Hệ thống File (tt)

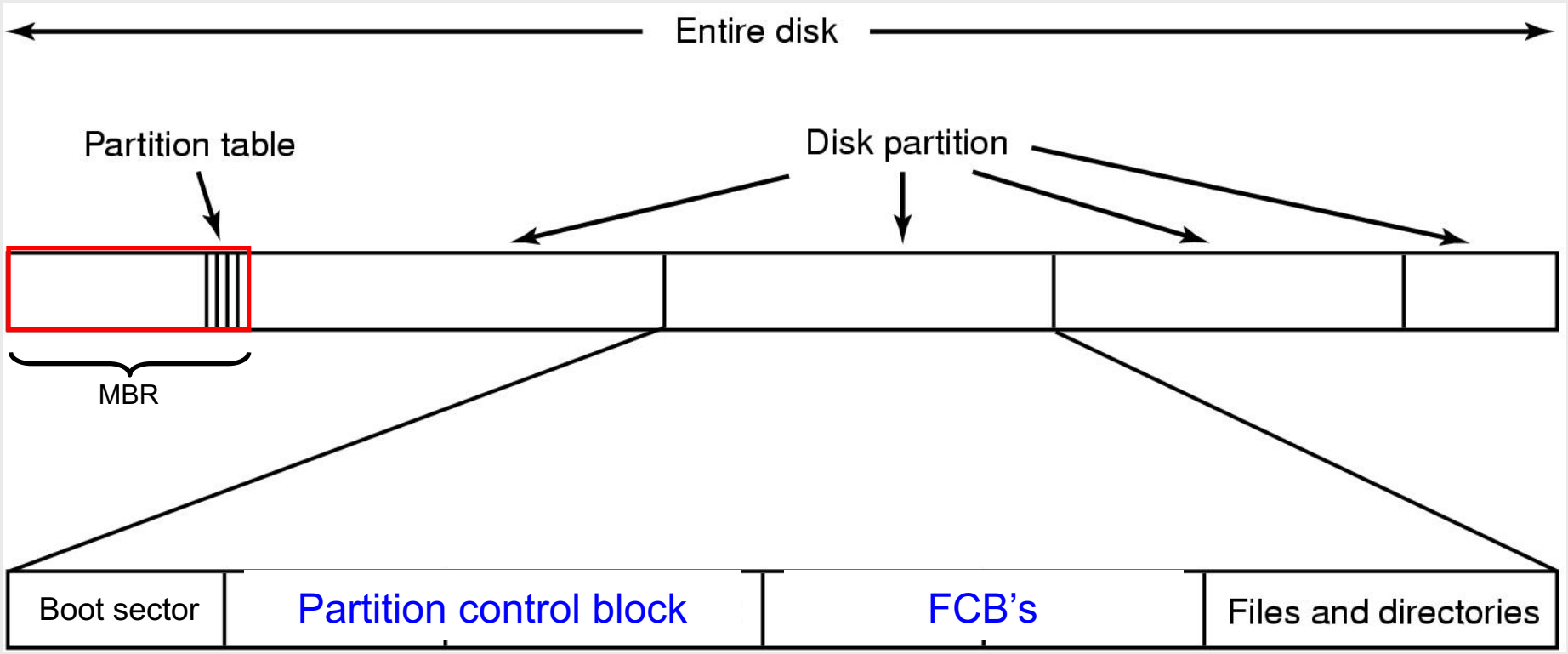
- Hiện thực hệ thống file và thư mục
- Các phương pháp quản lý không gian trống

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

- Kích thước 1 sector là 512 byte
  - Tuy nhiên, theo chuẩn mới kích thước 1 sector là 4K byte
- Hệ thống file (ext2, ext3, NTFS…) sử dụng block, hay còn gọi là cluster, để chỉ không gian đĩa nhỏ nhất cấp phát được cho một file
  - Block (hay cluster) là một chuỗi một số lượng cố định các sector liên tiếp nhau, ví dụ 4 sector
  - Hai block có block number khác nhau thì không giao nhau

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

- Tổ chức không gian đĩa (máy tính cá nhân – IBM PC) tổng quát cho một file system

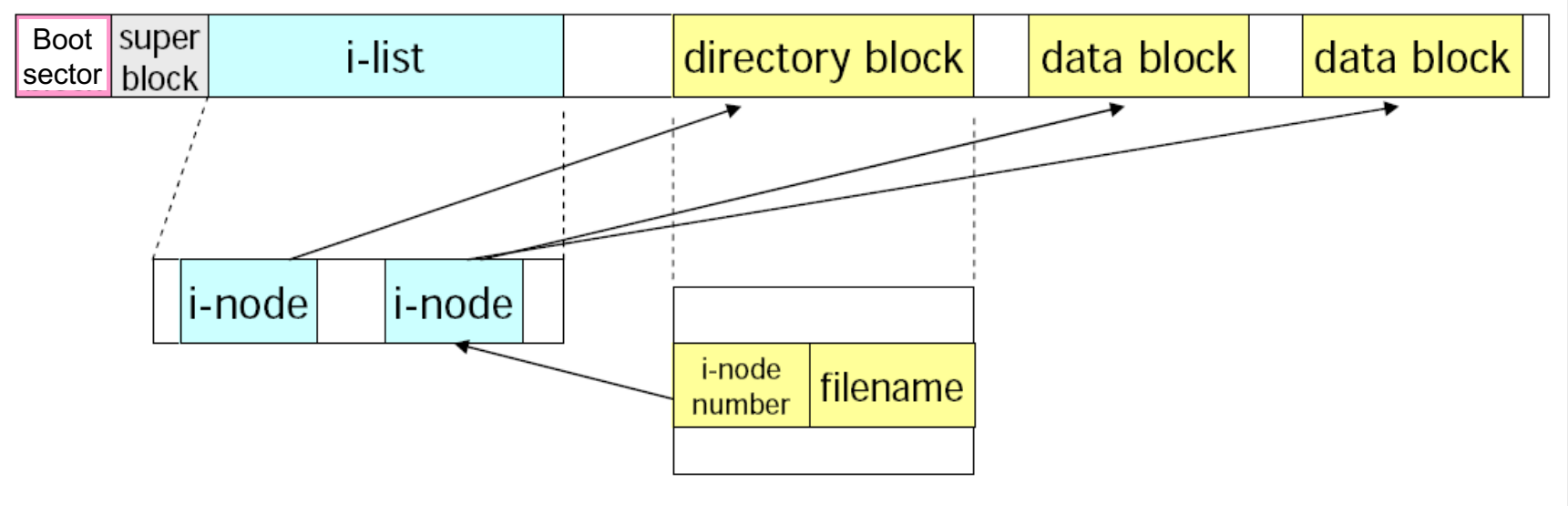


Sơ đồ bố trí hệ thống file (2/4)

- 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
  - i-node ~ FCB

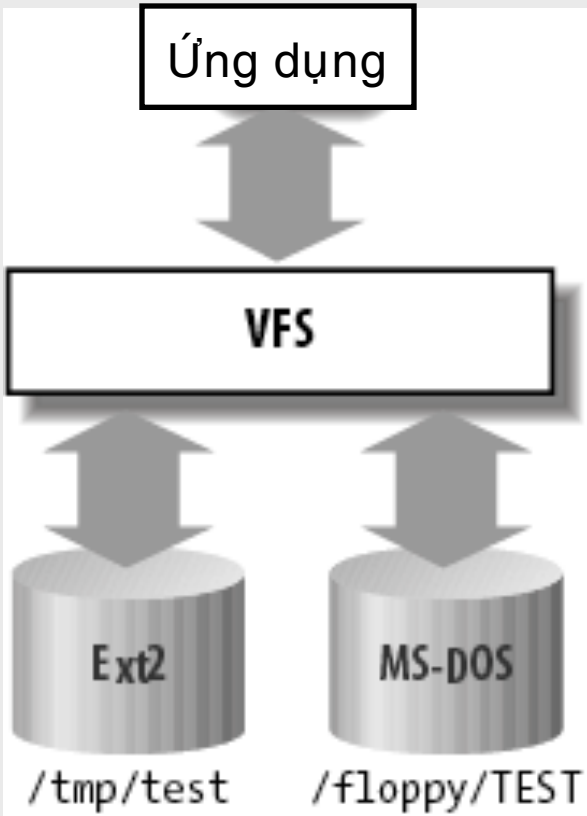


Sơ đồ bố trí hệ thống file (4/4)

- DOS sử dụng file system FAT
- 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 (read, write) cho ứng dụng độc lập với file system cụ thể



