

# HỆ ĐIỀU HÀNH

(Operation System)

Th.S Nguyễn Hoàng Thành

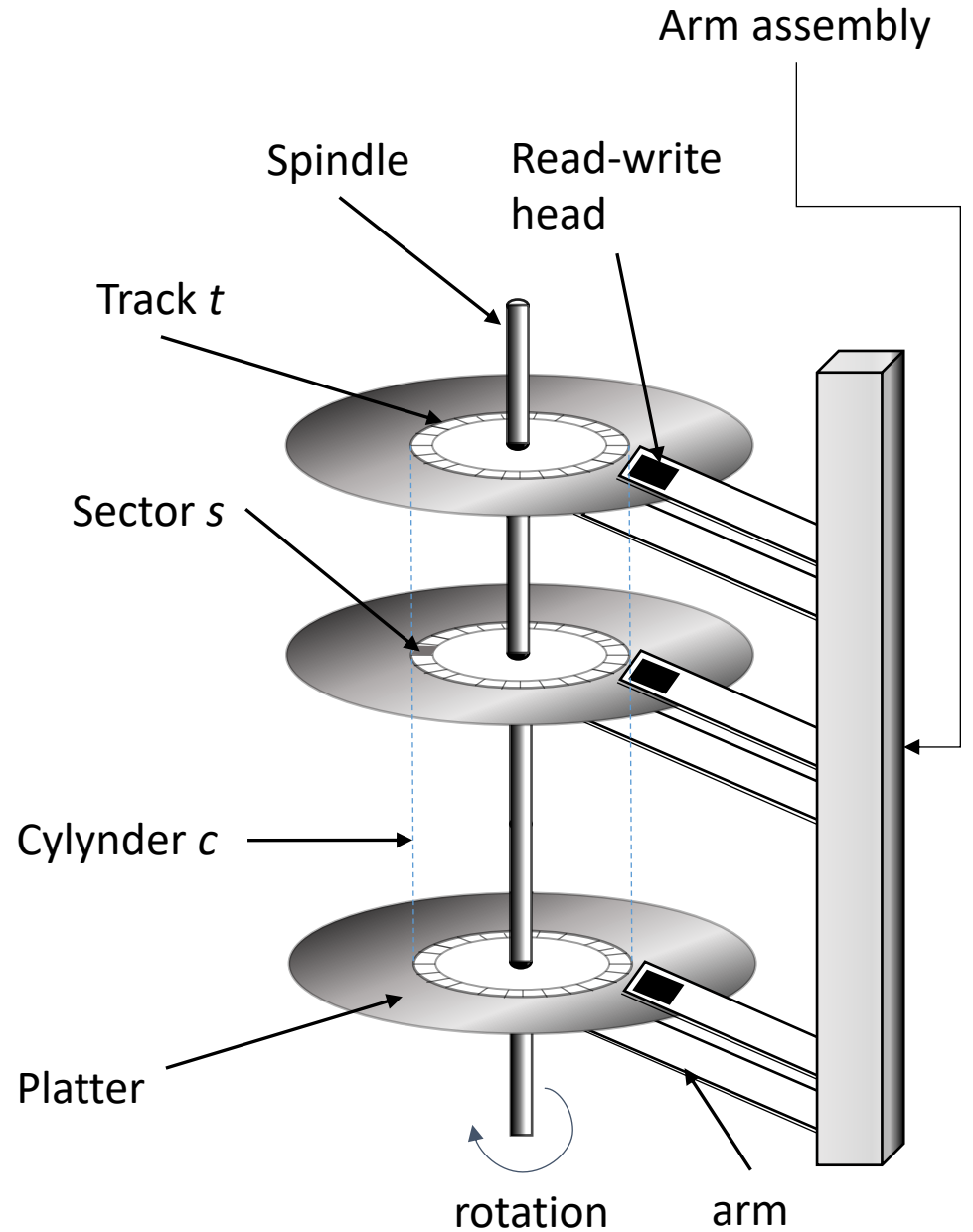
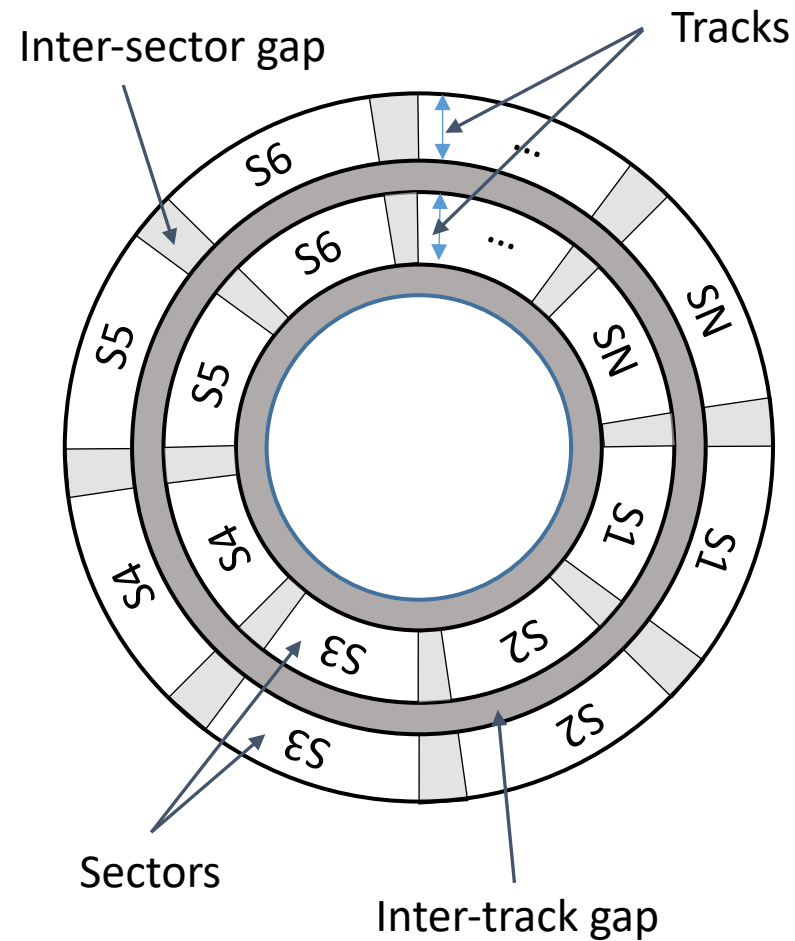
Email: [thanhnh@ptithcm.edu.vn](mailto:thanhnh@ptithcm.edu.vn)

Tel: 0909 682 711

# QUẢN LÝ HỆ THỐNG TẬP TIN

- **Đĩa cứng, tập tin, thư mục, bảng thư mục**
- **Các phương pháp cài đặt hệ thống tập tin**
- **Quản lý danh sách các khối trống**
- **Quản lý sự an toàn của hệ thống tập tin**
- **Một số hệ thống tập tin: MSDOS/Windows, UNIX**

# Đĩa cứng (hard disk)



# Tập tin (file)

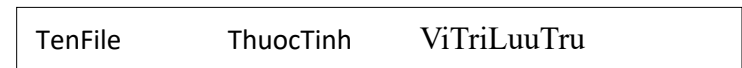
- Tập hợp thông tin được đặt tên và lưu trữ trên đĩa
- Thuộc tính file:
  - Tên file (file name)
  - Kiểu file (file type) - file văn bản và file nhị phân
  - Vị trí file: Danh sách các khối (cluster) trên đĩa đã cấp cho file
  - Kích thước file
  - Ngày giờ tạo file, người tạo file
  - Loại file: file ẩn, chỉ đọc, hệ thống, lưu trữ, file/thư mục
  - Bảo vệ file

# Cấu trúc thư mục (Directory Structure)

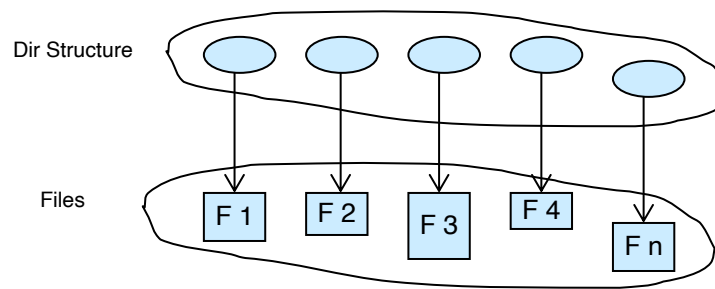
- Dùng để quản lý tất cả các file trên đĩa.
- Cấu trúc thư mục được ghi trên đĩa và gồm nhiều mục (Directory Entry)
- Mỗi mục lưu thông tin của một file hoặc thư mục con
- Các thao tác trên thư mục có thể là: Create, Delete, Open, Close, Read, Rename, Link, Unlink,...



