



# Hệ thống quản lý tập tin

**Môn học: Hệ điều hành**

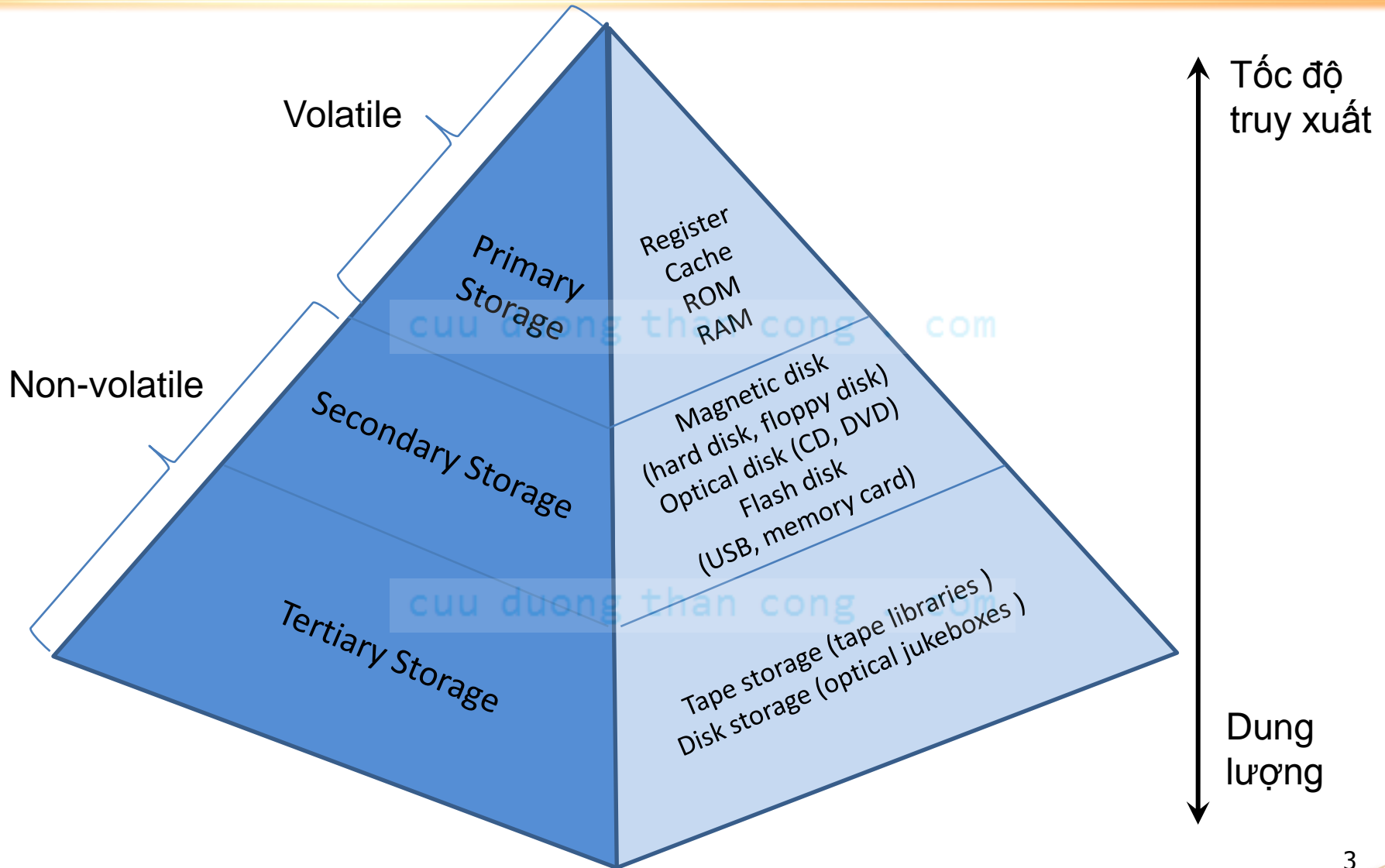
# Mục tiêu

- Trình bày cấu tạo đĩa từ
- Trình bày các khái niệm liên quan hệ thống tập tin
- Trình bày một số vấn đề khi cài đặt hệ thống quản lý tập tin trên đĩa
- Trình bày mô hình tổ chức hệ thống tập tin của một số hệ điều hành thông dụng

cuu duong than cong . com



# Phân cấp hệ thống lưu trữ



# Đĩa từ

- Đĩa từ - là những đĩa phẳng bằng thủy tinh hay bằng kim loại cứng được phủ từ để lưu dữ liệu

cuu duong than cong . com

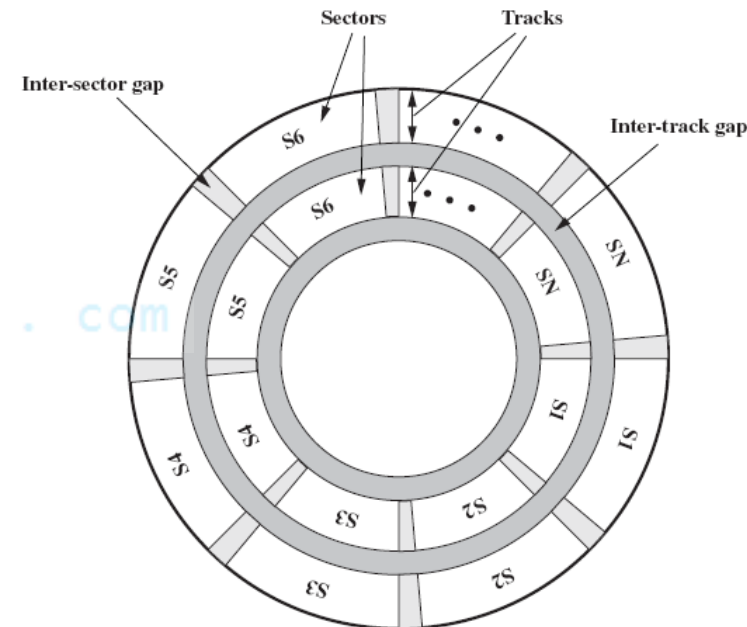


cuu duong than cong . com



# Cấu trúc vật lý

- Gồm nhiều lớp hình tròn, mỗi lớp phủ từ 1 hoặc cả 2 mặt (side)
- Mỗi mặt có tương ứng 1 đầu đọc (head) để đọc hoặc ghi dữ liệu
- Mỗi mặt có nhiều đường tròn đồng tâm (track)
- Mỗi đường tròn được chia nhỏ thành các cung tròn (sector), thông thường mỗi cung chứa 4096 điểm từ ( $\sim 4096$  bit = 512 byte)
- Mỗi lần đọc/ghi ít nhất 1 sector (512 byte)



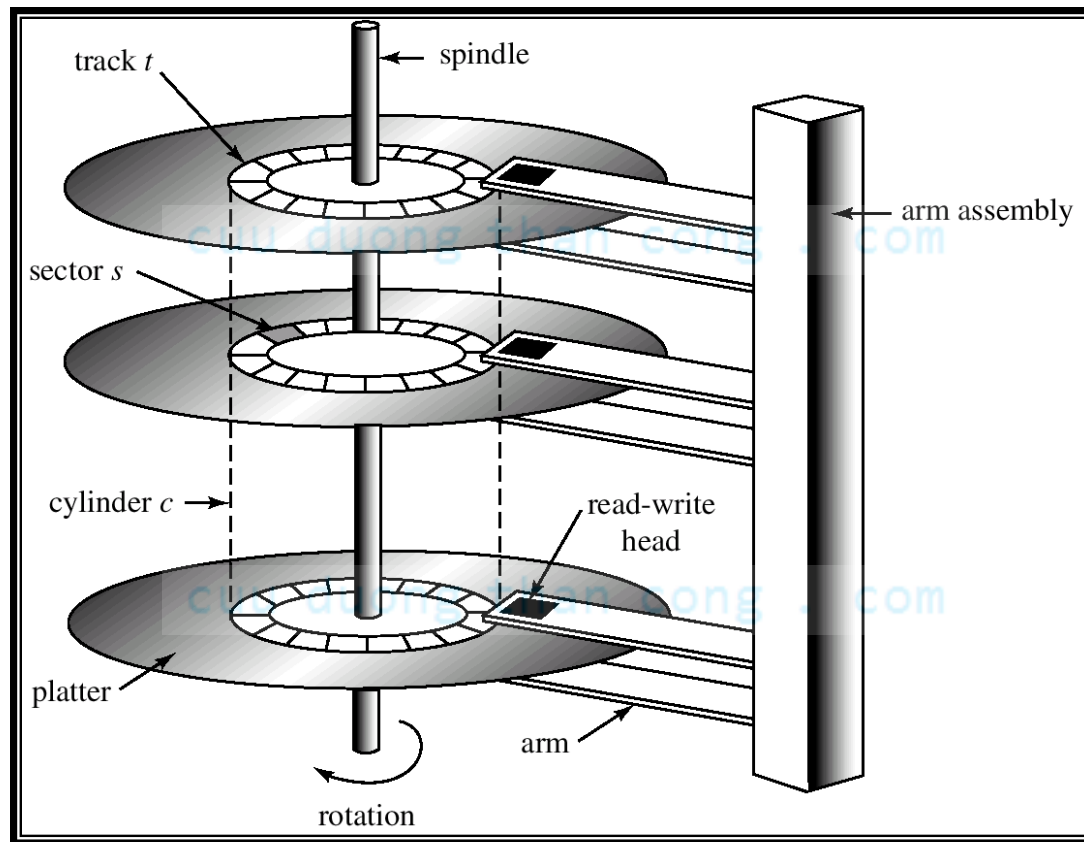
# Truy xuất mức vật lý

- Để truy xuất 1 sector cần phải chỉ ra vị trí của sector đó. Vị trí sector được thể hiện bằng 3 thông số: chỉ số sector, track và head
  - Head được đánh số từ trên xuống bắt đầu từ 0
  - Track được đánh số theo thứ tự từ ngoài vào bắt đầu từ 0
  - Sector được đánh số bắt đầu từ 1 theo chiều ngược với chiều quay của đĩa
- Địa chỉ sector vật lý có ký hiệu: (**sector, track, head**)
- Hàm truy xuất mức vật lý trong C for DOS:  
`int biosdisk (int cmd, int drive, int head, int track, int sector, int nsects, void *buffer)`
- Hàm truy xuất mức vật lý trong C for Windows ???



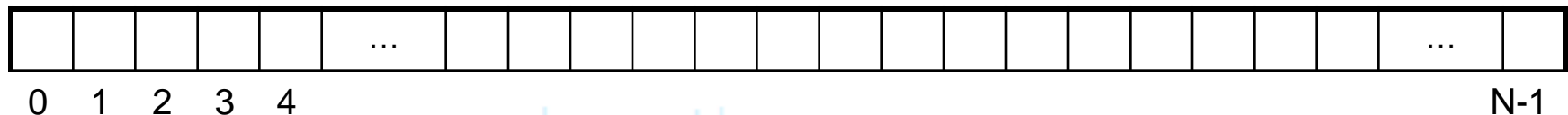
# Cơ chế đọc đĩa

- Access time = Seek time + Rotational time + Read time



# Tổ chức logic

- Do truy xuất mức vật lý phải dùng đến 3 tham số rất bất tiện nên tổ chức logic được đưa ra để dễ hiểu, dễ thao tác, dễ tính toán hơn
- Cylinder: là tập các track có cùng bán kính (cùng số hiệu) trên tất cả các mặt  
→ Nhận xét: truy xuất sector theo từng cylinder sẽ đảm bảo *sau khi truy xuất sector K thì truy xuất sector K+1 là nhanh hơn so với tất cả các sector khác*
- Tổ chức logic là một dãy sector được đánh chỉ số theo từng cylinder, bắt đầu từ 0



- Mỗi lần truy xuất (đọc/ ghi đĩa) chỉ có thể thực hiện trên N sector liên tiếp ( $N \geq 1$ )
- Hàm truy xuất mức logic trong C for DOS:  
`int absread (int drive, int nsects, long lsect, void *buffer).`  
`int abswrite (int drive, int nsects, long lsect, void *buffer);`
- Hàm truy xuất mức logic trong C for Windows ???



# Sector vật lý ↔ Sector logic

- Sector vật lý → Sector logic

$$l = t * st * hd + h * st + s - 1$$

- Sector logic → Sector vật lý

$$s = (l \bmod st) + 1$$

$$t = l \operatorname{div} (st * hd)$$

$$h = (l \operatorname{div} st) \bmod hd$$

Trong đó:

l : chỉ số sector logic

st : số sector /track

h : chỉ số head  
(head)

th : số track /side

t : chỉ số track

hd : tổng số side (head)

s : chỉ số sector vật lý

# Đĩa mềm 1.44 MB

- Có 2 head /disk, 80 track /head, 18 sector /track
- Dung lượng đĩa:  
 $2 \text{ head/disk} * 80 \text{ track/head} * 18 \text{ sector/track} = 2880 \text{ sector/disk}$   
 $= 0.5 \text{ KB/sector} * 2880 \text{ sector/disk} = 1440 \text{ KB/disk} (\sim 1.44 \text{ MB})$
- Sector logic có chỉ số từ 0 đến 2879 và tương ứng với sector vật lý như sau:

Sector Logic	Sector vật lý (Sector, Track, Head)
0	(1, 0, 0)
1	(2, 0, 0)
...	...
17	(18, 0, 0)
18	(1, 0, 1)
19	(2, 0, 1)
...	...
35	(18, 0, 1)
36	(1, 1, 0)
37	(2, 1, 0)
...	...

