CHƯƠNG 4: QUẢN LÝ TỆP TIN

4.0. Quan niệm về quản lý tệp tin (*Files Manager*)

Ở việc bẻ gãy một tiến trình hay sau khi hoàn thành một tiến trình, một câu hỏi được đặt ra là: Bạn lưu trữ dữ liệu của bạn như thế nào, để sau này, bạn có thể làm việc trở lại với cái bạn đã có? Tính chất này của dữ liệu gọi là cố định dữ liệu, người ta đạt được khi dữ liệu được viết vào bộ nhớ quảng đại trước khi kết thúc chương trình. Tuy nhiên, chúng ta vẫn còn một vấn đề: Ở phần lớn bộ nhớ quảng đại, danh sách các tệp tin quá dài, ấy nhưng người ta muốn thời gian truy cập phải ngắn. Do đó, trường hợp này chỉ có thể dẫn tới một cách thức tổ chức tệp tin để quản lý các tệp dữ liệu trong hệ thống máy tính.

4.1. Hệ thống tệp tin (*Files- System*)

Để phục vụ việc quản lý tệp tin, đầu tiên, chúng ta viết tất cả các tệp tin vào trong một danh sách và cần lưu ý: chúng ta cần phải biết một cái gì đó về tệp tin, thí dụ ngày tháng năm tạo lập, dung lượng, vị trí của bộ nhớ quảng đại, luật truy cập…

Một bảng như thế thì không khác mấy một ngân hàng dữ liệu. Nếu chúng ta dẫn tới mỗi thuộc tính dữ liệu và kể cả các tác vụ (như các thủ tục, các phương pháp tiến hành…), mà nó được xử lý, do đó, chúng ta nhận được một ngân hàng dữ liệu hướng đối tượng. Ngoài ra, nếu chúng ta có các đối tượng âm thanh và hình ảnh, mà chúng được một chương trình sinh ra hay được sử dụng, lúc đó, chúng ta cần dùng một ngân hàng dữ liệu đa phương dụng (*multimedia-databank*), trong đó, thông tin được chứa đựng để làm đồng bộ hai phương dụng. Một cách khái quát, khi không có đối tượng nào thì việc điều hành khi lưu trữ dữ liệu cố định được phát triển; còn đối với các ngân hàng dữ liệu lớn thì hệ thống điều hành cơ sở dữ liệu hướng đối tượng được phát triển.

Chúng ta có thể sử dụng tất cả các cơ cấu có trên danh mục các tệp tin của bộ nhớ quảng đại, các cơ chế này được tìm thấy để tổ chức một cách hiệu quả các ngân hang dữ liệu hình cây; với cấu trúc này, người ta có thể tìm kiếm rất nhanh các tệp tin với một dấu hiệu xác định (ngày tháng tạo lập tệp tin, tên tác giả…).

Tuy nhiên, chúng ta phải quan tâm tới vấn đề, mà chúng liên quan đến ngân hàng dữ liệu. Một vấn đề cơ bản là sự cố định dữ liệu: Nếu chúng ta tạo lập, loại bỏ hay thay đổi các dữ liệu của một tệp tin trên bộ nhớ quảng đại, do đó, điều đó phải được giải thích ở trong bảng danh mục các dữ liệu. Nếu tác vụ này bị bẽ gãy, thí dụ vì một lý do nào đó, tiến trình viết bị bẻ gãy (thí dụ khi mất nguồn điện), do đó, dữ liệu ở trong bảng danh mục thì không còn thích hợp nữa. Ở các tác vụ tiếp theo, người ta đã làm quen một cơ cấu quan trọng ở trong mục 2.3.2, đó là việc kết hợp nhiều tác vụ thành một hoạt động nhân tử. Để phát hiện những sai sót của hệ

thống tệp tin, tất cả các tác vụ ở trên danh mục tệp tin phải được thực thi với tư cách là những hoạt động nhân tử.

Việc ký hiệu tên tệp tin với một cái tên có ghép thêm những chữ số là một sự trợ giúp quan trọng đối với người sử dụng. Trong bảng danh mục các tệp tin ở hình 4.1 có thể tồn tại 2 tệp tin với các tên giống nhau Teptin1.dat và Teptin2. dat; chúng chỉ khác nhau bởi một chữ số, do đó, đối với việc quản lý tệp tin, chúng khác nhau rõ ràng. Hình 4.2 chỉ ra sơ đồ tổ chức quản lý tệp tin. Ở đây, mỗi tệp tin có thể đứng độc lập (Tệp tin X), hoặc có thể thuộc một nhóm các tệp tin (Tệp tin 1, Tệp tin m…)

Hình 4.2----------

Các nhóm nhỏ có thể gộp lại thành nhóm lớn hơn tuỳ thuộc vào người sử dụng, sao cho nó trở thành cấu trúc cây để sử dụng. Kiểu tổ chức tệp tin này thì rất phổ biến. Các đại diện nhóm được biểu thì là một thư mục.

Một kiểu tổ chức quản lý các tệp tin như thế được dùng để tổ chức ngân hàng dữ liệu, mà trong đó, tất cả các tệp tin được mô tả với các tính chất và các hiển thị nội dung. Sự lưu ý về các tệp tin thì tồn tại ở trong một thư mục dữ liệu. Trong hầu hết các hệ điều hành, những hệ thống tệp tin có sơ đồ đơn giản được dùng.

Hình 4.3----------

Một kiểu sơ đồ khác cũng thường được dùng, đó là kiểu sơ đồ tổ chức tệp tin kiểu mạng, hãy xem một thí dụ trong hình 4.3 trên đây. Ở đó chỉ ra 2 thư mục Ban a và Ban b có thể nối ngang với tệp tin Thongbao. dot của công ty, vì tệp tin này luôn luôn tồn tại.

4.2. Tên tệp tin (*Filesname*)

Dữ liệu của các tiến trình không chỉ phải được bảo vệ bền vững ở trên bộ nhớ quảng đại, mà đặc biệt phải tạo khả năng truy cập qua các tiến trình khác nhau, do đó, mỗi tệp tin được biểu thị một từ khoá rõ ràng. Vì cách tổ chức hệ thống quản lý tệp tin hầu hết do con người thực hiện, cũng giống như việc lập trình các tiến trình để truy cập các tệp tin, tên của tệp tin có thể giống nhau, chỉ khác, mỗi tệp tin có thêm một con số, coi như biểu thị một khoá tượng trưng…, thí dụ Vanban1.doc, Vanban2.doc…Cách làm này có lợi cho việc quản lý nội bộ các tệp tin (vị trí, dung lượng) một cách dễ dàng. Tệp tin có thể được thi hành hay được đặt tại một chỗ nào đó ở tron bộ nhớ mà tệp tin vẫn không bị thay đổi.