Cấu trúc thư mục

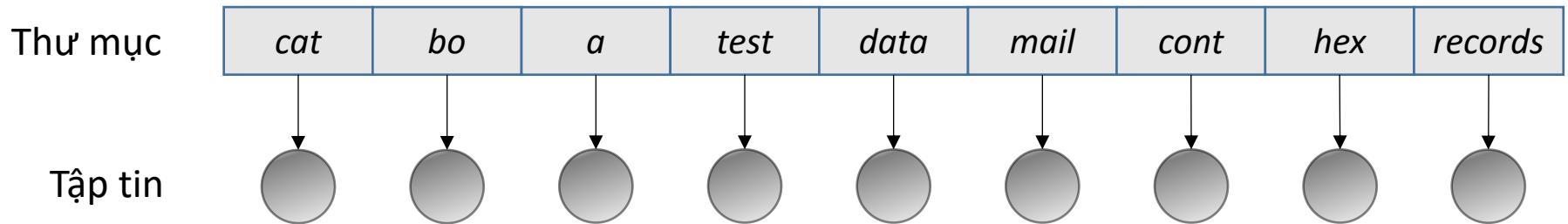


Một mục trong cấu trúc thư mục

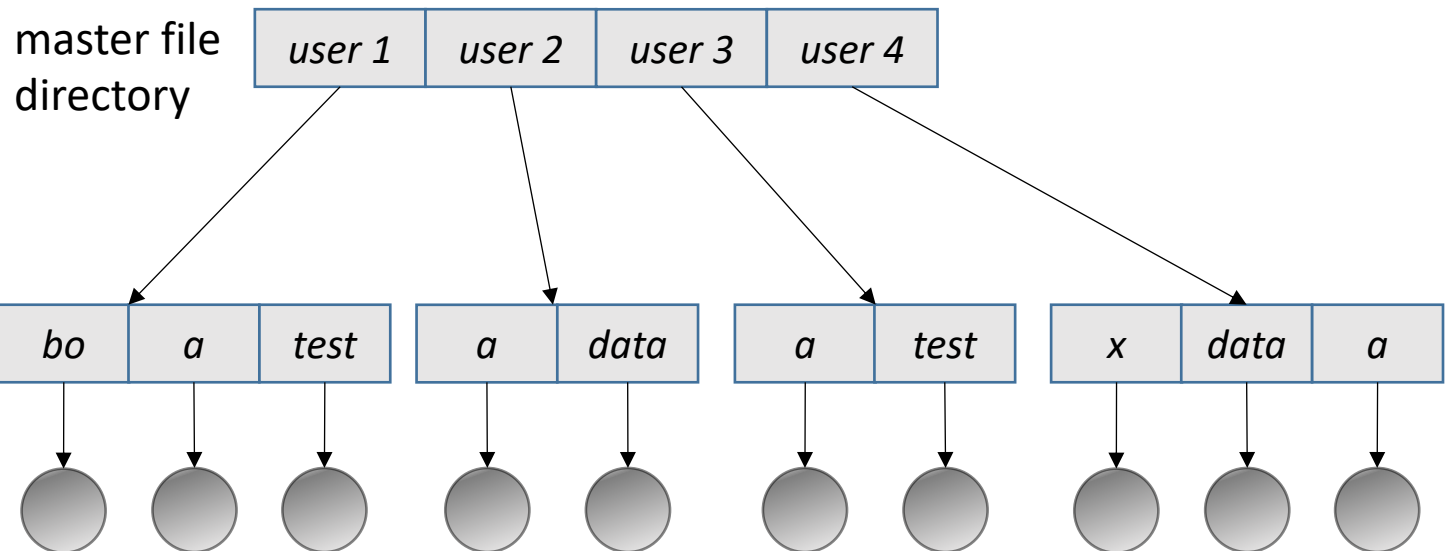


# Cấu trúc thư mục

## Thư mục một cấp: Single Level Directory

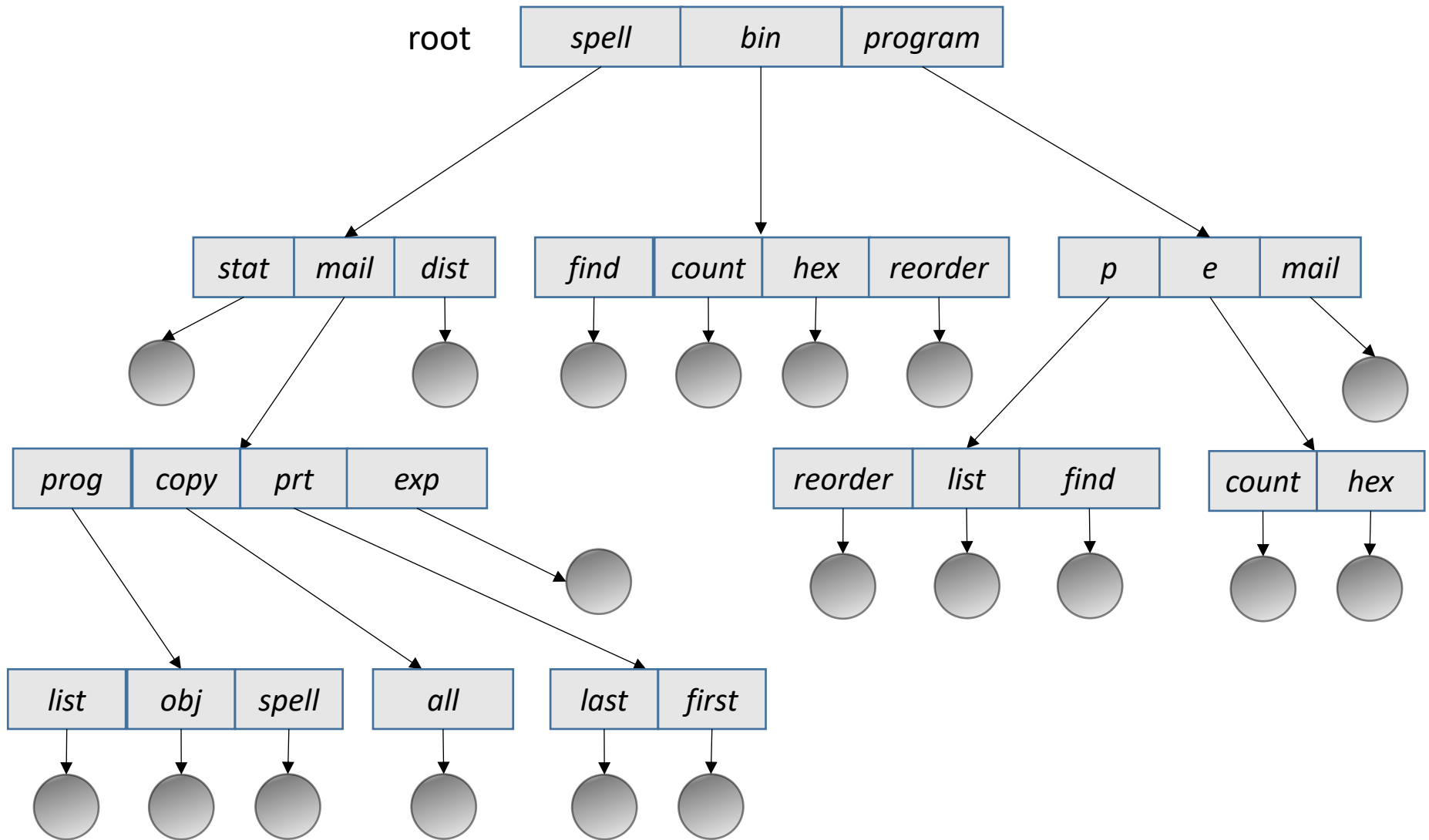


## Thư mục hai cấp: Two Level Directory

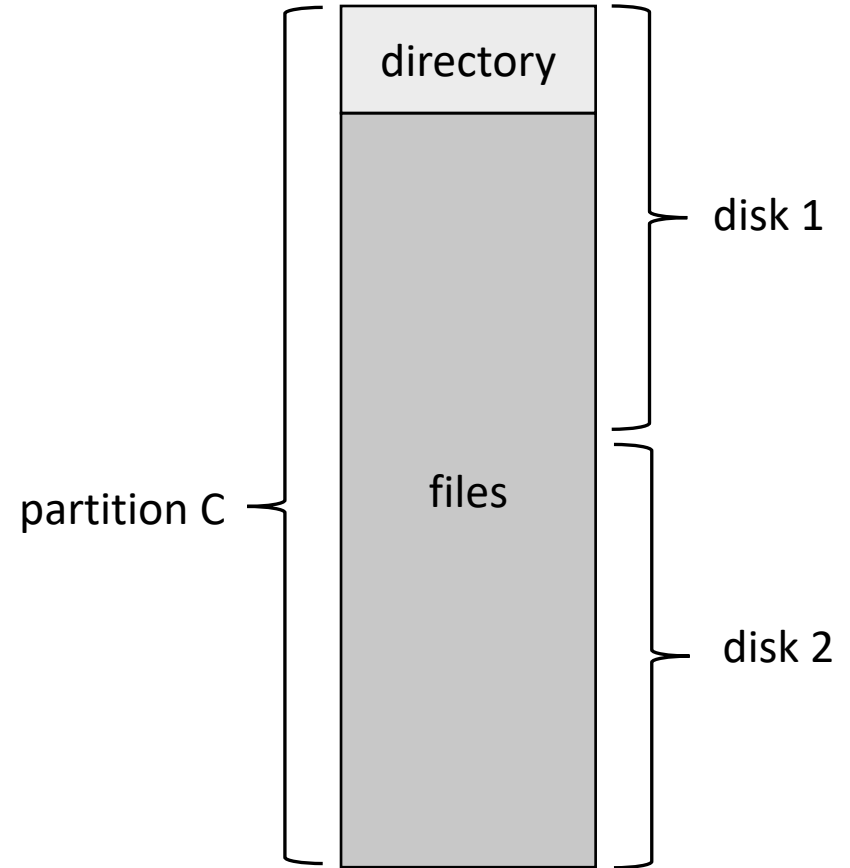