# Bài tập

1. Một đĩa cứng có 16 head, mỗi mặt có 684 track, và mỗi track có 18 sector thì sẽ có kích thước là bao nhiêu Megabyte ?
2. Cho biết sector vật lý (head 0, track 19, sector 6) tương ứng với sector logic nào trên đĩa mềm 1.44MB
  - a. 347
  - b. 348
  - c. 689
  - d. 690



# Một số khái niệm

- Tập tin
- Thư mục

cuu duong than cong . com

cuu duong than cong . com



# Bộ nhớ ngoài & Tập tin

- Một số hạn chế của bộ nhớ trong
  - Không lưu trữ dữ liệu lâu dài
  - Không chứa lượng thông tin lớn.
- Cần các thiết bị lưu trữ ngoài(bộ nhớ ngoài) để lưu trữ dữ liệu
- Tuy nhiên, có nhiều loại thiết bị lưu trữ ngoài (đĩa từ, CD/DVD, USB, thẻ nhớ,...); đa dạng về cấu trúc, khả năng lưu trữ, phương thức truy xuất, tốc độ truy xuất
- HĐH cung cấp cái nhìn logic và đồng nhất về việc lưu trữ thông tin
  - Trừu tượng hóa thông tin vật lý thành đơn vị lưu trữ logic
  - tập tin





# Tập tin

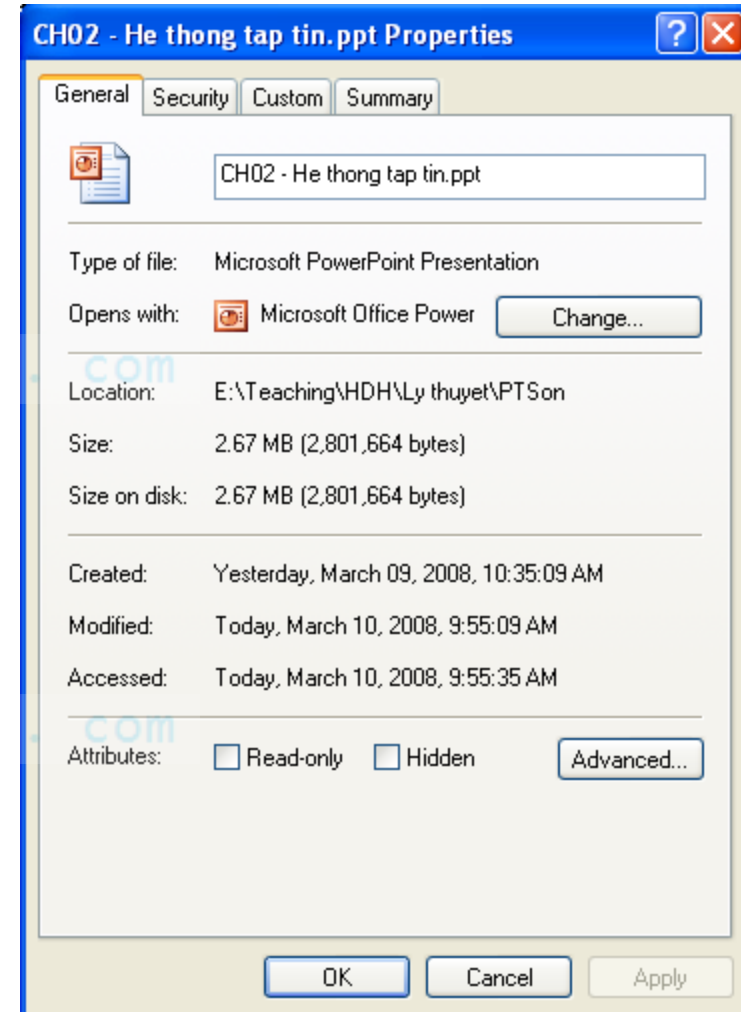
- Tập tin là gì ?
  - Lưu trữ tập hợp các thông tin có liên quan với nhau
  - Là một đơn vị lưu trữ luận lý che tổ chức vật lý của các thiết bị lưu trữ ngoài
  - Thường bao gồm 2 thành phần:
    - Thuộc tính
    - Nội dung
  - Mỗi hệ thống tập tin có cách thức tổ chức tập tin khác nhau



# Thuộc tính tập tin

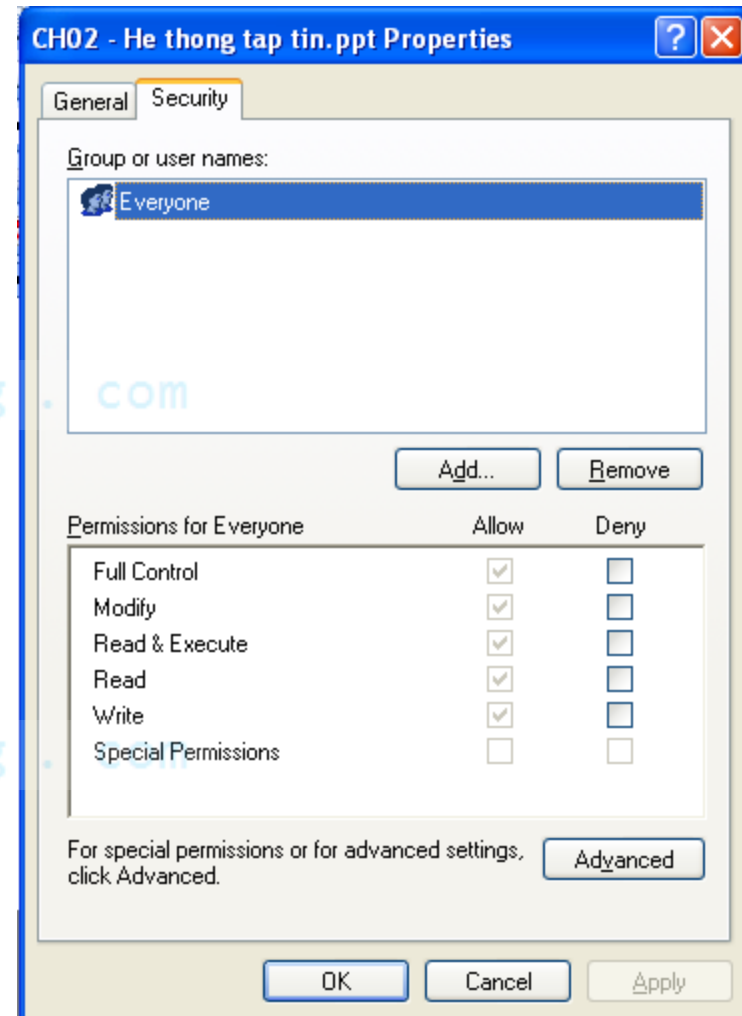
- Thuộc tính của tập tin trên các hệ thống tập tin khác nhau sẽ khác nhau, nhưng thường gồm các thuộc tính sau:

- Tên (tên + phần mở rộng)
- Người sở hữu
- Thuộc tính trạng thái: chỉ đọc, ẩn,...
- Kích thước
- Ngày giờ (tạo, truy cập, thay đổi)
- Thuộc tính bảo vệ
- Vị trí lưu trữ trên đĩa



# Cơ chế bảo vệ tập tin

- Người tạo /sở hữu tập tin có quyền kiểm soát:
  - Ai (người dùng /nhóm người dùng) có quyền gì trên tập tin
    - Đọc
    - Ghi
    - Thực thi
    - Thêm
    - Xóa
    - Liệt kê
    - Một số quyền đặc biệt khác



# Thao tác trên tập tin

- Một số thao tác cơ bản trên tập tin
  - *Tạo*
  - *Xóa*
  - *Đọc*
  - *Ghi*
  - *Định vị (seek)*
  - *Xóa nội dung (truncate)*
  - *Mở*
  - *Đóng*
- Một số thao tác khác: sao chép, di chuyển, đổi tên, ...



# Một số tính chất khác của tập tin

- Cấu trúc tập tin – do HĐH hay chương trình ứng dụng quyết định
  - Không cấu trúc
  - Có cấu trúc
- Loại tập tin
  - Tập tin văn bản (text file): chứa các dòng văn bản, cuối dòng có ký hiệu kết thúc dòng (end line)
  - Tập tin nhị phân (binary file): là tập tin có cấu trúc.
- Truy xuất tập tin
  - Tuần tự - Phải đọc từ đầu tập tin đến vị trí mong muốn, có thể quay lui (rewind)
  - Ngẫu nhiên - Có thể di chuyển (seek) đến đúng vị trí cần đọc





# Thư mục

- Thư mục là một loại tập tin đặc biệt, giúp tổ chức có hệ thống các tập tin trên hệ thống lưu trữ ngoài
  - Thuộc tính của thư mục tương tự của tập tin
  - Nội dung của thư mục: quản lý các tập tin, thư mục con của nó
    - Một cấp: đơn giản nhất, tất cả tập tin trên hệ thống cùng thư mục
    - Hai cấp: mỗi người dùng có 1 thư mục riêng
    - Cây phân cấp: được sử dụng phổ biến hiện nay
- Một số thao tác trên thư mục
  - Tạo
  - Xóa
  - Mở
  - Đóng
  - Liệt kê nội dung thư mục
  - Tìm kiếm tập tin
  - Duyệt hệ thống tập tin

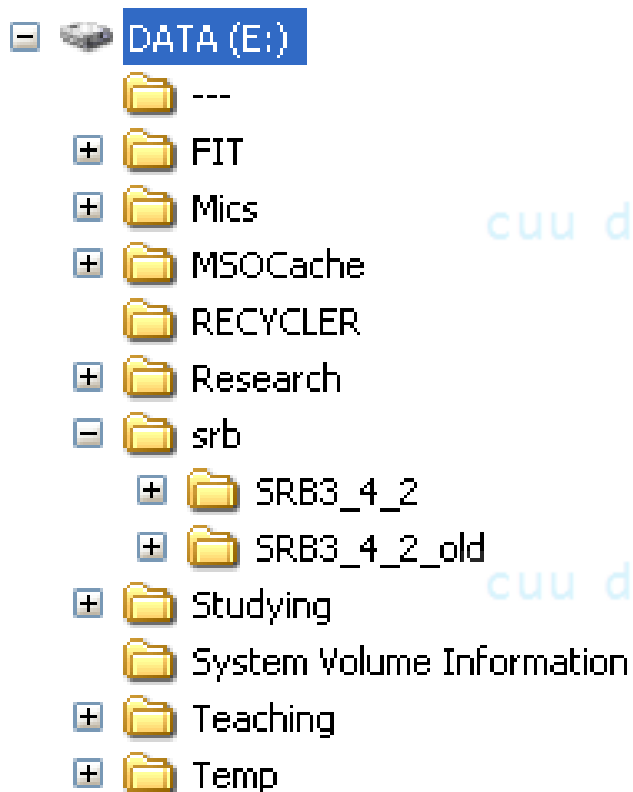


# Một số vấn đề tổ chức hệ thống tập tin

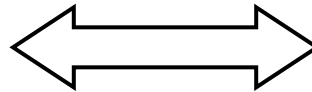
- Tổ chức thư mục
- Tổ chức tập tin
- Quản lý đĩa trống
- Tổ chức hệ thống tập tin trên đĩa từ
- Tổ chức hệ thống tập tin trong bộ nhớ
- Kết buộc hệ thống tập tin



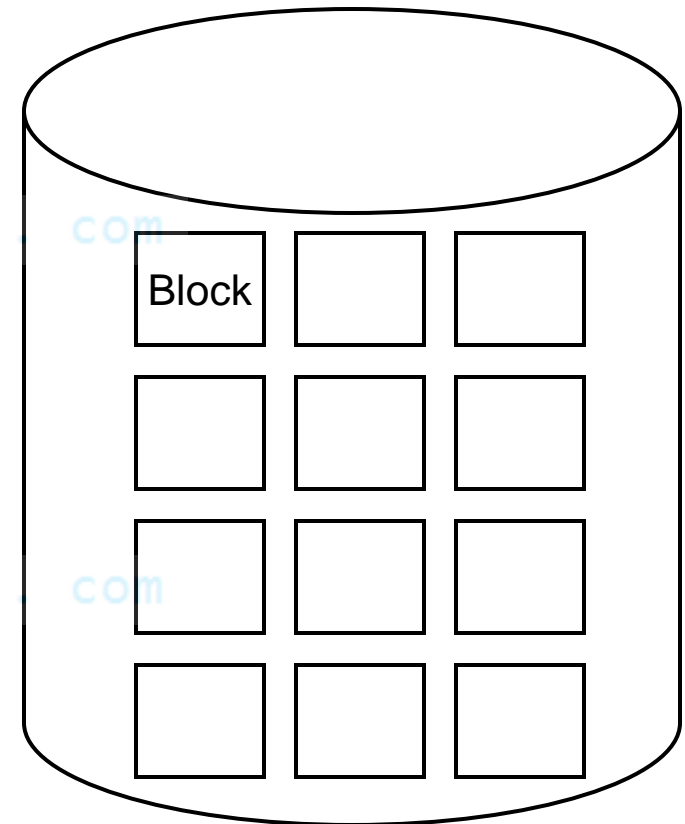
# Vấn đề



???



