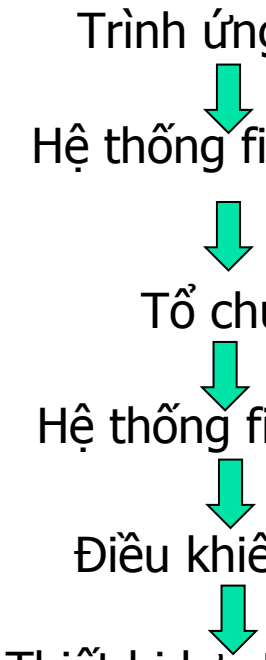
Hệ thống tập tin (tt.)

Nội dung (phần 2)

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

Cấu trúc hệ thống file

Theo hệ thống phân lớp chức năng: *Mỗi lớp tạo các chức năng hỗ trợ cho lớp trên trực tiếp & sử dụng các chức năng hỗ trợ lớp dưới trực tiếp*



Tạo file, xóa file, ...

Cấu trúc file: thư mục, FCB, bảo vệ, Bảo mật file, etc.

Chuyển đổi logic <-> Physic, q/lý không gian trống, etc.

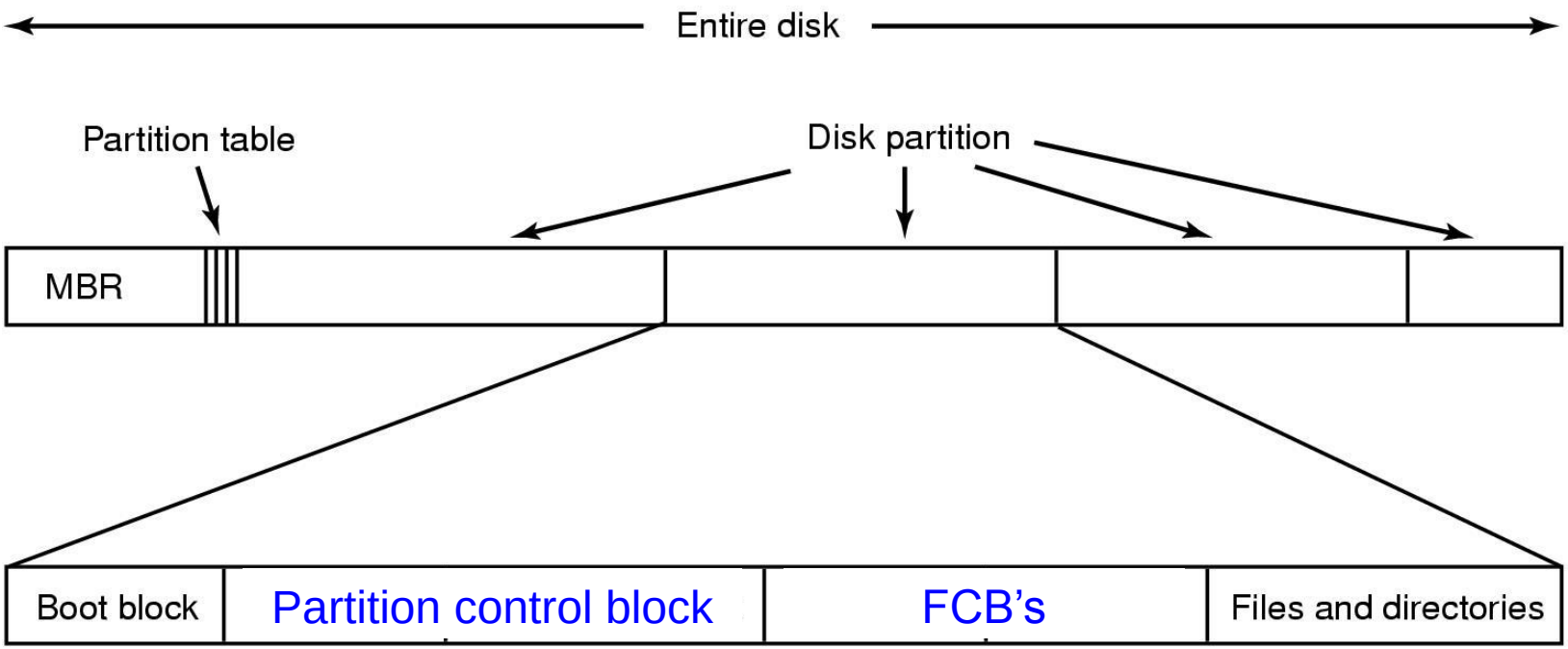
Lệnh căn bản đến Device Driver: MemBuf, Đọc, ghi khối dữ liệu lên disk (d?c?t?s?)

Device Driver, Xử lý ngắt quãng, vận chuyển dữ liệu (Mem <-> Disk)

Môi trường lưu trữ vật lý

Bố trí (layout) hệ thống file

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

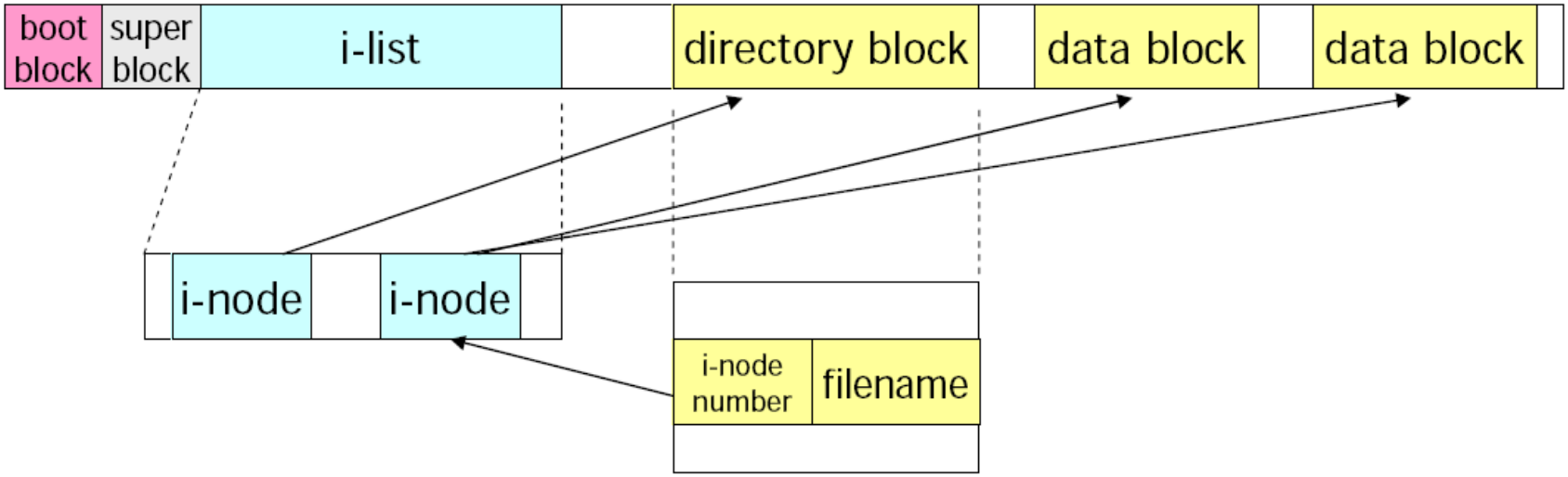


Bố trí hệ thống file (tt.)

- Partition control block
  - lưu số lượng blocks 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, UNIX File System: “superblock”
- 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, UNIX File System: “i-node”

Sơ đồ bố trí hệ thống file (tt.)

