MÔN HỆ ĐIỀU HÀNH

Chương 7

QUẢN LÝ HỆ THỐNG FILE

7.1 Giới thiệu hệ thống file
7.2 Giao tiếp sử dụng phân hệ quản lý file
7.3 Giao tiếp sử dụng phân hệ quản lý thư mục
7.4 Hiện thực file
7.5 Hiện thực thư mục
7.6 Quản lý các cluster chưa dùng
7.7 Các việc quản lý khác trên hệ thống file
7.8 Quản lý hệ thống file trên máy PC

Tài liệu tham khảo : chương 6, sách "Modern Operating Systems", Andrew S. Tanenbaum: , 2nd ed, Prentice Hall

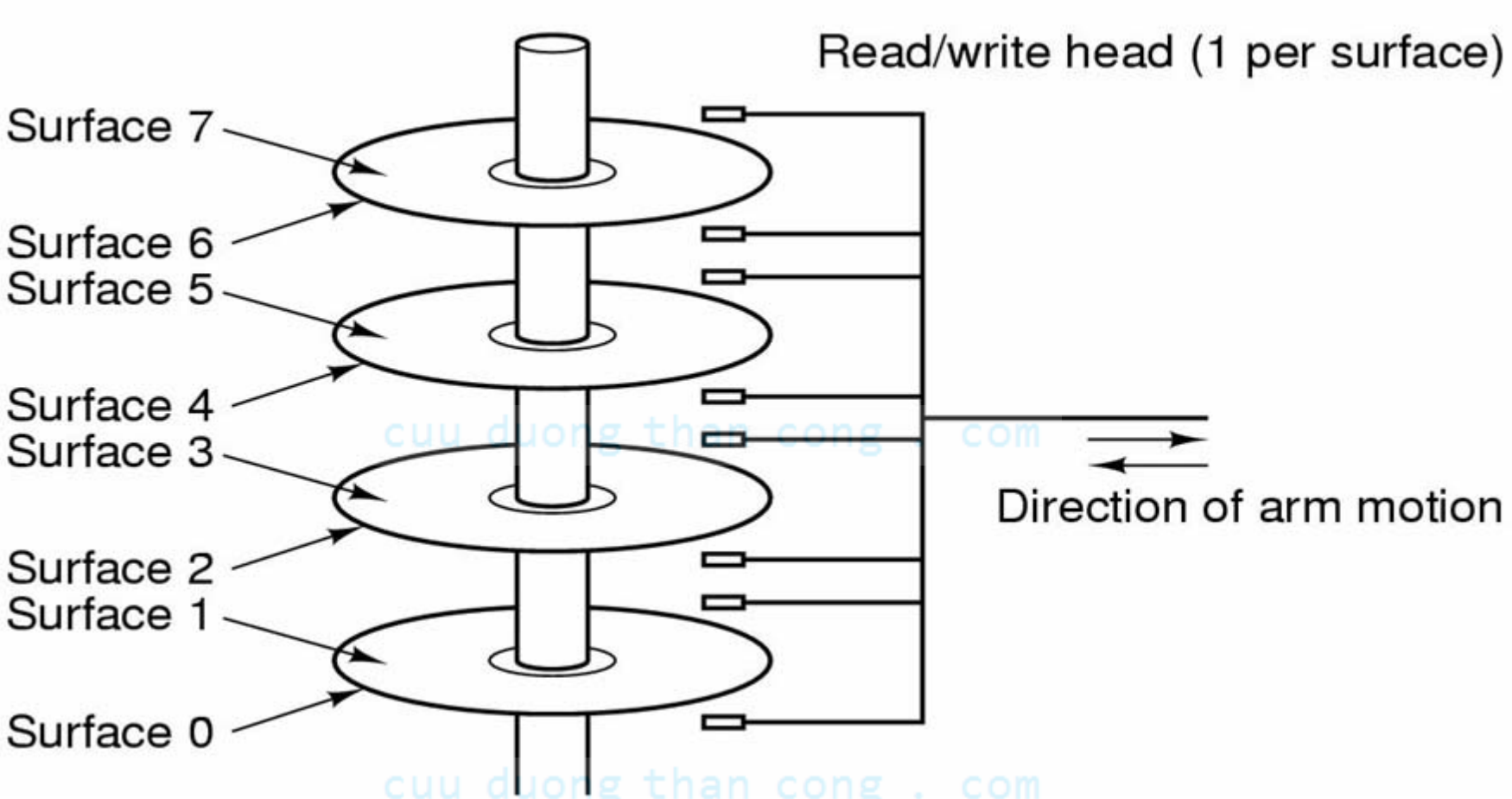
7.1 Giới thiệu hệ thống file

- Bộ nhớ nội của máy tính thường có dung lượng nhỏ, chỉ đủ chứa chương trình và dữ liệu đang được xử lý.
- Cần có thiết bị khác làm chỗ chứa các chương trình và dữ liệu đã/đang/sẽ xử lý, thiết bị này được gọi là bộ nhớ ngoài, nó cần có dung lượng rất lớn. Có rất nhiều kỹ thuật khác nhau để tạo ra bộ nhớ ngoài như đĩa cứng, băng, CDROM, flash ROM,...
- Để giúp user dùng các loại bộ nhớ ngoài dễ dàng, đồng nhất, HĐH sẽ trừu tượng hóa chúng thành 1 hệ thống cây phân cấp được gọi là hệ thống file. Chương này sẽ giới thiệu các kiến thức liên quan đến việc quản lý hệ thống file của HĐH.

Giới thiệu hệ thống file

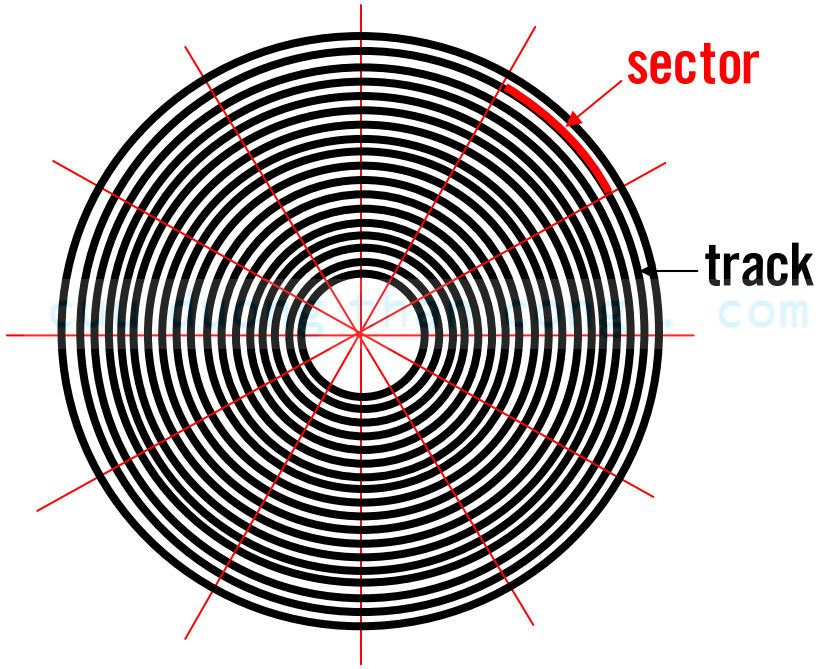
- Các tính chất thiết yếu của 1 hệ thống file :
  - nó cần có dung lượng rất lớn để chứa rất nhiều file chương trình và dữ liệu cần dùng trên máy tính.
  - nội dung được lưu trên hệ thống file phải tồn tại lâu dài, ngay cả khi process tạo ra nó đã chết.
  - nhiều process có thể truy xuất đồng thời vào từng phần tử trên hệ thống file.

Giới thiệu hệ thống file



Cấu trúc của một ổ đĩa cứng

Giới thiệu hệ thống file



Giới thiệu hệ thống file

Truy xuất đĩa vật lý :

- disk vật lý là không gian dữ liệu 3 chiều, mỗi disk = nhiều cylinder, mỗi cylinder gồm nhiều track (head – vòng tròn chứa tin) có cùng đường kính), mỗi track chứa nhiều cung chứa tin nhỏ được truy xuất độc lập nhau (sector). Sector là đơn vị truy xuất tin nhỏ nhất ở cấp vật lý (từ ngoài ta không thể truy xuất từng byte dữ liệu trên disk được).
- muốn truy xuất 1 sector, ta phải xác định tọa độ 3 chiều của nó (C,H,S) → rất khó tư duy.
- dữ liệu có nghĩa thường có độ lớn khác nhau và cần nhiều sector mới chứa đủ. Ở cấp vật lý này, người dùng phải tự quản lý danh sách các tọa độ sector 3 chiều được dùng để lưu trữ 1 dữ liệu có nghĩa nào đó → rất khó khăn và phiền hà → cần 1 cấp giao tiếp khác dễ sử dụng hơn.

Giới thiệu hệ thống file

Thời gian truy xuất 1 sector đĩa vật lý = $\sum$:

1. thời gian dời đầu đọc/ghi đến cyclinder cần truy xuất (thông qua từ 0 tới n xung tác động vào step motor). Đây là hoạt động tốn nhiều thời gian nhất.
2. thời gian chờ đĩa quay sector cần truy xuất về vị trí đầu đọc/ghi (trung bình là nữa vòng quay).
3. thời gian truy xuất dữ liệu của sector (tốn 1/n vòng quay).

Thời gian truy xuất 1 track/cylinder đĩa vật lý = $\sum$:

1. thời gian dời đầu đọc/ghi đến cyclinder cần truy xuất (thông qua từ 0 tới n xung tác động vào step motor). Đây là hoạt động tốn nhiều thời gian nhất.
2. thời gian truy xuất dữ liệu của track/cylinder (tốn 1 vòng quay).

So sánh những kết quả trên, ta thấy thời gian truy xuất 1 sector hay 1 track hay 1 cyclinder gồm nhiều track hầu như bằng nhau → nên truy xuất 1 lần 1 cyclinder hơn là 1 track hay 1 sector.

Giới thiệu hệ thống file

Truy xuất đĩa luận lý cấp 1 :

- disk luận lý cấp 1 là không gian dữ liệu 1 chiều, mỗi đĩa = danh sách nhiều đơn vị chứa tin có độ dài bằng nhau (cluster, block, sector luận lý). Độ dài của cluster cần độc lập với độ dài sector đĩa vật lý. Để dễ quản lý, thường ta chọn kích thước 1 cluster = 2^i sector.
- để truy xuất 1 cluster, ta chỉ cần xác định tọa độ 1 chiều (chỉ số) của nó (0 → n).
- dữ liệu có nghĩa thường có độ lớn khác nhau và cần nhiều cluster mới chứa đủ. Ở cấp luận lý này, người dùng phải tự quản lý danh sách các chỉ số cluster được dùng để lưu trữ 1 dữ liệu có nghĩa nào đó → vẫn còn khó khăn và phiền hà → cần 1 cấp giao tiếp khác dễ sử dụng hơn.

Giới thiệu hệ thống file

Truy xuất đĩa luận lý cấp 2 :

- disk luận lý cấp 2 là không gian dữ liệu 1 chiều, mỗi đĩa = danh sách nhiều đơn vị chứa tin có độ dài thay đổi tùy ý theo yêu cầu của người dùng (file). Mỗi file được nhận dạng bằng tên gợi nhớ thay vì bằng chỉ số. Để dễ quản lý, thường ta sẽ dùng n cluster đĩa để lưu nội dung của 1 file.
- muốn truy xuất 1 file, ta chỉ cần xác định tên gợi nhớ thay vì bằng chỉ số của nó.
- dữ liệu có nghĩa thường có độ lớn khác nhau, nhưng ở cấp độ này ta chỉ cần thay đổi kích thước file cho phù hợp → rất dễ quản lý thông tin chứa trên disk.
- thường disk có dung lượng rất lớn và chứa rất nhiều file dữ liệu nên việc đặt tên file dễ tranh chấp nhau, hơn nữa việc quản lý file sẽ gặp nhiều khó khăn (xóa/tạo mới, xác định file nào cần truy xuất,..) → cần 1 cấp giao tiếp khác dễ sử dụng hơn.

Giới thiệu hệ thống file

Truy xuất đĩa luận lý cấp 3 :

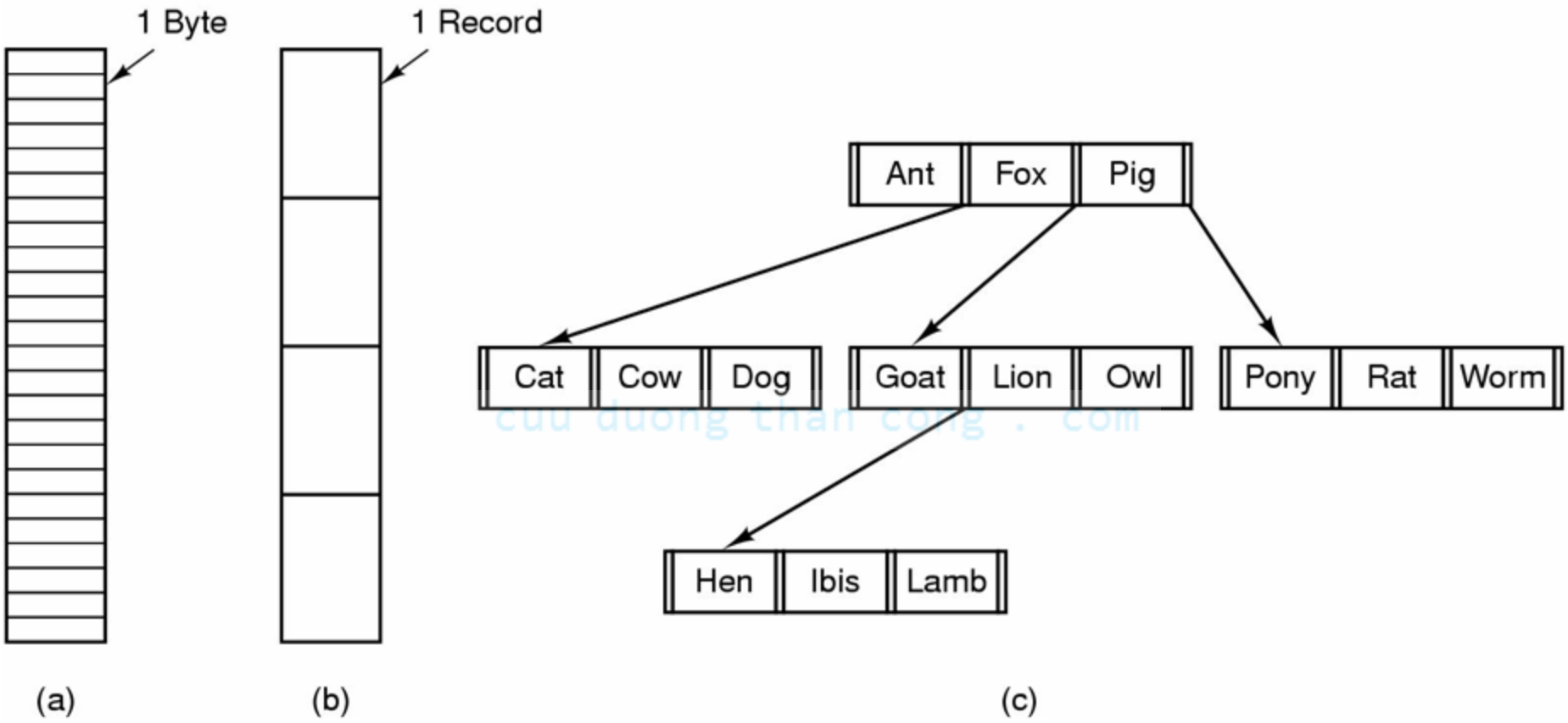
- disk luận lý cấp 3 là không gian dữ liệu dạng cây phân cấp, mỗi đĩa = thư mục gốc chứa nhiều phần tử con, mỗi phần tử con có thể là file hoặc thư mục...
- muốn truy xuất 1 file/thư mục, ta chỉ cần dùng pathname.
- có 2 loại pathname : tuyệt đối và tương đối. Pathname tuyệt đối sẽ xuất phát từ thư mục gốc để tìm phần tử cần truy xuất. Pathname tương đối xuất phát từ thư mục hiện hành (working) để tìm phần tử cần truy xuất. Tùy thuộc vào ngữ cảnh cụ thể mà loại pathname nào thích hợp hơn.

