



BÀI GIẢNG MÔN

# HỆ ĐIỀU HÀNH

Giảng viên:

TS. Nguyễn Văn Thuỷ

Bộ môn:

Khoa học máy tính- Khoa CNTT1

# CHƯƠNG 2: HỆ THỐNG FILE

## ❖ Chương này trả lời câu hỏi:

- HĐH cần quản lý dữ liệu (data) lưu trữ lâu dài trong các ổ ngoài thế nào?
- HĐH cần cung cấp các APIs thế nào cho các chương trình ứng dụng?
- Mục tiêu:
  - Tạo mới, sửa, xoá, thay tên, tìm kiếm file → **nhANH NHẤT**
  - File và thư mục (directiory) APIs

1. Các khái niệm
2. Các phương pháp truy cập file
3. Các thao tác với file
4. Thư mục
5. Cấp phát không gian cho file
6. Quản lý không gian trống trên đĩa
7. Độ tin cậy của hệ thống file
8. Bảo mật cho hệ thống file
9. Hệ thống file FAT

- *File được định nghĩa như tập hợp các thông tin liên quan đến nhau được đặt tên và được lưu trữ trên bộ nhớ ngoài*
- Thuộc tính của file:
  - Tên file
  - Kiểu file
  - Kích thước file
  - Người tạo file, người sở hữu
  - Quyền truy cập file
  - Thời gian tạo file, sửa file, truy cập lần cuối
  - Vị trí file

- Đặt tên cho file:
  - Cho phép xác định file
  - Là thông tin người dùng thường sử dụng nhất khi làm việc với file
  - Quy tắc đặt tên cho file của một số HDH:

Hệ điều hành	Độ dài tối đa	Phân biệt chữ hoa, chữ thường	Cho phép sử dụng dấu cách	Các ký tự cấm
MS-DOS	8 cho tên file 3 cho mở rộng	không	không	Bắt đầu bằng chữ cái hoặc số Không được chứa các ký tự / \ [ ] : ;   = , ^ ? @
Windows NT FAT	255 ký tự cho cả tên file và đường dẫn	không	có	Bắt đầu bằng chữ cái hoặc số Không được chứa các ký tự / \ [ ] : ;   = , ^ ? @
Windows NT NTFS	255	không	có	Không được chứa các ký tự / \ < > *   :
Linux (EXT3)	256	Có	có (nếu tên file chứa trong ngoặc kép)	Không được chứa các ký tự ! @ # \$ % ^ & * ( ) [ ] { } ' " / \ : ; < > `

- Cấu trúc file:
  - Các thông tin trong file có thể rất khác nhau
  - => Cấu trúc của file cũng rất khác nhau và phụ thuộc vào thông tin chứa trong file
  - HDH có cần biết và hỗ trợ các kiểu cấu trúc file?
  - Hỗ trợ cấu trúc file ở mức HDH:
    - Ưu điểm:
      - Các thao tác với file sẽ dễ dàng hơn đối với người lập trình ứng dụng
      - HDH có thể kiểm soát được các thao tác với file
    - Nhược điểm:
      - Tăng kích thước hệ thống
      - Tính mềm dẻo của HDH bị giảm
  - Thực tế các HDH chỉ coi file là tập hợp các byte không cấu trúc

1. Các khái niệm
- 2. Các phương pháp truy cập file**
- 3. Các thao tác với file**



- Truy cập tuần tự:
  - Thông tin được đọc, ghi theo từng byte/ bản ghi lần lượt từ đầu file
  - Sử dụng 1 con trỏ để định vị vị trí hiện thời trong file
- Truy cập trực tiếp:
  - File được xem như các khối/ bản ghi được đánh số
  - Các khối có thể truy cập theo thứ tự bất kỳ
- Truy cập dựa trên chỉ số:
  - File chứa 1 chỉ số riêng: gồm các khóa và con trỏ chỉ tới các bản ghi trong file
  - Truy cập: tìm khóa tương ứng trong chỉ mục, sau đó theo con trỏ xác định bản ghi và truy cập trực tiếp tới nó

- Tạo file:
  - Tạo file trống chưa có data; được dành 1 chỗ trong thư mục
- Xóa file:
  - Giải phóng không gian mà dữ liệu của file chiếm
  - Giải phóng chỗ của file trong thư mục
- Mở file:
  - Thực hiện trước khi ghi và đọc file
  - Đọc các thuộc tính của file vào MEM để tăng tốc độ
- Đóng file:
  - Xóa các thông tin về file ra khỏi bảng trong MEM
- Ghi vào file
- Đọc file

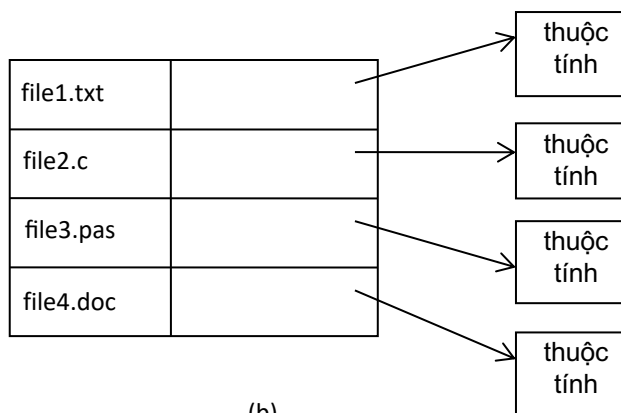
1. Các khái niệm
2. Các phương pháp truy cập file
3. Các thao tác với file
- 4. Thư mục**
5. Cấp phát không gian cho file
6. Quản lý không gian trống trên đĩa
7. Độ tin cậy của hệ thống file
8. Bảo mật cho hệ thống file
9. Hệ thống file FAT

- Không gian trên đĩa được chia thành các phần (partition/volume) gọi là đĩa logic
- Số lượng file lưu trữ trên đĩa rất lớn  $\Rightarrow$  phải tổ chức để dễ dàng quản lý, truy cập files
- Để quản lý file trên các đĩa logic, thông tin về file được lưu trong thư mục của đĩa
- Thư mục =  $\sum$  các khoản mục  $\sim$  files
- Khoản mục chứa các thông tin về file: tên, kích thước, vị trí, kiểu file,... hoặc con trỏ tới nơi lưu trữ thông tin này
- Coi thư mục như 1 bảng, mỗi dòng là khoản mục ứng với 1 file

- Các cách lưu thông tin về file trong thư mục:
  - Toàn bộ thuộc tính của file được lưu trong thư mục, file chỉ chứa data => kích thước khoản mục, thư mục lớn
  - Thư mục chỉ lưu thông tin tối thiểu cần thiết cho việc tìm kiếm vị trí file trên đĩa => kích thước giảm

file1.txt	Thuộc tính
file2.c	Thuộc tính
file3.pas	Thuộc tính
file4.doc	Thuộc tính

(a)



(b)

- Mở file:
  - HDH tìm trong thư mục khoản mục ứng với tên file cần mở
  - Đọc các thuộc tính và vị trí dữ liệu của file vào bảng chứa thông tin về các file đang mở
  - Nếu khoản mục trở tới CTDL khác chứa thuộc tính file, cấu trúc này sẽ được đọc vào bảng