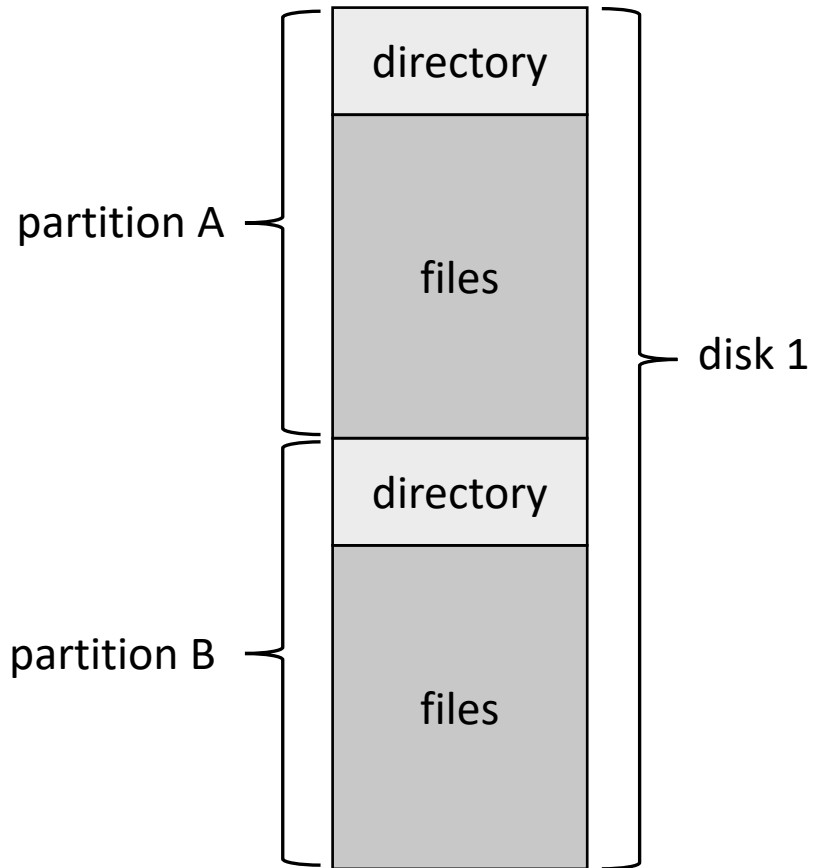


# Cấu trúc thư mục

Thư mục đa cấp: Tree-structured Directory

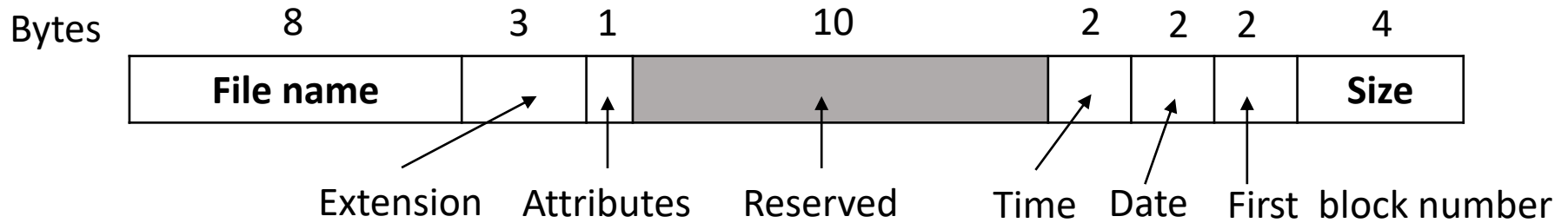


# Tổ chức phân vùng đĩa

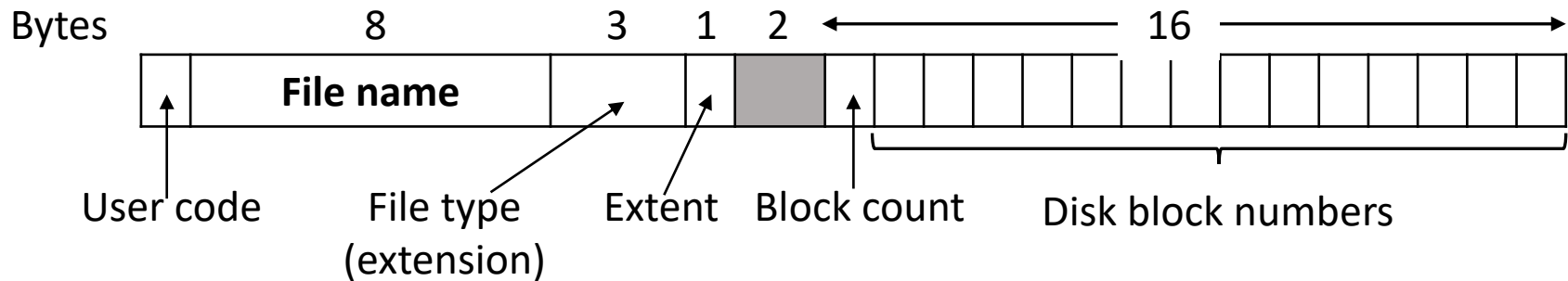




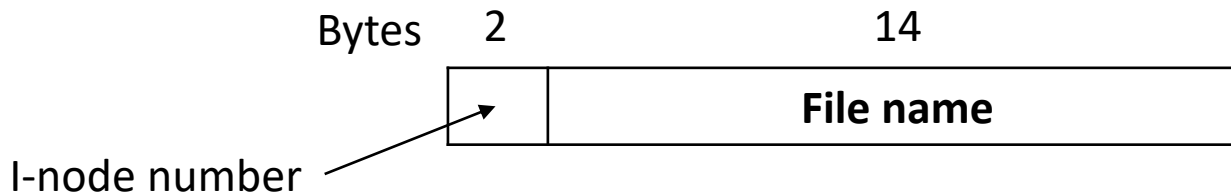
# Bảng thư mục (Directory Table)