7.2 Giao tiếp sử dụng phân hệ quản lý file

1. Tên file : dùng để nhận dạng file cần truy xuất. Các định dạng được dùng phổ biến là 8.3 và chuỗi ký tự bất kỳ <= 256 ký tự.

Extension	Meaning
file.bak	Backup file
file.c	C source program
file.gif	Compuserve Graphical Interchange Format image
file.hlp	Help file
file.html	World Wide Web HyperText Markup Language document
file.jpg	Still picture encoded with the JPEG standard
file.mp3	Music encoded in MPEG layer 3 audio format
file.mpg	Movie encoded with the MPEG standard
file.o	Object file (compiler output, not yet linked)
file.pdf	Portable Document Format file
file.ps	PostScript file
file.tex	Input for the TEX formatting program
file.txt	General text file
file.zip	Compressed archive

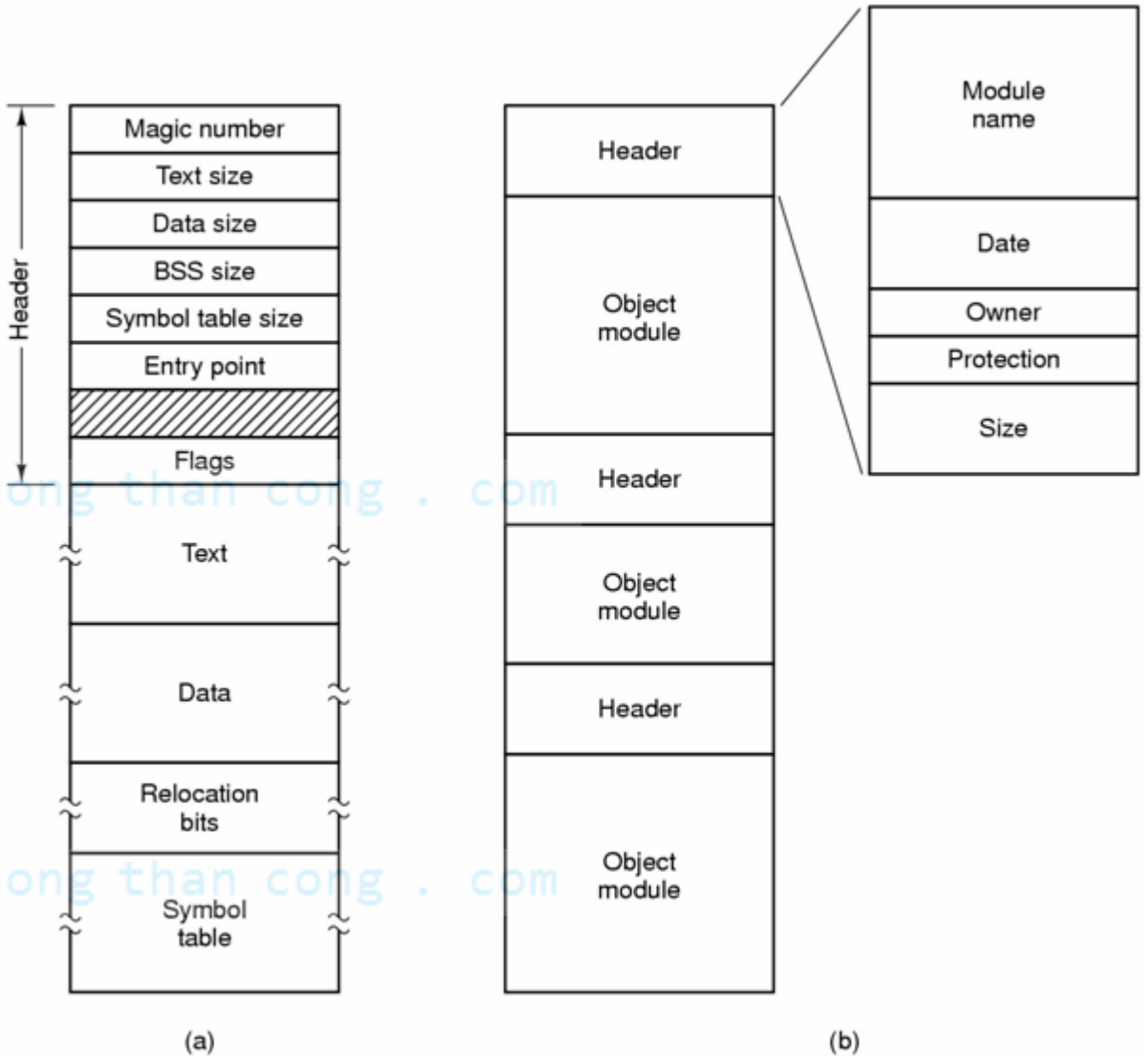
Giao tiếp sử dụng phân hệ quản lý file



2. Cấu trúc file (cấp HĐH) : có 3 cấu trúc phổ biến : a. danh sách các byte chưa có nghĩa b.danh sách các record ngữ nghĩa xác định c. cây các record ngữ nghĩa xác định.

Giao tiếp sử dụng phân hệ quản lý file

3. Cấu trúc ngữ nghĩa của file : do mỗi ứng dụng tự quy định, thường là 1 tập các record thông tin có liên quan nhau.



Giao tiếp sử dụng phân hệ quản lý file

4. Cơ chế truy xuất file :

- Truy xuất tuần tự (Sequential access) : đọc/ghi các byte/record từ đầu, không thể nhảy cóc.
- Truy xuất trực tiếp (Random access) : có thể dời đầu đọc/ghi đến bất kỳ vị trí nào trước khi đọc/ghi dữ liệu

Giao tiếp sử dụng phân hệ quản lý file

5. Thuộc tính file :
mỗi file có 1 tập các thuộc tính khác nhau để HĐH quản lý và kiểm soát việc truy xuất file của người dùng.

Attribute	Meaning
Protection	Who can access the file and in what way
Password	Password needed to access the file
Creator	ID of the person who created the file
Owner	Current owner
Read-only flag	0 for read/write; 1 for read only
Hidden flag	0 for normal; 1 for do not display in listings
System flag	0 for normal files; 1 for system file
Archive flag	0 for has been backed up; 1 for needs to be backed up
ASCII/binary flag	0 for ASCII file; 1 for binary file
Random access flag	0 for sequential access only; 1 for random access
Temporary flag	0 for normal; 1 for delete file on process exit
Lock flags	0 for unlocked; nonzero for locked
Record length	Number of bytes in a record
Key position	Offset of the key within each record
Key length	Number of bytes in the key field
Creation time	Date and time the file was created
Time of last access	Date and time the file was last accessed
Time of last change	Date and time the file has last changed
Current size	Number of bytes in the file
Maximum size	Number of bytes the file may grow to

Giao tiếp sử dụng phân hệ quản lý file

6. Các hàm API truy xuất file : HĐH cung cấp 1 tập các hàm chức năng phục vụ việc truy xuất file bởi ứng dụng.

6.1 Truy xuất nội dung file : gồm 3 bước

1. Mở/Tạo mới file :
   - CreateFile : tạo mới 1 file
   - OpenFile : mở/tạo mới 1 file để truy xuất
2. Truy xuất nội dung file :
   - Read : đọc n byte nội dung từ vị trí truy xuất hiện hành
   - Write : ghi n byte nội dung từ vị trí truy xuất hiện hành
   - Append : ghi thêm n byte vào đuôi file
   - Seek : dời vị trí truy xuất file về vị trí xác định
3. Đóng file :
   - Delete : xóa file đang tồn tại
   - Close : đóng file và không cho phép truy xuất nữa

6.2 Truy xuất các thuộc tính file :

Get attributes : đọc các thuộc tính của file

Set Attributes : thiết lập lại các thuộc tính file

Rename : đặt lại tên file

Giao tiếp sử dụng phân hệ quản lý file

7. Chương trình demo việc đọc/ghi file :

```
#include <fcntl.h>
#include <io.h>
#include <stdio.h>
//điểm nhập của chương trình
int main(int argc, char* argv[]) {
int fin, fout;
int flen;
char* buffer;
   //mở file để đọc vào
   if ((fin = open("c:\\s.exe",O_RDONLY|O_BINARY))< 0) {
      printf("Không thể mở file c:\\s.exe\n"); exit(1);
   }
   //mở file để ghi ra
   if ((fout = open("c:\\p.exe",O_WRONLY|O_BINARY|O_CREAT))< 0) {
      printf("Không thể mở file c:\\p.exe\n"); exit(1);
   }
```

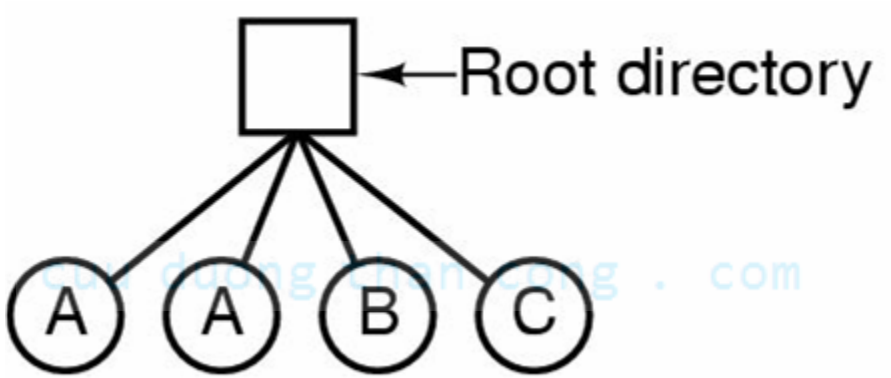
Giao tiếp sử dụng phân hệ quản lý file

```
   //xác định độ lớn của file
   flen = filelength (fin);
   //cấp phát động buffer chứa dữ liệu
   buffer = (char*) malloc(flen);
   if (buffer==0) {
      printf("Không thể cấp phát buffer chứa file\n");
      exit(1);
   }
   //đọc toàn bộ file vào buffer
   read(fin,buffer,flen);
   //ghi toàn bộ buffer ra file
   write(fout,buffer,flen);
   //đóng các file lại
   close (fin);
   close(fout);
}
```

7.3 Giao tiếp sử dụng phân hệ quản lý thư mục

1. Hệ thống file dùng thư mục 1 cấp :

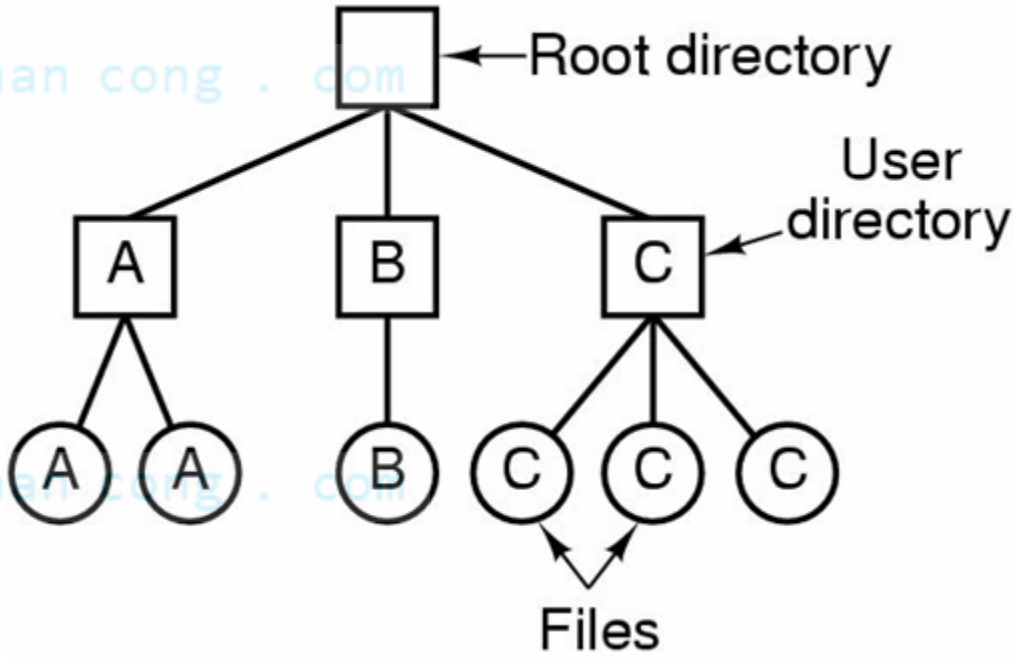
- Thiết bị chứa tin được trừu tượng thành 1 cây với 1 thư mục gốc.
- Tất cả các file đều được chứa trực tiếp vào thư mục gốc, không có thư mục con nào cả.
- Dạng thư mục 1 cấp rất thích hợp cho các thiết bị chứa tin có dung lượng nhỏ như đĩa mềm...



Giao tiếp sử dụng phân hệ quản lý thư mục

2. Hệ thống file dùng thư mục 2 cấp :

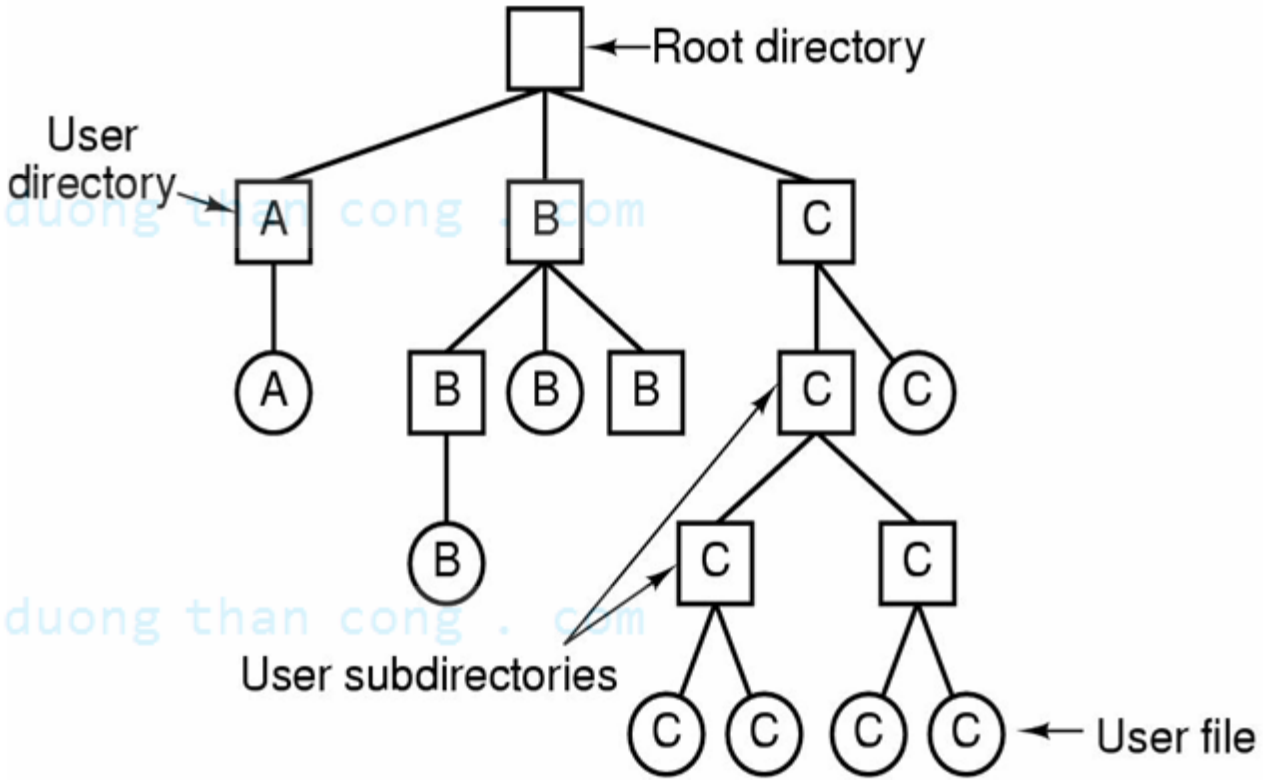
- Thiết bị chứa tin được trừu tượng thành 1 cây với 1 thư mục gốc. Thư mục gốc chứa nhiều thư mục con (cấp 2).
- Mỗi thư mục con cấp 2 chỉ chứa 1 số file chứ không chứa thư mục con nào cả.
- Dạng thư mục 2 cấp rất thích hợp cho các thiết bị chứa tin có dung lượng trung bình như đĩa flash ROM USB,...