- Layout của một partition chứa hệ thống file UNIX

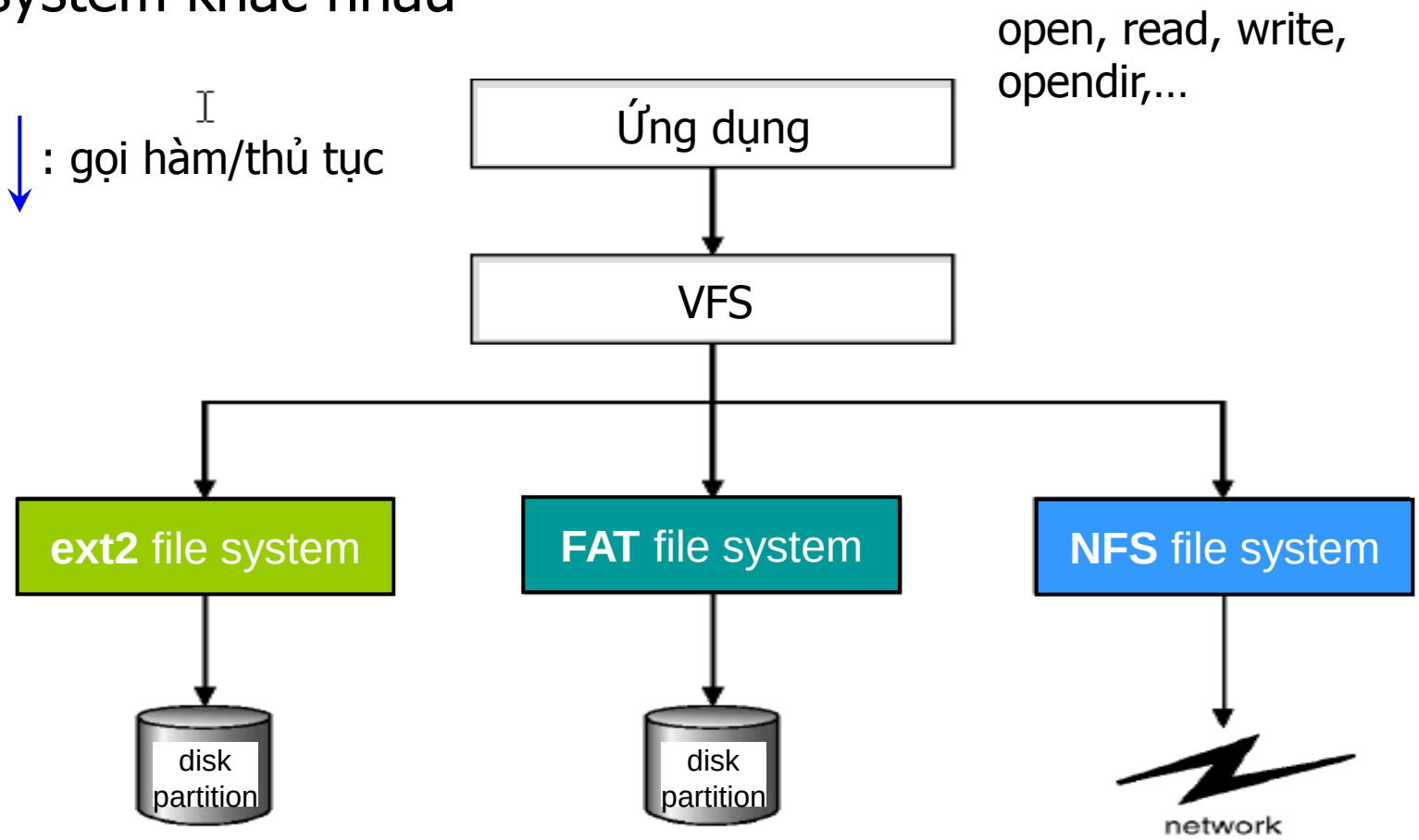


Sơ đồ bố trí hệ thống file (tt.)

- 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)

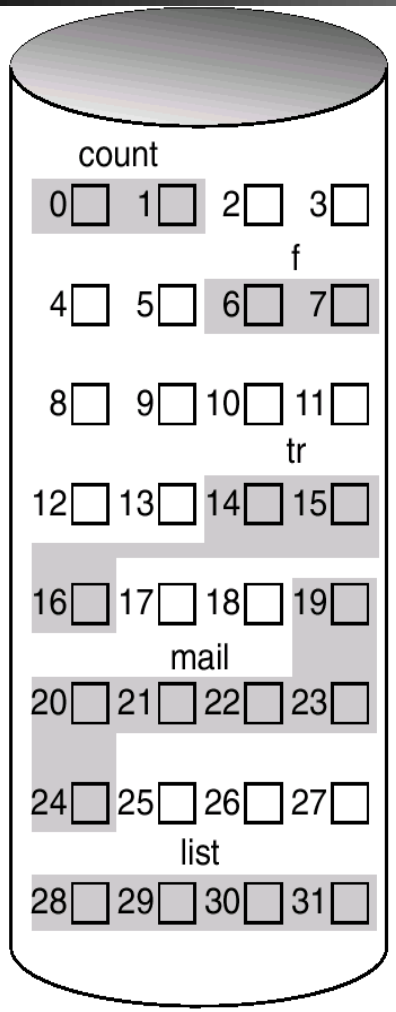
- VFS cung cấp một giao diện đồng nhất đến các loại file system khác nhau



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
  - 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

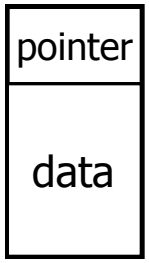


directory

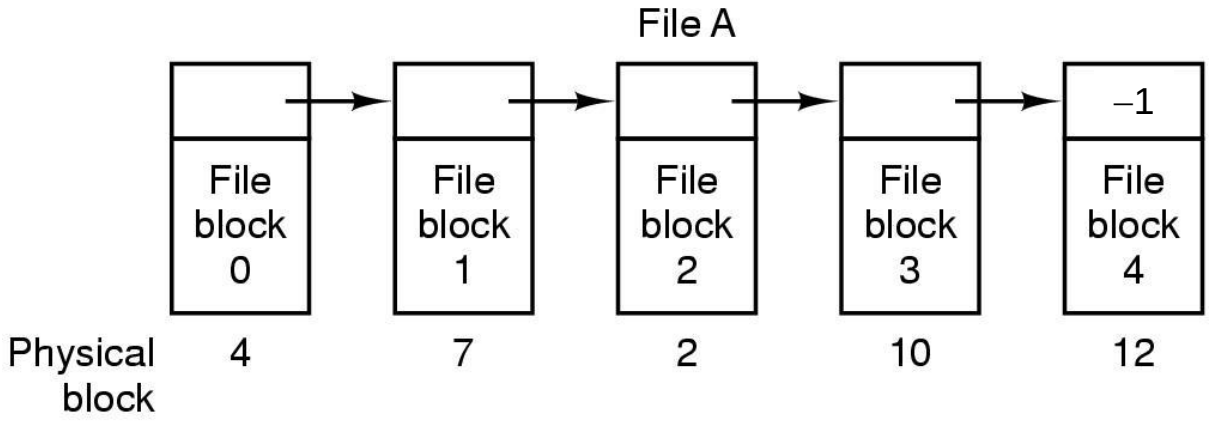
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

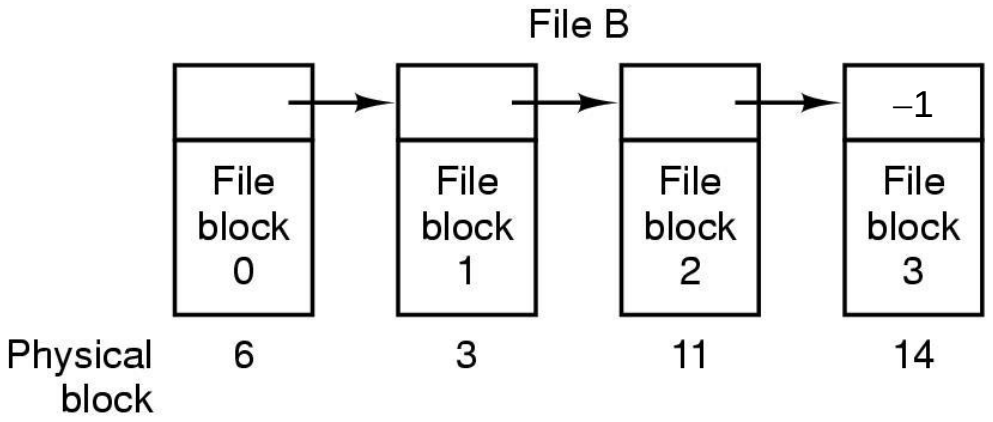
- Seek time? Di chuyển đầu đọc?
- Có thể truy xuất ngẫu nhiên một block của file: block nr = start + block offset
- Phân mảnh ngoại
- Vấn đề khi tạo file mới và khi cần thêm block cho file
- Ứng dụng: ISO-9660 (CDROM)