# IV. THƯ MỤC

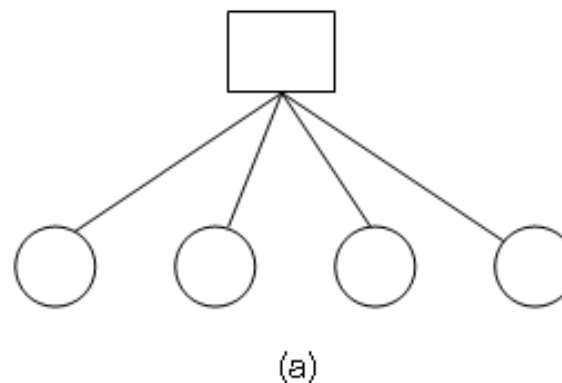
## 2. Các thao tác với thư mục



- **Tìm kiếm file:** cấu trúc thư mục phải cho phép tìm kiếm file theo tên file
- **Tạo file:** tạo khoản mục mới và thêm vào thư mục
- **Xóa file:** thông tin về file và khoản mục tương ứng bị xóa khỏi thư mục
- **Duyệt thư mục:** liệt kê các file trong thư mục và thông tin chứa trong khoản mục của file
- **Đổi tên file:** chỉ cần thực hiện với thư mục chứ không liên quan đến dữ liệu của file

# IV. THƯ MỤC

## 3. Cấu trúc hệ thống thư mục

- Thư mục 1 mức:
  - Đơn giản nhất
  - Chỉ có 1 thư mục duy nhất và tất cả các file được giữ trong thư mục này
  - Khó chọn tên cho file
  - Tìm kiếm file khó



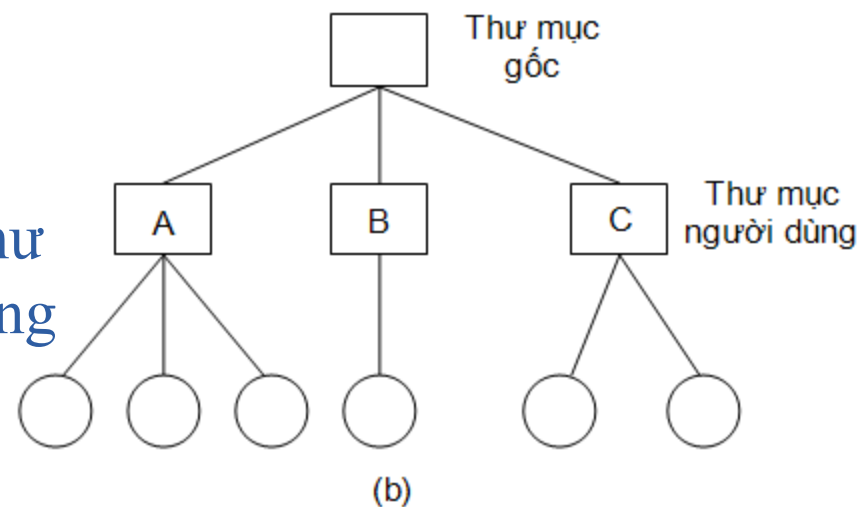
 = Thư mục       = File



# IV. THƯ MỤC

## 3. Cấu trúc hệ thống thư mục

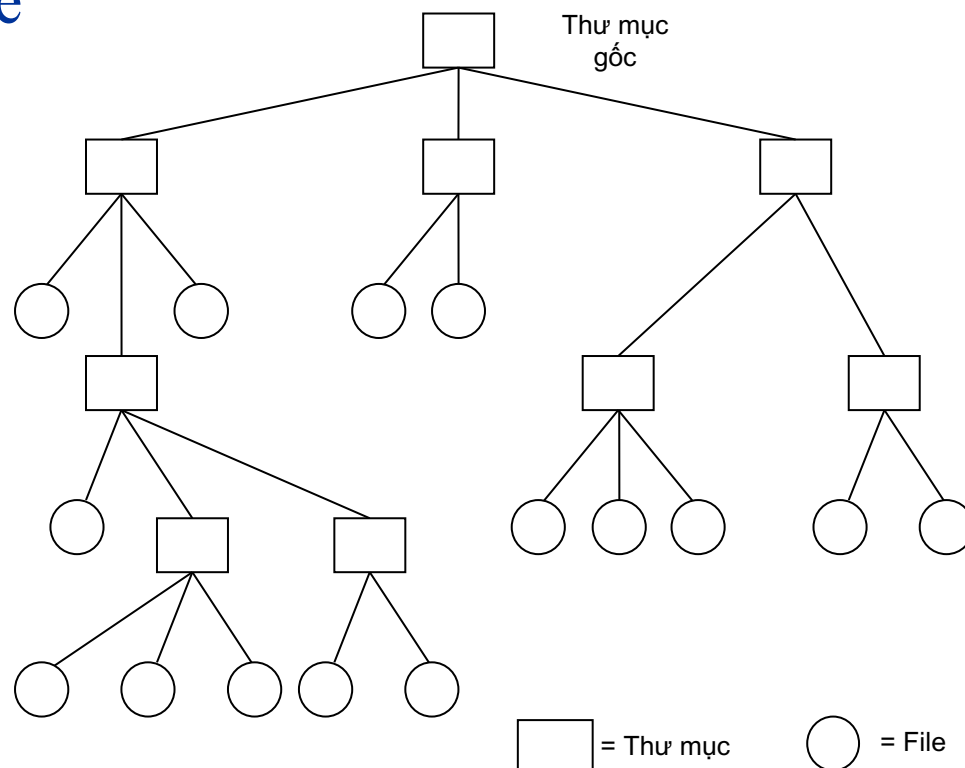
- Thư mục 2 mức:
  - Phân cho mỗi người dùng 1 thư mục riêng (UFD: User File Directory), chứa các file của mình
  - Khi người dùng truy cập file, file sẽ được tìm kiếm trong thư mục ứng với tên người đó
  - => các người dùng khác nhau có thể đặt tên file trùng nhau
  - Cô lập người dùng
  - Các file mà nhiều người dùng truy cập tới => chép vào từng thư mục của từng người dùng => lãng phí



# IV. THƯ MỤC

## 3. Cấu trúc hệ thống thư mục

- Thư mục cấu trúc cây:
  - Thư mục con có thể chứa các thư mục con khác và các files
  - Hệ thống thư mục được biểu diễn phân cấp như 1 cây: cành là thư mục, lá là file

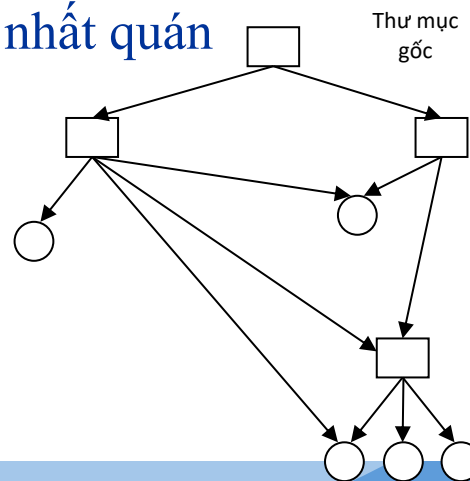


# IV. THƯ MỤC

## 3. Cấu trúc hệ thống thư mục

- Thư mục cấu trúc cây (tt):
  - Phân biệt khoản mục file và khoản mục của thư mục con: thêm bit đặc biệt trong khoản mục
    - 1: khoản mục của thư mục mức dưới
    - 0: khoản mục của file
  - Tại mỗi thời điểm, người dùng làm việc với thư mục hiện thời (current directory)
  - Tổ chức cây thư mục cho từng đĩa:
    - Trong hệ thống file như FAT của DOS, cây thư mục được xây cho từng đĩa. Hệ thống thư mục được coi là rừng, mỗi cây trên 1 đĩa
    - Linux: toàn hệ thống chỉ gồm 1 cây thư mục

- Thư mục cấu trúc đồ thị không tuần hoàn (acyclic graph ):
  - Chia sẻ files và thư mục để có thể xuất hiện ở nhiều thư mục riêng khác nhau
  - Mở rộng của cấu trúc cây: lá và cành có thể đồng thời thuộc về những cành khác nhau
  - Triển khai:
    - Sử dụng liên kết: con trỏ tới thư mục hoặc file khác
    - Tạo bản sao của file và thư mục cần chia sẻ và chứa vào các thư mục khác nhau => phải đảm bảo tính đồng bộ và nhất quán
  - Mềm dẻo nhưng phức tạp



