

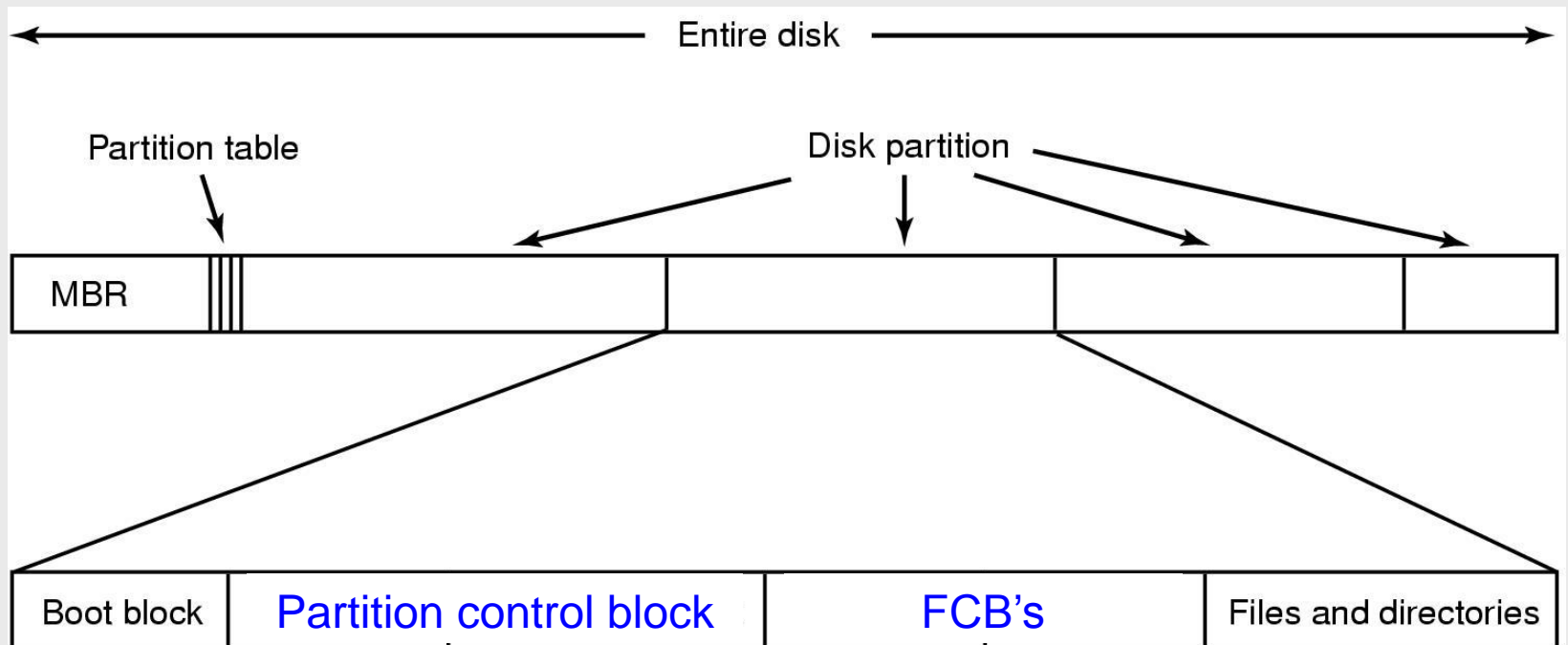
Chương 10: Hệ Thống File

10.B

- Hiện thực hệ thống file và thư mục
- Các phương pháp quản lý không gian trống
- Sao lưu và phục hồi

Sơ đồ bố trí (layout) hệ thống file (1/4)

- Tổ chức không gian đĩa (máy tính cá nhân – PC)



Sơ đồ bố trí hệ thống file (2/4)

■ Partition control block

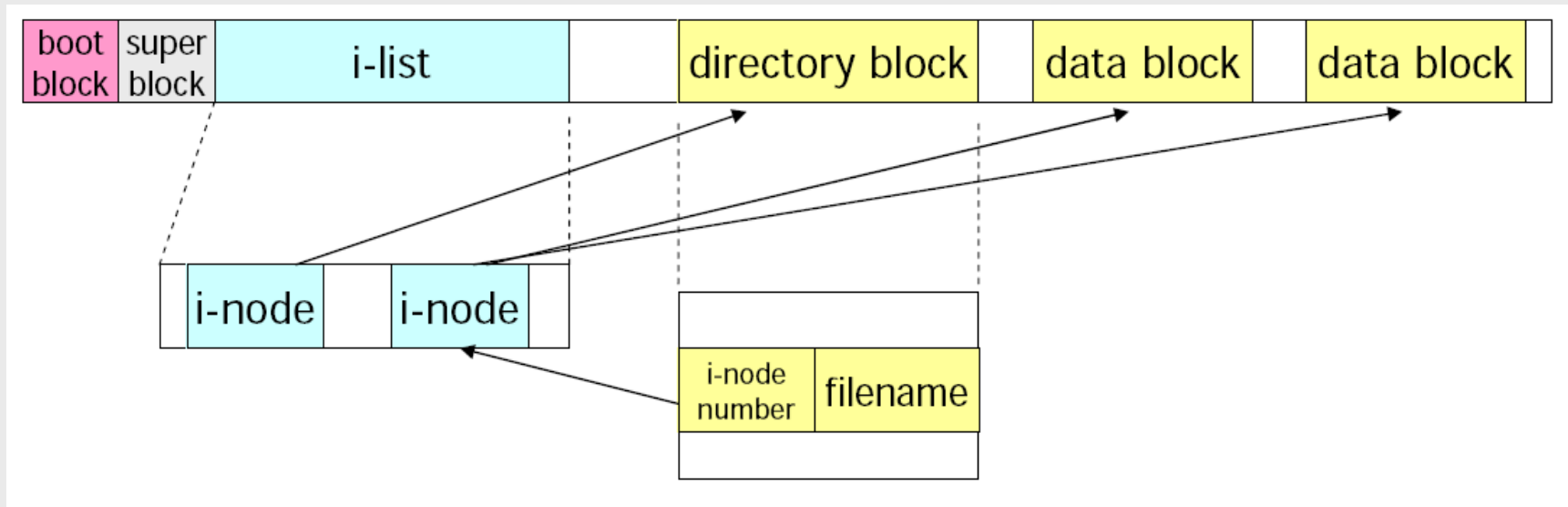
- lưu số lượng block trong partition, kích thước block, số lượng free block hiện thời và các con trỏ chỉ đến chúng,...
- lưu số lượng free FCB hiện thời và các con trỏ chỉ đến chúng,...
- Ví dụ “superblock” trong UNIX File System

■ File control block (FCB): mỗi file được quản lý thông qua FCB của nó

- lưu các thông tin về file, kể cả các con trỏ chỉ đến các data block của nó
- Ví dụ “i-node” trong UNIX File System:

Sơ đồ bố trí hệ thống file (3/4)