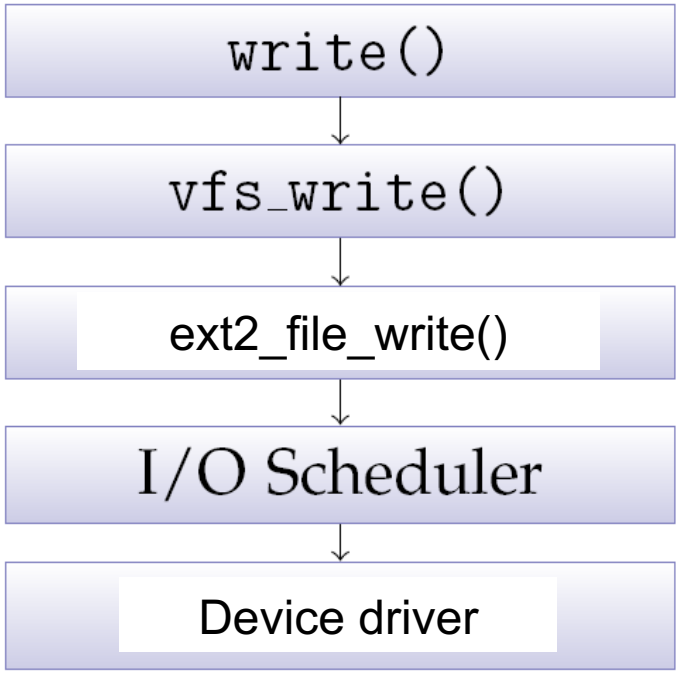
Copy file giữa hai file system khác nhau

```
inf = open("/floppy/TEST", O_RDONLY, 0);
outf = open("/tmp/test",
       O_WRONLY|O_CREAT|O_TRUNC, 0600);
do {
    i = read(inf, buf, 4096);
    write(outf, buf, i);
} while (i);
close(outf);
close(inf);
```

From 'Linux Internals, Summer 2006'

VFS

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



Ứng dụng trong user space

VFS

File system cụ thể, ở đây ext2

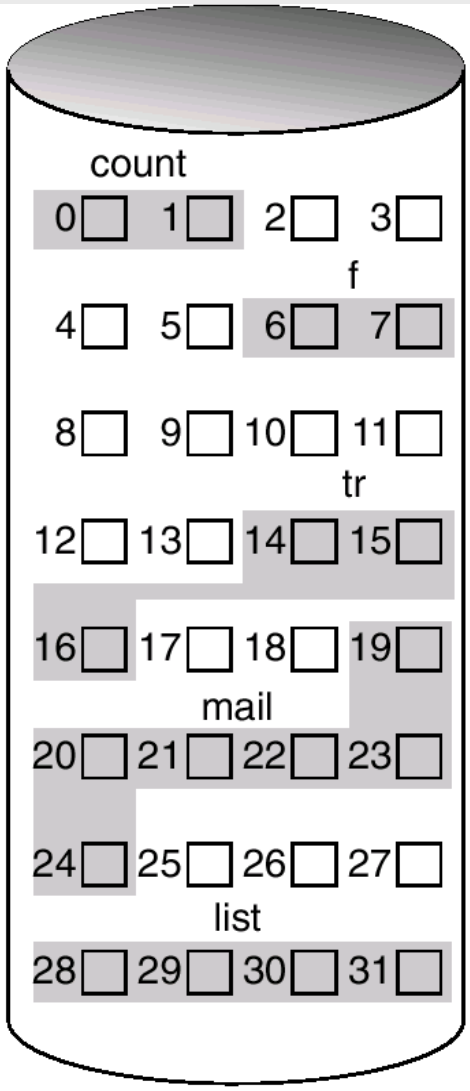
Định thời cho thiết bị block

from Lunde

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

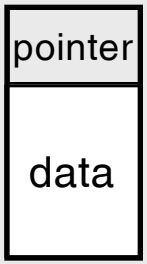


directory

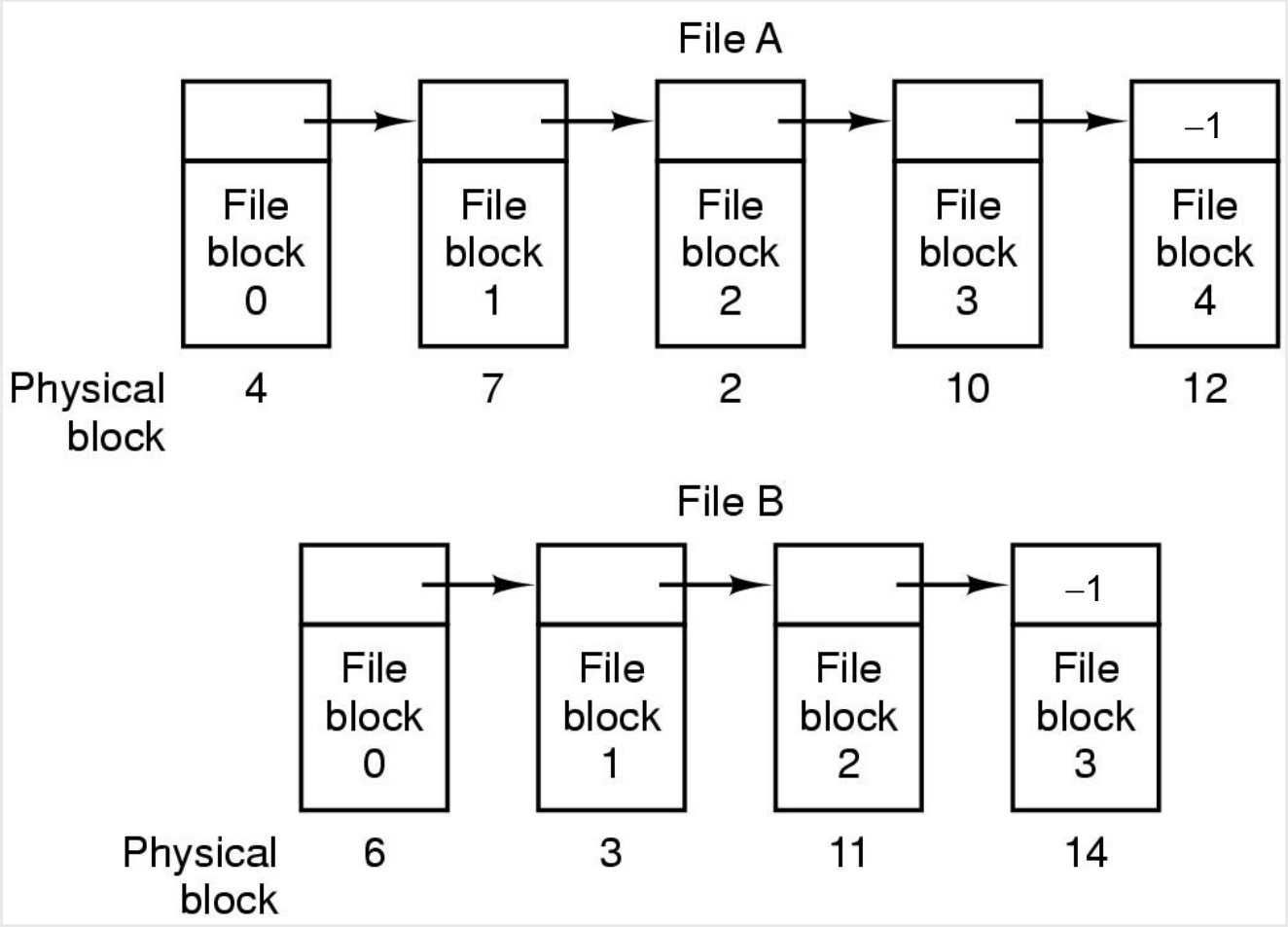
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

- Thời gian di chuyển đầu đọc?
- Có thể truy xuất ngẫu nhiên một block của file: block nr = start + block offset
- Phân mảnh ngoại
  - Có thể gặp khó khăn khi tạo file mới và khi cần thêm block cho file
- Ứng dụng: ISO-9660 (CDROM)

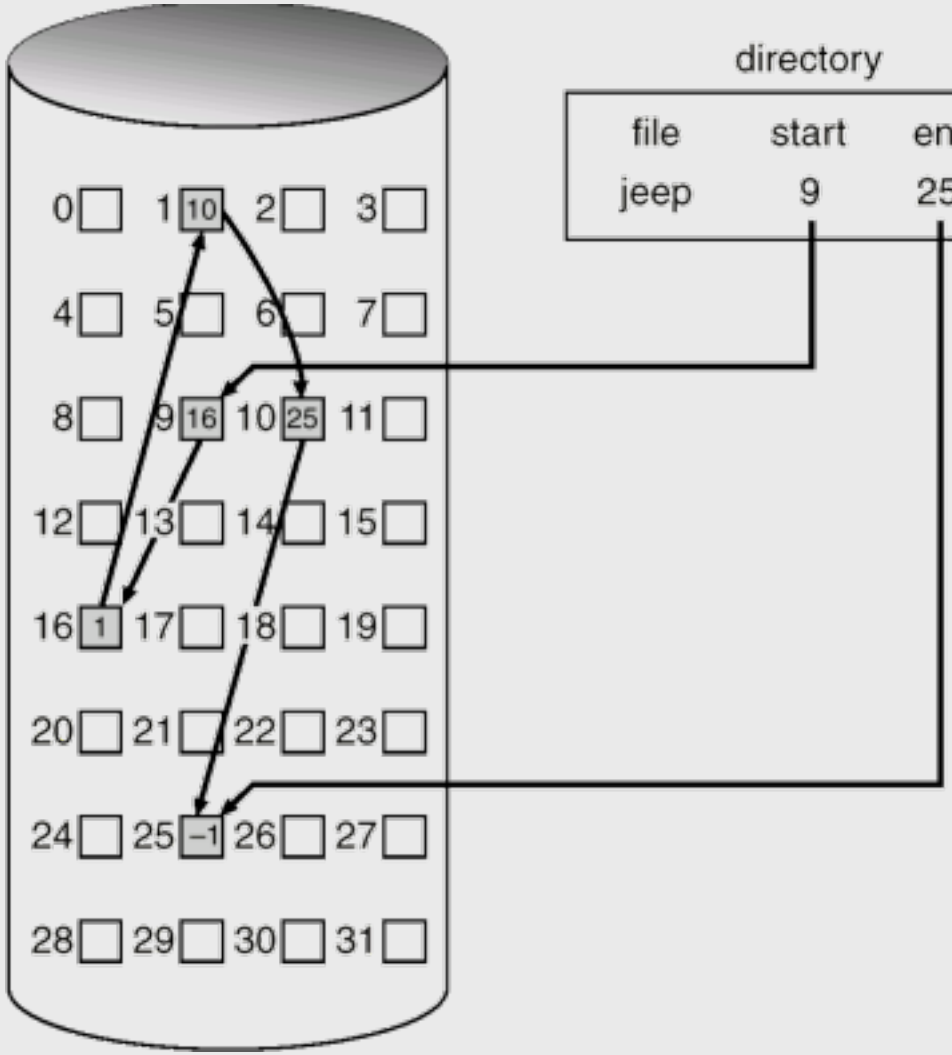
Cấp phát theo danh sách liên kết (1/2)



