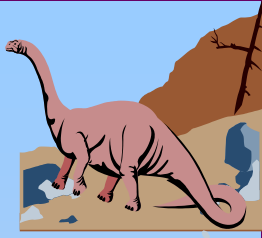
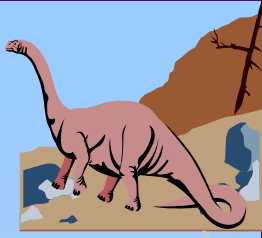
Chương 2: Hệ thống tập tin



- Đặt vấn đề
- Xây dựng các khái niệm cần thiết
- Thiết kế mô hình tập tin & thư mục
- Tổ chức hệ thống tập tin lên Volume
- Cài đặt các chức năng trên Volume

I. Đặt vấn đề

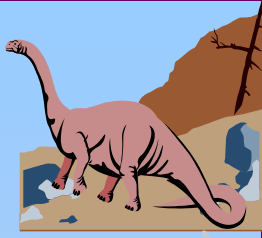


- Hệ thống lưu trữ của các loại máy tính cao cấp thường có kích thước rất lớn nên sẽ chứa rất nhiều dữ liệu.
- Hệ thống máy tính phải có khả năng lưu lại được các thông tin, dữ liệu (sao cho không mất khi tắt máy)
- Dữ liệu buộc phải lưu trữ trên bộ nhớ ngoài, mà tốc độ truy xuất trên bộ nhớ ngoài thì rất chậm (mỗi thao tác thường mất hơn 1ms), bắt buộc phải có các xử lý tối ưu tốc độ.
- Lượng dữ liệu lưu trữ quá lớn thì nếu không khéo việc quản lý /truy cập sẽ khó khăn & hao tốn thời gian.

☞ **Cần có các hình thức tổ chức, sắp xếp dữ liệu một cách hợp lý & xây dựng các thuật toán tối ưu để có thể truy cập nhanh chóng hiệu quả.**

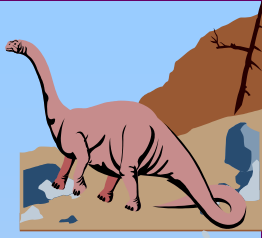
(đây chính là việc xây dựng Hệ thống quản lý tập tin – 1 thành phần chính của HĐH)

II. Xây dựng các khái niệm cần thiết



- **Tập Tin**
  - Khi lưu trữ nội dung thông tin ta cần gắn kèm những thuộc tính cần thiết như tên thông tin, kích thước, ngày giờ tạo ra,... để dễ dàng cho việc quản lý sau này. Tập hợp những thứ đó được gọi là Tập tin.
  - Tập tin là đơn vị lưu trữ thông tin.
- **Thư mục**
  - Khi có nhiều tập tin quá thì phải có hệ thống thư mục, các tập tin có cùng tính chất /đặc điểm nào đó nên đưa vào 1 thư mục, để việc tìm kiếm & quản lý tập tin được dễ dàng nhanh chóng.
  - Mỗi thư mục có thể có nhiều tập tin và nhiều thư mục con bên trong.
- **Volume**
  - Là một dãy "sector" được tổ chức theo một kiến trúc hợp lý để có thể lưu trữ lên đó một hệ thống tập tin & thư mục.
  - Các thiết bị lưu trữ dung lượng lớn có thể tổ chức trên đó nhiều vol.

III. Thiết kế mô hình tập tin & thư mục



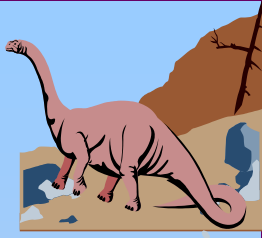
Mô hình thuộc tính của tập tin

- ***Nội dung tập tin***:

  - Là dãy byte tuần tự, dãy các record chiều dài cố định hay theo cấu trúc cây.

  - Được lưu trữ dưới dạng nén hay không nén, nếu có nén thì dùng những phương pháp nén nào và cơ chế phân tích tập tin để xác định phương pháp nén tương ứng là như thế nào

III. Thiết kế mô hình tập tin & thư mục



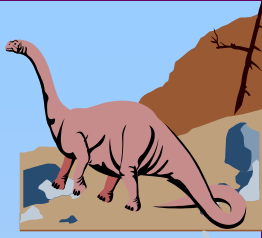
Mô hình thuộc tính của tập tin

- ***Tên tập tin***:

  - Ngoài phần tên chính ra có phần tên mở rộng hay không, bao nhiêu phần mở rộng.
  - Các thông tin chi tiết tương ứng cho từng phần:

  + chiều dài tối đa & tối thiểu
  + sử dụng bảng mã nào
  + có phân biệt chữ cái thường /hoa không
  + có cho dùng khoảng trắng không
  + các ký tự không được dùng
  + các ký tự đại diện
  + các chuỗi không được trùng
  + giá trị các byte rác khi tên chưa đạt tới chiều dài tối đa (byte kết thúc/ chiều dài)
  + ...