Cấp phát theo danh sách liên kết

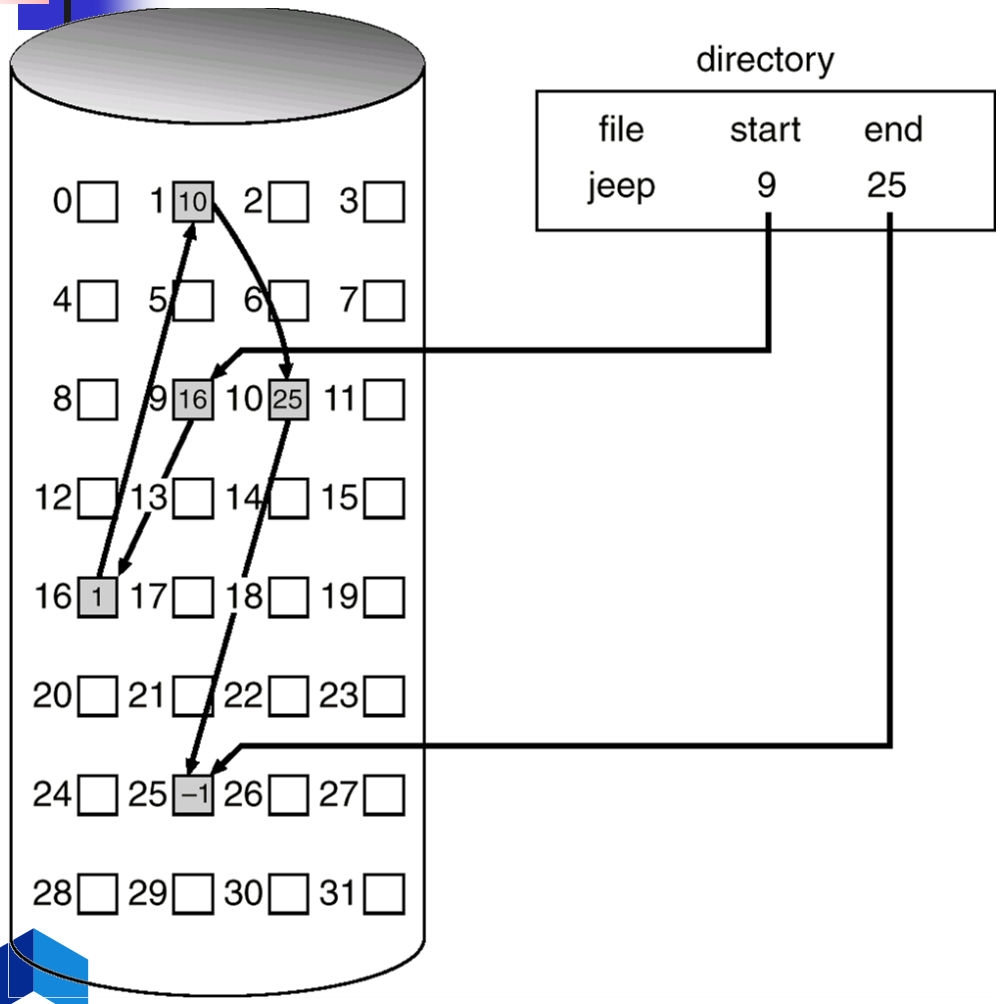


layout của block





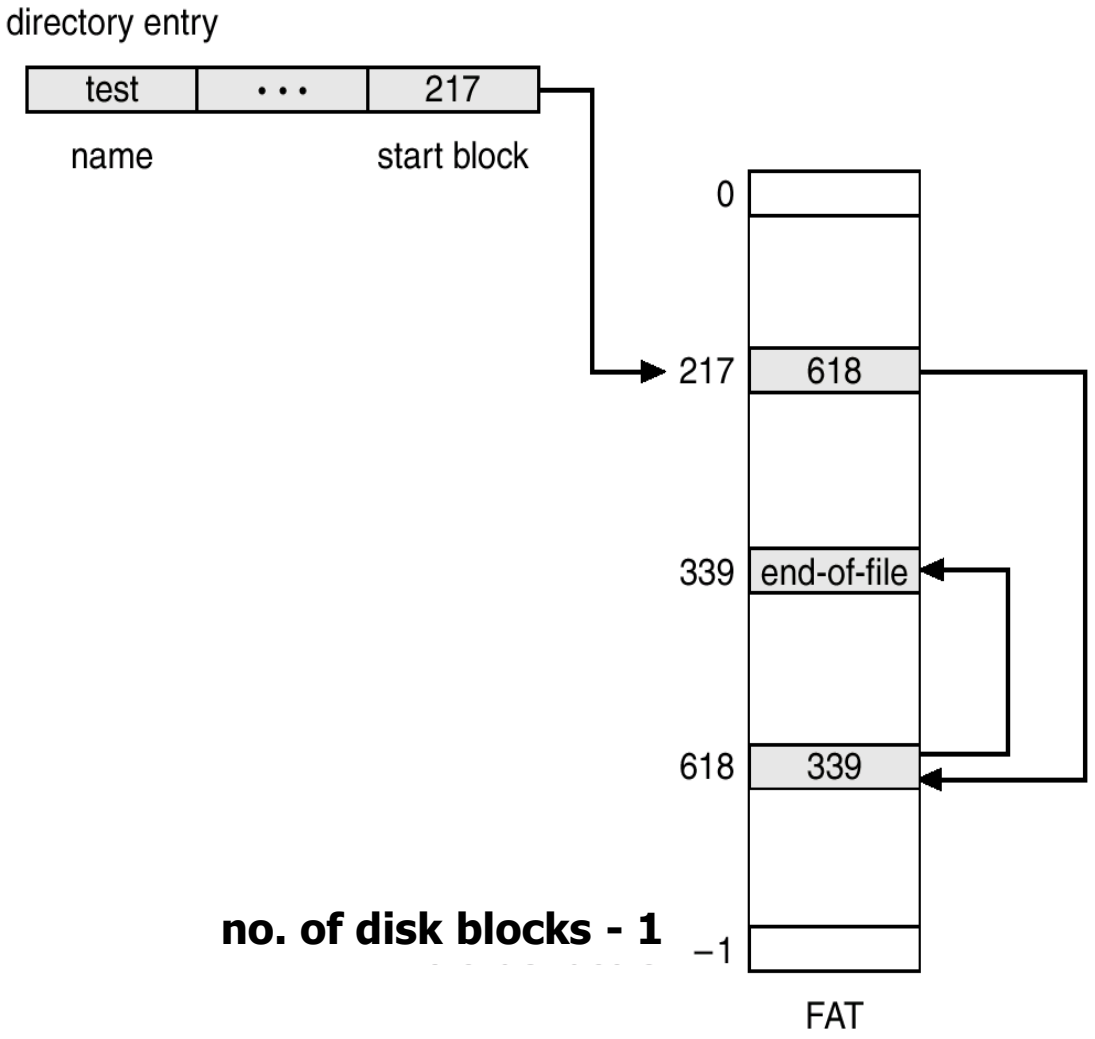
...theo danh sách liên kết (tt.)



- Ưu điểm
  - Dễ dàng thêm block cho file khi cần
  - Quản lý không gian trống bằng danh sách
  - 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

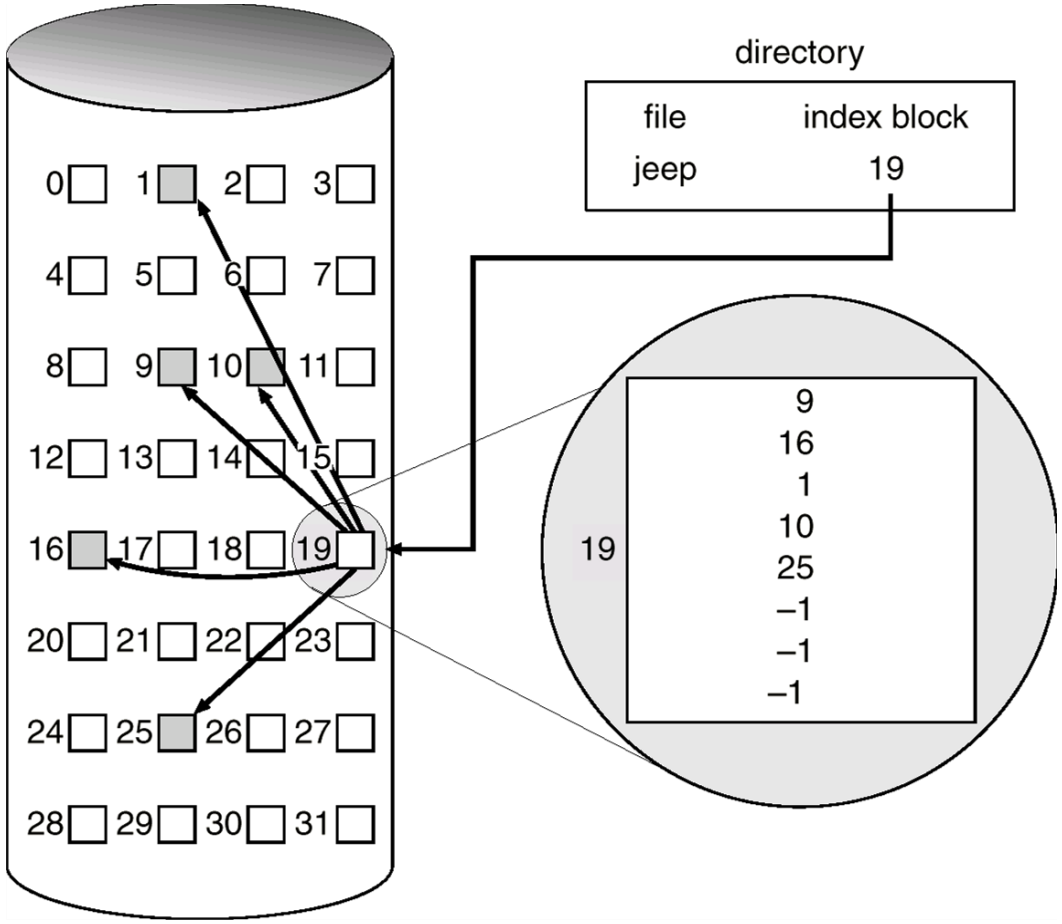
FAT – *một hiện thực cấp phát theo danh sách liên kết:*

- 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
  - Block với block nr i được tượng trưng bởi entry với chỉ số (index) i
    - Entry chứa block nr kế tiếp trong file, nếu file gồm nhiều block