4.2.1 Kiểu tệp tin và tạo tên tệp tin

Một cách truyền thống, nói chung tên tệp tin gồm hai phần, chúng được ngăn cách nhau bằng một dấu chấm: phía trái dấu chấm là tên riêng, còn bên phải là

phần mở rộng (*extension*). Phần mở rộng này cho thấy một sự trợ giúp về mục đích sử dụng của tệp tin. Thí dụ tên một tệp tin Vanban.txt, ý nói: tệp tin Vanban có kiểu mở rộng txt là một loại tệp tin text được cấu thành bởi các ký tự theo chuẩn ASCII. Có những quy ước khác nhau cho phần mở rộng, thí dụ như sau:

.dat cho dữ liệu hoặc việc tạo kiểu dạng phụ thuộc vào chương trình tạo lập;
.doc cho các văn bản text hoặc việc tạo dạng theo kiểu soạn thảo text;
.pas cho mã nguồn của chương trình viết bằng ngôn ngữ PASCAL;
.c cho mã nguồn của chương trình viết bằng ngôn ngữ C;
.h các tệp tin khai báo cho các chương trình viết bằng ngôn ngữ C;
.ps các tệp tin ngôn ngữ lệnh đồ hoạ cho máy in Laser;
.tar một hệ thống tệp tin lưu trữ trong một tệp tin;
.html tệp tin văn bản ASCII cho hệ thống text đặc biệt của trang WEB;
.Z ; .zip; .gz cho các tập tin nén lại;
.jog; .gi; .ti; .bmp cho các tệp tin về ảnh.

Người ta ghi nhớ rằng, phần mở rộng của tệp tin chứa đựng chỉ một ít hoặc chỉ thỉnh thoảng những ký tự cổ xưa. Điều đó được lý giải rằng, đối với một hệ thống tệp tin thông thường cho phép vài ký tự làm phần mở rộng và mặc khác, nó thuận tiện cho người sử dụng không phải viết quá nhiều.

Có rất nhiều kiểu phần mở rộng, mà chúng được tạo bởi hệ thống người sử dụng và chúng cũng phục vụ cho việc phân nhóm các tệp tin. Vì tên tệp tin nói chung là tuỳ ý và có thể được sử dụng thay đổi, do đó, những người lập trình đều đã chú ý tới cái đó, nhằm làm cho tệp tin chỉ rõ công dụng của chúng. Ở đầu tệp tin, một hay nhiều kiểu chữ số ảo (*magic number*) được viết, mà chúng cho phép một đặc tính chính xác của nội dung tệp tin.

Thí dụ về tên tệp tin ở trong Unix:

Tên tệp tin ở trong hệ điều hành Unix, có thể có tới 255 ký tự. Với Unix version V, số ký tự của tên tệp tin có ít hơn 14 ký tự cho phần chính và 10 ký tự cho phần mở rộng. Các tệp tin thực thi ở trong Unix có cấu trúc như sau:

```
TYPE File  Header = RECORD
a_magic:      LONG CARDINAL        {số ảo}
a_txt:        CARDINAL             {độ lớn của Codesegment}
a_data:       CARDINAL             {độ lớn Segment của các dữ liệu khởi xướng}
a_bss:        CARDINAL             {độ lớn Segment của các dữ liệu không khởi xướng}
a_syms:       CARDINAL             {độ lớn bảng biểu trưng Symboltable}
a_entry:      CARDINAL             {sự bắt đầu của chương trình}
…
END
```

Tuy nhiên, một số ảo đứng đầu tên tệp tin; chúng không chỉ dẫn tới sự giải thích về điều đó (cho thấy tệp tin có thể thực thi), mà còn, bằng phương pháp nào tệp tin được thực hiện:

a_magic= 407B {tạo dạng tất cả: mã không được bảo vệ cũng không được sử dụng cho 1 tiến trình khác}
a_magic= 410B {Text-Segment được bảo vệ Data- Segment được đặt ở giới hạn trang cuối cùng (4kB) ở trong bộ nhớ}
a_magic= 413B {Text- Segment bắt đầu ở giới hạn 4kB của tệp tin; Text và Data-Segment là bội số của 4kB}

Để phân biệt các loại tệp tin khác nhau, đó là sự đánh giá về khả năng có thể thực thi khi khởi động tệp tin. Loại phân biệt này thì rất đặc biệt. Sau đây chúng ta khảo sát vài thí dụ về kiểu phân biệt tệp tin này.

Thí dụ về kiểu và tên tệp tin:

Trong Unix hay trong Windows NT, kiểu cách một tệp tin được biểu thị nhờ một cái tên. Thí dụ tên tệp tin Script.ps.gz có ý nghĩa: đó là một tệp tin nguyên bản (Script), mà ấn bản của nó đã được nén lại ở dạng tái bút (postscript) với chương trình nén zip. Để đọc được tệp tin này, đầu tiên, người ta phải gọi chương trình gunzip, và sau đó, tệp tin chuyển tới cho máy in Postscript. Kết quả chỉ dẫn là sự cải biên: nó không chỉ chứa đựng trong tệp tin, mà còn chứa đựng ở trong kiểu tệp tin được nén. Đối với tệp tin này, nếu kiểu tệp tin được chỉ dẫn trong thư mục, do đó, người ta có thể thực hiện tác vụ thứ hai. Việc tạo kiểu tệp tin phải được cắt nghĩa ở trong tệp tin hay nó phải là kiểu bởi sự xếp nhiều lần của các kiểu. Điều đó chỉ có thể nói gọn là: phải giải thích kiểu mới lập rõ ràng và phải đưa vào những hoạt động thích hợp cho các kiểu đó.

Thí dụ về kiểu tệp tin và các hoạt động chương trình ở trong Unix:

Ở trong hệ điều hành Unix có một giao diện đồ hoạ người sử dụng, một sự quản lý tệp tin bao hàm trong giao diện này. Điều đó cho thấy, các tệp tin chuyên dụng ứng với mỗi phần mở rộng sẽ dẫn tới những tác động khác nhau của chương trình, điều đó được chỉ ra như là một Menu (thực đơn) khi kích chuột phải. Khi kích đôi chuột trái ở tại tên tệp tin, thì hoạt động được dẫn ra trong Menu sẽ bắt đầu khởi động.

Thí dụ về kiểu tệp tin và hoạt động chương trình ở Windows NT:

Dưới *Windows NT* có trình soạn thảo Editor, trình này quản lý một ngân hàng dữ liệu riêng của tất cả các ứng dụng, bộ kích tạo hệ điều hành và các cấu hình hệ thống…Tất cả các chương trình ở trong hệ thống cũng như các phần mở rộng đều được khai báo ở đó. Khi kích đôi chuột trái tại trên một tệp tin ở trong trình quản lý tệp tin thì một chương trình được gọi.