III. Thiết kế mô hình tập tin & thư mục



Mô hình thuộc tính của tập tin

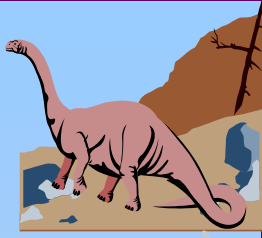
- ***Kích thước***:

  - Ngoài kích thước phần nội dung tập tin còn có kích thước phần nào khác không
  - Lưu trữ bằng bao nhiêu byte, dưới dạng số nguyên không dấu hay có dấu

  * Lưu ý: số nguyên được lưu ngược theo byte, ví dụ số 2020 (tức 7E4 ở hệ thập lục phân) lưu vào 4 byte ở offset 1 và 2 byte ở offset 6 sẽ như sau:

xx	E4	07	00	00	xx	E4	07	xx
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*

III. Thiết kế mô hình tập tin & thư mục



Mô hình thuộc tính của tập tin

- ***Thời điểm tạo***: ngày tháng năm & giờ phút giây lưu thành các trường riêng biệt hay ghép lại thành 1 dãy bit

Ví dụ: xét thời điểm ngày **30/03/2020 – 12:34:56** và cơ chế lưu trữ số là Little Engdian

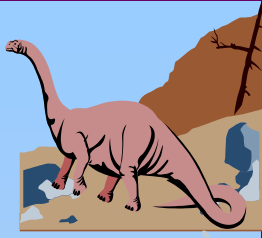
- Nếu lưu theo thứ tự <Ngày – Tháng – Năm – Giờ – Phút – Giây> mỗi thứ bằng 1 byte (riêng năm 2 byte) thì dãy byte tương ứng là: **1E 03 E4 07 0C 22 38**

- Nếu lưu ngày tháng năm trong 2 byte và giờ phút giây trong 2 byte kế tiếp theo hình thức: ***ngày chiếm 5bit, tháng 4bit, năm 7bit (lưu giá trị của hiệu <năm> - 1980); giây chiếm 5bit (lưu giá trị <giây>/2), phút 6bit, giờ 5bit*** thì dãy byte tương ứng là:

7E 50 5C 64

(vì 507Eh = 0101000001111110b, 645Ch = 0110010001011100b)

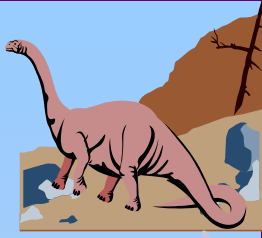
III. Thiết kế mô hình tập tin & thư mục



Mô hình thuộc tính của tập tin

- **Các thuộc tính trạng thái**: lưu riêng hay ghép chung trong 1 dãy bit và tổ chức vào 1 hoặc vài byte
- .. .. ..

III. Thiết kế mô hình tập tin & thư mục



Mô hình thuộc tính của tập tin

Ví dụ: Các thuộc tính của tập tin tin nhắn trên điện thoại di động có thể thiết kế như sau: *(giả sử sector có kích thước 162 Byte)*

- **Tên**: Kiểu chuỗi tối đa 11 ký tự - kết thúc bởi ký tự NULL, cho phép trùng, là tên người nhắn nếu có trong danh bạ - hoặc số điện thoại nhắn tới.

- **Nội dung**: lưu tuần tự các ký tự (kết thúc bởi ký tự NULL, nếu nhiều hơn 160B thì cắt ra nhiều đoạn 160B lưu trên nhiều sector, 2byte cuối của sector lưu chỉ số sector kế tiếp), các ký tự trong nội dung được lưu theo bảng mã chuẩn 1B hoặc bảng mã mở rộng 2B (UniCode dựng sẵn).

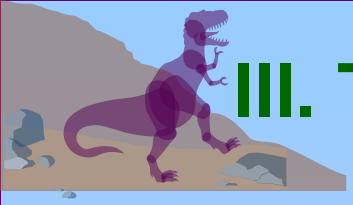
- **Bảng mã sử dụng**: số nguyên 1B

- **Chỉ số của dữ liệu hình ảnh đính kèm**: lưu dạng số nguyên 1B (bằng 255 nếu không có, dữ liệu hình ảnh được lưu ở nơi khác)

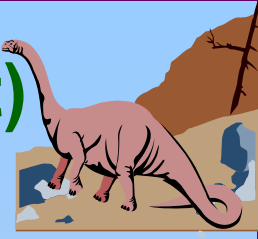
- **Trạng thái đã đọc hay chưa**: số nguyên 1B, lưu số lần mở lên xem nội dung (bằng 0 nếu chưa xem, nếu quá 255 lần thì vẫn lưu 255)

- **Ngày giờ gửi tới**: có thể lưu như ở ví dụ trước

- …

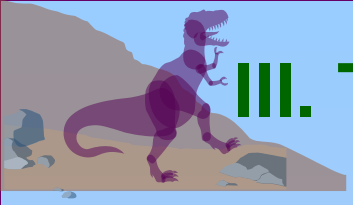


III. Thiết kế mô hình tập tin & thư mục (tt)

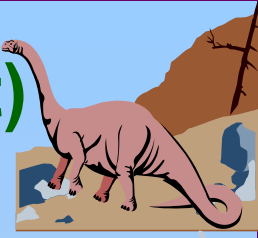


- **Mô hình thuộc tính của thư mục**
  - Thiết kế tương tự như tập tin. Vì có khá nhiều thuộc tính giống nhau nên có thể tổ chức một mô hình chung cho cả tập tin lẫn thư mục.
  - Khi này thư mục được coi là một tập tin đặc biệt và có 1 thuộc tính để phân biệt với tập tin bình thường.