layout của block



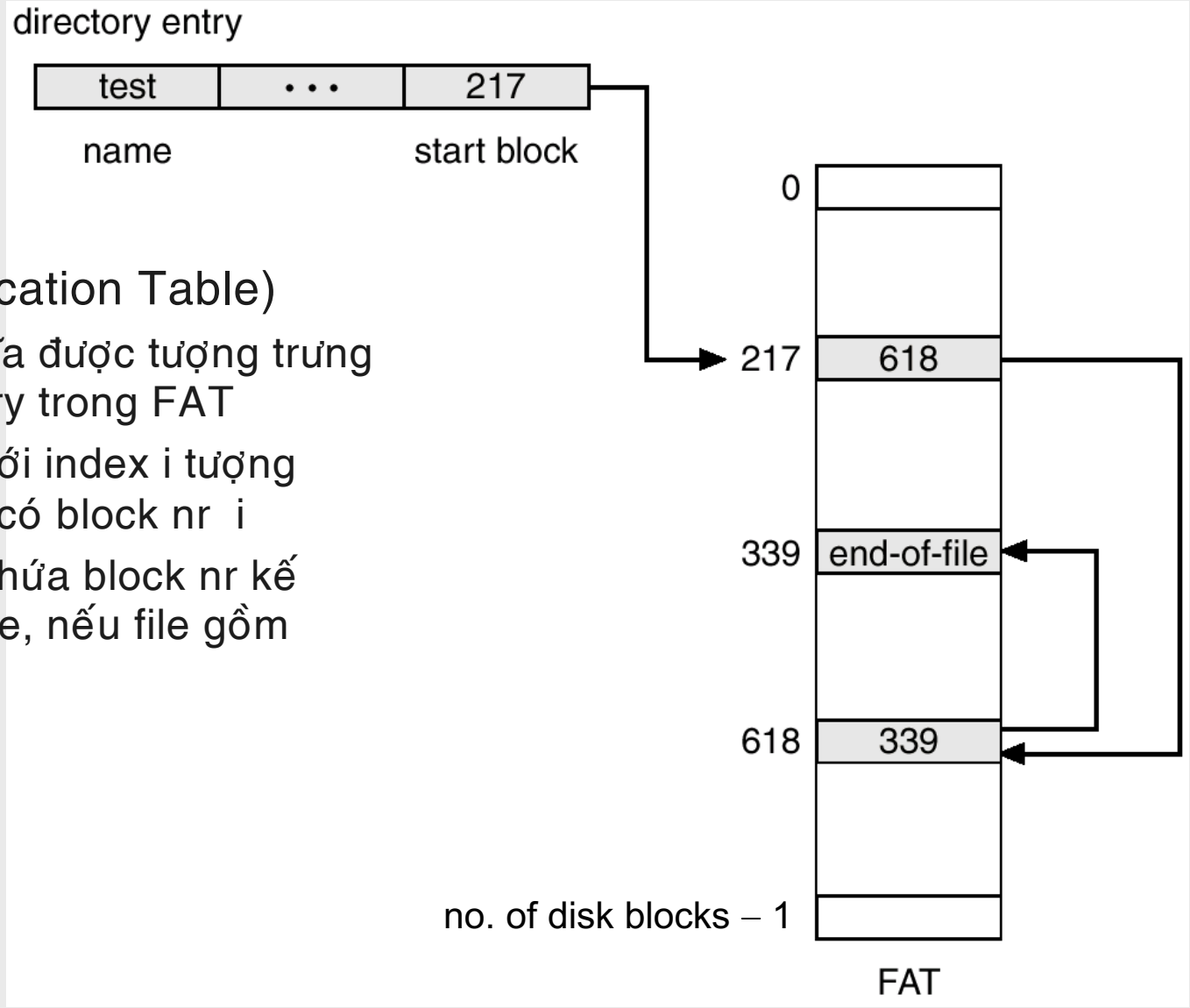
Cấp phát theo danh sách liên kết (2/2)



- 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 truy cập tuần tự
  - Tốn không gian lưu trữ các con trỏ
  - Độ tin cậy: pointer trong block có thể bị hỏng

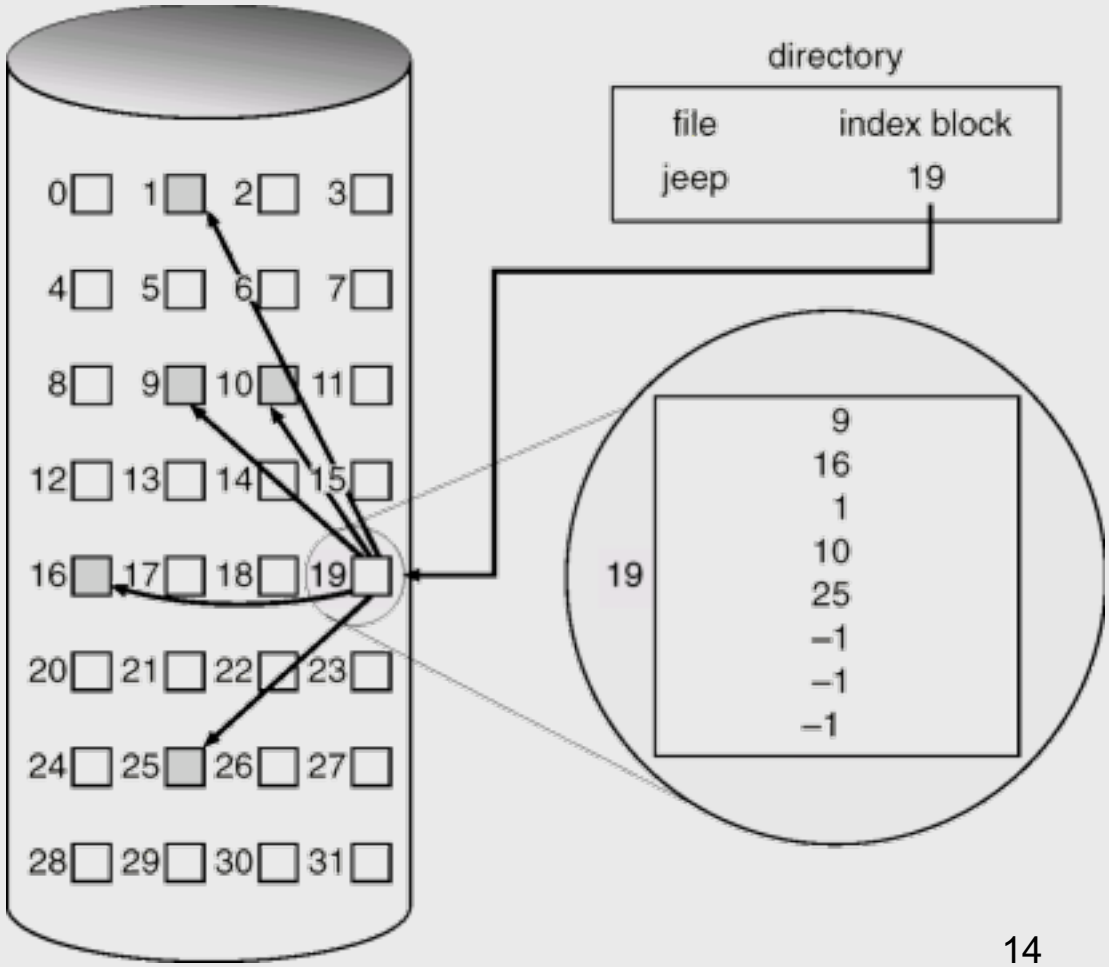
FAT – một hiện thực của cấp phát theo danh sách liên kết:

- 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)

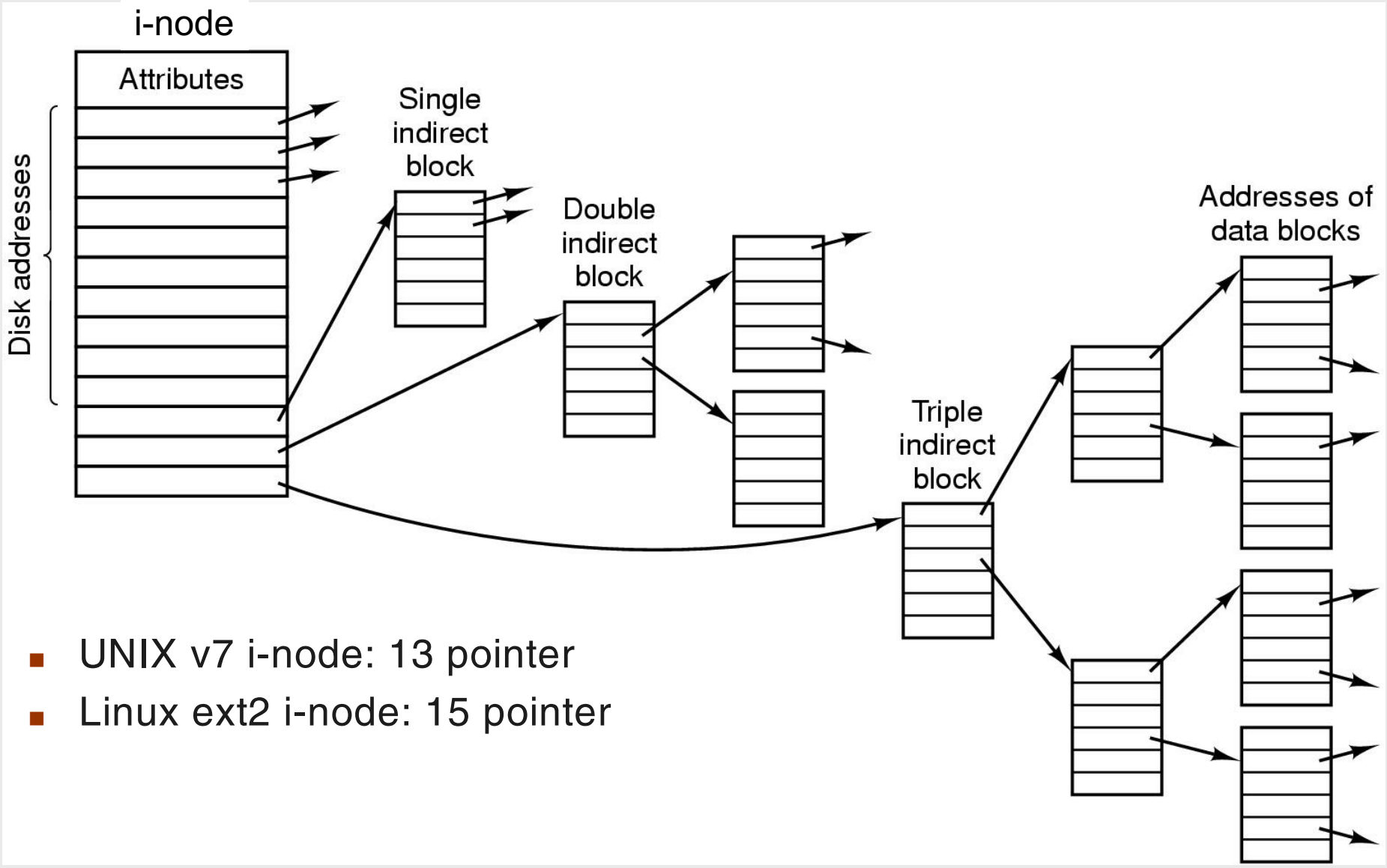
- Bảng/khối index (index block)
  - chứa địa chỉ các block của file
  - thứ tự các địa chỉ cũng là thứ tự các block của file



Cấp phát dùng chỉ mục (2/2)

- Ưu điểm
  - Random và sequential access
  - Không xảy ra phân mảnh ngoại
- Khuyết điểm
  - Tốn không gian lưu trữ bảng index dù file có kích thước chỉ vài block
- Vấn đề: Kích thước một file có thể nhỏ nhưng cũng có thể rất lớn. Kích thước index block bao nhiêu là phù hợp?
  - Giải quyết: multilevel index $\Rightarrow$ i-node

