CÀI ĐẶT HỆ THỐNG TẬP TIN

I Mục đích

Sau khi học xong chương này, người học nắm được những kiến thức sau:

- Hiểu việc lưu trữ các tập tin và truy xuất các tập tin trên các thiết bị lưu trữ phụ.
- Hiểu các phương pháp để thiết lập việc sử dụng tập tin
- Hiểu cách cấp phát không gian đĩa, phục hồi không gian trống, ghi vết vị trí dữ liệu

II Giới thiệu

Trong chương trước chúng ta thấy rằng, hệ thống tập tin cung cấp cơ chế cho việc lưu trữ trực tuyến (on-line storage) và truy xuất tới nội dung tập tin, gồm dữ liệu và chương trình. Hệ thống tập tin định vị vĩnh viễn trên thiết bị lưu trữ phụ. Các thiết bị này được thiết kế để quản lý lượng lớn thông tin không thay đổi.

Chương này tập trung chủ yếu với những vấn đề xoay quanh việc lưu trữ tập tin và truy xuất trên các thiết bị lưu trữ phụ. Chúng ta khám phá các cách để xây dựng cấu trúc sử dụng tập tin, cấp phát không gian đĩa và phục hồi không gian trống để ghi lại vị trí dữ liệu và để giao tiếp với các phần khác của hệ điều hành tới thiết bị lưu trữ phụ. Các vấn đề về năng lực được xem xét thông qua chương này.

III Cấu trúc hệ thống tập tin

Đĩa cung cấp số lượng thiết bị lưu trữ phụ mà trên đó hệ thống tập tin được duy trì. Có hai đặc điểm làm đĩa trở thành phương tiện tiện dụng cho việc lưu trữ nhiều tập tin:

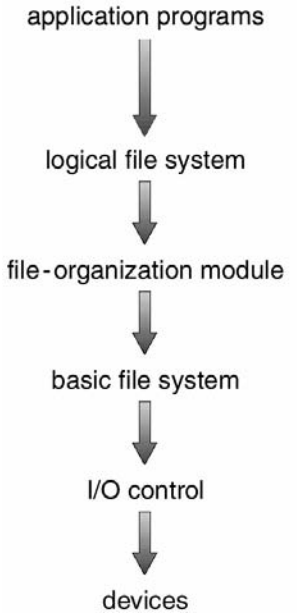
- Chúng có thể được viết lại bằng cách thay thế; có thể đọc một khối từ đĩa, sửa một khối và viết nó ngược trở lại đĩa trong cùng vị trí.
- Chúng có thể được truy xuất trực tiếp bất cứ khối thông tin nào trên đĩa.

Để cải tiến tính hiệu quả nhập/xuất, thay vì chuyển một byte tại một thời điểm, nhập/xuất chuyển giữa bộ nhớ và đĩa được thực hiện trong đơn vị khối. Mỗi khối là một hay nhiều **cung từ** (sector). Phụ thuộc ổ đĩa, các cung từ biến đổi từ 32 bytes tới 4096 bytes; thường là 512 bytes.

Để cung cấp việc truy xuất hiệu quả và tiện dụng tới đĩa, hệ điều hành áp đặt một hay nhiều **hệ thống tập tin** để cho phép dữ liệu được lưu trữ, định vị và truy xuất lại dễ dàng. Một hệ thống tập tin đặt ra hai vấn đề thiết kế rất khác nhau. Vấn đề đầu tiên là định nghĩa hệ thống tập tin nên quan tâm đến người dùng như thế nào. Tác vụ này liên quan đến việc định nghĩa một tập tin và thuộc tính của nó, các thao tác được phép trên một tập tin và các giải thuật và cấu trúc cho việc tổ chức tập tin. Vấn đề thứ hai là tạo giải thuật và cấu trúc dữ liệu để ánh xạ hệ thống tập tin luận lý vào các thiết bị lưu trữ phụ.

Hệ thống tập tin thường được tạo thành từ nhiều cấp khác nhau. Cấu trúc được hiển thị trong hình X-1 là một thí dụ của thiết kế phân cấp. Mỗi cấp trong thiết kế

dùng các đặc điểm của cấp thấp hơn để tạo các đặc điểm mới cho việc sử dụng bởi cấp cao hơn.



Hình 0-1 hệ thống tập tin phân tầng

- **Điều khiển nhập/xuất** (I/O control): là cấp thấp nhất chứa các trình điều khiển thiết bị và các bộ quản lý ngắt để chuyển thông tin giữa bộ nhớ chính và hệ thống đĩa. Trình điều khiển thiết bị thường viết các mẫu bit xác định tới các vị trí trong bộ nhớ của bộ điều khiển nhập/xuất để báo với bộ điều khiển vị trí trên thiết bị nào và hoạt động gì xảy ra.
- **Hệ thống tập tin cơ bản** (basic file system) chỉ cần phát ra các lệnh thông thường tới các trình điều khiển thiết bị tương ứng để đọc và viết các khối vật lý trên đĩa. Mỗi khối vật lý được xác định bởi địa chỉ đĩa (thí dụ, đĩa 1, cyclinder 73, track 2, sector 10).
- **Module tổ chức tập tin** (file-organization module) biết các tập tin và các khối luận lý cũng như các khối vật lý. Bằng cách biết kiểu cấp phát tập tin được dùng và vị trí của tập tin, module tổ chức tập tin có thể dịch các địa chỉ khối luận lý thành các địa chỉ khối vật lý cho hệ thống tập tin cơ bản để truyền. Các khối luận lý của mỗi tập tin được đánh số từ 0 (hay 1) tới N, ngược lại các khối vật lý chứa dữ liệu thường không khớp với các số luận lý vì thế một thao tác dịch được yêu cầu để định vị mỗi khối. Module tổ chức tập tin cũng chứa **bộ quản lý không gian trống** (free-space manager), mà nó ghi vết các khối không được cấp phát và cung cấp các khối này tới module tổ chức tập tin khi được yêu cầu.
- **Hệ thống tập tin luận lý** (logical file system) quản lý thông tin siêu dữ liệu (metadata). Metadata chứa tất cả cấu trúc hệ thống tập tin, ngoại trừ dữ liệu thật sự (hay nội dung của các tập tin). Hệ thống tập tin luận lý quản lý cấu trúc thư mục để cung cấp module tổ chức tập tin những thông tin yêu cầu sau đó, được cho tên tập tin ký hiệu. Nó duy trì cấu trúc tập tin bằng khối điều khiển tập tin. Một khối điều khiển tập tin (file control block-FCB) chứa thông tin về tập tin, gồm người sở hữu, quyền và vị trí của nội dung tập tin.

Nhiều hệ thống tập tin được cài đặt hiện nay. Hầu hết hệ điều hành hỗ trợ nhiều hơn một hệ thống tập tin. Mỗi hệ điều hành có hệ thống tập tin dựa trên cơ sở đĩa. UNIX dùng hệ thống tập tin UNIX (UNIX file system-UFS) như là cơ sở. Windows NT hỗ trợ các định dạng tập tin FAT, FAT32 và NTFS cũng như CD-ROM, DVD và các định dạng hệ thống tập tin đĩa mềm. Bằng cách dùng cấu trúc phân cấp cho việc cài đặt hệ thống tập tin, nên nhân bản mã là tối thiểu. Điều khiển nhập/xuất và mã hệ thống tập tin cơ bản có thể được dùng bởi nhiều hệ thống tập tin. Mỗi hệ thống tập tin có hệ thống tập tin luận lý và module tổ chức tập tin của chính nó.

IV Cài đặt hệ thống tập tin

Trong phần này chúng ta sẽ nghiên cứu các cấu trúc và các thao tác được dùng để cài đặt các thao tác của hệ thống tập tin.

IV.1 Tổng quan

Nhiều cấu trúc trên đĩa và trên bộ nhớ được dùng để cài đặt một hệ thống tập tin. Các cấu trúc này thay đổi dựa trên hệ điều hành và hệ thống tập tin nhưng có một số nguyên tắc chung được áp dụng. Trên đĩa, hệ thống tập tin chứa thông tin về cách khởi động hệ điều hành được lưu trữ ở đó, tổng số khối, số và vị trí của các khối trống, cấu trúc thư mục, các tập tin riêng biệt.