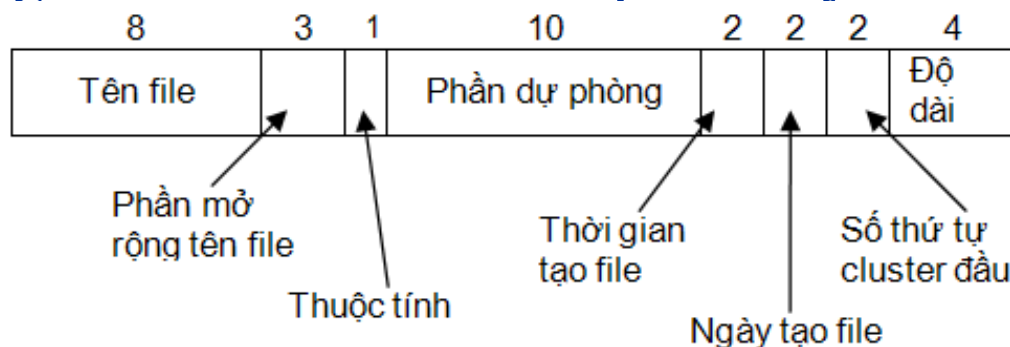
- Mô tả vị trí của file trong thư mục
- Đường dẫn tuyệt đối:
  - Đường dẫn từ gốc của cây thư mục, đi qua các thư mục trung gian, dẫn tới file
  - C:\bc\bin\bc.exe
- Đường dẫn tương đối:
  - Tính từ thư mục hiện thời
  - Thêm 2 khoản mục đặc biệt trong thư mục: “.”, “..”

- Danh sách:
  - Tổ chức thư mục dưới dạng danh sách các khoản mục
  - Tìm kiếm khoản mục được thực hiện bằng cách duyệt lần lượt danh sách
  - Thêm file mới vào thư mục:
    - Duyệt cả thư mục để kiểm tra xem khoản mục với tên file như vậy đã có chưa
    - Khoản mục mới được thêm vào cuối danh sách hoặc 1 ô trong bảng
  - Mở file, xóa file
  - Tìm kiếm trong danh sách chậm
  - Cache thư mục trong MEM

- Cây nhị phân:
  - Tăng tốc độ tìm kiếm nhờ CTDL có hỗ trợ sắp xếp
  - Hệ thống file NTFS của WinNT
- Bảng băm (hash table):
  - Dùng hàm băm để tính vị trí của khoản mục trong thư mục theo tên file
  - Thời gian tìm kiếm nhanh
  - Hàm băm phụ thuộc vào kích thước của bảng băm => kích thước bảng cố định

## 5. Tổ chức bên trong của thư mục

- Tổ chức thư mục của DOS:
  - Mỗi đĩa logic có cây thư mục riêng, bắt đầu từ thư mục gốc ROOT
  - Thư mục gốc được đặt ở phần đầu của đĩa, ngay sau sector khởi động BOOT và bảng FAT
  - Thư mục gốc chứa files và các thư mục con
  - Thư mục con có thể chứa files và các thư mục cấp dưới nữa
  - Được tổ chức dưới dạng bảng: mỗi khoản mục chiếm 1 dòng trong bảng và có kích thước cố định 32 bytes

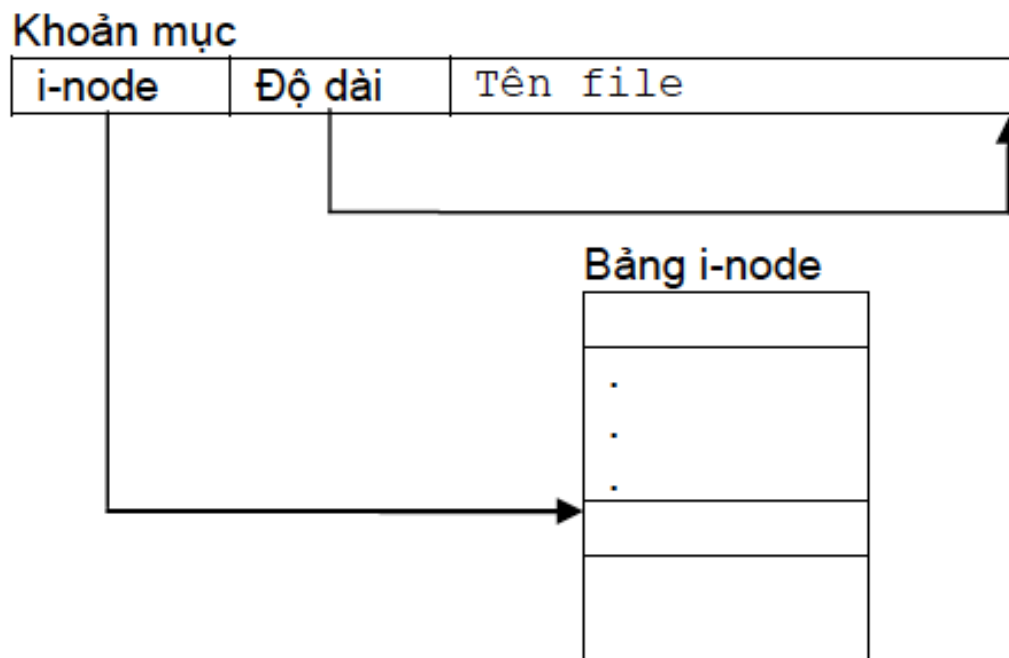




# IV. THƯ MỤC

## 5. Tổ chức bên trong của thư mục

- Tổ chức thư mục của Linux:
  - Thư mục hệ thống file Ext2 của Linux có cách tổ chức đơn giản
  - Khoản mục chứa tên file và địa chỉ I-node
  - Thông tin còn lại về các thuộc tính file và vị trí các khối dữ liệu được lưu trên I-node chứ không phải thư mục
  - Kích thước khoản mục phụ thuộc vào độ dài tên file
  - Phần đầu của khoản mục có trường cho biết kích thước khoản mục



1. Các khái niệm
2. Các phương pháp truy cập file
3. Các thao tác với file
4. Thư mục
- 5. Cấp phát không gian cho file**

## V. CẤP PHÁT KHÔNG GIAN CHO FILE

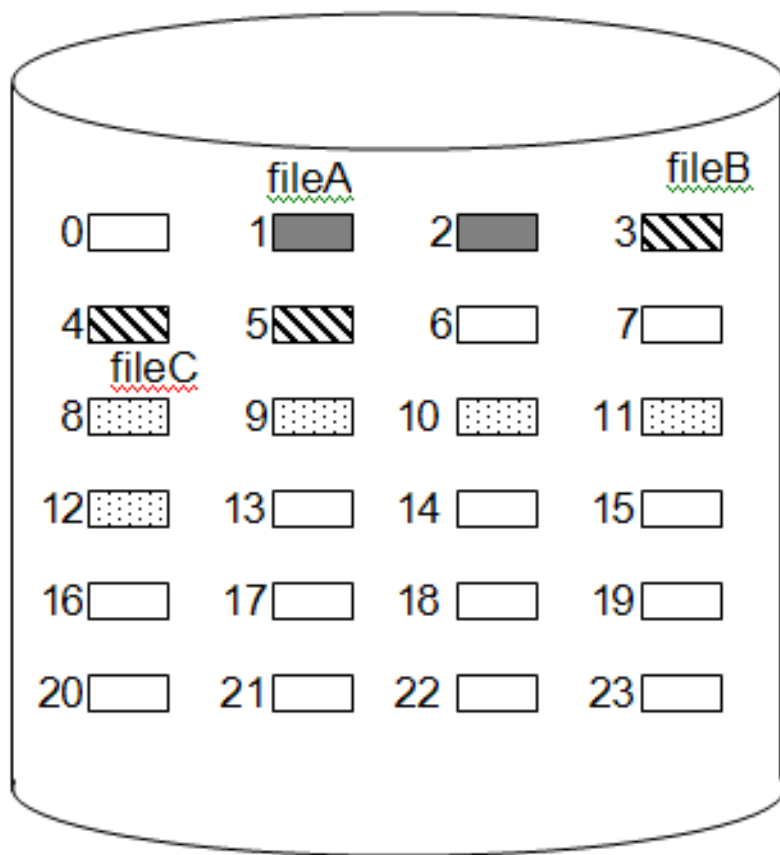
- Phép ánh xạ file: từ tên file có thể chỉ ra vị trí file trên đĩa
- Sơ bộ về tổ chức đĩa:
  - Thông tin được đọc/ghi theo từng khối sector
  - Nhóm các sector thành block hay cluster (khối)
- Trên đĩa: 1 file gồm 1 tập các khối. HDH chịu trách nhiệm cấp phát các khối cho file:
  - Không gian trên đĩa phải được cấp phát cho file
  - Cần theo dõi không gian trống sẵn sàng cho việc cấp phát
- Một số vấn đề:
  - Không gian tối đa yêu cầu cấp phát cho file 1 lần là bao nhiêu?
  - Không gian cấp phát cho file gọi là phần (portion). Kích thước phần ntn?
  - Theo dõi các phần được gán cho 1 file