Thiết bị lưu trữ

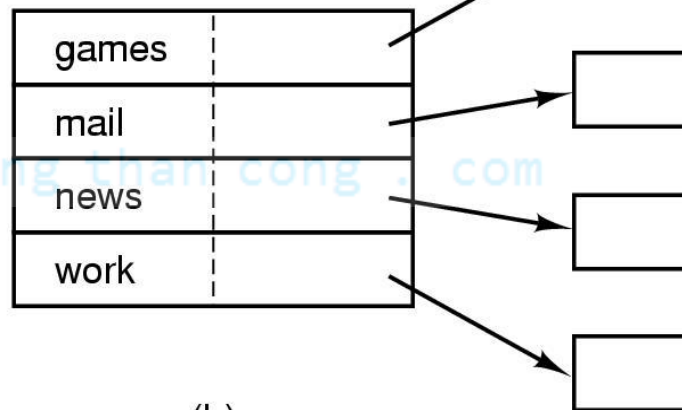


# Tổ chức thư mục

- Thường được tổ chức thành một bảng các phần tử (*directory entry*), gọi là bảng thư mục
- 2 cách tổ chức directory entry:
  - Entry chứa tên và các thuộc tính
  - Entry chứa tên và một con trỏ trỏ tới 1 cấu trúc chứa các thuộc tính

games	attributes
mail	attributes
news	attributes
work	attributes

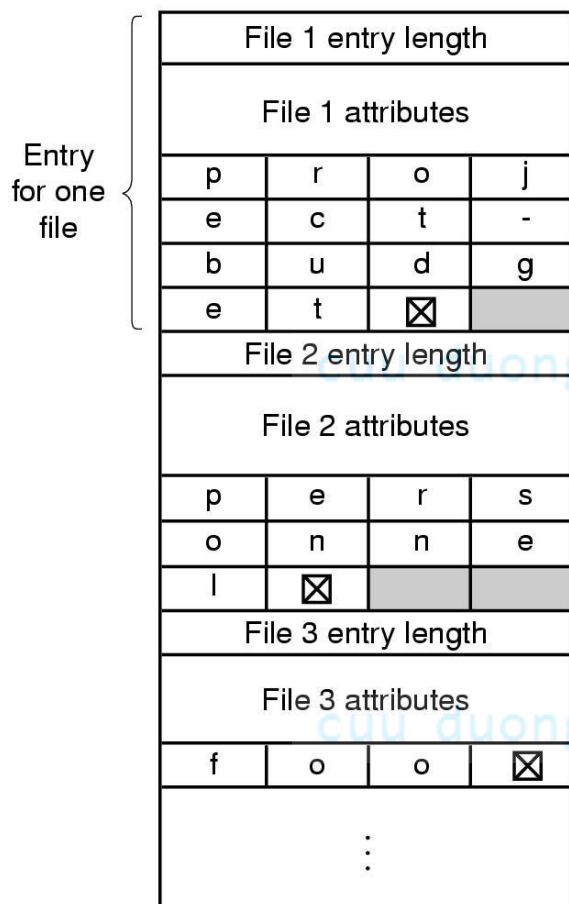
(a)



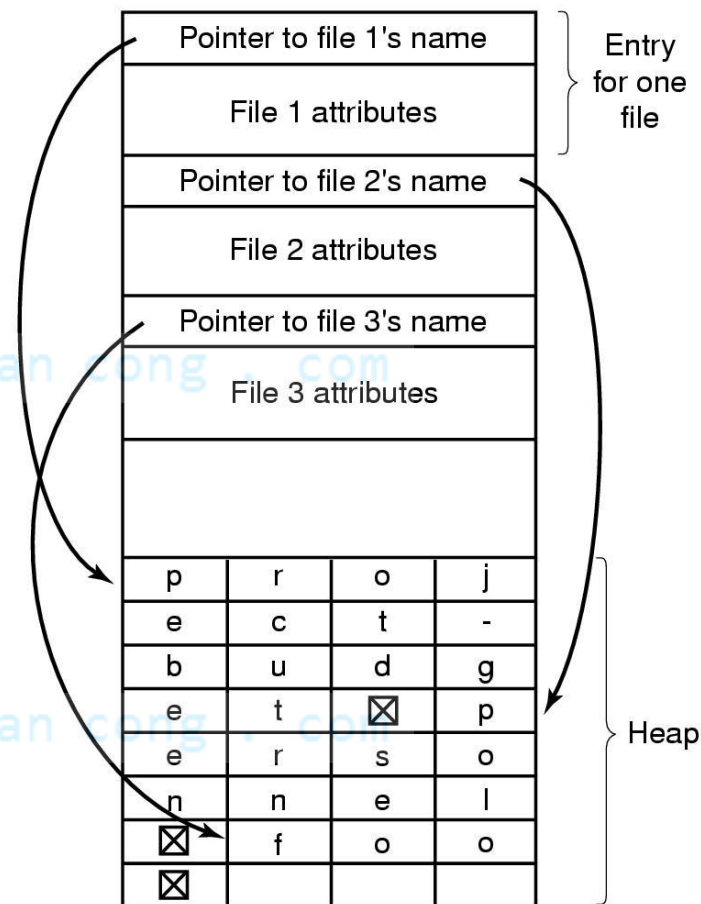
(b)

Data structure  
containing the  
attributes

# Vấn đề tên dài (long file name - LFN)



(a)



(b)

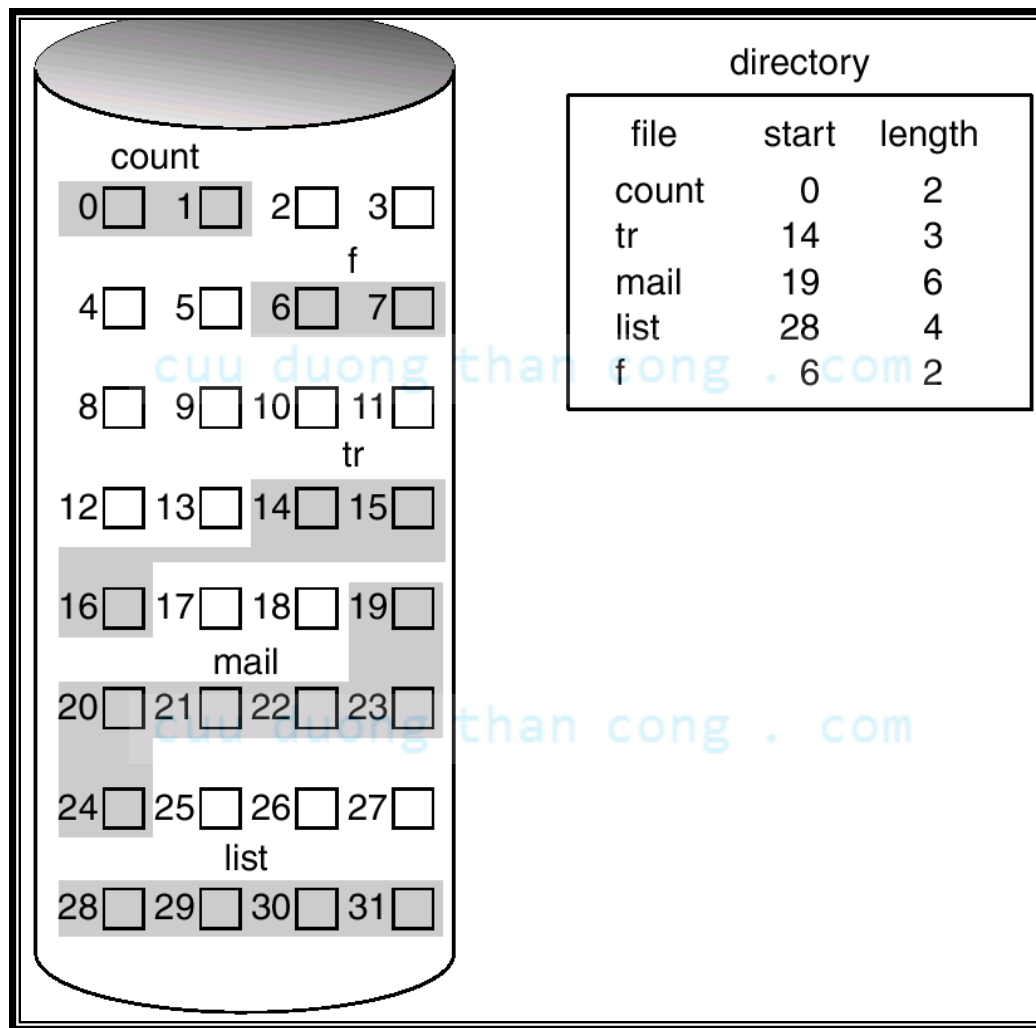


# Tổ chức tập tin

- Mỗi tập tin lưu nội dung trên một số block (khối lưu trữ) của thiết bị lưu trữ  
→ Làm sao biết được tập tin đang chiếm những block nào ?  
cuu duong than cong . com
- Phương pháp cấp phát mô tả cách thức cấp phát các block cho các tập tin
- Có 3 phương pháp cấp phát chính:
  - Cấp phát liên tục  
cuu duong than cong . com
  - Cấp phát theo kiểu danh sách liên kết
  - Cấp phát theo kiểu chỉ mục



# Cấp phát liên tục



# Cấp phát liên tục (tt)

- Mỗi tập tin chiếm các block liên tục trên đĩa
- Đơn giản, chỉ cần quản lý vị trí (chỉ số) block bắt đầu và chiều dài (số block)
- Hỗ trợ truy xuất tuần tự & truy xuất trực tiếp
- Vấn đề External fragmentation
- Vấn đề khi kích thước tập tin tăng

cuu duong than cong . com

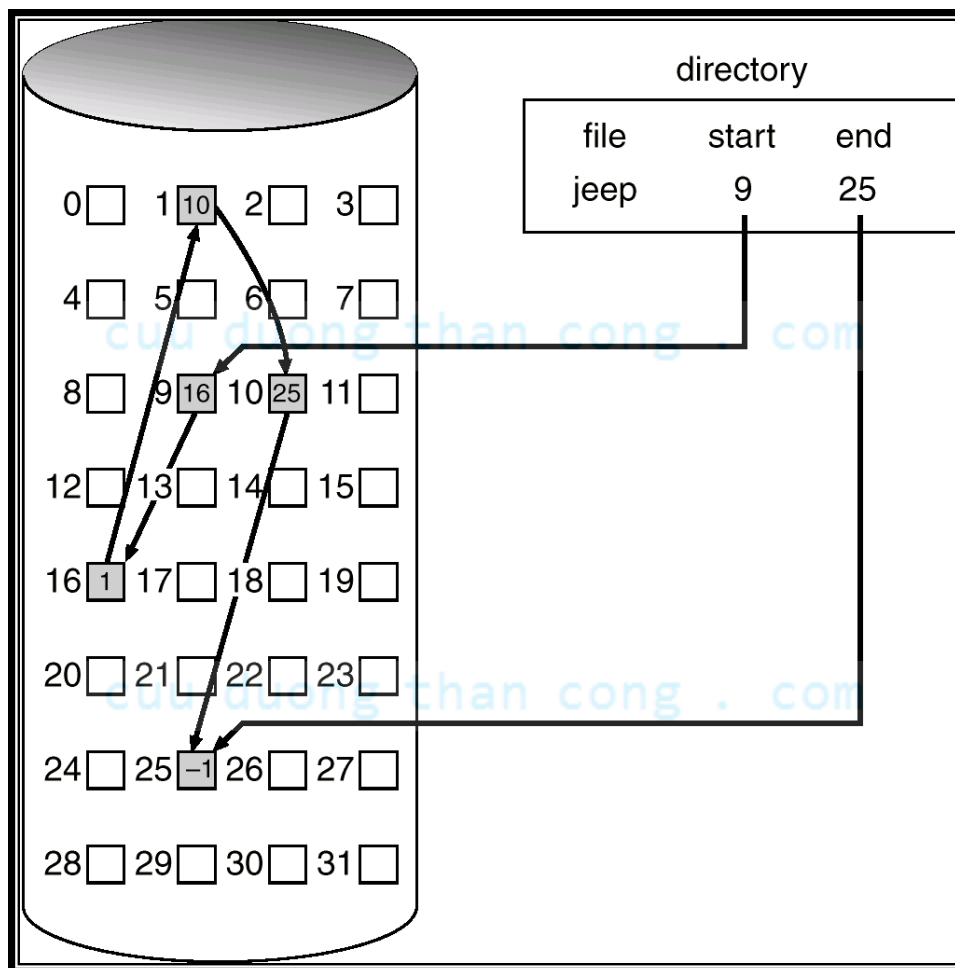


# Cấp phát liên tục (tt)

- Hệ thống tập tin cấp phát theo *extent*:
  - Extent là một tập các block liên tục
  - Cấp phát cho tập tin theo từng extent
  - Một tập tin có thể chiếm một hoặc nhiều extent không liên tục nhau
  - Kích thước các extent có thể khác nhau
  - Cần quản lý 3 thông tin: vị trí block bắt đầu, số block và một con trỏ trỏ tới block đầu tiên của extent kế tiếp
  - Vấn đề Internal fragmentation và External fragmentation