Các cấu trúc trên đĩa gồm:

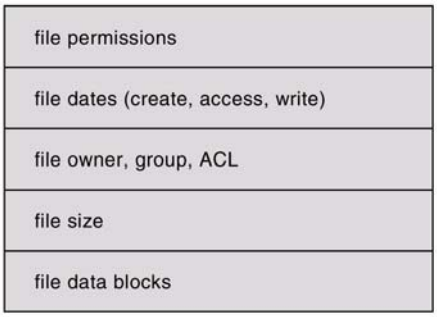
- **Khối điều khiển khởi động** (boot control block) có thể chứa thông tin được yêu cầu bởi hệ thống để khởi động một hệ điều hành từ phân khu đó. Nếu đĩa không chứa hệ điều hành thì khối này là rỗng. Điển hình, nó là khối đầu tiên của đĩa. Trong UFS, khối này được gọi là khối khởi động; trong NTFS, nó là cung khởi động phân khu (partition boot sector).
- **Khối điều khiển phân khu** (partition control block) chứa chi tiết về phân khu, như số lượng khối trong phân khu, kích thước khối, bộ đếm khối trống và con trỏ khối trống, bộ đếm FCB trống và con trỏ FCB. Trong UFS khối này được gọi là **siêu khối** (superblock); trong NTFS, nó là **bảng tập tin chính** (Master File Table)
- Một cấu trúc tập tin được dùng để tổ chức các tập tin
- Một FCB chứa nhiều chi tiết tập tin gồm các quyền tập tin, người sở hữu, kích thước, và vị trí của các khối dữ liệu. Trong UFS khối này được gọi là inode. Trong NTFS, thông tin này được lưu trong Master File Table dùng cấu trúc cơ sở dữ liệu quan hệ với một dòng cho một tập tin.

Thông tin trong bộ nhớ được dùng cho việc quản lý hệ thống tập tin và cải tiến năng lực qua lưu trữ (caching). Các cấu trúc này có thể bao gồm:

- Bảng phân khu trong bộ nhớ chứa thông tin về mỗi phân khu được gắn vào.
- Cấu trúc thư mục trong bộ nhớ quản lý thông tin thư mục của những thư mục vừa được truy xuất. (đối với các thư mục nơi mà các phân khu được gắn vào, nó có thể chứa một con trỏ chỉ tới bảng phân khu.)
- Bảng tập tin đang mở của hệ thống (system-wide open-file table) chứa bản sao của FCB của mỗi tập tin đang mở cũng như các thông tin khác.

- Bảng tập tin đang mở trên quá trình (per-process open-file table) chứa con trỏ chỉ tới mục từ tương ứng trong bảng tập tin đang mở của hệ thống cũng như những thông tin khác.

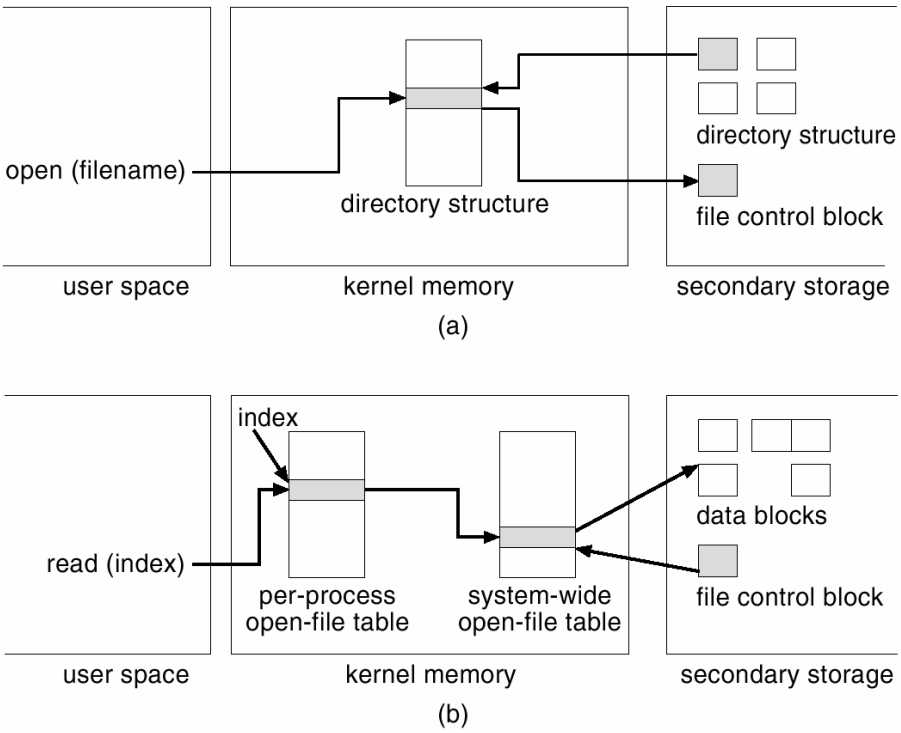
Để tạo một tập tin mới, một chương trình ứng dụng gọi hệ thống tập tin luận lý. Hệ thống tập tin luận lý biết định dạng của các cấu trúc thư mục. Để tạo một tập tin mới, nó cấp phát một FCB mới, đọc thư mục tương ứng vào bộ nhớ, cập nhật nó với tên tập tin mới và FCB, và viết nó trở lại đĩa. Một FCB điển hình được hiển thị trong hình X-2.



Hình 0-2 Một khối điều khiển tập tin điển hình

Một số hệ điều hành như UNIX xem một thư mục như là một tập tin-một tập tin với một trường kiểu hiển thị rằng nó là một thư mục. Các hệ điều hành khác như Windows NT cài đặt các lời gọi hệ thống riêng cho tập tin và thư mục và xem các thư mục như các thực thể tách rời từ các tập tin. Đối với cấu trúc lớn hơn, hệ thống tập tin luận lý có thể gọi module tổ chức tập tin để ánh xạ nhập/xuất thư mục vào số khối đĩa mà chúng được truyền trên cơ sở hệ thống tập tin và hệ thống điều khiển nhập/xuất. Module tổ chức tập tin cũng cấp phát các khối cho việc lưu trữ dữ liệu của tập tin.

Một tập tin được tạo, nó có thể được dùng cho nhập/xuất. Đầu tiên, nó phải được mở. Lời gọi *open* truyền tên tập tin tới hệ thống tập tin. Khi một tập tin được mở, cấu trúc thư mục thường được lưu vào bộ nhớ để tăng tốc độ các thao tác thư mục. Một khi tập tin được tìm thấy, FCB được chép vào bảng tập tin đang mở của hệ thống trong bộ nhớ. Bảng này không chỉ chứa FCB mà còn có các mục từ cho số đếm của số quá trình có mở tập tin.



Hình 0-3 Cấu trúc hệ thống trong bộ nhớ. (a) mở tập tin. (b) đọc tập tin.

Tiếp theo, một mục từ được tạo trong bảng tập tin đang mở trên quá trình, với một con trỏ chỉ tới mục từ trong bảng hệ thống tập tin đang mở của hệ thống và một số trường khác. Các trường khác này có thể chứa con trỏ chỉ tới vị trí hiện hành trong tập tin (cho các thao tác *read* hay *write* tiếp theo) và chế độ truy xuất trong tập tin được mở. Lời gọi *open* trả về một con trỏ chỉ tới mục từ tương ứng trong bảng hệ thống tập tin trên quá trình. Sau đó, tất cả thao tác tập tin được thực hiện bằng con trỏ này. Tên tập tin không phải là một phần của bảng tập tin đang mở, hệ thống không dùng nó một khi FCB tương ứng được định vị trên đĩa. Tên được cho đối với mục từ rất đa dạng. Các hệ thống UNIX chỉ tới nó như một bộ mô tả tập tin (file discriptor); Windows 2000 chỉ tới nó như một bộ quản lý tập tin (file handle). Do đó, với điều kiện là tập tin không đóng, tất cả các thao tác tập tin được thực hiện trên bảng tập tin đang mở.

Khi một quá trình đóng tập tin, mục từ trong bảng trên quá trình bị xoá và bộ đếm số lần mở của mục từ hệ thống giảm. Khi tất cả người dùng đóng tập tin, thông tin tập tin được cập nhật sẽ được chép trở lại tới cấu trúc thư mục dựa trên đĩa và mục từ bảng tập tin đang mở bị xóa.

Trong thực tế, lời gọi hệ thống *open* đầu tiên tìm bảng tập tin đang mở hệ thống để thấy nếu tập tin được sử dụng rồi bởi một quá trình khác. Nếu nó là mục từ bảng tập tin đang mở trên quá trình được tạo để chỉ tới bảng tập tin đang mở hệ thống đã có. Giải thuật này có thể tiết kiệm chi phí khi các tập tin đã mở rồi.
Các cấu trúc điều hành của việc cài đặt hệ thống tập tin được tóm tắt như hình X-3.

