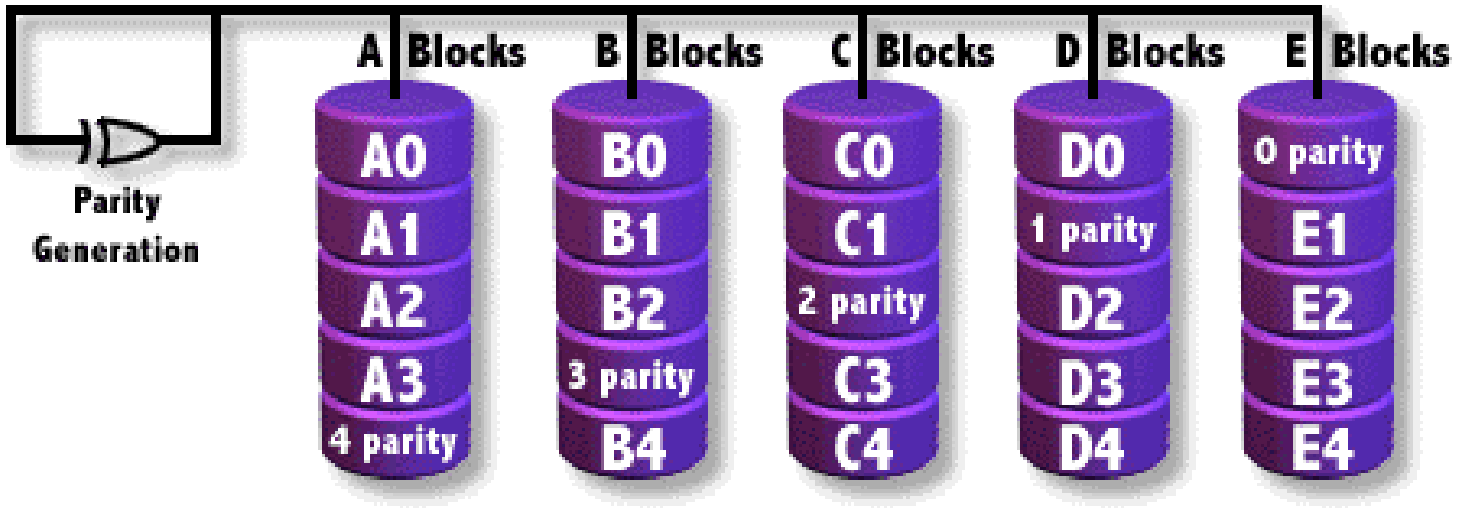
- Nhân bản dữ liệu trên các đĩa tách biệt
- Tính sẵn sàng & tốc độ đọc dữ liệu rất cao
- Yêu cầu dung lượng đĩa gấp đôi
- Tốc độ ghi chậm hơn

RAID-3



- Lưu dữ liệu trải đều trên các đĩa
- Sử dụng một đĩa lưu thông tin kiểm tra dữ liệu
- Tính sẵn sàng cao, chi phí hợp lý
- Hiệu suất I/O thấp

RAID-5



- Dữ liệu, thông tin kiểm tra được lưu trải đều trên các đĩa
- Tính sẵn sàng dữ liệu trung bình, chi phí hợp lý
- Tốc độ ghi thấp
- Yêu cầu phần cứng đặc biệt