# Cấp phát theo kiểu danh sách liên kết



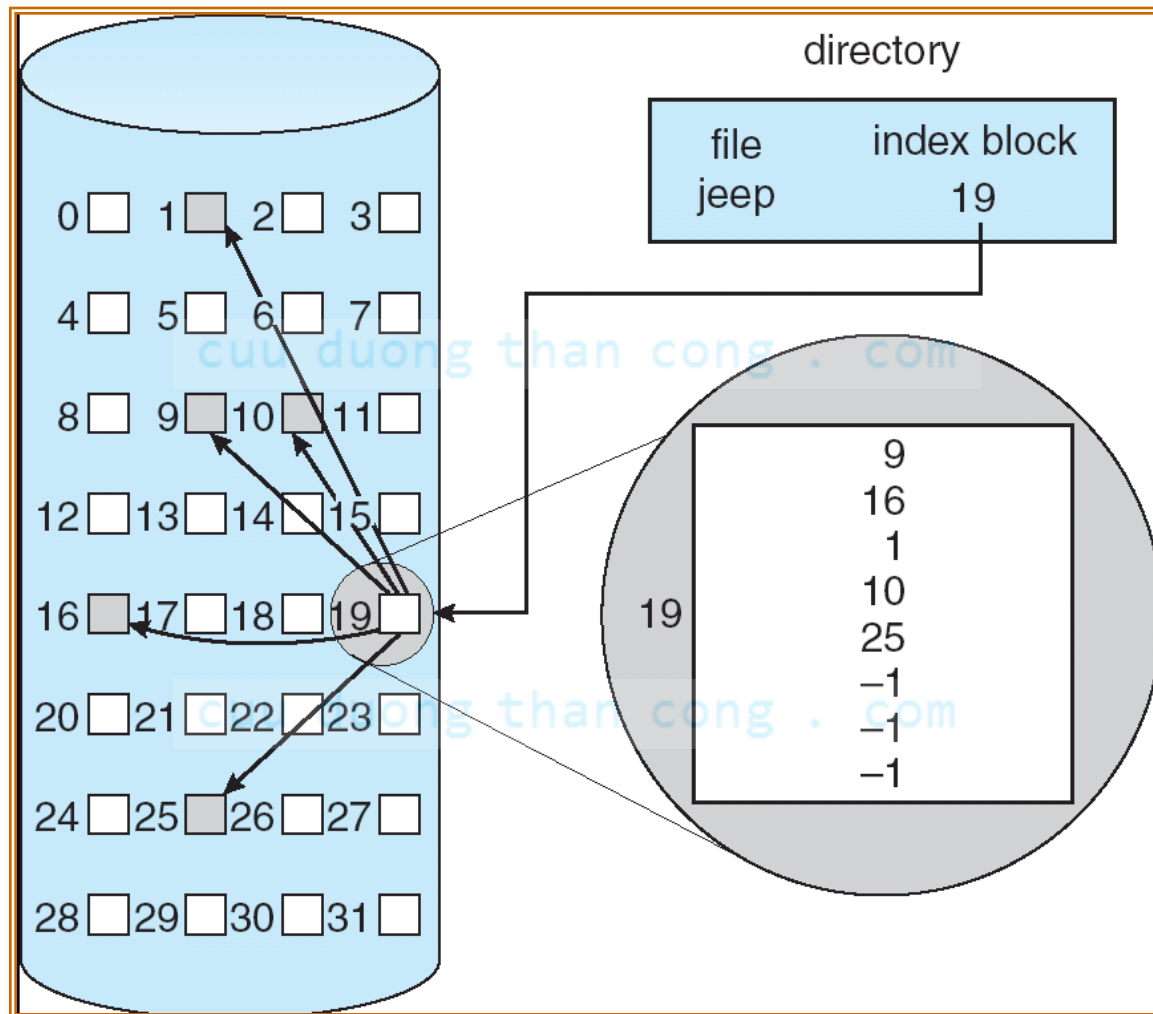


# Cấp phát theo kiểu danh sách liên kết (tt)

- Mỗi tập tin chiếm một tập các block theo kiểu danh sách liên kết.
- Mỗi block sẽ chứa thông tin về địa chỉ của block kế tiếp
- Các block có thể nằm rải rác trên đĩa
- Chỉ hỗ trợ truy xuất tuần tự
- Đơn giản, chỉ cần quản lý vị trí (chỉ số) block bắt đầu
- Không bị External fragmentation
- Tốn chi phí lưu địa chỉ block kế tiếp



# Cấp phát theo kiểu chỉ mục

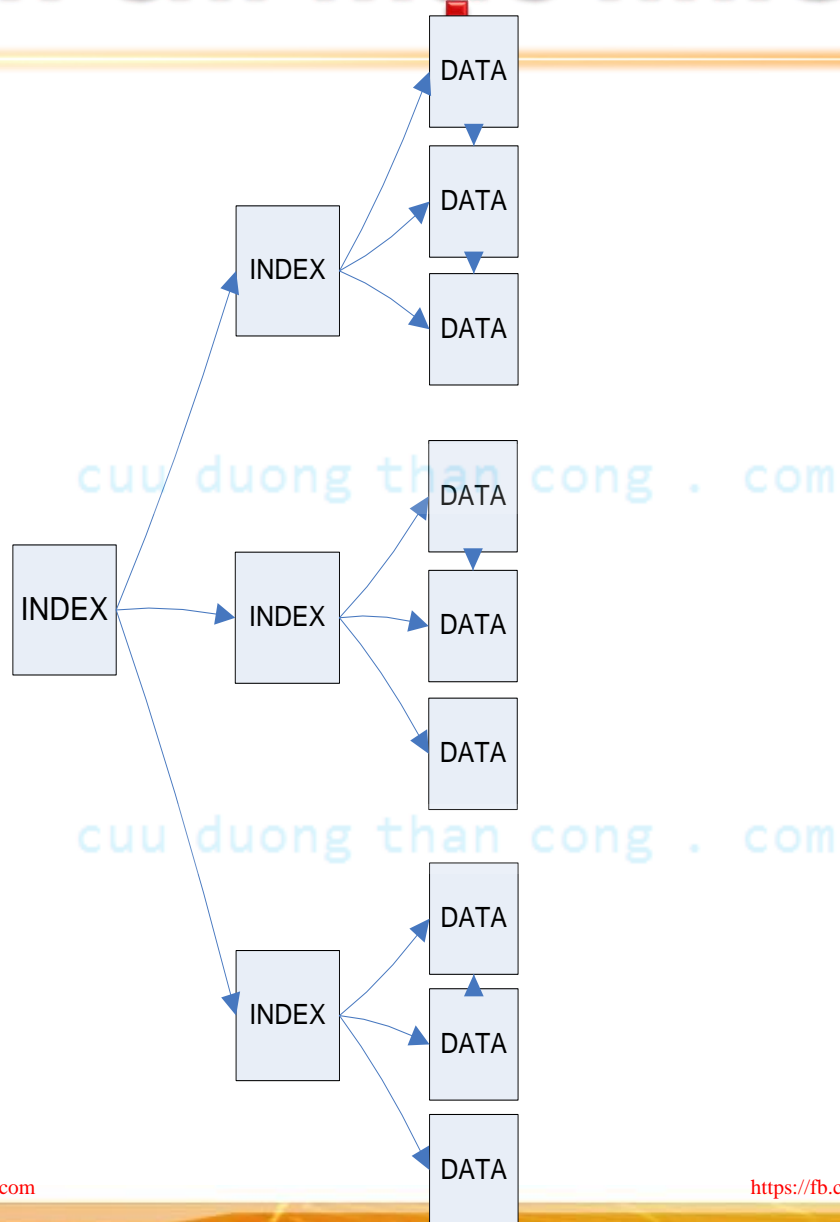


# Cấp phát theo kiểu chỉ mục (tt)

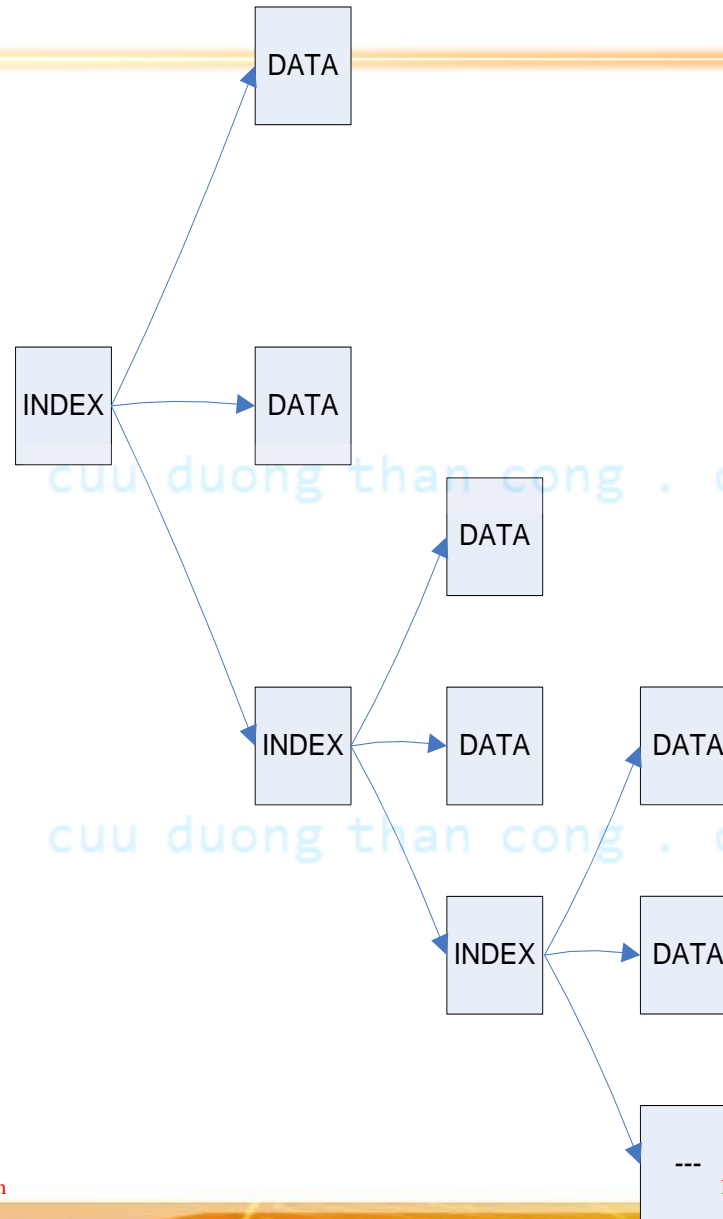
- Gồm một hoặc nhiều block làm bảng chỉ mục chứa địa chỉ của các block dữ liệu
- Hỗ trợ truy xuất tuần tự & truy xuất trực tiếp
- Tốn không gian đĩa để lưu các block chỉ mục
- Không bị External fragmentation
- Một số mô hình mở rộng
  - Mô hình chỉ mục nhiều cấp
  - Mô hình chỉ mục kết hợp danh sách liên kết
  - Mô hình chỉ mục nhiều cấp kết hợp danh sách liên kết