Giao tiếp sử dụng phân hệ quản lý thư mục

3. Hệ thống file dùng thư mục n cấp :

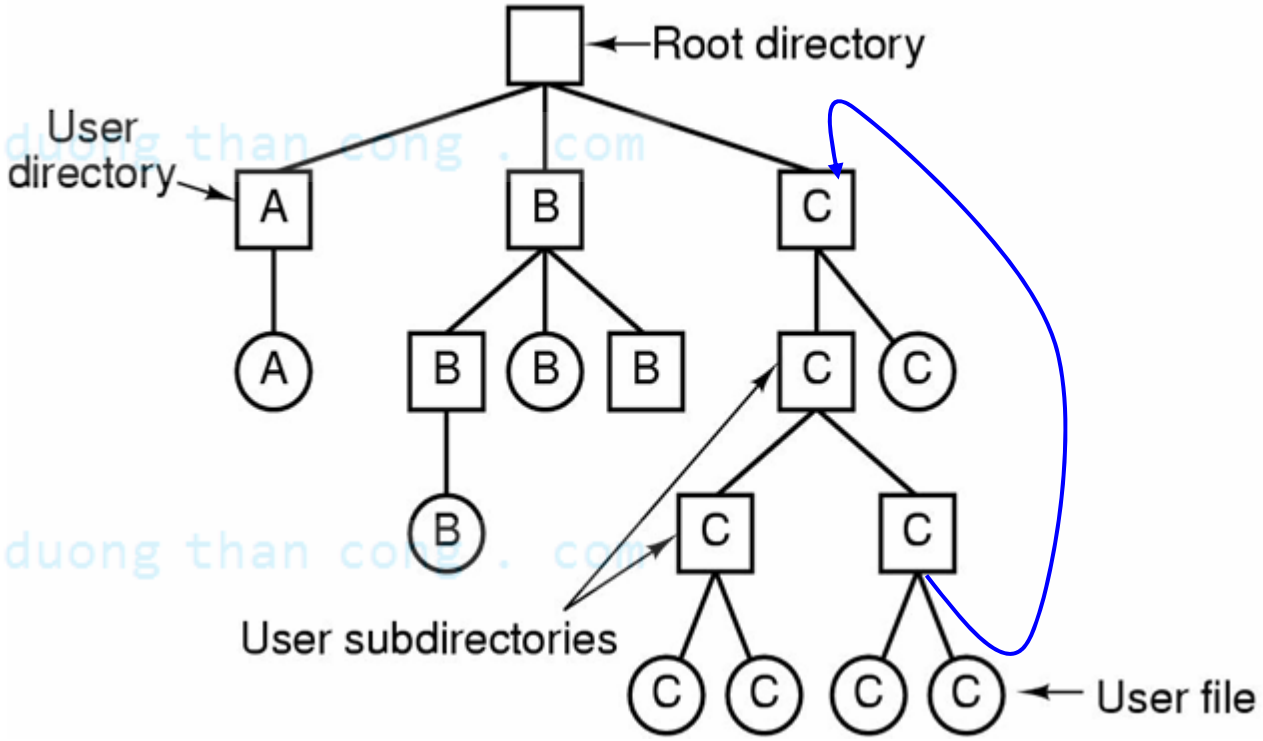
- Thiết bị chứa tin được trừu tượng thành 1 cây với 1 thư mục gốc. Thư mục gốc chứa nhiều thư mục con (cấp 2) và nhiều file.
- Mỗi thư mục con lại chứa 1 số file & 1 số thư mục con khác...
- Dạng thư mục n cấp rất thích hợp cho các thiết bị chứa tin có dung lượng lớn như đĩa cứng...



Giao tiếp sử dụng phân hệ quản lý thư mục

4. Hệ thống file dùng đồ thị thư mục n cấp :

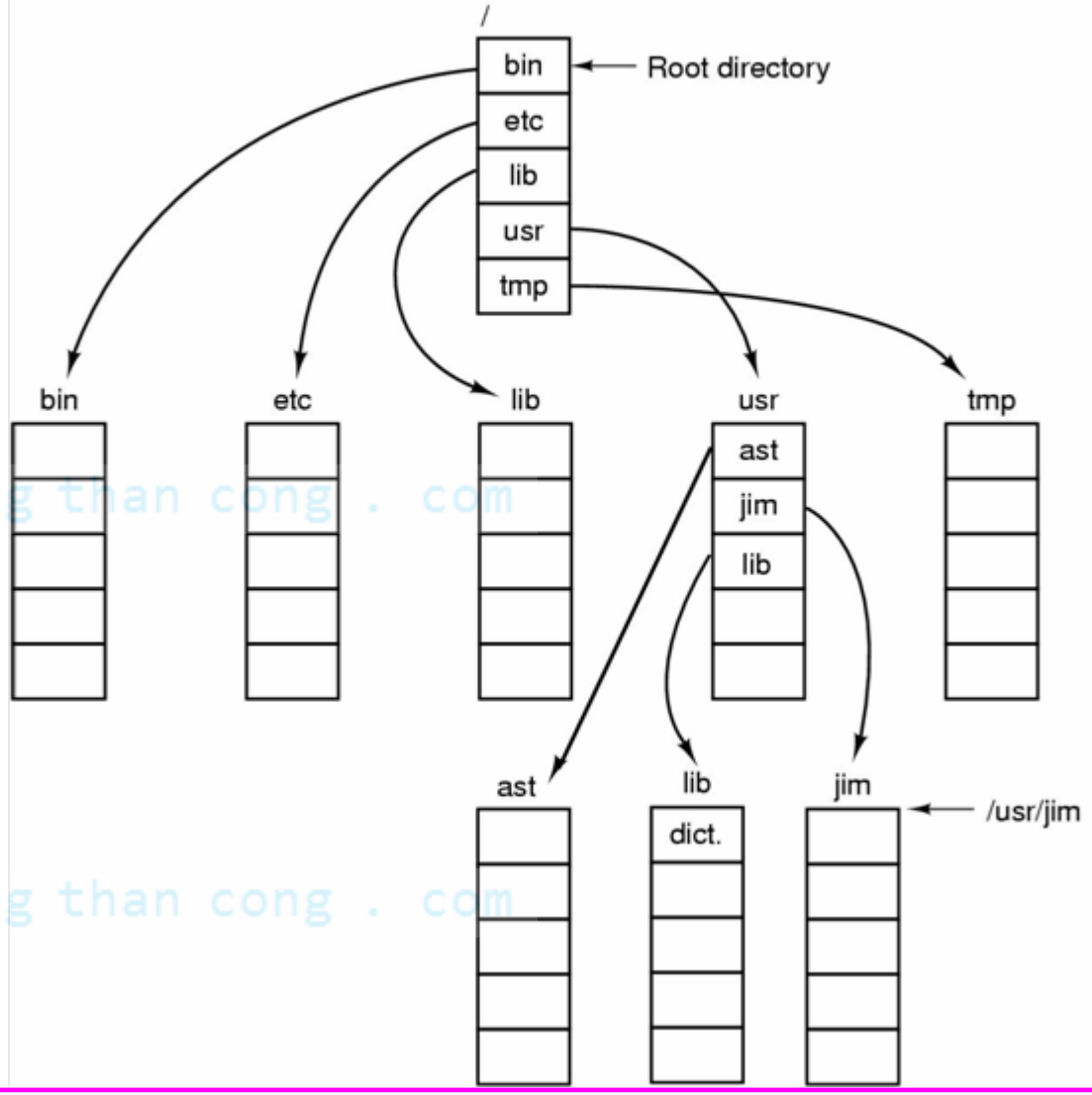
- Thiết bị chứa tin được trừu tượng thành 1 đồ thị với 1 thư mục gốc. Thư mục gốc chứa nhiều thư mục con (cấp 2) và nhiều file.
- Mỗi thư mục con lại chứa 1 số file & 1 số thư mục con khác. Thư mục con của 1 thư mục có thể là thư mục mới hay thư mục đã có sẵn...
- Dạng đồ thị thư mục rất thích hợp trong trường hợp quản lý các thư mục dùng chung...



Giao tiếp sử dụng phân hệ quản lý thư mục

5. Pathname :

- Pathname tương đối được chuyển thành pathname tuyệt đối trước khi xử lý tiếp.
- HĐH sẽ đọc lần lượt các thư mục trong pathname tuyệt đối, bắt đầu từ thư mục gốc, để dò lần ra phần tử cần truy xuất.



Giao tiếp sử dụng phân hệ quản lý thư mục

6. Các hàm API truy xuất thư mục : HĐH cung cấp 1 tập các hàm chức năng phục vụ việc truy xuất thư mục bởi ứng dụng.

6.1 Duyệt đọc nội dung thư mục :

FindFirst : đọc record chứa thông tin quản lý phần tử đầu tiên trong thư mục.

FindNext : đọc record chứa thông tin quản lý phần tử kế tiếp trong thư mục.

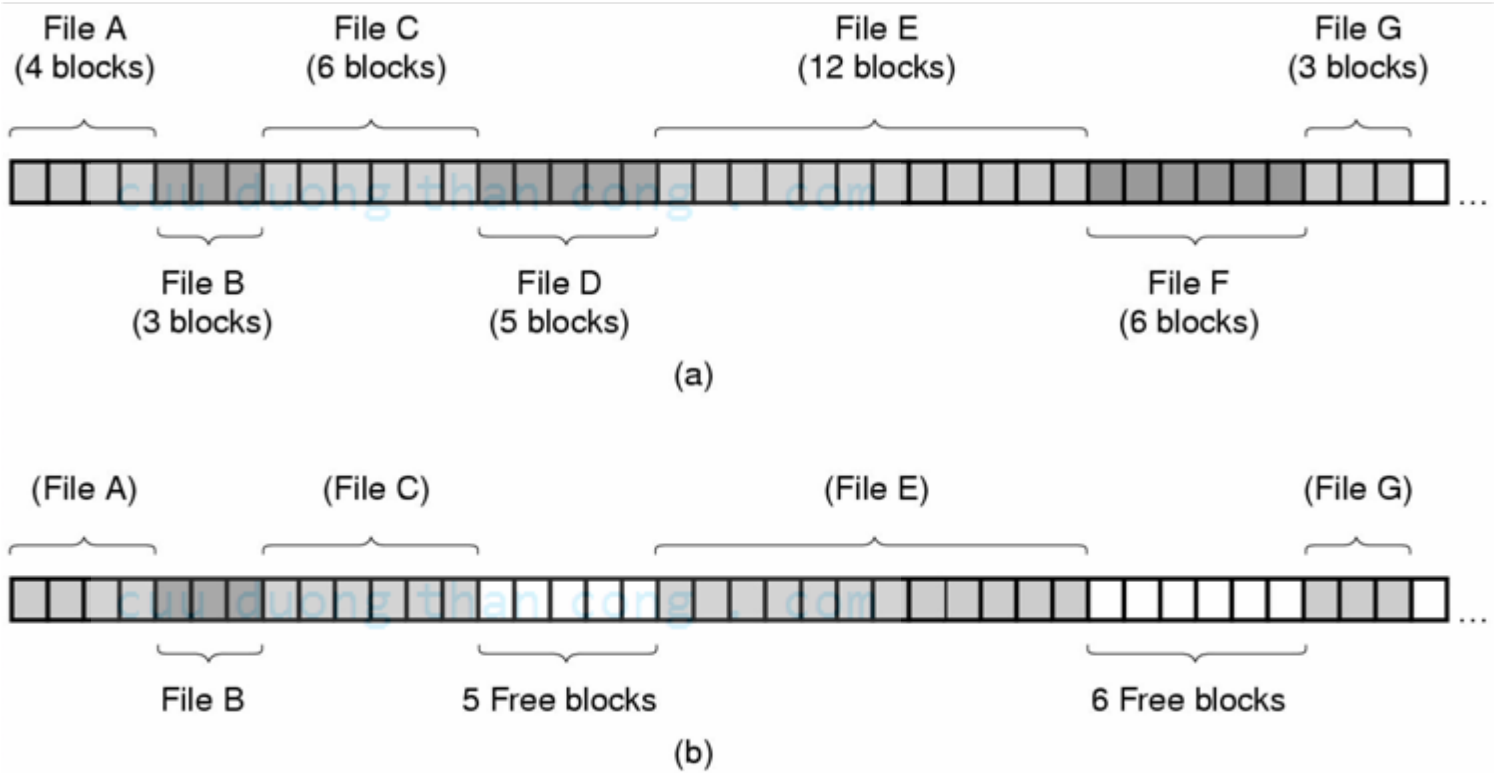
6.2 Hiệu chỉnh nội dung thư mục : 1 cách gián tiếp thông qua các lệnh sau :

- rename, create, delete, link, unlink, move file hay thư mục con.
- set_attributes file hay thư mục con...