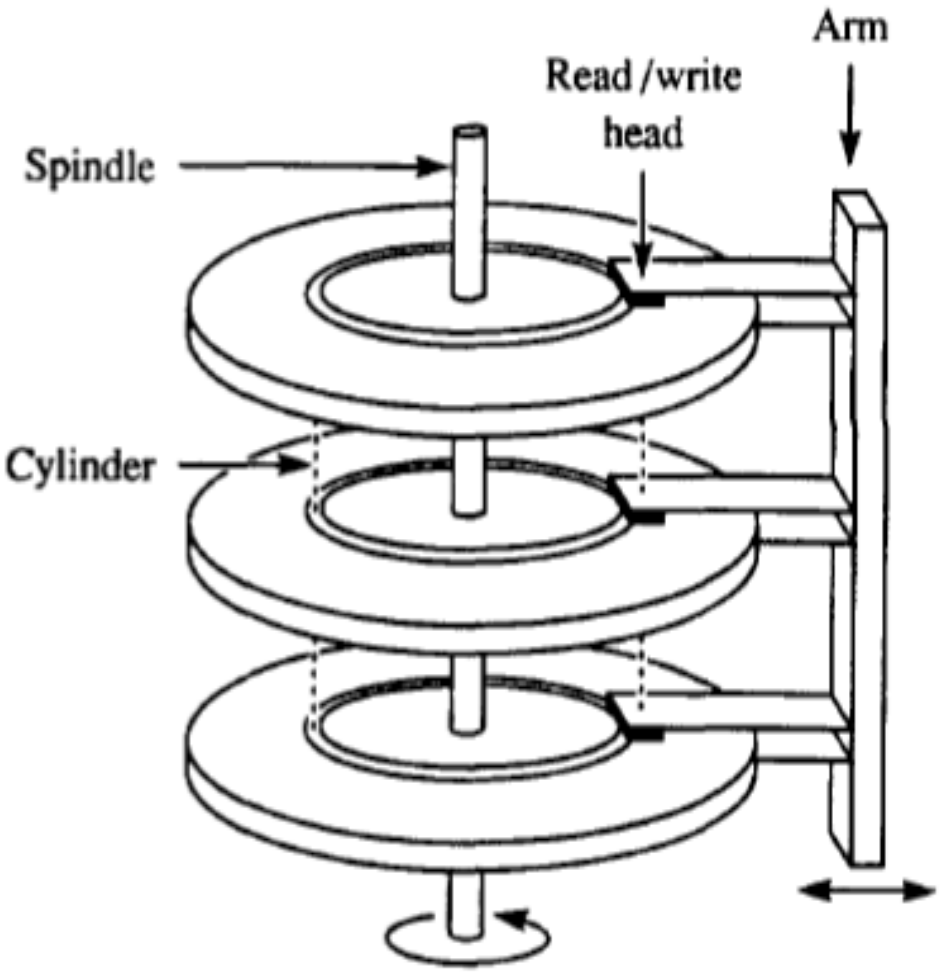
BÀI TẬP

1. So sánh thời gian các lệnh copy, move, delete trong tất cả các trường hợp có thể có.
2. Tại sao trong UNIX không có system call detete(…) để xoá file mà chỉ có system call unlink(…) để xoá một link đến file?
3. Đĩa có N block, dùng p/p grouping (4 block) để quản lý vùng trống. Tính thời gian trung bình để tìm được n khối trống và cấp phát cho file.

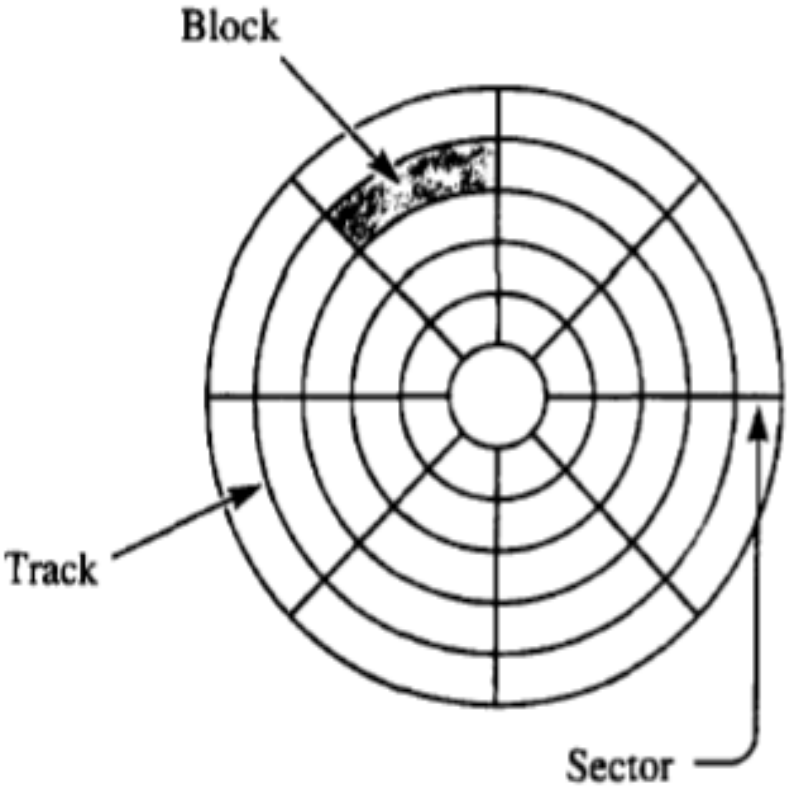
CHƯƠNG 8_3: QUẢN LÝ ĐĨA CỨNG

- Cấu trúc đĩa cứng
- Nội dung đĩa cứng
- Truy xuất đĩa & định thời truy xuất đĩa
- Quản lý đĩa
- Hiện thực hệ thống lưu trữ ổn định
- Các kỹ thuật tăng hiệu suất đĩa cứng

