

NGUYỄN LÝ HỆ ĐIỀU HÀNH

Phạm Đăng Hải
haipd@soict.hust.edu.vn

Bộ môn Khoa học Máy tính
Viện Công nghệ Thông tin & Truyền Thông



1 / 108

Ngày 14 tháng 2 năm 2020

Notes

Chương 4 Quản lý hệ thống file



2 / 108

Notes

Giới thiệu

- Bộ nhớ ngoài (*đĩa từ, băng từ, đĩa quang...*): dung lượng lớn và cho phép lưu trữ lâu dài
 - Được người dùng sử dụng lưu trữ dữ liệu và chương trình
 - Dữ liệu và chương trình được lưu dưới dạng file (*tập tin/tệp*)
 - ⇒ Tạo nên hệ thống file
 - Hệ thống file gồm 2 phần riêng biệt
 - Các file: Chứa dữ liệu/chương trình của hệ thống/người dùng
 - Cấu trúc thư mục : Cung cấp các thông tin về file
 - Hệ thống file lớn ⇒ Quản lý như thế nào?
 - Các thuộc tính của file, thao tác cần phải cung cấp?
 - Lưu trữ và truy xuất dữ liệu trên thiết bị lưu trữ như thế nào?
 - Phương pháp cung cấp không gian lưu trữ, quản lý vùng tự do
- ⇒ Khó khăn phải trong suốt với người dùng (*tính thuận tiện*)
- Các file dữ liệu /chương trình có thể sử dụng chung
 - Đảm bảo tính toàn vẹn dữ liệu và loại bỏ truy nhập bất hợp lệ?
 - Dữ liệu không lưu trữ tập trung ⇒ hệ thống file phân tán
 - Truy nhập file từ xa, đảm bảo tính toàn vẹn...



3 / 108

Notes

Chương 4: Quản lý hệ thống file

Nội dung chính

1

Hệ thống file

2

Cài đặt hệ thống file

3

Tổ chức thông tin trên đĩa từ

4

Hệ thống FAT



4 / 108

Notes

Chương 4: Quản lý hệ thống file

1. Hệ thống file

Nội dung chính

1

Hệ thống file

2


Cài đặt hệ thống file

3

Tổ chức thông tin trên đĩa từ

4

Hệ thống FAT



5 / 108

Notes

Chương 4: Quản lý hệ thống file

1. Hệ thống file

1.1 Khái niệm file

1


Hệ thống file

●

Khái niệm file

●

Cấu trúc thư mục



6 / 108

Notes

Chương 4: Quản lý hệ thống file
1. Hệ thống file
1.1 Khái niệm file

Các thao tác cơ bản

❶ Tạo file (*Create*)

❷ Ghi file (*Write*)

❸ Đọc file (*Read*)

❹ Thay đổi vị trí trong file (*Seek*)

❺ Xóa file (*Delete*)

❻ Thu gọn file (*Truncate*)

❼ ...



10 / 108

Notes

Chương 4: Quản lý hệ thống file
1. Hệ thống file
1.1 Khái niệm file

Các thao tác cơ bản : Tạo file

Thư mục file


hello.c	SoNT.dat
Vị trí	Vị trí
	19/04/2011

Create(SoNT.dat)

hello.c

Không gian lưu trữ

- Tìm vùng tự do trong không gian lưu trữ của hệ thống file
 - Cung cấp vùng trống như thế nào?
- Tạo một phần tử mới trong thư mục file
- Lưu tên file, vị trí của file và các thông tin khác



11 / 108

Notes

Chương 4: Quản lý hệ thống file
1. Hệ thống file
1.1 Khái niệm file

Các thao tác cơ bản : Ghi file

Thư mục file


hello.c	SoNT.dat	kiemtra.pdf
Vị trí	Vị trí	
	19/04/2011	

Write(SoNT.dat, 17)
Write(SoNT.dat, 19)
Write(SoNT.dat, 23)
Write(SoNT.dat, 29)

hello.c

Không gian lưu trữ

- Lỗi gọi hệ thống Write() yêu cầu tên file và dữ liệu được ghi
- Dùng tên file, tìm kiếm file trong thư mục file
- Dựa vào trường vị trí, tìm vị trí của file trên thiết bị lưu trữ
- Hệ thống lưu con trỏ ghi (*write pointer*) để chỉ ra vị trí ghi
 - Con trỏ ghi thay đổi sau mỗi thao tác ghi

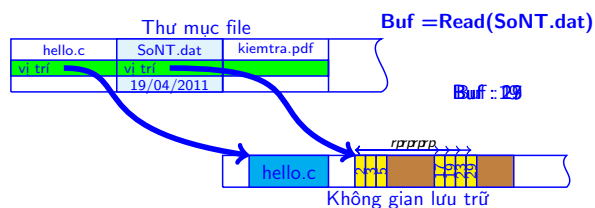


12 / 108

Notes

Chương 4: Quản lý hệ thống file
1. Hệ thống file
1.1 Khái niệm file

Các thao tác cơ bản : Đọc file



- Lỗi gọi hệ thống Read() yêu cầu tên file và vùng đệm ghi KQ
- Dùng tên file, tìm kiếm file trong thư mục file
- Dựa vào trường vị trí, tìm vị trí của file trên thiết bị lưu trữ
- Hệ thống lưu con trỏ đọc (*read pointer*) chỉ ra vị trí được đọc
 - Con trỏ đọc thay đổi sau mỗi thao tác đọc dữ liệu
- Dùng một con trỏ cho cả thao tác đọc và ghi: **con trỏ file**

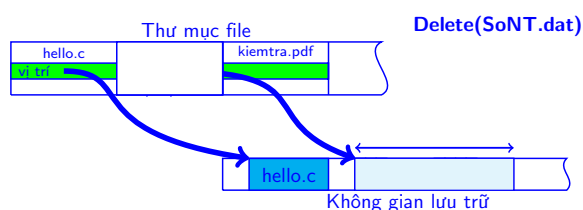


13 / 108

Notes

Chương 4: Quản lý hệ thống file
1. Hệ thống file
1.1 Khái niệm file

Các thao tác cơ bản : Xóa file



- Dùng tên file, tìm kiếm file trong thư mục file
- Vùng nhớ được xác định bởi 2 trường vị trí và kích thước được giải phóng để có thể sử dụng lại bởi các file khác
- Xóa phần tử tương ứng trong thư mục file
- Xóa logic / xóa vật lý



14 / 108

Notes

Chương 4: Quản lý hệ thống file
1. Hệ thống file
1.1 Khái niệm file

Các thao tác cơ bản : Thay đổi vị trí trong file và thu gọn file

- Thay đổi vị trí trong file
 - Duyệt thư mục để tìm phần tử tương ứng
 - Con trỏ file được thay bằng giá trị thích hợp
 - Thao tác này không yêu cầu một hoạt động vào/ra
- Thu gọn file
 - Được sử dụng khi người sử dụng muốn xóa nội dung file nhưng vẫn giữ nguyên các thuộc tính
 - Tìm kiếm file trong thư mục file
 - Đặt kích thước file về 0
 - Giải phóng vùng nhớ dành cho file



15 / 108

Notes

- Ngoài các thao tác cơ bản, còn tồn tại nhiều thao tác khác
 - Thêm dữ liệu vào cuối file (*append*)
 - Lấy/đặt thông tin thuộc tính file
 - Đổi tên file
- Có thể được đảm bảo thông qua các thao tác cơ bản.
Ví dụ copy file
 - Tạo file mới
 - Đọc dữ liệu từ file cũ
 - Ghi ra file mới