## 1. Cấp phát các khối liên tiếp

- Được cấp phát 1 khoảng không gian gồm các khối liên tiếp trên đĩa
- Vị trí file trên đĩa được xác định bởi vị trí khối đầu tiên và độ dài (số khối) mà file đó chiếm
- Khi có yêu cầu cấp phát, HDH sẽ chọn 1 vùng trống có số lượng khối đủ cấp cho file đó
- Bảng cấp phát file chỉ cần 1 khoản mục cho 1 file, chỉ ra khối bắt đầu, và độ dài của file tính = khối

# V. CẤP PHÁT KHÔNG GIAN CHO FILE

## 1. Cấp phát các khối liên tiếp (tt)

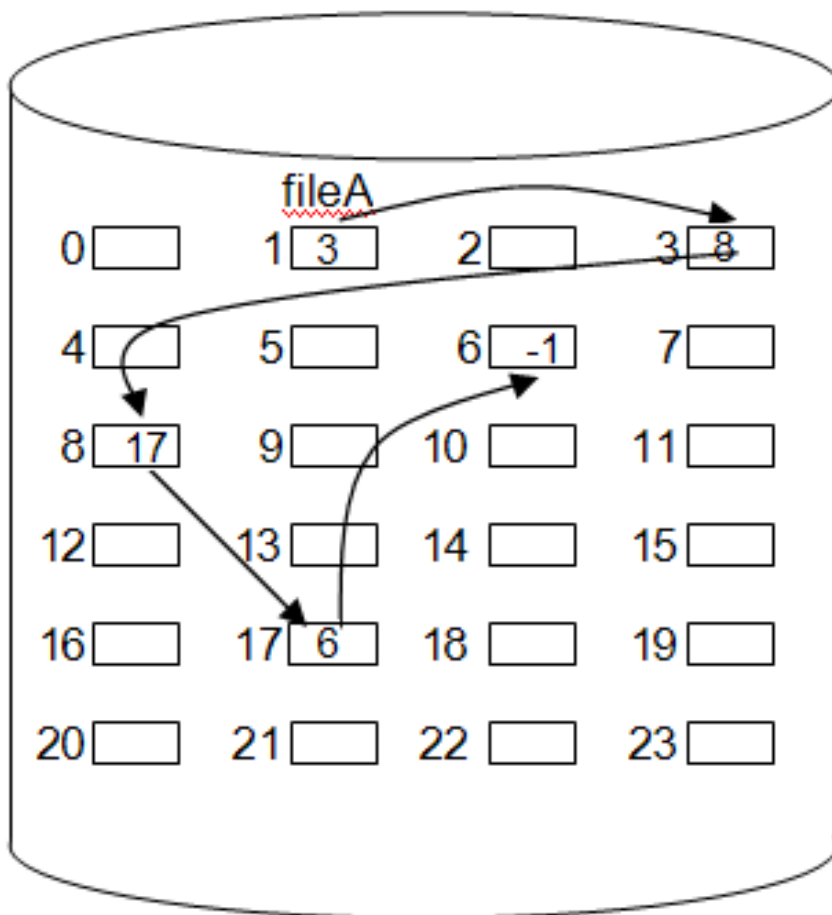


Tên file	Thư mục	
	Bắt đầu	Độ dài
fileA	1	2
fileB	3	3
fileB	8	5

## 1. Cấp phát các khối liên tiếp (tt)

- Ưu điểm:
  - Cho phép truy cập trực tiếp và tuần tự
  - Đơn giản, tốc độ cao
- Nhược điểm:
  - Phải biết trước kích thước file khi tạo
  - Khó tìm chỗ cho file
  - Gây phân mảnh ngoài:

- Các khối được kết nối với nhau thành danh sách kết nối; phần đầu mỗi khối chứa con trỏ trỏ tới khối tiếp theo
- Các khối thuộc về 1 file có thể nằm ở vị trí bất kì trên đĩa
- Khoản mục của thư mục chứa con trỏ tới khối đầu tiên của file
- Khi file được cấp thêm khối mới, khối đó được thêm vào cuối danh sách
- HDH đọc lần lượt từng khối và sử dụng con trỏ để xác định khối tiếp theo



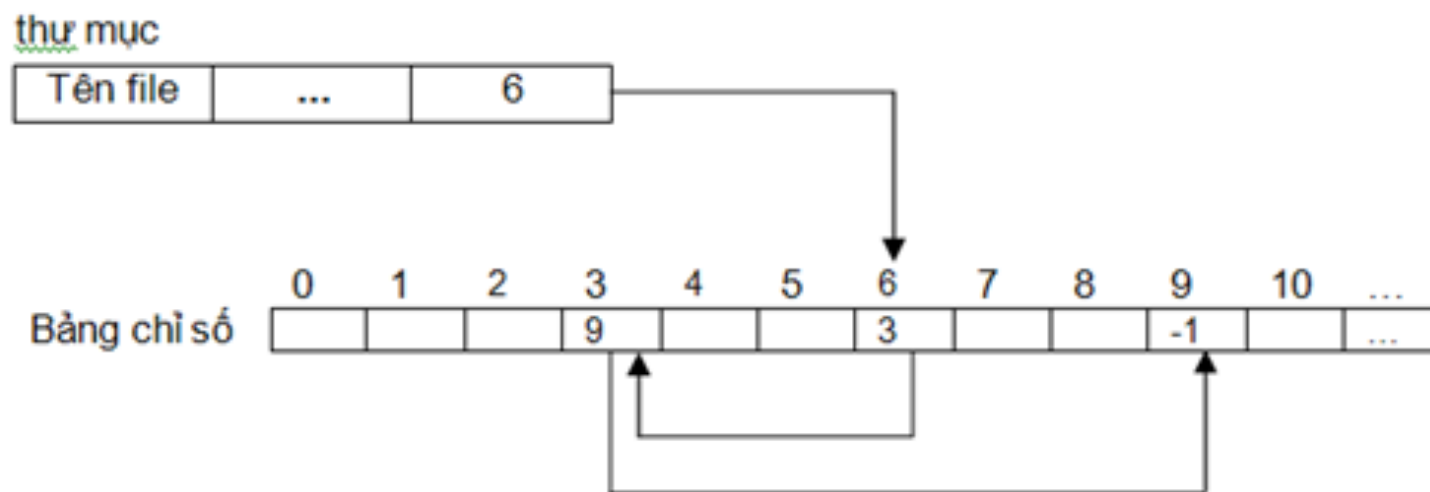
Thư mục	
Tên file	Bắt đầu
fileA	1



- Ưu điểm:
  - Không bị phân mảnh ngoài
  - Không yêu cầu biết trước kích thước file lúc tạo
  - Dễ tìm vị trí cho file, khoản mục đơn giản
- Nhược điểm:
  - Không hỗ trợ truy cập trực tiếp
  - Tốc độ truy cập không cao
  - Giảm độ tin cậy và tính toàn vẹn của hệ thống file