IV.2 Hệ thống tập tin ảo

Một phương pháp nổi bật cho việc cài đặt nhiều loại hệ thống tập tin là viết các chương trình con thư mục và tập tin cho mỗi loại. Đúng hơn là hầu hết các hệ điều hành, gồm UNIX, dùng các kỹ thuật hướng đối tượng để đơn giản hóa, tổ chức, và module hóa việc cài đặt. Sử dụng các phương pháp này cho phép nhiều loại hệ thống tập tin khác nhau được cài đặt trong cùng cấu trúc, gồm các hệ thống tập tin mạng như NFS. Người dùng có thể truy xuất các tập tin được chứa trong nhiều hệ thống tập tin trên đĩa cục bộ, hay ngay cả trên các hệ thống tập tin sẵn dùng qua mạng.

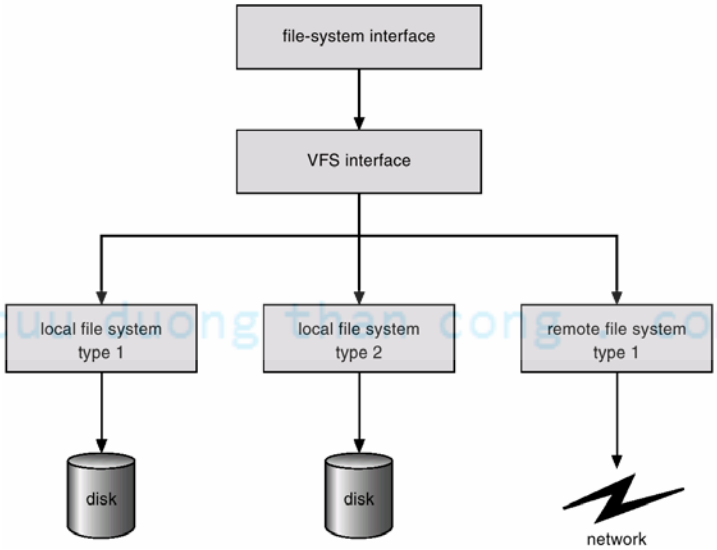
Các cấu trúc dữ liệu và thủ tục được dùng để cô lập chức năng lời gọi hệ thống cơ bản từ các chi tiết cài đặt. Do đó, cài đặt hệ thống tập tin chứa ba tầng chính; nó được mô tả dưới dạng lưu đồ trong hình X-4. Tầng đầu tiên là giao diện hệ thống tập tin dựa trên cơ sở lời gọi *open, read, write, close* và các bộ mô tả tập tin.

Tầng thứ hai được gọi là **hệ thống tập tin ảo** (Virtual File System-VFS); nó phục vụ hai chức năng quan trọng:

1) Nó tách biệt các thao tác hệ thống tập tin giống nhau từ việc cài đặt bằng cách định nghĩa một giao diện VFS rõ ràng. Nhiều cài đặt cho giao diện VFS có thể cùng tồn tại trên cùng một máy, cho phép truy xuất trong suốt tới các loại hệ thống tập tin khác nhau được gắn cục bộ.
2) VFS dựa trên cấu trúc biểu diễn tập tin, được gọi là **vnode**, chứa một bộ gắn bằng số (numerical designator) cho tập tin duy nhất qua mạng. (Inode của UNIX là duy nhất chỉ trong một hệ thống tập tin đơn). Tính duy nhất qua mạng được yêu cầu để hỗ trợ các hệ thống tập tin mạng. Nhân duy trì một cấu trúc vnode cho mỗi nút hoạt động (tập tin hay thư mục).

Do đó, VFS có sự khác biệt các tập tin cục bộ với các tập tin ở xa, và các tập tin cục bộ được phân biệt dựa theo loại hệ thống tập tin của chúng.

VFS kích hoạt các thao tác đặc tả hệ thống tập tin để quản lý các yêu cầu cục bộ dựa theo các loại hệ thống tập tin và ngay cả các lời gọi các thủ tục giao thức NFS cho các yêu cầu ở xa. Quản lý tập tin được xây dựng từ vnode tương ứng và được truyền như các tham số tới các thủ tục này. Tầng cài đặt kiểu hệ thống tập tin, hay giao thức hệ thống tập tin ở xa, là tầng dưới cùng của kiến trúc.



Hình 0-4 Hình ảnh dạng lưu đồ của hệ thống tập tin ảo

V Cài đặt thư mục

Chọn giải thuật cấp phát thư mục và quản lý thư mục có tác động lớn đến tính hiệu quả, năng lực, khả năng tin cậy của hệ thống tập tin. Do đó, chúng ta cần hiểu sự thoả hiệp liên quan trong các giải thuật này.

V.1 Danh sách tuyến tính

Phương pháp đơn giản nhất cho việc cài đặt thư mục là dùng một danh sách tuyến tính chứa tên tập tin với con trỏ chỉ tới các khối dữ liệu. một danh sách tuyến tính với các mục từ thư mục yêu cầu tìm kiếm tuyến tính để xác định một mục từ cụ thể. Phương pháp này đơn giản để lập trình nhưng mất nhiều thời gian để thực thi.

- Để tạo tập tin mới, trước tiên chúng ta phải tìm thư mục để đảm bảo rằng không có tập tin nào tồn tại với cùng một tên. Sau đó, chúng ta thêm một mục từ mới vào cuối thư mục.
- Để xoá một tập tin, chúng ta tìm kiếm thư mục cho tập tin được xác định bởi tên, sau đó giải phóng không gian được cấp phát tới nó.
- Để dùng lại mục từ thư mục, chúng ta có thể thực hiện một vài bước. Chúng ta có thể đánh dấu mục từ như không được dùng (bằng cách gán nó một tên đặc biệt, như một tên trống hay với một bit xác định trạng thái được dùng hoặc không được dùng trong mỗi mục từ), hay chúng ta có thể gán nó tới một danh sách của các mục từ thư mục trống. Một thay đổi thứ ba là chép mục từ cuối cùng trong thư mục vào vị trí trống và giảm chiều dài của thư mục. Một danh sách liên kết có thể được dùng để giảm thời gian xoá một tập tin.

Bất lợi thật sự của danh sách tuyến tính chứa các mục từ thư mục là tìm kiếm tuyến tính để tìm một tập tin. Thông tin thư mục được dùng thường xuyên và người dùng nhận thấy việc truy xuất tới tập tin là chậm. Để khắc phục nhược điểm này, nhiều hệ điều hành cài đặt một vùng lưu trữ phần mềm (software cache) để lưu hầu hết những thông tin thư mục được dùng gần nhất. Một chập dữ liệu được lưu trữ sẽ tránh đọc lại liên tục thông tin từ đĩa. Một danh sách được sắp xếp cho phép tìm kiếm nhị phân và giảm thời gian tìm kiếm trung bình. Tuy nhiên, yêu cầu mà một danh sách phải được sắp xếp có thể phức tạp việc tạo và xoá tập tin vì chúng ta phải di chuyển lượng thông tin liên tục để duy trì một thư mục được xếp thứ tự. Một cấu trúc dữ liệu cây tinh vi hơn như B-tree có thể giúp giải quyết vấn đề. Lợi điểm của danh sách được sắp xếp là liệt kê một thư mục có thứ tự mà không cần một bước sắp xếp riêng.

V.2 Bảng băm

Một cấu trúc dữ liệu khác thường được dùng cho một thư mục tập tin là bảng băm (hash table). Trong phương pháp này, một danh sách tuyến tính lưu trữ các mục từ thư mục nhưng một cấu trúc bảng băm cũng được dùng. Bảng băm lấy một giá trị được tính từ tên tập tin và trả về con trỏ chỉ tới tên tập tin trong danh sách tuyến tính. Do đó, nó có thể giảm rất lớn thời gian tìm kiếm thư mục. Chèn và xoá cũng tương đối đơn giản mặc dù có thể phát sinh đụng độ-những trường hợp có hai tên tập tin được băm cùng vị trí. Khó khăn chính với một bảng băm là kích thước của nó thường cố định và phụ thuộc vào hàm băm trên kích thước đó.

Thí dụ, giả sử rằng chúng ta thực hiện một bảng băm thăm dò tuyến tính quản lý 64 mục từ. Hàm băm chuyển các tập tin thành các số nguyên từ 0 tới 63, thí dụ bằng cách dùng số dư của phép chia cho 64. Sau đó, nếu chúng ta cố tạo tập tin thứ

65, chúng ta phải mở rộng bảng băm thư mục-tới 128 mục từ. Kết quả là chúng ta cần hàm băm mới phải ánh xạ tới dãy 0-127 và chúng ta phải sắp xếp lại các mục từ thư mục đã có để phản ánh giá trị hàm băm mới. Một cách khác, một bảng băm vòng có thể được dùng. Mỗi mục từ băm có thể là danh sách liên kết thay vì chỉ một giá trị riêng và chúng ta có thể giải quyết các đụng độ bằng cách thêm mục từ mới vào danh sách liên kết. Tìm kiếm có thể chậm vì tìm kiếm một tên có thể yêu cầu từ bước thông qua một danh sách liên kết của các mục từ bảng đụng độ; nhưng điều này vẫn nhanh hơn tìm kiếm tuyến tính qua toàn thư mục.