Notes

- Các thao tác file phải duyệt thư mục file \Rightarrow Lãng phí thời gian
- Để giải quyết, các tiến trình phải thực hiện mở file (*open*) trước khi thao tác với file
 - Thao tác mở file: tìm kiếm file trong thư mục file
 - Chép phần tử tương ứng vào bảng file mở
 - Chứa thông tin về các file đang được mở
 - Trả lại con trỏ của phần tử tương ứng trong bản file mở
- Khi có yêu cầu, HĐH tìm kiếm trong bảng file mở
 - Dùng con trỏ trả về của thao tác mở file
- Khi không sử dụng file nữa cần phải đóng (*close*) file.
 - HĐH sẽ loại bỏ phần tử tương ứng trong bảng file mở
- Thao tác đóng/mở file trong môi trường đa người dùng
 - Dùng 2 loại bảng file mở: Cho từng tiến trình và cho hệ thống
 - Ghi lại số tiến trình đang mở file (*File Open Counter*)
 - Tăng/Giảm bộ đếm khi có tiến trình mở/đóng file
 - Xóa p/tử tương ứng trong bảng file mở mức hệ thống khi bộ đếm bằng không



Notes

- 1 Hệ thống file**
- Khái niệm file
 - Cấu trúc thư mục



Notes

Chương 4: Quản lý hệ thống file

1. Hệ thống file

1.2 Cấu trúc thư mục

Các phân vùng (Partition)

Thư mục

Files

Thư mục


Files

Phân vùng A

Phân vùng B

Disk 1

- Đĩa được chia thành nhiều phân vùng
 - Partitions, Minidisks, Volumes
- Mỗi phân vùng được xử lý như vùng lưu trữ phân biệt
- Có thể chứa một HDH riêng



19 / 108

Notes

Chương 4: Quản lý hệ thống file

1. Hệ thống file

1.2 Cấu trúc thư mục

Các phân vùng (Partition)

Thư mục

Files


Disk 2

Disk 3

Phân vùng C

Kết hợp một vài đĩa thành một cấu trúc logic lớn

- Người dùng chỉ quan tâm tới cấu trúc file và thư mục logic
- Không quan tâm tới cách phân phối vật lý không gian đĩa cho files



20 / 108

Notes


Chương 4: Quản lý hệ thống file

1. Hệ thống file

1.2 Cấu trúc thư mục

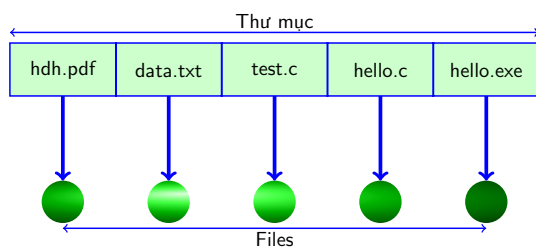
Các thao tác với thư mục

- Mỗi một phân khu lưu các thông tin về file trong nó
 - Các thông tin file được lưu trữ trong *thư mục thiết bị* - thư mục
- Thư mục là bảng chuyển cho phép ánh xạ từ một tên (*file*) thành một phần tử trong thư mục
 - Thư mục có thể được cài đặt bằng nhiều cách khác nhau
 - Yêu cầu các thao tác chèn, tạo mới, xóa, duyệt danh sách
- Các thao tác
 - Tìm kiếm file:** Tìm phần tử ứng với một file xác định
 - Tạo file:** Tạo file mới cần tạo phần tử trong thư mục
 - Xóa file:** Khi xóa file, xóa phần tử tương ứng trong thư mục
 - Liệt kê thư mục:** Liệt kê files và nội dung phần tử tương ứng trong thư mục
 - Đổi tên file:** Thay đổi tên file, vị trí trong cấu trúc thư mục
 - Duyệt hệ thống file:** Truy nhập tất cả thư mục và nội dung tất cả các files trong thư mục (*backup dữ liệu lên băng từ*)



21 / 108

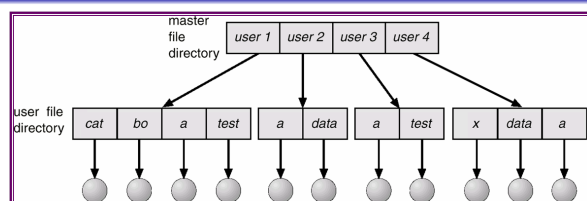
Notes



- Cấu trúc đơn giản nhất, các file nằm trong cùng một thư mục
- Số người dùng và số file lớn, khả năng trùng tên file cao
 - Mỗi người dùng một thư mục riêng



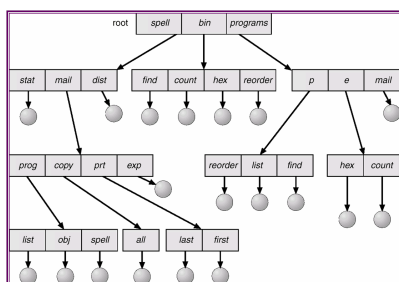
Notes



- Mỗi người sử dụng có một thư mục riêng, khi làm việc với file chỉ duyệt thư mục riêng
- Khi log in, hệ thống sẽ kiểm tra và cho phép người sử dụng làm việc với thư mục riêng
- Khi thêm một người dùng
 - Hệ thống tạo phần tử mới trong *Master file directory*
 - Tạo ra *User file directory*
- Giả quyết v/dề trùng tên; Hiệu quả khi người dùng độc lập
- Khó khăn khi muốn dùng chung file



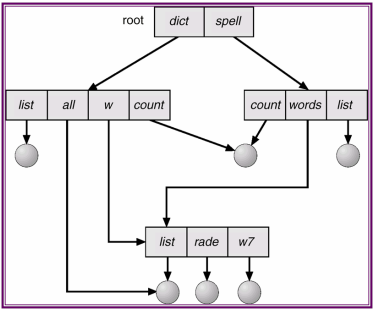
Notes



- Tồn tại một đường dẫn (*tương đối/tuyệt đối*) đến một file
- Thư mục con là file được xử lý đặc biệt (*bit đánh dấu*)
- Các thao tác tạo/xóa/duyệt... t/hiện trên thư mục hiện thời
 - Xóa thư mục con \Rightarrow Xóa hết các cây con của nó



Notes



- Người dùng có thể *link* đến một file của người dùng khác
- Khi duyệt thư mục (*backup*) file có thể duyệt nhiều lần
- Xóa file: liên kết/ nội dung (*người tạo file /liên kết cuối*)



Notes

1. Hệ thống file
2. Cài đặt hệ thống file
3. Tổ chức thông tin trên đĩa từ
4. Hệ thống FAT



Notes

2. Cài đặt hệ thống file
 - Cài đặt thư mục
 - Các phương pháp phân phối vùng lưu trữ
 - Quản lý vùng lưu trữ tự do



Notes

- ❶ Danh sách tuyến tính với con trỏ tới các khối dữ liệu
 - Đơn giản cho lập trình
 - Tốn thời gian khi thực hiện các thao tác với thư mục
 - Phải duyệt toàn bộ danh sách \Leftarrow **Dùng cây nhị phân?**

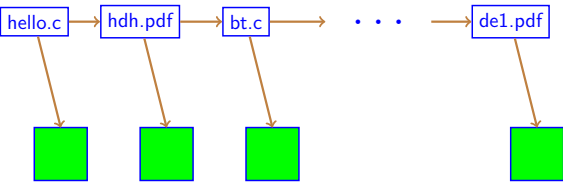
- ❷ Bảng băm - Bảng băm với danh sách tuyến tính
 - Giảm thời gian duyệt thư mục
 - Đòi hỏi có một hàm băm hiệu quả

$$h(\text{Name}) = \frac{\sum_{i=1}^{\text{Len}(\text{Name})} \text{ASCII}(\text{Name}[i])}{\text{Table_Size}}$$