Ví dụ: Nếu tập tin (bình thường) có 8 thuộc tính, thư mục có 6 thuộc tính, và cả hai có 5 thuộc tính giống nhau thì tập tin tổng quát sẽ có 5+3+1+1 = 10 thuộc tính



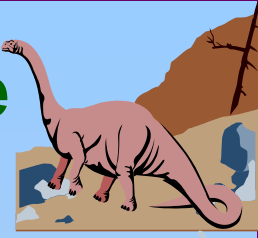
III. Thiết kế mô hình tập tin & thư mục (tt)



Mô hình chức năng

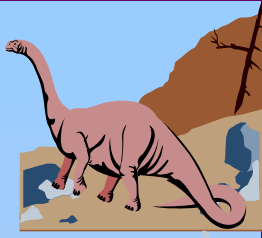
- ***Liệt kê danh sách các tập tin***: ở gốc /ở thư mục con /cây thư mục – theo thứ tự của thuộc tính nào, các thuộc tính nào cần hiển thị & hiển thị như thế nào,..
- ***Đổi tên***: tên mới có hợp lệ không, có bị trùng không
- ***Đổi mật khẩu truy xuất tập tin***: yêu cầu nhập mật khẩu cũ (nếu có) và nhập đúng thì yêu cầu nhập mật khẩu mới 2 lần. Nếu hợp lệ thì mã hóa nội dung tập tin theo công thức tương ứng với mật khẩu mới
- ***Xóa***: xóa bình thường, xóa nhưng không mất, xóa mất hẳn, xóa cây thư mục, xóa rác, xóa phần mềm,...
- ***Đọc***: ..
- ***Ghi***: ..
- ... ...

IV. Tổ chức hệ thống tập tin lên Volume



- **1. Các nhận xét & phân tích cần thiết**
  - *(i) Phải xác định các vị trí còn trống.*
  - *(ii) Mỗi sector (hoặc tổng quát hơn là mỗi block) chỉ thuộc tối đa 1 tập tin.*
  - *(iii) Tên & các thuộc tính của tập tin cần được lưu riêng vào 1 vùng.*
  - *(iv) Phải có thông tin vị trí bắt đầu của nội dung tập tin (do phân tích trên).*
  - *(v) Nội dung tập tin không bắt buộc phải liên tục.*
  - *(vi) Phải biết các vị trí chứa nội dung tập tin (do phân tích trên).*
  - *(vii) Phải biết các vị trí bị hư.*
  - *(viii) Nội dung tập tin nên lưu trữ theo đơn vị là CLUSTER (là dãy N sector liên tiếp – để dễ quản lý & việc truy xuất được nhanh hơn)*

IV. Tổ chức hệ thống tập tin lên Vol (tt)



2. Cluster

- ***Khái niệm:***
  - Đơn vị đọc ghi trên đĩa là sector, nhưng đơn vị lưu trữ nội dung tập tin không phải là một sector mà là một cluster gồm N sector liên tiếp (N≥1).
  - ☞ *Mỗi vị trí trong các phân tích trên sẽ là 1 cluster.*
  - ☞ *Cluster chỉ tồn tại trên vùng dữ liệu (DATA) – nơi chứa nội dung tập tin*
- ***Lý do phát sinh:***
  - Nếu sector trên vùng dữ liệu quá nhiều thì có thể sẽ khó hoặc không quản lý được, khi đó quản lý trên cluster sẽ dễ dàng hiệu quả hơn.
  - Nội dung tập tin thường chiếm nhiều sector và có thể không liên tục nên đặt đơn vị lưu trữ là cluster sẽ hạn chế được sự phân mảnh, đồng thời truy xuất một lần ***n*** sector liên tiếp nhanh hơn so với truy xuất ***n*** lần mà mỗi lần chỉ 1 sector.

IV. Tổ chức hệ thống tập tin lên Vol (tt)

2. Cluster

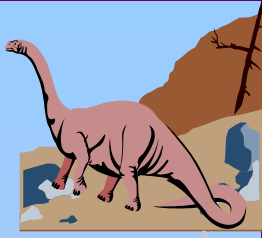
Hình thức tổ chức:

- Volume sẽ được chia thành 2 vùng: vùng dữ liệu (DATA) chứa nội dung tập tin và vùng hệ thống (SYSTEM) chứa các thông tin quản lý.
- Vùng SYSTEM có kích thước nhỏ hơn nhiều so với vùng DATA và phải truy xuất mỗi khi sử dụng Volume nên thường nằm ngay đầu Volume,
- Trên vùng DATA là một dãy các Cluster liên tiếp được đánh chỉ số theo thứ tự tăng dần (bắt đầu từ 0, 1 hay 2… tùy theo Hệ Điều Hành).

Ví dụ: Nếu Volume có kích thước 4014 sector, vùng SYSTEM chiếm 11 sector, mỗi cluster chiếm 4 sector, Cluster đầu tiên được đánh chỉ số là 2; thì phân bố Cluster trên Volume sẽ như sau:

				Cluster 2				Cluster 3				Cluster 4				...	Cluster 1001			
		...														...				
0	1	...	10	11	12	13	14	15	16	17	18	19	20	21	22	...	4007	4008	4009	4010
SYSTEM AREA				DATA AREA																

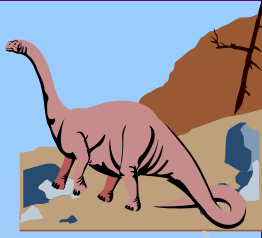
IV. Tổ chức hệ thống tập tin lên Vol (tt)



2. Cluster

- ***Kích thước Cluster:***
  - Phụ thuộc nhiều yếu tố: dung lượng vol, tốc độ truy xuất một dãy sector, kích thước của đa số tập tin sẽ lưu vào, số cluster tối đa có thể quản lý, nhu cầu của người sử dụng,…
  - Kích thước cluster càng lớn sẽ càng lãng phí đĩa, nhưng sẽ hạn chế sự phân mảnh tập tin và vì vậy có thể an toàn hơn & truy xuất nhanh hơn.
  - Trên các đĩa cứng hiện tại thì sector thường có kích thước 512B và Cluster thường có chiếm 4, 8 hoặc 16 sector (và là lũy thừa của 2 trên Windows).
  - Trên các đĩa /thẻ nhớ Flash thông dụng (dung lượng 128MB đến 2 GB), kích thước cluster nên là 8 hoặc 16KB. Nếu có nhiều tập tin nhỏ thì nên nén /ghép lại thành 1 tập tin kích thước lớn hơn.