7.4 Hiện thực file

1. File = danh sách đặc các cluster liên tiếp trên đĩa.

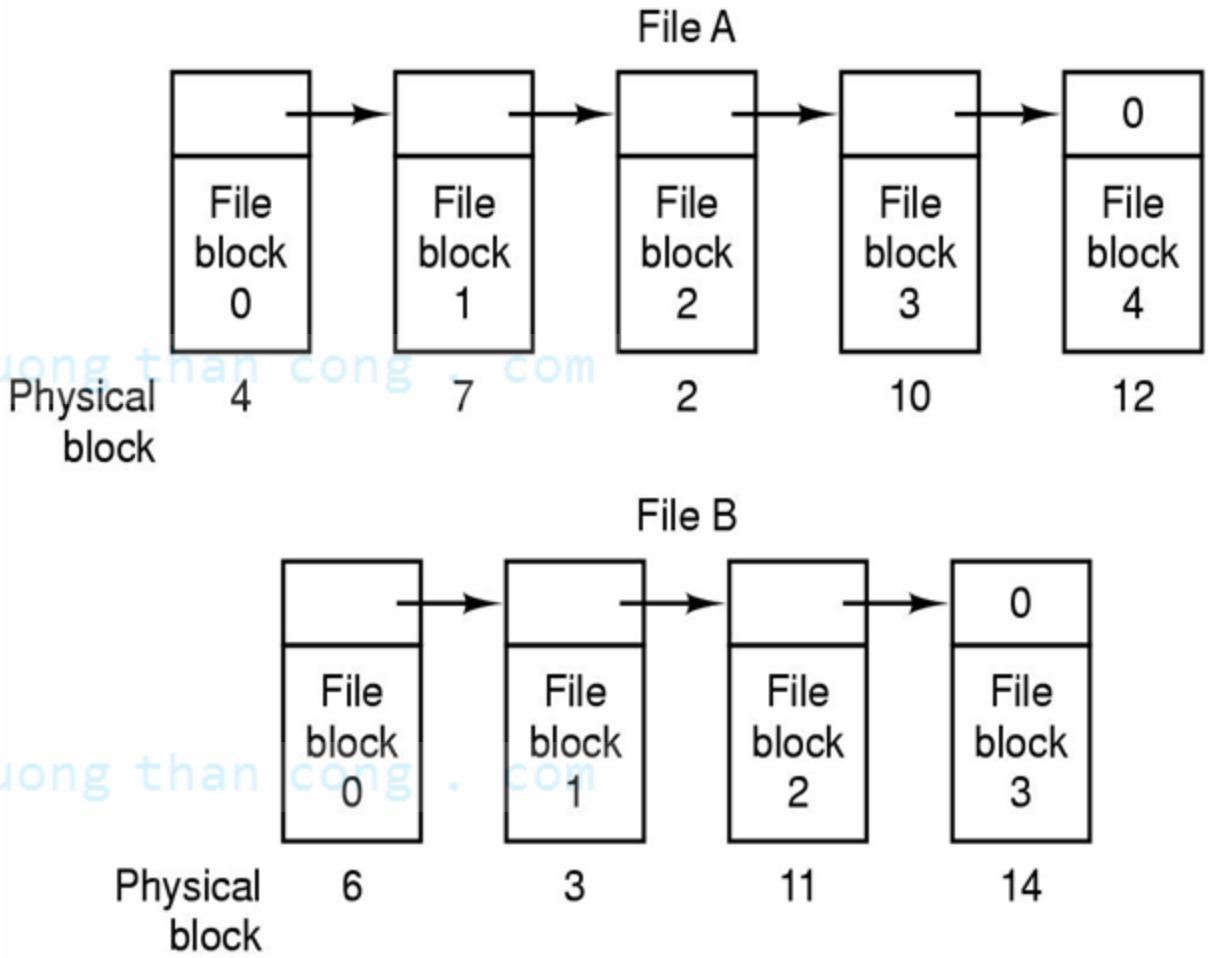
- Để quản lý 1 file, ta chỉ cần cặp thông số :
  (chỉ số cluster đầu file, số cluster cần dùng)
- rất bất tiện khi cần nới rộng kích thước file.



Hiện thực file

2. File = danh sách liên kết các cluster bất kỳ trên đĩa.

- truy xuất các cluster nội dung file theo cơ chế tuần tự → rất chậm, nhất là các dữ liệu nằm ở các cluster cuối của file lớn.
- kích thước phần chứa dữ liệu trong cluster bị xé lẻ → khó tính toán và xác định vị trí dữ liệu cần truy xuất.



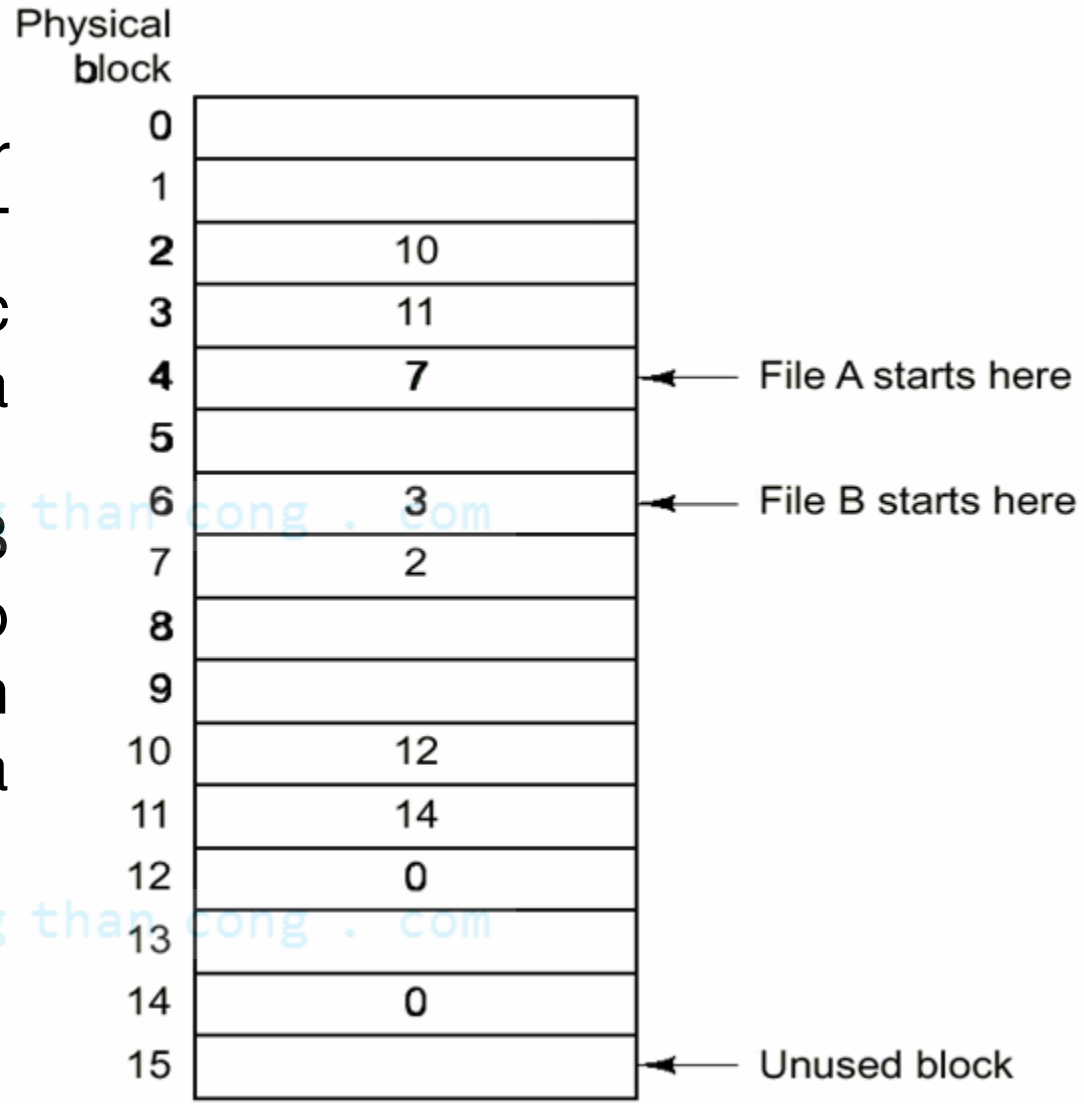
Hiện thực file

3. File = danh sách liên kết các cluster bất kỳ trên đĩa dùng FAT.

- tách phần link trong các cluster và tập hợp chúng thành 1 bảng, ta gọi bảng này là FAT (File Allocation Table).
- Số lượng phần tử trong bảng FAT = số cluster trên partition tương ứng, mỗi phần tử chứa pointer (dạng chỉ số) đến cluster đi ngay sau luận lý nó trong file.
- có nhiều loại FAT : FAT12, FAT16, FAT32. Fat i chứa maximum 2^i phần tử, mỗi phần tử dài i bit. Thí dụ FAT32 chứa tối đa 2^{32} phần tử, mỗi phần tử dài 32 bit, miêu tả chỉ số cluster từ 0 – 2^{32}-1.
- nếu HĐH nạp toàn bộ bảng FAT vào RAM thì việc truy xuất dữ liệu của các file là trực tiếp (giải quyết được khuyết điểm 1 của danh sách liên kết).
- mỗi cluster dữ liệu chỉ chứa dữ liệu → kích thước chẵn (giải quyết được khuyết điểm 2 của danh sách liên kết).
- thực tế cần dung hòa giữa dung lượng RAM dành cho FAT và hiệu quả truy xuất file → nạp từng 'window' của FAT vào RAM khi cần thiết.

Hiện thực file

- nếu ta biết file A có cluster đầu là 4, dựa vào bảng FAT bên phải ta sẽ tìm được các cluster còn lại của file A là 7, 2, 10, 12.
- tương tự, nếu ta biết file B có cluster đầu là 6, dựa vào bảng FAT bên phải ta sẽ tìm được các cluster còn lại của file B là 3, 11, 14.



Hiện thực file

Kích thước cluster và partition dùng FAT :

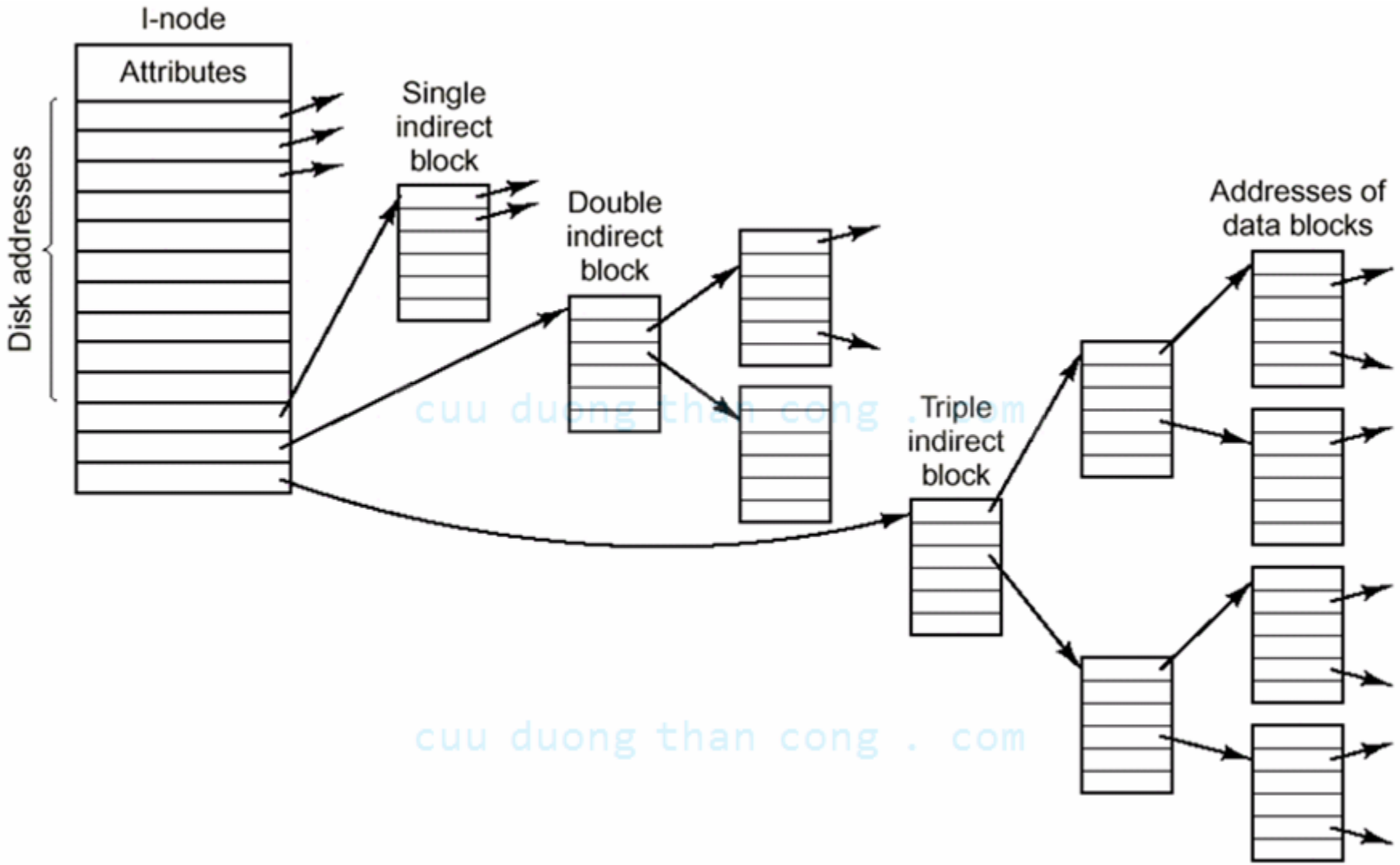
Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

Hiện thực file

4. File = dsách liên kết các cluster trên đĩa dùng bảng chỉ số.

- tập hợp các phần link trong các cluster của từng file thành 1 bảng chỉ số cho file đó. Bảng này được chứa trong 1 cấu trúc quản lý file (i-node trên Unix).
- Do kích thước các file rất khác nhau nên bảng chỉ số dài ngắn khác nhau → gây khó khăn trong việc quản lý → Unix dùng bảng chỉ số nhiều cấp :
  - 10 phần tử đầu chứa chỉ số các cluster dữ liệu.
  - 1 phần tử chỉ số đến cluster chứa các chỉ số các cluster dữ liệu (C11).
  - 1 phần tử chỉ số đến cluster chứa các chỉ số các cluster C11 (C12).
  - 1 phần tử chỉ số đến cluster chứa các chỉ số các cluster C12 (C13).
- mỗi khi cần truy xuất file, HĐH nạp i-node của file vào RAM → việc truy xuất dữ liệu của file là trực tiếp.
- mỗi cluster dữ liệu chỉ chứa dữ liệu → kích thước chẵn (giải quyết được khuyết điểm 2 của danh sách liên kết).