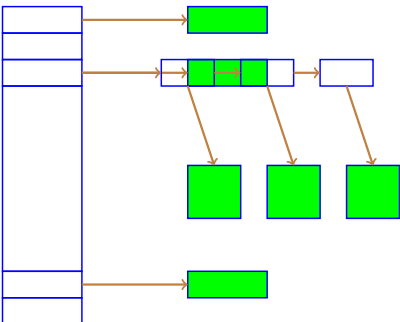
- Vấn đề đụng độ \Leftarrow hàm băm trả về cùng một kết quả với 2 tên file khác nhau
- Vấn đề kích thước cố định \rightarrow Tăng kích thước phải tính toán lại những phần đã tồn tại



Notes



Notes



Notes

2 Cài đặt hệ thống file

- Cài đặt thư mục
- Các phương pháp phân phối vùng lưu trữ
- Quản lý vùng lưu trữ tự do



Notes

Mục đích

- Tăng hiệu năng truy nhập tuần tự
- Dễ dàng truy nhập ngẫu nhiên tới file
- Dễ dàng quản lý file

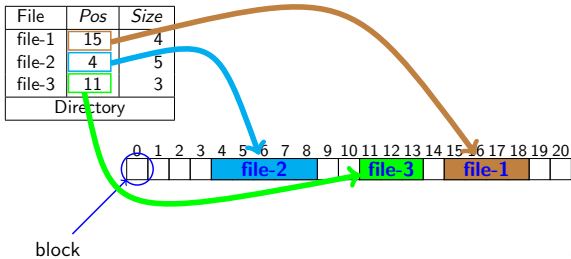
Phương pháp

- 1 Phân phối liên tục (*Continuous Allocation*)
- 2 Phân phối liên kết (*Linked List Allocation*)
- 3 Phân phối chỉ mục (*Indexed Allocation*)



Notes

Nguyên tắc: File được phân phối các khối nhớ liên tiếp nhau



Notes

Phân phối liên tục (tiếp tục)

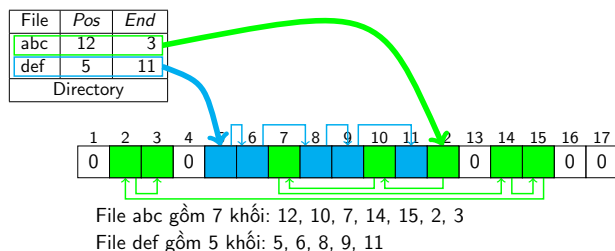
- File có độ dài n và bắt đầu ở khối b sẽ chiếm các khối $b, b + 1, \dots, b + n - 1$
 - Hai khối b và $b + 1$ liên tiếp nhau
 - ⇒ Không phải dịch chuyển đầu từ khi đọc (trừ sector cuối)
 - ⇒ Tốc độ truy nhập nhanh
 - Cho phép truy nhập trực tiếp khối i của file
 - ⇒ truy nhập khối $b + i - 1$ trên thiết bị lưu trữ
- Lựa chọn vùng trống khi có yêu cầu lưu trữ?
 - Các chiến lược *First-Fit* / *Worst Fit* / *Best Fit*
 - Hiện tượng phân đoạn ngoài
- Khó khăn khi muốn tăng kích thước của file



Notes

Phân phối liên kết

Nguyên tắc: File được phân phối các khối nhớ không liên tục.
Cuối mỗi khối là con trỏ, trỏ tới khối tiếp theo



Notes

Phân phối liên kết(tiếp tục)

- Chỉ áp dụng hiệu quả cho các file truy nhập tuần tự
 - Để truy nhập khối thứ n , phải duyệt qua $n - 1$ khối trước đó
 - Các khối không liên tục, phải định vị lại đầu từ
 - Tốc độ truy nhập chậm
 - Các khối trong file được liên kết bởi con trỏ. Nếu con trỏ lỗi?
 - Bị mất dữ liệu do mất liên kết tới khối
 - Liên kết tới khối không có dữ liệu hoặc khối của file khác
- Giải quyết:** Sử dụng nhiều con trỏ trong mỗi khối ⇒ **Tổn nhớ**
- Áp dụng: FAT**
 - Được sử dụng như danh sách liên kết
 - Gồm nhiều phần tử, mỗi phần tử ứng với một khối
 - Mỗi phần tử trong FAT, chứa khối tiếp theo của file
 - Khối cuối cùng có giá trị đặc biệt (*FFFF*)
 - Khối bị hỏng có giá trị (*FFF7*)
 - Khối chưa sử dụng có giá trị (*0*)
 - Trường vị trí trong bản ghi file, chứa khối đầu tiên của file



Notes

Chương 4: Quản lý hệ thống file

2. Cài đặt hệ thống file

2.2 Các phương pháp phân phối vùng lưu trữ

Phân phối chỉ mục

Nguyên tắc: Mỗi file có một khối chỉ mục chính (*index block*) chứa danh sách các khối dữ liệu của file

File

Index block

abc

5

def

12

Directory

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

File abc gồm 6 khối: 2, 15, 4, 8, 9, 11

File def gồm 2 khối: 10, 13

10

13

-1

...

-1

37 / 108

Notes

Chương 4: Quản lý hệ thống file

2. Cài đặt hệ thống file

2.2 Các phương pháp phân phối vùng lưu trữ

Phân phối Phân phối chỉ mục (tiếp tục)

- Phần tử thứ i của khối chỉ mục trỏ tới khối thứ i của file
 - Đọc khối i dùng con trỏ được khi tại p/tử i của khối chỉ mục
- Tạo file, các phần tử của khối chỉ mục có giá trị *null* (-1)
- Cần thêm khối i , địa chỉ khối được cấp, được đưa vào p/tử i
- Nhận xét
 - Không gây hiện tượng phân đoạn ngoài
 - Cho phép truy nhập trực tiếp
 - Cần khối chỉ mục: file có k/thước nhỏ, vẫn cần 2 khối
 - Khối cho dữ liệu
 - Khối chỉ khối chỉ mục (*chỉ dùng 1 phần tử*)

Giải quyết: Giảm kích thước khối \Rightarrow Giảm phí tổn bộ nhớ \Rightarrow Vấn đề về kích thước file có thể lưu trữ.

- Sơ đồ liên kết
 - Liên kết các khối chỉ mục lại
 - P/tử cuối của khối chỉ mục trỏ tới khối chỉ mục khác nếu cần
- Index nhiều mức
 - Dùng một khối chỉ mục trỏ tới các khối chỉ mục khác

38 / 108

Notes

Chương 4: Quản lý hệ thống file

2. Cài đặt hệ thống file

2.2 Các phương pháp phân phối vùng lưu trữ

Sơ đồ kết hợp (UNIX)

mode

owners (2)

timestamps (3)

size block

count

direct blocks

single indirect

double indirect

triple indirect

data

data

data

...

data

data

data

data

data

data

data

data

- 12 *direct block* trỏ tới data block
- Single indirect block* chứa địa chỉ khối *direct block*
- Double indirect block* chứa địa chỉ khối *Single indirect block*
- Triple indirect block* chứa địa chỉ khối *Double indirect*

39 / 108

Notes

2 Cài đặt hệ thống file

- Cài đặt thư mục
- Các phương pháp phân phối vùng lưu trữ
- Quản lý vùng lưu trữ tự do



Notes

- 1 Bit vector
 - Mỗi block thể hiện bởi 1 bit (1 : free; 0 : allocated)
 - Dễ dàng tìm ra n khối nhớ liên tục
 - Cần có lệnh cho phép làm việc với bit
- 2 Danh sách liên kết (*link list*)
 - Lưu giữ con trỏ tới khối đĩa trống đầu tiên
 - Khối nhớ này chứa con trỏ tới khối đĩa trống tiếp theo
 - Không hiệu quả khi duyệt danh sách
- 3 Nhóm (*Grouping*)
 - Lưu trữ địa chỉ n khối tự do trong khối tự do đầu tiên
 - $n - 1$ khối đầu tự do, khối n chứa đ/chỉ của n khối tự do tiếp
 - Ưu điểm: Tìm vùng nhớ tự do nhanh chóng
- 4 Bộ đếm (*Counting*)
 - Do các khối nhớ liên tục được c/cấp và g/phóng đồng thời
 - **Nguyên tắc:** Lưu địa chỉ khối nhớ tự do đầu tiên và kích thước vùng nhớ liên tục trong DS quản lý vùng trống
 - Hiệu quả khi bộ đếm lớn hơn 1