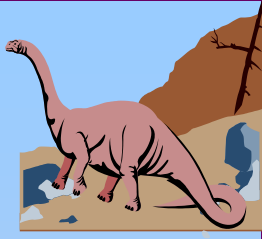
IV. Tổ chức hệ thống tập tin lên Vol (tt)



3. Bảng quản lý Cluster

- ***Khái niệm:***
  - Được đưa ra để xác định danh sách các cluster chứa nội dung của một tập tin và các cluster trống.
  - Có thể dùng 1 bảng chung, cũng có thể tổ chức nhiều bảng. Thông thường mỗi phần tử trên bảng sẽ quản lý 1 cluster tương ứng trên vùng DATA
- ***Lý do phát sinh:***
  - Phải xác định các cluster còn trống để có thể chép tập tin vào Vol.
  - Phải xác định danh sách các cluster chứa nội dung của một tập tin để có thể đọc nội dung tập tin.
  - Có thể lưu thông tin quản lý ngay trên cluster nhưng khi đó quản lý & truy xuất rất chậm nên nhất thiết phải lập ra bảng này để nhanh hơn.

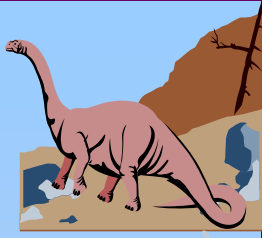
IV. Tổ chức hệ thống tập tin lên Vol (tt)



3. Bảng quản lý Cluster

- ***Hình thức tổ chức:***
  - Quản lý cluster hư:
    - Lưu bằng 1 danh sách trực tiếp: giá trị mỗi phần tử là chỉ số của 1 cluster hư, vì số cluster hư rất ít nên danh sách này có thể qui định là 1 hoặc vài sector.
    - Kết hợp với các trạng thái luận lý khác trong 1 bảng chỉ mục.
  - Quản lý Cluster trống:
    - Dùng hình thức **bitmap:** mỗi bit quản lý 1 cluster tương ứng - cluster K trống khi giá trị của bit K là 0, *bit K nằm tại byte (K div 8) và là bit (K mod 8) trên byte đó*.
    - Quản lý theo dạng chỉ mục: mỗi phần tử của bảng quản lý là 1 con số nói lên trạng thái của cluster mang chỉ số tương ứng.
    - Quản lý vùng trống: tổ chức 1 danh sách các phần tử, mỗi phần tử chứa vị trí bắt đầu & kích thước của vùng trống tương ứng.

IV. Tổ chức hệ thống tập tin lên Vol (tt)



3. Bảng quản lý Cluster

- ***Hình thức tổ chức:***
  - Quản lý **chuỗi các cluster chứa nội dung của 1 tập tin**:
    - Lưu trữ nội dung tập tin trên dãy cluster liên tiếp (danh sách đặc).
    - Sử dụng cấu trúc danh sách liên kết (xâu)
    - Sử dụng cấu trúc xâu kết hợp chỉ mục (index).
    - Sử dụng cấu trúc cây (kết hợp chỉ mục).

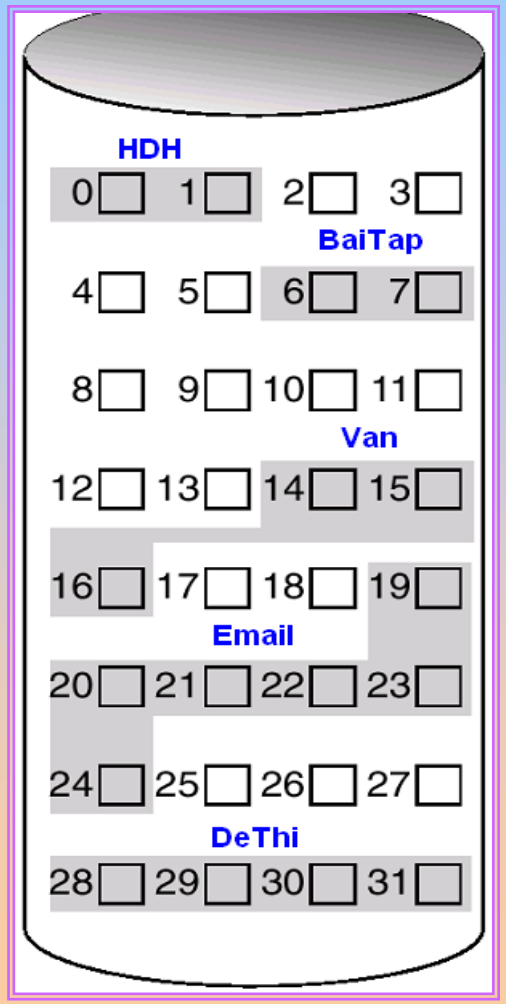
IV. Tổ chức hệ thống tập tin lên Vol (tt)

3. Bảng quản lý Cluster

Hình thức tổ chức:

- **Quản lý chuỗi các cluster khi nội dung tập tin lưu trữ liên tiếp**: Chỉ cần lưu vị trí cluster bắt đầu và kích thước tập tin (theo byte).