Cấp phát dùng chỉ mục

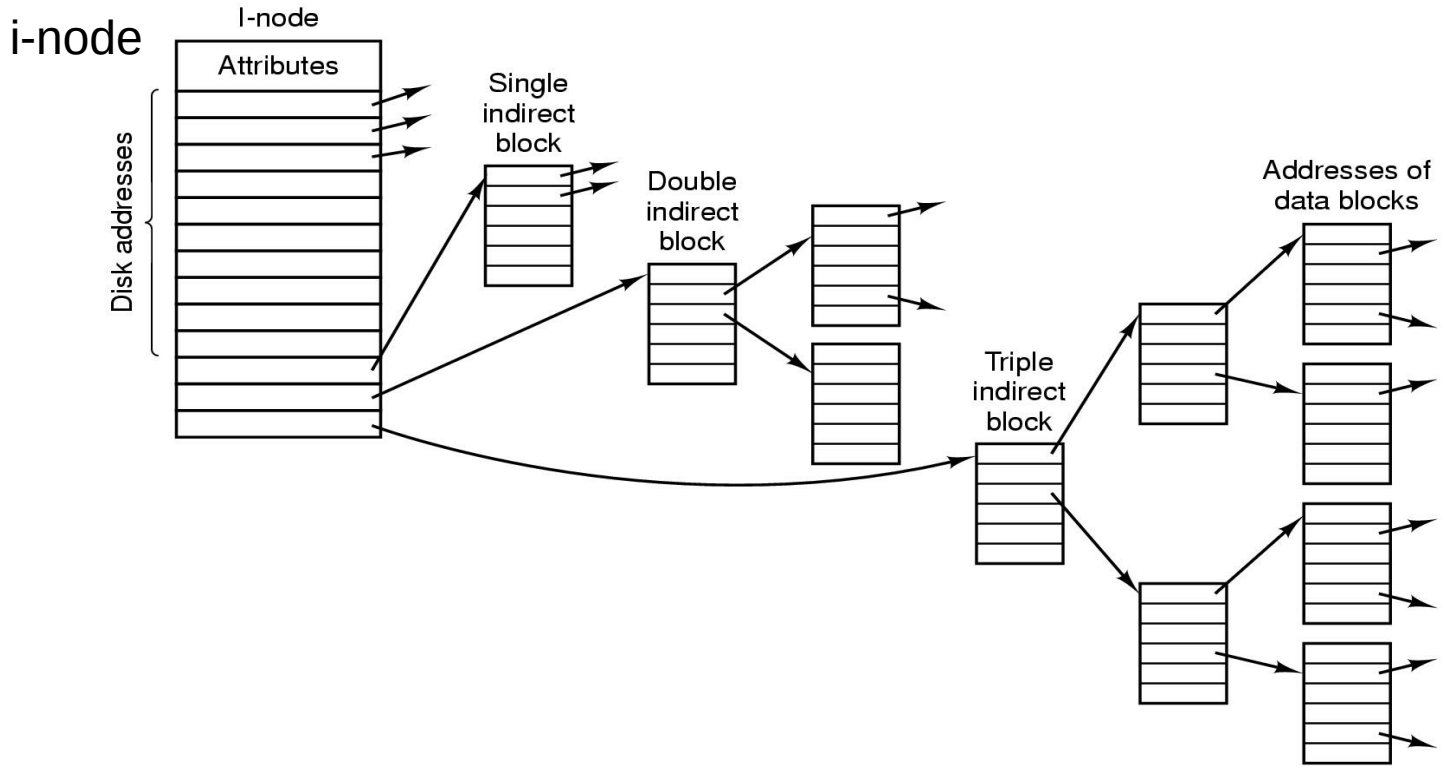
- Bảng index (index block)
  - chứa địa chỉ các block của file
  - thứ tự các địa chỉ trên trong bảng cũng là thứ tự các block trong file



Cấp phát dùng chỉ mục (tt.)