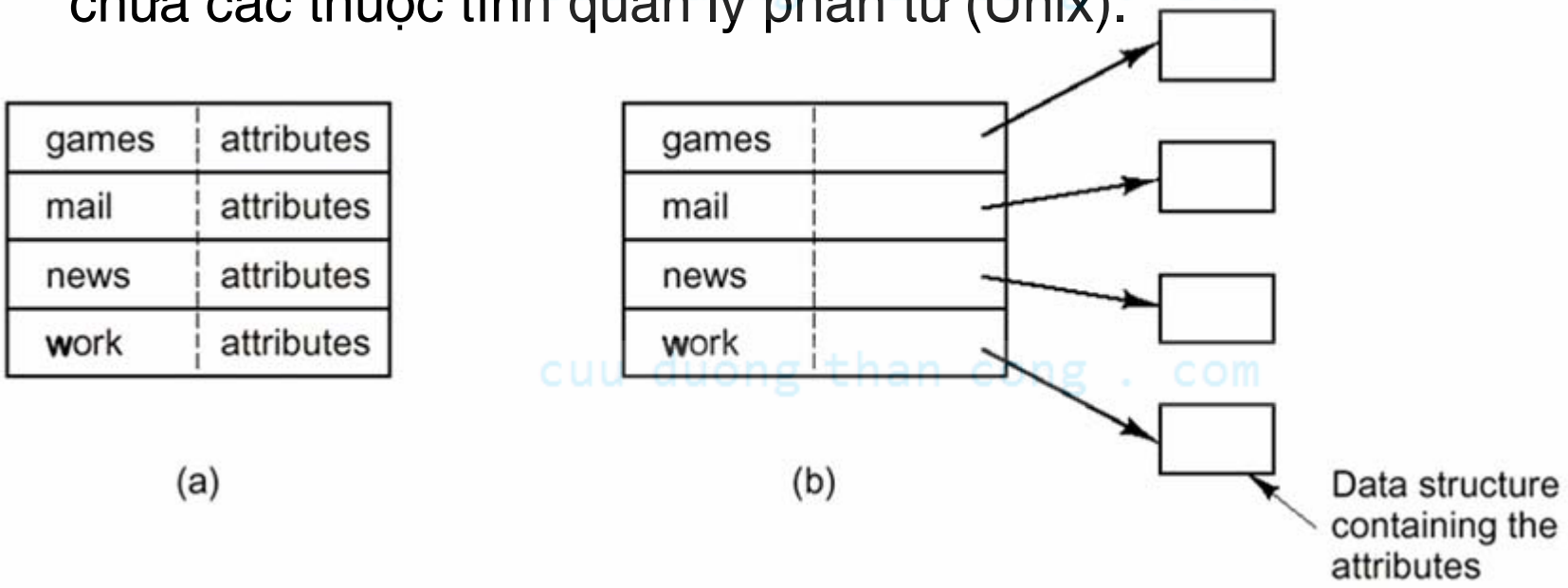
Hiện thực file



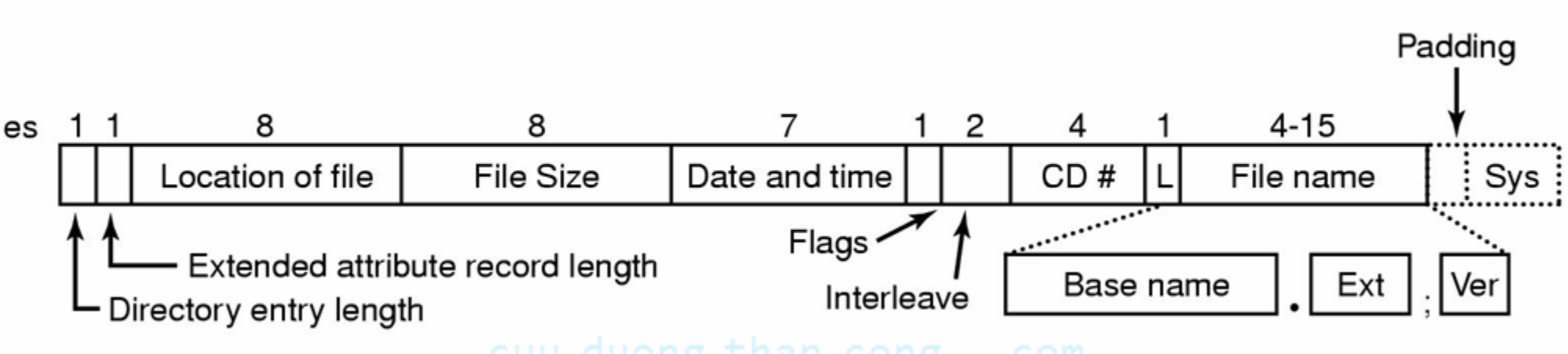
7.5 Hiện thực thư mục

1. Thư mục = file (gồm nhiều cluster đĩa).

- cấu trúc ngữ nghĩa của thư mục = danh sách đặc các record, mỗi record (entry) quản lý thông tin của 1 phần tử con, gồm 2 thành phần chính : tên gợi nhớ của phần tử, các thuộc tính quản lý phần tử.
- Hình a. : tất cả các thuộc tính quản lý phần tử đều được để trong entry thư mục (FAT).
- Hình b. : entry thư mục chỉ chứa chỉ số i-node, i-node mới là record chứa các thuộc tính quản lý phần tử (Unix).

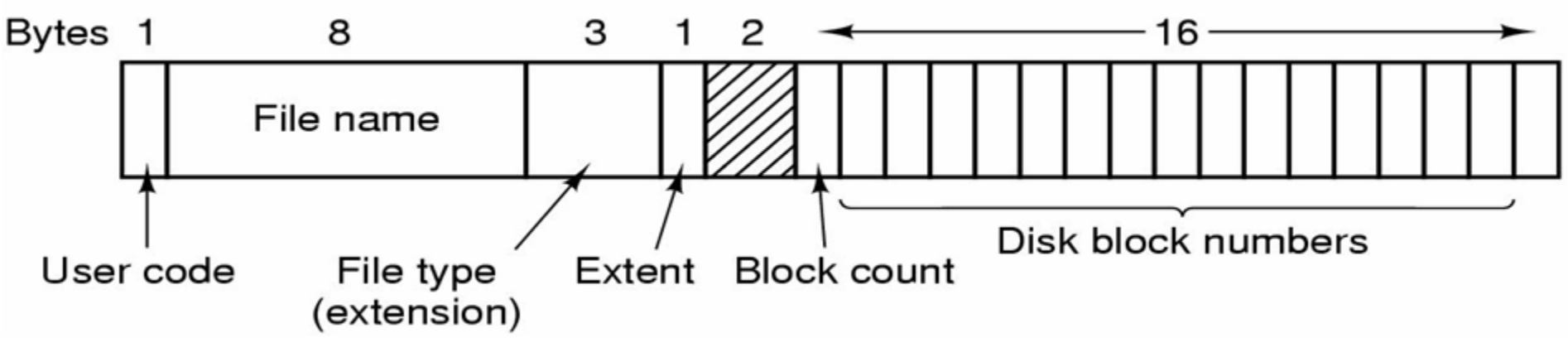


7.5 Hiện thực thư mục

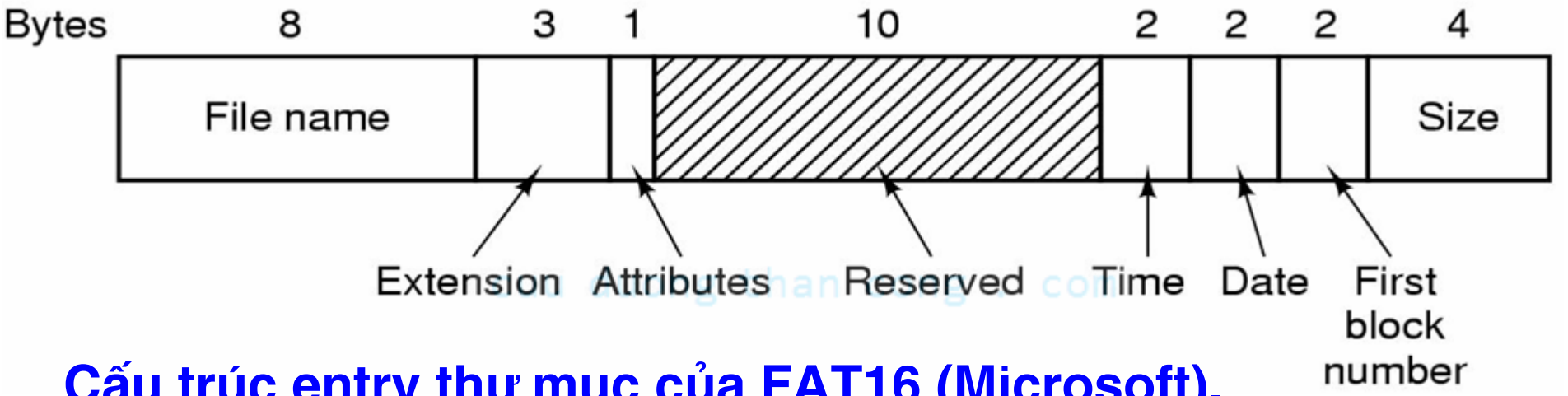


Cấu trúc entry thư mục của chuẩn ISO9660 (CDROM).

7.5 Hiện thực thư mục



Cấu trúc entry thư mục trên HĐH CP/M.

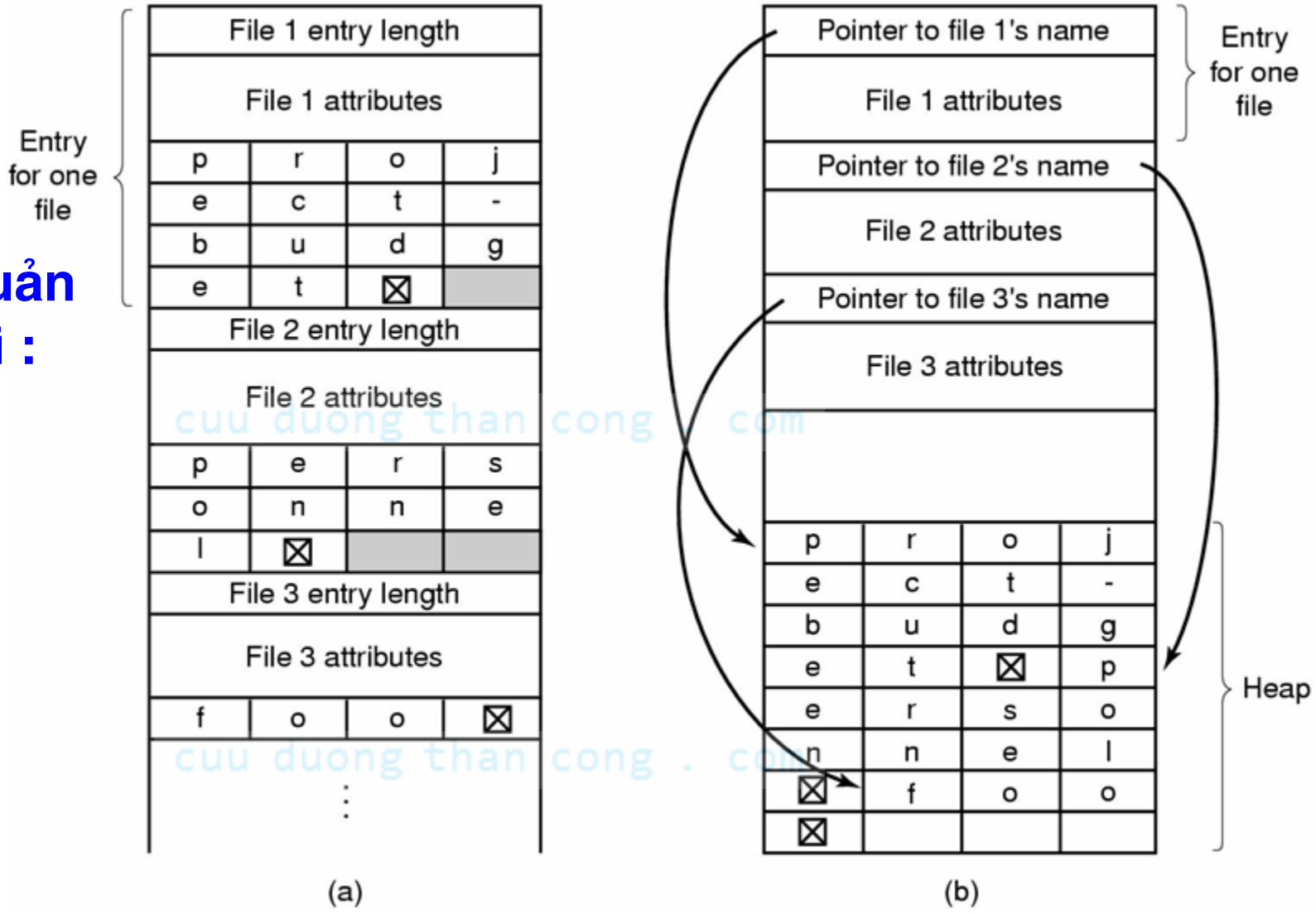


Cấu trúc entry thư mục của FAT16 (Microsoft).

Hiện thực thư mục

Có 2 cách quản lý tên file dài :

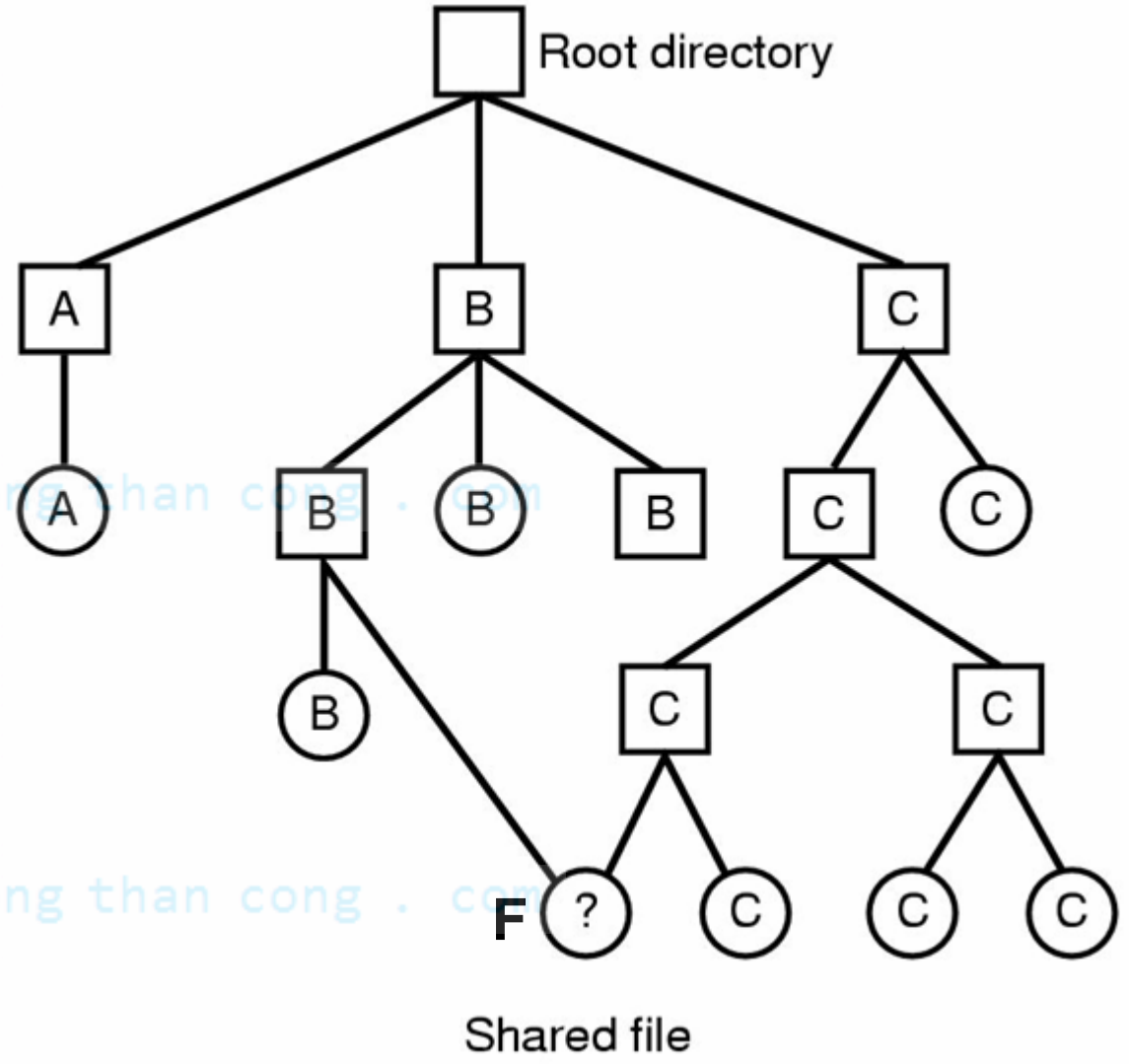
- Hình a. : in-line.
- Hình b. : in a heap.