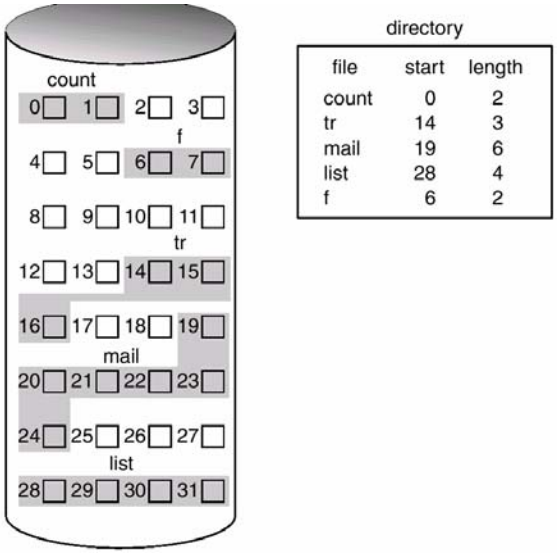
VI Các phương pháp cấp phát

Tính tự nhiên của truy xuất trực tiếp đĩa cho phép chúng ta khả năng linh hoạt trong việc cài đặt tập tin. Trong hầu hết mọi trường hợp, nhiều tập tin sẽ được lưu trên cùng đĩa. Vấn đề chính là không gian cấp phát tới các tập tin này như thế nào để mà không gian đĩa được sử dụng hiệu quả và các tập tin có thể được truy xuất nhanh chóng. Ba phương pháp quan trọng cho việc cấp phát không gian đĩa được sử dụng rộng rãi: cấp phát kề, liên kết và chỉ mục. Mỗi phương pháp có ưu và nhược điểm. Một số hệ thống hỗ trợ cả ba. Thông dụng hơn, một hệ thống sẽ dùng một phương pháp cụ thể cho tất cả tập tin.

VI.1 Cấp phát kề

Phương pháp cấp phát kề yêu cầu mỗi tập tin chiếm một tập hợp các khối kề nhau trên đĩa. Các địa chỉ đĩa định nghĩa một thứ tự tuyến tính trên đĩa. Với thứ tự này, giả sử rằng chỉ một công việc đang truy xuất đĩa, truy xuất khối b+1 sau khi khối b không yêu cầu di chuyển trước. Khi di chuyển đầu đọc được yêu cầu (từ cung từ cuối cùng của cylinder tới cung từ đầu tiên của cylinder tiếp theo), nó chỉ di chuyển một rãnh (track). Do đó, số lượng tìm kiếm đĩa được yêu cầu cho truy xuất kề tới các tập tin được cấp phát là nhỏ nhất, như là thời gian tìm kiếm khi tìm kiếm cuối cùng được yêu cầu. Hệ điều hành IBM VM/CMS dùng cấp phát kề.

Cấp phát kề của một tập tin được định nghĩa bởi địa chỉ đĩa và chiều dài (tính bằng đơn vị khối) của khối đầu tiên. Nếu tập tin có n khối và bắt đầu tại khối b thì nó chiếm các khối b, b+1, b+2,..,b+n-1. Mục từ thư mục cho mỗi tập tin hiển thị địa chỉ của khối bắt đầu và chiều dài của vùng được cấp phát cho tập tin này (hình X-5).



Hình 0-5 Không gian đĩa được cấp phát kề

Truy xuất một tập tin được cấp phát kề rất dễ. Đối với truy xuất tuần tự, hệ thống tập tin nhớ địa chỉ đĩa của khối cuối cùng được tham chiếu và đọc khối tiếp theo. Để truy xuất khối i của tập tin mà bắt đầu tại khối b, chúng ta có thể lập tức truy xuất khối b+i. Do đó, cả truy xuất tuần tự và truy xuất trực tiếp có thể được hỗ trợ bởi cấp phát kề.

Tuy nhiên, cấp phát kề có một số vấn đề. Một khó khăn là tìm không gian cho một tập tin mới. Việc cài đặt của hệ thống quản lý không gian trống xác định tác vụ này được hoàn thành như thế nào. Bất cứ hệ thống quản lý nào có thể được dùng nhưng nhanh chậm khác nhau.

Vấn đề cấp phát không gian kề có thể được xem là vấn đề cấp phát lưu trữ động của ứng dụng để thoả mãn yêu cầu kích thước n từ danh sách các lỗ trống. First fit và best fit là những chiến lược chung nhất được dùng để chọn lỗ trống từ tập hợp các lỗ trống sẵn dùng. Những mô phỏng hiển thị rằng cả hai first fit và best fit hiệu quả hơn worst fit về thời gian và sử dụng không gian lưu trữ. First fit hay best fit đều không phải là giải thuật tốt nhất nhưng thường thì first fit nhanh hơn best fit.

Các giải thuật này gặp phải vấn đề phân mãnh ngoài. Khi các tập tin được cấp phát và xoá, không gian đĩa trống bị chia thành những mảnh nhỏ. Phân mãnh ngoài tồn tại bất cứ khi nào không gian trống được chia thành những đoạn. Nó trở thành một vấn đề khi đoạn kề lớn nhất không đủ cho một yêu cầu; lưu trữ được phân thành nhiều lỗ, không lỗ nào đủ lớn để lưu dữ liệu. Phụ thuộc vào tổng lượng lưu trữ đĩa và kích thước tập tin trung bình, phân mãnh ngoài có thể là vấn đề chính hay phụ.

Một số hệ thống vi tính cũ dùng cấp phát kề trên đĩa mềm. Để ngăn ngừa lượng lớn không gian đĩa phân mãnh ngoài, người dùng phải chạy thủ tục ***đóng gói lại***. Thủ tục này chép toàn bộ hệ thống tập tin tới một đĩa khác hay trên băng từ. Kế đến, đĩa mềm ban đầu được giải phóng hoàn toàn, tạo một không gian trống kề lớn. Sau đó, thủ tục này chép các tập tin trở lại trên đĩa mềm bằng cách cấp phát không gian kề từ một lỗ lớn. Cơ chế này hiệu quả trong việc hợp nhất (compacts) tất cả không gian trống thành một không gian trống kề, giải quyết vấn đề phân mãnh. Chi phí cho việc hợp nhất là thời gian. Chi phí thời gian phục vụ cho các đĩa cứng lớn dùng cấp phát kề, ở đây hợp nhất tất cả không gian mất hàng giờ. Trong thời gian này, các thao tác hệ thống thông thường không thể được phép vì thế hợp nhất tránh được tất cả chi phí do có thay đổi về dữ liệu.

Một vấn đề khác với cấp phát kề là xác định bao nhiêu không gian được yêu cầu cho một tập tin. Khi một tập tin được tạo, toàn bộ không gian nó cần phải được tìm kiếm và được cấp phát. Người tạo (chương trình hay người) biết kích thước tập tin được tạo như thế nào? Trong một số trường hợp việc xác định này tương đối đơn giản (thí dụ chép một tập tin đã có); tuy nhiên, kích thước của tập tin xuất có thể khó để ước lượng.

Nếu chúng ta cấp quá ít không gian tới một tập tin, chúng ta thấy rằng tập tin không thể mở rộng. Đặc biệt với một chiến lược cấp phát best fit, không gian trên cả hai phía của tập tin đang được dùng. Do đó, chúng ta không thể làm cho tập tin lớn hơn. Hai khả năng có thể. Thứ nhất, chương trình người dùng có thể được kết thúc với một thông báo lỗi hợp lý. Sau đó, người dùng phải cấp phát nhiều không gian hơn và chạy chương trình lại. Việc lặp này có thể gây ra chi phí. Để ngăn chặn chúng, người dùng sẽ mô phỏng nhiều hơn lượng không gian được yêu cầu. Điều này dẫn đến không gian bị lãng phí.

Một khả năng khác là tìm một lỗ trống lớn hơn, chép nội dung của tập tin tới không gian trống mới, và giải phóng không gian trước đó. Một loạt các hoạt động này có thể được lặp lại với điều kiện là không gian tồn tại mặc dù nó tiêu tốn nhiều thời gian. Tuy nhiên, trong trường hợp này người dùng không bao giờ yêu cầu được thông báo về những gì đang xảy ra; hệ thống tiếp tục mặc dù vấn đề phát sinh.
Ngay cả nếu toàn bộ không gian được yêu cầu cho tập tin được biết trước, cấp phát trước là không đủ. Một tập tin sẽ lớn lên trong khoảng thời gian dài phải được cấp phát đủ không gian cho kích thước cuối cùng của nó mặc dù không gian đó có thể không được dùng cho khoảng thời gian dài. Do đó, tập tin có lượng lớn phân mãnh trong.