Người ta lưu ý rằng, trong hai thí dụ vừa nêu, thông tin về kiểu tệp tin thường không được hệ điều hành quản lý, mà nó tồn tại trong các tệp tin đặc biệt hay được chứa đựng và được quản lý với các chương trình đặc biệt. Điều đó thì không có vấn đề gì. Kiểu tốt nhất cho việc quản lý các tệp tin là tổ chức tất cả tệp tin thành một ngân hàng dữ liệu tổng thể; trong đó, các tính chất bổ sung của các tệp tin được thực hiện nhờ các đặc tính bổ sung

Một vấn đề khác là việc làm sáng tỏ sự tham chiếu tệp tin, nếu các tệp tin ở trong hệ thống quản lý files thì cho phép tên tệp tin dài; nhưng khi ứng dụng (khi thực thi), tệp tin chỉ có thể được phép ngắn lại. Thí dụ có hai tên tệp tin MethodeForUser.txt và MethodeForEvery.txt cùng tồn tại ở trong hệ thống tệp tin. Khi hai tệp tin muốn thực thi, thì 8 ký tự đầu tiên được lưu ý. Trong trường hợp này, 8 ký tự đầu tiên của hai tệp tin giống nhau. Do vậy, ở đây, một phương pháp được tìm thấy cho phép sắp xếp các tệp tin được rõ ràng hơn.

Thí dụ về sự chuyển đổi tên tệp tin trong Windows NT:

Vì hệ điều hành *Windows NT* có thể quản lý cả các hệ thống tệp tin DOS bên cạnh hệ thống tệp tin NTFS (windows NT File System). Cho nên, ở đây vấn đề đặt ra là phải chuyển đổi các tệp tin có tên dài trong hệ thống NTFS thành tên tệp tin rõ ràng trong hệ thống MS-DOS. Điều này đạt được bằng các giải thuật sau đây:

(1) Tất cả các ký tự của tên tệp tin không hợp lý ở trong MS-DOS được chuyển đổi thành tên tệp tin trong hệ thống NTFS bằng cách cắt bỏ: các ký tự trống, 16 ký tự của 16 Bit- Unicode, các điểm ở đầu, ở cuối và trong khoảng tên tệp tin.

(2) Tất cả 6 ký tự đầu tiên của chuỗi ký tự đứng trước dấu chấm phân đoạn phần mở rộng được cắt bỏ; và sau đó, ký hiệu ~1 được thay vào trước điểm phân đoạn phần mở rộng. Chuỗi ký tự sau điểm phân đoạn phần mở rộng chỉ để lại 3 ký tự, các ký tự ở cuối phần này cắt bỏ và chuỗi ký tự còn lại chuyển thành chuỗi các chữ cái lớn.

(3) Nếu có một tệp tin, mà nó chỉ ra sự giống nhau với tệp tin khác, do đó, thay vì ký hiệu ~1, ký hiệu ~2 được thay vào chỗ đó. Nếu tệp tin này đã tồn tại, thì thay bằng ký hiệu ~3 hay ~4.. cho đến khi không tồn tại sự giống nhau.

4.2.2. Tên đường dẫn

Qua sơ đồ sắp xếp các tệp tin, việc biểu diễn tên tệp tin rõ ràng tạo ra khả năng để sắp xếp các nút có dạng hình cây, nhờ vậy, một đường dẫn đi tới một tệp tin được thực hiện thuận tiện. Tên đường dẫn của một tệp tin được tạo lập từ một chuỗi các tên nút và chứa đựng ký tự tách chia giữa các tên nút. Đó là một dấu hiệu đặc biệt, nó là biểu tượng tiêu biểu cho mỗi hệ điều hành. Thí dụ ở trong hình 4.3, tên tệp tin Vanban1.txt có đường dẫn trong hệ điều hành Unix:

Institut5/Rudi/Vanban1.txt

còn ở trong hệ điều hành Windows NT có dạng:

Institut5/Rudi/Vanban1.txt

Về điều đó, do có sự kết nối ngang, nên có nhiều tên đường dẫn cho một tệp tin, thí dụ đối với tệp tin Briefvorlage.dot có các đường dẫn:

Institut5/Rudi/briefvorlage.dot và Congty/Dieuhanh/briefvorlage.dot.

Những nút ở trên của hệ thống cây thư mục tệp tin được gọi là gốc, nó được biểu thị bởi một biểu trưng đặc biệt: dấu xiên trái(/) ở trong Unix, dấu xiên phải (\) ở trong Windows NT. Khi thông dịch tên tệp tin, nó được thực hiện từng bước từ

dưới lên trên theo hướng gốc cây thư mục tệp tin. Thí dụ trong hình 4.3, tệp tin Brief.doc ở thư mục Rudi mà cả ở thư mục Hans, tên đường dẫn của nó được biểu diễn mỗi dấu đại diện cho một thư mục nào đó:

../../Rudi/Brief1.doc

Tên đường dẫn tương đối bắt đầu từ một vị trí nào đó của cây tới tệp tin. Tên đường dẫn tuyệt đối được biểu thị từ gốc đến ngọn là tên tệp tin. Thí dụ ở trong hình 4.4 chỉ ra một đường dẫn tương đối ../Daten/Dat1.a với tệp tin Dat1.a của chương trình Prog.

Hình 4.4 ----------

4.2.3. Thí dụ về cây thư mục ở Unix

Trong Unix, sơ đồ một hệ thống tệp tin thông dụng được chỉ ra trong hình 4.2, nhờ đó không gian tên tệp tin được tạo lập. Ngoài ra ở cấu trúc cây này người ta có thể nối ngang qua giữa các tệp tin bởi lệnh "ln".

Đối với đặc điểm của sơ đồ tên, người ta phải phân biệt, các thư mục đã được thu xếp với các tên tệp tin trên ổ đĩa hay chưa (?). Nếu chúng đã được thu xếp, do đó, một sự nối ngang trực tiếp là có thể. Trường hợp này gọi là kết nối vật lý hay kết nối cứng. Nếu ngược lại, chúng được thu xếp khác nhau trên ổ đĩa, do đó, người ta có thể thiết đặt một sự nối ngang logic và dẫn ra tên đường dẫn của chúng. Trường hợp này gọi là kết nối tượng trưng (*symbol-link*).

Sự khác nhau ở các kết nối ngang nói ở trên là ở chỗ, việc xoá bỏ tên tệp tin hay thư mục khi có nhiều tiến trình hay nhiều người sử dụng truy cập chúng. Nếu chúng ta lưu ý tới hệ thống cây thư mục ở trong hình 4.5, chúng ta thấy rằng không có sự phân biệt giữa tên tệp tin và tên thư mục.

Hình 4.5----------

Nếu các thư mục Gruppe1 và Gruppe2 được ghép ở các ổ đĩa khác nhau, do đó, các nối ngang Rudi/Datei2 và Hans/Datei2 thì khác biệt với các nối ngang Gruppe1/Hans/Datei3 và Gruppe2/Datei3: ở đây cặp thứ nhất là nối vật lý, còn cặp thứ hai là nối logic.

Nếu bây giờ ở Unix một tệp tin được xoá bỏ, do đó, nó sẽ tồn tại tối thiểu cho đến khi có một sự kết nối vật lý tồn tại. Ở trong thí dụ đã nêu, nếu tệp tin có đường dẫn Rudi/Datei2 được xoá bỏ, do đó, tệp tin chỉ còn lại dưới cái tên có đường dẫn Hans/Datei2 và có thể sử dụng được. Ngược lại nếu tệp tin có đường dẫn Hans/Datei3 xoá bỏ; do đó, tệp tin tương ứng cũng bị xoá bỏ; sự tham chiếu tên (sysbol-link) ở trong thư mục Gruppe2 vẫn còn tồn tại, nhưng khi truy cập tệp tin thì ngay lập tức xuất hiện lỗi khi tham chiếu tên tệp tin này, tức là một tệp tin với cái tên này không còn nữa.