**Cấu trúc một mục trong bảng thư mục của MSDOS/WINDOWS (FAT)**



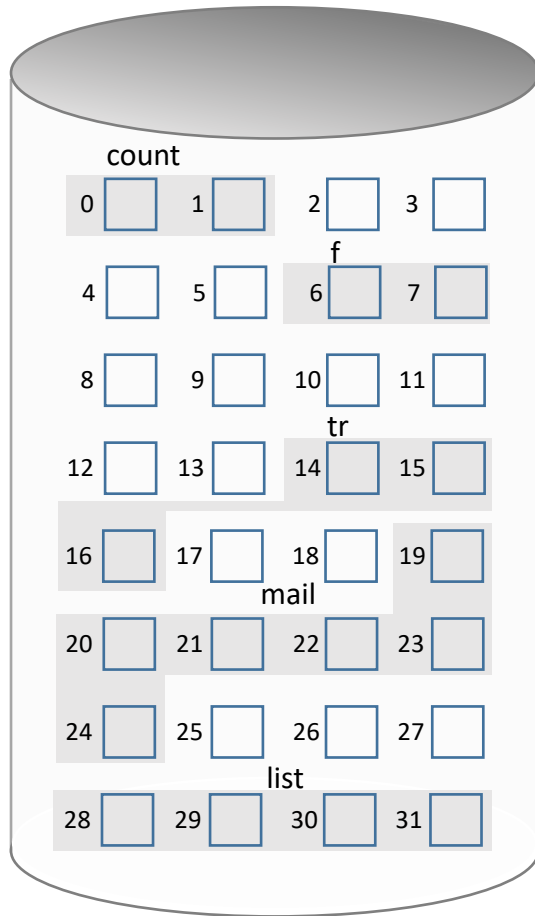
**Cấu trúc một mục trong bảng thư mục của CP/M**



**Cấu trúc một mục trong bảng thư mục của UNIX**

# Cấp phát khối nhớ liên tục

- Start: số hiệu khối bắt đầu
- length : số khối đã cấp cho file



directory

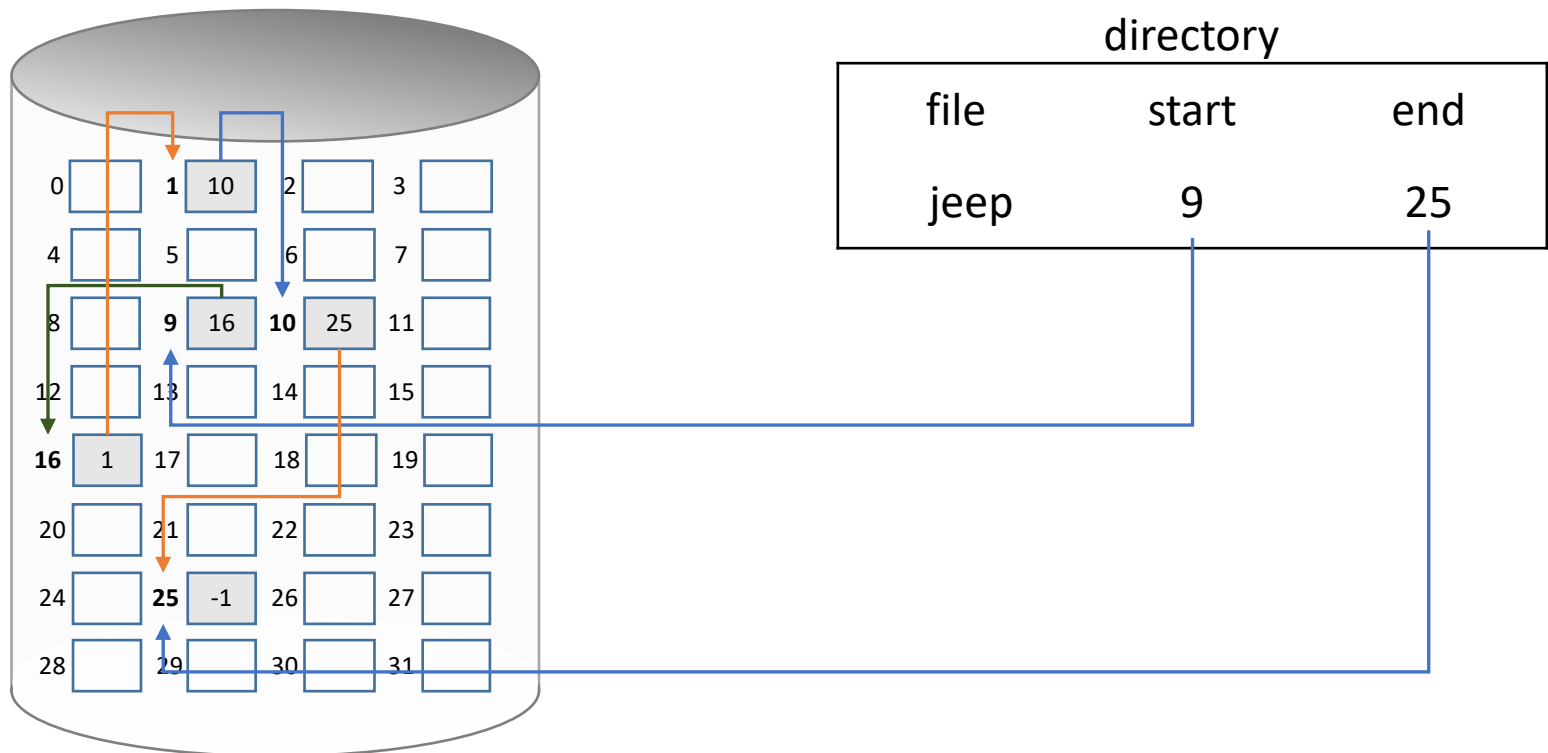
File	Start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

# Cấp phát khối nhớ không liên tục

- Bảng thư mục
- Sử dụng thêm một trong các cấu trúc:
  - danh sách liên kết
  - bảng chỉ mục
  - bảng cấp phát file
  - bảng I-Nodes

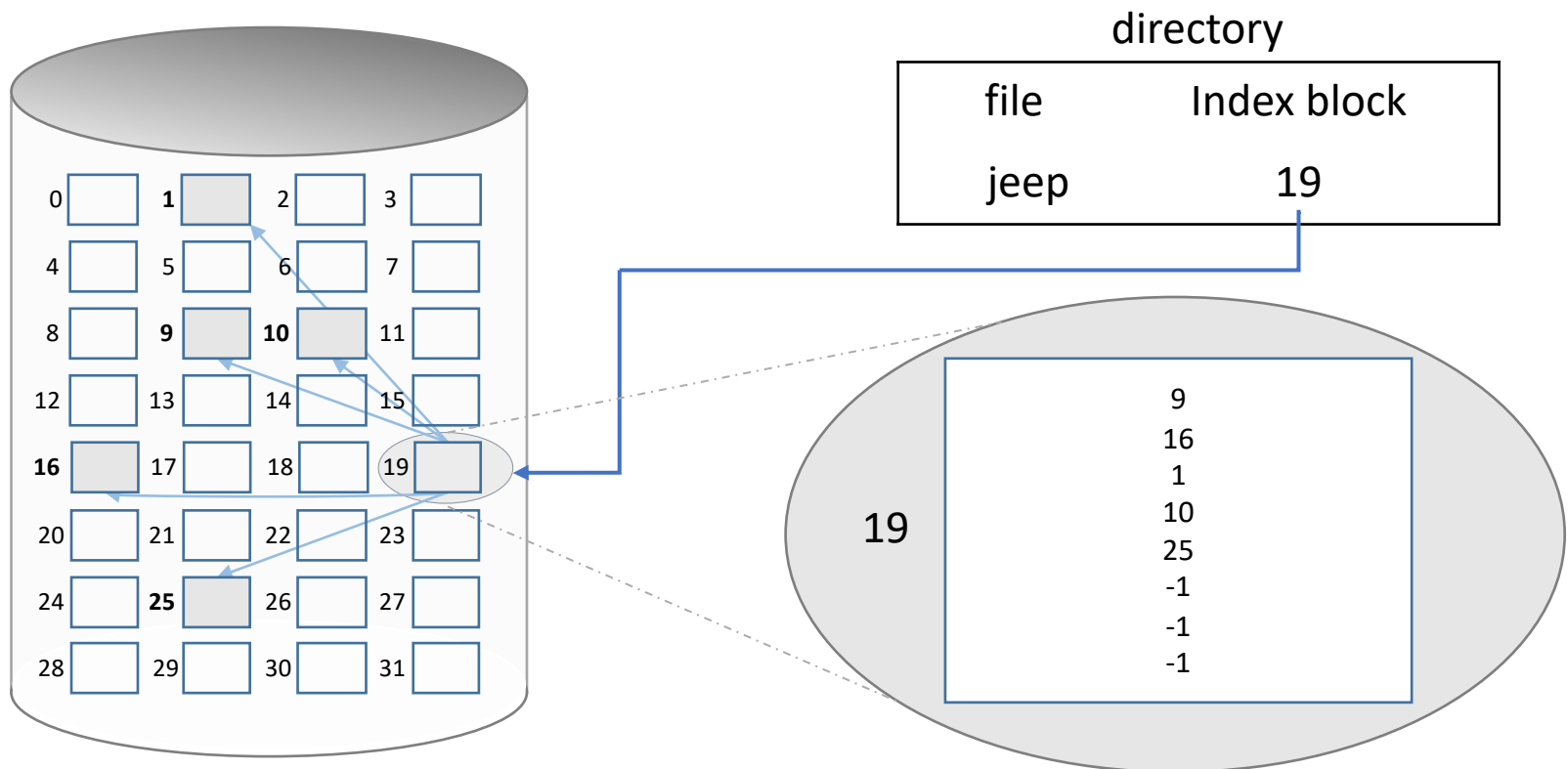
# Cấp phát khối nhớ không liên tục – Danh sách liên kết