CẤU TRÚC ĐĨA CỨNG

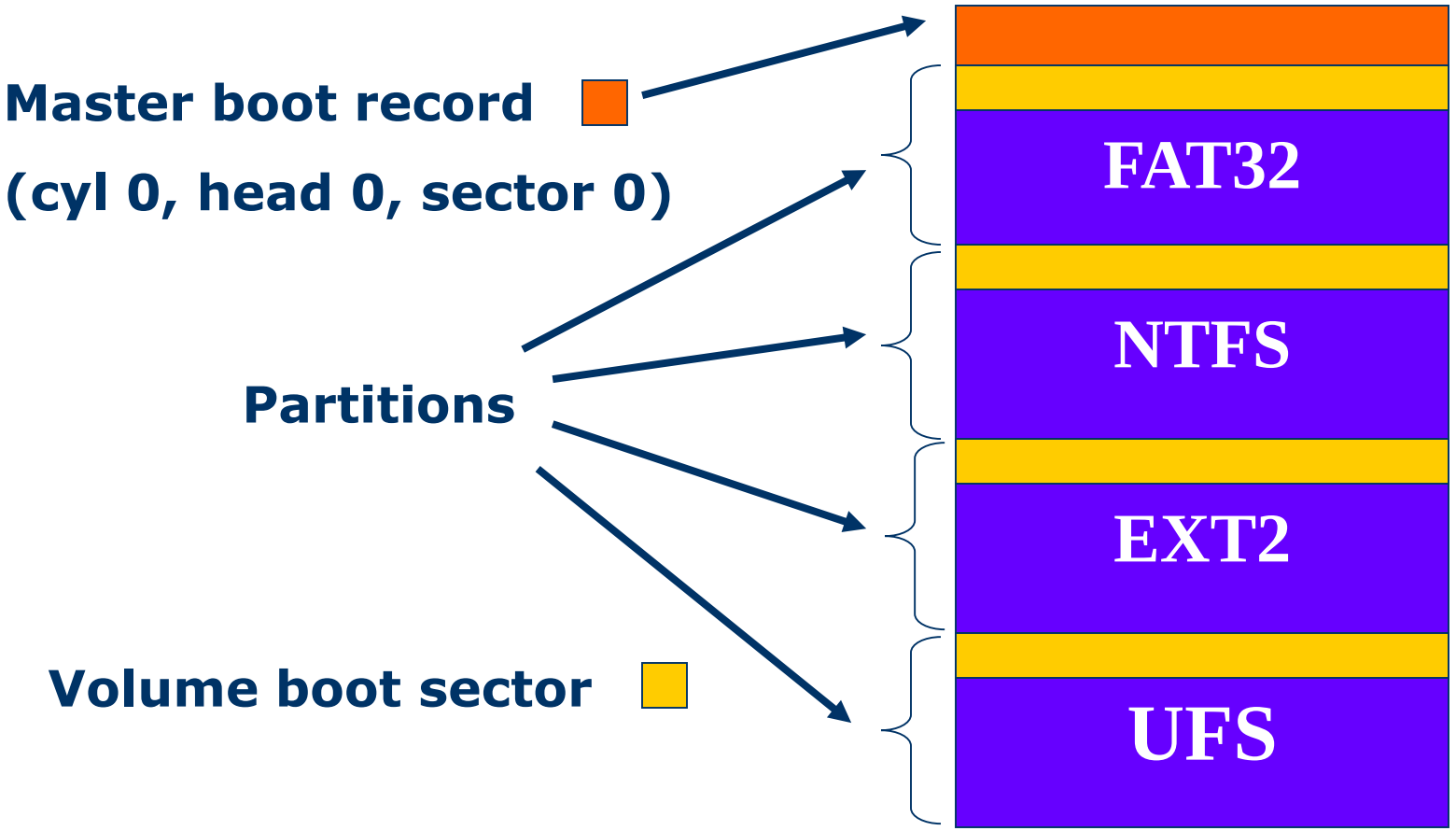


(a) A hard disk drive.



(b) A single disk.

NỘI DUNG LUẬN LÝ ĐĨA CỨNG



NỘI DUNG ĐĨA CỨNG

- Master Boot Record
  - Master Partition Table:
    - Chứa thông tin về từng partition: partition ID, Activity flags, start CHS, end CHS…
    - Link tới Extended Partition Table (chứa thông tin về ổ đĩa luận lý thứ 1 trên đĩa)
  - Master Boot Code:
    - Chứa mã nạp OS ở các partition active
- Partition
  - Vùng không gian liên tục trên đĩa
  - Chứa 1 hệ thống file hoặc n ổ đĩa luận lý (logical volume)
  - Mỗi ổ đĩa luận lý có 1 Volume Boot Sector (VBS)
    - Disk Parameter Block: thông tin về đĩa luận lý
    - Volume Boot Code: mã để khởi động OS trên ổ luận lý này

TRÌNH TỰ KHỞI ĐỘNG HỆ THỐNG

- Power-On Self Test (POST)
  - Kiểm tra phân cứng
  - Chạy các hàm BIOS mở rộng trong các ROM ở các mạch ngoại vi
- BIOS gọi interrupt 13h, nạp MBR và khởi động Master Boot Code (MBC)
- MBC nạp VBS của partition chính tích cực đầu tiên trên đĩa khởi động
- Volume Boot Code khởi động OS
- Các BIOS & OS mới có thể cho boot từ CDROM, đĩa mềm, đĩa ZIP hoặc qua mạng (Remote Boot)

TRUY XUẤT ĐĨA CỨNG

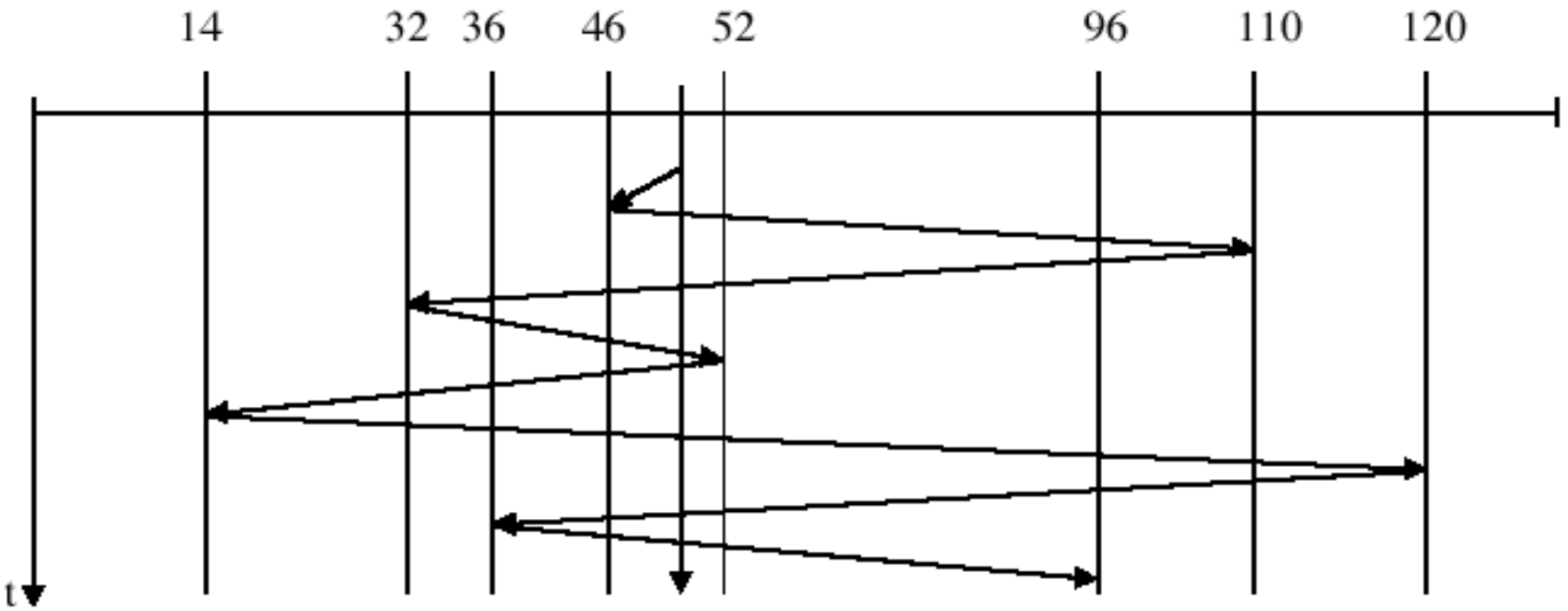
- 3 yếu tố ảnh hưởng thời gian truy xuất đĩa
  - **Seek time:** thời gian di chuyển đầu đọc tới track
  - **Latency:** thời gian để quay đĩa sao cho sector cần đọc nằm dưới đầu đọc
  - **Transfer time**: thời gian đọc/ ghi dữ liệu lên sector
- Thực tế:
  - Seek time >> latency time > transfer time
- Tối ưu seek time ➔ định thời truy xuất đĩa
- Tối ưu latency time:
  - Làm đĩa nhỏ, quay nhanh hơn, lưu trữ dữ liệu liên quan gần nhau
  - Chọn kích thước sector, nơi lưu trữ các file thường dùng hợp lý