- Layout của một partition chứa hệ thống file UNIX



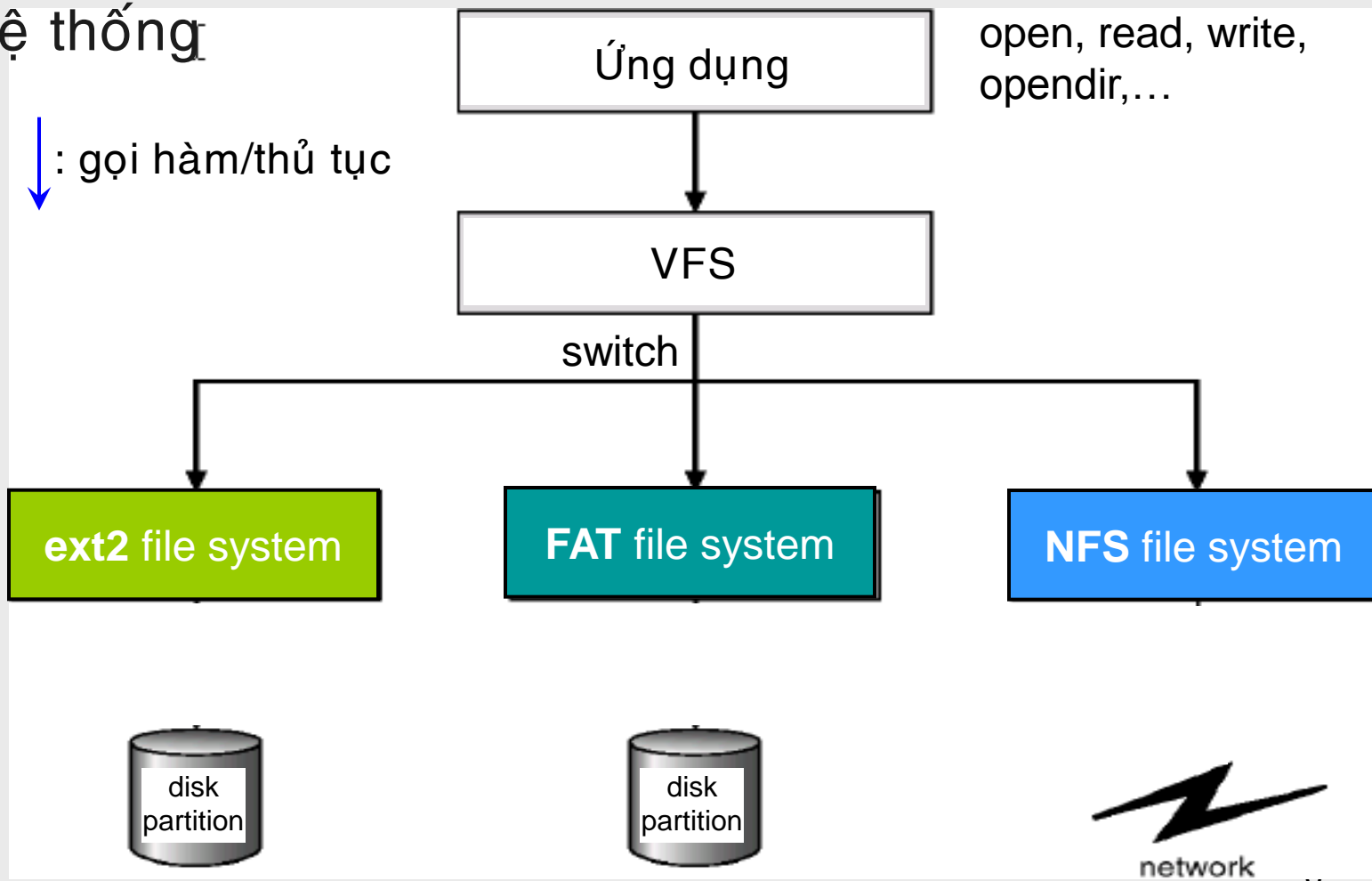
Sơ đồ bố trí hệ thống file (4/4)

- **FAT** dùng để chỉ bảng FAT và cũng dùng để chỉ hệ thống file
- Layout của một partition chứa hệ thống file FAT



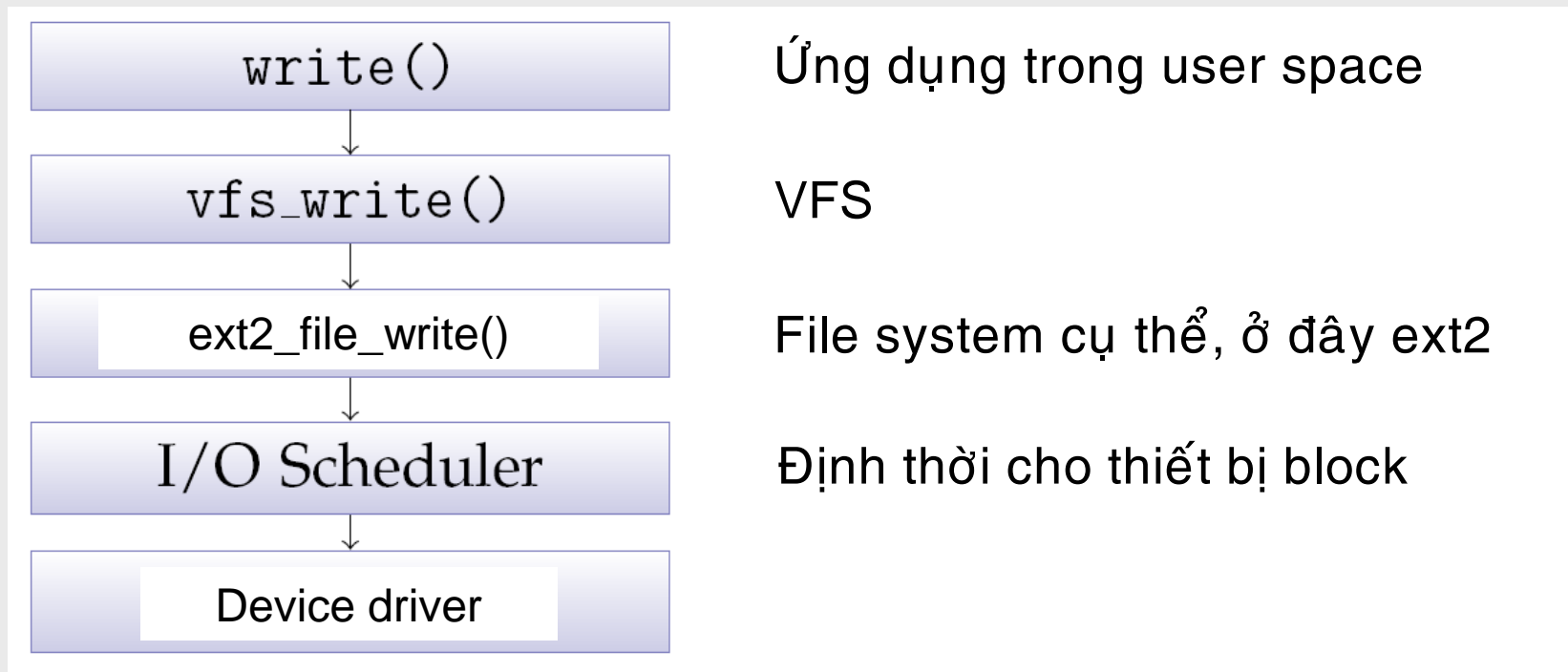
VFS (Virtual File System)

- Cung cấp giao diện đồng nhất cho ứng dụng độc lập với file system cụ thể → nhiều file system khác nhau trong cùng hệ thống



VFS (Virtual File System)

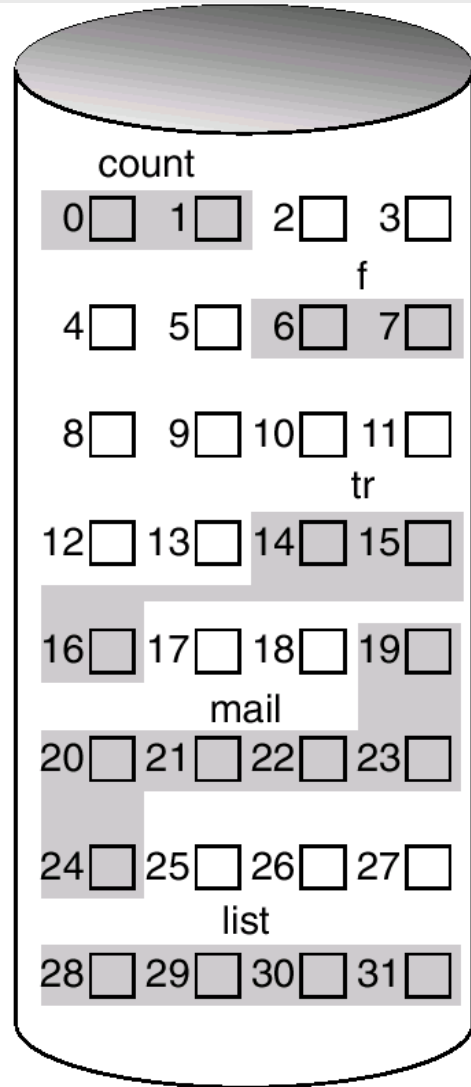
- Vị trí của VFS và file system software trong I/O call path (Linux)



Hiện thực file

- Cấp phát không gian lưu trữ cho file/directory, mục tiêu:
 - sử dụng không gian đĩa hữu hiệu
 - truy cập file nhanh
- Nếu số lượng và kích thước file không thay đổi động thì hiện thực file như thế nào?
- Các phương pháp cấp phát phổ biến
 - Cấp phát *liên tục* (contiguous allocation)
 - Cấp phát *theo danh sách liên kết* (linked list allocation)
 - Cấp phát *dùng chỉ mục* (indexed allocation)