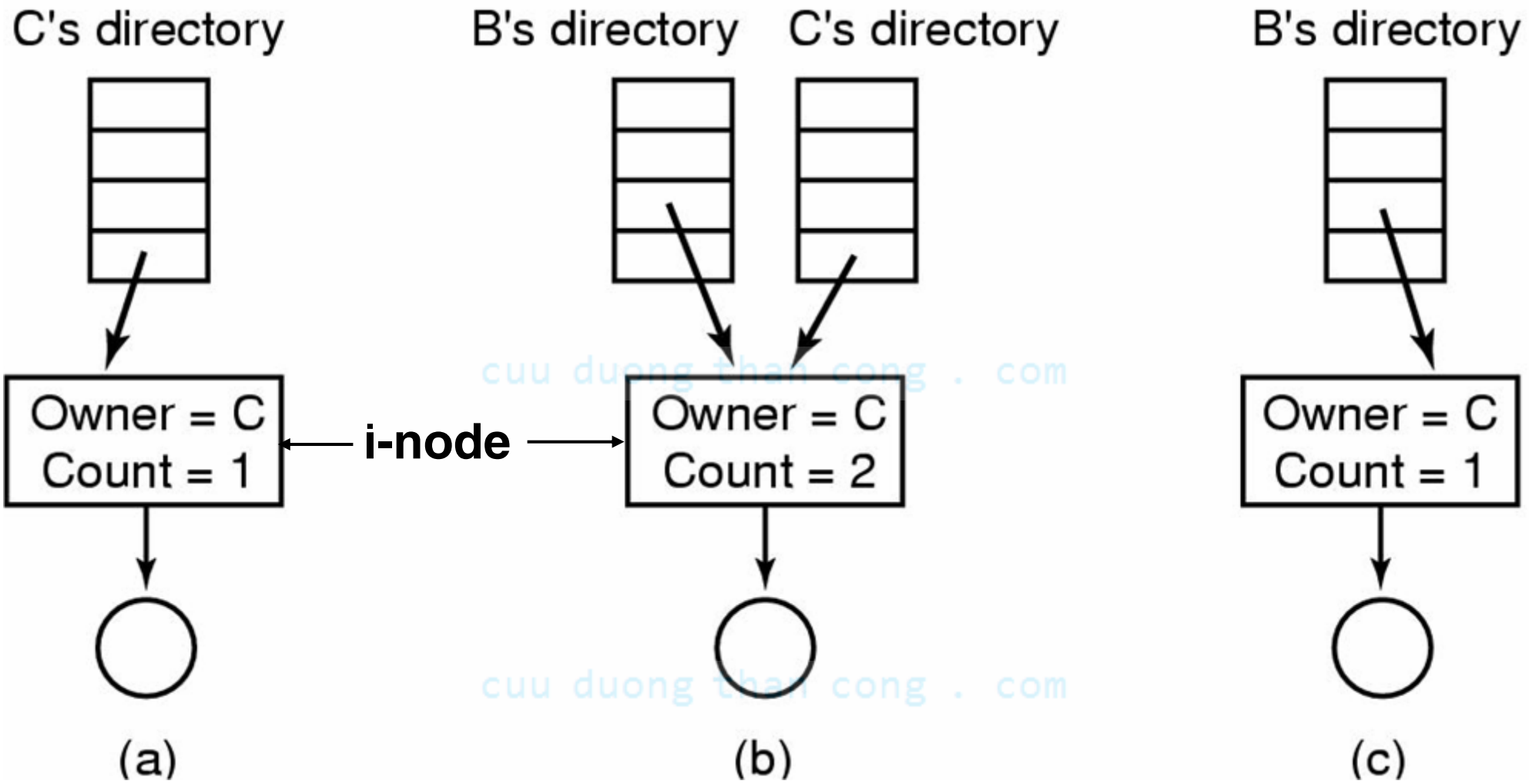
Hiện thực thư mục

Dùng i-node để quản lý các thuộc tính của file/thư mục rất thích hợp cho việc dùng chung file/thư mục.

- File F là file dùng chung cho 2 thư mục B và C.



Hiện thực thư mục



7.6 Quản lý các cluster chưa dùng

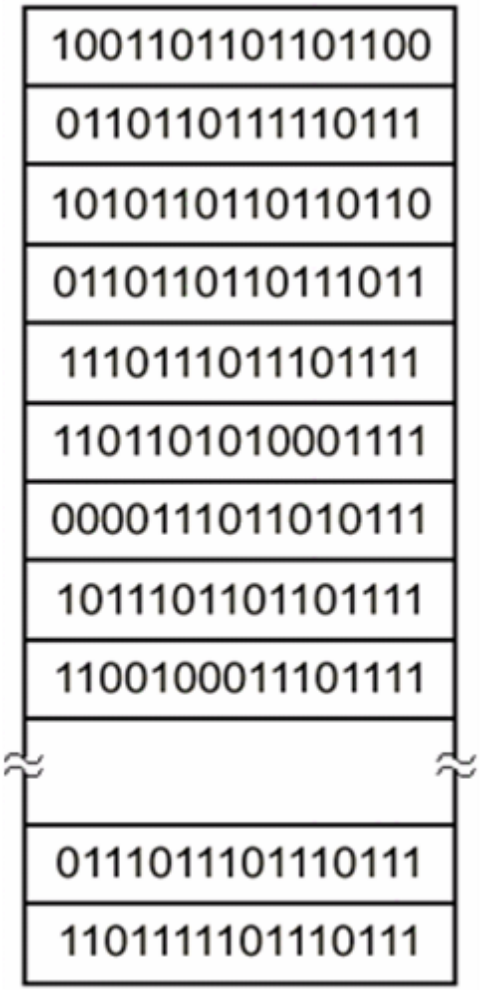
Tại từng thời điểm, mỗi cluster thuộc 1 trong 2 trạng thái sau :

- đã dùng, cluster này phải thuộc về 1 file nào đó → truy xuất file đó ta sẽ tìm được cluster này.
- chưa dùng nên chưa thuộc file nào cả → cần có cơ chế quản lý các cluster này.

Quản lý các cluster chưa dùng

1. Dùng bảng trạng thái

- bảng trạng thái có N phần tử (N : số cluster của partition), mỗi phần tử là 1 bit nhị phân, nếu = 0 thì cluster tương ứng chưa dùng, nếu = 1 thì cluster tương ứng đã dùng rồi bởi file nào đó.
- khi ứng dụng xin cấp phát 1 cluster, HĐH sẽ dò tuần tự trong bảng trạng thái để tìm 1 phần tử (chỉ số i) = 0 → cluster i chưa dùng.
- Việc dò tuần tự là không hiệu quả, ngay cả có cải tiến bằng cách lưu giữ vị trí đã dò tìm lần cuối để mỗi khi tìm cluster chưa dùng, ta chỉ dò từ vị trí lưu giữ chứ không phải dò từ đầu bảng.

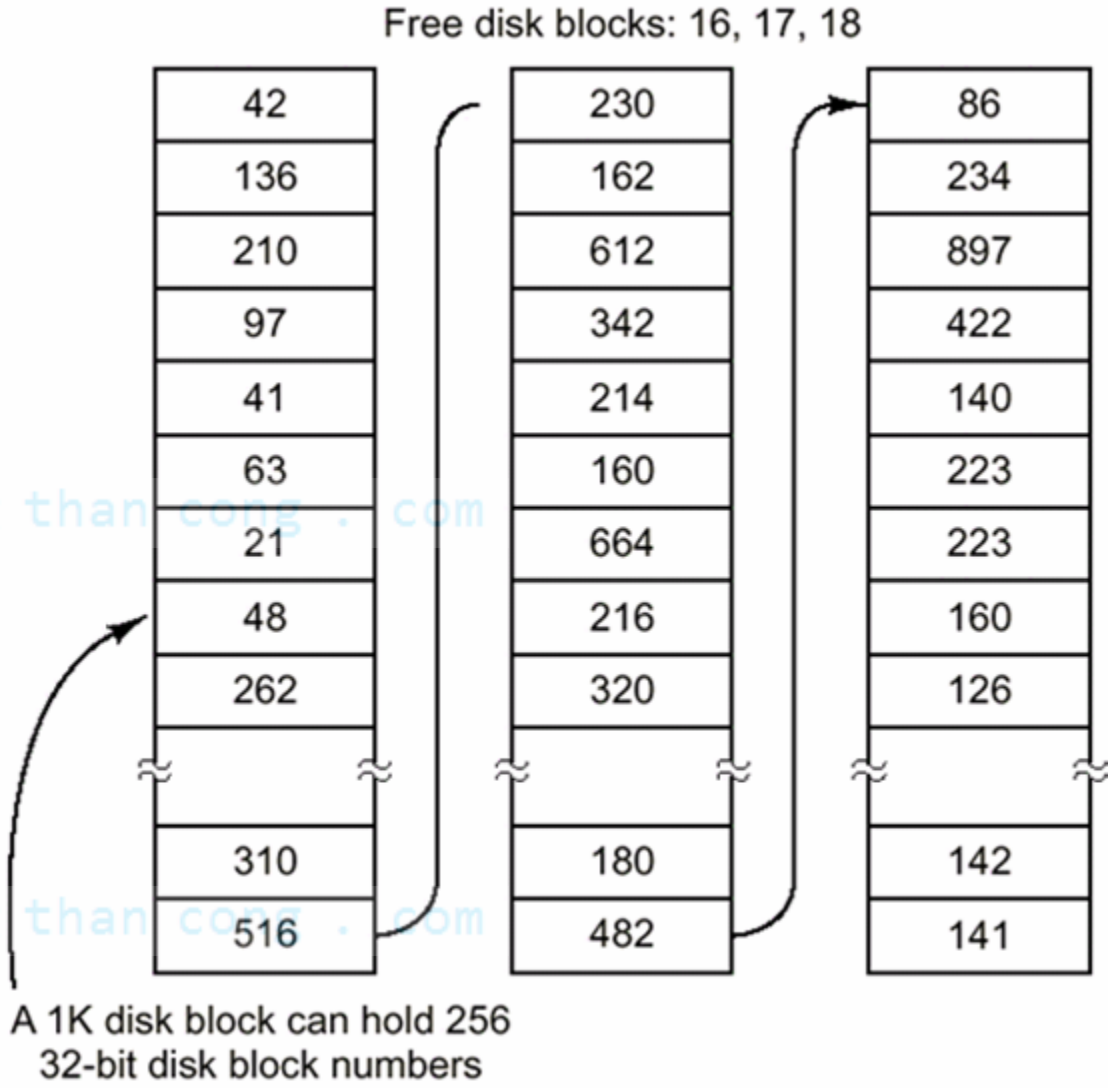


A bit map

Quản lý các cluster chưa dùng

2. Dùng danh sách liên kết các cluster chưa dùng

- danh sách chỉ chứa các cluster chưa dùng.
- khi ứng dụng xin cấp phát 1 cluster, HĐH sẽ lấy ngay cluster ở đầu danh sách rồi hiệu chỉnh lại đầu danh sách mới.
- tốn nhiều bộ nhớ cho danh sách liên kết, nhất là lúc partition mới format, chưa có file nào cả.



Quản lý các cluster chưa dùng

3. Tính nhất quán của hệ thống file

- Về ý nghĩa, tại từng thời điểm, mỗi cluster hoặc thuộc 1 file nào đó hoặc chưa dùng và được quản lý trong danh sách cluster chưa dùng.
- hệ thống file được gọi là đang nhất quán nếu từng cluster trong hệ thống thỏa mãn điều kiện nêu trên.

Hệ thống bị mất tính nhất quán khi 1 trong các điều kiện sau đã xảy ra :

1. 1 cluster được dùng bởi nhiều hơn 1 file.
2. 1 cluster được dùng bởi 1 file nhưng vẫn xuất hiện trong danh sách cluster chưa dùng.
3. 1 cluster không thuộc file nào nhưng tồn tại nhiều lần trong danh sách các cluster chưa dùng.
4. 1 cluster không thuộc file nào và không xuất hiện trong danh sách các cluster chưa dùng.

Quản lý các cluster chưa dùng

4. Trình phát hiện mất tính nhất quán của hệ thống file và chữa trị (scandisk) :

1. dùng 2 cấu trúc dữ liệu chính yếu :

   used[N], trong đó used[i] = số file sử dụng cluster i (thường là 0 hay 1).

   noused[N], trong đó noused[i] = số lần cluster i xuất hiện trong danh sách cluster chưa dùng (thường là 0 hay 1).

2. khởi động giá trị ban đầu của 2 cấu trúc dữ liệu :

   for (i =0; i <N; i++) used[i] = noused[i] = 0;

3. xây dựng vector used[N] bằng cách duyệt đệ quy hệ thống file từ thư mục gốc, mỗi lần gặp 1 file/thư mục, ta duyệt từng cluster được nó sử dụng, mỗi lần gặp cluster i, ta thực hiện used[i]++.

4. xây dựng vector noused[N] bằng cách duyệt danh sách các cluster chưa dùng, mỗi lần gặp cluster i, ta thực hiện noused[i]++.

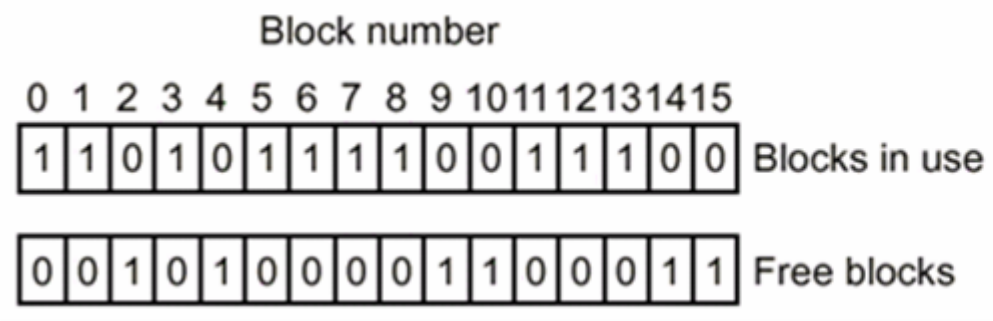
Quản lý các cluster chưa dùng

4. Trình phát hiện mất tính nhất quán của hệ thống file và chữa trị

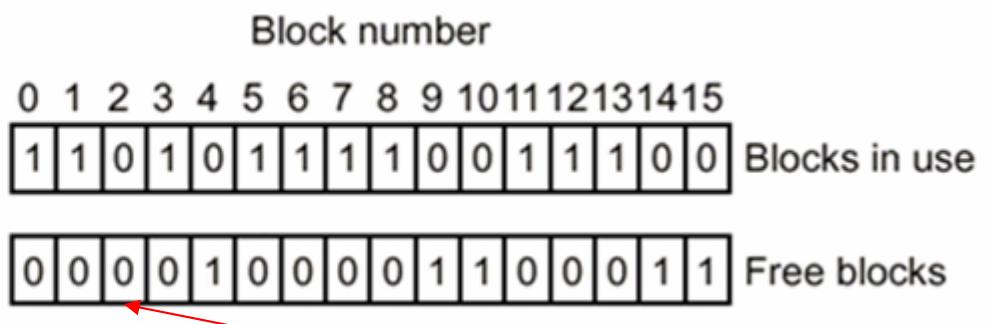
5. Lặp kiểm tra trạng thái của từng cluster :

```
for (i =0; i <N; i++) {
    if (used[i] + noused[i] == 1) continue; //tốt
    if (used[i] >1) //case 1: báo sai để user tự sửa
    if (noused[i] >1) //case 3: chỉ giữ lại 1 lần cluster i
    if (used[i] + noused[i] > 1) //case 2: bỏ i khỏi danh sách chưa dùng
    if (used[i]+noused[i] ==0) //case 4: thêm i vào danh sách chưa dùng
}
```