CÁC GIẢI THUẬT ĐỊNH THỜI ĐĨA

- Bài toán: Có n yêu cầu đọc đĩa ở các track khác nhau x1, x2, … , xN vào các thời điểm tương ứng t1, t2, …, tN
  ➔ phục vụ các yêu cầu đó vào thời điểm nào?
- Tiêu chuẩn đánh giá
  - Công bằng
  - Hiệu suất cao
  - Thời gian đáp ứng trung bình thấp
  - Dự đoán được thời gian phục vụ
- Một số giải thuật tiêu biểu:
  - FCFS
  - SSTF
  - SCAN, N-step-SCAN, C-SCAN
  - CLOOK

ĐỊNH THỜI TRUY XUẤT ĐĨA –FCFS

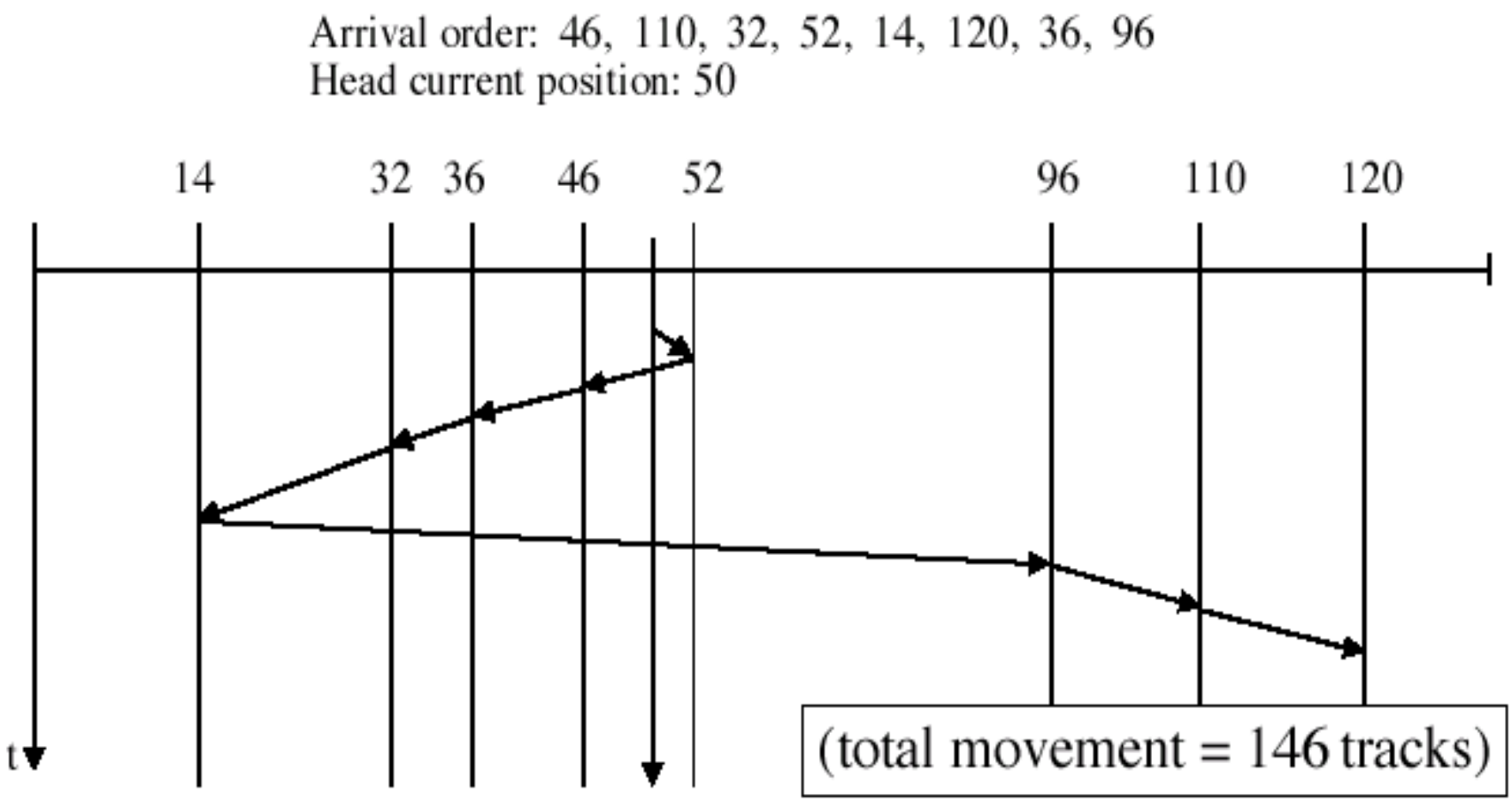
Arrival order: 46, 110, 32, 52, 14, 120, 36, 96 (track addresses)
Head current position: 50