Cấp phát liên tục

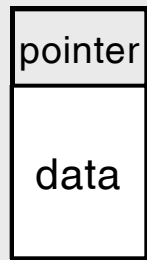


directory

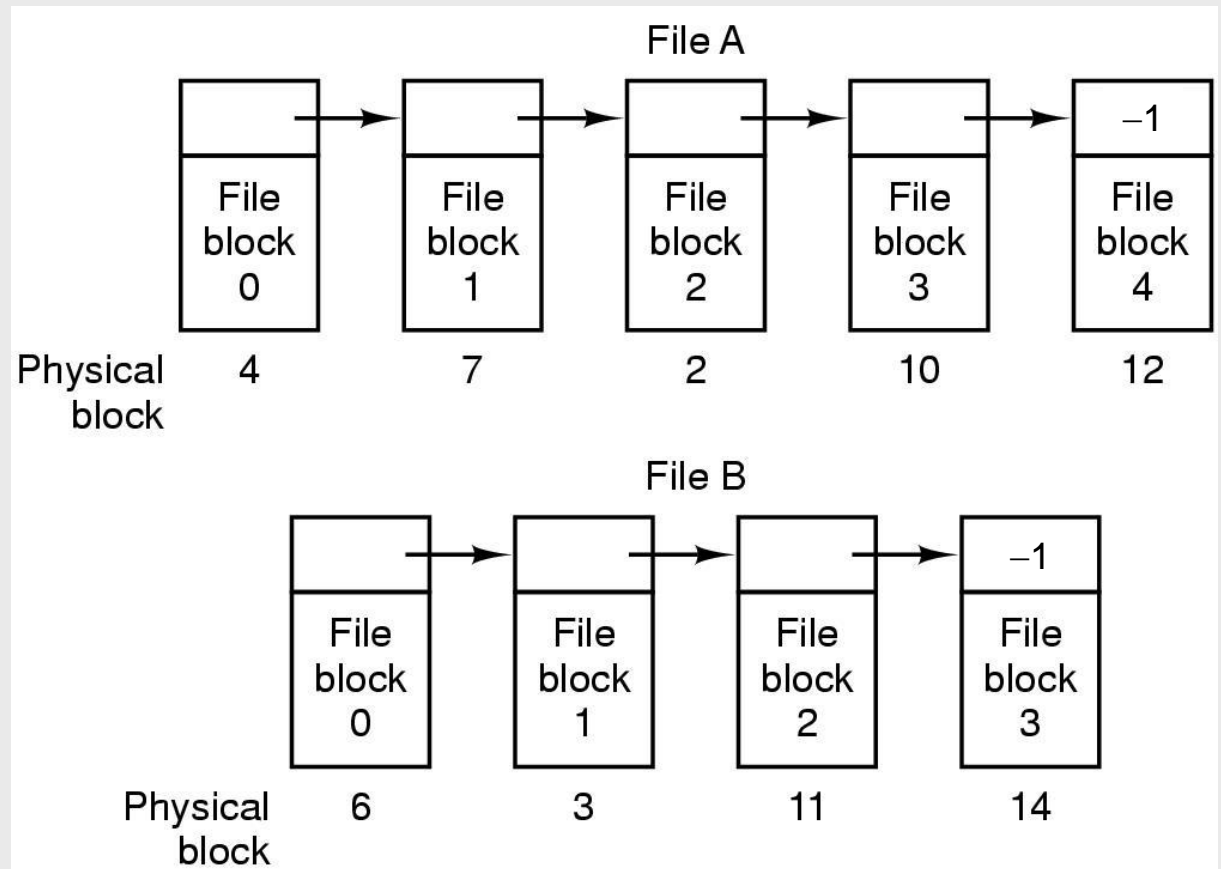
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

- Seek time? Di chuyển đầu đọc?
- Có thể truy xuất ngẫu nhiên một block của file: $\text{block nr} = \text{start} + \text{block offset}$
- Phân mảnh ngoại
- Vấn đề khi tạo file mới và khi cần thêm block cho file
- Ứng dụng: ISO-9660 (CDROM)

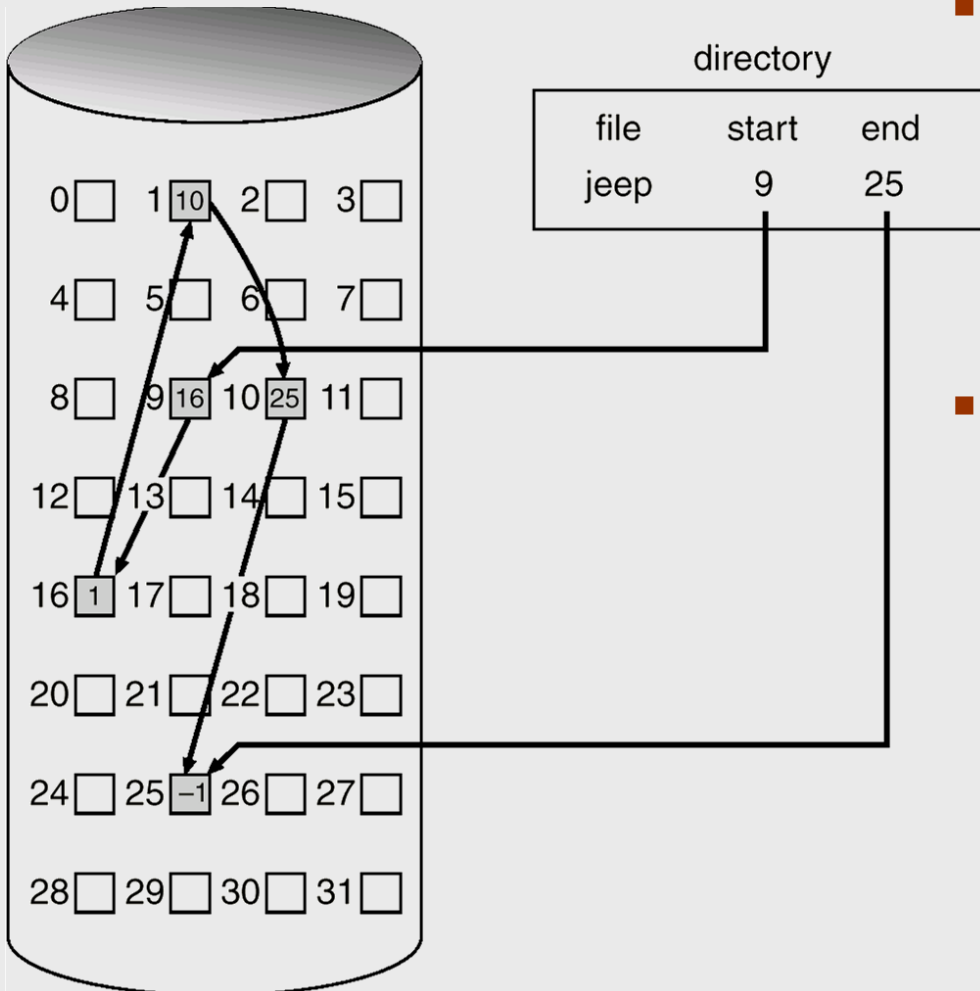
Cấp phát theo danh sách liên kết (1/2)



layout của block



Cấp phát theo danh sách liên kết (2/2)