- 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

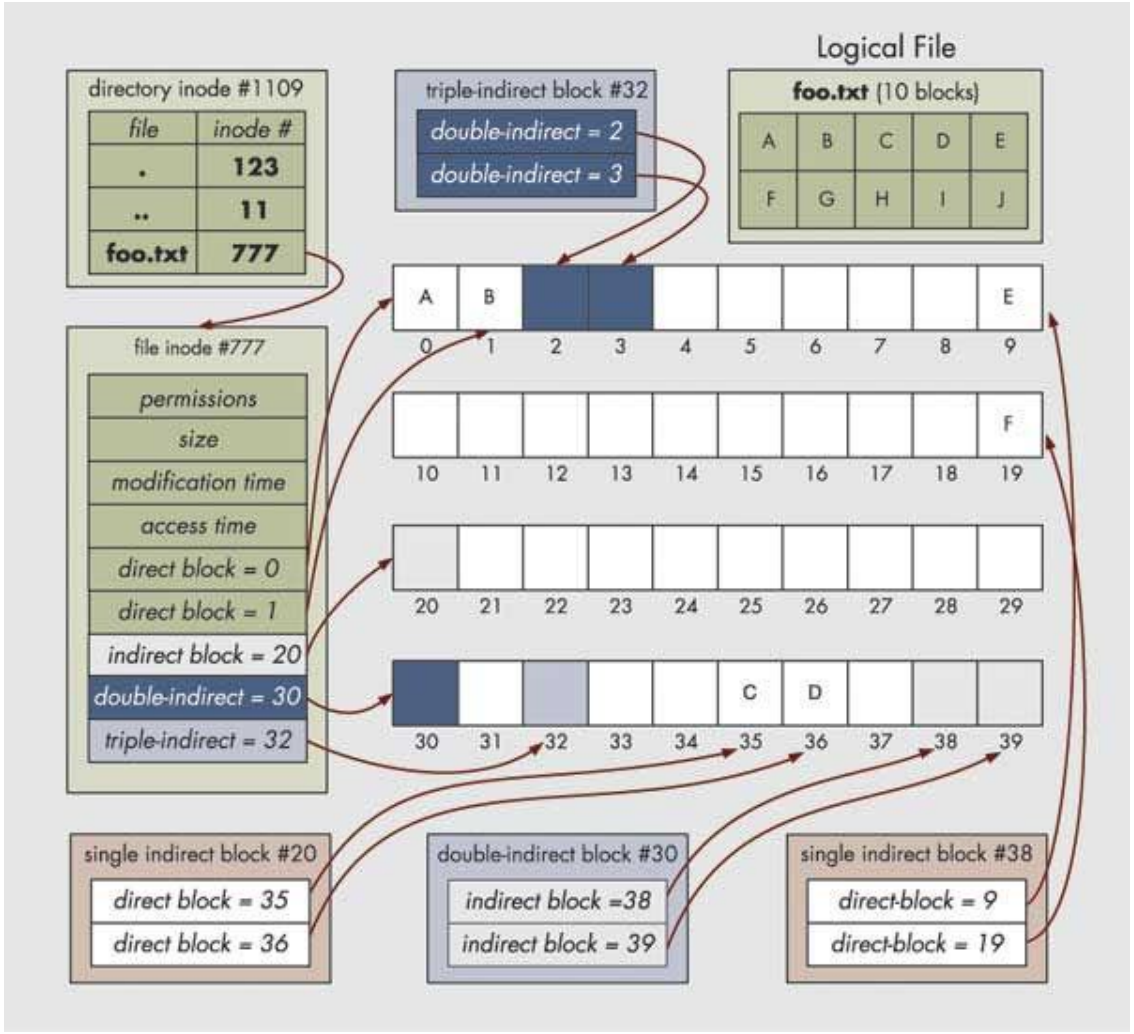
i-node
một hiện thực của index block



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

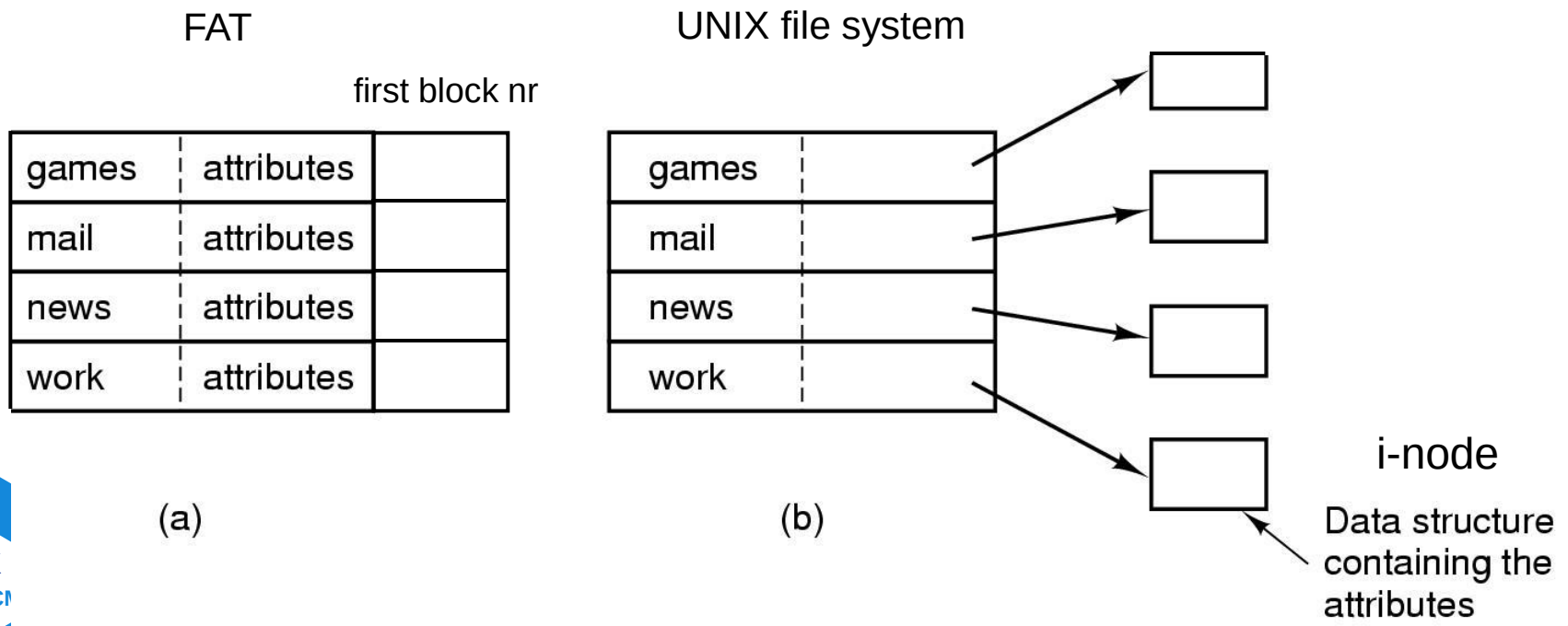
Hiện thực file dùng i-node

- Ví dụ



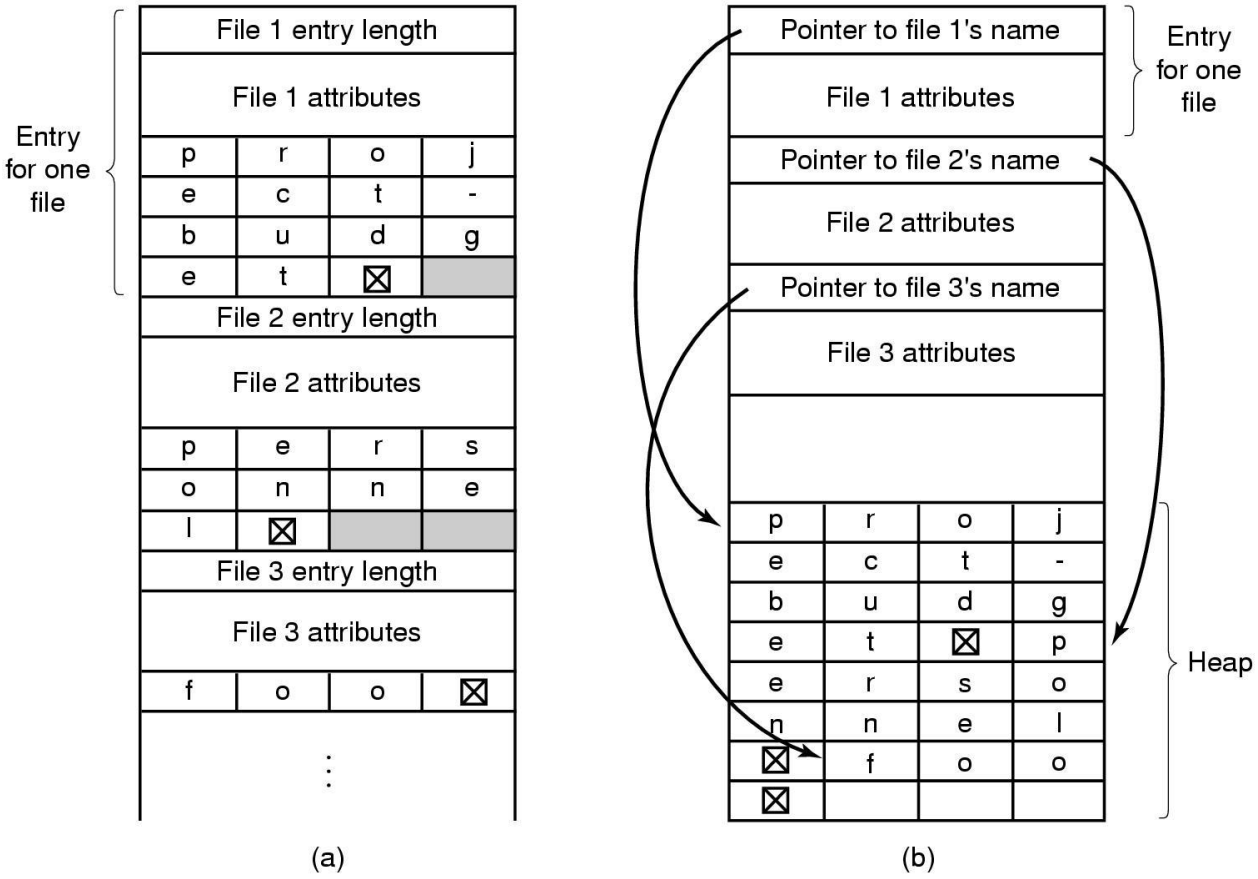
Hiện thực thư mục

- 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,…



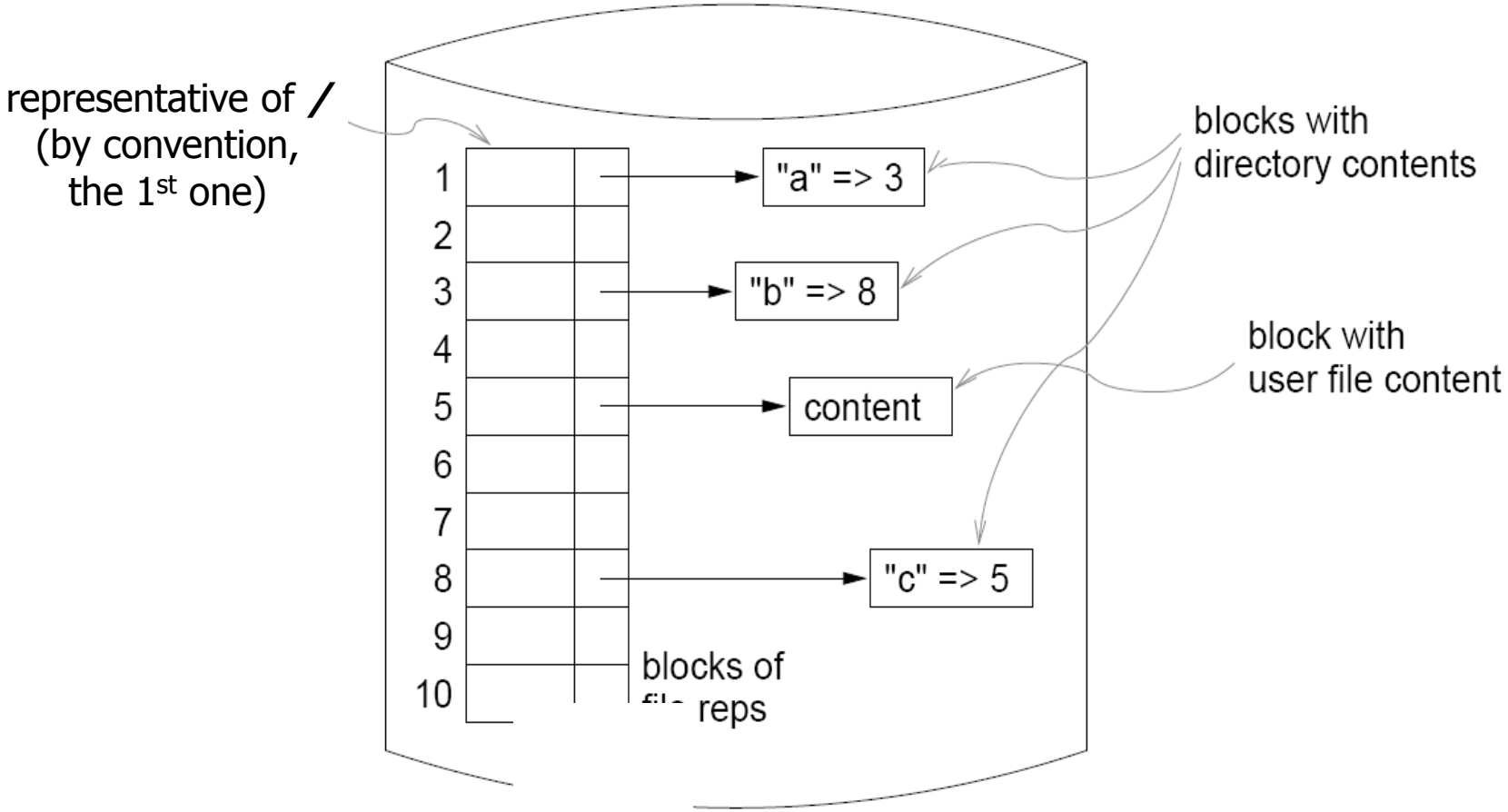
Hiện thực tên file dài

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



Duyệt path name lấy block nr của file

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

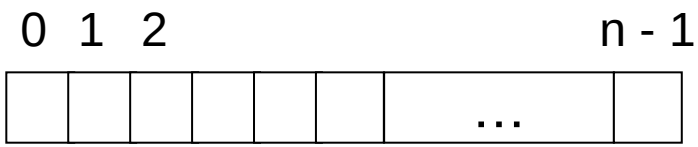


Quản lý không gian trống

Các phương pháp

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

Phương pháp bit vector (bit map)