Total head movement = 454 tracks ➔ **Nhận xét ?**

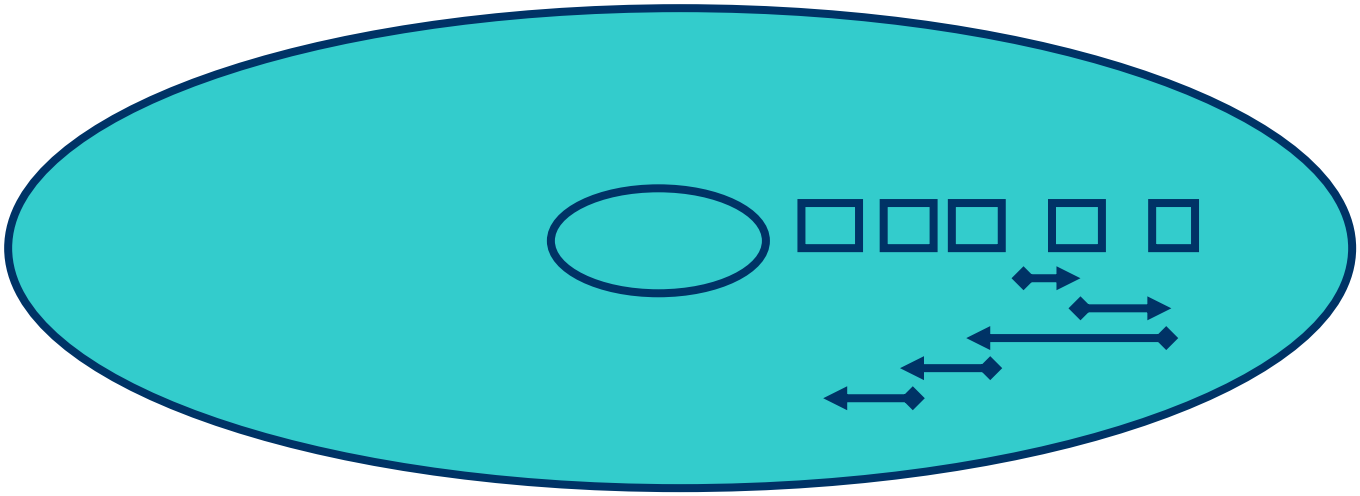
GIẢI THUẬT SSTF
(Shortes Seek Time First)

- **Phục vụ yêu cầu đọc gần vị trí đầu đọc hiện tại nhất.**



GIẢI THUẬT SCAN

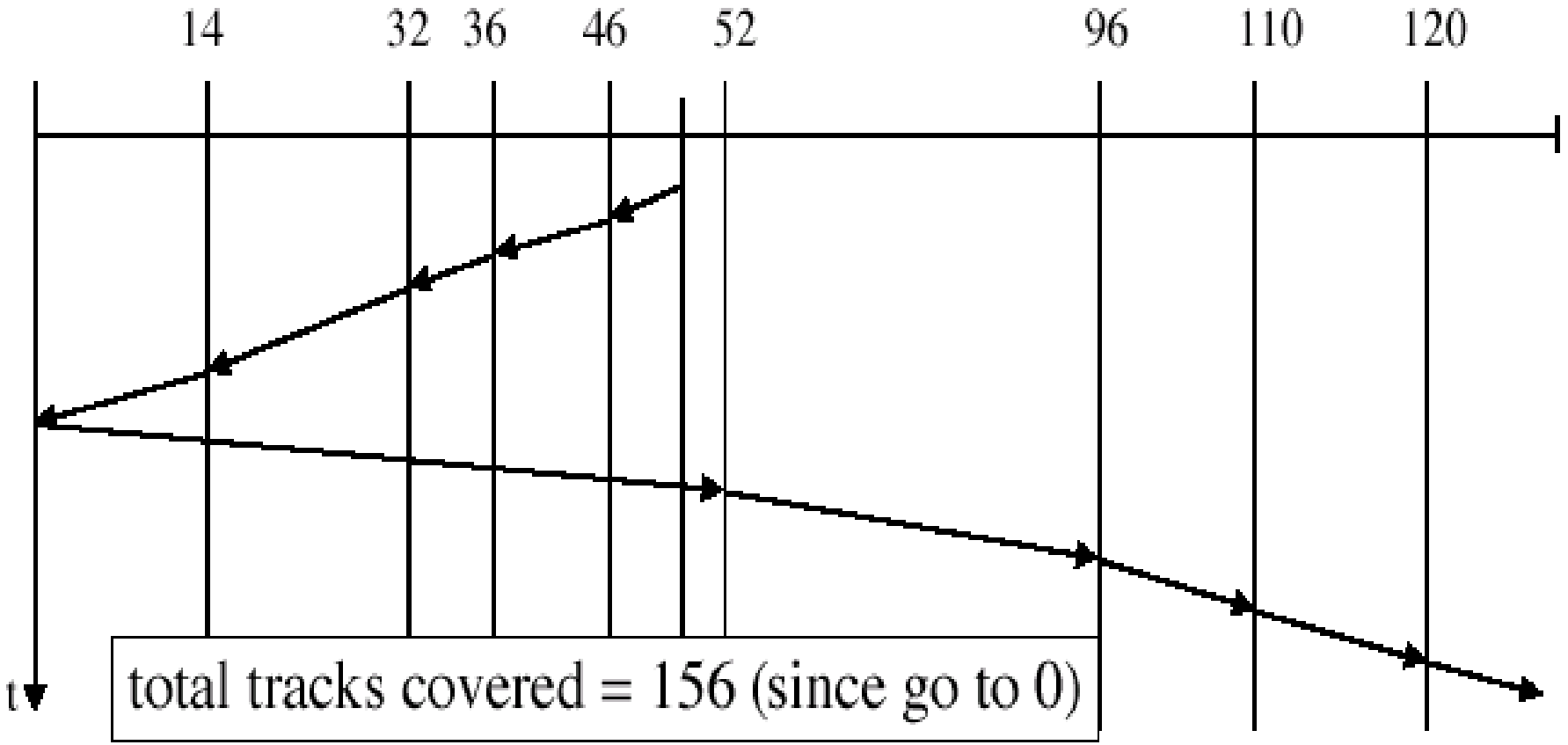
- Phục vụ theo hướng phục vụ từ trong ra ngoài
- Khi đầu đọc ra tới track ngoài cùng, phục vụ theo hướng ngược lại từ ngoài vào trong



- Nhận xét?