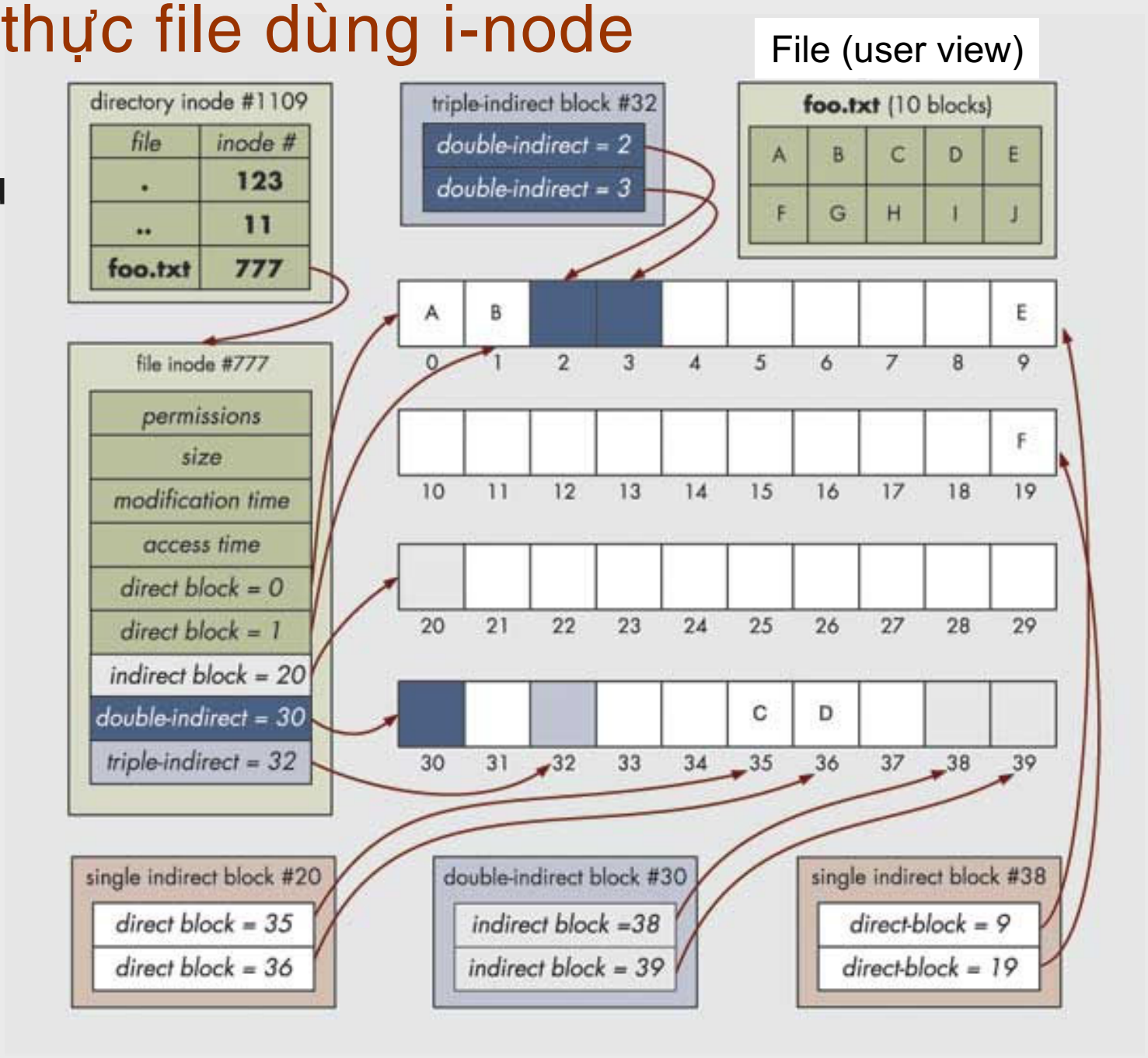
i-node – một hiện thực của index block



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

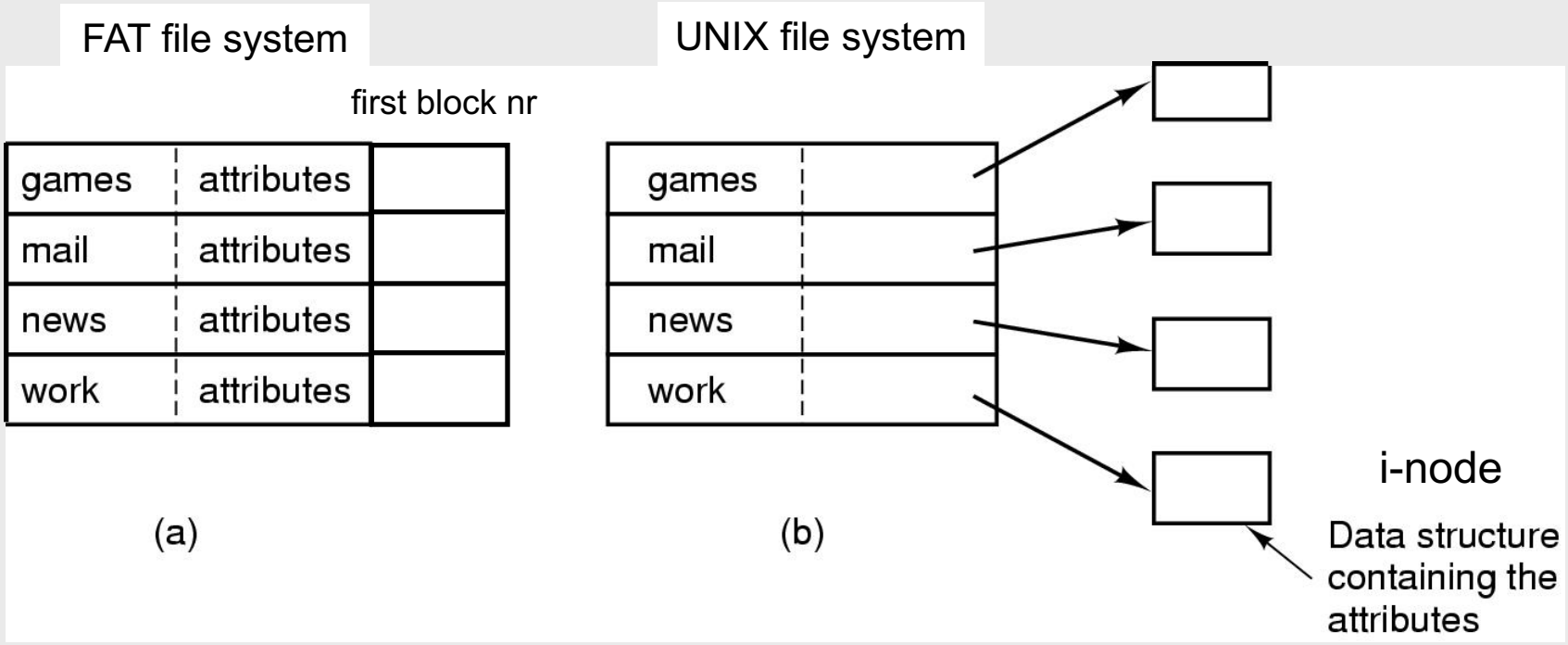
Hiện thực file dùng i-node

- Ví dụ



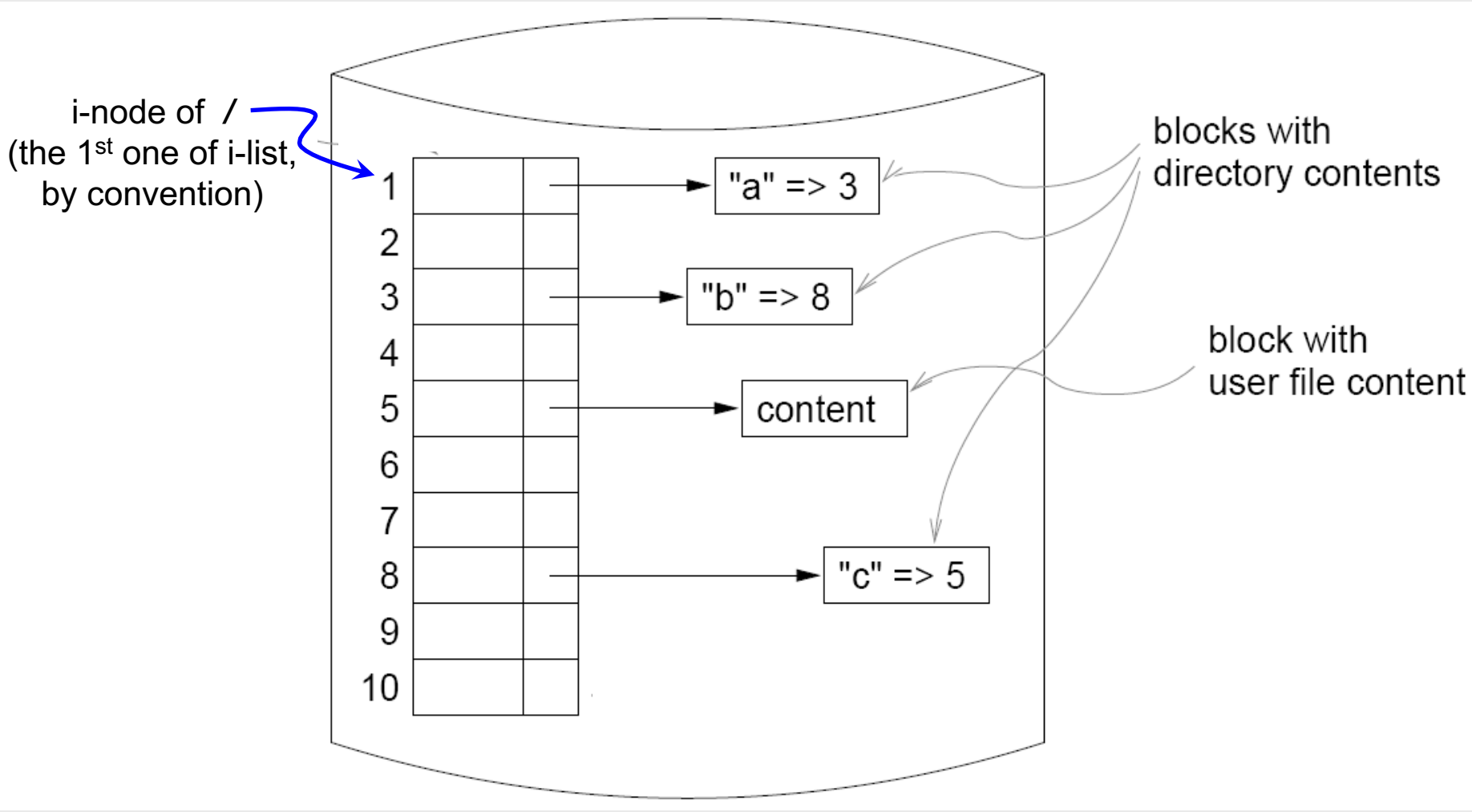
Hiện thực thư mục

- 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,…

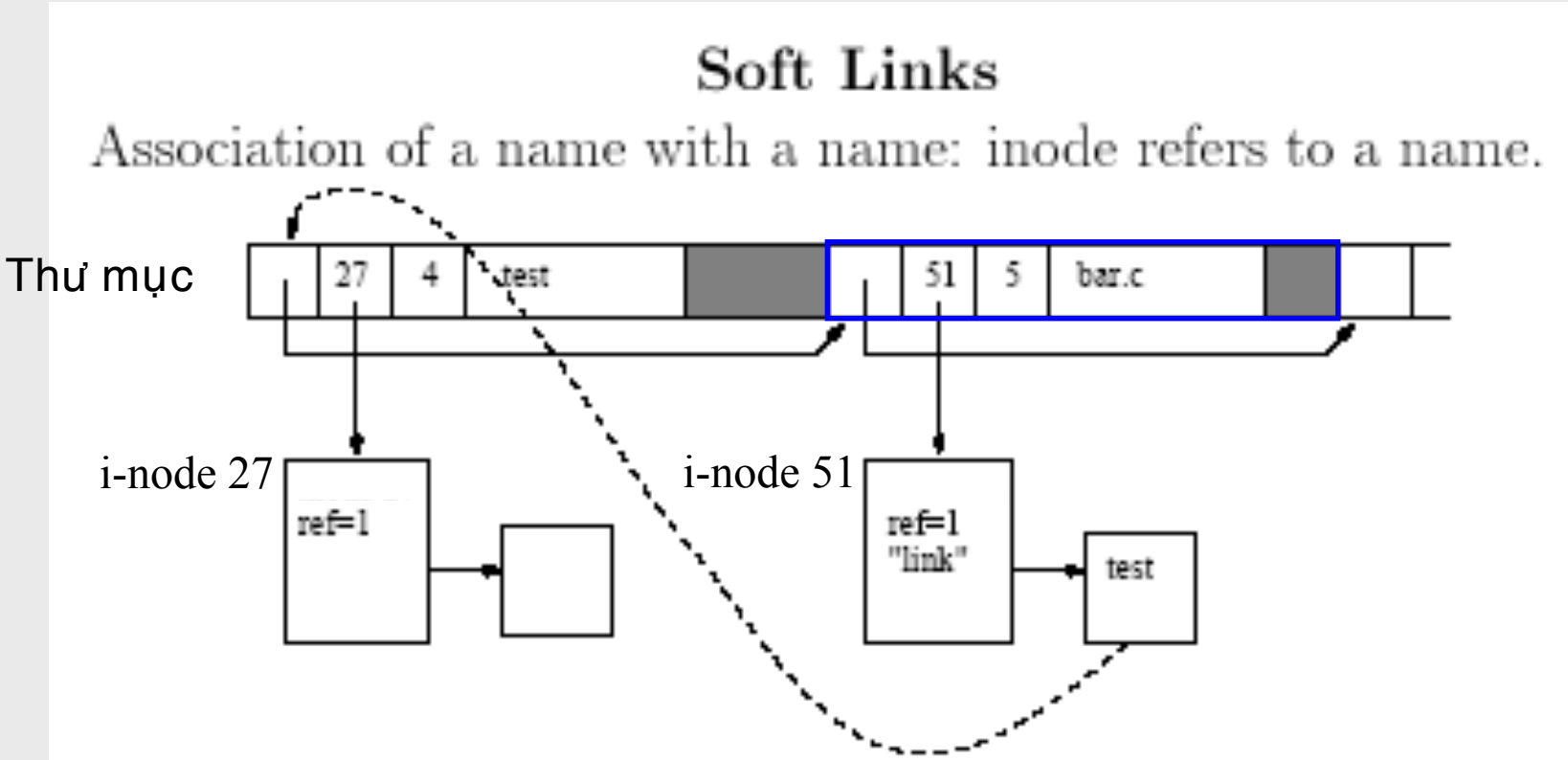


Duyệt path name để lấy block nr của file

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



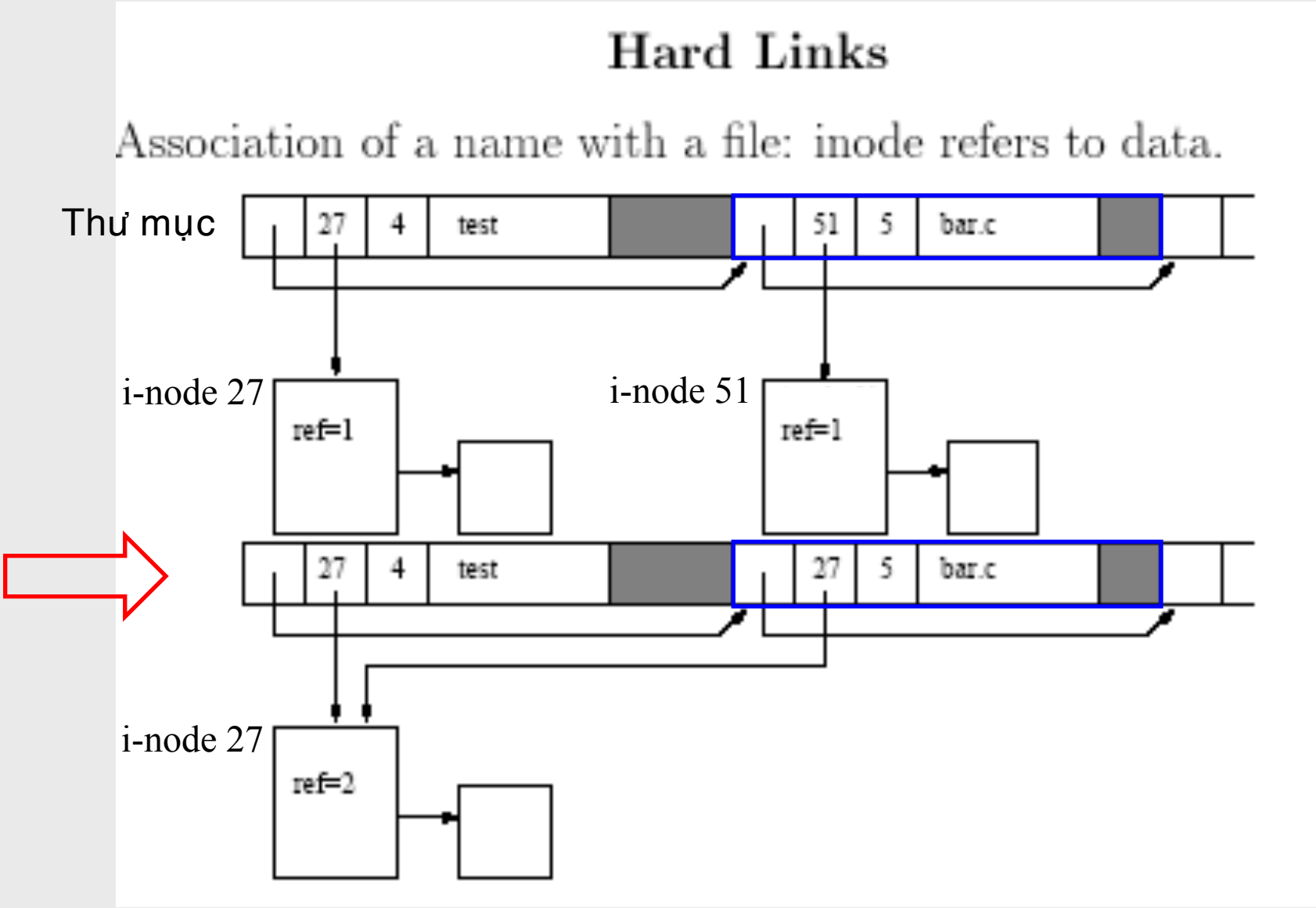
i-node: chia sẻ file (1/2)



which gives

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