  Ví dụ: Nếu cluster có kích thước là 1K, và các tập tin trên vol là: HDH (0, 2006), BaiTap (6, 2007), Van (14, 2050), Email(19, 6000), DeThi(28, 4096) thì phân bố các cluster của những tập tin đó sẽ như hình bên



Khuyết điểm: tập tin khó tăng trưởng & có thể không lưu được 1 tập tin kích thước bình thường dù không gian trống còn rất lớn.

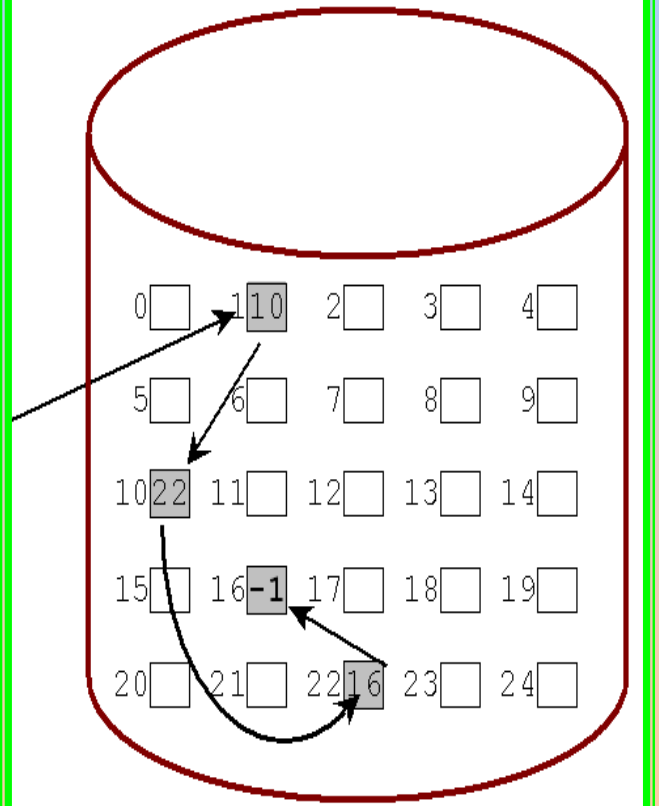
IV. Tổ chức hệ thống tập tin lên Vol (tt)

3. Bảng quản lý Cluster

- ***Hình thức tổ chức:***
  - **Quản lý chuỗi các cluster của tập tin bằng xâu**: Mỗi cluster sẽ là 1 phần tử của xâu, gồm 2 vùng: vùng chứa nội dung tập tin và vùng liên kết chứa chỉ số cluster kế sau.

    Ví dụ: Nếu tập tin chiếm các cluster là 1, 10, 22, 16 thì phân bố các cluster của những tập tin đó trong đĩa sẽ như hình vẽ

    Khuyết điểm: việc xác định danh sách các cluster của tập tin sẽ rất chậm – vì xem như phải truy xuất luôn nội dung tập tin.



IV. Tổ chức hệ thống tập tin lên Vol (tt)

3. Bảng quản lý Cluster

- ***Hình thức tổ chức:***
  - **Quản lý chuỗi các cluster của tập tin bằng xâu kết hợp chỉ mục:** mỗi phần tử là 1 số nguyên dùng để quản lý 1 cluster (phần tử K quản lý cluster K). Với qui định nếu phần tử K trên bảng có giá trị là L và L bằng:
    - FREE /BAD: cluster K đang ở trạng thái trống /hư.
    - EOF: cluster K là cluster cuối của tập tin.
    - Khác: cluster K chứa nội dung tập tin & kế sau là cluster L.

    Ví dụ: Nếu tập tin chiếm các cluster là 1, 10, 22, 16 thì nội dung bảng sẽ như sau:

	10									22					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
eof						16									
16	17	18	19	20	21	22	23	24	25	26	27	28	...	...	

IV. Tổ chức hệ thống tập tin lên Vol (tt)

3. Bảng quản lý Cluster

Hình thức tổ chức:

Quản lý chuỗi các cluster của tập tin bằng xâu kết hợp chỉ mục(tt):

- Hình thức này được áp dụng cho bảng FAT (File Allocation Table) của DOS & Windows, cũng như rất nhiều HĐH trên các thiết bị kỹ thuật số.
- Có 3 loại FAT: FAT12, FAT16 & FAT32 (mỗi phần tử của bảng có kích thước 12, 16, hoặc 32 bit). Nếu phần tử K của FAT có giá trị L thì trạng thái của cluster K là:

Trạng thái của Cluster K	Giá trị của L			Ghi chú
	FAT12	FAT16	FAT32	
Trống	0	0	0	= **FREE**
Hư	FF7	FFF7	0FFFFFF7	= **BAD**
Cluster cuối của tập tin	FFF	FFFF	0FFFFFFF	= **EOF**
Chứa nội dung tập tin – và có cluster kế sau là L	2..FEF	2..FFEF	2..0FFFFFEF	