Notes

- 1 Hệ thống file
- 2 Cài đặt hệ thống file
- 3 Tổ chức thông tin trên đĩa từ
- 4 Hệ thống FAT



Notes

3 Tổ chức thông tin trên đĩa từ

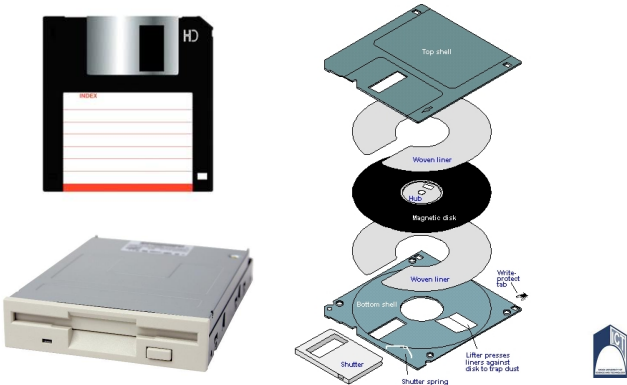
- Cấu trúc vật lý của đĩa
- Cấu trúc logic của đĩa



Notes

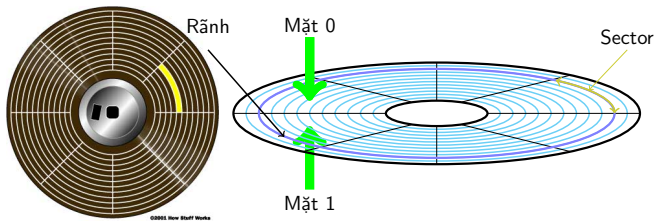


Notes



Notes

Cấu trúc vật lý đĩa mềm



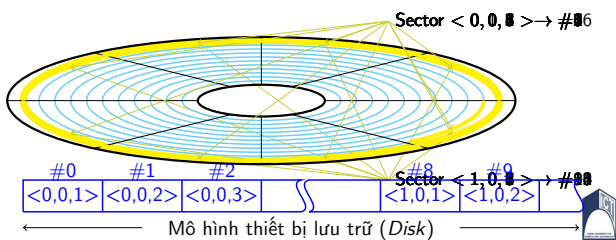
- Mặt đĩa. Mỗi mặt đĩa được đọc bởi một đầu đọc (*Header*)
 - Các đầu từ được đánh số 0, 1
- Rãnh đĩa (*Track*): Các vòng tròn đồng tâm
 - Được đánh số 0, 1, ... từ ngoài vào trong
- Cung từ (*Sector*)
 - Được đánh số 1, 2, ...



Notes

Định vị thông tin trên đĩa mềm

- Sector đơn vị thông tin hệ thống dùng làm việc với đĩa
- Sector xác định qua tọa độ 3 chiều: Header, Track, Sector
 - Ví dụ: Boot Sector của đĩa mềm: Sector <0, 0, 1>
- Sector được xác định qua số hiệu sector (*tọa độ 1 chiều*)
 - Vị trí tương đối so với sector đầu tiên của đĩa



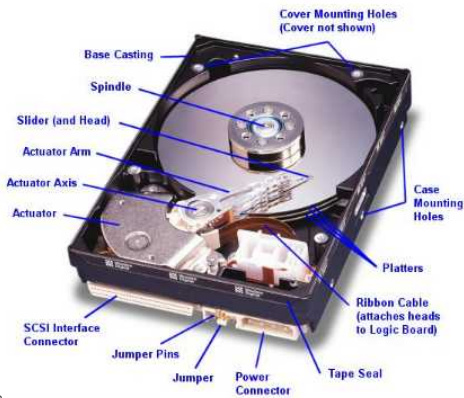
Notes

Đĩa cứng



Notes

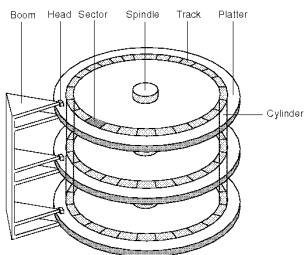
Đĩa cứng



49 / 108

Notes

Cấu trúc vật lý đĩa cứng



Cấu trúc

- Gồm nhiều mặt đĩa, được đánh số từ 0,1
- Các rãnh cùng bán kính tạo nên cylinder, được đánh số từ 0, 1,...
- Các sector trên mỗi mặt của mỗi cylinder, được đánh số từ 1,2,...

Định vị thông tin

- Tọa độ 3 chiều (H, C, S)
- Tọa độ 1 chiều: Số hiệu sector
- Nguyên tắc như với đĩa mềm: Sector→Header→Cylinder

50 / 108

Notes

Truy nhập sector trên đĩa

- Sector là đơn vị thông tin máy tính dùng để làm việc với đĩa từ
- Có thể truy nhập (đọc/ghi/format/...) tới từng sector
- Truy nhập sử dụng ngắt BIOS 13h (chức năng 2, 3, 5,...)
 - Không phụ thuộc hệ điều hành
 - Sector được xác định theo địa chỉ <H,C,S>
- Truy nhập sử dụng lời gọi hệ thống
 - Ngắt của hệ điều hành
 - Ví dụ: MSDOS cung cấp ngắt 25h/26h cho phép đọc/ghi các sector theo địa chỉ tuyến tính
 - Sử dụng hàm WIN32 API
 - CreateFile()/ReadFile()/WriteFile()...

51 / 108

Notes

Chương 4: Quản lý hệ thống file
3. Tổ chức thông tin trên đĩa từ
3.1 Cấu trúc vật lý của đĩa

Sử dụng ngắt 13h

Thanh ghi	Ý nghĩa
AH	2h:Đọc sector; 3h: Ghi Sector
AL	Số sector cần đọc
DH	Các sector phải trên cùng một mặt, một rãnh
DL	Số hiệu mặt đĩa
	Số hiệu ổ đĩa. 0h:A;
	80h: Đĩa cứng thứ nhất; 81h Đĩa cứng thứ 2
CH	Số hiệu Track/Cylinder
	(Sử dụng 10 bit, trong đó lấy 2 bit cao của CL)
CL	Số hiệu sector (chỉ sử dụng 6 bit thấp)
ES:BX	Trỏ tới vùng đệm, nơi sẽ chứa dữ liệu đọc được (khi AH=2h) hoặc dữ liệu ghi ra đĩa (Khi AH=3h)
CarryFlag	CF=0 không có lỗi; CL chứa số sector đọc được CF=1 Có lỗi, AH chứa mã lỗi

WinXP hạn chế sử dụng ngắt 13h để truy nhập trực tiếp

Notes

Chương 4: Quản lý hệ thống file
3. Tổ chức thông tin trên đĩa từ
3.1 Cấu trúc vật lý của đĩa

Sử dụng ngắt 13h (Ví dụ)

```
#include <stdio.h>
#include <dos.h>
int main(int argc, char *argv[]){
    union REGS regs;
    struct SREGS sregs;
    int Buf[512];
    int i;
    regs.h.ah = 0x02;  regs.h.al = 0x01;
    regs.h.dh = 0x00;  regs.h.dl = 0x80;
    regs.h.ch = 0x00;  regs.h.cl = 0x01;
    regs.x.bx = FP_OFF(Buf);
    sregs.es = FP_SEG(Buf);
    int86x(0x13,&regs,&regs,&sregs);
    for(i=0;i<512;i++)  printf("%4X",Buf[i]);
    return 0;
}
```