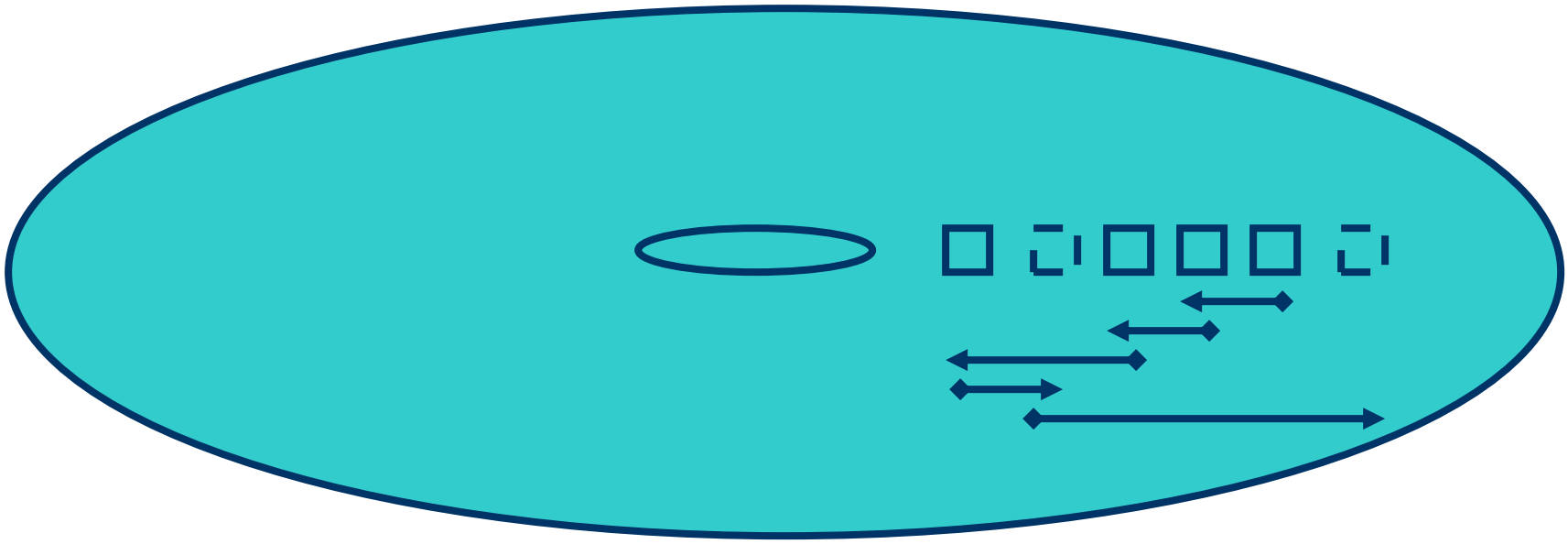
VÍ DỤ VỀ GIẢI THUẬT SCAN

Arrival order: 46, 110, 32, 52, 14, 120, 36, 96
Head current position: 50, moving toward to 0.