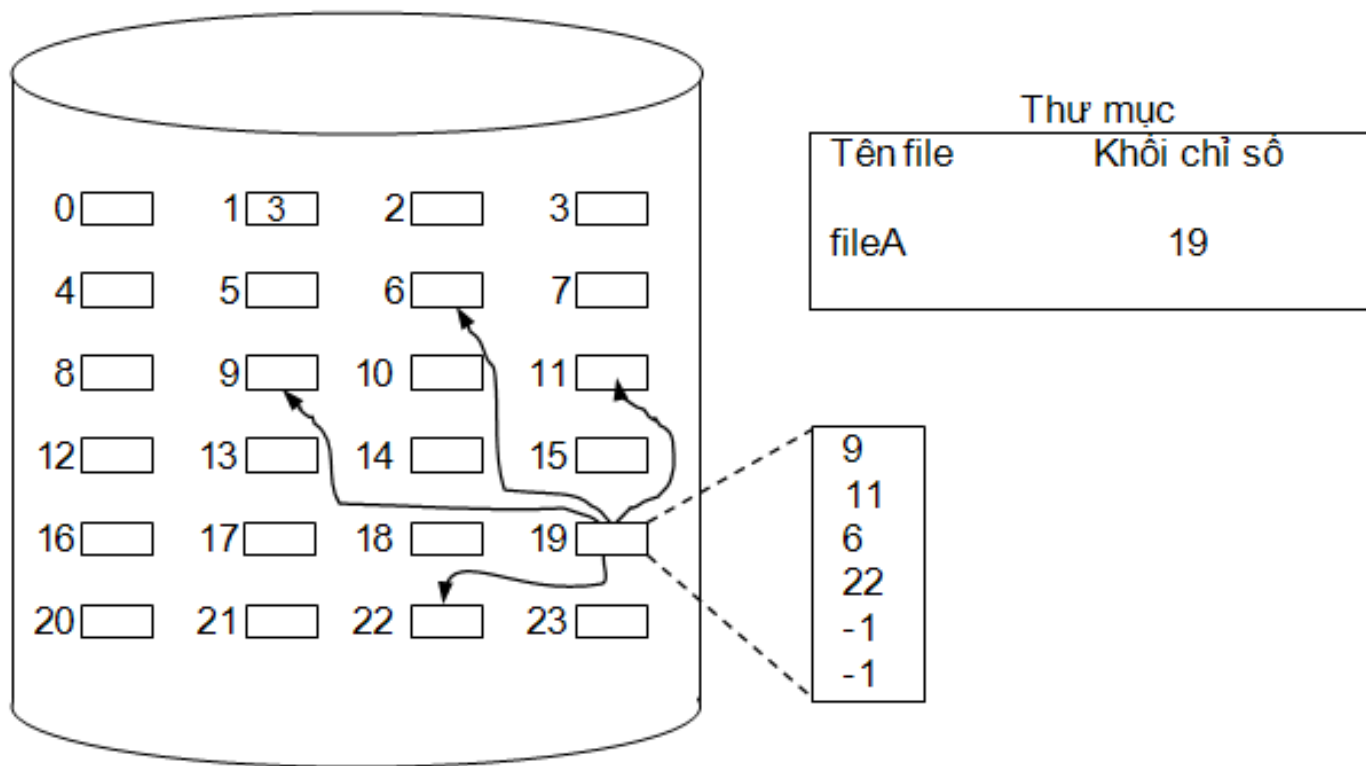
## 3. Sử dụng danh sách kết nối trên bảng chỉ số

- Bảng chỉ số: mỗi ô của bảng ứng với 1 khối của đĩa
- Con trỏ tới khối tiếp theo của file được chứa trong ô tương ứng của bảng
- Mỗi đĩa logic có 1 bảng chỉ số được lưu ở vị trí xác định
- Kích thước mỗi ô trên bảng phụ thuộc vào số lượng khối trên đĩa



- Cho phép tiến hành truy cập file trực tiếp: đi theo chuỗi con trỏ chứa trong bảng chỉ mục
- Bảng FAT (File Allocation Table): được lưu ở đầu mỗi đĩa logic sau sector khởi động
- FAT12, FAT16, FAT32: mỗi ô của bảng có kích thước 12, 16, 32 bit

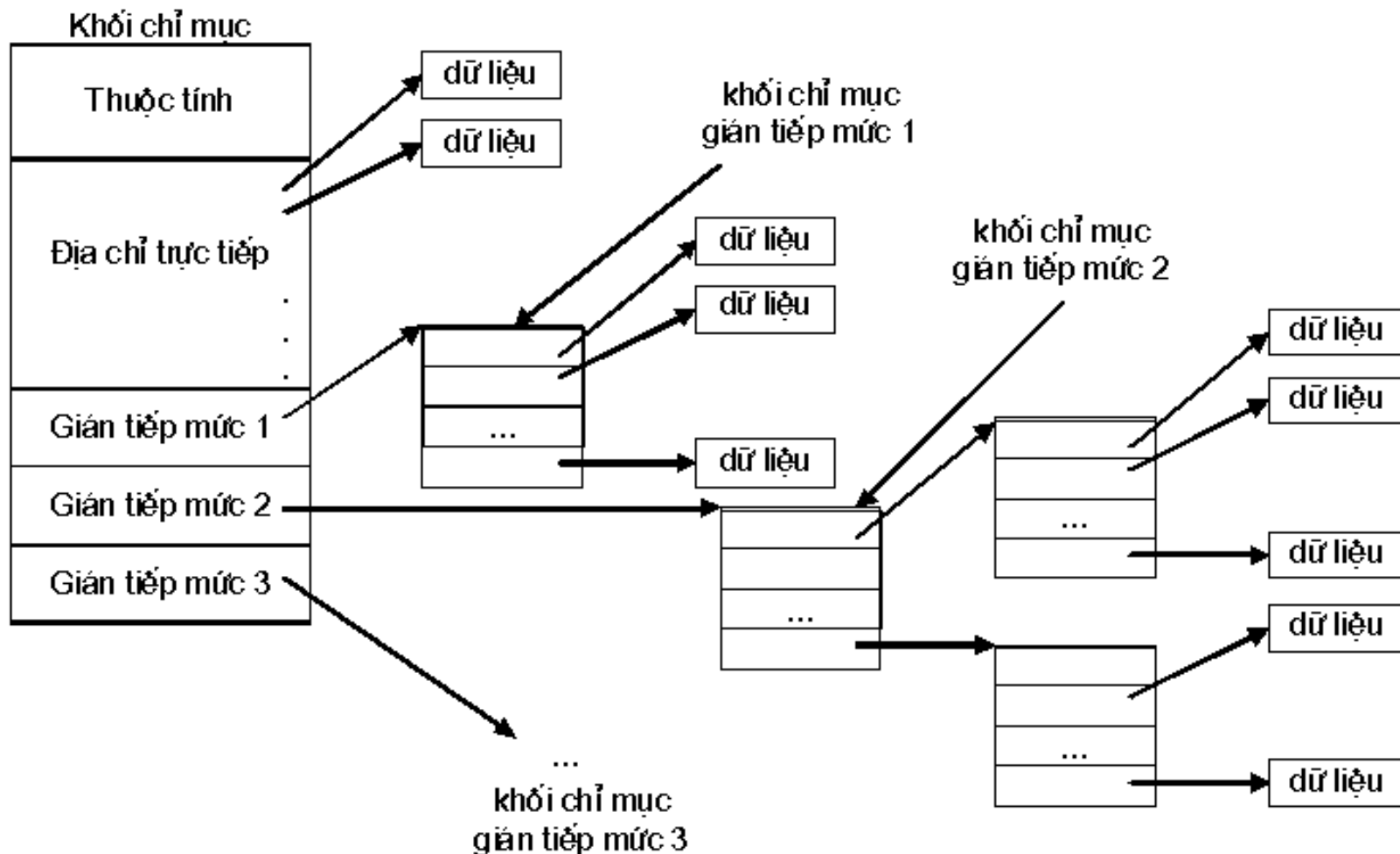
- Tất cả con trỏ tới các khối thuộc về 1 file được tập trung 1 chỗ → khối chỉ mục (I-node)
- Khối chứa thuộc tính của file và vị trí các khối của file trên đĩa lưu trong 1 mảng.
- Ô thứ i của mảng chứa con trỏ tới khối thứ i của file
- Khoản mục của file trong thư mục chứa con trỏ tới khối chỉ mục này