i-node: chia sẻ file (2/2)

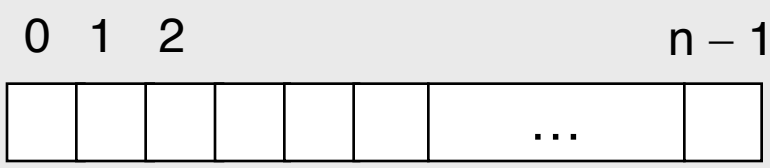


Quản lý không gian trống

Các phương pháp

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

Phương pháp bit vector (bit map)



bit[i] = { 0 ⇒ block i còn trống; 1 ⇒ block i đã được cấp

Ví dụ:

bit vector 00111100…

⇔

block 0, 1 trống

block 2, 3, 4, 5 đã được cấp

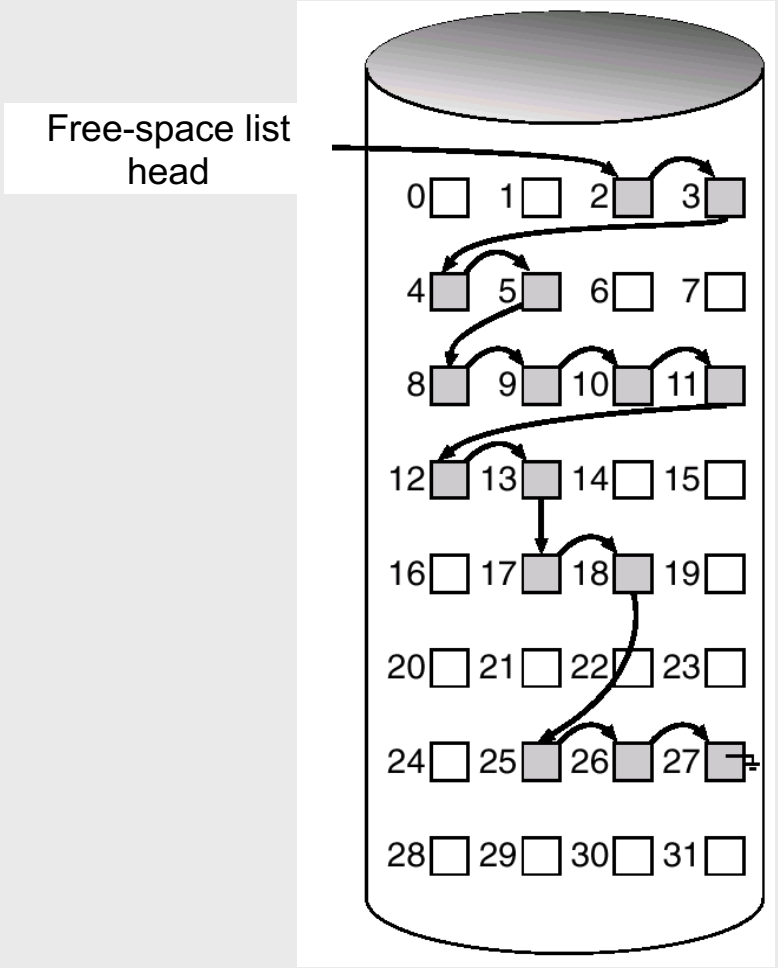
block 6, 7 trống

…

- Ưu điểm: Đơn giản và hiệu quả khi cần tìm khối trống đầu tiên hoặc chuỗi khối trống liên tục
  - Thao tác trên bit
- Khuyết điểm: 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 (32 KB)