Điều quyết định trong hệ điều hành Unix là để một tệp tin được xoá hoàn toàn, thì phụ thuộc vào bộ đếm để đếm số tham chiếu kết nối vật lý. Tuy nhiên, cơ chế

này của hệ điều hành Unix là một phương tiện đơn giản ở trong phạm vi đa tiến trình, nhưng cũng có thể dẫn tới lỗi. Chúng ta nhận thấy rằng, chúng ta đang ở tại thư mục Rudi của hình 4.5 và thực hiện kết nối vật lý trên thư mục Gruppe1. Vì việc xóa bỏ một thư mục chưa trống thì cấm (để tránh lỗi), do đó, chúng ta không thể xoá được thư mục Gruppe2. Ngược lại, ở tại thư mục Gruppe1: ở đây có một kết nối thứ hai của thư mục Rudi, cho nên cho phép xoá bỏ tên tệp tin được. Điều đó dẫn tới tình huống, rằng bây giờ thư mục Gruppe1 với thư mục con và các tệp tin ở phía dưới còn tồn tại, tuy nhiên những thứ đó không còn dùng được nữa, vì chúng là những thư mục trống, và do đó cũng không thể xoá bỏ được. Điều đó thì cũng khong thể tránh được: tuy kết nối ngang là có thể, nhưng việc kiểm tra các vòng nối và việc tách chia các sơ đồ tệp tin tại mỗi tác vụ xoá là không nên làm. Từ lý do này, ở ấn bản mới của Unix về cây thư mục thì chỉ còn được dùng kiểu kết nối logic.

Đối với việc tạo lập hay xoá bỏ tệp tin, ở trong Unix có một có chế thông dụng: Nếu một tiến trình tạo lập một tệp tin và sau đó, nhiều tiến trình mở tệp tin này, do đó, tệp tin không bị biến mất khi xoá, nhờ người tạo lập. Và nó sẽ tồn tại cho tới khi tiến trình cuối cùng gọi thủ tục close() và như vậy, bộ đếm sự tham chiếu trở về 0.

Hệ thống tệp tin hình cây tổng quát được phân chia thành các sơ đồ nhánh khác nhau, mà các tệp tin của chúng tồn tại trên các ổ đĩa khác nhau. Khi khởi động hệ thống (bootstrap), hệ thống tệp tin của các ổ đĩa khác nhau vào một nút gốc (root) bằng một hàm gọi hệ thống mount(), mà ở đó, nút gốc của mỗi hệ thống tệp tin còn được phản ảnh bởi tên của một thư mục (*mount point direction*). Hình 4.6 chỉ ra một cây thư mục tệp tin như vậy.

Hình 4.6----------

4.2.4. Cây thư mục ở Windows NT

Cây thư mục ở trong hệ điều hành Windows NT mô tả toàn bộ các đối tượng toàn cục; một cách độc lập với cái đó, khi chúng là các tệp tin thuần tuý hay những đối tượng khác như các kênh trao đổi thông tin (*communication-canal hay named pips*), các bộ nhớ chia xẻ (*shared memory*), các cờ hiệu (*semaphore*), các biến cố hay các tiến trình. Các đối tượng với tên gọi toàn cục được trao cho các cơ chế bảo vệ tương ứng khi truy cập.

Hệ thống cây thư mục bắt đầu với nút gốc bởi dấu xiên phải (\), ở đó tồn tại các đối tượng tệp tin hay các đối tượng thư mục. Các thư mục đối tượng là các đối tượng chứa đựng tên của đối tượng với những thuộc tính (biến số, hằng số…), với các phương pháp (tạo lập, mở hay đọc lướt các thư mục). Hệ thống này không chỉ bao gồm trạng thái nhân, mà cả trạng thái người sử dụng, nghĩa là các thư mục không chỉ có thể đặt vào nhân hệ điều hành Windows NT, mà còn có thể đặt vào cả các hệ thống khác như OS/2, POSIX…Đối với mỗi loại đối tượng (tệp tin, tiến

trình…) có những ấn bản chuyên dụng cho 3 phương pháp sử dụng các đơn thể nhân khác nhau (điều hành I/O, điều hành tiến trình…).

Tuy nhiên, để tạo lập thư mục toàn cục từ các thư mục đối tượng hay việc thiết đặt các đường dẫn kết nối ngang (như trong Unix) đều có thể thực hiện một cách thuận lợi ở trong Windows NT.

Ở cây thư mục của việc điều hành đối tượng thì tồn tại nhiều đối tượng khác nhau (xem hình 4.7). Chẳng hạn đối tượng "A:" là một kết nối ngang logic với đối tượng hệ thống tệp tin "Floppy". Với mỗi đối tượng thì một phương pháp dò tìm được chuyên môn hoá. Nếu trình soạn thảo Editor muốn mở tệp tin có đường dẫn A:\Texte\bs_files.doc, do đó, tại cửa sổ điều hành đối tượng, nó hỏi đối tượng này. Bằng phương pháp dò tìm, trình điều hành đối tượng tìm kiếm cây thư mục của nó: đối tượng cần tới ở chỗ nào (?). Nếu nó đi tới đối tượng "A:", thì khi đó, phương pháp kết nối logic được áp dụng.

Hình 4.7----------

Chuỗi ký tự "A:" được thay thế bởi chuỗi ký tự "\Device\Floppy0" và sau đó, được chuyển cho trình điều hành đối tượng. Trình này sẽ xử lý đường dẫn cho tới khi gặp đối tượng tệp tin "Floppy0". Phương pháp dò tìm này làm việc trên cây thư mục của trình điều hành hệ thống các tệp tin, nó dò tìm đường dẫn với các thủ tục đặc biệt, cho tới khi một đối tượng tệp tin với bs_files.doc có thể được đưa trở lại.

Với cơ chế này, ở trong Windows NT có thể tích hợp một cách thống nhất các hệ thống tệp tin khác nhau như hệ thống FAT (file allocation table) của MS-DOS, hệ thống tệp tin hiệu suất cao của OS/2 và hệ thống tệp tin NT của Windows.

Đối với việc xoá một tệp tin cũng như xoá một đối tượng được nhiều tiến trình sử dụng, thì các cơ chế được dẫn tới tương tự như trong Unix. Vì lý do thực thi, có hai bộ đếm tham chiếu được dẫn tới: một bộ để đếm số lượng các chức danh đối tượng, khi đó chúng được đón nhận ở các tiến trình người sử dụng; và một bộ nữa để đếm số lượng que chỉ thị, khi đó chúng được đưa vào hệ điều hành (đáng lẽ chúng được trao cho bởi các đối tượng khi truy cập nhanh). Nếu bộ đếm tham chiếu của người sử dụng mà tham chiếu tới tên tệp tin bị giảm xuống tới 0, thì do đó, đối tượng sẽ bị xóa ở trong cây thư mục; và cũng vì thế, không có một tiến trình nào có thể truy cập được. Tuy nhiên, nó vẫn còn ở lại trong bộ nhớ cho tới khi bộ đếm các tham chiếu của hệ điều hành giảm xuống 0. Sau đó bộ nhớ được giải phóng và được sự dụng trở lại. Với cơ chế này đã tránh được: một đối tượng được nhiều tiến trình xử lý bị xoá một cách nhầm lẫn trước khi các tiến trình khác thực thi xong.