- start: số hiệu của khối đầu tiên
- end: số hiệu của khối kết thúc
- Mỗi khối:
  - một số byte đầu/cuối để lưu số hiệu khối kế tiếp của file (4bytes)
  - phần còn lại của khối sẽ lưu dữ liệu của file



# Cấp phát khối nhớ không liên tục - Bảng chỉ mục

- Mỗi file có một **bảng chỉ mục** chiếm một hoặc vài khối
- Bảng chỉ mục chứa tất cả các số hiệu khối của một file.
- Một **mục trong bảng thư mục** sẽ lưu **số hiệu khối chứa bảng chỉ mục** của file



## Cấp phát khối nhớ không liên tục – Bảng chỉ mục – Ví dụ

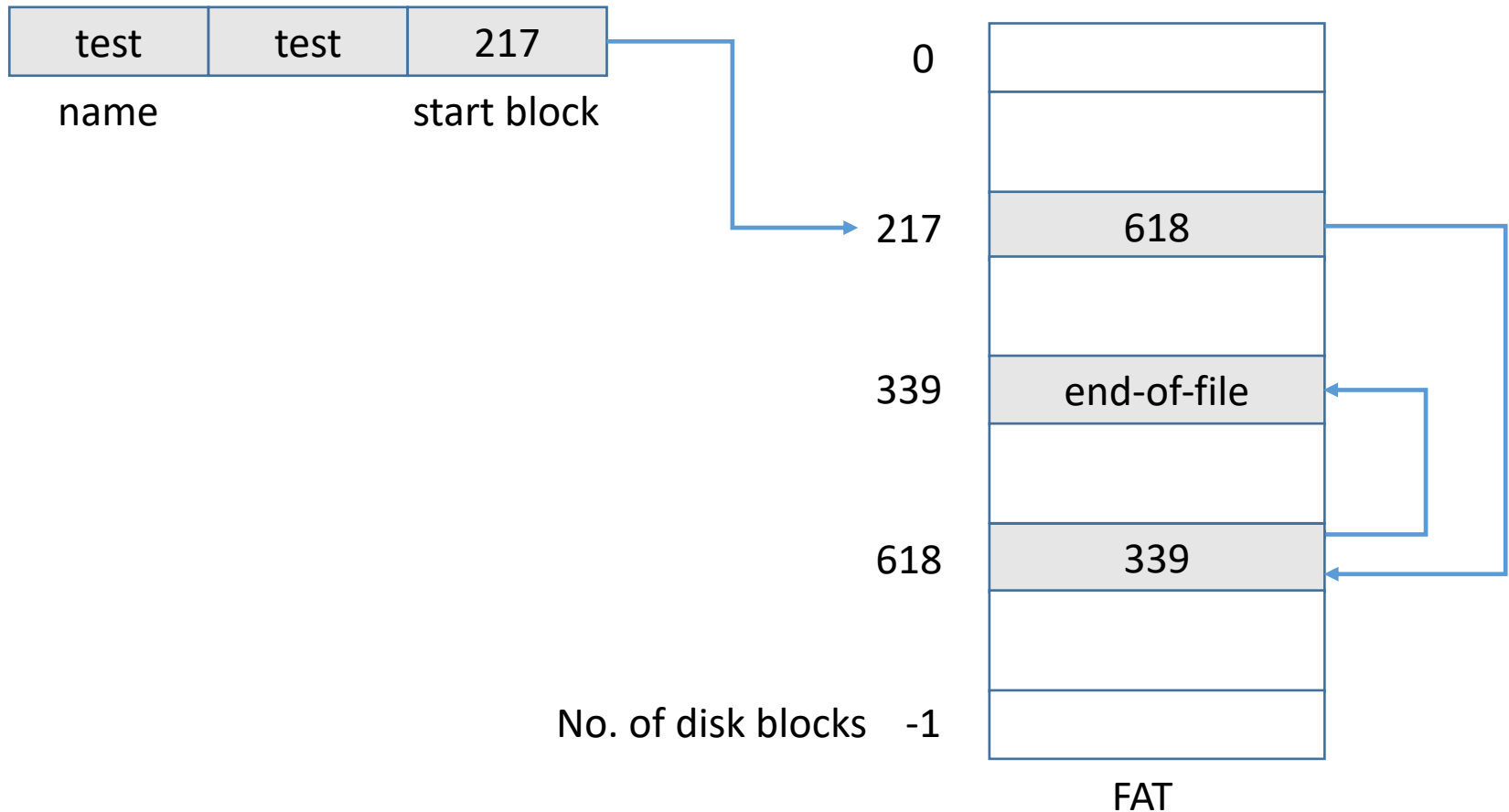
- Đĩa cứng có dung lượng 32 MB
- kích thước 1 khối là 512 Bytes.
- Hỏi kích thước một bảng chỉ mục là bao nhiêu nếu muốn quản lý file có kích thước lớn nhất là 256K?
- $32 \text{ MB} = 32 \text{ MB} = 2^5 \times 2^{10} \times 2^{10} \text{ B} = 2^{25} = 2^{16}$  khối (mỗi khối 512 Bytes ( $2^9$ ))  $\Rightarrow$  có  $2^{16}$  địa chỉ trên đĩa  $\Rightarrow$  **mỗi phần tử bảng chỉ mục cần 2 byte.**
- File kích thước 256KB = 256 x 1024 byte = 512 khối  $\Rightarrow$  **bảng chỉ mục cần có 512 phần tử**  $\Rightarrow$  một bảng chỉ mục chiếm hai khối (512 phần tử x 2 bytes)!

# Cấp phát khối nhớ không liên tục - Bảng cấp phát File (FAT)

- Mỗi mục trong bảng thư mục chứa số hiệu của khối đầu tiên
- Số hiệu các khối còn lại của file sẽ được lưu trong một bảng gọi là bảng cấp phát file (bảng FAT).
- Để bảo vệ các số hiệu khối đã cấp cho file, truy xuất file ngẫu nhiên dễ dàng hơn, kích thước file dễ mở rộng
- FAT bị giới hạn bởi kích thước bộ nhớ dành cho nó.
- MSDOS, OS/2, Windows (FAT) sử dụng để quản lý File.

# Cấp phát khối nhớ không liên tục - Bảng cấp phát File (FAT)

- Ví dụ 1: file test.txt được lưu trữ lần lượt: 217, 618, 339.



Mô hình cấp phát không liên tục, sử dụng bảng FAT



# Cấp phát khối nhớ không liên tục - Bảng cấp phát File (FAT)

■ Ví dụ 2:

■ file A: 4, 7, 2, 10, 12;