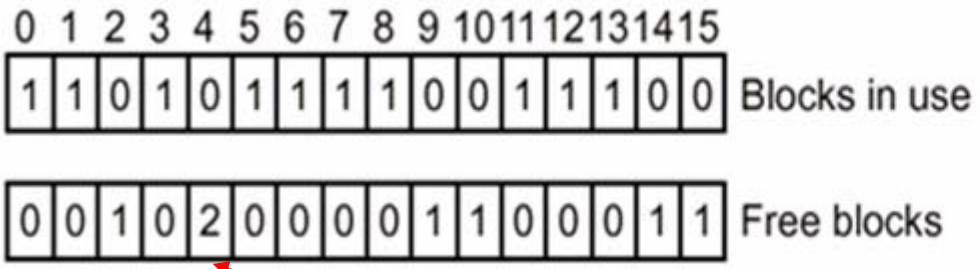
Quản lý các cluster chưa dùng



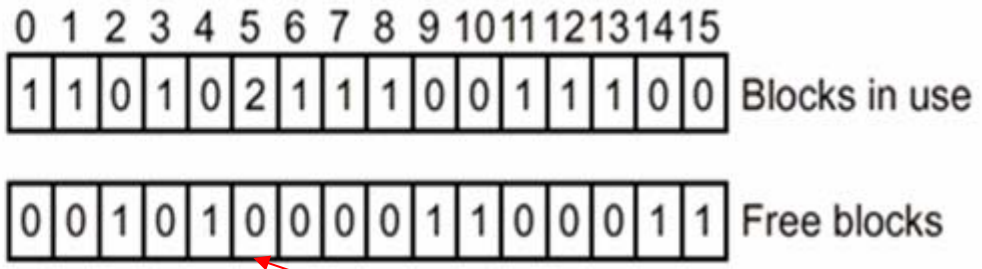
a. còn nhất quán



b. bị mất nhất quán (case 4) ở



c. bị mất nhất quán (case 3) ở

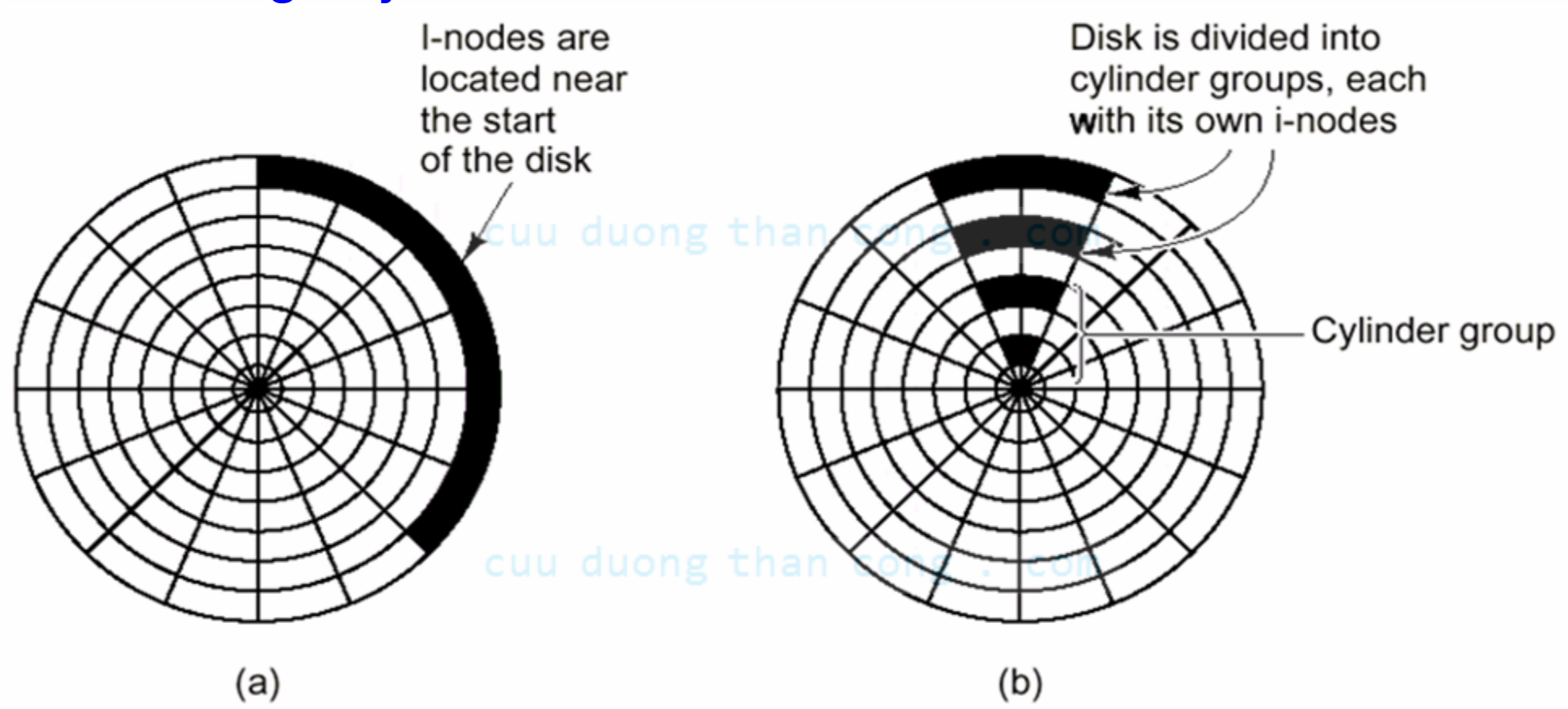


d. bị mất nhất quán (case 1) ở

7.7 Các việc quản lý khác trên hệ thống file

Tốc độ truy xuất dữ liệu

1. Cấp phát các cluster chứa dữ liệu và các cluster quản lý của từng file trên cùng 1 cylinder :



7.7 Các việc quản lý khác trên hệ thống file

Tốc độ truy xuất dữ liệu

- **Các cluster dữ liệu của từng file nằm liền kề nhau**

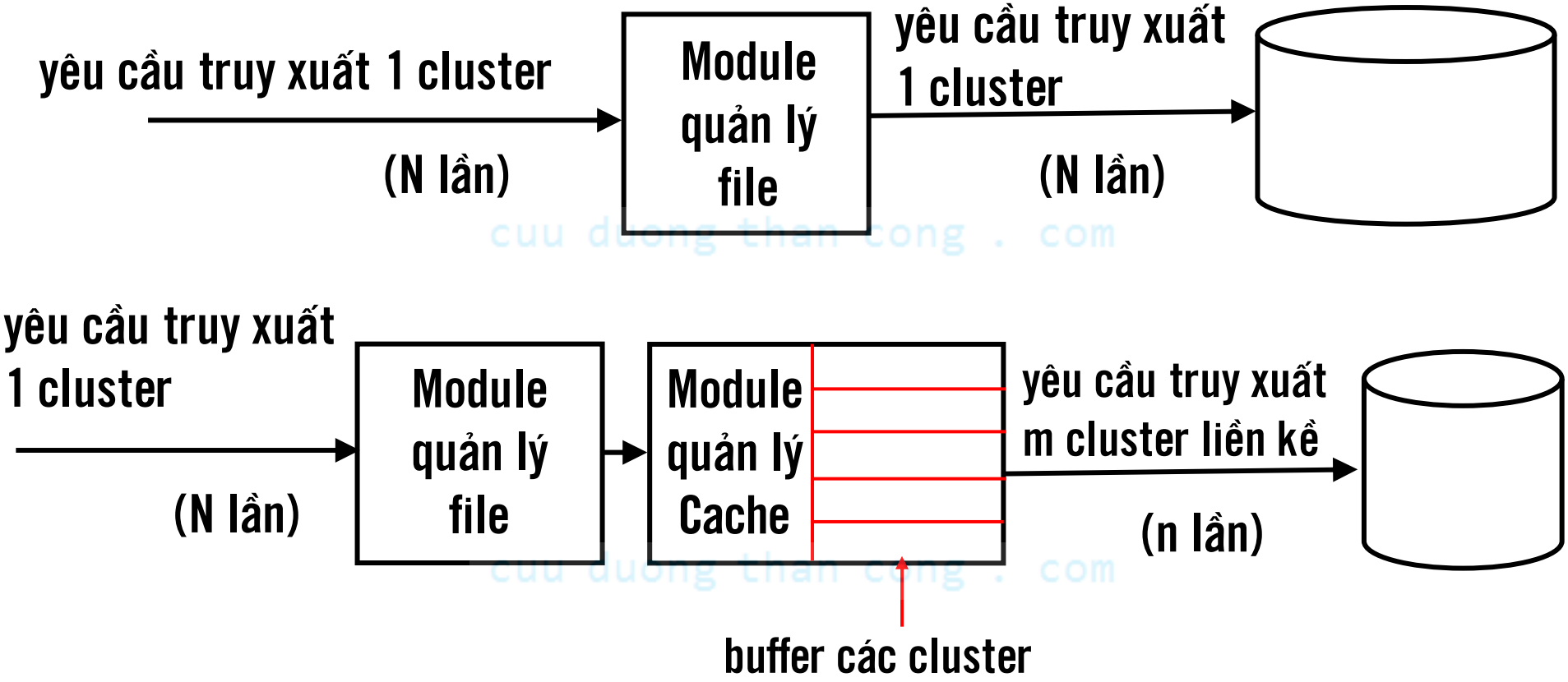
  (dùng trình defragmentation khi thích hợp).

3. **Mua disk có các tham số sau tốt nhất : tốc độ quay đĩa, tốc độ step motor, chuẩn giao tiếp SCSI, SATA,...**

7.7 Các việc quản lý khác trên hệ thống file

Tốc độ truy xuất dữ liệu

4. Dùng cơ chế Cache các cluster cần đọc/ghi :



7.7 Các việc quản lý khác trên hệ thống file

Tốc độ truy xuất dữ liệu

5. **Lập lịch phục vụ truy xuất các cluster từ các ứng dụng chạy đồng thời :** sao cho độ dời đầu đọc/ghi (số xung clock gởi tới step motor) tối thiểu.

   Thí dụ tại thời điểm t đầu đọc/ghi ở cylinder 0, có 4 yêu cầu đọc/ghi sau đây của 4 ứng dụng độc lập :

   - P1 yêu cầu truy xuất cylinder 1000.
   - P2 yêu cầu truy xuất cylinder 0.
   - P3 yêu cầu truy xuất cylinder 999.
   - P4 yêu cầu truy xuất cylinder 1.

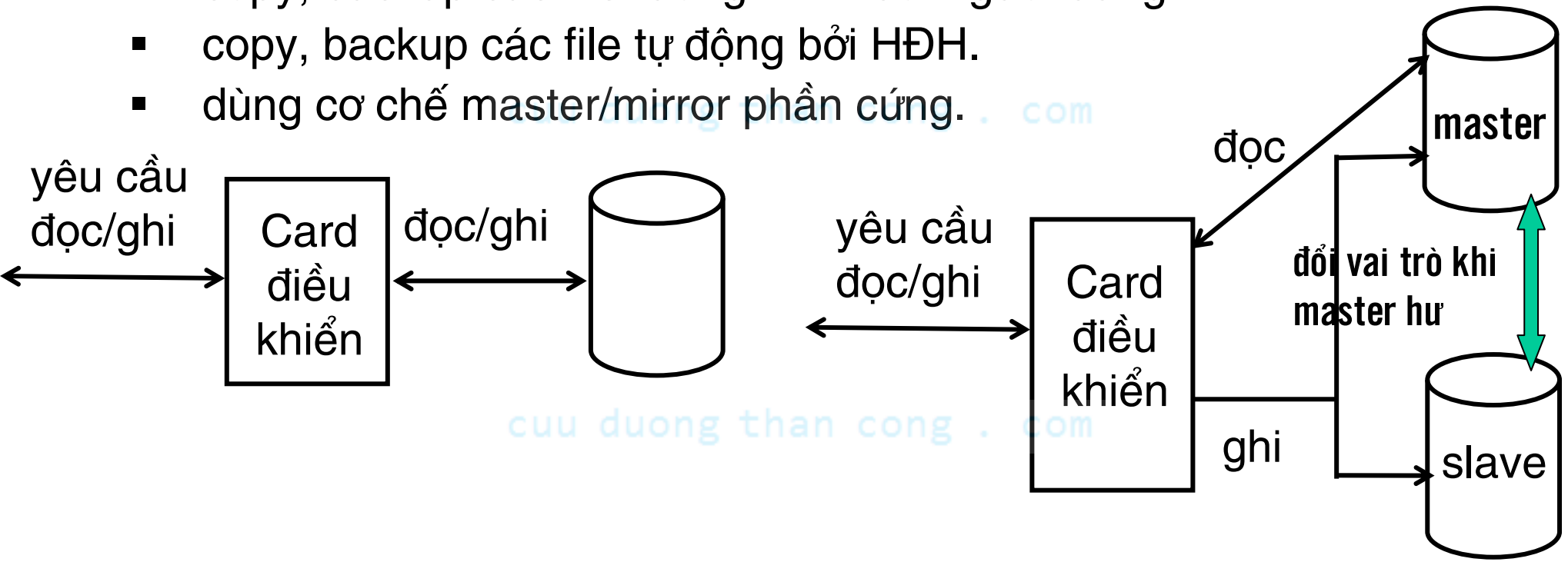
   Nếu lập lịch phục vụ P1 → P2 → P3 → P4 thi cần step motor 1000+1000+999+998 = 3997 lần.

   Nếu lập lịch phục vụ P2 → P4 → P3 → P1 thi cần step motor 0+1+998+1 =1000 lần, ít hơn 4 lần so với cách lập lịch trước.

7.7 Các việc quản lý khác trên hệ thống file

Tính tin cậy & sẵn sàng trong truy xuất file

1. Trang bị disk của hãng nổi tiếng, uy tín về thương hiệu → giá cao hơn nhiều so với sự mong đợi.
2. Dùng cơ chế dư thừa :
   - copy, backup các file tường minh bởi người dùng.
   - copy, backup các file tự động bởi HĐH.
   - dùng cơ chế master/mirror phần cứng.



7.7 Các việc quản lý khác trên hệ thống file

Bảo mật trong truy xuất file

1. Quản lý người dùng bằng Account : mỗi người được cấp 1 account. Hệ thống sẽ phục vụ người dùng thông qua qui trình 2 bước :
   - Xác nhận người dùng : hệ thống sẽ yêu cầu người dùng cung cấp 1 số thông tin để xác định mình là ai, có hợp pháp không (thí dụ username+password của account). Nếu hệ thống xác nhận người dùng là hợp pháp, hệ thống sẽ biết người dùng đó đang dùng Account nào (thứ i trong danh sách quản lý).
   - Phục vụ người dùng : mỗi lần người dùng i ra lệnh (đọc/ghi/xóa/... file A), hệ thống sẽ kiểm tra lệnh đó có tương thích với quyền được ghi nhận trong account tương ứng với user không. Nếu không thì báo sai, nếu có thì phục vụ.

7.7 Các việc quản lý khác trên hệ thống file

Bảo mật trong truy xuất file

2. Ma trận đặc tả quyền truy xuất tài nguyên của các user :

- Hàng i ghi nhận các quyền của user i trên các tài nguyên.
- Cột j ghi nhận các quyền của từng user trên tài nguyên j.
- Khi có lệnh của user i trên tài nguyên j, hệ thống sẽ xem phần tử (i,j) có cho phép chức năng tương ứng không ?