Một bộ đếm tham chiếu đôi cũng tránh được: thí dụ một đối tượng tiến trình được sinh ra bởi một tiến trình và sau đó, với sự kết thúc của tiến trình này, nó đi tới kết thúc; tuy nhiên, vẫn còn một tiến trình có quan hệ với đối tượng. Sự tham chiếu của hệ điều hành tới tiến trình thứ hai đảm bảo sự tồn tại của đối tượng tệp tin cho tới khi quan hệ nói trên không còn nữa.

4.3. Thuộc tính tệp tin và cơ chế bảo vệ

Trong một thư mục có chứa tên một tệp tin, thì hầu hết các thông tin về tệp tin được bảo vệ. Bên cạnh độ lớn tệp tin (tính bằng Byte hay bằng các khối trang), ngày tháng tạo lập và ngày tháng điều chỉnh, còn có các thuộc tính khác (còn gọi là các cờ hiệu khác) như ẩn khuất (*Hidden*), hệ thống.

Một dạng đặc biệt của các thông tin trạng thái là các thông tin bảo vệ như luật truy cập của con người và của các chương trình đối với tệp tin. Tương tự như đối với các cơ chế bảo vệ bộ nhớ, người ta đã xem xét để loại bỏ các chức năng lỗi của chương trình như các lỗi khi truy xuất chương trình của người sử dụng nhờ những biện pháp có mục đích: Những biện pháp do Uỷ ban POSI-6 đề nghị có nội dùng như sau:

+ Phải đảm bảo nguyên tắc thu gọn đặc quyền ít nhất (*least privilege*) đối với việc thực hiện một nhiệm vụ.

+ Phải bổ sung việc điều khiển khi truy cập qua các thông báo rời rạc (*discretionary access control*). Việc truy xuất cưỡng bức thì độc lập với người tạo lập. Việc truy xuất đối tượng chỉ xảy ra bởi các tiến trình với các đặc quyền lớn hơn.

+ Phải lưu ý các ghi chép về trạng thái của đối tượng nhằm có thể phát hiện các nguyên nhân và các người làm việc trong hệ thống khi sử dụng sai trái.

Ngược lại, hệ điều hành chỉ thực thi rất giới hạn các yêu cầu kể trên.

4.3.1. Các đặc quyền truy cập ở Unix

Ở hệ điều hành Unix có các biến trạng thái (còn gọi là các cờ) khác nhau: đọc (ký hiệu r), viết (w), thực thi (x). Tất cả các tệp tin và các thư mục đều có một trạng thái như nhau.

Các ký hiệu vừa nói có ý nghĩa như sau:

r cho phép đọc danh sách tệp tin;

w cho phép thay đổi vị trí tệp tin trong danh sách;

x cho phép thực thi hay tìm kiếm tệp tin trong danh sách.

Về quyền truy cập, hệ điều hành Unix phân biệt 3 hạng: chủ nhân hệ thống (*owner*), thành viên một nhóm (*user group member*) và mọi người khác (*other*).

Chỉ dẫn về các quyền truy cập

Lệnh ls là để xem các tệp tin và các quyền truy cập chúng. Lệnh ls-al để xem các tệp tin của một thư mục, thí dụ với lệnh này, ta nhận được đoạn thông báo:

```
drwxr-xr-x      brause   512     Ap23   15:55    .
drwxr-xr-x      obene    512     May17  17:53    ..
- rw- r--r –    brause   44181   Ap23   15:56    data1.txt
```

Ở cột đầu tiên là các quyền truy cập của các tệp tin. Trong đó: d là thư mục (directory), còn 9 ký tự tiếp theo được phân làm 3 cho các người sử dụng owner (chủ nhân), grouper (nhóm trưởng), other (người khác) với r: đọc, w: viết, x: tìm

kiểm và dấu gạch ngang (-) để chỉ một quyền truy cập không được chọn. Những cột còn lại của bảng thông báo này là tên người sử dụng, dung lượng (Byte), ngày tháng tạo lập, thời gian tạo lập và cuối cùng là tên tệp tin. Ở dòng đầu tiên của cột tên tệp tin, có một dấu chấm (.) để chỉ tên tệp tin ở một thư mục hiện hành của người sử dụng brause có dung lượng 512Byte, được tạo lập ngày 23 tháng 4 lúc 15giờ 55 phút. Ở dòng thứ 2 của cột này có dấu hai chấm (..) cũng giải thích tương tự.

Ở hệ điều hành Unix, để thực hiện một chương trình, có thể kết nối các quyền truy cập của riêng mình trong chương trình với hàm *userId* cho người thực hiện và *groupId* cho người quản lý nhóm. Do đó, các chức năng của chương trình hệ thống có thể được mọi người sử dụng thực hiện, nếu thiết đặt các trạng thái *userId và set groupId*. Còn nếu tại một thư mục, trạng thái sticky bit được thiết đặt, do đó, một người sử dụng bình thường không thể xoá hay gọi các tệp tin của người sử dụng khác trong thư mục này.

Chỉ những ấn bản đặc biệt của hệ điều hành Unix mới có các danh sách điều khiển truy xuất với nhiều cơ chế khác nhau. Cũng với lý do này, sự hoạt động mạnh mẽ của nhóm O/Open đã thống nhất và tạo ra được nhiều tiêu chuẩn bảo vệ hệ thống Unix.

4.3.2. Quyền truy cập ở trong Windows NT

Hệ thống tệp tin ở trong hệ điều hành Windows NT được quản lý bởi một cơ chế định hướng đối tượng. Quyền truy cập có giá trị đối với mỗi đối tượng trong cây thư mục của Windows NT (như các tiến trình, các cờ hiệu, khoảng nhớ…). Độc lập với các quyền truy cập này còn có những tính chất đặc biệt đối với các đối tượng tệp tin:

Các thuộc tính:
+ Tên tệp tin
+ Kiểu thiết bị mà trên đó tệp tin tồn tại,
+ Byte offset: là tình trạng hiện hành của tệp tin,
+ Shared mode: là trạng thái (đọc/viết/xoá) của tệp tin trong khi sử dùng,
+ Open mode: là kiểu và phương pháp tác vụ của tệp tin (đồng bộ/ không đồng bộ, có / không có bộ đệm Cache, truy cập tuần tự/ bất kỳ…)
+ file disposition: biểu thị tệp tin bền vững hay tuỳ ý.