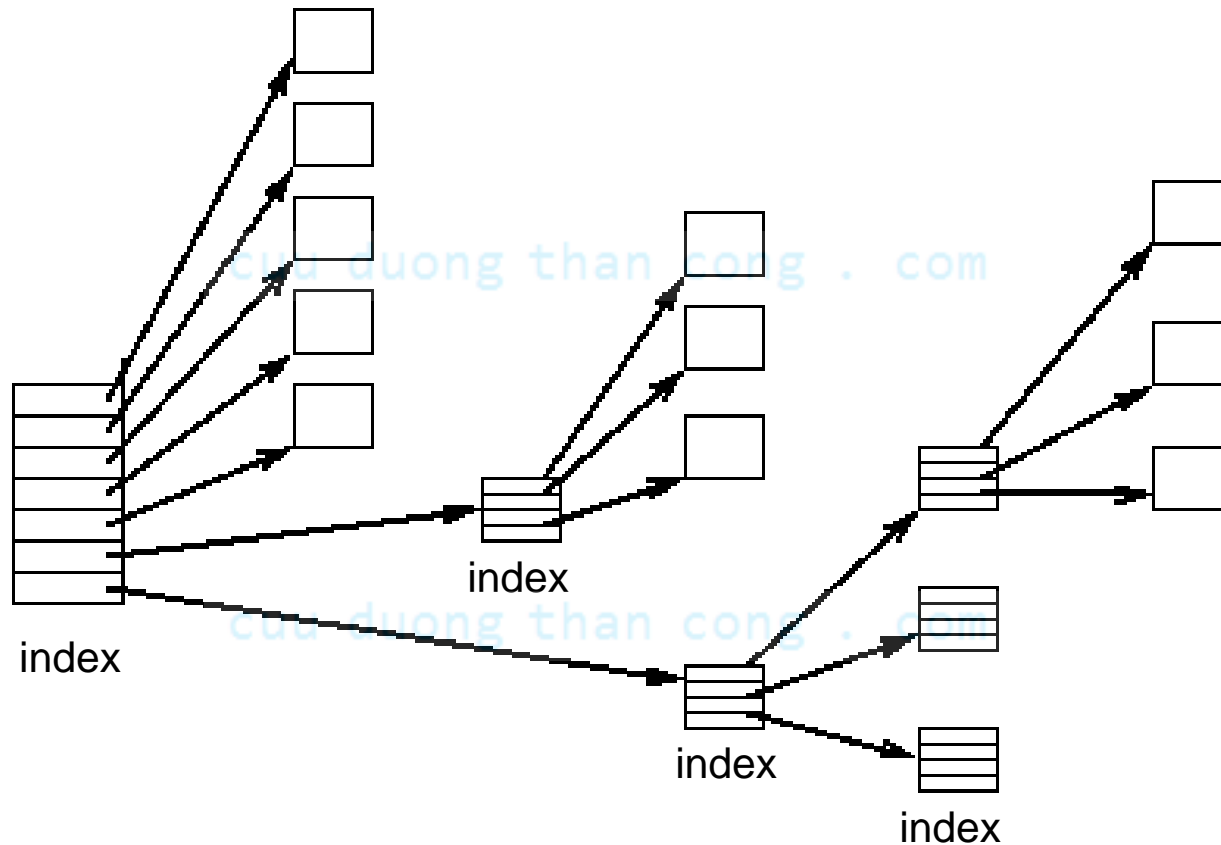
# Mô hình chỉ mục nhiều cấp



# Mô hình chỉ mục kết hợp danh sách liên kết

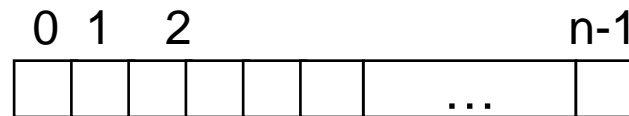


# Mô hình chỉ mục nhiều cấp kết hợp danh sách liên kết



# Quản lý không gian đĩa trống

- Bit vector (Bit map)
  - Mỗi block được biểu diễn bằng 1 bit



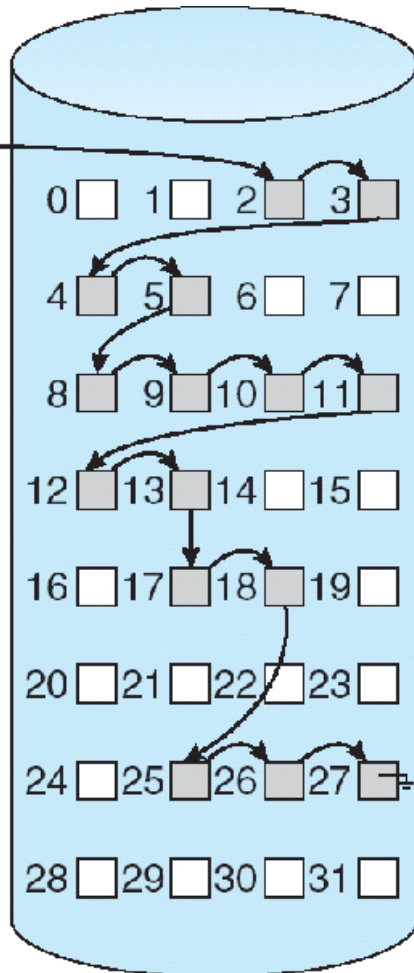
$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ trống} \\ 1 \Rightarrow \text{block}[i] \text{ đã dùng} \end{cases}$$

- Bit vector tốn không gian đĩa. Ví dụ:
  - kích thước 1 block =  $2^{12}$  bytes
  - kích thước đĩa =  $2^{30}$  bytes (1 gigabyte)
  - $n = 2^{30} / 2^{12} = 2^{18}$  bits (or 32K bytes)
- HĐH Macintosh

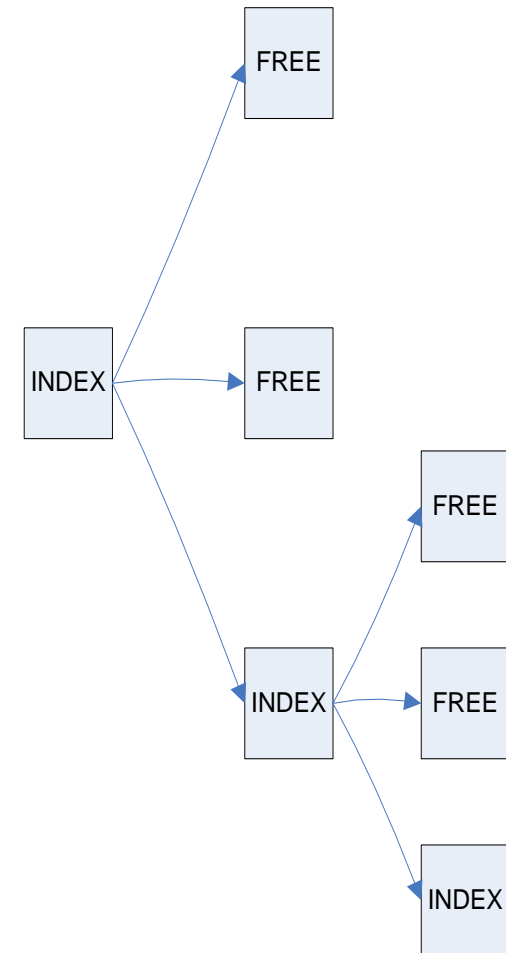




# Quản lý không gian đĩa trống (tt)

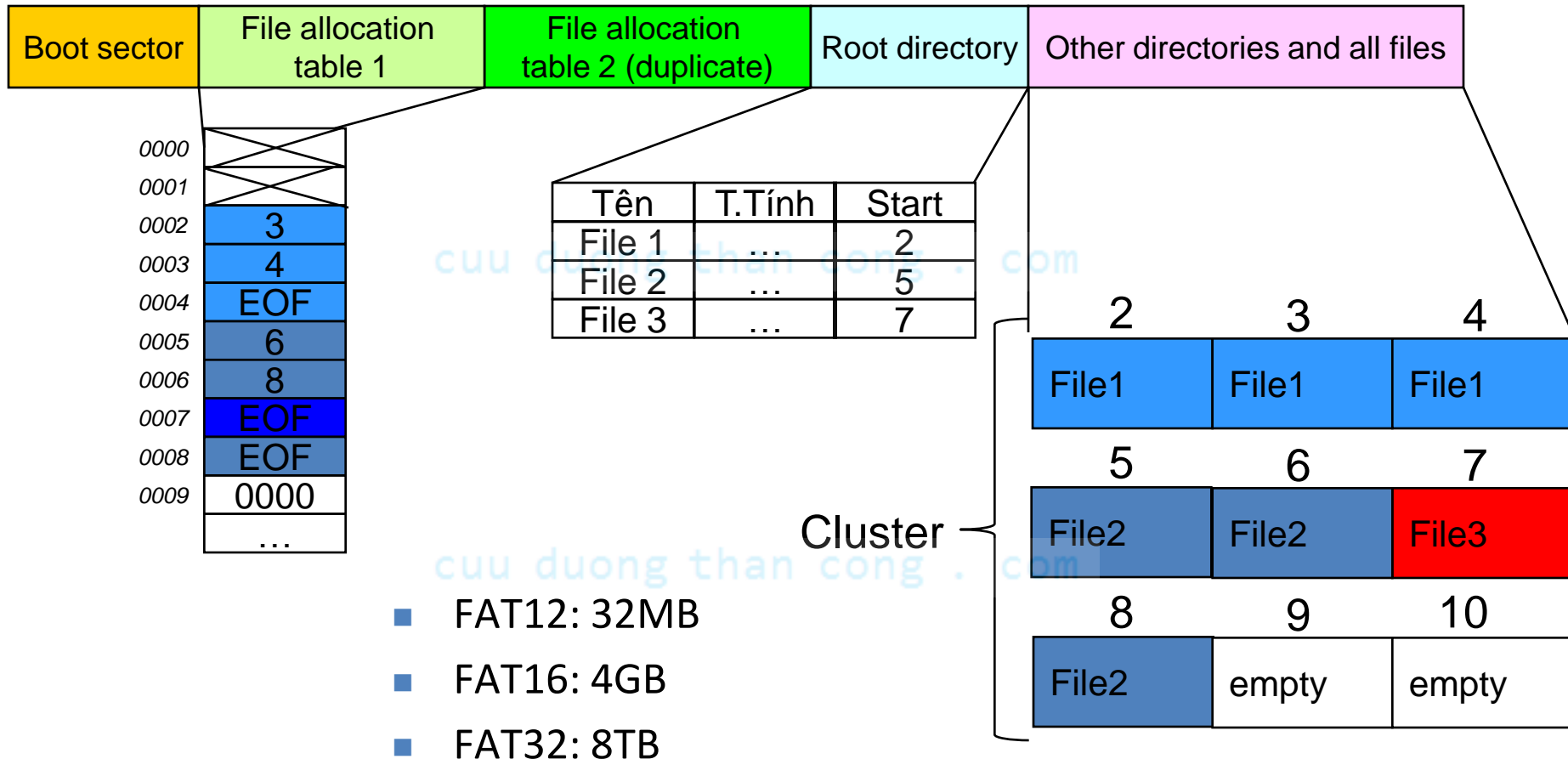


- Danh sách liên kết
  - Chi phí duyệt danh sách cao
  - Không tốn không gian đĩa
- Grouping
  - Chứa danh sách các block trống
  - Dễ tìm một lượng lớn các block trống
- Counting
  - Chứa địa chỉ block trống đầu tiên và số lượng các block trống liên tục tiếp theo



# Hệ thống tập tin FAT (12, 16, 32)

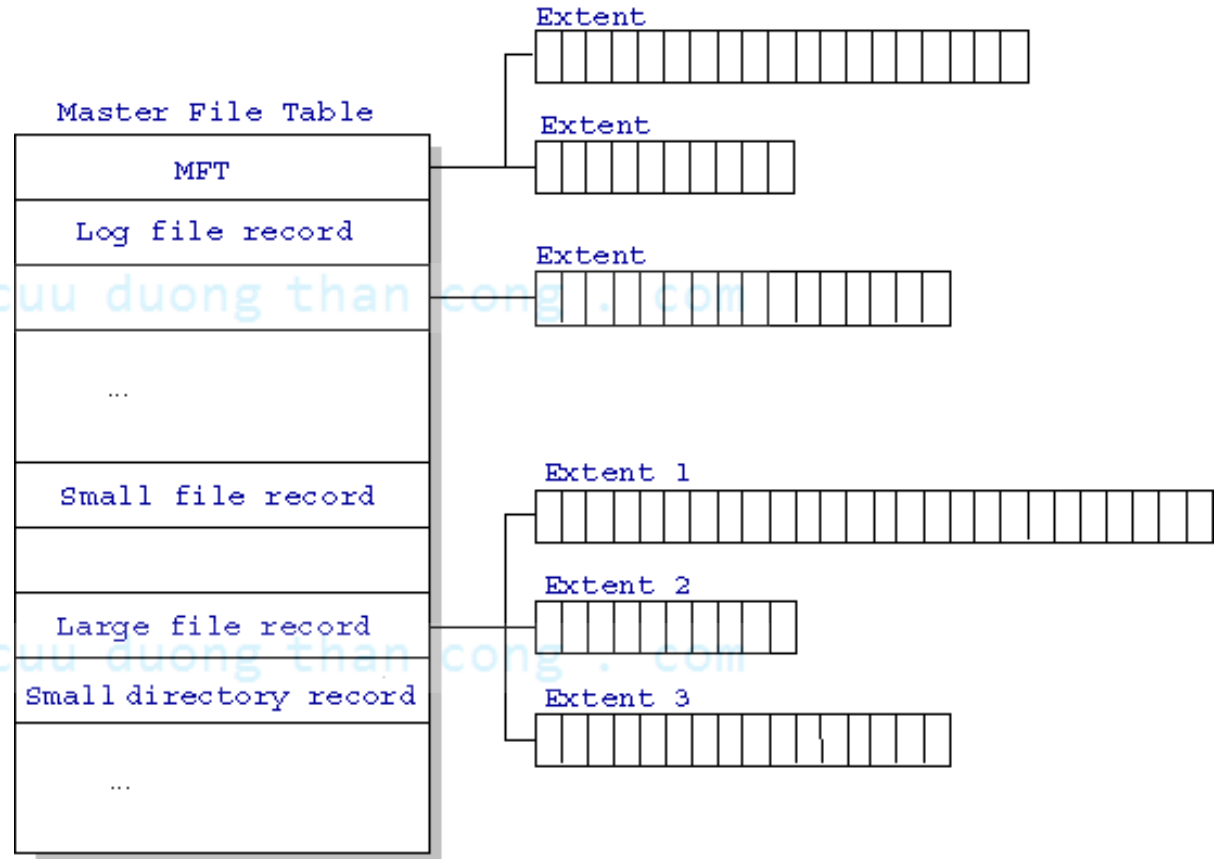
## File Allocation Table



# NTFS (New Technology File System)



16 exabytes  
(16 billion GB)