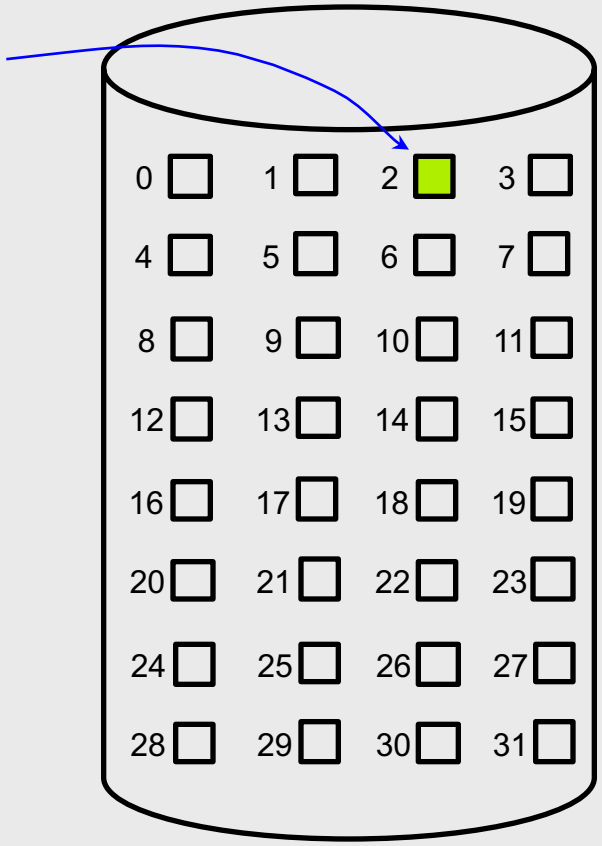
Phương pháp dùng linked list

- Phương pháp
  - Liên kết các khối trống với nhau
  - Chỉ cần giữ con trỏ đến khối nhớ trống đầu tiên trên đĩa, có thể cache trong bộ nhớ chính để tăng tốc
- Ưu điểm: Ít lãng phí không gian đĩa, vd so với bit vector
- Khuyết điểm: Không hiệu quả; trong trường hợp xấu nhất phải duyệt toàn bộ đĩa để tìm không gian trống liên tục (trường hợp cấp phát liên tục)
- [Làm sao tiếp nhận một khối trống được trả về?]



Grouping

- Phương pháp grouping
  - Các địa chỉ của *n* khối trống được lưu trong một khối trống ban đầu.
  - Khối trống thứ *n* chứa các địa chỉ của *n* khối trống kế tiếp.
- Nhận xét: grouping là mở rộng của phương pháp dùng linked list (lấy *n* = 1).

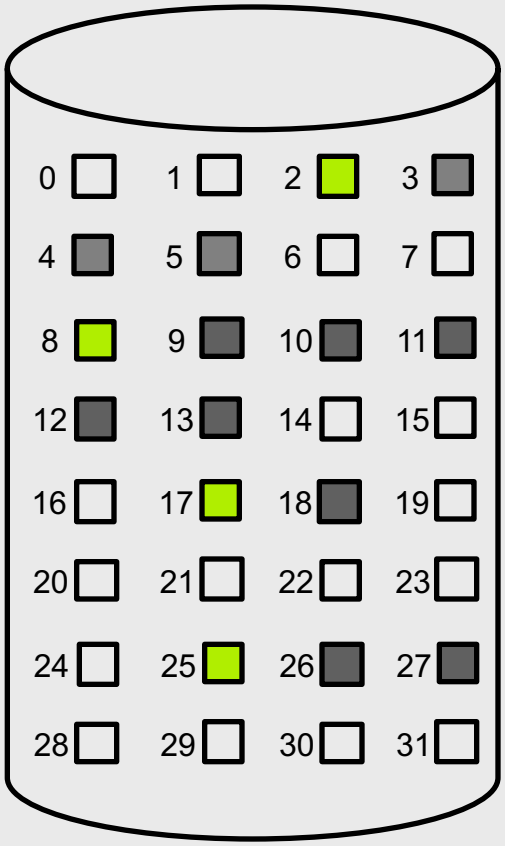


- Grouping với *n* = 3

  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

Counting

- Phương pháp counting
  - Sử dụng các khối chỉ mục (index block)
    - mỗi entry: địa chỉ của khối trống đầu tiên của nhóm khối trống liên tục và một số đếm số lượng khối trống.
  - Có thể cấp phát hoặc tiếp nhận đồng thời nhiều chuỗi khối trống liên tục.



- Nội dung index block (ở đây block 21)

start	count
2	4
8	6
17	2
25	3

Journaling file system

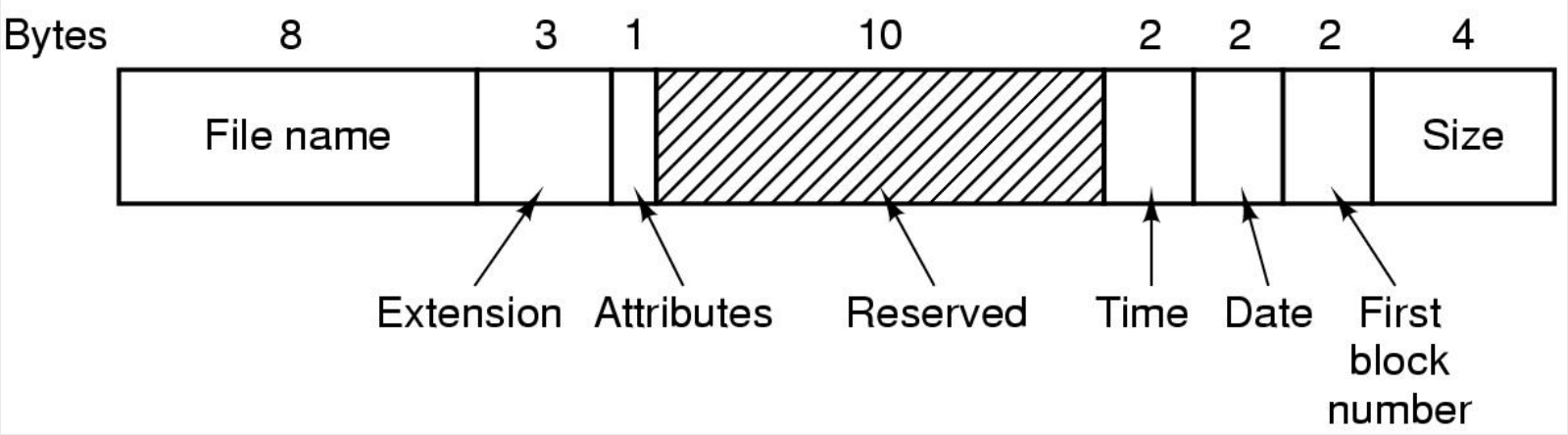
- Journaling file system
  - Ghi nhận các lần cập nhật trên file system thành các giao tác (transaction)
  - Mọi transaction đều phải được ghi nhận trong log file
  - Một transaction được xem là hoàn tất (commit) $\leftrightarrow$ đã được ghi nhận đầy đủ trong log file (lúc này, file system có thể chưa 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

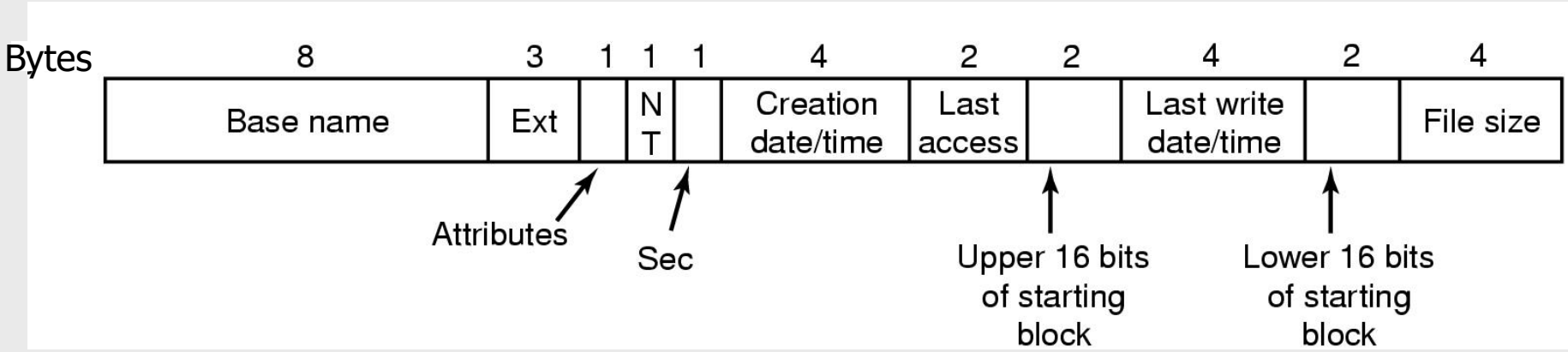
MS-DOS File System

- MS-DOS directory entry



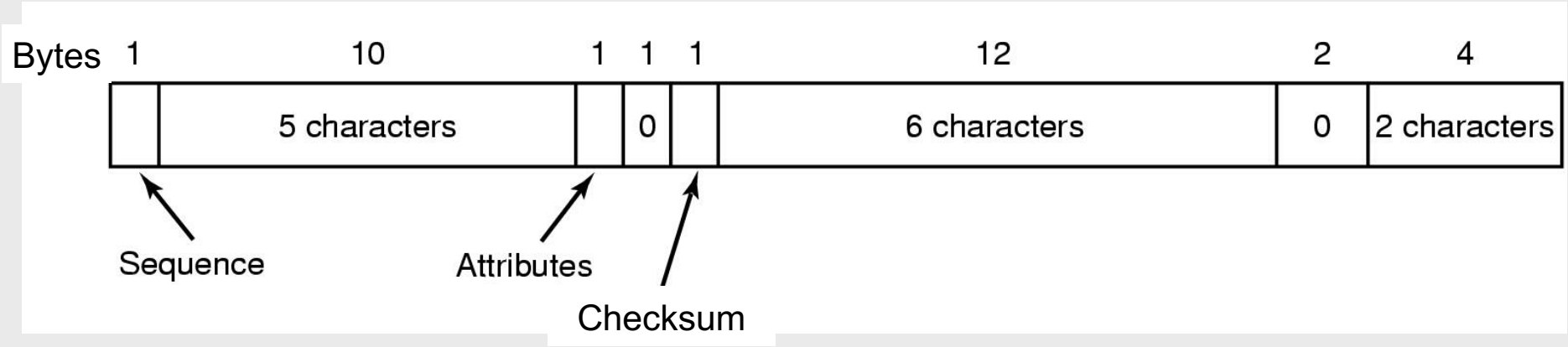
Windows 98 File System (1/3)