Các phương pháp:
Các hàm CreateFile(), OpenFile(), ReadFile(), WriteFile(), CloseFile(), dùng để đọc chọn hay thay thế các thông tin tệp tin, các thuộc tính mở rộng, các độ lớn tệp tin (Byte), các thông tin về thiết bị hay để đọc chọn một thư mục…

Mỗi đối tượng tệp tin là một bản sao của thông tin điều khiển của một tệp tin; nó cũng có thể dẫn tới nhiều đối tượng, mà chúng tham chiếu chính tệp tin này.

Do đó, có những thông tin toàn cục được lưu trữ trong tệp tin (không phải ở trong đối tượng tệp tin) và cũng có thể được thay đổi ở đó.

Mỗi tệp tin chứa đựng những thuộc tính tệp tin khác nhau; khi thực hiện chúng là những dòng dữ liệu biến thiên theo độ lớn tệp tin. Trước hết, chúng bao gồm những thông tin sau đây:

Các thông tin chuẩn:

+ Ngày tháng và thời gian tạo lập, ngày tháng và thời gian truy xuất gần đây và lần thay thế cuối cùng;

+ Dung lượng tệp tin hiện hành;

+ Thuộc tính logic của tệp tin biểu thị bởi giá trị Yes/No. Thí dụ: tệp tin hệ thống, tệp tin ẩn, tệp tin lưu trữ, tệp tin điều khiển, tệp tin chỉ đọc, tệp tin nén…

Tên tệp tin:

Trong cây thư mục với kết nối vật lý, tên tệp tin có thể dài; ở các kết nối khác của MS-DOS, tệp tin có tên ngắn.

Các dữ liệu bảo vệ:

Chúng được chứa đựng trong danh sách điều khiển việc truy xuất đối với mọi chủ nhân của tệp tin.

Nội dung tệp tin:

Thuộc tính này chứa đựng các dữ liệu riêng lẻ; tại các thư mục, một cấu trúc chỉ số được lưu trữ cùng với các tệp tin.

Điều được quan tâm là các dữ liệu thực chất được chứa đựng một trong các thuộc tính nêu ở trên. Vì người sử dụng có thể tạo ra các thuộc tính, cho nên các dòng dữ liệu cũng được bổ sung liên tục. Khác với tên tệp tin chính, thí dụ *MyFile.dat*, các dữ liệu phụ của người lập trình được tham chiếu với một cái tên bổ sung; tên này được tách làm hai phần ngăn cách nhau bởi dấu hai chấm, thí dụ *MyFile.dat:MyCommentar*. Điều đó đã tạo nên nhiều thông tin bổ sung để treo vào một tệp tin (chẳng hạn tên của một chương trình xử lý hay ngữ cảnh khi xử lý lần cuối…) mà không hề thay đổi các dữ liệu chính. Do vậy, đối với mỗi dòng dữ liệu, một thông tin trạng thái được dẫn tới, ví như độ lớn hiện hành lớn nhất được cấp phát, các cờ hiệu đối với các phần của tệp tin…

Khi thay thế hay dẫn trở lại, các thuộc tính logic được trợ giúp nhờ phương pháp kết hợp. Thí dụ một tệp tin được nén lại một cách tự động, tức là tệp tin nhận được thuộc tính nén lại. Điều đó cũng có giá trị đối với toàn bộ cây thư mục.

Các cơ chế bảo vệ trong hệ điều hành Windows NT được tách chia một cách mạnh mẽ hơn trong Unix. Đối với mỗi tệp tin có một danh sách truy cập được chi tiết hoá; trong đó, các quyền truy cập của người sử dụng được thực hiện. Việc bổ sung thêm tên tiêu chuẩn như Adminitrator (quản lý), System, Creator (tạo lập), Quest (Khách), EveryOne (mọi người)…, người ta có thể chuyên môn hoá những

người sử dụng tiếp theo và các quyền của họ; nghĩa là họ được phép hay không được phép về một quyền truy cập nào đó.

Hệ thống bảo vệ này thì không giới hạn trên các tệp tin, nó được áp dụng một cách tổng hợp ở tất cả các đối tượng toàn cục ở đây thư mục trong Windows NT và được chỉ huy một cách thống nhất. Tiếp đó, một sự kiểm tra đối với việc truy cập tới tệp tin hay thư mục được trợ giúp.

4.4. Các chức năng tệp tin

Có rất nhiều kiểu tác vụ xảy ra trên các tệp tin. Trong mục này, chúng ta khảo sát vài kiểu đặc trưng cho tệp tin trên bộ nhớ quảng đại (các loại ổ đĩa cứng, ổ đĩa mềm, băng nhựa camara…).

4.4.1. Các chức năng chuẩn

Trong hầu hết các hệ điều hành đều có vài chức năng cơ bản, mà với chúng, các tệp tin có thể được đọc và viết. Sau đây là các chức năng cơ bản đó.

- *Tạo lập tệp tin (Create File)*:

Để thiết đặt một tệp tin, các thông số bao gồm một chuỗi các ký tự để chuyên môn hoá các kiểu truy cập (viết/ đọc, tuần tự/ tự chọn). Khi gọi một chức năng, người ta nhận được một sự tham chiếu tệp tin, mà với sự tham chiếu này, người ta có thể truy cập tới tất cả các chức năng tiếp theo của tệp tin. Một sự tham chiếu như thế (phân đoạn tệp tin, chức danh tệp tin) có thể là một con số (chỉ số của một trường nội bộ của việc điền vào tệp tin) hay một bộ chỉ thị ở một cấu trúc bên trong tệp tin.

- *Mở một tệp tin (OpenFiele)*:

Khi mở một tệp tin đang tồn tại, các cấu trúc dữ liệu khác nhau được khởi xướng, do đó, việc truy cập tiếp theo diễn ra nhanh hơn. Thuộc cái đó có việc kiểm tra các quyền truy cập cũng như cơ chế các bộ đệm, cơ chế các cấu trúc truy cập.

- *Đóng tệp tin (Close File)*:

Để đóng một tệp tin, cần phải mô tả các thông tin quản lý tệp tin tại bộ nhớ quảng đại và được phép nối tiếp việc sử dụng không gian các cấu trúc dữ liệu của việc quản lý tệp tin ở bộ nhớ chính.

Tương tự như vậy, khi trao đổi thông tin giữa các tiến trình ở mục 2.4.1, người ta có thể thay thế đôi lệnh OpenFile() và CloseFile(). Thay vì nối cứng vật lý sự trao đổi thông tin, người ta được phép có một sự trao đổi thông tin không cần kết nối vật lý: Tất cả mọi sự truy cập được thực hiện theo dãy tuần tự của chúng mà không cần dùng lệnh OpenFile(). Tuy nhiên, điều đó thì không thật chính thức đối với một cách tổ chức dữ liệu cục bộ: Thay vì truy cập tệp tin, người ta chỉ xếp đặt một lần (thí dụ việc kiểm tra quyền truy cập, việc xem xét các khối tệp tin…)

người ta phải thực hiện điều đó mỗi lần truy cập, do đó dẫn tới chi phí quản lý giảm thiểu đáng kể.