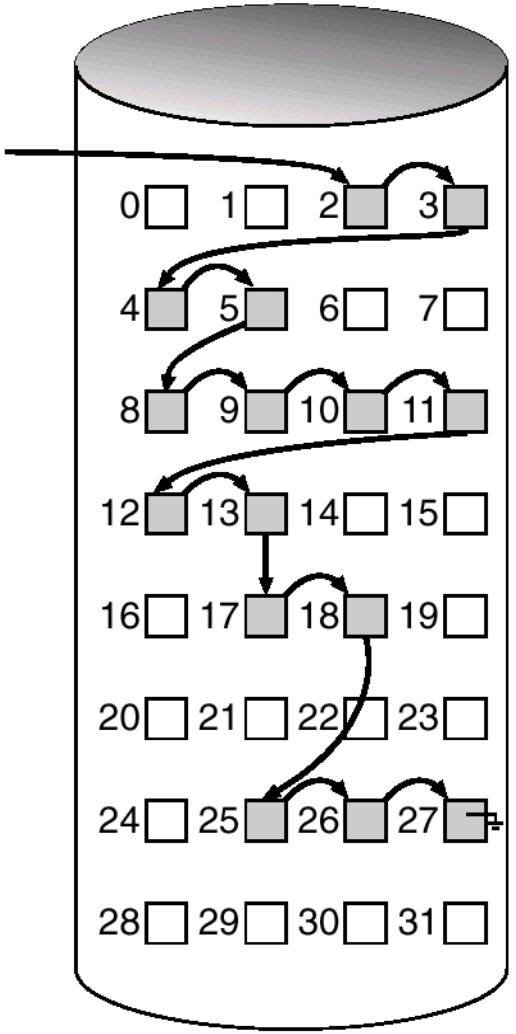
$$bit[\,i\,] = \begin{cases} 0 \Rightarrow \text{block i còn trống} \\ 1 \Rightarrow \text{block i đã được cấp} \end{cases}$$

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} bytes
  - Kích thước đĩa = 2^{30} bytes
  - 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
- Khuyết: Không hiệu quả; trong trường hợp xấu nhất phải duyệt toàn bộ đĩa để tìm không gian trống liên tục

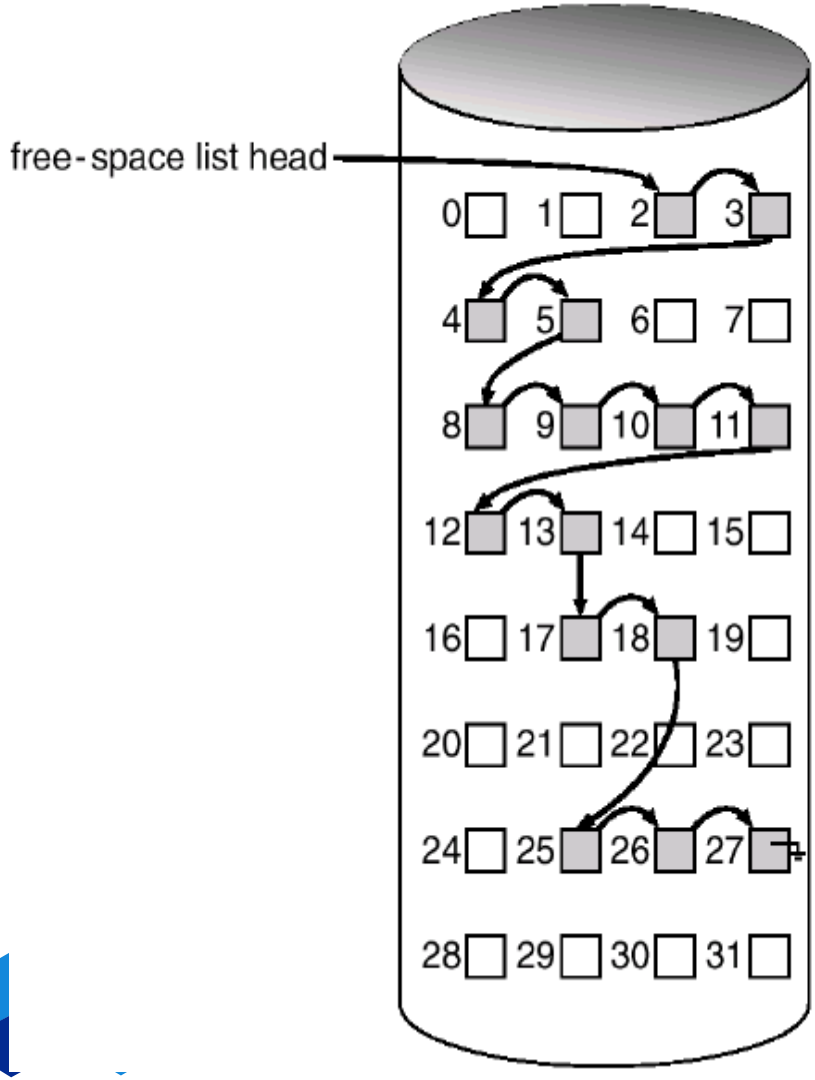


Grouping và counting

- 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 (tt.)

- Ví dụ: Phương pháp linked list

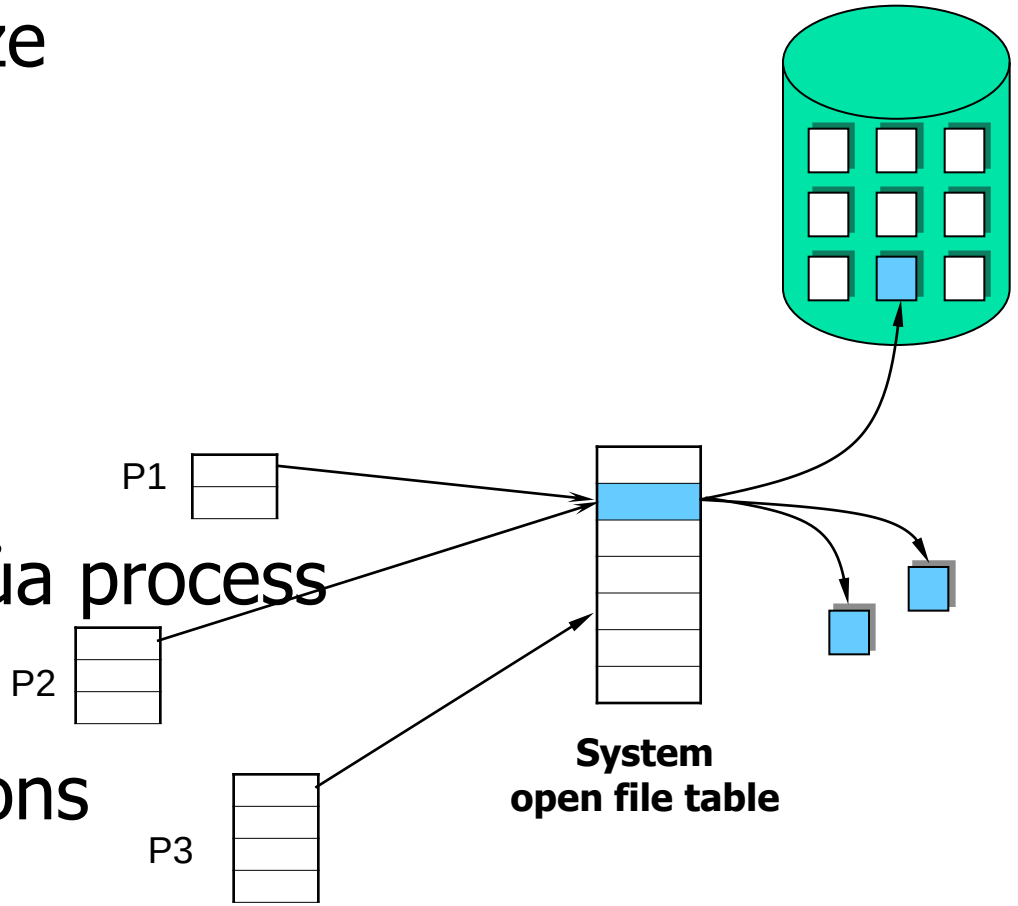


- Phương pháp grouping: n = 3
  - Block 2 lưu 3, 4, 5
  - Block 5 lưu 8, 9, 10
  - Block 10 lưu 11, 12, 13
  - Block 13 lưu 17, 28, 25
  - Block 25 lưu 26, 27