- Chọn kích thước I-node:
  - Nhỏ: tiết kiệm không gian nhưng không đủ con trỏ tới các khối nếu file lớn
  - Lớn: với file nhỏ chỉ chiếm 1 vài ô thì lãng phí
  - Giải pháp:
    - Thay đổi kích thước i-node = sử dụng danh sách kết nối
    - Sử dụng I-node có cấu trúc nhiều mức

## 4. Sử dụng khối chỉ mục (index block/ node)

- I-node cấu trúc nhiều mức:



- Ưu điểm:
  - Cho phép truy cập trực tiếp
  - Các khối thuộc 1 file không cần nằm liên tiếp nhau
- Nhược điểm:
  - Tốc độ truy cập file chậm



1. Các khái niệm
2. Các phương pháp truy cập file
3. Các thao tác với file
4. Thư mục
5. Cấp phát không gian cho file
- 6. Quản lý không gian trống trên đĩa**
7. Độ tin cậy của hệ thống file
8. Bảo mật cho hệ thống file
9. Hệ thống file FAT

- Kích thước khối:
  - Kích thước khối lớn:
    - Giảm kích thước bảng chỉ mục, tăng tốc độ đọc file;
    - Bị phân mảnh trong
  - Kích thước khối nhỏ:
    - Mỗi file chiếm nhiều khối nhớ, nằm rải rác trên đĩa
    - Thời gian đọc file lâu
  - Chọn kích thước khối tùy thuộc:
    - Kích thước đĩa: đĩa lớn, chọn kích thước khối lớn => thời gian truy cập nhanh, đơn giản hóa việc quản lý
    - Kích thước file: hệ thống sử dụng nhiều file lớn, kích thước tăng và ngược
  - Kích thước khối thường là lũy thừa 2 của sector và nằm trong khoảng từ 512B tới 32 KB

## 1. Bảng bit

- Vector bit là mảng 1 chiều
- Mỗi ô có kích thước 1 bit tương ứng với một khối trên đĩa
- Khối được cấp phát có bit tương ứng là 0, khối trống: 1 hoặc ngược lại
- Dễ tìm 1 hoặc nhóm các khối trống liên tiếp
- Với đĩa có kích thước lớn, đọc toàn bộ vector bit vào MEM có thể đòi hỏi khá nhiều không gian nhớ

- Các khối trống được liên kết với nhau thành danh sách
- Mỗi khối trống chứa con trỏ chỉ tới khối trống tiếp theo
- Địa chỉ khối trống đầu tiên được lưu ở vị trí đặc biệt trên đĩa và được HDH giữ trong MEM khi cần làm việc với các file
- Đòi hỏi truy cập lần lượt khi cần duyệt danh sách này
- HDH có thể cấp phát ngay các khối ở đầu danh sách

- Các khối nằm liền nhau thường được cấp phát và giải phóng đồng thời
- Lưu vị trí khối trồng đầu tiên của vùng các khối trồng liên tiếp và số lượng các khối trồng nằm liền sau đó
- Thông tin trên được lưu vào danh sách riêng

1. Các khái niệm
2. Các phương pháp truy cập file
3. Các thao tác với file
4. Thư mục
5. Cấp phát không gian cho file
6. Quản lý không gian trống trên đĩa
7. **Độ tin cậy của hệ thống file**
8. Bảo mật cho hệ thống file
9. Hệ thống file FAT

- Phát hiện và loại trừ khối hỏng
  - Phương pháp 1:
    - Một sector trên đĩa được dành riêng chứa danh sách các khối hỏng
    - Một số khối không hỏng được dành riêng để dự trữ
    - Các khối hỏng sẽ được thay thế bởi các khối dự trữ bằng cách thay thế địa chỉ truy cập tới khối hỏng thành truy cập tới khối dự trữ
  - Phương pháp 2:
    - Tập trung tất cả các khối hỏng thành 1 file
    - => được coi như đã cấp phát và không được sử dụng nữa

## 2. Sao dự phòng

- Tạo ra một bản sao của đĩa trên một vật mang khác
- Sao lưu toàn bộ (full backup):
  - Ghi toàn bộ thông tin trên đĩa ra vật mang tin khác
  - Chắc chắn nhưng tốn nhiều thời gian
- Sao lưu tăng dần (incremental backup):
  - Được sử dụng sau khi đã tiến hành full backup ít nhất 1 lần
  - Chỉ ghi lại các file đã bị thay đổi sau lần sao lưu cuối cùng
  - Hệ thống lưu trữ thông tin về các lần lưu trữ file
  - DOS: file thay đổi, archive bit = 1
- Kết hợp:
  - Full backup: hàng tuần/ tháng
  - Incremental backup: hàng ngày



- Hệ thống file chứa nhiều CTDL có mối liên kết => thông tin về liên kết bị hư hại, tính toàn vẹn của hệ thống bị phá vỡ
- Các khối không có mặt trong danh sách các khối trống, đồng thời cũng không có mặt trong một file nào
- Một khối có thể vừa thuộc về một file nào đó vừa có mặt trong danh sách khối trống
- HDH có các chương trình kiểm tra tính toàn vẹn của hệ thống file, được chạy khi hệ thống khởi động, đặc biệt là sau sự cố
- Ví dụ, window dùng SCANDISK để kiểm tra tính toàn vẹn của ổ cứng.

- Ví dụ trong hệ UNIX:
  - Tạo hai số đếm cho mỗi khối:
    - Số đếm thứ nhất: số lần khối đó xuất hiện trong danh sách khối trống.
    - Số đếm thứ hai: số lần khối xuất hiện trong file
  - Tất cả số đếm được khởi tạo bằng 0
  - Duyệt danh sách khối trống và toàn bộ i-node của các file
    - Một khối xuất hiện trong danh sách khối trống, số đếm tương ứng thứ nhất được tăng một đơn vị
    - Nếu khối xuất hiện trong i-node của file, số đếm tương ứng thứ hai được tăng một đơn vị
  - Tổng 2 số đếm = 1

Số lần xuất hiện trong danh  
sách trống

Số thứ tự khối					
0	1	2	3	4	5
0	1	0	1	0	1

Số lần xuất hiện trong file

Số thứ tự khối					
0	1	2	3	4	5
0	0	1	1	3	2

- Giao tác (transaction) là một tập hợp các thao tác cần phải được thực hiện trọn vẹn cùng với nhau (cập nhật ở bảng file trống, và bảng i-node)
- Với hệ thống file: mỗi giao tác sẽ bao gồm những thao tác thay đổi liên kết cần thực hiện cùng nhau
- Toàn bộ trạng thái hệ thống file được ghi lại trong file log
- Nếu giao tác không được thực hiện trọn vẹn, HDH sử dụng thông tin từ log để khôi phục hệ thống file về trạng thái không lỗi trước khi thực hiện giao tác