- *Đọc và viết tệp tin (Read File/ Write File)*:

Gọi hệ thống như là một tham số nhận được sự tham chiếu tệp tin và một dung lượng của bộ đệm để đọc hay viết. Nếu có một tệp tin mặc định và phù hợp với dung lượng bộ đệm, do đó, tệp tin được thực hiện.

- *Tìm kiếm tệp tin (Seek File)*:

Một tệp tin được tổ chức một cách tuần tự đơn giản và chiếm một vị trí, mà tại đó, người ta có thể viết hay đọc. Vị trí tệp tin này không chỉ được mô phỏng như những thông tin quản lý, mà còn có thể được thay thế bởi một chương trình ở trong các hệ thống các tệp tin như vậy. Bằng việt xử lý tuần tự còn gọi là truy cập tuần tự (*sequential access*), thí dụ khoảng từ tính của ổ đĩa mềm, người ta dẫn tới việc truy cập tuỳ chọn còn gọi là truy cập ngẫu nhiên (*random access*). Điều đó thì tiện lợi khi làm việc với ổ đĩa CD ROM.

Theo quan điểm hệ điều hành, các dịch vụ hệ thống nhận được sự trợ giúp một cách khác nhau. Điều đó được bao hàm trong câu hỏi, liệu việc đọc/ viết có được lưu trữ, hay liệu chương trình người sử dụng phải tự làm điều đó, khi phản ứng của hệ thống đặt lên một trạng thái không bình thường (?). Nếu không có dữ liệu nào có thể được đọc hay được viết, vì tại hàm Read() không có dữ liệu nào được sử dụng, hay tại hàm Write() không có ổ đĩa nào sẵn sàng làm việc. Gần như hàm Read() ở trong trường hợp này thì ngăn hãm tiến trình người sử dụng, còn hàm Write() thì không. Vì lý do này, các dữ liệu được lưu trữ; khi bộ đệm tràn hay ổ đĩa có khiếm khuyết, một thông báo lỗi được đưa trở lại. Một cách thuận lợi, khi đó có nhiều chức năng được bổ sung như: DeleteFile(), RemameFile(), CopyFile(), AppendFile(), FlushBuffer(),… Tất cả đều phụ thuộc vào hệ điều hành một cách mạnh mẽ.

4.4.2. Các chức năng truy cập ở trong Unix

Ở trong hệ điều hành Unix, các gọi hệ thống có dạng fd=creat(name, mode) và fd=open(name, mode) đã cung cấp cho ta một số bộ mô tả tệp tin (*fd: file discriptor*). Nó chính là sự tham chiếu đối với tất cả sự truy cập khác nhau, như:

+ read (fd, buffer, nbytes) đọc n Bytes ở trong bộ đệm;
+ write (fd, buffer, nbytes) viết n Bytes từ một bộ đệm;
+ close (fd) viết một tệp tin.

Bộ mô tả tệp tin này là chỉ số (*Index*) ở trong một trường của bảng mô tả tệp tin (*file descript table*). Số lượng các tệp tin và số lượng của sự điền vào (tức số lượng lớn nhất đối với fd) là cố định, nó được thông báo khi có sự dịch đổi của hệ điều hành; đồng thời, nó xác định độ lớn của cấu trúc quản lý đối với các tệp tin được lưu trữ ở trong cấu trúc người sử dụng của một tiến trình.

Bộ mô tả tệp tin này đóng vai trò quan trọng trong hệ điều hành Unix. Theo tiêu chuẩn, trước khi khởi xướng một tiến trình (chẳng hạn một chương trình), bộ mô tả tệp tin được ghi nhớ: fd =0 cho việc nhập số liệu (*stdin*), fd =1 cho việc xuất số liệu (*stdout*) và fd=2 cho việc xuất số liệu khi có lỗi (*stderr*). Ở trong Unix, người ta lợi dụng điều này để kết nối các chương trình với nhau, nhằm dẫn tới một chức năng nào đó. Ở mục 2.4, chúng ta đã làm quan với cấu trúc trao đổi thông tin pipe, thí dụ:

Programm1 | Programm2 | … | ProgrammN

Để thực hiện dòng lệnh này, tiến trình cha thiết đặt nhiều tiến trình con và tạo ra đối với mỗi một sự kết nối trao đổi thông tin một *pipe*. Sau đó, tiến trình cha khởi xướng các bộ mô tả tệp tin (*file descriptors*) cho mỗi tiến trình; khi đó, chúng sẽ dẫn tới: pipe vào với fd =1 ở tiến trình gởi đi, pipe ra với fd =0 ở tiến trình nhận. Hình 4.8 mô tả một hệ thống pipe ở trong Unix.

Hình 4.8----------

Việc sắp xếp các bộ mô tả tệp tin chỉ là sự quy ước; mỗi tiến trình có thể đạt được các sự sắp xếp các bộ mô tả tệp tin khác nhau qua việc đóng và mở có mục đích các tệp tin: Khi mở, bộ mô tả tệp tin trống với con số hệ thống nhỏ nhất được sử dụng đầu tiên.

4.4.3. Các chức năng truy cập tệp tin ở Windows NT

Các chức năng cơ bản đối với việc truy cập tệp tin được tóm lược như là những phương pháp về thiết laap các thuộc tính tệp tin. Các phương pháp hiệu nghiệm kể trên còn dẫn tới nhiều chức năng đối với các tệp tin như FlushBuffer() cũng như các tác vụ thư mục đặc trưng (đọc, viết, tìm kiếm thư mục…). Nói chung, một bản phác thảo được quan tâm đó là: Mọi tác vụ dù có cấu trúc như thế nào đều có thể thay đổi trên bộ nhớ quảng đại, thì chúng cũng được thực thi như là những biến đổi nhân tử (*atomar transaction*). Thuộc cái đó còn có một dịch vụ đặc biệt gọi là dịch vụ đăng ký tệp tin (*log file service: LFS*); mỗi lần truy cập tệp tin, dịch vụ này được gọi bởi nhân hệ điều hành và viết bản ghi đăng ký cho hệ thống tệp tin. Trước khi các tệp tin riêng lẻ được thay đổi trên bộ nhớ quảng đại (nhờ một biến cố), do đó, các bản ghi đăng ký (*log records*) của tất cả các tác vụ (đối với một biến cố) được viết trên bộ nhớ quảng đại. Bản phác thảo được chỉ ra bởi trình viết trước luc đăng ký (*write ahead logging*) đã tạo điều kiện để viết lại hệ thống tệp tin một cách sạch sẽ khi hệ thống có sự cố. Nếu sự cố xảy ra trước khi viết vào bộ đệm dùng để đăng ký (*log buffer*), do đó, tất cả các tác vụ bị lảng quên và vì thế phải thực hiện mới mẽ từ đầu. Nếu sự cố xảy ra giữa khoảng kết thúc đăng ký I/O và kết thúc tệp tin I/O, khi đó, cần phải kiểm tra khi cho máy hoạt động trở lại, phải xem xét và hiệu chỉnh chỗ nào đã thực hiện việc ghi chép, chỗ nào chưa (?). Sau mỗi tình huống, những tác vụ còn sai hỏng đối với mỗi biến cố được gọi tới và được thực hiện tiếp tục.