Notes

Chương 4: Quản lý hệ thống file
3. Tổ chức thông tin trên đĩa từ
3.1 Cấu trúc vật lý của đĩa

Sử dụng WIN32 API

- HANDLE CreateFile(...): Mở file/thiết bị vào ra
 - LPCTSTR lpFileName, ⇒ Tên file/thiết bị vào ra
 - "\\\\.\\C:" Phân vùng / Ổ đĩa C
 - "\\\\.\\PhysicalDrive0" Ổ đĩa cứng thứ nhất
 - DWORD dwDesiredAccess,⇒ Thao tác với thiết bị
 - DWORD dwShareMode,⇒ Cho phép dùng chung
 - LPSECURITY_ATTRIBUTES lpSecurityAttributes (NULL),
 - DWORD dwCreationDisposition,⇒ Hành động thực hiện
 - DWORD dwFlagsAndAttributes, ⇒ Thuộc tính
 - HANDLE hTemplateFile (NULL)
- BOOL ReadFile(...)
 - HANDLE hFile,⇒File muốn đọc
 - LPVOID lpBuffer, ⇒ Vùng đệm chứa dữ liệu
 - DWORD nNumberOfBytesToRead,⇒, số byte cần đọc
 - LPDWORD lpNumberOfBytesRead,⇒ số byte đọc được
 - LPOVERLAPPED lpOverlapped (NULL)
- BOOL WriteFile(...) ⇒Tham số tương tự ReadFile()

Notes

Chương 4: Quản lý hệ thống file

3. Tổ chức thông tin trên đĩa từ


3.1 Cấu trúc vật lý của đĩa

Sử dụng WIN32 API (Ví dụ)

```
#include <windows.h>
#include <stdio.h>

int main(int argc, char *argv[]){
    HANDLE hDisk;
    BYTE Buf[512];
    int byteread,i;

    hDisk=CreateFile("\\\\.\\PhysicalDrive0",GENERIC_READ,
        FILE_SHARE_READ | FILE_SHARE_WRITE,
        NULL, OPEN_EXISTING,0,NULL);
    if (hDisk==INVALID_HANDLE_VALUE) printf("Loi thiet bi");
    else {
        ReadFile(hDisk,Buf,512,&byteread,NULL);
        for(i=0;i<512;i++)  printf("%4X",Buf[i]);
        CloseHandle(hDisk);
    }
    return 0;
}
```



Notes


Chương 4: Quản lý hệ thống file

3. Tổ chức thông tin trên đĩa từ

3.1 Cấu trúc vật lý của đĩa

Kết quả thực hiện

33	C0	8E	D0	BC	00	7C	FB	50	07	50	1F	FC	BE	1B	7C	BF	1B	06	50
52	B9	E5	01	F3	A4	CB	BD	BE	07	B1	04	38	6E	00	7C	09	75	13	83
C5	10	E2	F4	CD	18	8B	F5	83	C6	10	49	74	19	38	2C	74	F6	A0	85
07	B4	07	8B	F0	AC	3C	00	74	FC	BB	07	00	B4	0E	CD	10	EB	F2	88
4E	10	E8	46	00	73	2A	FE	46	10	80	7E	04	0B	74	0B	80	7E	04	0C
74	05	A0	B6	07	75	D2	80	46	02	06	83	46	08	06	83	56	00	00	E8
21	00	73	05	A0	B6	07	EB	BC	81	3E	FE	70	55	AA	74	0B	80	7E	10
00	74	C8	A0	B7	07	EB	A9	8B	FC	1E	57	8B	F5	CB	BF	05	00	8A	56
00	B4	08	CD	13	72	23	8A	G1	24	3F	98	8A	DE	8A	FC	43	F7	E3	8B
D1	86	D6	B1	06	D2	EE	42	F7	E2	39	56	0A	77	23	72	05	39	46	08
73	1C	B8	01	02	BB	00	7C	8B	4E	02	8B	56	00	CD	13	73	51	4F	74
4E	32	E4	8A	56	00	CD	13	EB	E4	8A	56	00	60	BB	AA	55	B4	41	CD
13	72	36	81	FB	55	AA	75	30	F6	C1	01	74	2B	61	60	6A	00	6A	00
FF	76	0A	FF	76	08	6A	00	68	00	7C	6A	01	6A	10	B4	42	8B	F4	CD
13	61	61	73	0E	4F	74	0B	32	E4	8A	56	00	CD	13	EB	D6	61	B9	C3
49	6E	76	61	6C	69	64	20	70	61	72	74	69	74	69	6F	6E	20	74	61
62	6C	65	00	45	72	72	6F	72	20	6C	6F	61	64	69	6E	67	20	6F	70
65	72	61	74	69	6E	67	20	73	79	73	74	65	6D	00	4D	69	73	73	69
6E	67	20	6F	70	65	72	61	74	69	6E	67	20	73	79	73	74	65	6D	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	02	00	00	C1	FF	0F	FE	FF	FF	31	41	8A	03	0E	D3	1D	01	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	55	AA								



Notes

Chương 4: Quản lý hệ thống file

3. Tổ chức thông tin trên đĩa từ

3.2 Cấu trúc logic của đĩa

3

Tổ chức thông tin trên đĩa từ

•

Cấu trúc vật lý của đĩa

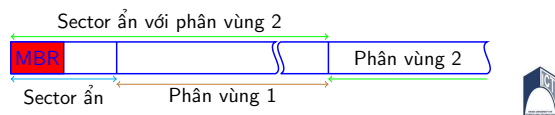
•

Cấu trúc logic của đĩa



Notes

- Đĩa mềm: Mỗi hệ điều hành có một chiến lược quản lý riêng
- Đĩa cứng (*Có dung lượng lớn*)
 - Được chia thành nhiều phân vùng (Partitions, Volumes,...)
 - Mỗi vùng là tập hợp các *Cylinder* liên tiếp nhau
 - Người dùng ấn định kích thước (Ví dụ dùng: *fdisk*)
 - Mỗi phân vùng có thể được quản lý bởi một HDH riêng
 - HDH *format* phân vùng theo định dạng được sử dụng
 - Tồn tại nhiều hệ thống khác nhau: FAT, NTFS, EXT3,...
 - Trước tất cả các phân vùng là các sector bị che
 - **Master Boot Record (MBR)**: Sector đầu tiên của đĩa



58 / 108

Notes

- Sector quan trọng nhất của đĩa
- Là sector đầu tiên trên đĩa (*Số hiệu 0 hoặc địa chỉ <0, 0, 1>*)
- Cấu trúc gồm 3 phần

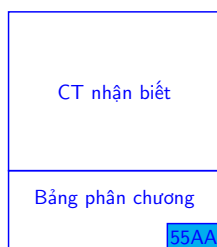
Chương trình nhận biết

- Đọc bảng phân chương để biết
 - Vị trí các phân vùng
 - Phân vùng tích cực (*chứa HDH*)
- Đọc và thực hiện sector đầu tiên của phân vùng tích cực

Bảng phân chương (64bytes)

- Gồm 4 phần tử, mỗi phần tử 16 bytes
- Mỗi phần tử chứa thông tin một vùng
 - Vị trí, kích thước, hệ thống chiếm giữ

Chữ ký hệ thống (luôn là 55AA)