Để tối thiểu các khó khăn này, một số hệ điều hành dùng một cơ chế cấp phát kề được hiệu chỉnh. Trong cơ chế này đoạn không gian kề được cấp phát trước và sau đó khi lượng không gian đó không đủ lớn, một đoạn không gian kề khác, một ***đoạn mở rộng*** (extent), được thêm vào cấp phát ban đầu. Sau đó, vị trí của các khối tập tin được ghi lại như một vị trí và một bộ đếm khối cộng với một liên kết tới khối đầu tiên của đoạn mở rộng tiếp theo. Trên một số hệ thống, người sở hữu tập tin có thể đặt kích thước đoạn mở rộng, nhưng việc đặt này có thể không hiệu quả nếu người sở hữu không đúng. Phân mãnh trong vẫn còn là vấn đề nếu đoạn mở rộng quá lớn và phân mãnh ngoài có thể là vấn đề khi các đoạn mở rộng có kích thước khác nhau được cấp phát và thu hồi.

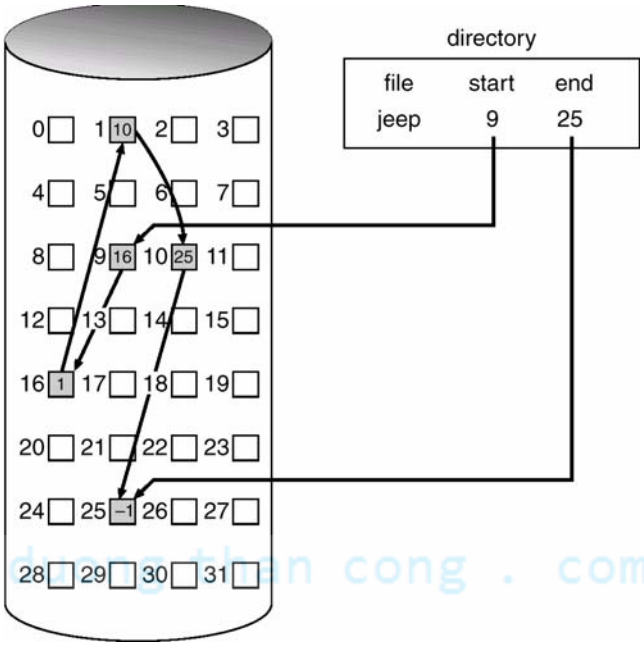
VI.2 Cấp phát liên kết

Cấp phát liên kết giải quyết vấn đề của cấp phát kề. Với cấp phát liên kết, mỗi tập tin là một danh sách các khối đĩa được liên kết; các khối đĩa có thể được phân tán khắp nơi trên đĩa. Thư mục chứa một con trỏ chỉ tới khối đầu tiên và các khối cuối cùng của tập tin. Thí dụ, một tập tin có 5 khối có thể bắt đầu tại khối số 9, tiếp tục là khối 16, sau đó khối 1, khối 10 và cuối cùng khối 25 (như hình X-6). Mỗi khối chứa một con trỏ chỉ tới khối kế tiếp. Các con trỏ này không được làm sẵn dùng cho người dùng. Do đó, nếu mỗi khối là 512 bytes, và địa chỉ đĩa (con trỏ) yêu cầu 4 bytes thì phần chứa dữ liệu của khối là 508 bytes.

Để tạo một tập tin mới, chúng ta đơn giản tạo một mục từ mới trong thư mục. Với cấp phát liên kết, mỗi mục từ thư mục có một con trỏ chỉ tới khối đĩa đầu tiên của tập tin. Con trỏ này được khởi tạo tới *nil* (giá trị con trỏ cuối danh sách) để chỉ một tập tin rỗng. Trường kích thước cũng được đặt tới 0. Một thao tác viết tới tập tin làm một khối trống được tìm thấy bằng hệ thống quản lý không gian trống, sau đó khối

mới này được viết tới và được liên kết tới cuối tập tin. Để đọc một tập tin, chúng ta đơn giản đọc các khối bằng cách lần theo các con trỏ từ khối này tới khối khác.

Không có sự phân mãnh ngoài với cấp phát liên kết, và bất cứ khối trống trên danh sách không gian trống có thể được dùng để thoả mãn yêu cầu. Kích thước của một tập tin không cần được khai báo khi tập tin đó được tạo. Một tập tin có thể tiếp tục lớn lên với điều kiện là các khối trống sẵn có. Do đó, nó không bao giờ cần thiết để hợp nhất không gian trống.



Hình 0-6 cấp phát không gian đĩa liên kết

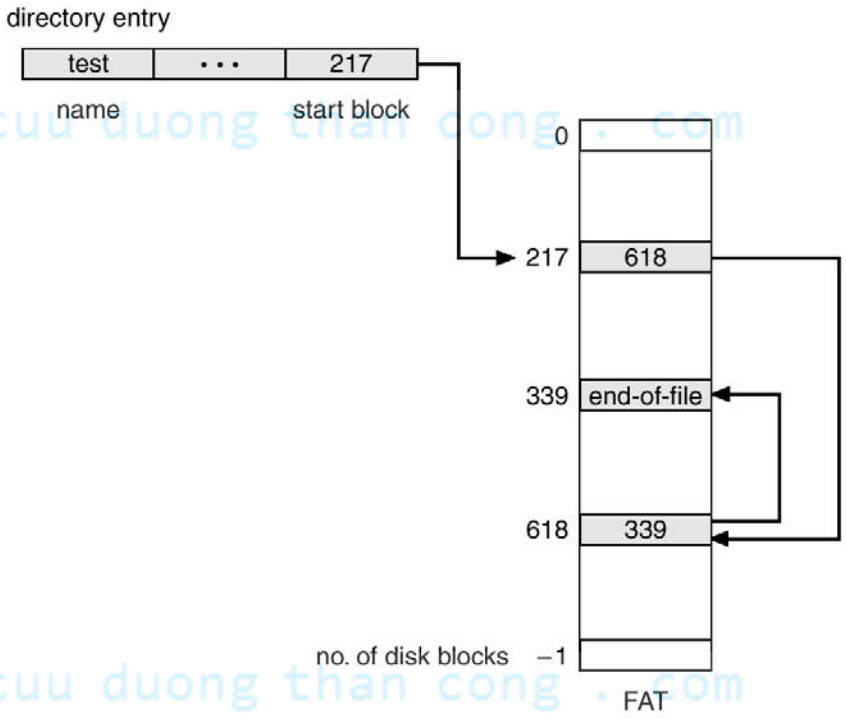
Tuy nhiên, cấp phát liên kết có một vài nhược điểm. Vấn đề chủ yếu là nó có thể được dùng hiệu quả chỉ cho các tập tin truy xuất tuần tự. Để tìm khối thứ i của tập tin, chúng ta phải bắt đầu tại điểm bắt đầu của tập tin đó, và lần theo con trỏ cho đến khi chúng ta nhận được khối thứ i. Mỗi truy xuất tới con trỏ yêu cầu một thao tác đọc đĩa, và đôi khi là một tìm kiếm đĩa. Do đó, nó không đủ hỗ trợ một khả năng truy xuất trực tiếp cho các tập tin cấp phát liên kết.

Một nhược điểm khác của cấp phát liên kết là không gian được yêu cầu cho các con trỏ. Nếu một con trỏ yêu cầu 4 bytes của khối 512 bytes thì 0.77% của đĩa được dùng cho các con trỏ thay vì là thông tin.

Một giải pháp thông thường để giải quyết vấn đề này là tập hợp các khối vào các **nhóm** (clusters) và cấp phát các nhóm hơn là các khối. Thí dụ, hệ thống tập tin có thể định nghĩa nhóm gồm 4 khối và thao tác trên đĩa chỉ trong đơn vị nhóm thì các con trỏ dùng % nhỏ hơn của không gian của tập tin. Phương pháp này cho phép ánh xạ khối luận lý tới vật lý vẫn còn đơn giản, nhưng cải tiến thông lượng đĩa và giảm không gian được yêu cầu cho cấp phát khối và quản lý danh sách trống. Chi phí của tiếp cận này là tăng phân mãnh trong vì nhiều không gian hơn bị lãng phí nếu một nhóm chỉ đầy một phần hơn là một khối đầy một phần. Các nhóm có thể được dùng để cải tiến thời gian truy xuất đĩa cho nhiều giải thuật khác nhau vì thế chúng được dùng trong hầu hết các hệ điều hành.