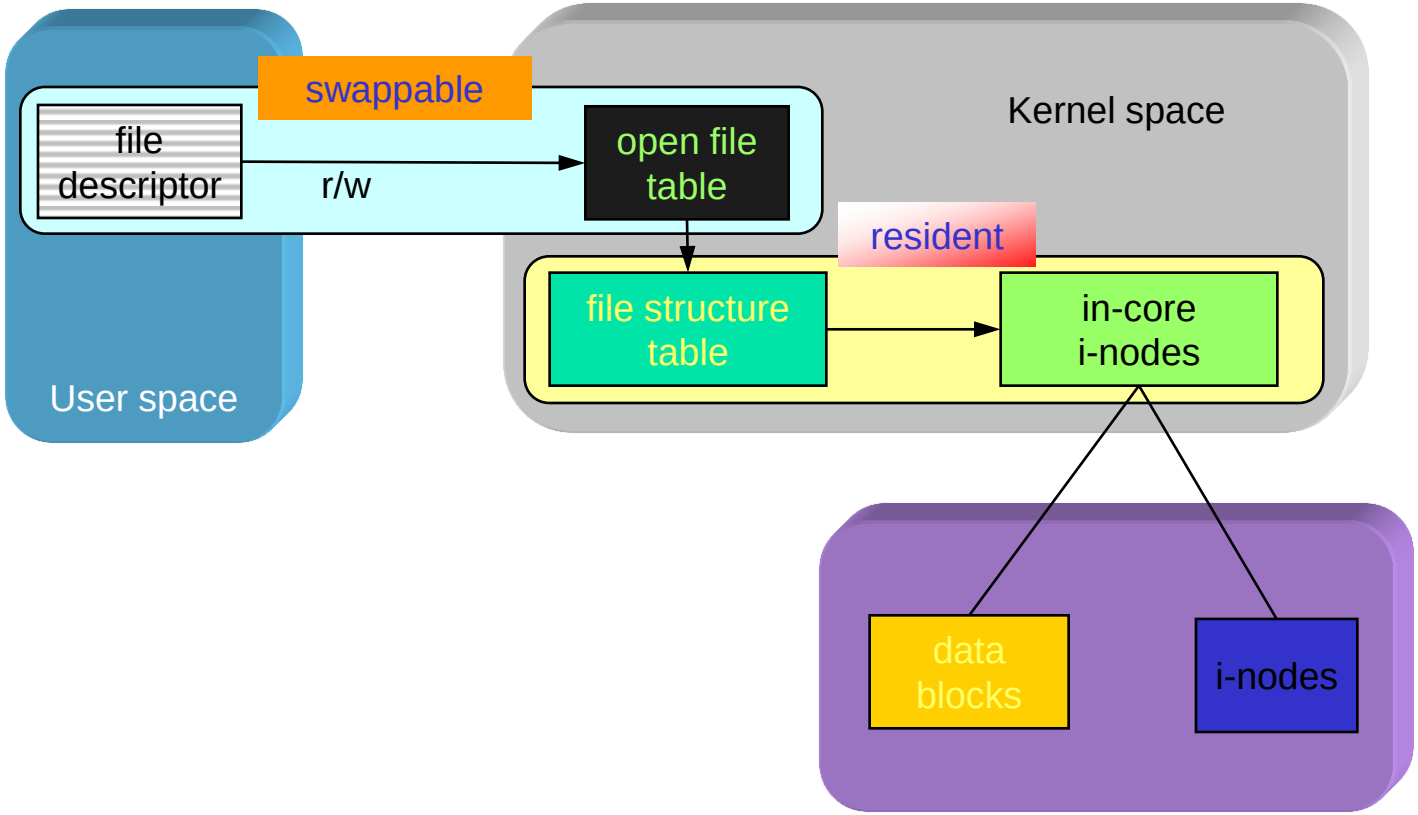
- Phương pháp counting: nội dung index block
  - 2 4
  - 8 6
  - 17 2
  - 25 3

Open file structures

- Thuộc tính toàn cục (global attributes)
  - Disk location, size
  - Times
  - Buffers
  - Open count
  - Lock(s)
- Thuộc tính riêng của process
  - File pointer
  - Access permissions



UNIX: open file structure

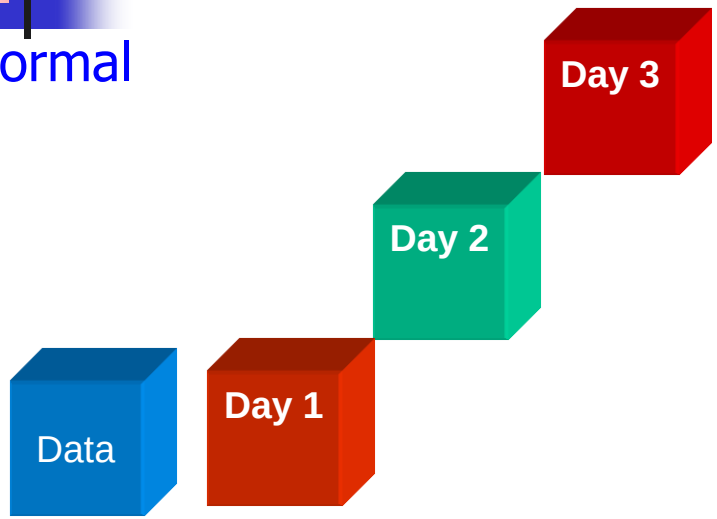


Sao lưu và phục hồi dữ liệu

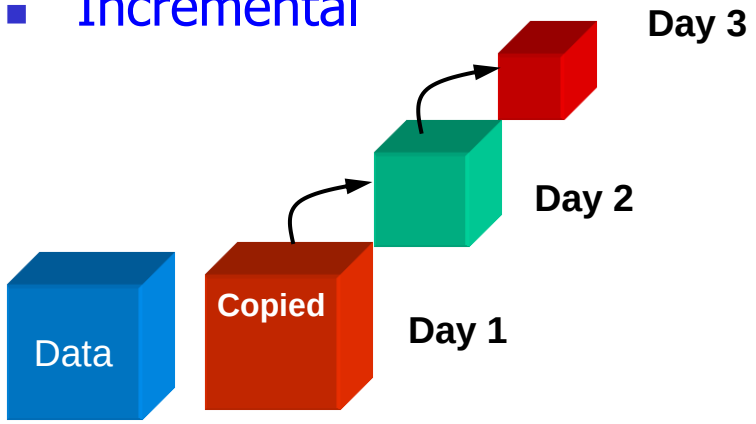
- Kiểm tra sự nhất quán dữ liệu (consistency checker) – so sánh dữ liệu trong cấu trúc thư mục với các khối dữ liệu trên đĩa và sửa chữa các lỗi không nhất quán dữ liệu giữa hai bên.
- Dùng chương trình hệ thống để sao lưu (backup) dữ liệu từ đĩa sang các thiết bị lưu trữ phụ khác như đĩa mềm, đĩa quang, băng từ,... và phục hồi dữ liệu bị mất từ bản sao lưu.