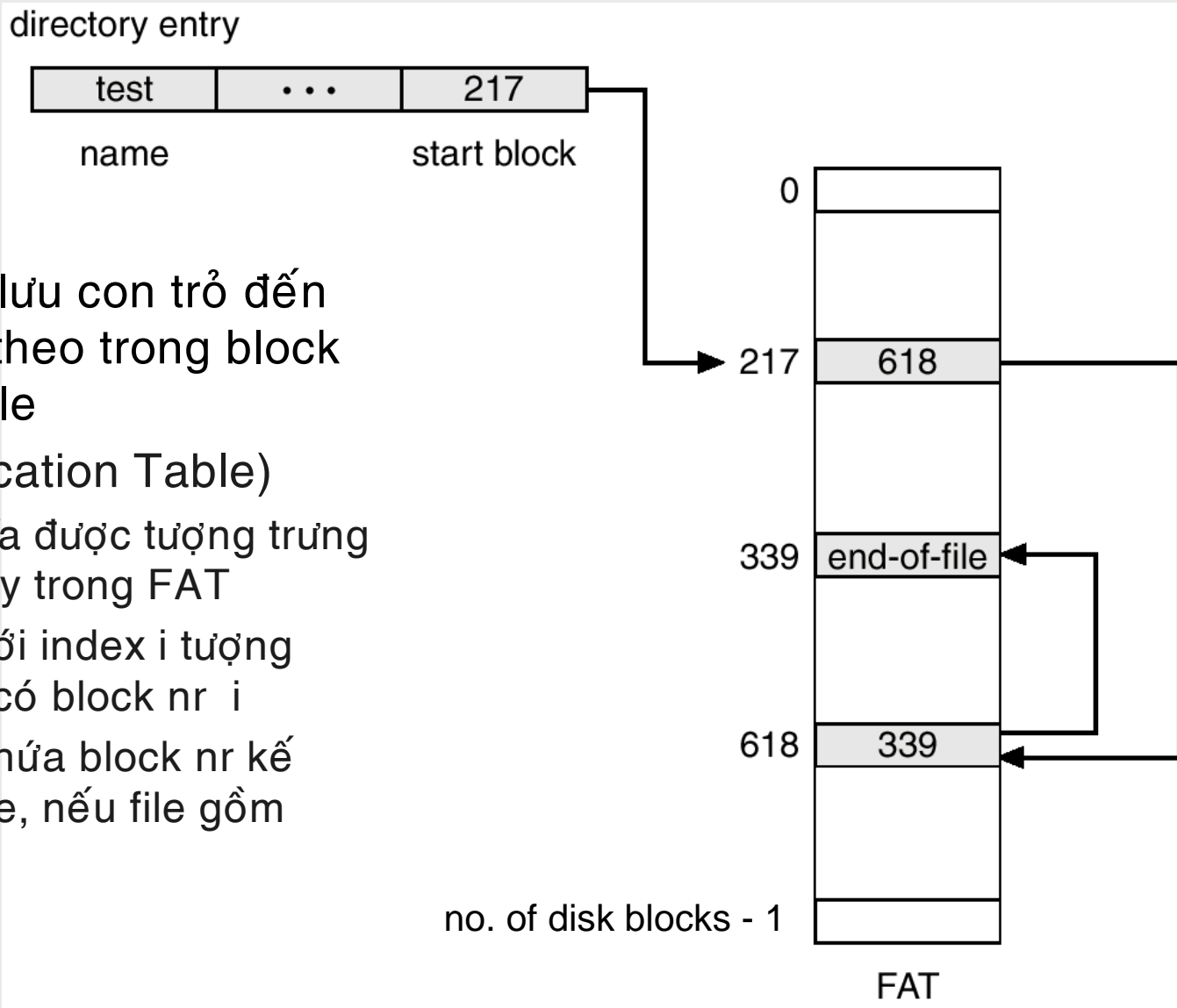
■ Ưu điểm

- Dễ dàng thêm block cho file khi cần
- Quản lý không gian trống bằng danh sách liên kết
- Không có phân mảnh ngoại

■ Nhược điểm

- Chỉ truy xuất hiệu quả đối với sequential-access file
- Tốn không gian lưu trữ các con trỏ
- Độ tin cậy: pointer trong block có thể bị hỏng

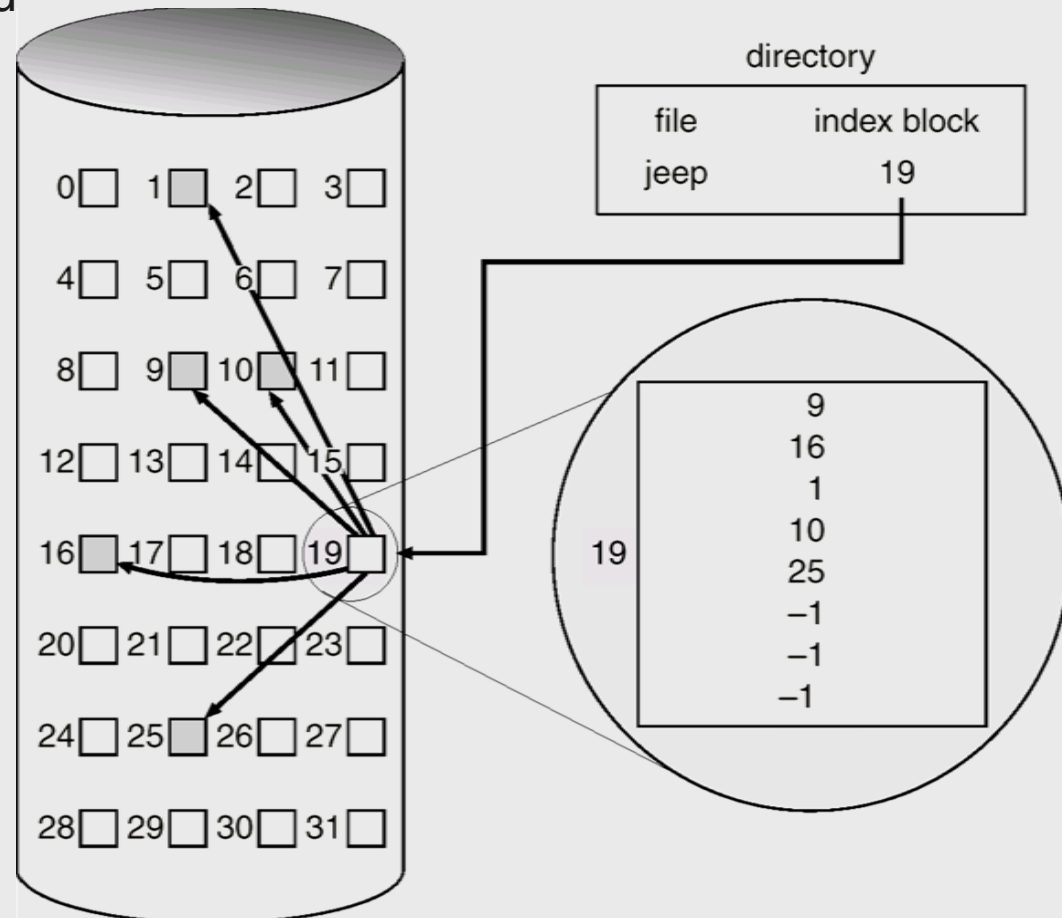
FAT – một hiện thực của cấp phát theo danh sách liên kết:



- Nhưng không lưu con trỏ đến file block tiếp theo trong block chứa dữ liệu file
- **FAT** (File Allocation Table)
 - Mỗi block đĩa được tượng trưng bởi một entry trong FAT
 - FAT entry với index i tượng trưng block có block nr i
 - FAT entry chứa block nr kế tiếp trong file, nếu file gồm nhiều block

Cấp phát dùng chỉ mục (1/2)

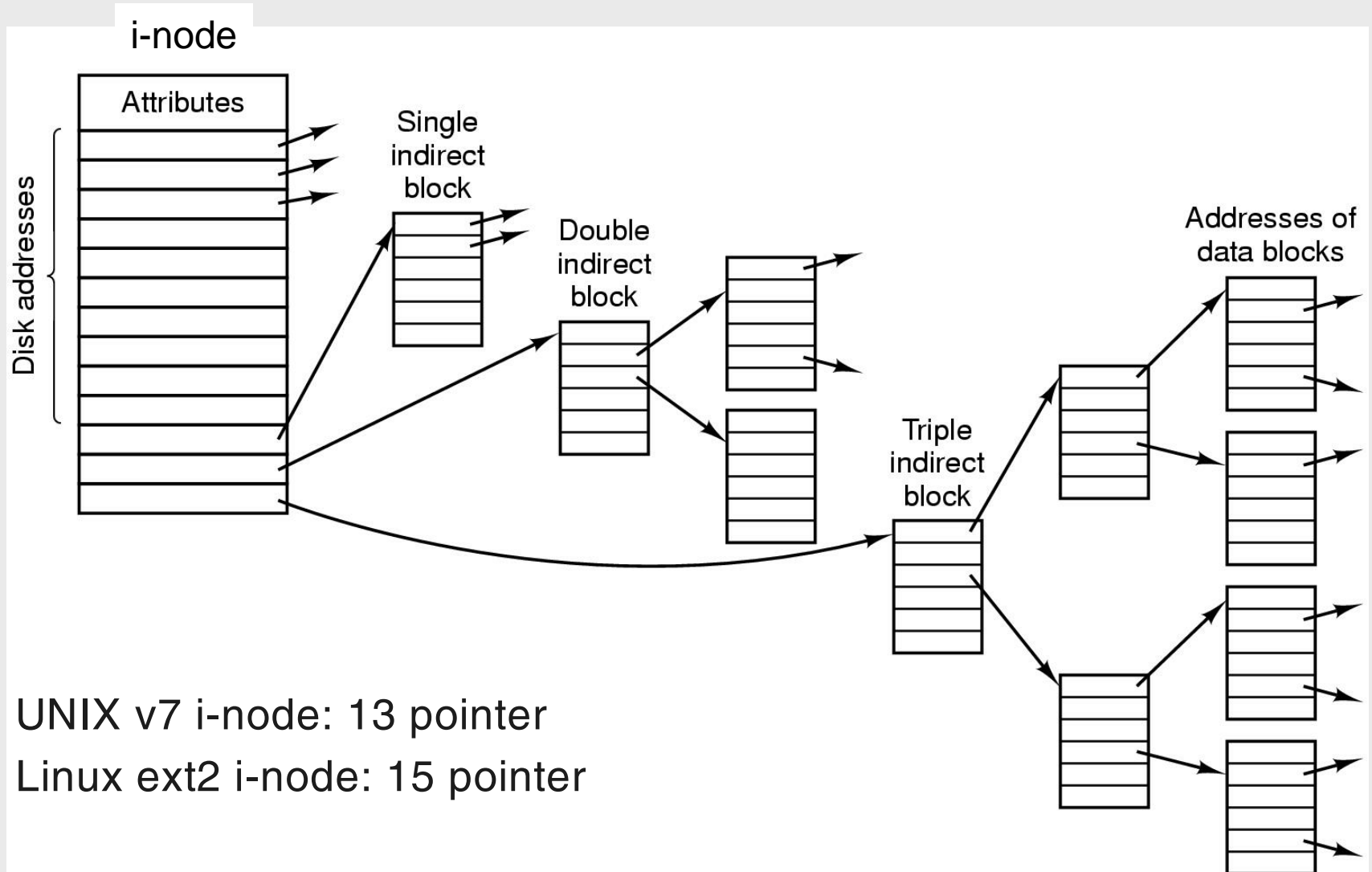
- **Bảng index** (index block)
 - chứa địa chỉ các block của file
 - thứ tự các địa chỉ cũng là thứ tự các block của file