■ file B: 6, 3, 11, 14

Bảng thư mục

...	(A, 4)	...	(B, 6)	...
-----	--------	-----	--------	-----

Entry của A

Entry của B

File A starts here

File B starts here

Unused block

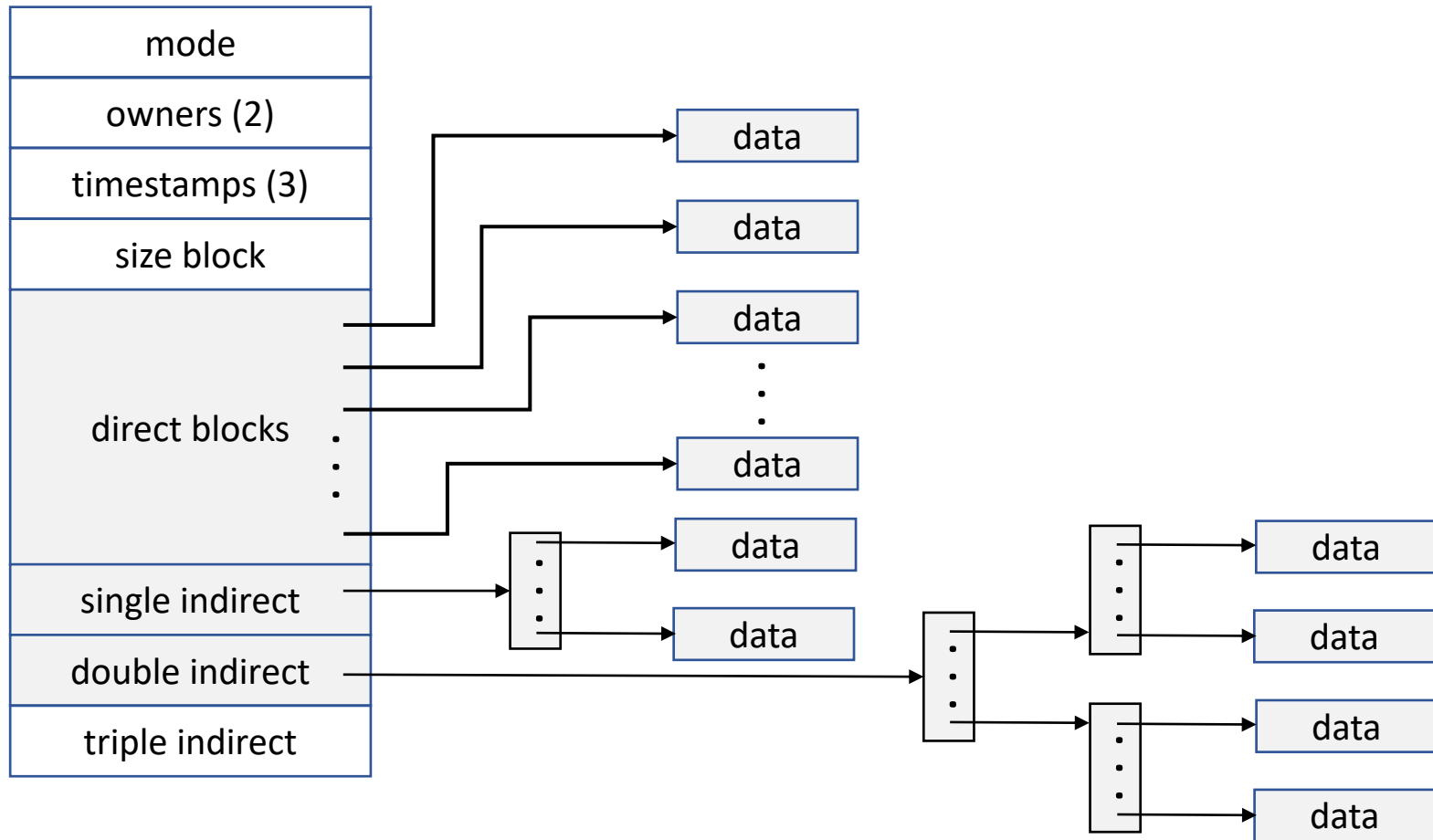
Bảng FAT

Physical  
block

	0
	1
10	2
11	3
7	4
	5
3	6
2	7
	8
	9
12	10
14	11
-1	12
	13
-1	14
	15

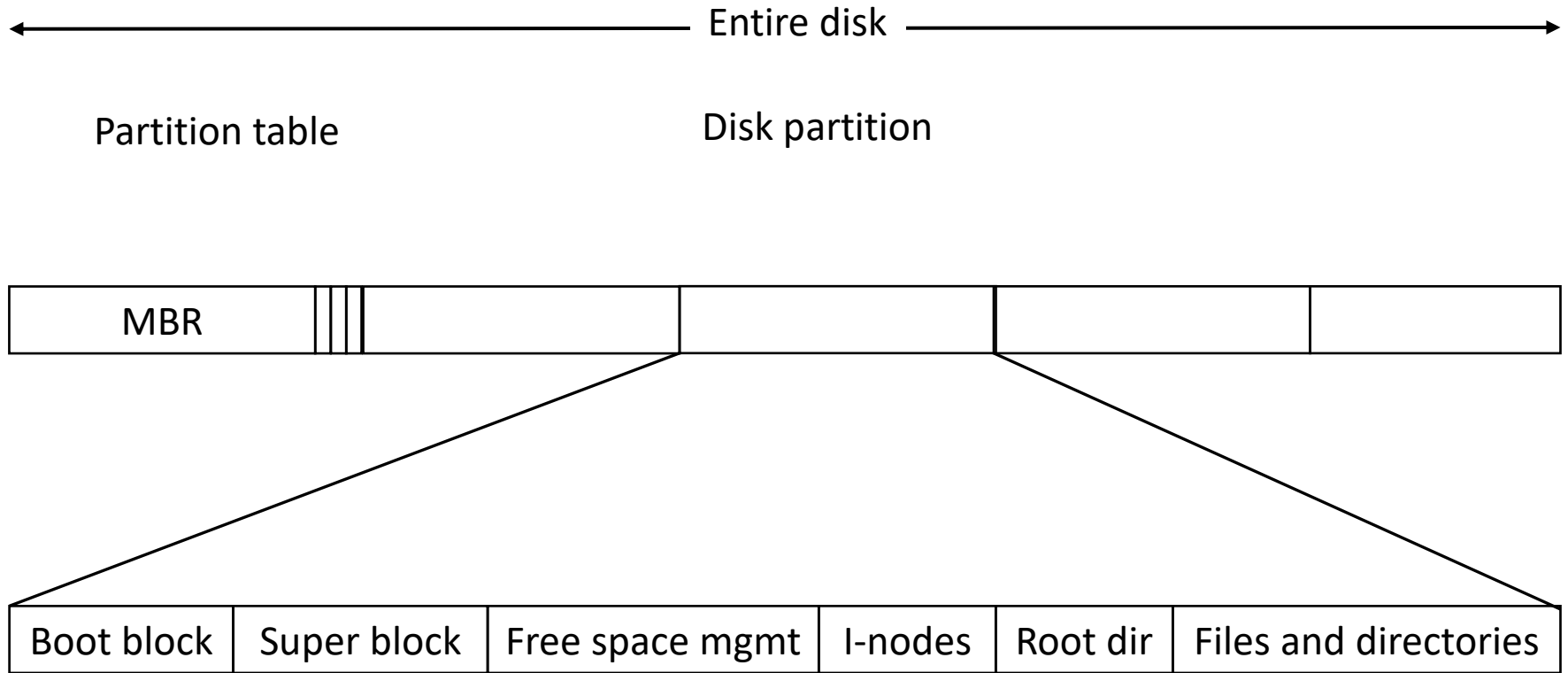
**Mô hình cấp phát không liên tục, sử dụng bảng FAT, có 2 file**