Standard  
information

Filename

Security  
descriptor

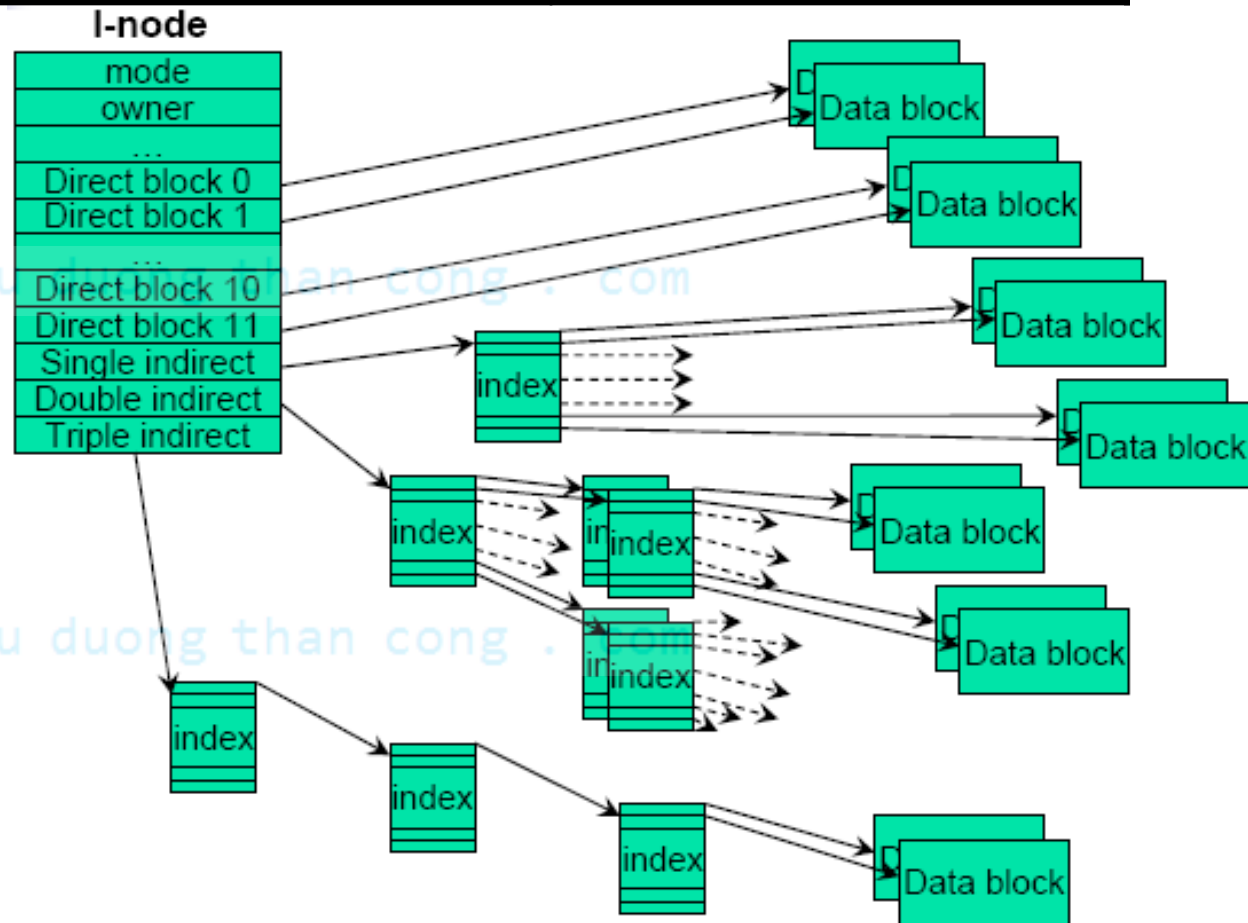
Data

# Hệ thống tập tin trên Unix /Linux

## Cấu trúc I-node

boot block	super block	I-node	files and directories
------------	-------------	--------	-----------------------

- Gián tiếp cấp 1:  
cấp này trở tới 256  
địa chỉ. Tổng 256KB
- Gián tiếp cấp 2:  
 $256 \times 256 = 65 \text{ MB}$
- Gián tiếp cấp 3:  
 $256 \times 256 \times 256 = 16 \text{ GB}$



# Hệ thống tập tin trên UNIX V7

Root directory

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

Looking up  
usr yields  
i-node 6

I-node 6  
is for /usr

Mode size times
132

I-node 6  
says that  
/usr is in  
block 132

Block 132  
is /usr  
directory

6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

/usr/ast  
is i-node  
26

I-node 26  
is for  
/usr/ast

Mode size times
406

I-node 26  
says that  
/usr/ast is in  
block 406

Block 406  
is /usr/ast  
directory

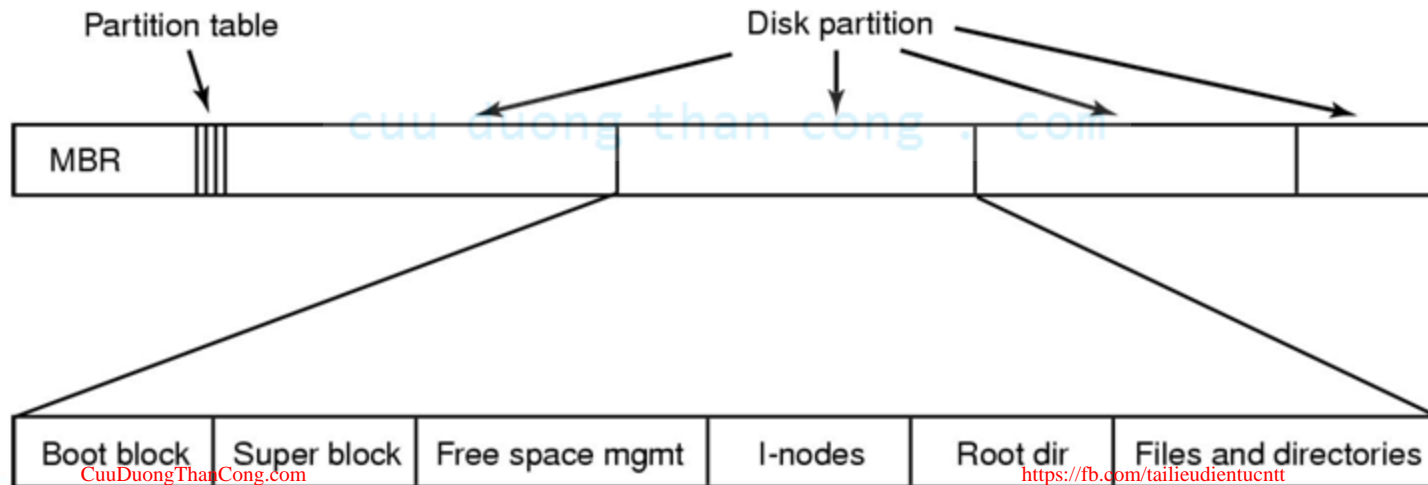
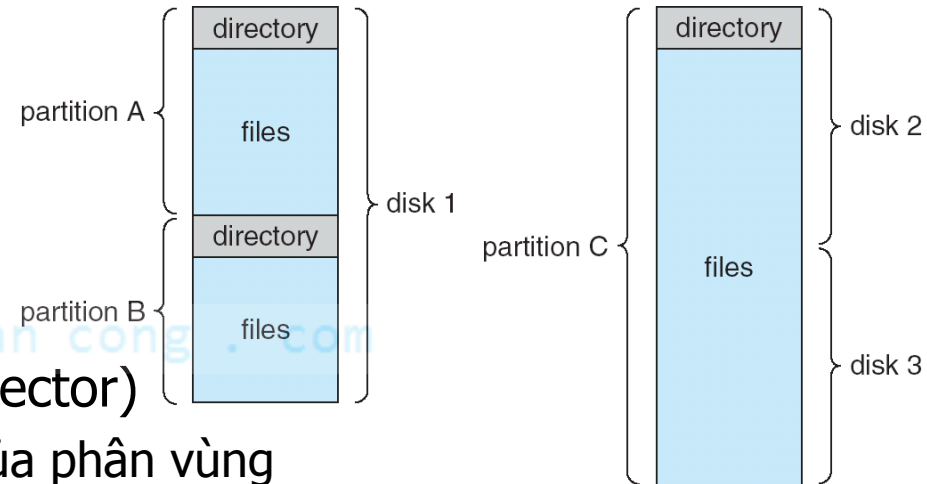
26	.
6	..
64	grants
92	books
60	mbox
81	minix
17	src

/usr/ast/mbox  
is i-node  
60



# Tổ chức hệ thống tập tin trên đĩa từ

- Master Boot Record (MBR): thường nằm tại sector logic 0, kích thước 512 bytes
- Phân vùng (Partition):
  - Primary
  - ExtendedTối đa 4 phân vùng
- Boot block + Super block (Boot sector)
  - Chứa các thông số quan trọng của phân vùng
  - Chứa một đoạn chương trình nhỏ để nạp HĐH khi khởi động máy





# Master Boot Record

Master Boot Record, Base Offset: 0

Offset	Title	Value
0	Master boot	B8 00 00 8E D0 BC 00 7C 8E D8 FC B9 00 01 8B F4 B
1B8	Windows di	8C73F4D0
1B8	Same revers	D0F4738C

- Đoạn chương trình để giúp khởi động hệ thống
- Bảng mô tả thông tin các phân vùng logic
  - TYPE-ID = 0x07 : Windows
  - TYPE-ID = 0x83 : Linux
  - TYPE-ID = 0x00 : Không sử dụng.
- Thông tin nhân diện MBR

Partition Table Entry #1

1B8	80 = active	00
1BF	Start head	1
1C0	Start sector	1
1C0	Start cylinde	0
1C2	Partition typ	DE
1C3	End head	254
1C4	End sector	63
1C4	End cylinder	5
1C6	Sectors pre	63
1CA	Sectors in p	96327

Partition Table Entry #2

1CE	80 = active	80
1CF	Start head	0
1D0	Start sector	1
1D0	Start cylinde	6

Address		Description		Size in bytes
Hex	Dec			
0000	0	Code Area		max. 446
01B8	440	Optional Disk signature		4
01BC	444	Usually Nulls; 0x0000		2
01BE	446	Table of primary partitions (Four 16-byte entries, IBM Partition Table scheme)		64
01FE	510	55h	MBR signature; 0xAA55 <sup>[1]</sup>	2
01FF	511	AAh		
MBR, total size: 446 + 64 + 2 =				512



# Master Boot Record

Address			Description	Size in bytes
<u>Hex</u>	<u>Oct</u>	<u>Dec</u>		
0000	0000	0	Code Area	440 (max. 446)
01B8	0670	440	Optional Disk signature	4
01BC	0674	444	Usually Nulls; 0x0000	2
01BE	0676	446	Table of primary partitions (Four 16-byte entries, IBM Partition Table scheme)	64
01FE	0776	510	55h	MBR signature; 0xAA55 <sup>[1]</sup>
01FF	0777	511	AAh	
MBR, total size: 446 + 64 + 2 =				512

Nguồn: wikipedia



# Bảng mô tả các phân vùng - 1

Offset	Field length (bytes)	Description
0x00	1	status <sup>[7]</sup> (0x80 = bootable, 0x00 = non-bootable, other = invalid <sup>[8]</sup> )
0x01	3	<a href="#">CHS address</a> of first <a href="#">block</a> in partition. <sup>[9]</sup> The format is described in the next 3 bytes.
0x01	1	head <sup>[10]</sup>
0x02	1	sector is in bits 5–0 <sup>[11]</sup> ; bits 9–8 of cylinder are in bits 7–6
0x03	1	bits 7–0 of cylinder <sup>[12]</sup>
0x04	1	<a href="#">partition type</a> <sup>[13]</sup>
0x05	3	<a href="#">CHS address</a> of last <a href="#">block</a> in partition. <sup>[14]</sup> The format is described in the next 3 bytes.
0x05	1	head
0x06	1	sector is in bits 5–0; bits 9–8 of cylinder are in bits 7–6
0x07	1	bits 7–0 of cylinder
0x08	4	<a href="#">LBA</a> of first sector in the partition
0x0C	4	number of <a href="#">blocks</a> in partition, in little-endian format

44



# Bảng mô tả các phân vùng - 2

- Type:
  - 0x07 : Phân vùng chứa "Windows"
  - 0x83 : Phân vùng chứa "Linux"
  - 0x00 : Phân vùng không sử dụng.

cuu duong than cong . com

*Tham khảo thêm:*

*[http://www.win.tue.nl/~aeb/partitions/partition\\_types-1.html](http://www.win.tue.nl/~aeb/partitions/partition_types-1.html)*

cuu duong than cong . com



# Master Boot Record – Ví dụ -

1

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000000000	90	E9	7D	01	FA	33	C0	8E	D0	8E	C0	8E	D8	BC	00	7C	é}.ú3A D A 0%.
0000000010	8B	F4	FB	BF	00	06	B9	00	01	F3	A5	BB	20	06	FF	E3	ôúú...¹...ó#» .ýã
0000000020	90	90	BE	7D	07	81	3C	AA	55	75	11	E8	58	00	73	0C	%}. <²Uu.èX.s.
0000000030	E8	65	00	72	07	E8	B1	00	72	3B	EB	2C	BE	7D	07	C7	èe.r.è±.r;è,%.Ç
0000000040	04	00	00	BA	80	00	BE	BE	07	B9	04	00	F6	04	80	75	...²  .%%.¹...ö.  u
0000000050	07	83	C6	10	E2	F6	EB	1D	8A	74	01	8B	4C	02	BB	00	.. Æ.âöè.. t.. L>..
0000000060	7C	B8	01	02	CD	13	72	0D	81	3E	FE	7D	55	AA	75	05	,...í.r. >þ}U³u.
0000000070	EA	00	7C	00	00	BE	6A	07	AC	0A	C0	74	FE	BB	07	00	è. ...%j.~.Àtp>..
0000000080	B4	0E	CD	10	EB	F2	BB	00	7E	C6	07	13	C6	47	01	00	'.í.èö>..~Æ..ÆG..
0000000090	B2	80	B8	00	E0	CD	13	C3	BF	00	7E	BA	F0	01	B3	A0	² ...àí.Ãö.~²ð.³
00000000A0	E8	84	00	72	0C	B1	01	E8	48	00	72	05	E8	19	00	73	è r.±.èH.r.è..s
00000000B0	16	F6	C3	10	75	05	80	CB	10	EB	E5	81	FA	70	01	74	..öÃ.u.. È.èâ úp.t
00000000C0	05	BA	70	01	EB	D8	F9	C3	81	BD	FE	01	55	AA	75	17	..²p.èöuÃ xp.U³u.
00000000D0	8B	75	02	81	FE	BE	01	77	0E	03	F7	81	3C	AA	55	75	u.. þ%.w..÷ <²Uu
00000000E0	06	F6	44	02	01	75	01	F9	C3	BF	00	7C	B1	0A	E8	01	..öD..u..ùÃö.. ±.è.
00000000F0	00	C3	52	57	83	C2	02	B0	01	EE	42	8A	C1	EE	42	32	..ÃRW Ã..*..íB ÁíB2
0000000100	C0	EE	42	EE	42	8A	C3	EE	42	B0	20	EE	E8	33	00	EC	ÁíBíB ÃíB*..íè3.i
0000000110	24	FD	3C	58	75	0D	83	EA	07	B9	00	01	FA	F3	6D	FB	\$ý<Xu.. é.¹...úómú
0000000120	F8	EB	01	F9	5F	5A	C3	52	83	C2	07	EC	A8	80	75	0F	øè.ù_ZÃR Ã.í'' u.
0000000130	4A	8A	C3	EE	42	EC	24	D0	3C	50	75	03	F8	EB	01	F9	J ÃíBí\$ð<Pu.øè.ù
0000000140	5A	C3	51	8B	0E	6C	04	83	C1	12	81	C2	FF	01	EC	8A	ZÃQ ..l.. Ã.  Ãý.i
0000000150	E0	80	E4	D8	80	FC	58	74	06	3B	0E	6C	04	75	EF	81	à ä0 üXt...l.uí
0000000160	EA	FF	01	B9	00	20	E2	FE	59	C3	0D	0A	45	72	72	6F	éý.¹..ápYÃ..Erro
0000000170	72	20	4C	6F	61	64	69	6E	67	20	4F	53	00	55	AA	00	r Loading OS.U³.
0000000180	00	E9	80	FE	00	00	00	00	68	00	00	00	00	00	00	00	..é þ....h.....
0000000190	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000001B0	00	00	00	00	00	00	00	00	00	0A	01	0A	00	00	80	01	..... .
00000001C0	01	00	0C	FE	FF	FF	3F	00	00	00	C1	52	A8	04	00	00	...þýý?...ÁR''...
00000001D0	C1	FF	0F	FE	FF	FF	00	53	A8	04	C1	91	A8	04	00	00	Áý.þýý.S''..Á''...
00000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA	.....U³