Một vấn đề khác của cấp phát liên kết là khả năng tin cậy. Vì các tập tin được liên kết với nhau bởi các con trỏ được phân tán khắp đĩa, xem xét điều gì xảy ra nếu một con trỏ bị mất hay bị phá hỏng. Một con bọ (bug) trong phần mềm hệ điều hành hay lỗi phần cứng đĩa có thể dẫn tới việc chọn con trỏ sai. Lỗi này có thể dẫn tới việc liên kết vào danh sách không gian trống hay vào một tập tin khác. Các giải pháp một phần là dùng các danh sách liên kết đôi hay lưu tên tập tin và số khối tương đối trong mỗi khối; tuy nhiên, các cơ chế này yêu cầu nhiều chi phí hơn cho mỗi tập tin.

Một thay đổi quan trọng trên phương pháp cấp phát liên kết là dùng **bảng cấp phát tập tin** (file allocation table-FAT). Điều này đơn giản nhưng là phương pháp cấp phát không gian đĩa hiệu quả được dùng bởi hệ điều hành MS-DOS và OS/2. Một phần đĩa tại phần bắt đầu của mỗi phân khu được thiết lập để chứa bảng này. Bảng này có một mục từ cho mỗi khối đĩa và được lập chỉ mục bởi khối đĩa. FAT được dùng nhiều như là một danh sách liên kết. Mục từ thư mục chứa số khối của khối đầu tiên trong tập tin. Mục từ bảng được lập chỉ mục bởi số khối đó sau đó chứa số khối của khối tiếp theo trong tập tin. Chuỗi này tiếp tục cho đến khi khối cuối cùng, có giá trị cuối tập tin đặc biệt như mục từ bảng. Các khối không được dùng được hiển thị bởi giá trị bảng 0. Cấp phát một khối mới tới một tập tin là một vấn đề đơn giản cho việc tìm mục từ bảng có giá trị 0 đầu tiên và thay thế giá trị kết thúc tập tin trước đó với địa chỉ của khối mới. Sau đó, số 0 được thay thế với giá trị kết thúc tập tin. Một thí dụ minh hoạ là cấu trúc FAT của hình X-7 cho một tập tin chứa các khối đĩa 217, 618 và 339.



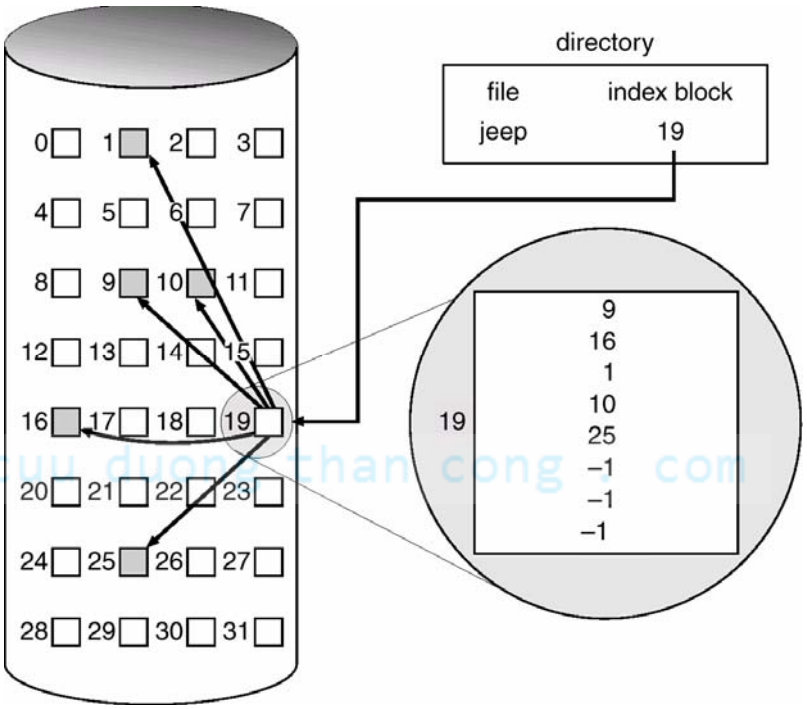
Hình 0-7 Bảng cấp phát tập tin

Cơ chế cấp phát FAT có thể dẫn tới số lượng lớn tìm kiếm đầu đọc đĩa nếu FAT không được lưu trữ(cache). Đầu đọc đĩa phải di chuyển tới điểm bắt đầu của phân khu để đọc FAT và tìm vị trí khối sau đó di chuyển tới vị trí của chính khối đĩa đó. Trong trường hợp xấu nhất, cả hai di chuyển xảy ra cho mỗi khối đĩa. Lợi điểm là thời gian truy xuất ngẫu nhiên được cải tiến vì đầu đọc đĩa có thể tìm vị trí của bất cứ khối nào bằng cách đọc thông tin trong FAT.

VI.3 Cấp phát được lập chỉ mục

Cấp phát liên kết giải quyết việc phân mãnh ngoài và vấn đề khai báo kích thước của cấp phát kề. Tuy nhiên, cấp phát liên kết không hỗ trợ truy xuất trực tiếp hiệu quả vì các con trỏ chỉ tới các khối được phân tán với chính các khối đó qua đĩa và cần được lấy lại trong thứ tự. Cấp phát được lập chỉ mục giải quyết vấn đề này bằng cách mang tất cả con trỏ vào một vị trí: **khối chỉ mục** (index block).

Mỗi tập tin có khối chỉ mục của chính nó, khối này là một mảng các địa chỉ khối đĩa. Mục từ thứ i trong khối chỉ mục chỉ tới khối i của tập tin. Thư mục chứa địa chỉ của khối chỉ mục (như hình X-8). Để đọc khối i, chúng ta dùng con trỏ trong mục từ khối chỉ mục để tìm và đọc khối mong muốn. Cơ chế này tương tự như cơ chế phân trang.



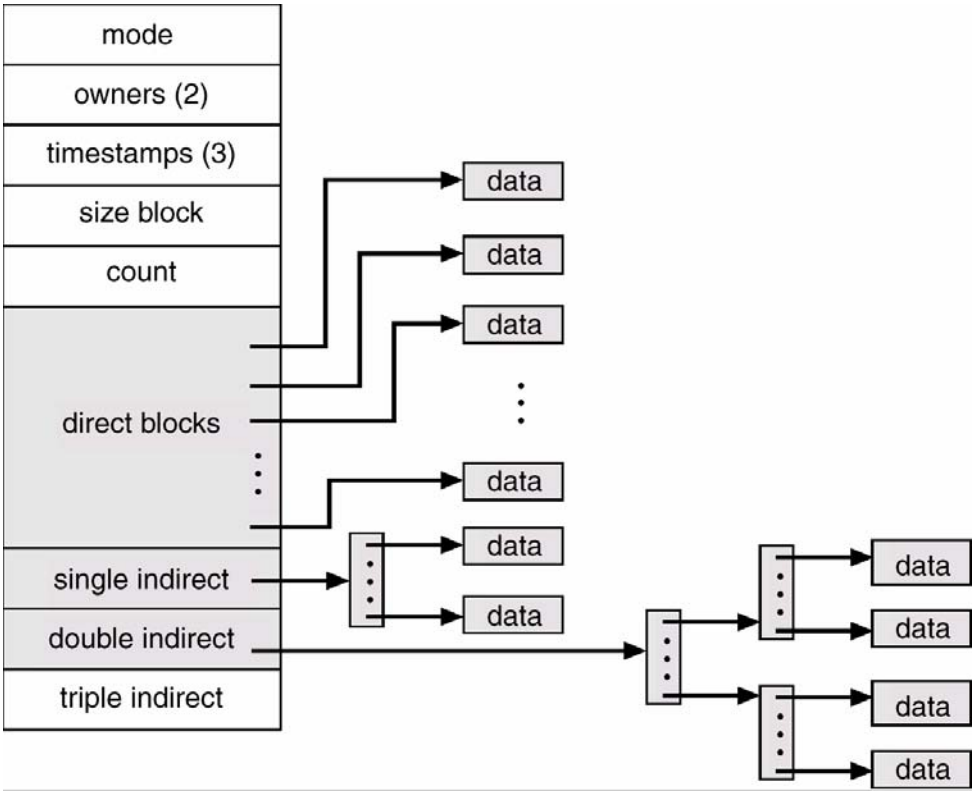
Hình 0-8 Cấp phát không gian đĩa được lập chỉ mục

Khi một tập tin được tạo, tất cả con trỏ trong khối chỉ mục được đặt tới nil. Khi khối thứ i được viết đầu tiên, khối được chứa từ bộ quản lý không gian trống và địa chỉ của nó được đặt trong mục từ khối chỉ mục.