# Cấp phát khối nhớ không liên tục - Bảng I-nodes



Cấu trúc một I-node trong bảng I-nodes

# Phương pháp tổ chức quản lý đĩa bằng I-nodes



**Mô hình cấp phát không liên tục, sử dụng bảng I-nodes tổng quát**

# Phương pháp tổ chức quản lý đĩa bằng I-nodes

Thư mục gốc

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

Tra cứu usr  
tìm được i-node 6

I-node 6  
Dành cho /user

Mode size times
132

I-node 6 cho  
biết thông tin  
/user được lưu  
tại block 132

Block 132 là  
thư mục /usr

1	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

/usr/ast là i-  
node 26

I-node 6  
Dành cho /user

Mode size times
406

I-node 6 cho  
biết thông tin  
/usr/ast được  
lưu tại block  
406

Block 406 là thư  
mục /usr/ast

26	.
6	..
64	grants
92	books
60	mbox
81	minix
17	src

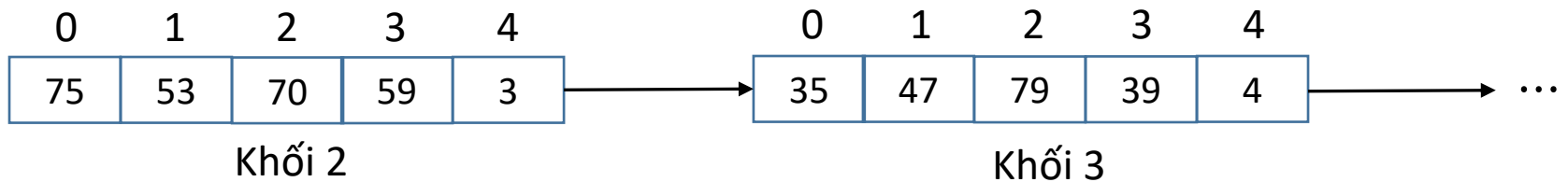
/usr/ast là i-  
node 26

**mô hình cấp phát không liên tục, sử dụng bảng I-nodes chi tiết**

# Quản lý các khối trống

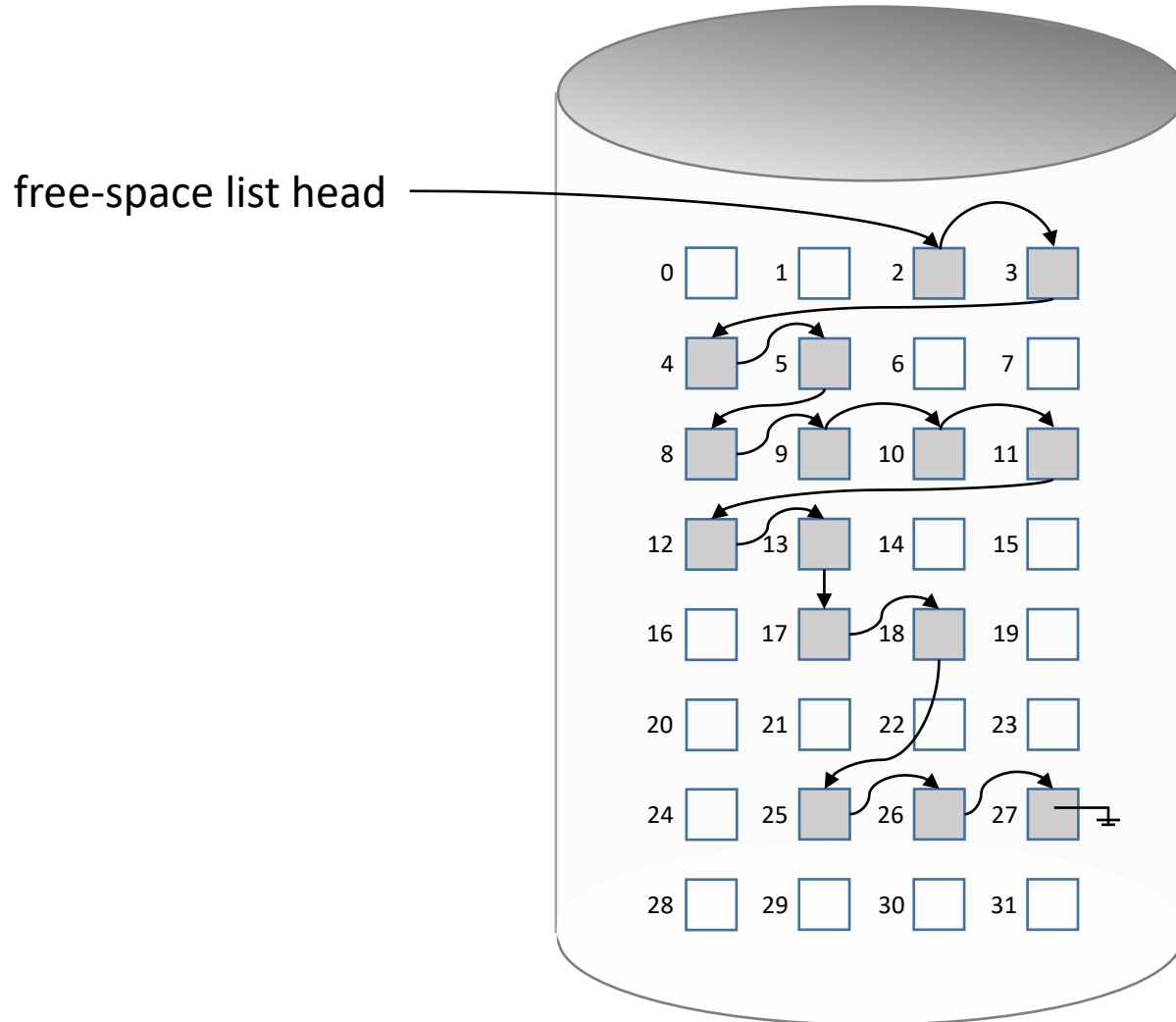
- Danh sách liên kết
  - Mỗi nút trong dslk là một khối (block) chứa một bảng:
    - Các số hiệu khối trống
  - Phần tử cuối của bảng lưu số hiệu khối tiếp theo trong danh sách
- Vector bit (dãy bit)
  - Bit thứ  $i = 1$  là khối (cluster) thứ  $i$  trống, bit thứ  $i = 0$  là khối thứ  $i$  đã sử dụng.
  - Vector bit được lưu trên một hoặc nhiều khối (cluster) đĩa, khi cần sẽ đọc vào bộ nhớ để xử lý nhanh

# Quản lý các khối trống – Danh sách liên kết



- Ví dụ: Một đĩa 20M, dùng khối (cluster) có kích thước 1 K. Để quản lý đĩa này, nếu đĩa hoàn toàn trống thì DSLK cần bao nhiêu khối (block) (số nút tối đa của dslk)?
  - $20M = 20 \times 2^{10}$  khối  $\sim 2^{15}$  khối  $\Rightarrow$  cần dùng 16 bit = 2 byte để lưu một số hiệu khối (cluster)
  - $\Rightarrow$  1 block = 1024 byte lưu được 511 số hiệu khối (cluster) trống
  - $\Rightarrow$  để quản lý đĩa có 20M hoàn toàn trống, dslk cần  $20M = 20 \times 2^{10} / 511 \sim 40$  khối (block)!
- **Tốn khá nhiều khối nhớ cho dslk nếu đĩa hoàn toàn trống, nhưng sẽ ít tốn khối nhớ cho dslk nếu đĩa gần đầy.**

# Quản lý danh sách các khối trống sử dụng danh sách liên kết



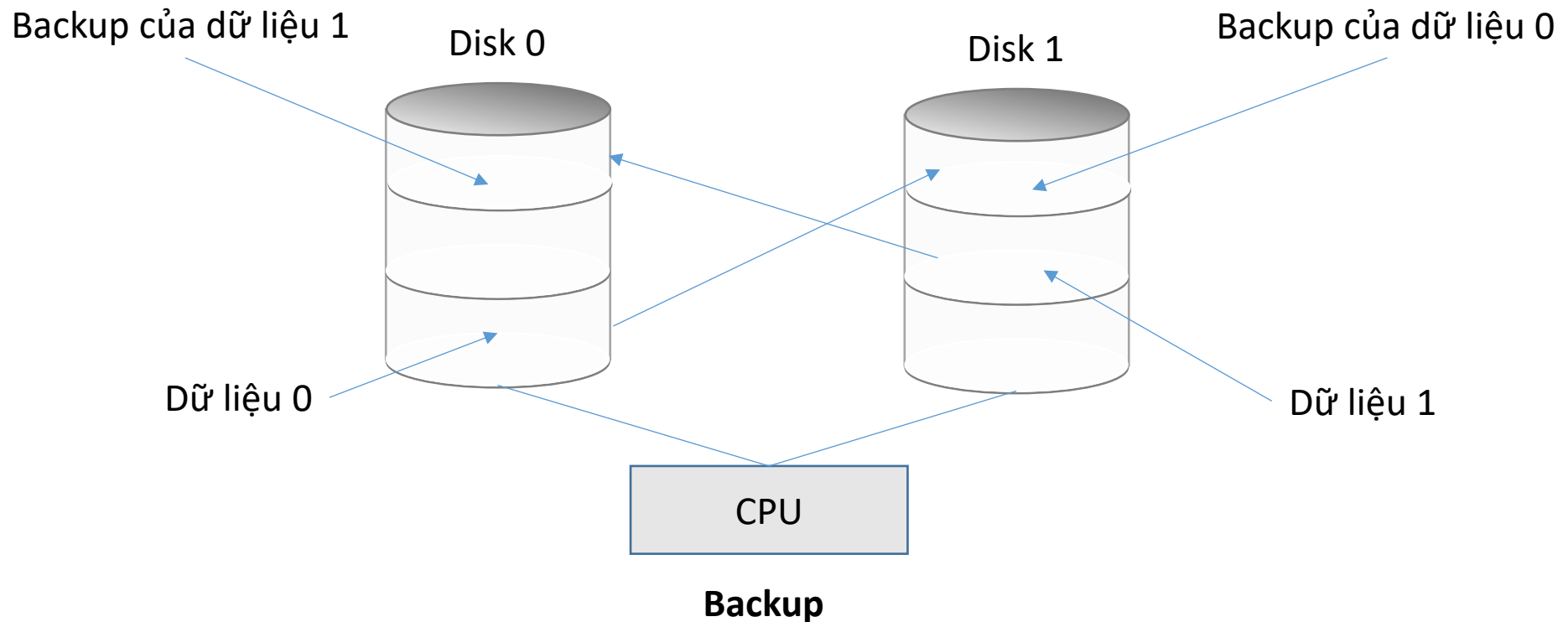
# Quản lý các khối trống – Vector bit

- Vector bit ít tốn khối nhớ hơn là dslk nhưng kích thước vector bit là cố định và HĐH cần đồng bộ vector bit trong bộ nhớ và vector bit trên đĩa.
- Ví dụ: Một đĩa 20M, dùng khối có kích thước 1 K. Để quản lý đĩa này, nếu đĩa hoàn toàn trống và dùng vector bit, hãy tính kích thước vector bit.
- Đĩa 20M có  $20 \times 210$  khối (cluster) nên kích thước vector bit là  $20 \times 210 \text{ bit} = 20 \times 210 / 8 / 210 \text{ KB} \sim 3 \text{ khối}$ .