IV. Tổ chức hệ thống tập tin lên Vol (tt)

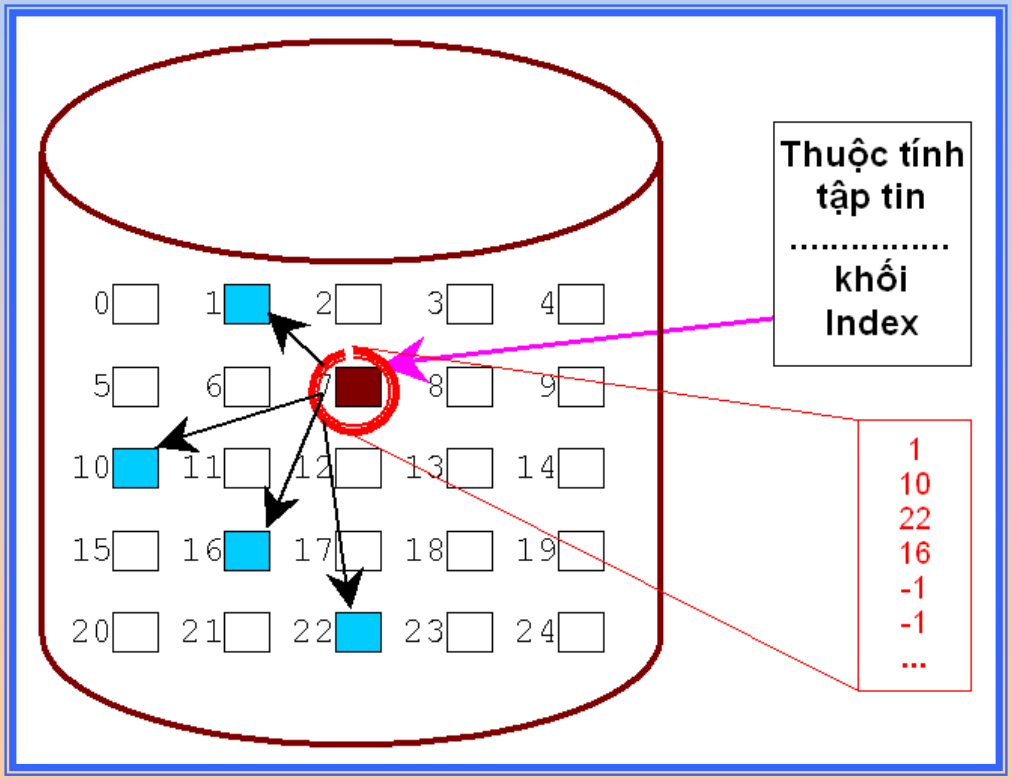
3. Bảng quản lý Cluster

- *Hình thức tổ chức:*
  - **Quản lý chuỗi các cluster của tập tin bằng cấu trúc cây:**

    Khi số khối mà tập tin chiếm quá lớn thì quản lý bằng xâu sẽ dễ mất mát dữ liệu nghiêm trọng.

➔ Cần phải quản lý bằng cây nhiều nhánh, nút lá của cây sẽ chứa danh sách chỉ số trực tiếp của các cluster.

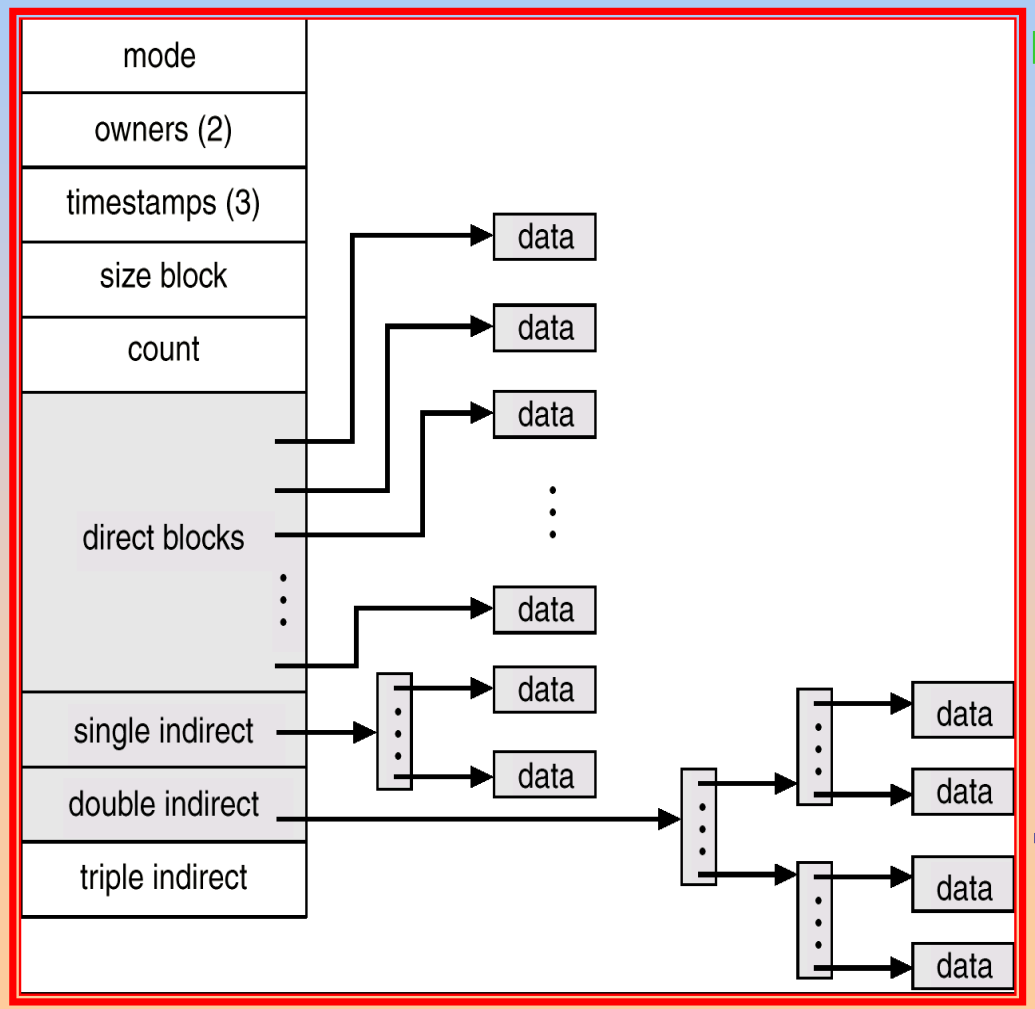
Ví dụ: Nếu tập tin chiếm các khối là 1, 10, 22, 16 thì nội dung khối index (khối 7) sẽ như hình vẽ.