# Master Boot Record – Ví dụ - 2

Partit ion	Starting			Ending			Status	Type	First sector	Sector#
	H	T	S	H	T	S				
1	1	0	1	254	1023	63	0x80	0x0C	63	78140097
2	0	1023	1	254	1023	63	0x00	0x0F	78140160	78156225

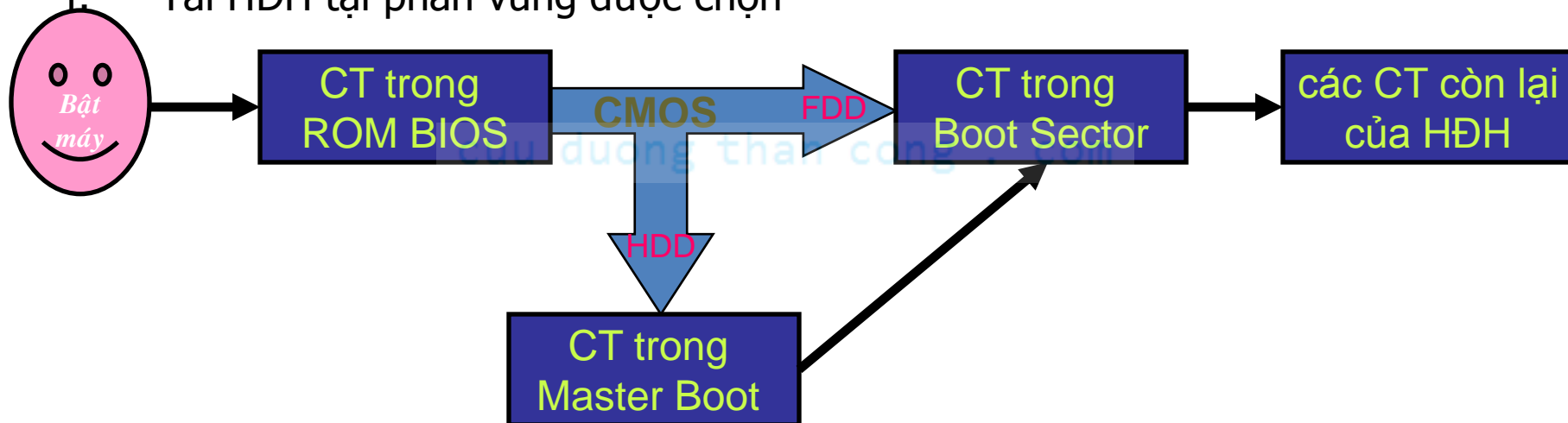
cuu duong than cong . com

cuu duong than cong . com



# Quá trình khởi động hệ thống từ đĩa cứng

1. POST (Power-On Self-Test)
2. Tải MBR để đọc thông tin bảng phân vùng.  
Tìm phân vùng "active".  
Nếu không tìm thấy phân vùng "active", MBR có thể tải một boot loader và chuyển điều khiển cho nó. Boot loader này sẽ cho phép chọn HĐH trên một phân vùng
3. Chuyển quyền điều khiển về cho đoạn mã chương trình nằm trong Boot Sector của phân vùng được chọn
4. Tải HĐH tại phân vùng được chọn





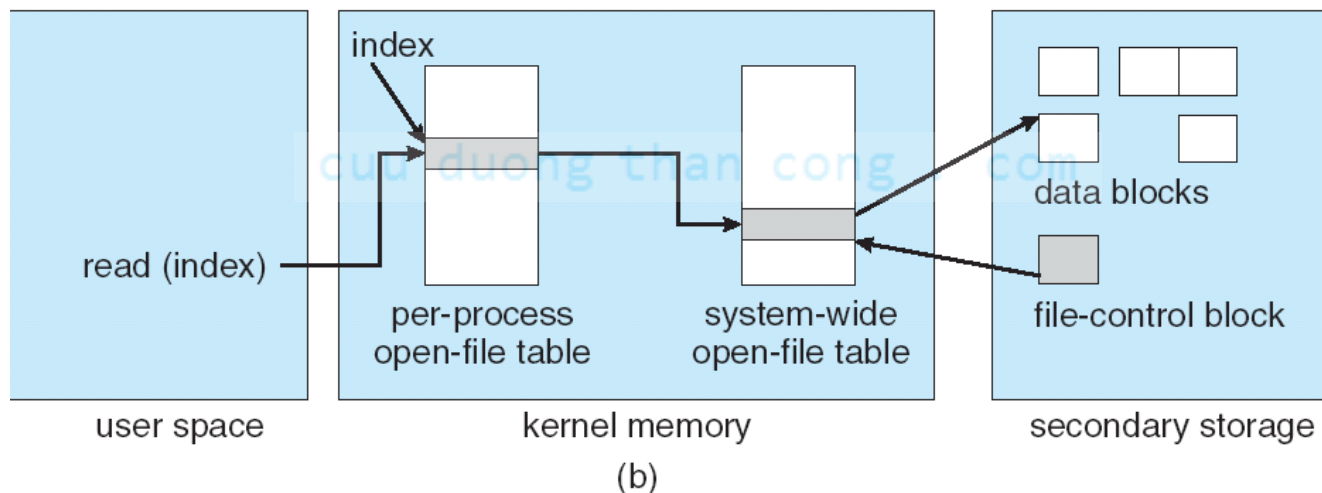
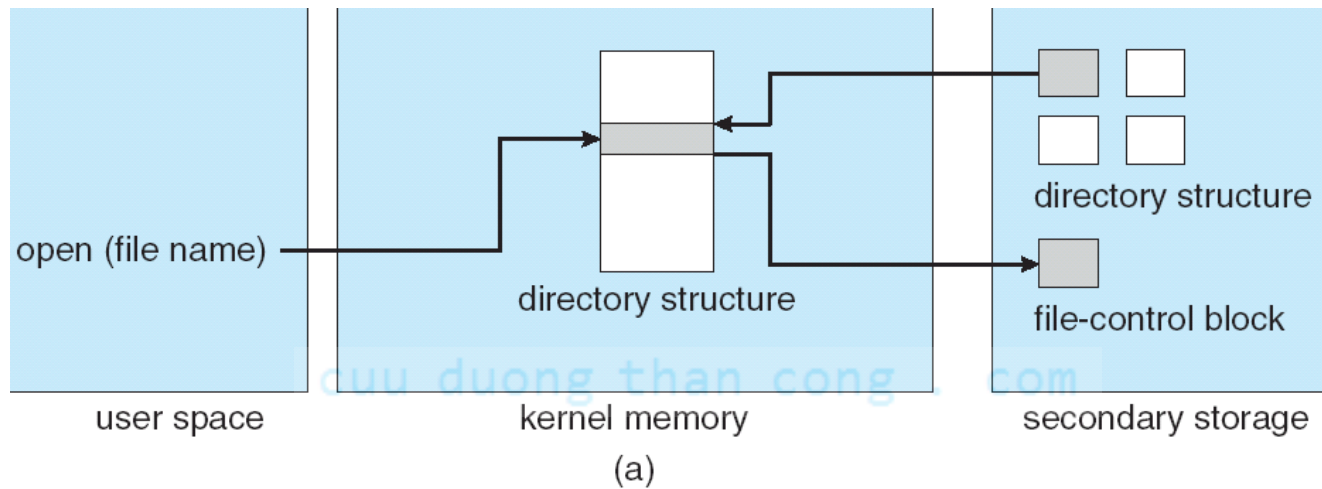
# Tổ chức hệ thống tập tin trong bộ nhớ chính

- Vấn đề:
  - Thao tác với nhiều tập tin tại một thời điểm ?
  - Thao tác trên cùng một tập tin tại một thời điểm ?
- Các thông tin cần lưu trữ trong bộ nhớ:
  - Mounted Volume Table – Danh sách các volume được sử dụng trên hệ thống
  - Directory Structure – Thông tin các thư mục mới được sử dụng
    - Con trỏ trỏ tới volume tương ứng
  - System-wide open-file Table – Danh sách các tập tin đang được mở trên hệ thống
    - Con trỏ tập tin, định vị tập tin trên đĩa
    - Quyền truy cập
    - Biến đếm tập tin đang mở
  - Per-process open-file Table – Danh sách các tập tin mà tiến trình đang thao tác
    - Con trỏ trỏ tới tập tin đang mở tương ứng trong system-wide open-file table





# Tổ chức hệ thống tập tin trong bộ nhớ chính



# Kết buộc (Mount) hệ thống tập tin

- Một hệ thống tập tin phải được kết buộc (mount) trước khi có thể truy xuất (giống như tập tin phải được mở trước khi sử dụng)
- Các HĐH thường phát hiện và tự động kết buộc các hệ thống tập tin tồn tại trên hệ thống
  - Windows kết buộc hệ thống tập tin vào ổ đĩa
  - Linux kết buộc hệ thống tập tin vào một thư mục
- Một số HĐH cung cấp lệnh để thực hiện việc kết buộc hệ thống tập tin
  - Ví dụ: lệnh *mount* (Linux)



# Đĩa từ - thuật toán đọc đĩa

- First-Come-First-Serve (FCFS)
- Shortest Seek Time First (SSTF)
- SCAN, C-SCAN
- Look, C-Look