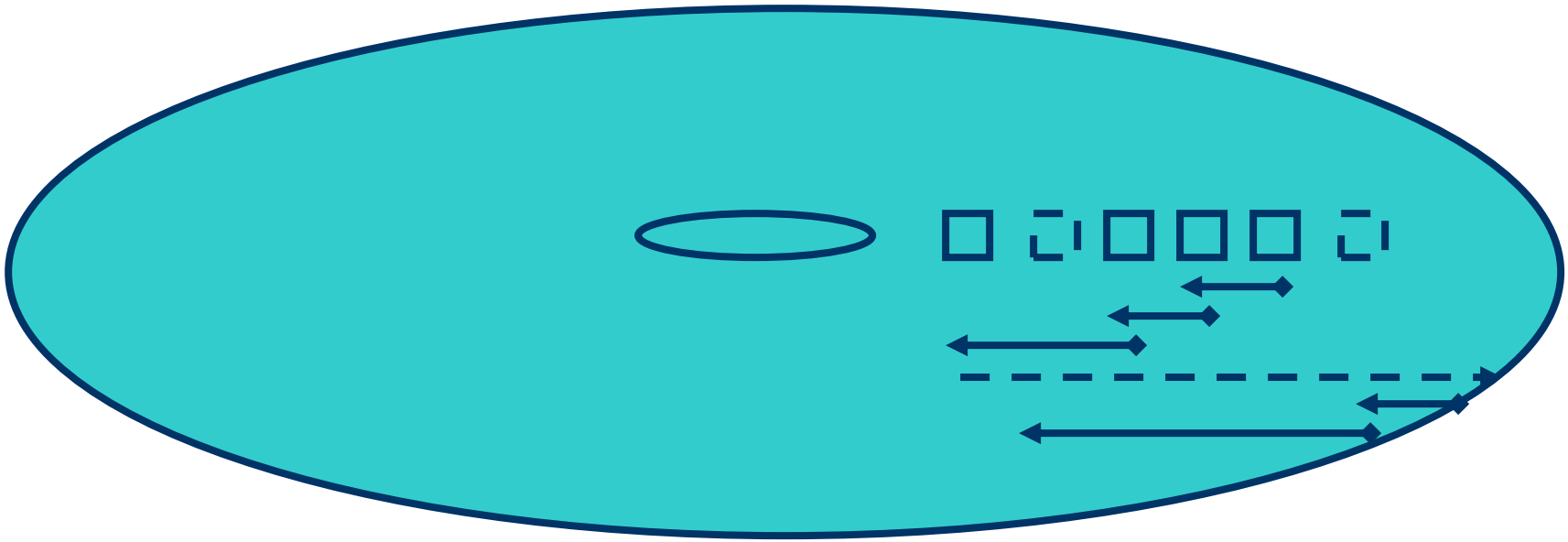
GIẢI THUẬT N-step-SCAN

- Nhóm các yêu cầu truy xuất lại
- Phục vụ nguyên 1 nhóm yêu cầu theo 1 chiều di chuyển của đầu đọc



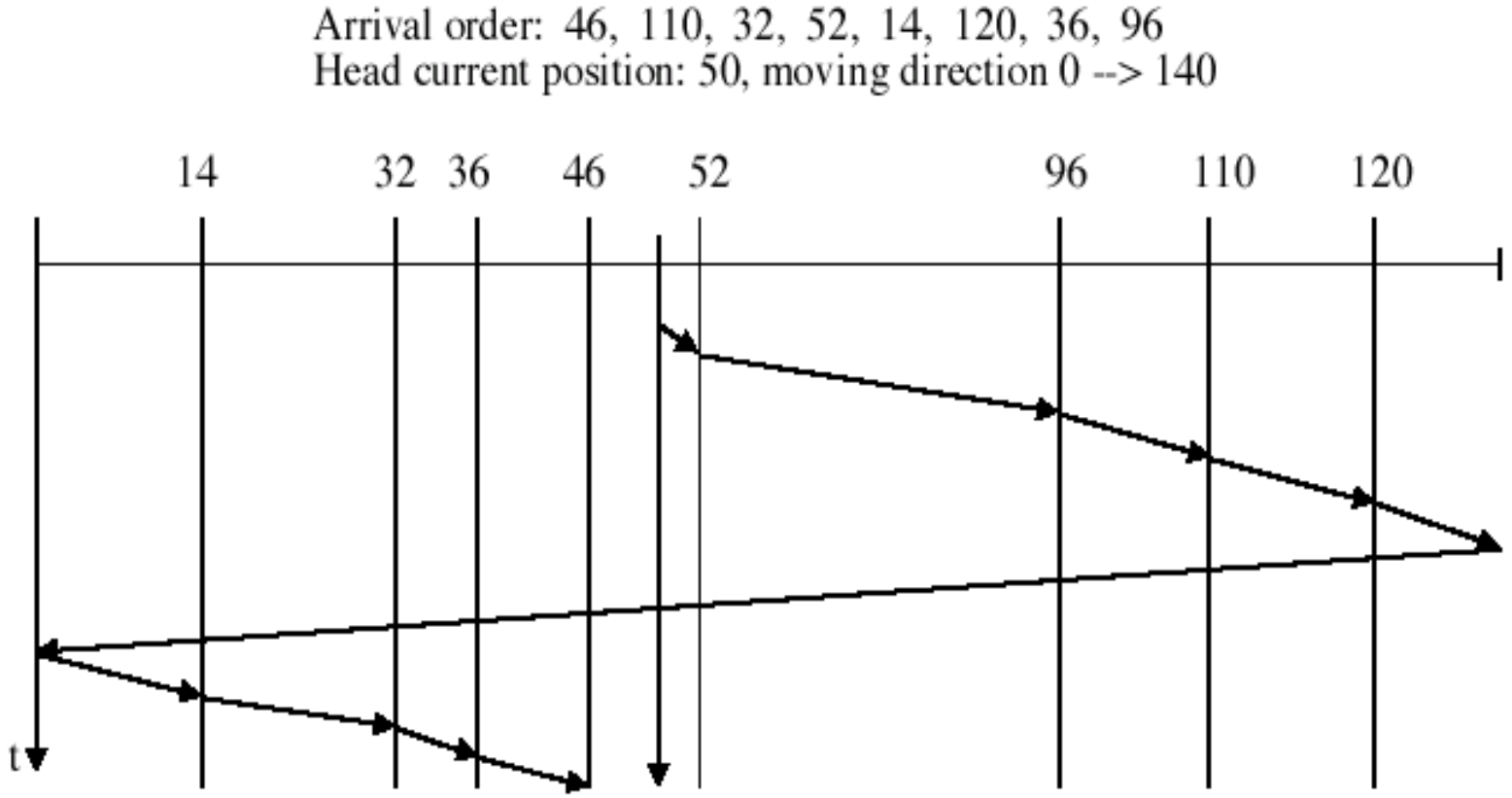
GIẢI THUẬT C-SCAN

- Như giải thuật N-step-SCAN nhưng theo chỉ phục vụ theo 1 hướng duy nhất
- Nhận xét?