IV. Tổ chức hệ thống tập tin lên Vol (tt)

3. Bảng quản lý Cluster

Hình thức tổ chức:



- **Quản lý chuỗi các cluster của tập tin bằng cấu trúc cây(tt)**

  Ví dụ: Kiến trúc I-node của UNIX:

  - Mỗi khối có 4KB.
  - Mỗi phần tử là 1 số nguyên 4 byte.
  - Trong mỗi bảng Indirect có 1024 con trỏ, mỗi con trỏ trỏ đến 1 bảng cấp kế dưới.
  - Trong mỗi bảng Single Indirect có 1024 chỉ số cluster.
  - Ở I-node gốc có 10 chỉ số cluster.

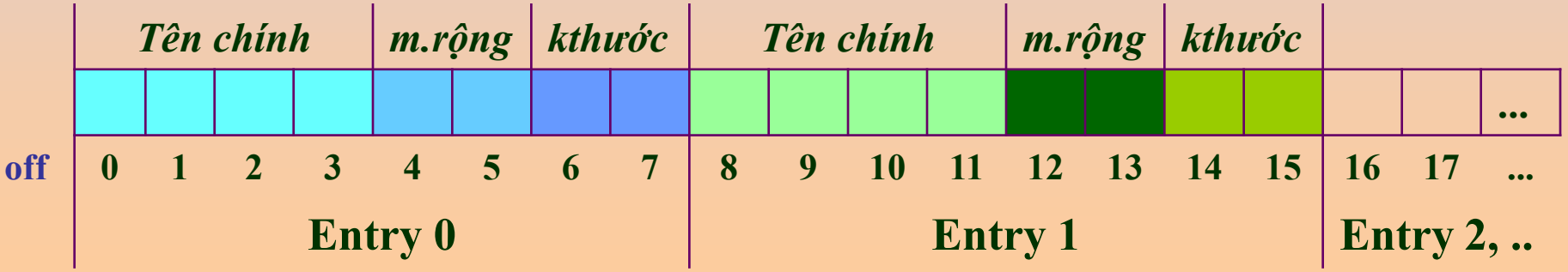
➔ Hệ thống dùng cây 1024 nhánh & mỗi tập tin có thể có hơn 1 tỉ khối.

IV. Tổ chức hệ thống tập tin lên Vol (tt)

4. Bảng thư mục

- ***Khái niệm:***
  - Là một dãy phần tử (entry), mỗi phần tử chứa tên & các thuộc tính của một tập tin
  - Thường gồm 2 loại: RDET (Root Directory Entry Table) và SDET (Sub Directory Entry Table)

    Ví dụ, nếu mô hình thuộc tính tập tin được thiết kế chỉ gồm các thành phần: tên chính (chuỗi tối đa 4 ký tự), tên mở rộng (chuỗi tối đa 2 ký tự), kích thước tập tin (số nguyên 2 byte) thì RDET sẽ như sau:



IV. Tổ chức hệ thống tập tin lên Vol (tt)

4. Bảng thư mục

- ***Lý do phát sinh:***
  - Để tăng tốc thời gian truy xuất nội dung tập tin, tối ưu hiệu quả quản lý

Ví dụ, với cấu trúc Entry của DOS như sau:

Offset	Độ dài	Ý nghĩa
0	8	Tên chính *(lưu bằng bảng mã ASCII, ký tự rác có mã 20h)*
8	3	Tên mở rộng *(tương tự như trên)*
B	1	Thuộc tính trạng thái *(0-0-A-D-V-S-H-R)*
C	A	Không dùng *(để dành cho các version sau)*
16	2	Giờ cập nhật *(giây/2: 5bit, phút: 6bit, giờ: 5bit)*
18	2	Ngày cập nhật *(ngày: 5, tháng: 4, năm-1980: 7)*
1A	2	Cluster bắt đầu *(2..TổngsốCluster+1 hoặc EOF)*
1C	4	Kích thước *(bằng 0 nếu entry là thư mục)*

IV. Tổ chức hệ thống tập tin lên Vol (tt)

4. Bảng thư mục

Thì entry có dãy byte:

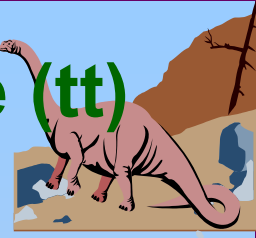
44 45 54 48 49 20 20 20 44 4F 43 20 00 00 00 00