59 / 108

Notes

	Stt	Ofs	Size	Ý nghĩa
địa chỉ đầu	1	0	1B	Phân vùng tích cực? 80h nếu đúng; 0: Data
	2	1	1B	Số hiệu mặt đĩa đầu của phân vùng
	3	2	1W	Số hiệu sector và cylinder đầu của phân vùng
đ/c cuối	4	4	1B	Mã nhận diện hệ thống. 05/0F: Partition mở rộng; 06:Big Dos; 07:NTFS; 0B: FAT32,...
	5	5	1B	Số hiệu đầu đọc cuối
	6	6	1W	Số hiệu sector và cylinder cuối của phân vùng. (<i>Số hiệu sector chỉ dùng 6 bit thấp</i>)
	7	8	1DW	Địa chỉ đầu, tính theo số hiệu sector
	8	12	1DW	Số sector trong phân vùng

60 / 108

Notes

00 01 01 00	07 FE 3F F8	3F 00 00 00	7A 09 3D 00
80 00 01 F9	0B FE BF 30	B9 09 3D 00	38 7B 4C 00
00 00 81 EB	0F FE FF FF	2B 1D B7 00	72 13 7A 00
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
55 AA			

Giải mã

Boot	Vị trí đầu			Vị trí cuối			#sector	số sector
	Hdr	Cyl	Sec	HdR	Cyl	Sec		
No	1	0	1	254	248	63	63	4000122
Yes	0	249	1	254	560	63	4000185	5012280
No	0	747	1	254	1023	63	12000555	8000370
-	0	0	0	0	0	0	0	

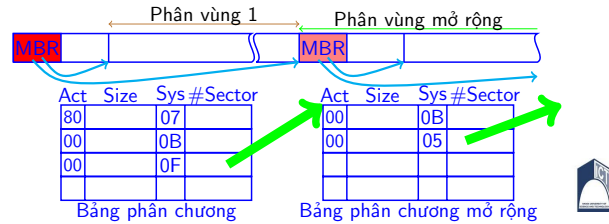
Notes

80 01 01 00	07 FE FF FF	3F 00 00 00	2C 92 00 02
00 00 C1 FF	0F FE FF FF	31 41 8A 03	0E D3 1D 01
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
55 AA			

Active	Begin			Sys	End			Relative Sector	Number Of Sector
	Hdr	Cyl	Sct		Hdr	Cyl	Sct		
YES	1	0	1	07	254	1023	63	63	33591852
NO	0	1023	3697	0F	254	1023	63	59392385	18731790
NO	0	0	0	00	0	0	0	0	0
NO	0	0	0	00	0	0	0	0	0

Notes

- Khi trường nhận diện có giá trị 05 hoặc 0F, partition tương ứng là partition mở rộng
- Partition mở rộng được tổ chức như một đĩa cứng vật lý
 - Sector đầu tiên là MBR, chứa thông tin về các phân vùng trong partition mở rộng này
 - Các phân tử trong partition mở rộng có thể là partition rộng
 - Cho phép tạo hơn 4 ổ đĩa logic



Notes

Ví dụ về bảng phân chương mở rộng 1

00	01	01	00	07	FF	FF	FF	3F	00	00	00	11	2F	F7	01
00	00	C1	FF	0F	EF	FF	FF	50	2F	F7	01	B0	23	B1	02
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
55	AA														

	Begin			Sys	End			Relative	Number
Active	Hdr	Cyl	Sct		Hdr	Cyl	Sct	Sector	Of Sector
YES	1	0	1	07	239	1023	63	63	32976657
NO	0	1023	1	0F	239	1023	63	32976720	45163440
NO	0	0	0	00	0	0	0	0	0
NO	0	0	0	00	0	0	0	0	0

Notes

Ví dụ về bảng phân chương mở rộng 2

Extended Partition (Sector number 32976720)...																
00	01	C1	FF	06	EF	FF	FF	3F	00	00	00	51	E8	76	01	
00	00	C1	FF	05	EF	FF	FF	90	E8	76	01	20	3B	3A	01	
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
55	AA															

		Begin			End			Relative	Number
Active		Sys							Of
	Hdr	Cyl	Sct		Hdr	Cyl	Sct	Sector	Sector
NO	1	1023	1	06	239	1023	63	63	24569937
NO	0	1023	1	05	239	1023	63	24570000	20593440
NO	0	0	0	00	0	0	0	0	0
NO	0	0	0	00	0	0	0	0	0

Notes

Ví dụ về bảng phân chương mở rộng 3

```

Extended Partition (Sector number 57546720)...
00 01 C1 FF 0B EF FF FF 3F 00 00 00 E1 3A 3A 01
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
55 AA

```

Active	Begin			End			Relative Sector	Number Of Sector
	Hdr	Cyl	Sct	Hdr	Cyl	Sct		
NO	1	1023	1	0B	239	1023	63	20593377
NO	0	0	0	00	0	0	0	0
NO	0	0	0	00	0	0	0	0
NO	0	0	0	00	0	0	0	0

Notes

- **LBA**: Logical Block Addressing
- **UEFI**: Unified Extensible Firmware Interfaze
- **BIOS** : Basic Input Output System
 - Địa chỉ một sector bị giới hạn 32 bit
⇒ Kích thước phân vùng bị giới hạn
- **GPT**
 - **LBA 0**: Protective Master Boot Record
 - **LBA 1**: Header GPT
 - **LBA 2→33**: Bảng phân chương, chiếm 32 sector
 - Gồm 128 phần tử, mỗi phần tử 128 byte
 - Sử dụng 8 byte để ghi địa chỉ (LBA) đầu và cuối của phân vùng
 - **LBA 34** → ...: Các phân vùng
 - **LBA -33** → -2 32 sector, backup cho bảng phân chương
 - **LBA -1** Sector cuối, backup cho Header GPT



Notes

- 1 Hệ thống file
- 2 Cài đặt hệ thống file
- 3 Tổ chức thông tin trên đĩa từ
- 4 **Hệ thống FAT**



Notes

- Tồn tại nhiều hệ thống file khác nhau
- Hệ thống FAT
 - FAT 12/ FAT16 dùng cho MSDOS
 - FAT32 dùng từ WIN98
 - 12/16/32: Số bit dùng để định danh cluster
 - Hệ thống NTFS
 - Sử dụng trong WINNT, WIN2000
 - Dùng 64 bit để xác định một cluster
 - Ưu việt hơn FAT trong bảo mật, mã hóa, nén dữ liệu,...
 - Hệ thống EXT3
 - Sử dụng trong Linux
 - Hệ thống CDFS
 - Hệ thống quản lý file trong CDROM
 - Hạn chế về độ sâu cây thư mục và kích thước tên
 - Hệ thốngs UDF
 - Phát triển từ CDFS cho DVD-ROM, hỗ trợ tên file dài



Notes

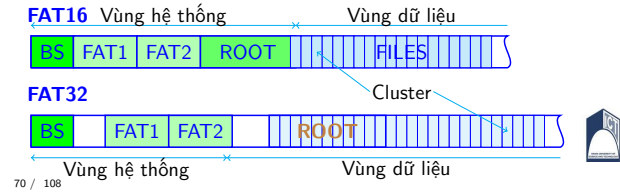
FAT12/16

- Số cluster lớn nhất FAT12: $2^{12} - 18$; FAT16 : $2^{16} - 18$
- K/thước max: FAT12: 32MB; FAT16: 2GB/4GB (32K/64K Cluster)

FAT32

- Chỉ dùng 28 bit \Rightarrow Số cluster lớn nhất $2^{28} - 18$
- K/thước max: 2TB/8GB/16TB (8KB/32KB/64KB Cluster)

Cấu trúc logic của hệ thống FAT

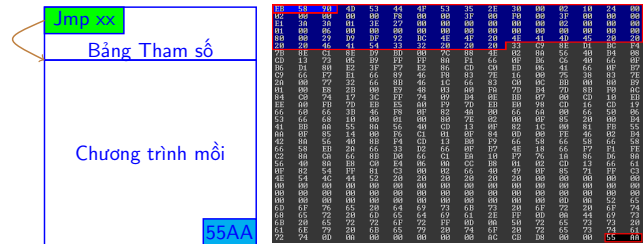


Notes

4 Hệ thống FAT

- Boot sector
- Bảng FAT (File Allocation Table)
- Thư mục gốc

Notes



- Sector đầu tiên của phân vùng
- Cấu trúc gồm 3 phần
 - Bảng tham số đĩa (BPB: Bios Parameter Block)
 - Chương trình môi (Boot strap loader)
 - Chữ ký hệ thống (luôn là 55AA)