1. Các khái niệm
2. Các phương pháp truy cập file
3. Các thao tác với file
4. Thư mục
5. Cấp phát không gian cho file
6. Quản lý không gian trống trên đĩa
7. Độ tin cậy của hệ thống file
- 8. Bảo mật cho hệ thống file**
9. Hệ thống file FAT

- Ngăn cản việc truy cập trái phép các thông tin lưu trữ trong file và thư mục
- Hạn chế các thao tác truy cập tới file hoặc thư mục
- Dùng mật khẩu:
  - Người dùng phải nhớ nhiều mật khẩu
  - Mỗi khi thao tác với tài nguyên lại gõ mật khẩu

- Sử dụng danh sách quản lý truy cập ACL (Access Control List)
  - Mỗi file được gán danh sách đi kèm, chứa thông tin định danh người dùng và các quyền người đó được thực hiện với file
  - ACL thường được lưu trữ như thuộc tính của file/ thư mục
  - Thường được sử dụng cùng với cơ chế đăng nhập
  - Các quyền truy cập cơ bản:
    - Quyền đọc (r)
    - Quyền ghi, thay đổi (w)
    - Quyền xóa
    - Quyền thay đổi chủ file (change owner)

1. Các khái niệm
2. Các phương pháp truy cập file
3. Các thao tác với file
4. Thư mục
5. Cấp phát không gian cho file
6. Quản lý không gian trống trên đĩa
7. Độ tin cậy của hệ thống file
8. Bảo mật cho hệ thống file
9. **Hệ thống file FAT**

- File Allocation Table (FAT) → MS-DOS
- Sau đó được sử dụng ở Windows 3.0, 3.1, 95/98, ME
- Hiện nay, FAT là hệ thống file thông dụng: sử dụng trong các hệ điều hành -> quản lý thẻ nhớ, đĩa mềm, đĩa CD, và phương tiện trung gian trao đổi file giữa các HĐH khác.
- Nhiều hệ thống nhúng sử dụng FAT để quản lý file do sự đơn giản của FAT
- Nhược điểm: chậm, độ tin cậy-bảo mật ko cao
- 3 phiên bản: FAT12, FAT16, FAT32
- Chữ số chỉ kích thước ô bảng FAT tương ứng 12, 16 và 32 bit



# IX. HỆ THỐNG FILE FAT

## 1. ĐĨA LOGIC

- Đơn vị cấp phát không gian trên đĩa (khối logic) là cluster (lũy thừa 2 của số lượng sector)

Boot sector và các khối dự phòng	Bảng FAT1	Bảng FAT2	Thư mục gốc (chỉ có trên FAT12 và FAT16)	Phần còn lại cho tới cuối đĩa chứa các file và thư mục của đĩa lô gic
----------------------------------	-----------	-----------	--	---

- Boot sector:
  - Sector đầu tiên của đĩa logic
  - Chứa thông tin mô tả cấu trúc đĩa logic: kích thước sector, cluster, kích thước bảng FAT
  - Chứa mã chương trình khởi động HĐH nếu đĩa logic là đĩa khởi động
- FAT: bảng chỉ số quản lý cấp phát khối cho file
- Thư mục gốc ROOT
- Vùng dữ liệu: chứa các file và thư mục của đĩa logic

### 32 Byte đầu tiên

Vị trí	Độ dài	Ý nghĩa
0	3	Lệnh Jump. Chỉ thị cho CPU bỏ qua phần thông tin và nhảy tới thực hiện phần mã khởi động của hệ điều hành nếu đây là đĩa khởi động hệ điều hành.
3	8	Tên hãng sản xuất, bổ sung dấu trắng ở cuối cho đủ 8B. Ví dụ: IBM 3.3, MSDOS5.0.v.v.
11	2	Bytes per sector. Kích thước sector tính bằng byte. Giá trị thường gặp là 512 đối với đĩa cứng. Đây cũng là vị trí bắt đầu của Khối Thông số BIOS (BIOS Parameter Block, viết tắt là BPB)
13	1	Sectors per cluster. Số sector trong một cluster, luôn là lũy thừa của 2 và không lớn hơn 128.
14	2	Reserved sectors. Số lượng sector dành cho vùng đầu đĩa đến trước FAT, bao gồm boot sector và các sector dự phòng.
16	1	Số lượng bảng FAT. Thường bằng 2.
17	2	Số khoản mục tối đa trong thư mục gốc ROOT. Chỉ sử dụng cho FAT12 và FAT16. Bằng 0 với FAT32.
19	2	Total sector. Tổng số sector trên đĩa. Nếu bằng không thì số lượng sector được ghi bằng 4 byte tại vị trí 0x20.
21	1	Mô tả loại đĩa. Ví dụ 0xF0 là đĩa mềm 3.5" hai mặt với 80 rãnh trên mỗi mặt, 0xF1 là đĩa cứng .v.v.
22	2	Sectors per FAT. Kích thước FAT tính bằng sector (đối với FAT12/16)
24	2	Sectors per track. Số sector trên một rãnh.
26	2	Number of heads. Số lượng đầu đọc (mặt đĩa được sử dụng)
28	4	Hidden sectors. Số lượng sector ẩn.
32	4	Total sector. Tổng số sector trên đĩa cho trường hợp có nhiều hơn 65535.

## Các byte tiếp theo với FAT12/16

Vị trí	Độ dài	Ý nghĩa
36	1	Số thứ tự vật lý của đĩa (0: đĩa mềm, 80h: đĩa cứng .v.v.)
37	1	Dự phòng
38	1	Dấu hiệu của phần mã môi. Chứa giá trị 0x29 (ký tự ' ') hoặc 0x28.
39	4	Số xê ri của đĩa (Volume Serial Number) được tạo lúc format đĩa
43	11	Volume Label. Nhân của đĩa được tạo khi format.
54	8	Tên hệ thống file FAT, ví dụ "FAT12 ", "FAT16 ".
62	448	Mã môi hệ điều hành, đây là phần chương trình tải hệ điều hành khi khởi động.
510	2	Dấu hiệu Boot sector (0x55 0xAA)

## Các byte tiếp theo với FAT32

Vị trí	Độ dài	Ý nghĩa
36	4	Sectors per FAT. Kích thước FAT tính bằng sector.
0x28	2	Cờ của FAT
0x2a	2	Version. Phiên bản.
0x2c	4	Số thứ tự của cluster đầu tiên của thư mục gốc root.
0x30	2	Số sector của Information Sector. Đây là phần nằm trong số sector dự phòng ngay sau boot sector.
0x32	2	Số thứ tự sector đầu tiên của bản sao của boot sector (nếu có)
0x34	12	Dự phòng
0x40	1	Số thứ tự vật lý của đĩa
0x41	1	Dự phòng
0x42	1	Dấu hiệu của phần mã mở rộng.
0x43	4	Số xê ri của đĩa (Volume Serial Number)
0x47	11	Volume Label
0x52	8	"FAT32 "
0x5a	420	Mã môi hệ điều hành
0x1FE	2	Dấu hiệu Boot sector (0x55 0xAA)

- Bảng chỉ số quản lý các khối và các file
- Quản lý các cluster trên đĩa và các file theo nguyên tắc:
  - Các khối thuộc cùng 1 file được liên kết thành 1 danh sách
  - Con trỏ được chứa trong ô tương ứng của bảng FAT
- Mỗi ô trong bảng FAT tương ứng với một cluster trên đĩa, chứa 1 trong các thông tin:
  - STT cluster tiếp theo trong danh sách các khối của file
  - Dấu hiệu kết thúc nếu ô tương ứng với cluster cuối cùng của file
  - Dấu hiệu đánh dấu cluster hỏng, không được sử dụng
  - Dấu hiệu đánh dấu cluster dự phòng
  - Bằng 0 nếu cluster trống, chưa cấp phát cho file nào