- Mở rộng MS-DOS directory entry để dùng trong Windows 98



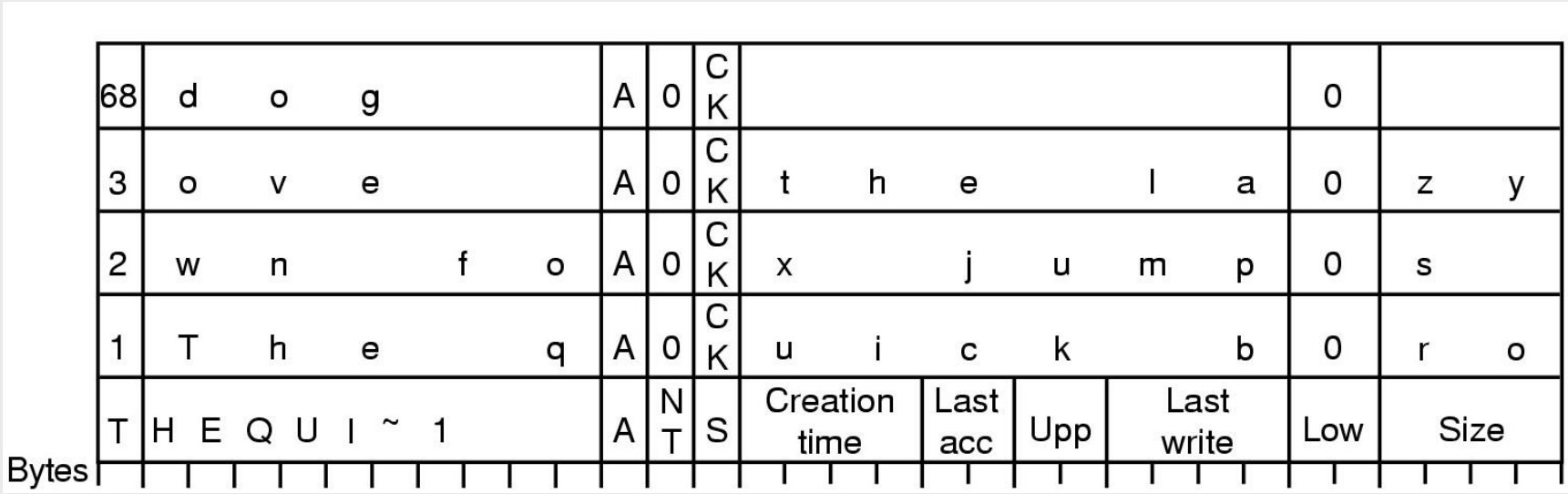
Windows 98 File System (2/3)

- Một entry cho (một phần của) một tên file dài trong Windows 98



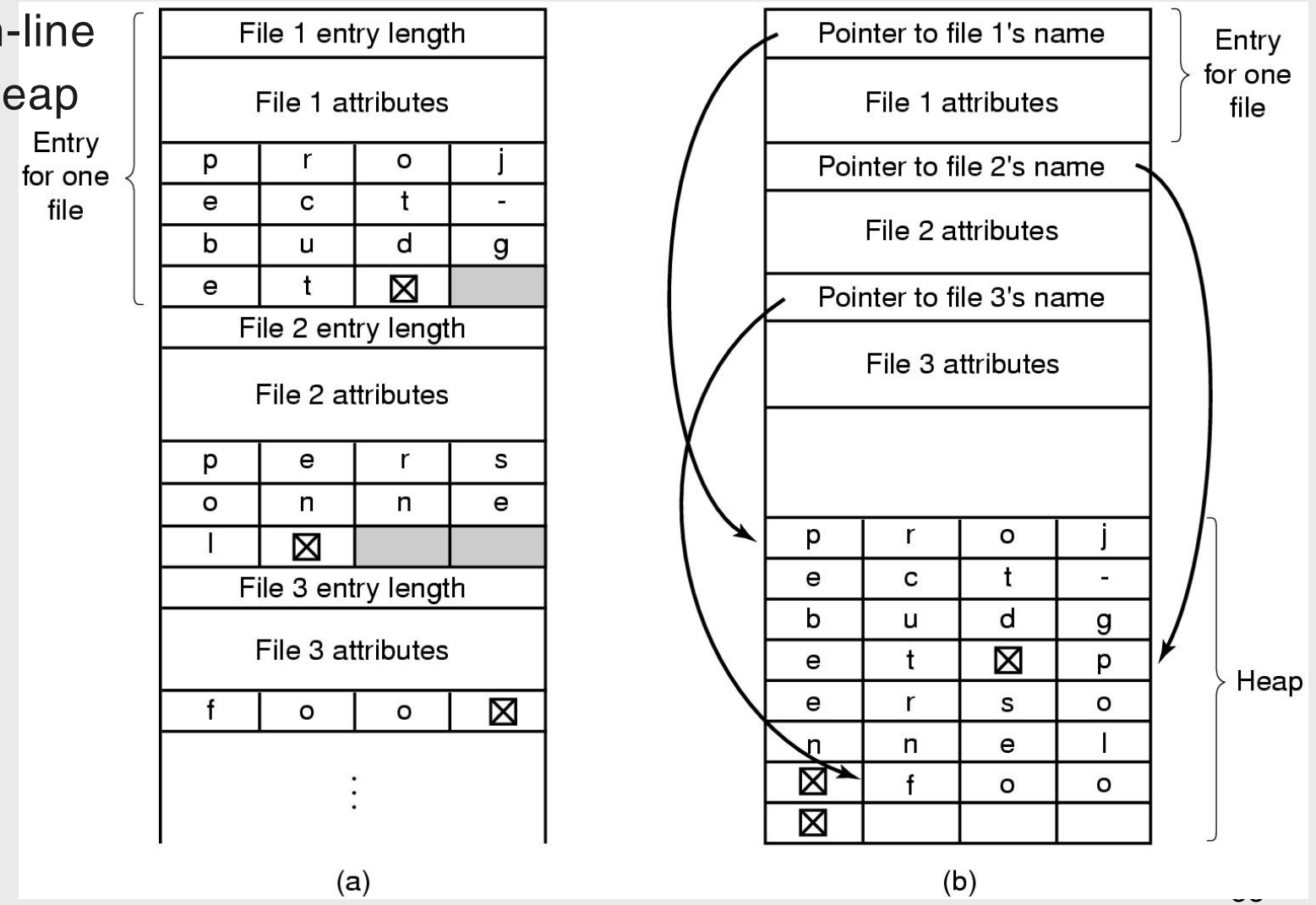
Windows 98 File System (3/3)

- Minh họa cách một tên dài được lưu trong Windows 98



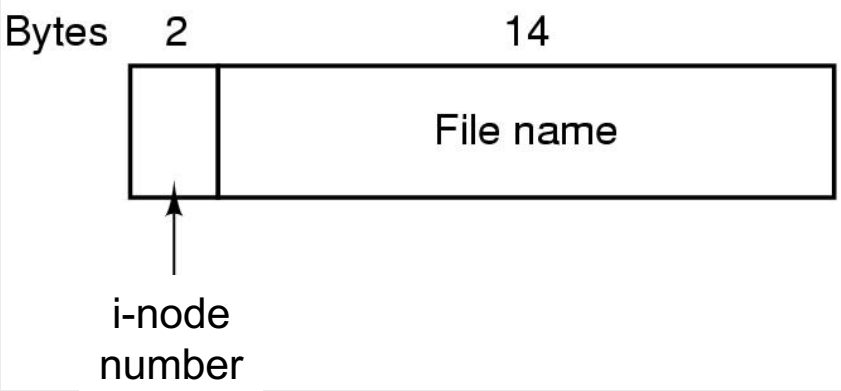
Hiện thực tên file dài

- Giải quyết vấn đề tên file dài (Win98, 2000, XP, *NIX,…)
  - (a) In-line
  - (b) Heap



UNIX V7 File System (1/2)

- Một UNIX V7 directory entry



UNIX V7 File System (2/2)

- Các bước để dò tìm `/usr/ast/mbox`