# Quản lý sự an toàn của hệ thống tập tin

- **Quản lý khối bị hỏng**
  - Dùng phần mềm: file
  - Dùng phần cứng: sector
- **Sao lưu file (Backup)**



# Hệ thống tập tin của MSDOS/Windows (FAT)

- Boot Sector
- bảng FAT
- bảng ROOT DIR

Boot sector	Bảng FAT1	Bảng FAT2	Bảng DIR	DATA
-------------	-----------	-----------	----------	------

# Boot Sector

- Ở sector đầu tiên, track 0, side 0 của đĩa mềm, đối với đĩa cứng thì vị trí này là bảng partition, rồi mới tới boot sector của partition thứ nhất, đối với các partition khác, boot sector là sector đầu tiên.
- Boot sector chứa bảng tham số đĩa BPB (Bios Parameter Block) và chứa đoạn mã boot dùng để nạp các file hệ thống.
- Trên máy IBM PC sau khi thực hiện thao tác POST (Power On Self Test), ROM BIOS tìm boot sector hợp lệ, đọc boot sector vào địa chỉ 0X7C00, gán CS=0000h, IP=7C00h và cho thực thi lệnh đầu tiên trong boot sector (lệnh JMP).

# Cấu trúc Boot sector

Offset(hex)	Size(byte)	Content	Giải thích
00	3	JMP	Lệnh nhảy đến đầu đoạn mã boot
03	8	Version	Phiên bản hệ điều hành
0B	2	SectSiz	số byte /một sector, thường là 512 (đây là bắt đầu của BPB)
0D	1	ClustSize	số sector / một cluster (khác 0)
0E	2	ResSecs	số sector dành riêng trước bảng FAT, tính luôn boot sector (đối với FAT12, FAT16 =1, FAT32=20h)
10	1	FatCnt	số bảng FAT (thường là 2)
11	2	RootSiz	Số mục trong bảng ROOT DIR
13	2	TotSecs	Tổng số sector trên đĩa hay trên một partition (<=65535)
15	1	MediaDescriptor	Byte nhận dạng đĩa (F8:đĩa cứng, F0:1.44 MB)

# Cấu trúc Boot sector (tt)

Offset(hex)	Size(byte)	Content	Giải thích
16	2	FatSize	Số sector /một bảng FAT ( $\leq 65535$ ). FAT32=0
18	2	TrkSecs	số sector / một track
1A	2	HeadCnt	số đầu đọc
1C	4	HidnSec	số sector ẩn (ở giữa bảng partition và partion)
20	4	BigTotSecs	Tổng số sector trên đĩa hay trên một partition ( $> 65535$ )
24	4	BigFatSize	Số sector /một bảng FAT ( $> 65535$ )
...	...	...	...
1D			(kết thúc BPB)
1E			đầu đoạn mã boot
...			...
1FF			cuối đoạn mã boot

# Bảng FAT

- Nằm sau boot sector.
- Mỗi mục của FAT quản lý một khối dữ.
- Kích thước khối được lưu trong boot sector thông thường từ 1 đến 8 sector.
- Có ba loại FAT là FAT 12 (4096 khối) và FAT 16 (64 K khối), FAT 32 (4G khối trên một partition).

# Cấu trúc một mục (entry) trong bảng FAT

(0) 000	Cluster còn trống
(0)002 - (F)FEF	Cluster chứa dữ liệu của File, giá trị này là số hiệu cluster kế.
(F)FF0 - (F)FF6	Dành riêng, không dùng
(F)FF7	Cluster hỏng
(F)FF8 - (F)FFF	Cluster cuối cùng của chuỗi (kết thúc file)

# ROOT DIR

- Nằm ngay sau FAT
- Mỗi mục của bảng DIR là 32 byte.
- Khi hệ thống mở một File/ thư mục, MS-DOS tìm tên File/thư mục trong bảng ROOT DIR, lấy số hiệu khối đầu đã cấp cho file/thư mục và tìm các số hiệu khối tiếp theo trong bảng FAT.

Boot sector	Bảng FAT1	Bảng FAT2	Bảng DIR	DATA
-------------	-----------	-----------	----------	------



# Cấu trúc một mục trong bảng ROOT DIR/ SUB DIR

Kích thước	Thuộc tính
8 byte	Tên File
3 byte	Phần mở rộng
1 byte	Thuộc tính : A – D – V – S - H - R
10 byte	Dành riêng để sử dụng sau này.
2 byte	Giờ : 5 bit cho giờ, 6 bit cho phút, 5 bit cho giây (thiếu 1, nên lưu đơn vị 2 giây)
2 byte	Ngày : 7 bit cho năm (từ 1980), 4 bit cho tháng, 5 bit cho ngày.
2 byte	Số hiệu khối đầu tiên của file
4 byte	Kích thước File

## Giá trị của byte thuộc tính:

**1** : File chỉ đọc (Read Only)

**2** : File ẩn (Hidden)

**4** : File hệ thống (System)

**8** : nhãn đĩa (Volume)

**16** : thư mục con (Directory)

**32** : File chưa được sao lưu (Archive)

# Hệ thống tập tin của MSDOS/Windows (FAT)

- Xét đĩa 1.44MB, được format dưới hệ điều hành MS-DOS/WINDOWS (FAT12).

<b>Số sector</b>	1	9	9	14	Còn lại
<b>Lưu trữ</b>	Boot sector	FAT12	FAT12	DIR	DATA

<b>Byte</b>	0-7	8-10	11	12-21	22-23	24-25	26-27	28-31
<b>Lưu trữ</b>	Tên file	Phần mở rộng	ADVSHR	Dành riêng	Giờ	Ngày	Số hiệu khối đầu	Kích thước file

Bảng DIR=14 sector = 7168 byte = 224 entry (mỗi entry 32 byte), 1 sector = 16 entry

<b>Byte</b>	0, 1, 2	3, 4, 5	...	4606, 4007
<b>Lưu trữ</b>	e0, e1 (số hiệu đĩa)	e2, e3	...	e3070, e3071

Bảng FAT12 = 9 secotr = 4608 bytes = 3072 entry (Mỗi entry 12 bit)

# Hệ thống tập tin của Windows NT

- **Cấu trúc volume của NTFS:** partition boot sector, Master File Table, các file hệ thống, vùng dữ liệu.
- **Partition boot sector:** là sector khởi động của partition.
- **Master File Table (MFT):**
  - lưu các thông tin về tất cả file/thư mục + danh sách các khối trống.
  - được tổ chức thành nhiều dòng.
- **Các file hệ thống:** (1Mb)
  - MFT2: bản sao của MFT
  - Log file: thông tin dùng cho việc phục hồi.
  - Cluster bitmap: biểu diễn thông tin lưu trữ của các cluster
  - Bảng định nghĩa thuộc tính: định nghĩa các kiểu thuộc tính hỗ trợ cho volume.

# Hệ thống file của UNIX

- Các khối sau: khối boot, khối đặc biệt, I-nodes, các khối dữ liệu.
- Khối boot: chứa mã khởi động của hệ thống.
- Khối super block : chứa thông tin về toàn bộ hệ thống file, bao gồm:
  - Kích thước của toàn bộ hệ thống file.
  - Địa chỉ của khối dữ liệu đầu tiên.
  - Số lượng khối trống và danh sách khối trống.
  - Số lượng I-node trống và danh sách I-node trống.
  - Ngày super block được cập nhật sau cùng.
  - Tên của hệ thống file.