Cấp phát được lập chỉ mục hỗ trợ truy xuất trực tiếp, không gặp phải sự phân mãnh ngoài vì bất cứ khối trống trên đĩa có thể đáp ứng yêu cầu thêm không gian. Cấp phát được lập chỉ mục gặp phải sự lãng phí không gian. Chi phí con trỏ của khối chỉ mục thường lớn hơn chi phí con trỏ của cấp phát liên kết. Xét trường hợp thông thường trong đó chúng ta có một tập tin với chỉ một hoặc hai khối. Với cấp phát liên kết, chúng ta mất không gian của chỉ một con trỏ trên khối (một hay hai con trỏ). Với cấp phát được lập chỉ mục, toàn bộ khối chỉ mục phải được cấp phát thậm chí nếu một hay hai con trỏ là khác nil.

Điểm này sinh ra câu hỏi khối chỉ mục nên lớn bao nhiêu? Mỗi tập tin phải có một khối chỉ mục để mà chúng ta muốn khối chỉ mục nhỏ nhất có thể. Tuy nhiên, nếu khối chỉ mục quá nhỏ nó không thể quản lý đủ các con trỏ cho một tập tin lớn và một cơ chế sẽ phải sẵn có để giải quyết vấn đề này:

- **Cơ chế liên kết** (linked scheme): một khối chỉ mục thường là một khối đĩa. Do đó, nó có thể được đọc và viết trực tiếp bởi chính nó. Để cho phép đối với các tập tin lớn, chúng ta có thể liên kết nhiều khối chỉ mục với nhau. Thí dụ, một khối chỉ mục có thể chứa một header nhỏ cho tên tập tin và một tập hợp của các địa chỉ 100 khối đĩa đầu tiên. Địa chỉ tiếp theo (từ cuối cùng trong khối chỉ mục) là nil (đối với một tập tin nhỏ) hay một con trỏ tới khối chỉ mục khác (cho một tập tin lớn)
- **Chỉ mục nhiều cấp** (multilevel index): một biến dạng của biểu diễn liên kết là dùng khối chỉ mục cấp 1 để chỉ tới khối chỉ mục cấp 2. Khối cấp 2 chỉ tới các khối tập tin. Để truy xuất một khối, hệ điều hành dùng chỉ mục cấp 1 để tìm một khối chỉ mục cấp 2 và khối đó tìm khối dữ liệu mong muốn. Tiếp cận này có thể được tiếp tục tới cấp 3 hay cấp 4, tuỳ thuộc vào kích thước tập tin lớn nhất được mong muốn. Với khối có kích thước 4,096 bytes, chúng ta có thể lưu 1,024 con trỏ 4 bytes trong một khối chỉ mục. Chỉ mục hai cấp cho phép 1,048,576 khối dữ liệu, cho phép tập tin có kích thước tới 4GB.
- **Cơ chế kết hợp** (combined scheme): một biến dạng khác được dùng trong UFS là giữ 15 con trỏ đầu tiên của khối chỉ mục trong inode của tập tin. 12 con trỏ đầu tiên của 15 con trỏ này chỉ tới khối trực tiếp (direct blocks); nghĩa là chúng chứa các địa chỉ của khối mà chứa dữ liệu của tập tin. Do đó, dữ liệu đối với các tập tin nhỏ (không lớn hơn 12 khối) không cần một khối chỉ mục riêng. Nếu kích thước khối là 4 KB, thì tới 48 KB dữ liệu có thể truy xuất trực tiếp. 3 con trỏ tiếp theo chỉ tới các khối gián tiếp (indirect blocks). Con trỏ khối gián tiếp thứ nhất là địa chỉ của khối gián tiếp đơn (single indirect blocks). Khối gián tiếp đơn là một khối chỉ mục không chứa dữ liệu nhưng chứa địa chỉ của các khối chứa dữ liệu. Sau đó, có con trỏ khối gián tiếp đôi (double indirect block) chứa địa chỉ của một khối mà khối này chứa địa chỉ của các khối chứa con trỏ chỉ tới khối dữ liệu thật sự. Con trỏ cuối cùng chứa chứa địa chỉ của khối gián tiếp ba (triple indirect block). Với phương pháp này, số khối có thể được cấp phát tới một tập tin vượt quá lượng không gian có thể đánh địa chỉ bởi các con trỏ tập tin 4 bytes hay 4 GB. Nhiều cài đặt UNIX gồm Solaris và AIX của IBM hỗ trợ tới 64 bit con trỏ tập tin. Các con trỏ có kích thước này cho phép các tập tin và hệ thống tập tin có kích thước tới terabytes. Một inode được hiển thị trong hình X-9:



Hình 0-9 Inode của UNIX

Cơ chế cấp phát lập chỉ mục gặp một số khó khăn về năng lực như cấp phát liên kết. Đặc biệt, các khối chỉ mục có thể được lưu trữ (cache) trong bộ nhớ; nhưng các khối dữ liệu có thể được trãi rộng khắp phân khu.

VI.4 Năng lực

Các phương pháp cấp phát ở trên khác nhau về tính hiệu quả lưu trữ và thời gian truy xuất khối dữ liệu. Cả hai yếu tố này là tiêu chuẩn quan trọng trong việc chọn phương pháp hợp lý hay các phương pháp cho một hệ điều hành cài đặt.
Trước khi chọn một phương pháp, chúng ta cần xác định hệ thống sẽ được dùng như thế nào. Một hệ thống với hầu hết truy xuất tuần tự nên dùng một phương pháp khác từ hệ thống với hầu hết truy xuất ngẫu nhiên. Đối với bất cứ loại truy xuất nào, cấp phát kề yêu cầu chỉ một truy xuất để đạt được một khối đĩa. Vì chúng ta có thể giữ dễ dàng địa chỉ khởi đầu của tập tin trong bộ nhớ, chúng ta có thể tính lập tức địa chỉ đĩa của khối thứ i (hay khối kế tiếp) và đọc nó trực tiếp.

Đối với cấp phát liên kết, chúng ta cũng có thể giữ địa chỉ khối kế tiếp trong bộ nhớ và đọc nó trực tiếp. Phương pháp này là tốt cho truy xuất tuần tự; tuy nhiên, đối với truy xuất trực tiếp một truy xuất tới khối thứ i phải yêu cầu đọc I đĩa. Vấn đề này minh hoạ lý do cấp phát liên kết không được dùng cho một ứng dụng yêu cầu truy xuất trực tiếp.

Do đó, một số hệ thống hỗ trợ các tập tin truy xuất trực tiếp bằng cách dùng cấp phát kề và truy xuất tuần tự bởi cấp phát liên kết. Đối với các hệ thống này, loại truy xuất được thực hiện phải được khai báo khi tập tin được tạo. Một tập tin được tạo cho truy xuất tuần tự sẽ được liên kết và không thể được dùng cho truy xuất trực tiếp. Một tập tin được tạo cho truy xuất trực tiếp sẽ kề nhau và có thể hỗ trợ cả hai truy xuất trực tiếp và truy xuất tuần tự nhưng chiều dài tối đa của nó phải được khai báo

khi nó được tạo. Trong trường hợp này, hệ điều hành phải có cấu trúc dữ liệu hợp lý và các giải thuật để hỗ trợ cả hai phương pháp cấp phát. Các tập tin có thể được chuyển từ một kiểu này sang một kiểu khác bằng cách tạo một tập tin mới của loại mong muốn và các nội dung của tập tin cũ được chép vào tập tin mới. Sau đó, tập tin cũ có thể bị xoá và tập tin mới được đổi tên.

Cấp phát dạng chỉ mục phức tạp hơn. Nếu khối chỉ mục đã ở trong bộ nhớ rồi thì truy xuất có thể được thực hiện trực tiếp. Tuy nhiên, giữ khối chỉ mục trong bộ nhớ yêu cầu không gian có thể xem xét. Nếu không gian bộ nhớ này không sẵn dùng thì chúng ta phải đọc trước khối chỉ mục và sau đó khối dữ liệu mong muốn. Đối với chỉ mục hai cấp, đọc hai khối chỉ mục là cần thiết. Đối với tập tin rất lớn, truy xuất một khối gần cuối tập tin yêu cầu đọc tất cả khối chỉ mục để lần theo chuỗi con trỏ trước khi khối dữ liệu được yêu cầu cuối cùng được đọc. Do đó, năng lực của cấp phát chỉ mục phụ thuộc cấu trúc chỉ mục trên kích thước tập tin và vị trí của khối mong muốn.

Một số hệ thống kết hợp cấp phát kề và cấp phát chỉ mục bằng cách dùng cấp phát kề cho các tập tin nhỏ (ba hay bốn khối) và tự động chuyển tới cấp phát chỉ mục nếu tập tin lớn lên. Vì hầu hết các tập tin là nhỏ và cấp phát kề là hiệu quả cho các tập tin nhỏ, năng lực trung bình là rất tốt.