Notes

Chương 4: Quản lý hệ thống file

4. Hệ thống FAT

4.1 Boot sector

Cấu trúc bảng tham số đĩa - Phần chung

Stt	Ofs	Kt	Giá trị mẫu	Ý nghĩa
1	0	3B	EB 3C 90	Nhảy đến đầu chương trình khởi
2	3	8B	MSDOS5.0	Tên hệ thống file đã format đĩa
3	11	1W	00 02	K/thước 1 sector, thường là 512
4	13	1B	40	Số sector cho một cluster (32K-Cluster)
5	14	1W	01 00	Số scts đứng trước FAT/Số scts để dành
6	16	1B	02	Số bảng FAT
7	17	1W	00 02	Số phần tử của ROOT. FAT32: 00 00
8	19	1W	00 00	Σ sector trên đĩa (< 32M) hoặc 0000
9	21	1B	F8	Khuôn dạng đĩa (F8:HD, F0: Đĩa1.44M)
10	22	1W	D1 09	Số sector cho một bảng FAT(209)
11	24	1W	3F 00	Số sector cho một rãnh (63)
12	26	1W	40 00	Số đầu đọc ghi (64)
13	28	1DW	3F 00 00 00	Số sector ẩn- Sectors trước volume
14	32	1DW	41 0C 34 00	Tổng số sector trên đĩa (3411009)

73 / 108

Notes

Chương 4: Quản lý hệ thống file

4. Hệ thống FAT

4.1 Boot sector

Cấu trúc bảng tham số đĩa - Phần dành cho FAT12/FAT16

Stt	Ofs	Kt	Giá trị mẫu	Ý nghĩa
15	36	1B	80h	Số hiệu ổ đĩa vật lý 0: ổ A; 80h: ổ C
16	37	1B	00	Để dành/Byte cao cho trường #ổ đĩa
17	38	1B	29h	Boot sector mở rộng 29h
18	39	1DW	D513 5B24	Volumn Serial number(245B-13D5)
19	43	11B	NO NAME	Volumn Label: nhãn đĩa (không dùng)
20	54	8B	FAT16	Để dành, thường là đoạn text miêu tả dạng FAT
21	62	-		Bootstrap loader

Ví dụ

EB 3C 90 4D 53 44 4F 53 35 2E 30 00 02 02 06 00
02 00 02 00 00 F8 F5 00 3F 00 FF 00 3F 00 00 00
C1 EB 01 00 00 00 29 A6 EA D4 70 4E 4F 20 4E 41
4D 45 20 20 20 20 46 41 54 31 36 20 20 20 33 C9

74 / 108

Notes

Chương 4: Quản lý hệ thống file

4. Hệ thống FAT

4.1 Boot sector

Ví dụ giả mã bảng tham số đĩa của FAT16

Đĩa 1.44MB (528 sectors) (512B)

Jmp +3C

EB 3C 90 4D 53 44 4F 53 35 2E 30 00 02 02 06 00
02 00 02 00 00 F8 F5 00 3F 00 FF 00 3F 00 00 00
C1 EB 01 00 00 00 29 A6 EA D4 70 4E 4F 20 4E 41
4D 45 20 20 20 20 46 41 54 31 36 20 20 20 33 C9

75 / 108

Notes

4 Hệ thống FAT

- Boot sector
- Bảng FAT (File Allocation Table)
- Thư mục gốc



Notes

FAT được sử dụng để quản lý các khối nhớ (*blocks/clusters*) trong vùng dữ liệu của bộ nhớ lưu trữ

- Khối nhớ đang sử dụng
 - Phân phối cho từng file/thư mục
- Khối nhớ tự do
- Khối nhớ bị hỏng

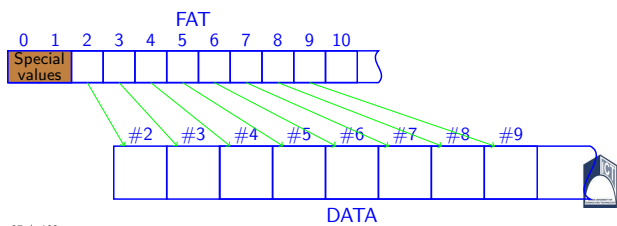
Thực hiện như thế nào ?



Notes

FAT gồm nhiều phần tử

- Mỗi phần tử có thể 12bit, 16bit, 32bit
- Mỗi phần tử ứng với 1 khối (*cluster*) trên vùng dữ liệu
 - 2 phần tử đầu (0,1) có ý nghĩa đặc biệt
 - Khuôn dạng đĩa, Bit shutdown, Bit diskerror
- Phần tử thứ 2 ứng với cluster đầu của phần Data



Notes

Mỗi phần tử của bảng FAT mang một giá trị đặc trưng cho tính chất của cluster tương ứng

FAT[(32)16]12	Ý nghĩa
[(0000)0]000h	Cluster tương ứng tự do
[(0000)0]001h	Giá trị không sử dụng
[(0000)0]002h →[(0FFF)F]FEFh	Cluster đang được sử dụng. Giá trị đóng vai trò con trỏ, trỏ tới cluster tiếp theo của file
[(0FFF)F]FF0h →[(0FFF)F]FF6h	Các giá trị để dành, chưa được sử dụng
[(0FFF)F]FF7h	Đánh dấu cluster tương ứng bị hỏng
[(0FFF)F]FF8h→ →[(0FFF)F]FFFh	Cluster đang đc sử dụng và là cluster cuối cùng của file (<i>EOC:End Of Cluster chain</i>). Thực tế thường dùng giá trị [(0FFF)F]FFFh

Notes

Root entry

ABC	TXT	A		Time	Date	008	Size
-----	-----	---	--	------	------	-----	------

FAT	2	3	4	5	6	7		9	10	11	12	13	14	15	16	17	18	19
Special values	03	04	05	FFF	00	00	09	10	12	FF7	15	00	00	16	FFF	00	21	FFF

DATA

#2	#3	#4	#5	#6	#7	#8	#9	#10	#11

#12	#13	#14	#15	#16	#17	#18	#19	#20	#21

Notes

```
#include <windows.h>
#include <stdio.h>
int main(int argc, char *argv[]){
    HANDLE hDisk;
    BYTE Buf[512];
    DWORD FAT[128];
    WORD FATAddr;    DWORD bytread, i;
    hDisk = CreateFile("\\\\.\\F:", GENERIC_READ,
        FILE_SHARE_READ | FILE_SHARE_WRITE,
        NULL, OPEN_EXISTING, 0, NULL);
    ReadFile(hDisk, Buf, 512, &bytread, NULL);
    memcpy(&FATAddr, &Buf[14], 2); //Offset 14 Sector trước FAT
    SetFilePointer(hDisk, FATAddr * 512, NULL, FILE_BEGIN);
    ReadFile(hDisk, FAT, 512, &bytread, NULL);
    for(i=0; i<128; i++) printf(" %08X ", FAT[i]);
    CloseHandle(hDisk);
    return 0;
}
```