	Object							
	File1	File2	File3	File4	File5	File6	Printer1	Plotter2
Domain 1	Read	Read Write						
2			Read	Read Write Execute	Read Write		Write	
3						Read Write Execute	Write	Write

7.7 Các việc quản lý khác trên hệ thống file

Bảo mật trong truy xuất file

Ma trận đặc tả quyền truy xuất tài nguyên của các user :

1. Kích thước rất lớn, nhất là số lượng cột của ma trận.
2. Thay đổi nhanh chóng, từng cột có thể được thêm vào/xóa đi rất nhanh theo thời gian khi file tương ứng được tạo ra/xóa đi.

   Dùng ma trận đặc 2 chiều là không thích hợp. Cần tìm cách biểu diễn ma trận quyền truy xuất sao cho 2 vấn đề trên được được khắc phục.

Lưu ý ma trận đặc tả quyền truy xuất là ma trận rất thưa vì tuyệt đại đa số các phần tử trong ma trận đều có giá trị NULL.

7.7 Các việc quản lý khác trên hệ thống file

Bảo mật trong truy xuất file

3. Biểu diễn ma trận quyền truy xuất bằng các danh sách liên kết theo hàng :
   1. Kết hợp mỗi user 1 danh sách liên kết (Capability), mỗi phần tử trong danh sách cho biết file nào được truy xuất và các hoạt động cụ thể nào (Read/Write/Execute,...) được phép trên file đó. Hệ điều hành Novell Netware dùng cách này.
   2. Trong thực tế số lượng phần tử trong danh sách Capability kết hợp với từng user vẫn còn rất lớn, người ta tối giản thêm bằng cách gom các file trong cùng thư mục thành 1 phần tử, phần tử này là thư mục với nghĩa : nếu user i có các quyền nào đó trên thư mục xác định thì cũng sẽ có các quyền đó trên mọi phần tử con của nó trong cây gia phả. Với cách tối giản này, danh sách các phần tử được phép truy xuất bởi từng user sẽ giảm thiểu rất lớn, chỉ cần vài 3 phần tử là đủ.

7.7 Các việc quản lý khác trên hệ thống file

Bảo mật trong truy xuất file

4. Biểu diễn ma trận quyền truy xuất bằng các danh sách liên kết theo cột :
   1. Kết hợp mỗi tài nguyên 1 danh sách liên kết (ACL – Access Control List), mỗi phần tử trong danh sách cho biết user nào được truy xuất và các hoạt động cụ thể nào (Read/Write/Execute,...) mà user đó được phép trên tài nguyên tương ứng. Hệ điều hành Windows, Linux dùng cách này.
   2. Trong thực tế số lượng phần tử trong danh sách ACL kết hợp với từng tài nguyên vẫn còn lớn, hơn nữa thường có nhiều user có cùng khả năng trên tài nguyên, do đó Unix đã tối giản ACL bằng bảng tra gồm 3 phần tử như sau :

7.7 Các việc quản lý khác trên hệ thống file

Bảo mật trong truy xuất file

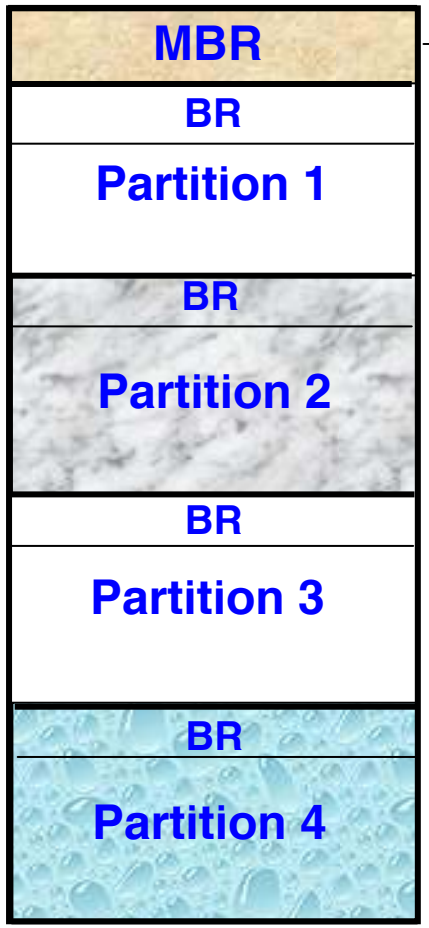
4. Biểu diễn ma trận quyền truy xuất bằng các danh sách liên kết theo cột :

	Read	Write	Execute
Owner	0/1	0/1	0/1
Group	0/1	0/1	0/1
Other	0/1	0/1	0/1

Bảng tra trên dài 9 bit, đây là field Protection trong i-node chứa các thông tin thuộc tính của file tương ứng.

7.8 Quản lý hệ thống file trên máy PC

Cấu trúc 1 đĩa cứng PC



Mỗi khi boot máy, ROM BIOS sẽ đọc MBR vào RAM và chạy nó.

Trình bootstrap bằng mã máy, có nhiệm vụ tìm partition active và nạp BR của partition này vào máy và giao điều khiển cho nó

446 byte

Mỗi đĩa cứng có thể chia tối đa 4 partition độc lập, thông tin về 1 partition được lưu trong 1 record 16 byte.

16 byte
16 byte
16 byte
16 byte

Quản lý hệ thống file trên máy PC

Cấu trúc record đặc tả partition của bảng quản lý các partition :

Byte Offset	Length	Sample Value	Field Name and Definition
0x0	BYTE	0x80	80. Active partition, 00 : No Active.
0x1	BYTE	0x01	Starting Head.
0x2	6 bits	0x01 *	Starting Sector. Only bits 0-5 are used.
0x3	10 bits	0x00 *	Starting Cylinder.
0x4	BYTE	0x07	System ID. Defines the volume type. values.
0x5	BYTE	0xFE	Ending Head.
0x6	6 bits	0xBF *	Ending Sector. Only bits 0-5 are used.
0x7	10 bits	0x09 *	Ending Cylinder.
0x8	DWORD	0x3F000000	Relative Sectors.
0xC	DWORD	0x4BF57F00	Total Sectors.

Quản lý hệ thống file trên máy PC

Cấu trúc BootSector của partition FAT32 :

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	53 bytes	BPB (BIOS Parameter Block)
0x40	26 bytes	Extended BPB
0x5A	420 bytes	Bootstrap Code
0x01FE	WORD	End of Sector Marker

Quản lý hệ thống file trên máy PC

Cấu trúc BPB của FAT32 :

Byte Offset	Length	Sample Value	Field Name and Definition
0x0B	WORD	0x0002	Bytes Per Sector. (512byte)
0x0D	BYTE	0x08	Sectors Per Cluster.
0x0E	WORD	0x0200	Reserved Sectors.
0x10	BYTE	0x02	Number of FATs.
0x11	WORD	0x0000	this field must be set to zero.
0x13	WORD	0x0000	this field must be set to zero.
0x15	BYTE	0xF8	Media Descriptor. (0xF8 -> hard disk)
0x16	WORD	0x0000	this field must be set to zero.
0x18	WORD	0x3F00	Sectors Per Track.
0x1A	WORD	0xFF00	Number of Heads.
0x1C	DWORD	0xEE39D700	Hidden Sectors (before the boot sector)
0x20	DWORD	0x7F324E00	Large Sectors (total number of sectors)
0x24	DWORD	0x83130000	Sectors Per FAT (FAT32 only).

Quản lý hệ thống file trên máy PC

Cấu trúc BPB của FAT32 (tt) :

Byte Offset	Length	Sample Value	Field Name and Definition
0x28	WORD	0x0000	Extended Flags
0x2A	WORD	0x0000	File System Version (FAT32 only).
0x2C	DWORD	0x02000000	Root Cluster Number
0x30	WORD	0x0100	File System Information Sector Number
0x34	WORD	0x0600	Backup Boot Sector
0x36	12 bytes	0x0000...	Reserved

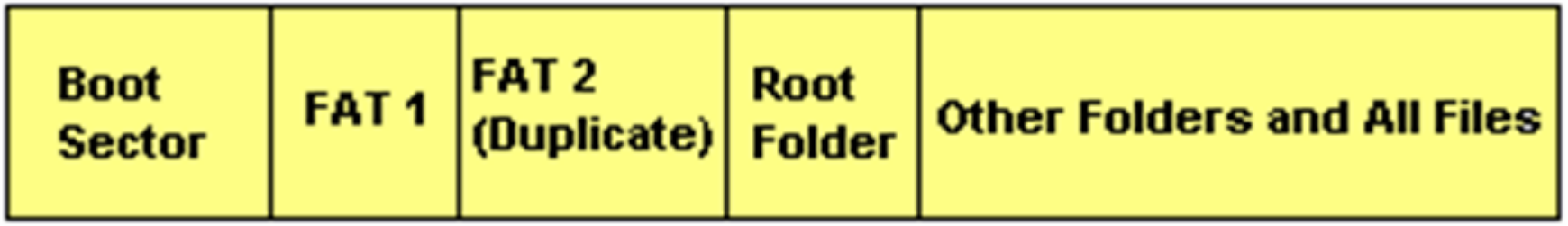
Quản lý hệ thống file trên máy PC

Cấu trúc Extended BPB của FAT32 :

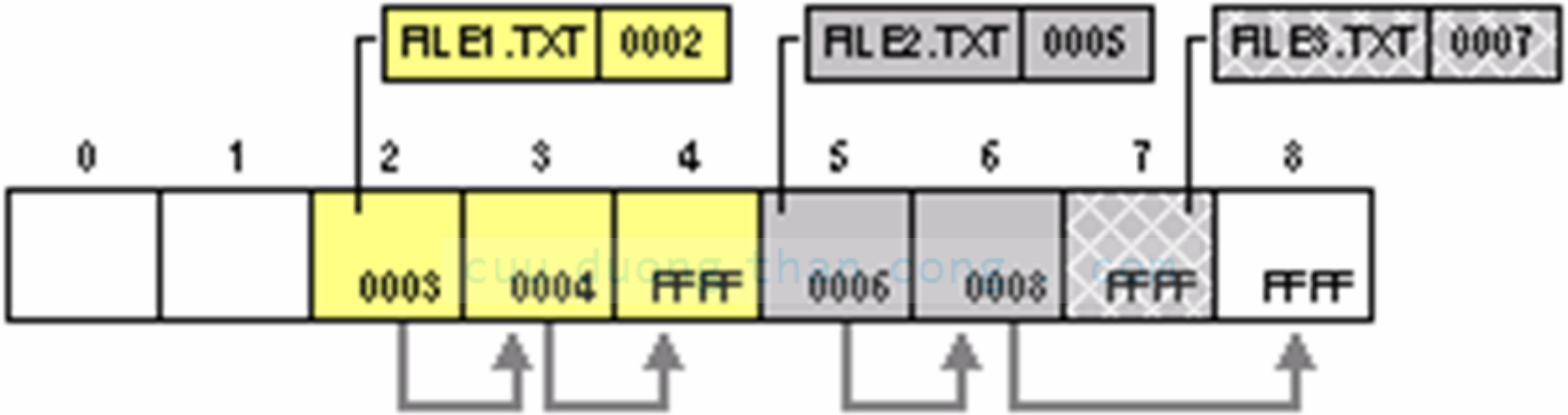
Byte Offset	Length	Sample Value	Field Name and Definition
0x40	BYTE	0x80	hard disks -> 0x80
0x41	BYTE	0x00	Reserved
0x42	BYTE	0x29	Extended Boot Signature
0x43	DWORD	0xA88B3652	Volume Serial Number
0x47	11 bytes	NO NAME	Volume Label
0x52	LONGLONG	FAT32	System ID, chuỗi “FAT32”.

Quản lý hệ thống file trên máy PC

Cấu trúc thông tin của partition FAT32 :

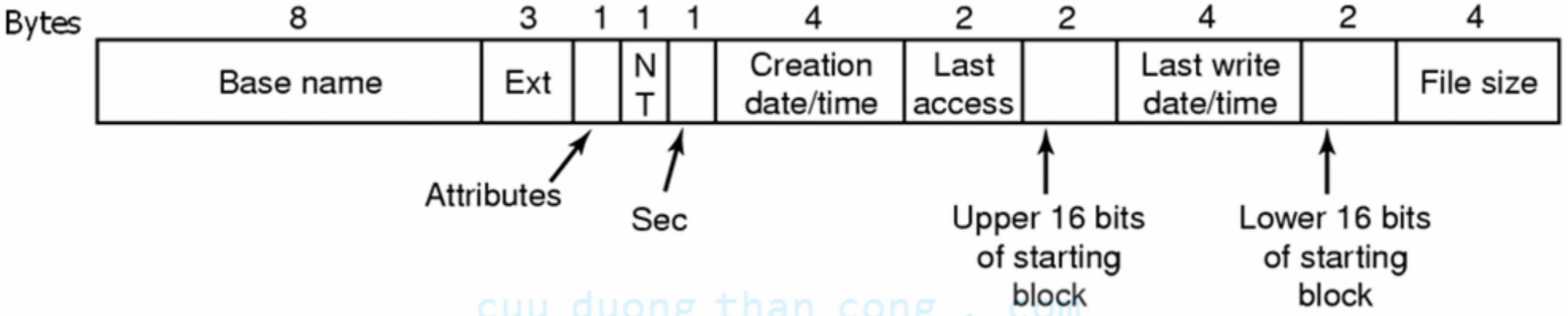


Truy xuất các cluster dữ liệu của từng file dựa vào entry thư mục và bảng FAT

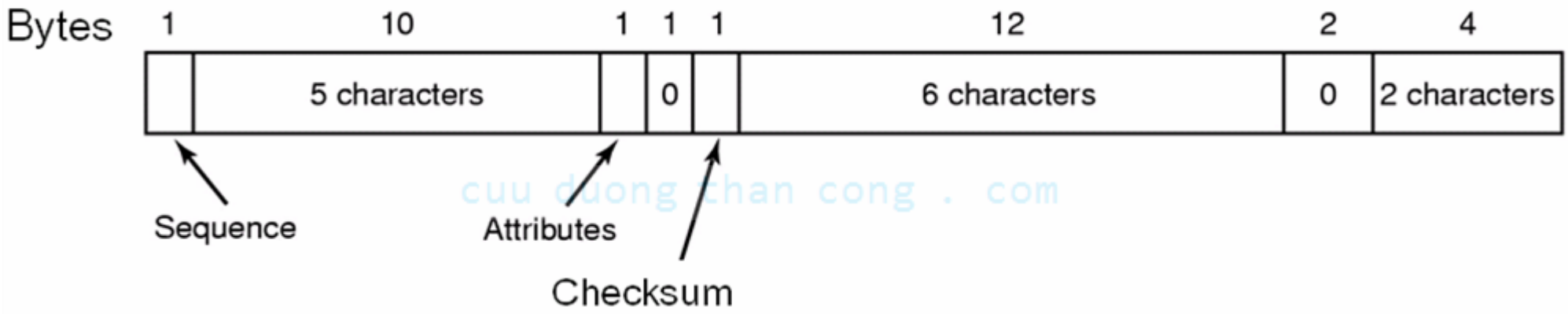


Quản lý hệ thống file trên máy PC

Cấu trúc entry thư mục (entry chính) của partition FAT32 :



Cấu trúc entry thư mục (entry nới rộng) của partition FAT32 :



Quản lý hệ thống file trên máy PC

Thí dụ về các entry thư mục FAT32 được dùng để quản lý file có tên là "The quick brown fox jumps over the lazydog"

68	d o g	A	0	CK		0	
3	o v e	A	0	CK	t h e l a	0	z y
2	w n f o	A	0	CK	x j u m p	0	s
1	T h e q	A	0	CK	u i c k b	0	r o
T	H E Q U I ~ 1	A	NT	S	Creation time / Last acc / Upp / Last write	Low	Size

Bytes