Các kiểu sao lưu dữ liệu

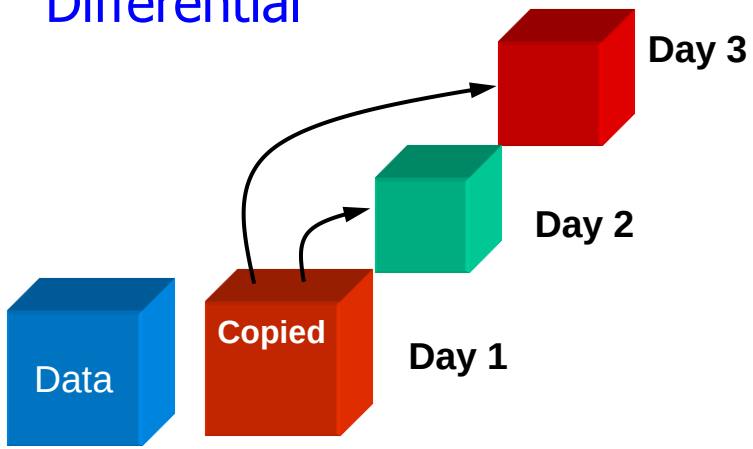
- Normal



- Incremental



- Differential



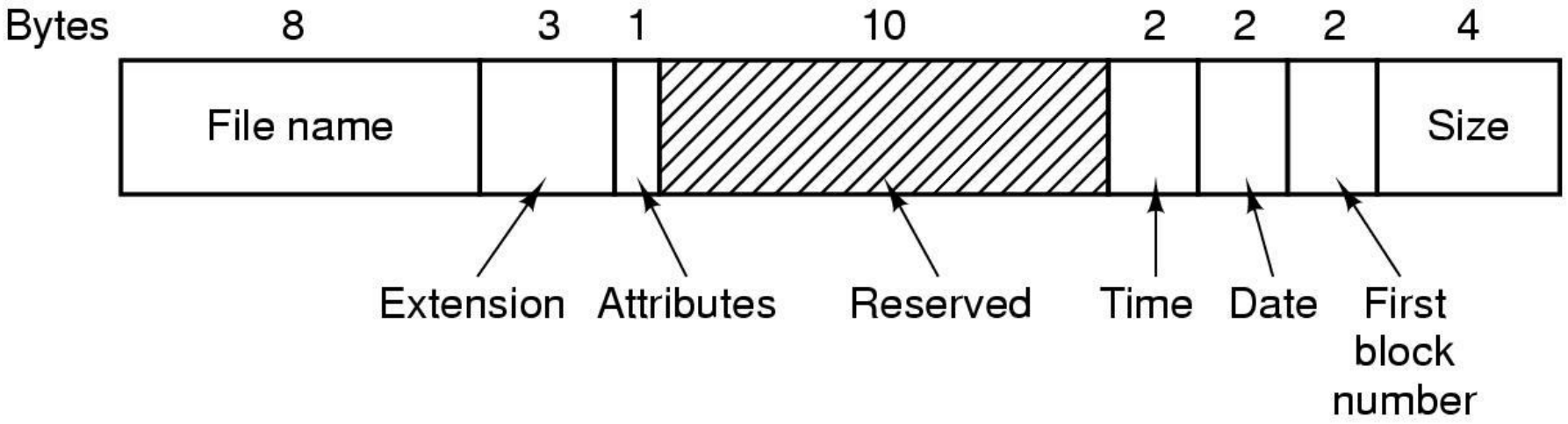
Journaling file system

- 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



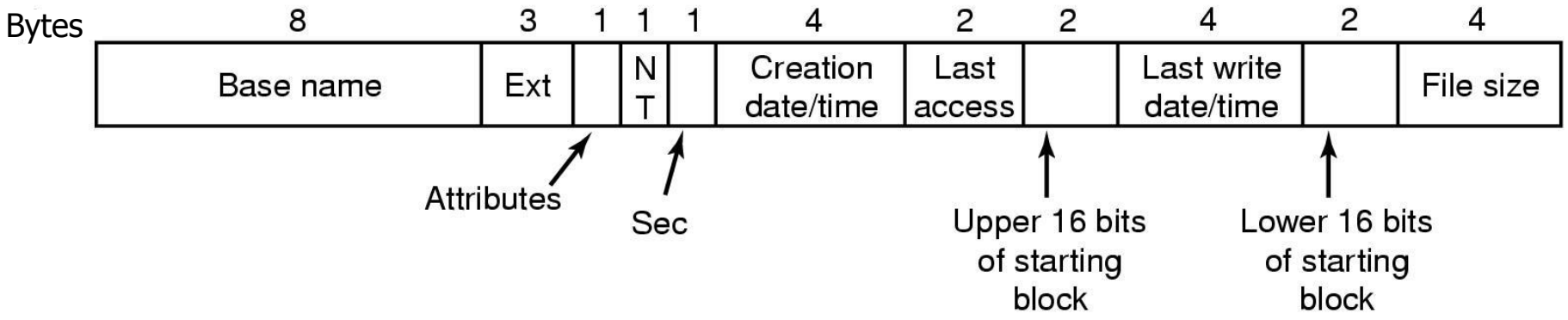
MS-DOS File System (tt.)

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

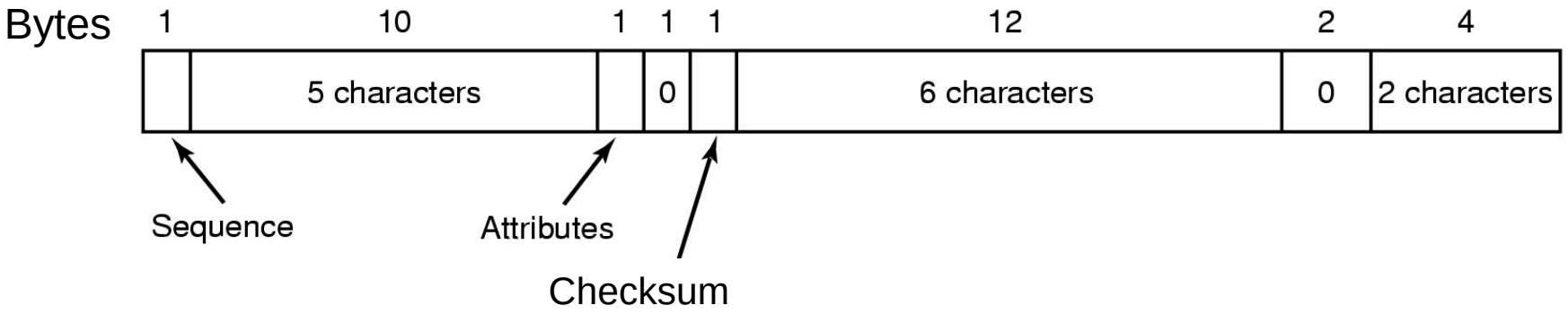
Windows 98 File System

- Extended MOS-DOS directory entry used in Windows 98



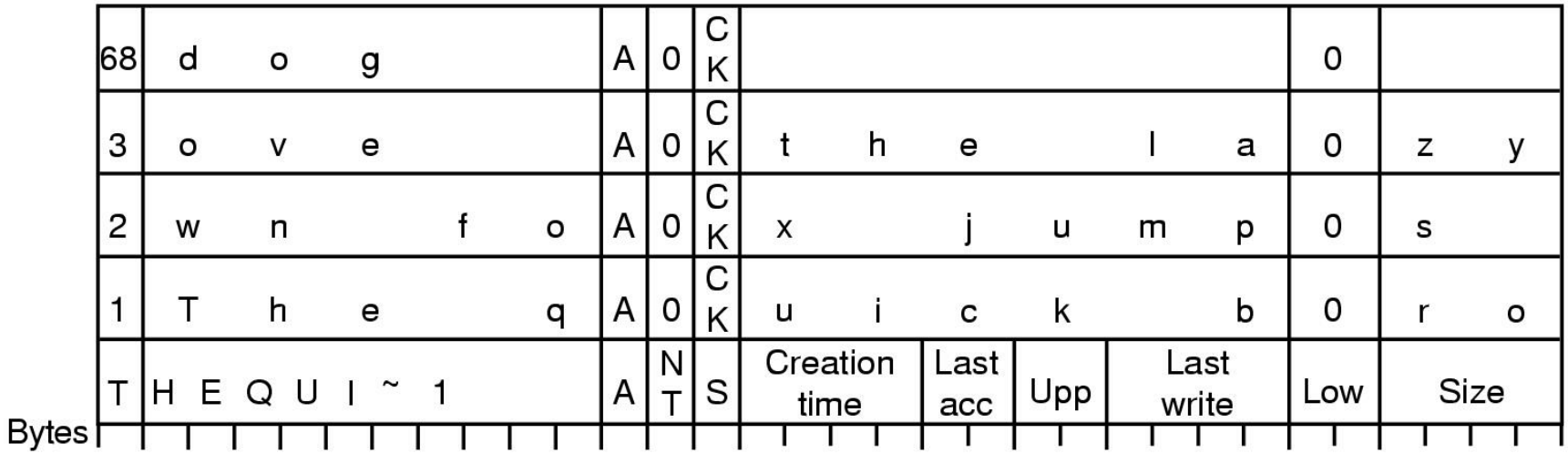
Windows 98 File System (tt.)

- An entry for (part of) a long file name in Windows 98



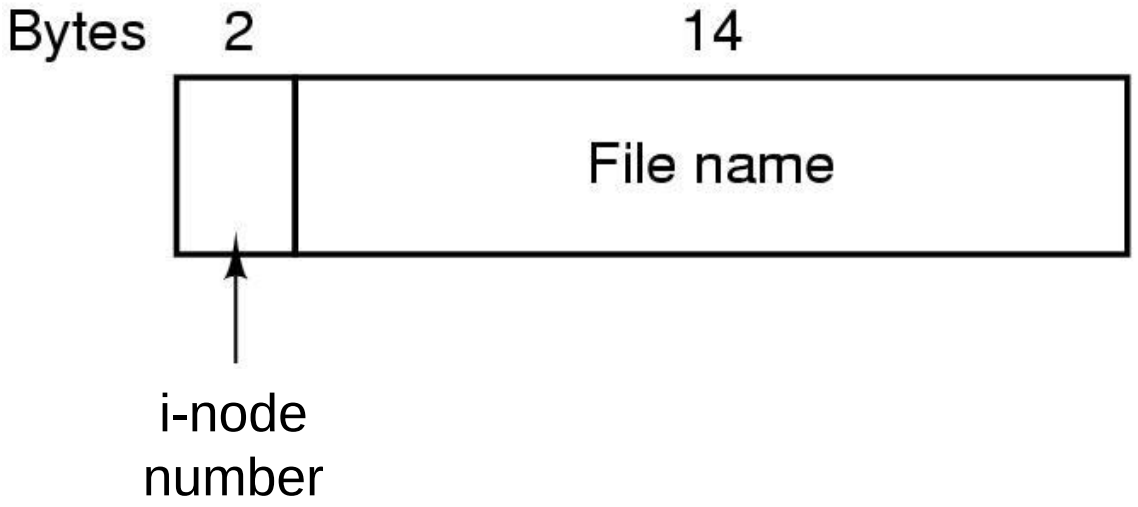
Windows 98 File System (tt.)

- An example of how a long name is stored in Windows 98



UNIX V7 File System

- A UNIX V7 directory entry



UNIX V7 File System (tt.)

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