Notes

Chương 4: Quản lý hệ thống file

4. Hệ thống FAT

4.3 Thư mục gốc

Cấu trúc một phần tử

Stt	Ofs	Size	Ý nghĩa
1	0	8B	Tên file
2	8	3B	Phần mở rộng
3	11	1B	Thuộc tính của file
4	12	10B	Không dùng với FAT12/FAT16. Sử dụng với FAT32
4.1	12	1B	Dễ dành
4.2	13	1B	Thời điểm tạo file, theo đơn vị 10ms
4.3	14	1W	Thời điểm tạo file (<i>giờ - phút - giây</i>)
4.4	16	1W	Ngày tạo file (<i>tạo bởi ứng dụng hoặc bởi copy sang</i>)
4.5	18	1W	Ngày truy nhập cuối
4.6	20	1W	Số hiệu cluster bắt đầu của file(<i>FAT32: Phần cao</i>)
5	22	1W	Thời gian cập nhật cuối cùng
6	24	1W	Ngày cập nhật cuối (<i>không y/cầu sau ngày tạo file</i>)
7	26	1W	Số hiệu cluster bắt đầu của file (<i>FAT32: Phần thấp</i>)
8	28	1DW	Kích thước tính bằng byte

94 / 108

Notes

Chương 4: Quản lý hệ thống file

4. Hệ thống FAT

4.3 Thư mục gốc

Cấu trúc một phần tử :Tên file

- Chuỗi ASCII chứa tên file. Các ký tự là chữ in
- Không chấp nhận khoảng trống ở giữa
 - Các câu lệnh copy, del,... không nhận biết tên có dấu trắng
- Nếu ít hơn 8 ký tự, được chèn các ký tự trống cho đủ 8
- Ký tự đầu có thể mang ý nghĩa đặc biệt
 - 00h: Phần tử đầu tiên của phần chưa dùng đến
 - E5h (ký tự "ø"): File tương ứng với phần tử này đã bị xóa.
 - 2Eh (ký tự "."): Đây là thư mục con
 - Trường số hiệu cluster bắt đầu chỉ đến chính nó
 - Cấu trúc như thư mục con giống như thư mục gốc: gồm các phần tử 32bytes
 - 2Eh2Eh (ký tự ".."): Đây là thư mục cha của thư mục hiện tại
 - Trường số hiệu cluster bắt đầu chỉ đến thư mục cha
 - Nếu cha là gốc, #cluster bắt đầu bằng zero (FAT12/16)
 - Thư mục con nằm trên phần Data, được quản lý như một file
 - ⇒ File của các bản ghi file
 - FAT12/16: Thư mục gốc ở vị trí xác định; FAT32: Thư mục gốc cũng nằm trong phần data

95 / 108

Notes

Chương 4: Quản lý hệ thống file

4. Hệ thống FAT

4.3 Thư mục gốc

Thư mục con

Name	A	Clst
DATA	IV	00
Sub-1	ID	
File-A	IA	
Sub-2	ID	

ROOT

DATA

Sub-1

File-F

Sub-11

File-C

File-D

File-B

File-A

Sub-2

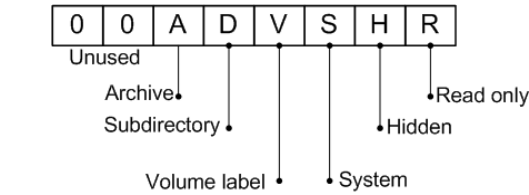
File-E

.	ID
..	ID
File-C	IA
File-D	IA

Sub-directory

96 / 108

Notes



Ví dụ: Byte thuộc tính **0Fh**:

0	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---

⇒ Có các thuộc tính **Volume label+System+Hidden+Read only**

Ghi chú: Giá trị byte thuộc tính **0x0F** không sử dụng trong *MS-DOS* ⇒ Dùng để đánh dấu là phần tử *Long File Name*



Notes



Ví dụ: 15 giờ 34 phút 45 giây

0	1	1	1	1	1	1	0	0	0	1	0	1	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Có giá trị : **7C56**



Notes



Ví dụ: 17 tháng 5 năm 2011

0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Có giá trị : **3EB1**



Notes

Chương 4: Quản lý hệ thống file
4. Hệ thống FAT
4.3 Thư mục gốc

Ví dụ: Nội dung của ROOT

F:\>DIR

Volume in drive F is DATA
Volume Serial Number is DC27-F353

Directory of F:\

05/05/2011	06:36 AM	<DIR>	Exemples
04/26/2011	11:35 AM		465 ReadBiosSector.c
05/04/2011	03:14 PM		2,749 READMBR.C
05/05/2011	06:52 PM	<DIR>	Temps
05/05/2011	06:35 AM		2,696,504 Bài giảng chương 4.pdf
		3 File(s)	2,699,718 bytes
		2 Dir(s)	14,247,424 bytes free

F:\>_

103 / 108

Notes

Chương 4: Quản lý hệ thống file
4. Hệ thống FAT
4.3 Thư mục gốc

Giải mã ROOT 1

DATA

Nhãn đĩa

44	41	54	41	20	20	20	20	20	20	20	08	00	00	00	00
00	00	00	00	00	00	64	25	A5	3E	00	00	00	00	00	00

#Cluster : 0 Size : 0

File đã bị xóa

05	44	48	20	20	20	20	20	50	44	46	20	18	00	93	34
A5	3E	A5	3E	00	00	6F	34	A5	3E	03	00	38	25	29	00

104 / 108

Notes

Chương 4: Quản lý hệ thống file
4. Hệ thống FAT
4.3 Thư mục gốc

Giải mã ROOT 2

File ReadMBR.C

Tên file: READMBR

Mở rộng: C

Lưu trữ

600ms

Create time 11h28m12s

52	45	41	44	40	42	52	20	43	20	20	20	00	3C	86	5B
A5	3E	A5	3E	00	00	CF	79	A4	3E	40	2E	BD	00	00	00

Create date 05/05/2011

Last access 05/05/2011

Modified time 15h14m30s

Modified date 04/05/2011

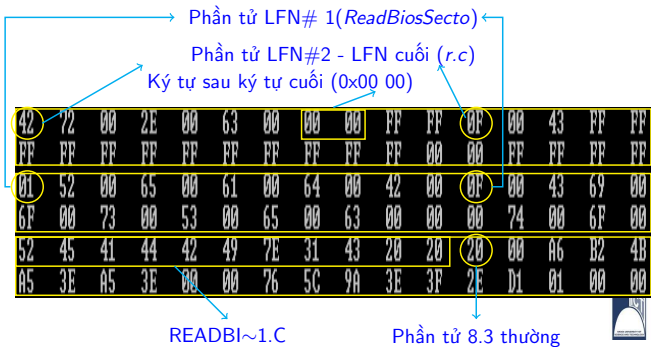
First cluster 11840

File size :2749

105 / 108

Notes

File: ReadBiosSector.c



Notes

- Viết chương trình Diskedit
 - Cho phép xem (và sửa chữa) từng sector của một đĩa cứng.
 - Các sector được hiển thị dưới cả 2 dạng: Hexa và ASCII
- Viết chương trình liệt kê tất cả các phân vùng của ổ đĩa cứng.
 - Nếu phân vùng sử dụng hệ thống file FAT32 hoặc NTFS, đưa ra các thông tin tương ứng
- Viết chương trình đưa ra nội dung của thư mục gốc của đĩa cứng sử dụng FAT32
 - Chỉ sử dụng thủ tục đọc sector trên đĩa
- Nghiên cứu cách tổ chức của các hệ thống file NTFS, EXT3
- Xây dựng một hệ thống file trên một đĩa ảo



Notes

- Hệ thống file**
 - Khái niệm file
 - Cấu trúc thư mục
- Cài đặt hệ thống file**
 - Cài đặt thư mục
 - Các phương pháp phân phối vùng lưu trữ
 - Quản lý vùng lưu trữ tự do
- Tổ chức thông tin trên đĩa từ**
 - Cấu trúc vật lý của đĩa
 - Cấu trúc logic của đĩa
- Hệ thống FAT**
 - Boot sector
 - Bảng FAT (File Allocation Table)
 - Thư mục gốc



Notes