Nhiều tối ưu khác là có thể và đang được dùng. Với sự chênh lệch tốc độ giữa CPU và đĩa, nó là không hợp lý để thêm hàng ngàn chỉ thị tới hệ điều hành để tiết kiệm chỉ một vài di chuyển của đầu đọc. Ngoài ra, sự chênh lệch này tăng theo thời gian, tới điểm nơi mà hàng trăm của hàng ngàn chỉ thị phù hợp có thể được dùng để tối ưu sự di chuyển của đầu đọc.

VII Quản lý không gian trống

Vì không gian trống là giới hạn nên chúng ta cần dùng lại không gian từ các tập tin bị xoá cho các tập tin mới nếu có thể. Để giữ vết của không gian đĩa trống, hệ thống duy trì một danh sách không gian trống. Danh sách không gian trống ghi lại tất cả khối đĩa trống. Để tạo tập tin, chúng ta tìm trong danh sách không gian trống lượng không gian được yêu cầu và cấp phát không gian đó tới tập tin mới. Sau đó, không gian này được xoá từ danh sách không gian trống. Khi một tập tin bị xoá, không gian đĩa của nó được thêm vào danh sách không gian trống. Mặc dù tên của nó là danh sách nhưng danh sách không gian trống có thể không được cài như một danh sách.

VII.1 Bit vector

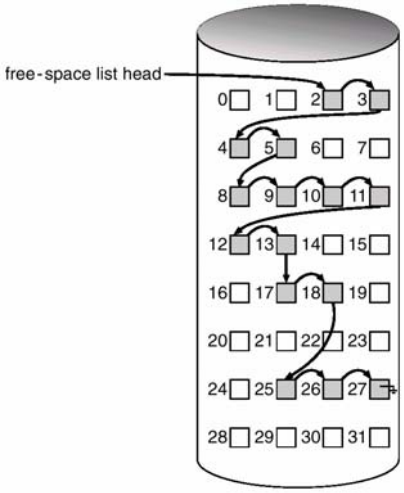
Thường thì danh sách không gian trống được cài đặt như một bản đồ bit (bit map) hay một vector bit (bit vector). Mỗi khối được biểu diễn bởi 1 bit. Nếu khối là trống, bit của nó được đặt là 1, nếu khối được cấp phát bit của nó được đặt là 0. Thí dụ, xét một đĩa khi các khối 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, và 27 là trống và các khối còn lại được cấp phát. Bản đồ bit không gian trống sẽ là:
0011110011111100011000000111000000…

Lợi điểm chính của tiếp cận này là tính tương đối đơn giản và hiệu quả của nó trong việc tìm khối trống đầu tiên, hay n khối trống tiếp theo trên đĩa.

Một lần nữa, chúng ta thấy các đặc điểm phần cứng định hướng chức năng phần mềm. Tuy nhiên, các vector bit là không đủ trừ khi toàn bộ vector được giữ trong bộ nhớ chính. Giữ nó trong bộ nhớ chính là có thể cho các đĩa nhỏ hơn, như trên các máy vi tính nhưng không thể cho các máy lớn hơn. Một đĩa 1.3 GB với khối 512

bytes sẽ cần một bản đồ bit 332 KB để ghi lại các khối trống. Gom bốn khối vào một nhóm có thể giảm số này xuống còn 83 KB trên đĩa.

VII.2 Danh sách liên kết



Hình 0-10 danh sách không gian trống được liên kết trên đĩa

Một tiếp cận khác để quản lý bộ nhớ trống là liên kết tất cả khối trống, giữ một con trỏ tới khối trống đầu tiên trong một vị trí đặc biệt trên đĩa và lưu nó trong bộ nhớ. Khối đầu tiên này chứa con trỏ chỉ tới khối đĩa trống tiếp theo,..Trong thí dụ trên, chúng ta có thể giữ một con trỏ chỉ tới khối 2 như là khối trống đầu tiên. Khối 2 sẽ chứa một con trỏ chỉ tới khối 3, khối này sẽ chỉ tới khối 4,…(như hình X-10). Tuy nhiên, cơ chế này không hiệu quả để duyệt danh sách, chúng ta phải đọc mỗi khối, yêu cầu thời gian nhập/xuất đáng kể. Tuy nhiên, duyệt danh sách trống không là hoạt động thường xuyên. Thường thì, hệ điều hành cần một khối trống để mà nó có thể cấp phát khối đó tới một tập tin, vì thế khối đầu tiên trong danh sách trống được dùng. Phương pháp FAT kết hợp với đếm khối trống thành cấu trúc dữ liệu cấp phát.

VII.3 Nhóm

Thay đổi tiếp cận danh sách trống để lưu địa chỉ của n khối trống trong khối trống đầu tiên. n-1 khối đầu tiên này thật sự là khối trống. Khối cuối cùng chứa địa chỉ của n khối trống khác, …Sự quan trọng của việc cài đặt này là địa chỉ của một số lượng lớn khối trống có thể được tìm thấy nhanh chóng, không giống như trong tiếp cận danh sách liên kết chuẩn.

VII.4 Bộ đếm

Một tiếp cận khác đạt được lợi điểm trong thực tế là nhiều khối kề có thể được cấp phát và giải phóng cùng lúc, đặc biệt khi không gian được cấp phát với giải thuật cấp phát kề hay thông qua nhóm. Do đó, thay vì giữ một danh sách n địa chỉ đĩa trống, chúng ta có thể giữ địa chỉ của khối trống đầu tiên và số n khối kề trống theo sau khối đầu tiên. Mỗi mục từ trong danh sách không gian trống sau đó chứa một địa chỉ đĩa và bộ đếm. Mặc dù mỗi mục từ yêu cầu nhiều không gian hơn một địa chỉ đĩa đơn, nhưng toàn bộ danh sách sẽ ngắn hơn với điều kiện là bộ đếm lớn hơn 1.

VIII Tóm tắt

Hệ thống tập tin định vị không đổi trên thiết bị lưu trữ phụ được thiết kế để quản lý một lượng lớn dữ liệu không đổi. Phương tiện lưu trữ phụ phổ biến nhất là đĩa.

Đĩa vật lý có thể được chia thành nhiều phân khu để điều khiển việc sử dụng phương tiện và cho phép nhiều hệ thống tập tin (có thể khác nhau) trên đĩa. Các hệ thống tập tin này được gắn vào kiến trúc hệ thống tập tin luận lý để làm cho chúng sẵn dùng. Các hệ thống tập tin thường được cài đặt trong một kiến trúc phân tầng hay module. Những cấp thấp hơn giải quyết các thuộc tính vật lý của các thiết bị lưu trữ. Cấp cao hơn giải quyết các tên tập tin biểu tượng và các thuộc tính luận lý của tập tin. Các cấp trung gian ánh xạ các khái niệm tập tin luận lý thành các thuộc tính thiết bị vật lý.

Mỗi kiểu hệ thống tập tin có các cấu trúc và giải thuật khác nhau. Một tầng VFS cho phép các tầng cao hơn giải quyết mỗi kiểu hệ thống tập tin khác nhau trong cùng một cách. Ngay cả các hệ thống tập tin ở xa có thể được tích hợp vào cấu trúc thư mục của hệ thống và được hoạt động trên các lời gọi hệ thống chuẩn bằng giao diện VFS.

Những tập tin khác nhau có thể được cấp phát không gian trên đĩa trong 3 cách: kề, liên kết hay chỉ mục. Cấp phát kề có thể gặp phải sự phân mãnh ngoài. Truy xuất trực tiếp là kém hiệu quả với cấp phát liên kết. Cấp phát chỉ mục yêu cầu chi phí đáng kể cho khối chỉ mục của nó. Các giải thuật này có thể tối ưu trong nhiều cách. Không gian kề có thể lớn lên thông qua đoạn mở rộng để tăng khả năng linh hoạt và giảm phân mãnh ngoài. Cấp phát chỉ mục có thể được thực hiện trong việc nhóm nhiều khối để tăng thông lượng và giảm số lượng các mục từ chỉ mục được yêu cầu. Lập chỉ mục trong các nhóm là tương tự như cấp phát kề với các đoạn mở rộng.

Các phương pháp cấp phát không gian trống cũng ảnh hưởng tới tính hiệu quả của không gian đĩa, năng lực hệ thống tập tin và khả năng tin cậy của thiết bị lưu trữ phụ. Các phương pháp được dùng gồm các vector bit và các danh sách liên kết. Các tối ưu gồm nhóm, đếm và FAT, mà đặt danh sách liên kết trong một vùng kề.