# IX. HỆ THỐNG FILE FAT

## BẢNG FAT

- Cluster đầu tiên của vùng dữ liệu được đánh STT là 2
- 2 ô đầu tiên của bảng FAT không dùng để quản lý cluster

FAT12	FAT16	FAT32	Ý nghĩa
0x000	0x0000	0x00000000	Cluster trống
0x001	0x0001	0x00000001	Cluster dự phòng, không được sử dụng
0x002–0xFEFF	0x0002–0xFFEF	0x00000002–0xFFFFFFFF	Cluster đã được cấp cho file. Chứa số thứ tự cluster tiếp theo của file.
0xFF0–0xFF6	0xFFFF0–0xFFFF6	0xFFFFFFFF0–0xFFFFFFFF6	Cluster dự phòng
0xFF7	0xFFFF7	0xFFFFFFFF7	Cluster hỏng.
0xFF8–0xFFFF	0xFFFF8–0xFFFFF	0xFFFFFFFF8–0xFFFFFFFFF	Cluster cuối cùng của file

## IX. HỆ THỐNG FILE FAT

### THƯ MỤC GỐC (ROOT)

- Mỗi thư mục được lưu trong bảng thư mục, thực chất là 1 file đặc biệt chứa các khoản mục của thư mục
- Mỗi khoản mục chứa thông tin về một file hoặc thư mục con của thư mục đang xét
- Với FAT12/16, thư mục trên cùng của đĩa được chứa trong 1 vùng đặc biệt gọi là thư mục gốc
- Các thư mục mức thấp hơn/ thư mục gốc của FAT32 được chứa trong vùng dữ liệu trên đĩa cùng với các file
- Mỗi thư mục gồm các khoản mục 32 byte xếp liền nhau

# IX. HỆ THỐNG FILE FAT

## THƯ MỤC GỐC (ROOT)

Vị trí	Độ dài	Mô tả
0	8	Tên file, thêm bằng dấu trắng ở cuối nếu ngắn hơn 8 byte
8	3	Phần mở rộng, thêm bằng dấu trắng ở cuối nếu ngắn hơn 3 byte
11	1	Byte thuộc tính của file. Các bit của byte này nếu bằng 1 sẽ có ý nghĩa như sau: Bit 0: file chỉ được đọc; Bit 1: file ẩn; Bit 2: file hệ thống; Bit 3: Volume label; Bit 4: thư mục con Bit 5: archive; Bit 6: thiết bị nhớ khác (dùng cho hệ điều hành); Bit 7: không sử dụng Byte thuộc tính bằng 0x0F là dấu hiệu của file tên dài.
12	1	Dự phòng
13	1	Thời gian tạo file tính theo đơn vị 10ms, giá trị từ 0 đến 199
14	2	Thời gian tạo file theo format sau: bit 15-11: giờ (0-23); bit 10-5: phút (0-59); bit 4-0: giây/2 (0-29)
16	2	Ngày tạo file theo format sau. Bit 15-9: năm (0-1980, 127 = 2107); bit 8-5: tháng (1-12); bit 4-0: ngày (1-31)
18	2	Ngày truy cập cuối, theo format như ngày tạo file
20	2	2 byte cao của số thứ tự cluster đầu tiên của file trong FAT32
22	2	Thời gian sửa file lần cuối, theo format thời gian tạo file
24	2	Ngày sửa file lần cuối, theo format như ngày tạo file
26	2	Số thứ tự cluster đầu tiên của file trong FAT12/16.
28	4	Kích thước file tính bằng byte. Bằng 0 với thư mục con



- `int absread(int drive, int nsects, long lsect, void *buffer)`
  - `drive`: ổ đĩa cần đọc, A: 0, B:1, C:2
  - `nsects`: số sector cần đọc
  - `lsect`: vị trí sector bắt đầu đọc
  - `buffer`: vùng nhớ lưu nội dung thông tin cần đọc

- Vị trí sector bắt đầu: reserved sector (byte 14, 15 trong bootsector)
- Tổng số sector cần đọc: sectors per FAT (byte 22, 23)

- Vị trí sector bắt đầu:  $\text{reserved sector} + \text{NoOfFATs} * \text{sectors per FAT}$
- Tổng số sector cần đọc:  $\text{NoOfRootEntries} * 32 / \text{BytesPerSector}$

1. Viết chương trình để hiển thị thông tin boot sector
2. Viết chương trình đọc FAT và in nội dung 100 ô fat đầu tiên lên màn hình
3. Viết chương trình đọc ROOT và in nội dung giống lệnh DIR
4. Cho 1 tên file thuộc ROOT. Viết chương trình tìm tất cả các cluster của file đó
5. Viết chương trình đếm số cluster trống trong 100 cluster đầu tiên của ổ đĩa

Cho 1 tên file thuộc ROOT. Viết chương trình tìm tất cả các cluster của file đó

1. Đọc Boot sector: lưu vào bs
2. Đọc ROOT: lưu trong ROOT
3. Đọc bảng FAT: lưu trong mảng FAT
4. Giả sử tên file đưa vào lưu trong chuỗi tenfile;
5. Kiểm tra xem file “tenfile” có nằm trong thư mục gốc??
  1. - duyệt lần lượt các khoản mục (i từ 0 tới bs->root\_entries) xem có khoản mục nào mà ROOT[i]->filename trùng với “tenfile” nhập vào hay không??
  2. Giả sử ROOT[i]->filename == “tenfile” => khoản mục i trong thư mục gốc quản lý “tenfile” đang xét

3. Cluster đầu tiên lưu trữ data cho file “tenfile” là  $ROOT[i] \rightarrow first\_cluster$

6. giả sử  $ROOT[i] \rightarrow first\_cluster = x$ ;

While ( $x < 0xffff8$ )

{

    cout << x;

$x = FAT[x]$ ;

}

Viết chương trình đếm số cluster trống của ổ đĩa

-đọc boot sector   absread (2, 1, 0, bs);

-Đọc FAT           absread(2, bs->sector\_per\_FAT, bs->reserved\_sector, FAT);

-Int dem=0;

-For (int I =2; i<(bs->sector\_per\_FAT \* bs->bytes\_per\_sector)/2; i++)

-If (FAT[i]==0) dem++;

## Hệ thống FAT16

Bài 1: Viết đoạn chương trình đếm số cluster trống trong 100 cluster đầu tiên của ổ đĩa D.

Bài 2: Viết đoạn chương trình in nội dung của 50 ô FAT đầu tiên của ổ đĩa C ra màn hình

Bài 3: Giả sử bảng FAT đã được đọc vào bộ nhớ tại địa chỉ `<< int *fat >>`. Giả sử một file được lưu trữ trên cluster đầu tiên là n. Viết đoạn chương trình liệt kê các cluster thuộc về file đó.



# IX. HỆ THỐNG FILE FAT

## BÀI TẬP THỰC HÀNH

```
struct bootsector
{
    char jump_instruction[3];
    char OEM_ID[8];
    int bytes_per_sector;
    char sector_per_cluster;
    int reserved_sectors;
    char number_of_fats;
    int root_entries;
    int small_sectors;
    char media_descriptor;
    int sectors_per_fat;
    int sectors_per_track;
    int number_of_heads;
    long hidden_sectors;
    long large_sectors;
    char physical_drive_number;
    char reserved;
    char extended_boot_signature;
    char volume_serial_number[4];
    char volume_label[11];
    char file_system_type[8];
    char bootstrap_code[448];
    char end_of_sector_marker[2];
};

bootsector *bs= new bootsector();

absread(2,1,0,bs);
```