Cấp phát dùng chỉ mục (2/2)

- Ưu điểm
 - Random và sequential access
 - Không có phân mảnh ngoại
- Khuyết điểm
 - Tốn không gian lưu trữ bảng index khi file có kích thước chỉ vài block
- Vấn đề: kích thước index block bao nhiêu là phù hợp?
 - Giải quyết: multilevel index \Rightarrow i-node

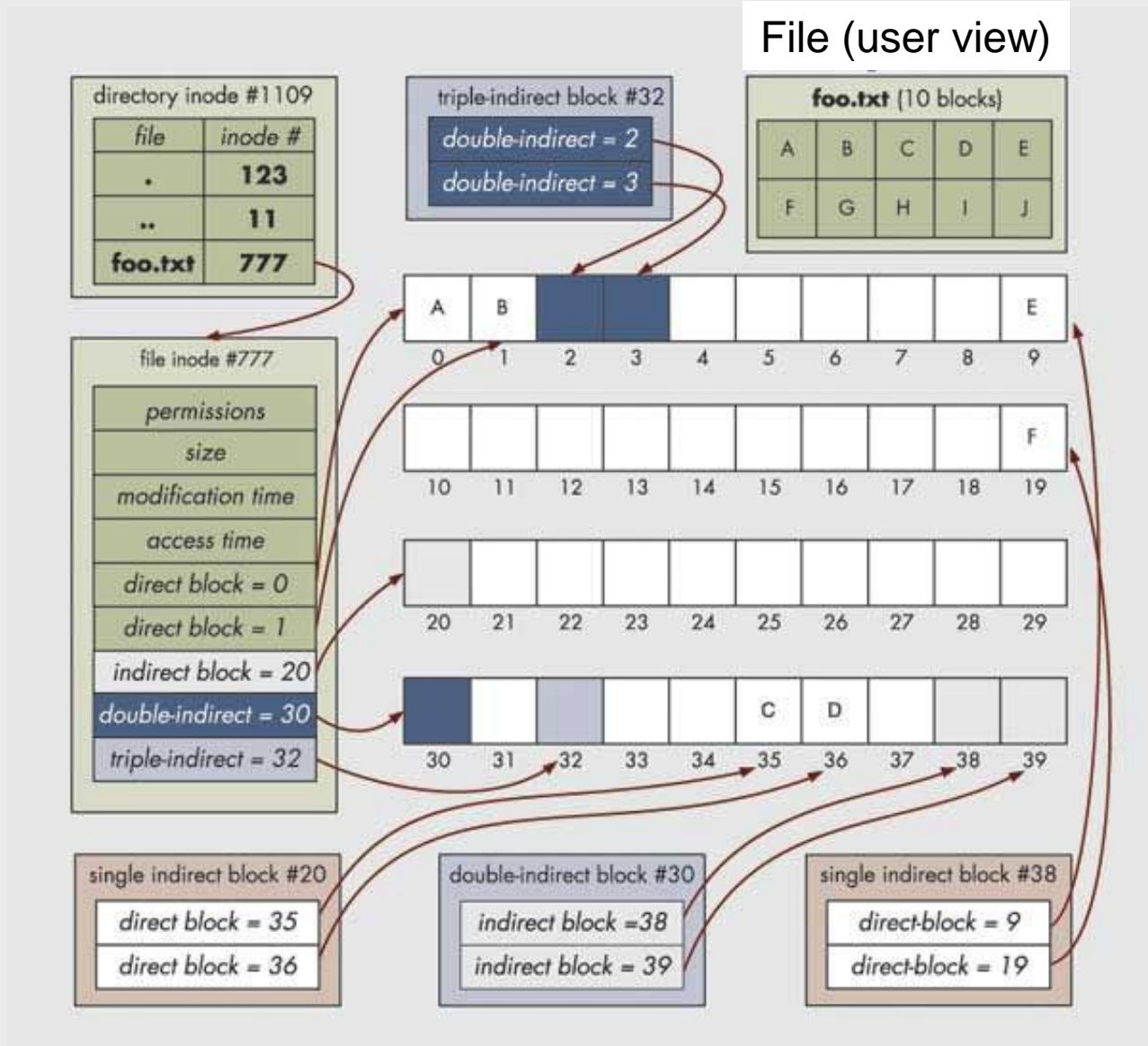
i-node – một hiện thực của index block



- UNIX v7 i-node: 13 pointer
- Linux ext2 i-node: 15 pointer

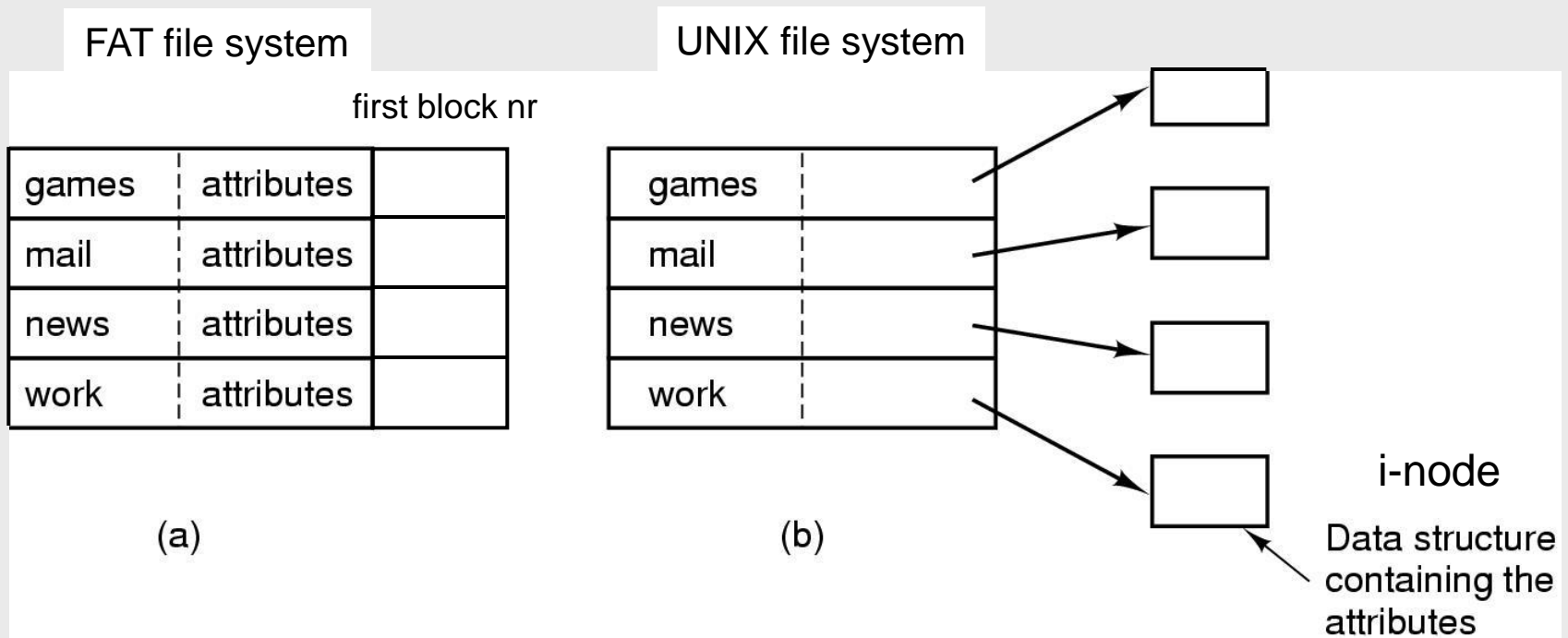
Hiện thực file dùng i-node

- Ví dụ