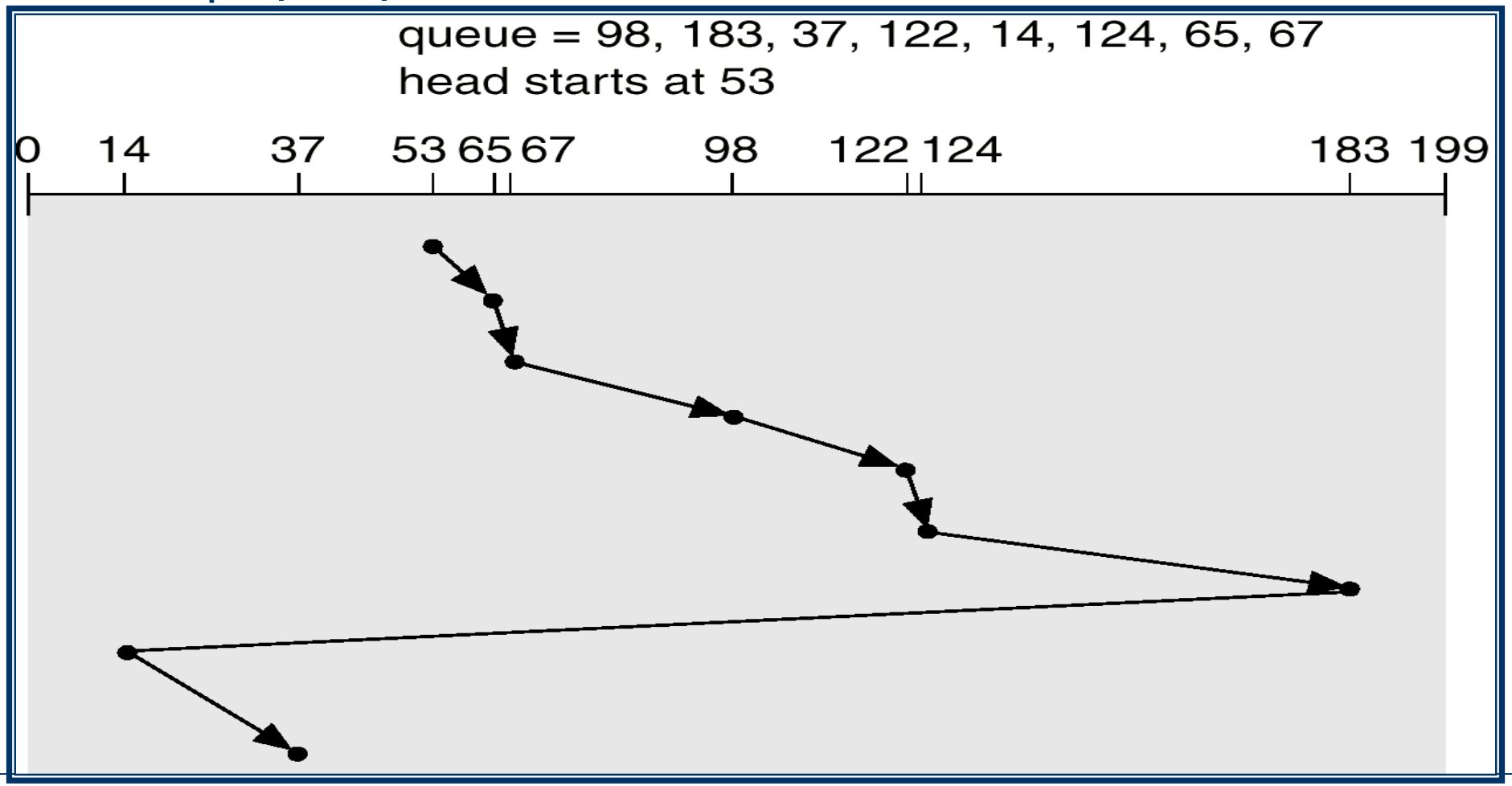
VÍ DỤ VỀ GIẢI THUẬT C-SCAN

- Như giải thuật N-step-SCAN nhưng chỉ phục vụ theo 1 hướng duy nhất
- Nhận xét?



GIẢI THUẬT C-LOOK

- Như C-SCAN, nhưng chỉ di chuyển đầu đọc tới track ngoài cùng được phục vụ rồi quay lại track trong cùng cần phục vụ



QUẢN LÝ ĐĨA

- Low-level formatting: chia đĩa ra các sector để disk controller có thể đọc, ghi được
- Lưu cấu trúc dữ liệu của OS lên đĩa
  - Partitioning: phân vùng đĩa
  - High-level formatting: tạo hệ thống file trên partition
- Tạo boot block
- Xử lý lỗi: kỹ thuật sector sparing
- Quản lý vùng swap
  - Tạo vùng swap khi nào?
  - Sử dụng dùng swap-map
- Lắp đặt đĩa
  - qua cổng I/O
  - qua mạng (Network Attached Storage)

HỆ THỐNG LƯU TRỮ ỔN ĐỊNH (Stable Storage System)

- Đảm bảo thông tin lưu trữ luôn tồn tại dù bất kỳ lỗi nào xảy ra trong quá trình đọc/ghi.
- Các vấn đề xảy khi đọc/ghi đĩa thường:
  - Ghi thành công: block đích chứa thông tin mới
  - Thất bại một phần: block đích chứa thông tin sai
  - Thất bại hoàn toàn: block đích chứa thông tin như cũ
- Hiện thực: dùng 2 block vật lý cho 1logical block
  - Ghi thông tin vào block (vật lý) thứ 1 rồi thứ 2.
  - Việc ghi thành công ⇔ block thứ 2 ghi xong
  - Kiểm tra sự giống nhau của 2 block → phát hiện lỗi và xử lý để đảm bảo tính nhất quán thông tin

CÁC KỸ THUẬT TĂNG HIỆU SUẤT ĐĨA CỨNG

- Lưu dữ liệu truy xuất thường xuyên trong bộ nhớ
  - virtual disk, disk caching
- Kỹ thuật bufferring
  - Read – ahead, write-behind
- Defragment đĩa ➔ giảm seek time
- Phân vùng đĩa ➔ phân mảnh bị giới hạn
- Interleaving ➔ giảm latency time
- Nén dữ liệu
- Đặt các ứng dụng/ file/ directory structure ở giữa đĩa
- Dùng hệ nhiều đĩa cứng (RAID system)
- Hiện thực giải thuật định thời đĩa băng phần cứng