cuu duong than cong . com

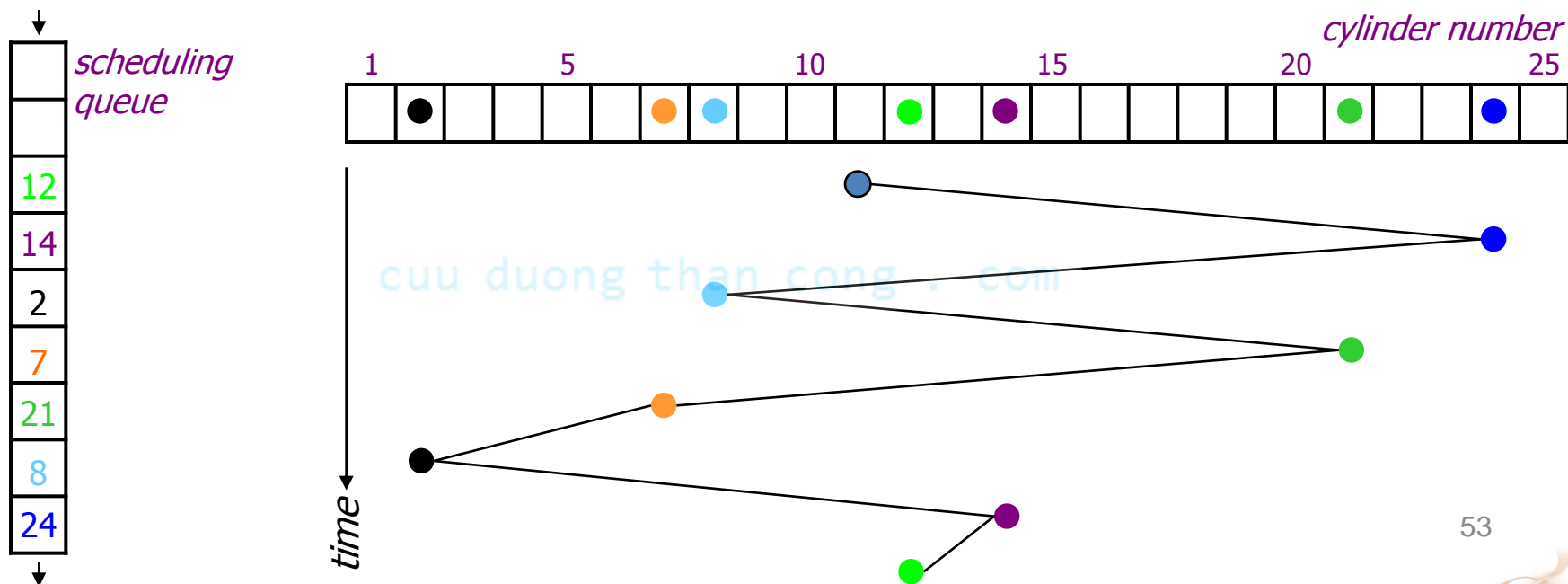


# First Come First Serve - FCFS

- Phục vụ theo thứ tự yêu cầu
- Đơn giản nhưng không đáp ứng tốt dịch vụ

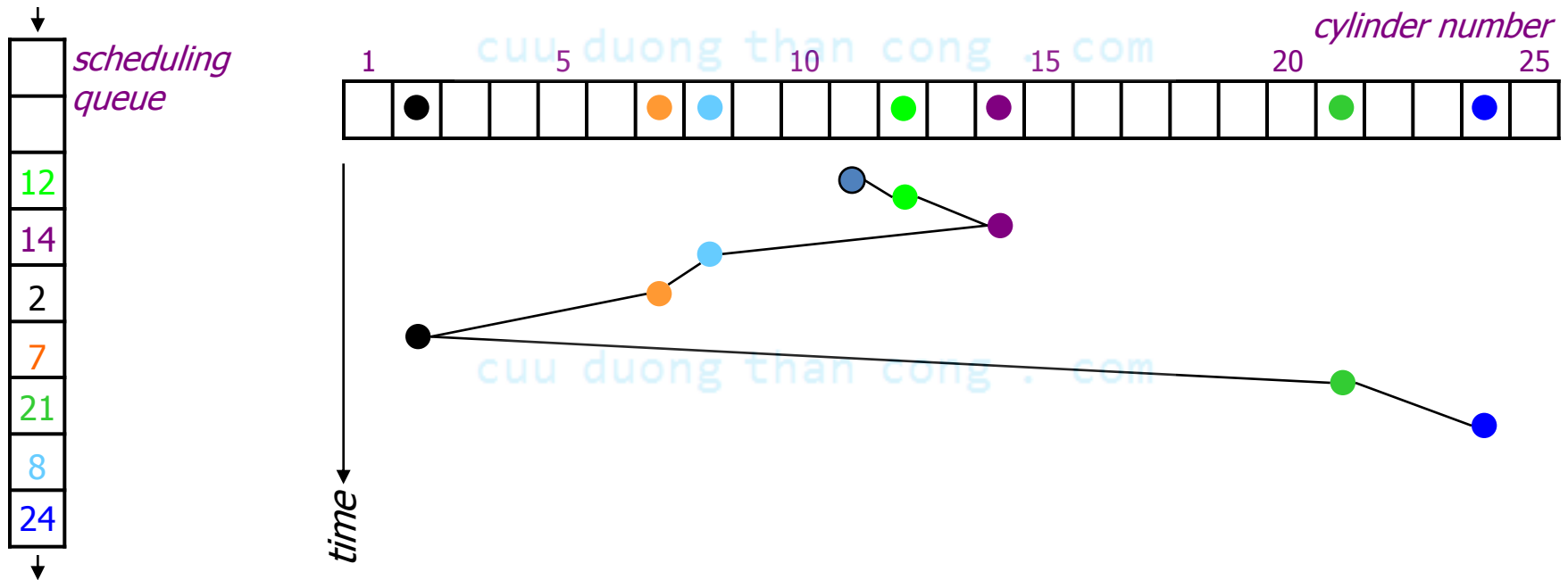
Các khối cần đọc (đầu đọc hiện tại tại vị trí 11):

24 8 21 7 2 14 12



# Shortest Seek Time First - SSTF

- Chọn nhu cầu gần với vị trí hiện hành nhất.
- ➔ Có nhiều yêu cầu chờ ..chờ...và chờ...



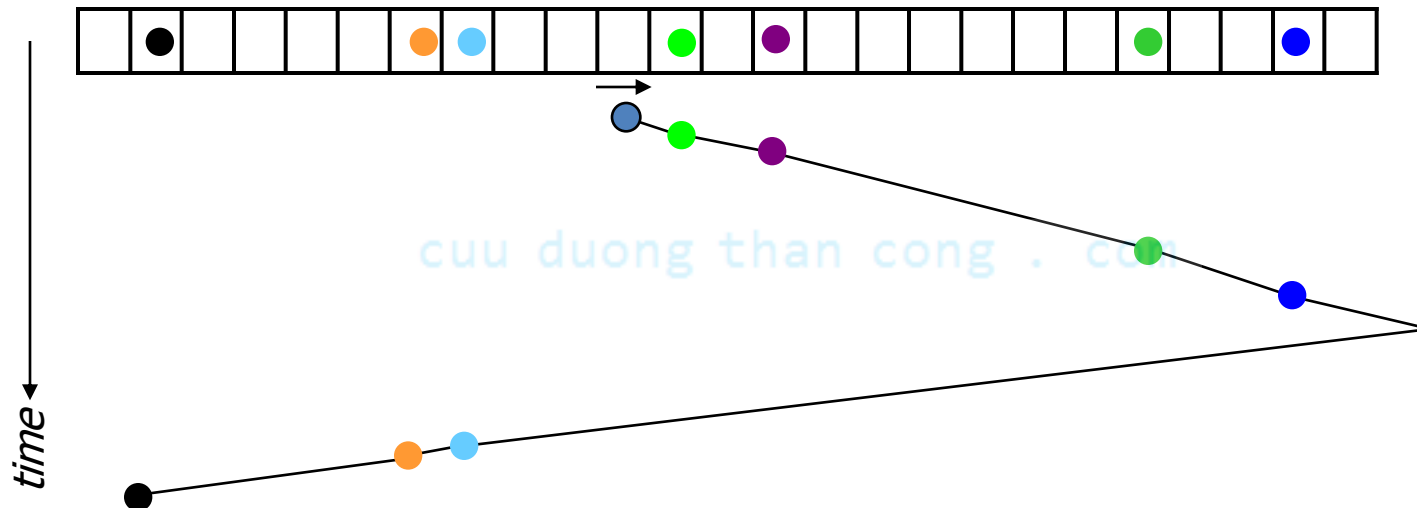
# SCAN

- Di chuyển đầu đọc về 1 phía của đĩa đến block xa nhất sau đó di chuyển về phía kia.
- Còn gọi là thuật toán thang máy.

Các khối cần đọc (đầu đọc hiện tại tại vị trí 11):

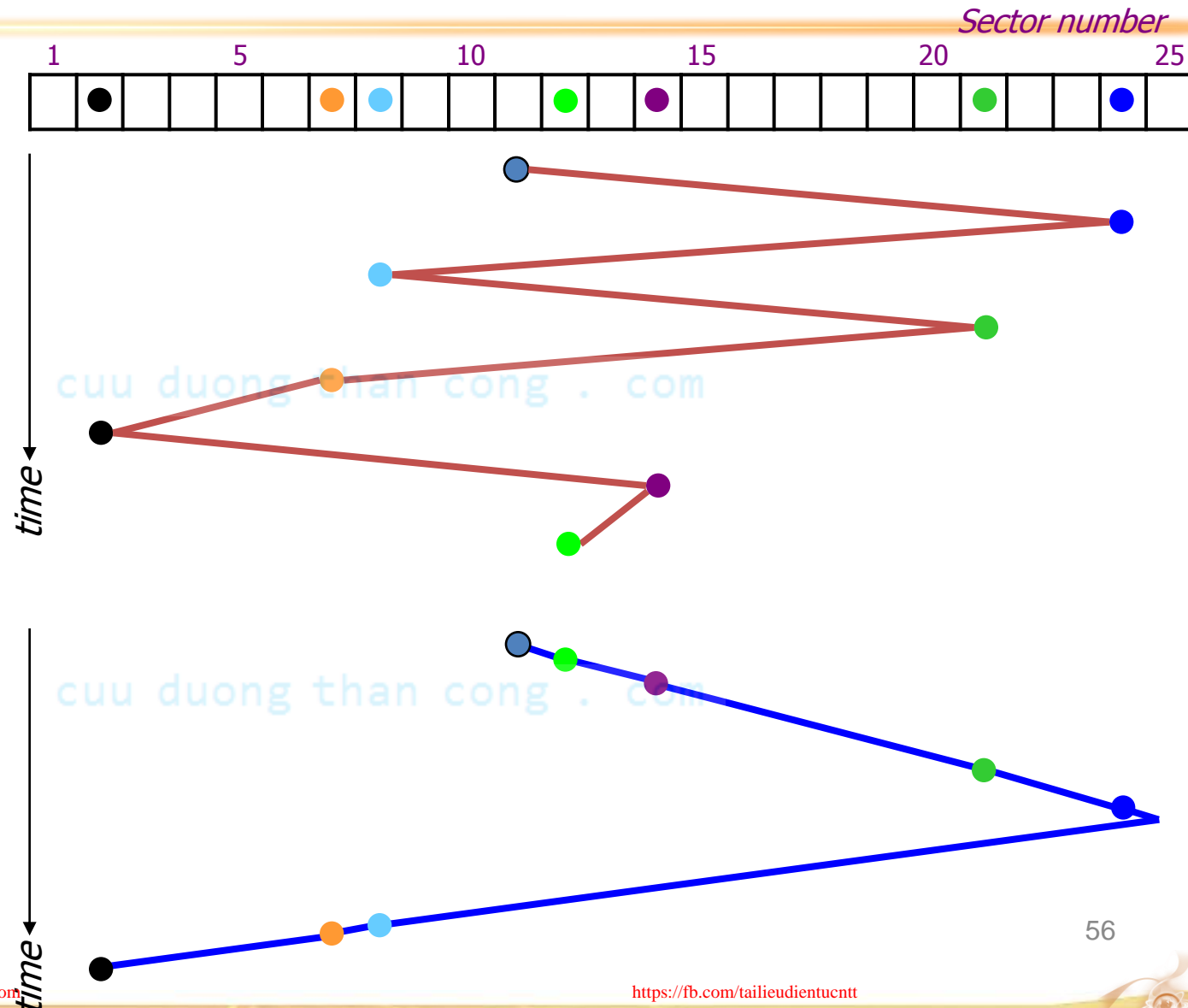
12 14 2 7 21 8 24

cuu duong than cong . com



# SCAN vs. FCFS

- Trong trường hợp này, SCAN tốt hơn FCFS vì hạn chế sự di chuyển của đầu đọc đĩa

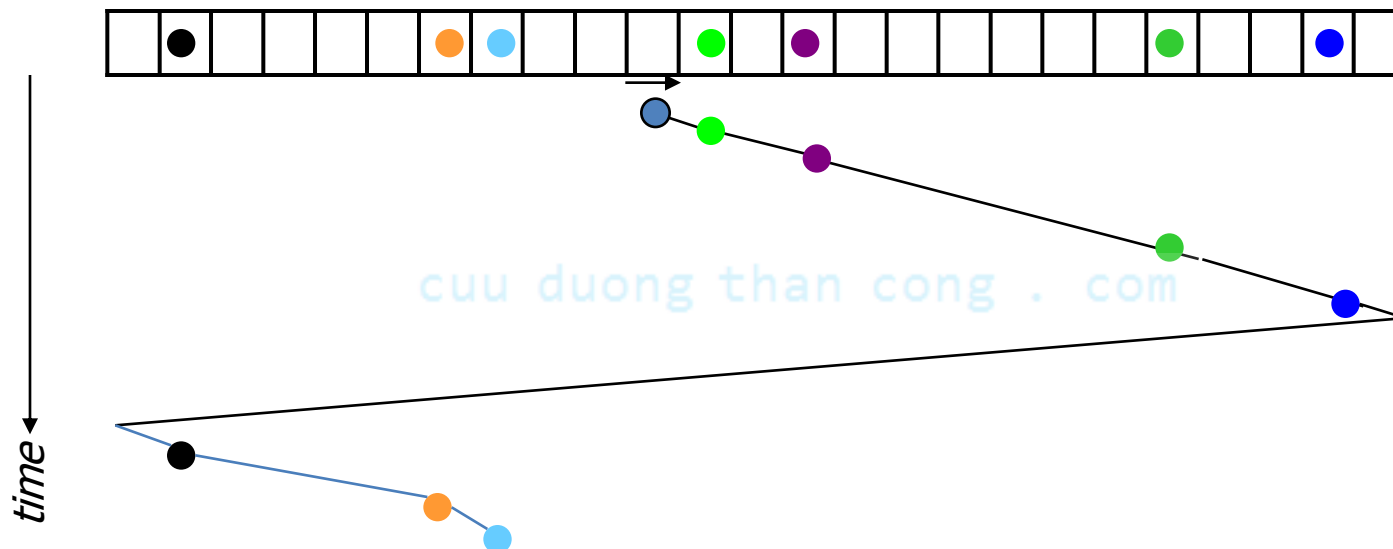




# C-SCAN

- Nguyên tắc:
  - Tương tự thuật toán SCAN.
  - Chỉ khác khi di chuyển đến 1 đầu của đĩa thì trở về vị trí bắt đầu của đĩa. Các khối cần đọc (đầu đọc hiện tại vị trí 11):

12 14 2 7 21 8 24



# LOOK – C-LOOK

- Nhận xét:
  - Hai thuật toán lập lịch SCAN và C-SCAN luôn luôn di chuyển đầu đọc của đĩa từ đầu này sang đầu kia và di chuyển đến khối cuối cùng ở mỗi hướng.
- Nguyên tắc:
  - Giống SCAN và C-SCAN nhưng chỉ di chuyển đầu đọc đến khối xa nhất chứ không đến cuối.



# LOOK – C-LOOK

Các khối cần đọc (đầu đọc hiện tại tại vị trí 11):

12 14 2 7 21 8 24