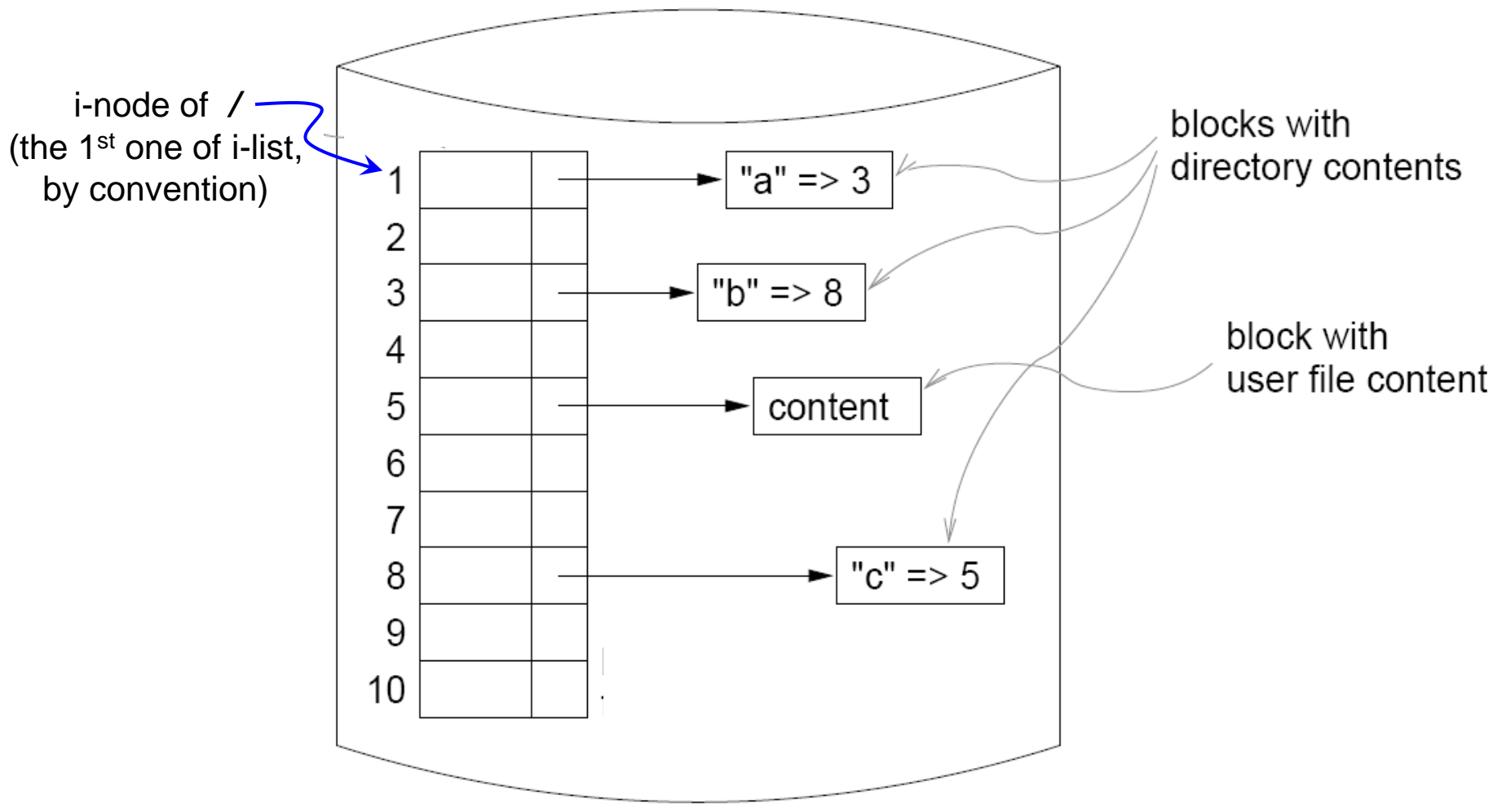
Hiện thực thư mục

- Thư mục được dùng để chứa bảng ánh xạ từ tên file (chuỗi ký tự ASCII) đến thông tin cần thiết để định vị các block dữ liệu của file
- Tổ chức thư mục
 - Danh sách tuyến tính (array hay linear list), bảng băm,...



Duyệt path name để lấy block nr của file

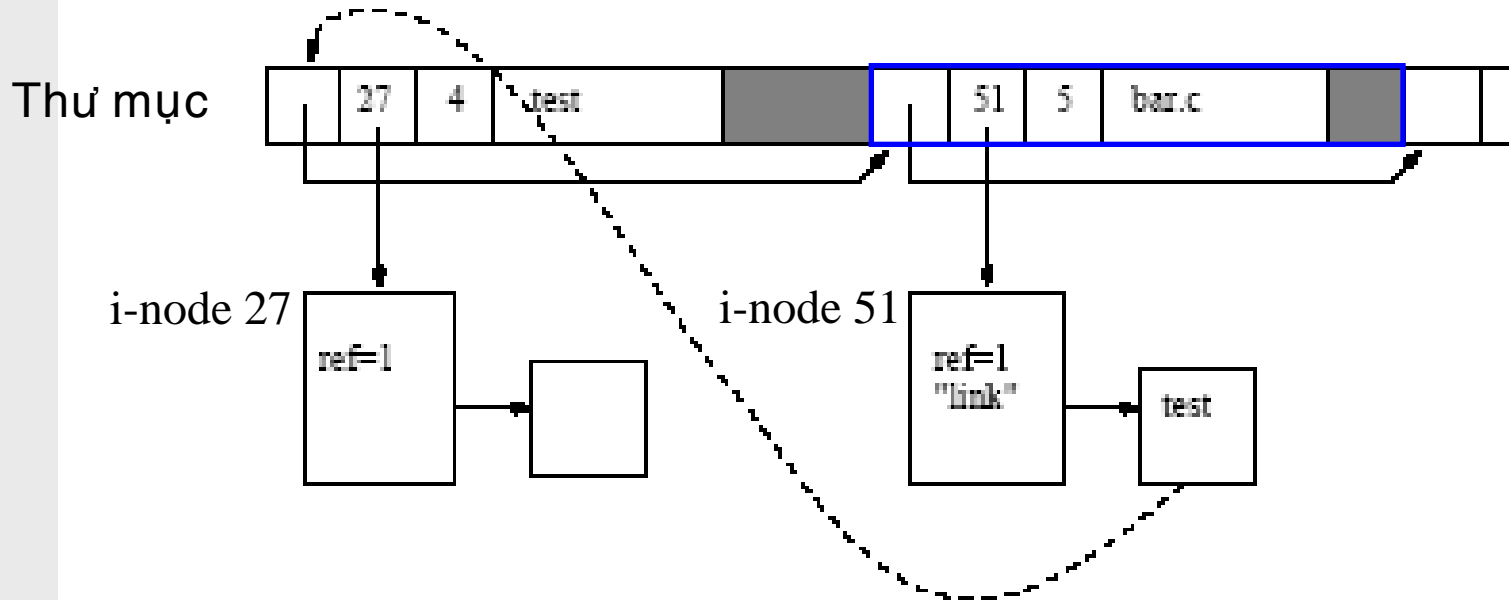
- Ví dụ: Xác định các block dữ liệu của file /a/b/c



i-node: chia sẻ file (1/2)

Soft Links

Association of a name with a name: inode refers to a name.