00 00 00 00 00 00 16 4A BB 34 14 00 1E 0A 00 00

sẽ có giá trị các thuộc tính như sau:

- Tên: **Dethi.Doc**
- Kích thước: <số 4byte tại offset 1Ch> = A1Eh = **2590 byte**
- Cluster bắt đầu: <số 2byte tại offset 1Ah> = 14h = **20**
- Ngày cập nhật: **27/05/2006** (vì dãy 16bit tương ứng tại offset 18h là 34BBh = 0011010010111011b nên ngày = 11011b = 27, tháng = 0101b = 5, năm = 1980 + 0011010b = 2006).
- Giờ cập nhật là 9:16:44 (vì dãy bit tương ứng là 4A16h = 01001010000010110b nên giây = 2 * 10110b = 44, phút = 010000b = 16, giờ = 01001b = 9)

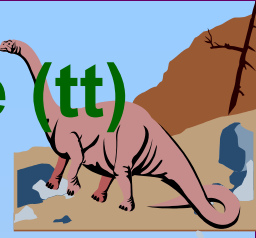
IV. Tổ chức hệ thống tập tin lên Volume (tt)



5. Boot Sector

- ***Khái niệm:***
  - *Chứa một đoạn chương trình nhỏ để nạp Hệ Điều Hành khi khởi động máy (Boot).*
  - *Chứa các thông số quan trọng của volume: loại volume, kích thước vol, kích thước cluster, kích thước bảng quản lý cluster, ...*
  - *Vùng BootSector (chứa 1 số sector) nằm ngay đầu volume, Boot Sector là sector đầu tiên (trong trường hợp không thể tổ chức đủ thông tin trong sector đầu thì tổ chức tiếp trên các sector còn lại trong vùng).*

IV. Tổ chức hệ thống tập tin lên Volume (tt)



- **5. Boot Sector**
  - ***Hình thức tổ chức:***
    - *Đoạn chương trình Boot trong sector này sẽ nạp và chạy tiếp các đoạn chương trình trong các sector khác (vì đoạn chương trình trong 1 sector quá nhỏ).*
    - *Mỗi thông số được qui định nằm tại một offset cụ thể cố định nào đó (với kích thước lưu trữ & kiểu dữ liệu tương ứng).*
    - *Nếu cần thiết, có thể tổ chức các thông số trên các sector khác trong vùng BootSector, cũng có thể tổ chức các thông số ngay trên các thành phần quản lý thư mục và quản lý khối mà không cần đến vùng Boot Sector*

IV. Tổ chức hệ thống tập tin lên Volume (tt)

5. Boot Sector

Ví dụ, cấu trúc Boot Sector của kiến trúc FAT như sau:

Offset	Số byte	Ý nghĩa
0	3	Lệnh nhảy đến đầu đoạn mã Boot
3	8	Tên công ty /version của HĐH
B	**2**	**Số byte của sector** (thường là 512)
D	**1**	**Số sector của cluster (S_C)**
E	**2**	**Số sector trước bảng FAT (S_B)**
10	**1**	**Số lượng bảng FAT (N_F)**, thường là 2
11	**2**	**Số Entry của RDET (S_R)**
13	**2**	**Số sector của volume (S_V)**
16	**2**	**Số sector của FAT (S_F)**
20	**4**	**Số sector của volume** (nếu số 2 byte tại offset 13h là 0)
3E	1CF	Đoạn chương trình Boot nạp tiếp HĐH khi khởi động máy
1FE	2	Dấu hiệu kết thúc BootSector (luôn là AA55h)

IV. Tổ chức hệ thống tập tin lên Volume (tt)

5. Boot Sector

thì với Boot Sector có phần đầu:

```
EB 3C 90 4D 53 57 49 4E 34 2E 31 00 02 10 01 00
02 00 02 00 00 F8 FF 00 3F 00 FF 00 3F 00 00 00
C2 EE 0F 00 80 00 29 DE 1C 49 15 20 20 20 20 20
20 20 20 20 20 20 46 41 54 31 36 20 20 20 33 C9
8E D1 BC F0 7B 8E D9 B8 00 20 8E C0 FC BD 00 7C
38 4E 24 7D 24 8B C1 99 E8 3C 01 72 1C 83 EB 3A
66 A1 1C 7C 26 66 3B 07 26 8A 57 FC 75 06 80 CA
02 88 56 02 80 C3 10 73 EB 33 C9 8A 46 10 98 F7
```

các thông tin có thể suy ra:

* *2 byte tại offset 0B là: 00, 02 ➔ Số byte trên mỗi sector của vol là: 0200h = 512d (byte)*
* *Giá trị của byte tại offset 0D là: 10 ➔ Số sector trên mỗi cluster của vol là: S_C = 10h = 16d*
* *2 byte tại offset 0E là: 01, 00 ➔ Số sector trước vùng FAT là: S_B = 0001h = 1d (sector)*
* *Giá trị của byte tại offset 10 là: 02 ➔ Số bảng FAT của vol là: N_F = 02h = 2d (bảng)*
* *2 byte tại offset 16 là: FF, 00 ➔ Kích thước bảng FAT là: S_F = 00FFh = 255d (sector)*
* *4 byte tại offset 20 là: C2, EE, 0F, 00 ➔ Tổng số sector trên vol là: S_V = FEEC2h = 1044162d*
  (vì 2 byte tại offset 13 đều là 00 nên kích thước vol được lấy ở 4 byte tại offset 20)