which gives

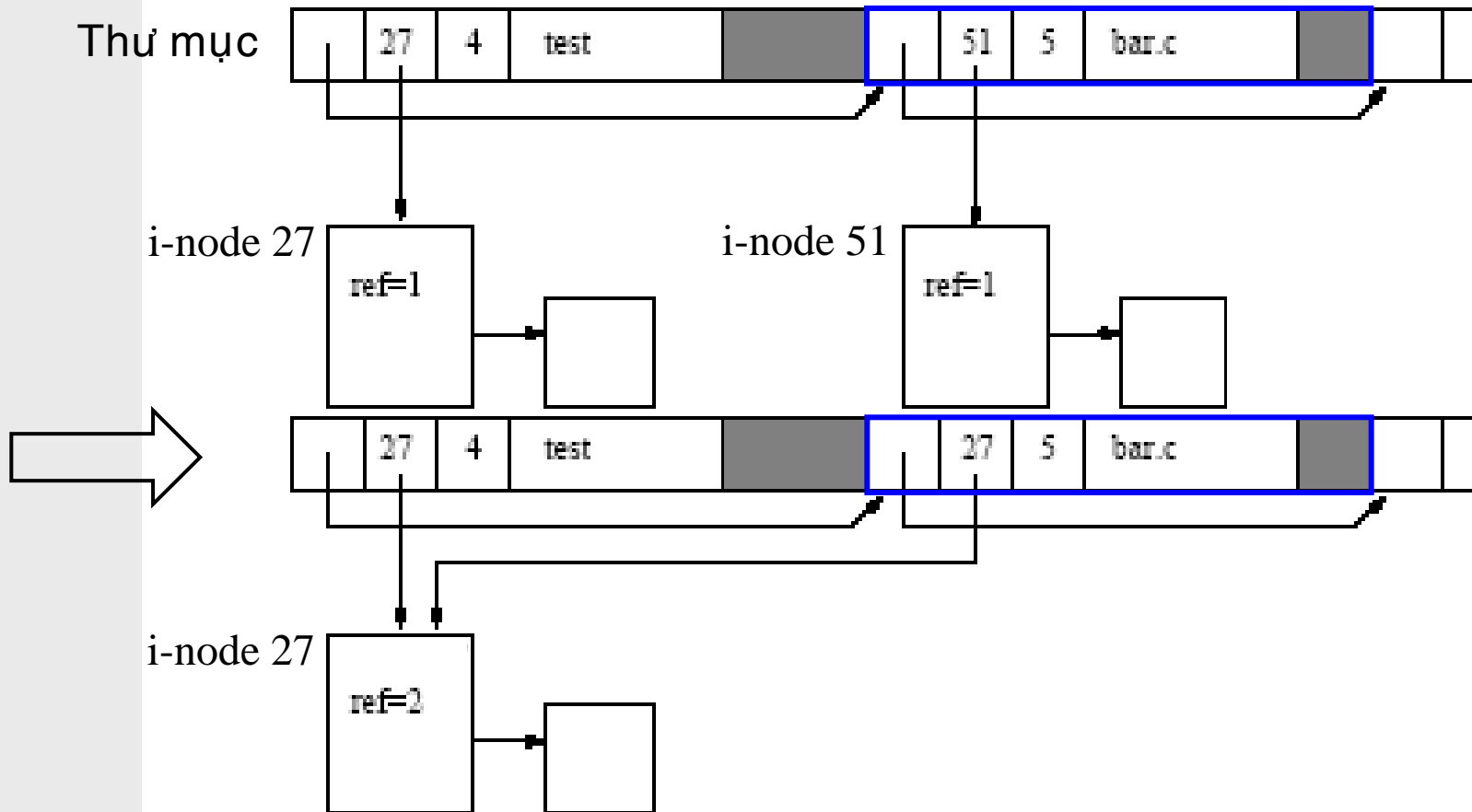
```
> ls -li
```

```
51 lrwxrwxrwx 1 (...) bar.c -> test
27 -rw-r--r-- 1 (...) test
```

i-node: chia sẻ file (2/2)

Hard Links

Association of a name with a file: inode refers to data.

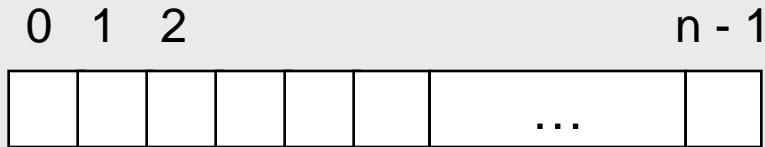


Quản lý không gian trống

Các phương pháp

- Bit vector (bit map)
- Linked list
- Grouping
- Counting

Phương pháp bit vector (bit map)



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

Ví dụ:

bit vector 00111100...

↔

block 0, 1 trống

block 2, 3, 4, 5 đã được cấp

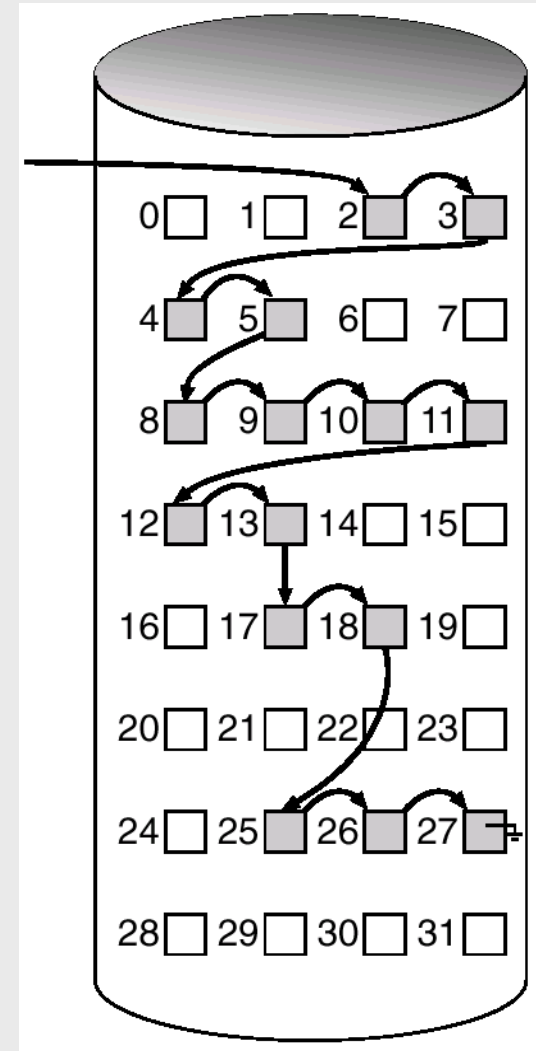
block 6, 7 trống

...

- Ưu: Đơn giản và hiệu quả khi cần tìm khối trống đầu tiên hoặc chuỗi khối trống liên tục
 - Thao tác trên bit
- Khuyết: Cần không gian lưu trữ. Ví dụ
 - Kích thước block = 2^{12} byte
 - Kích thước đĩa = 2^{30} byte
 - $n = 2^{30}/2^{12} = 2^{18}$ bit (32KB)

Phương pháp dùng linked list

- Phương pháp
 - Liên kết các khối trống với nhau
 - Chỉ cần giữ con trỏ đến khối nhớ trống đầu tiên trên đĩa hoặc cache trong bộ nhớ chính để tăng tốc
- Ưu: Ít lãng phí không gian đĩa
- Nhược: Không hiệu quả; trong trường hợp xấu nhất phải duyệt toàn bộ đĩa để tìm không gian trống liên tục



Grouping và counting (1/2)

■ Phương pháp **grouping**

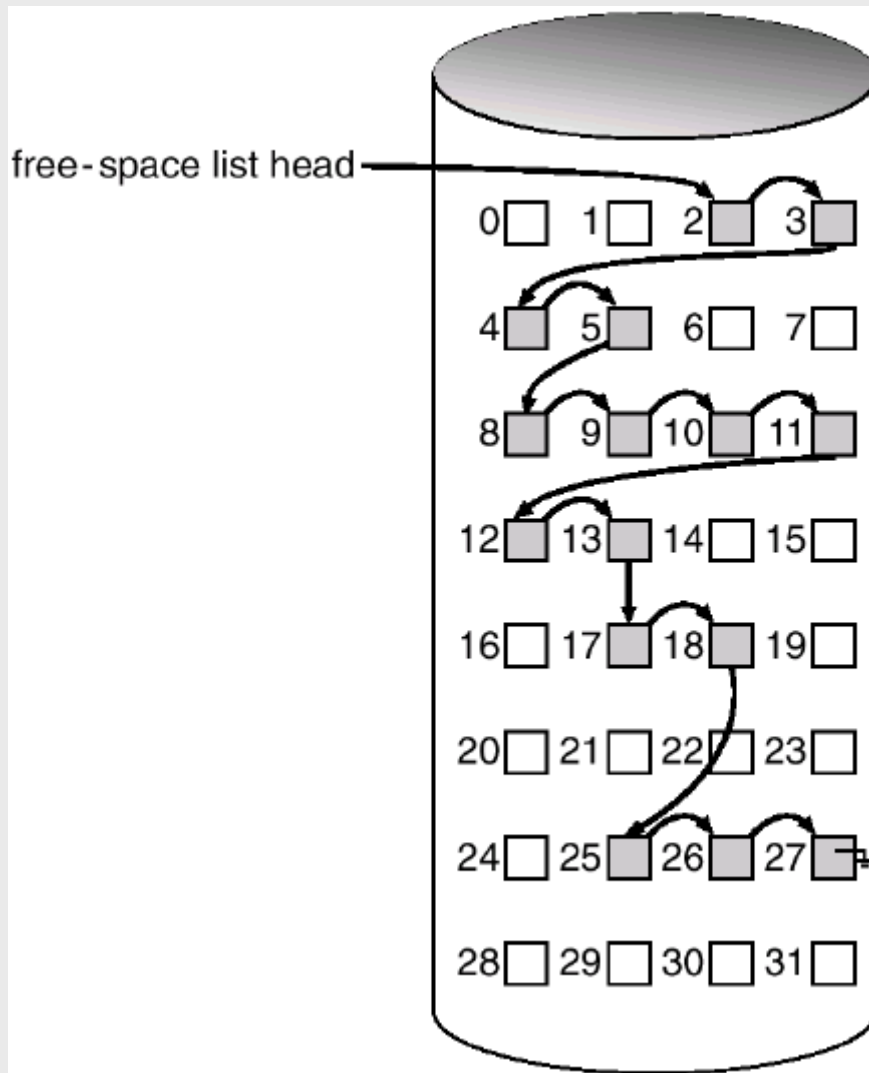
- Địa chỉ của n khối trống được lưu trong khối trống đầu tiên.
- Khối nhớ thứ n chứa địa chỉ của n khối nhớ trống kế tiếp.

■ Phương pháp **counting**

- Tổ chức bảng chỉ mục
 - ▶ mỗi entry: địa chỉ của khối trống đầu tiên trong nhóm khối trống liên tục và một số đếm số lượng khối trống.
- Có thể cấp phát hoặc thu hồi đồng thời nhiều khối nhớ liên tục.

Grouping và counting (2/2)

- Ví dụ: Phương pháp linked list



- Phương pháp grouping: $n = 3$

Block 2 lưu 3, 4, 5

Block 5 lưu 8, 9, 10

Block 10 lưu 11, 12, 13

Block 13 lưu 17, 28, 25

Block 25 lưu 26, 27

- Phương pháp counting: nội dung index block

2 4

8 6

17 2

25 3

Journaling file system

■ Journaling file system

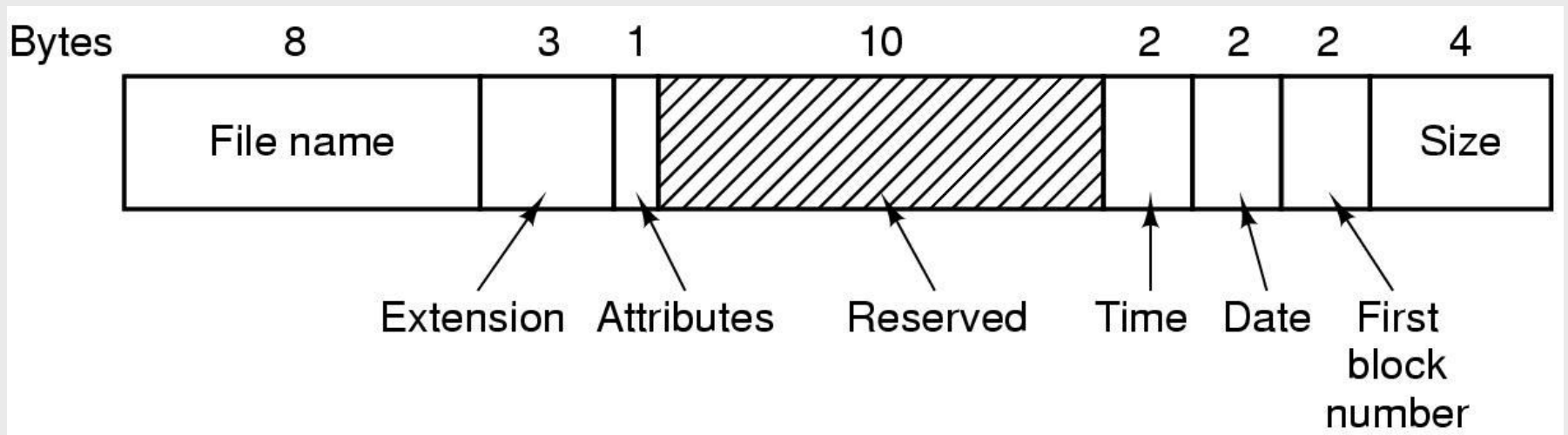
- Ghi nhận các lần cập nhật trên file system thành các giao tác (transaction)
- Mọi transaction đều phải được ghi nhận trong log file
- Một transaction được xem là hoàn tất (commit) \leftrightarrow đã được ghi nhận đầy đủ trong log file (lúc này, file system có thể chưa được cập nhật)
- Khi file system được cập nhật với đầy đủ mọi tác vụ trong transaction thì transaction sẽ được xóa đi trong log file
- Nếu file system bị hỏng \rightarrow hệ điều hành dựa vào các transaction trong log file để sửa chữa

■ Tham khảo thêm Linux-ext3, JFS, NTFS

Phụ lục

MS-DOS File System (1/2)

- MS-DOS directory entry



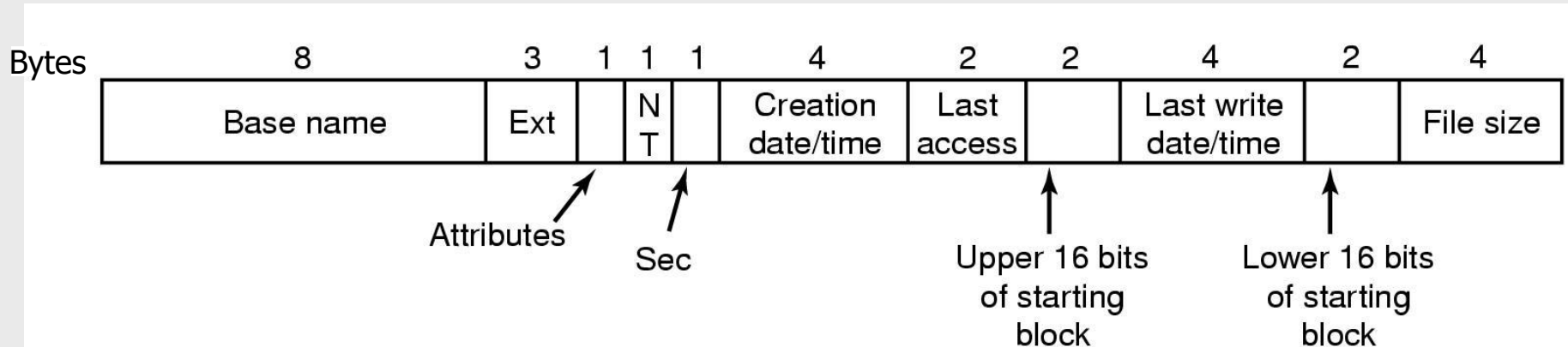
MS-DOS File System (2/2)

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

- Maximum partition for different block sizes
- The empty boxes represent forbidden combinations

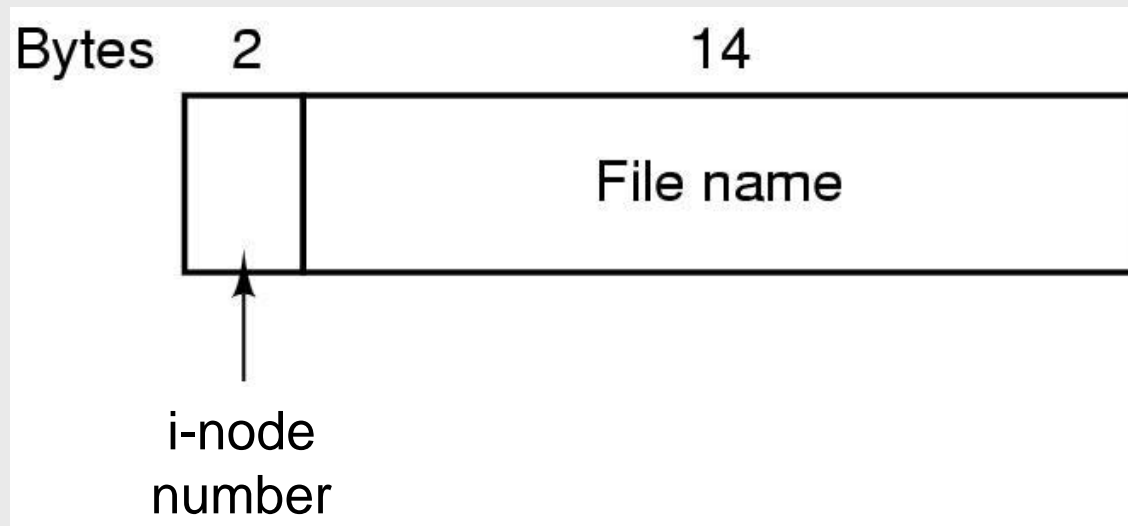
Windows 98 File System (1/3)

- Extended MOS-DOS directory entry used in Windows 98



UNIX V7 File System (1/2)

- A UNIX V7 directory entry



UNIX V7 File System (2/2)

- The steps in looking up `/usr/ast/mbox`