Tuy nhiên, bản phác thảo này có một vài giả định quan trọng, chúng không nhất thiết phải được thực hiện trong từng trường hợp. Do vậy, sự truyền đạt các số liệu đăng ký phải hoàn toàn không có lỗi, tức là, tệp tin đăng ký nên linh động, vì việc chuyển tải dữ liệu thường chứa đựng lỗi, do đó tệp tin sẽ không sử dụng được. Nếu giả sử có một khiếm khuyết thụ động dẫn hệ thống tới một trạng thái an toàn (*fail save*); nghĩa là một lỗi tích cực cũng chưa xoá được dữ liệu.

4.4.4. Các chức năng truy cập có cấu trúc

Những chức năng truy cập được trình bày cho tới nay là một loại đơn giản và chúng chỉ mới có ý nghĩa tách chia các giải thuật về truy cập các tệp tin. Trong rất nhiều hệ thống, có các tác vụ để tạo điều kiện thiết lập một sự truy cập có cấu trúc tới tệp tin, mà việc truy cập này được hướng tới sự tổ chức logic của dữ liệu ở trong tệp tin, sau đây sẽ lần lượt nói tới từng loại:

Các tệp tin tuần tự (sequential files):

Với các máy tính được nói ở trên, người ta đã xử lý các danh sách dữ liệu dài, mà chúng được tổ chức như là những bản ghi được sắp xếp một cách tuần tự. Vì những bàn ghi tồn tại ở những tệp tin rộng rãi ở trên đĩa mềm và lần lượt được xử lý. Hình thức tổ chức này đạt được thời gian hơi lâu. Kiểu tệp tin theo ngôn ngữ Pascal xuất phát từ các tác vụ đọc/ viết tuần tự của toàn bộ các bản ghi dữ liệu trong khoảng hai tác vụ put() và get().

Tuy nhiên, nếu người ta muốn truy cập trên các dữ liệu ở trong dãy tuần tự khác, do đó, điều đó xảy ra lại càng vô cùng chậm.

Các tệp tin tuỳ chọn (random access files):

Nhược điểm nêu ở trên nhằm dẫn tới việc khắc phục một cơ chế tệp tin, mà nó cho phép một sự truy cập trên bản ghi số liệu được sắp xếp theo một dãy tuần tự bất kỳ. Ý tưởng này có thể được thực hiện bởi sự tạo lập của hệ thống ổ đĩa cứng. Hầu hết các hệ thống tệp tin đều tạo điều kiện cho kiểu truy cập này nhờ việc biểu thị vị trí truy cập tệp tin.

Khi truy cập tệp tin tuỳ chọn hay tuần tự, một kiểu truy cập này được thực hiện nhờ một kiểu khác; tuy nhiên, nó không nhằm mục đích từ bỏ một loại nào cả. Người ta có thể thực hiện các tệp tin tuần tự trên nhiều hướng khác nhau của ổ đĩa. Đối với các tệp tin tuỳ chọn thì điều đó không có lợi lắm.

Tuy nhiên, nếu có một cơ chế nội dung tệp tin được đưa ra, thì người ta thấy rằng, tiến hành truy cập cả hai kiểu sẽ chậm hơn một kiểu. Tóm lại, người ta phải tôn trọng cơ chế tệp tin và phải thực thi một cách hiệu nghiệm bằng các phương tiện điều hành hiện có. Từ lý do này, trong các hệ thống ngân hàng dữ liệu còn có những phương pháp truy cập hiệu nghiệm hơn.

Các tệp tin chỉ số tuần tự (index sequential files):

Loại tệp tin này được cấu thành từ các bản ghi dữ liệu (*data records*), chúng được sắp xếp theo một tiêu chuẩn gọi là sắp xếp theo chìa khoá (key) và khởi đầu của tệp tin là một chỉ số. Trong thư mục chỉ số, các kiểu "đóng/mở" bằng chìa khoá được kiến tạo và được thực thi (thí dụ kiểu điền vào các tệp tin quản lý nhân sự: họ tên, ngày sinh, quê quán…); người ta có thể truy cập nhanh chóng từng phần của tệp tin; từng phần này lại được trợ cứu một khoảng giá trị thích ứng của mỗi chìa khoá. Vì một kiểu cấu trúc như thế thì đặc trưng cho tất cả các hệ thống tệp tin của các thư mục. Chúng ta có thể nhìn thấy điều này một cách chính xác hơn: Hình 4.9 chỉ ra cấu trúc kiểu chỉ số với một thí dụ về tệp tin quản lý nhân sự. Tệp tin này được sắp xếp theo một chìa khoá, thí dụ theo lứa tuổi. Các bản ghi dữ liệu riêng lẻ được tạo lập qua mũi tên xuất phát từ bậc số 0.

Hình 4.9**********

Chìa khoá lớn nhất (con số lớn nhất) của một tệp tin (ô hình chữ nhật như hình 4.9) bắt đầu từ chỉ số của bậc số 0 được mô tả ở chỉ số của bậc số 1; chìa khoá lớn nhất của một đoạn tệp tin của bậc số 1được chỉ ra ở chỉ số của bậc số 2… Nếu người ta tìm bản ghi dữ liệu của một chìa khoá xác định x, do đó, người ta phải xác định khoảng xuất phát của chỉ số thứ 2, mà trong khoảng này tồn tại chìa khoá cần tìm, và xuất phát từ giới hạn trên của khoảng này, nhờ bậc sắp xếp, các tệp tin tuần tự dịch chuyển từ dưới lên cho tới khi người ta tìm thấy con số của bản ghi dữ liệu. Sau đó, người ta có thể đọc bản ghi dữ liệu mà nó được tạo ra có mục đích từ tệp tin.

Nếu người ta thực nghiệm để mô tả cấu trúc logic cho các đối tượng vật lý - vết đường (track), hình trụ (cylinder), hình quạt (sector), do đó người ta có thể dễ dàng nhận được vấn đề để tái bản các bản ghi dữ liệu và để tổ chức bản ghi một cách mới mẽ. Thí dụ, việc tổ chức bản ghi với chìa khóa số 41 (xem hình 4.9) để tạo nên những vấn đề tương đối đầy đủ, vì mỗi hộp chữ nhật có khả năng lớn nhất là 3 chìa khoá và nó luôn luôn chứa đựng sẵn sàng 3 chìa khoá này. Do đó, chúng cần tới các cấu trúc trợ giúp cho việc tổ chức các dòng dữ liệu phát sinh khi thay đổi thường xuyên các dữ liệu và chúng đòi hỏi ngay một bản ghi mới và đầy đủ tệp tin. Đặc biệt, điều đó xuất hiện khi tổ chức dữ liệu với rất nhiều chìa khóa và thường hay thay đổi. Nếu chỉ số thư mục cho ta biết nội dùng thư mục của một hệ thống các tệp tin và với các chìa khoá này, chúng chứa đựng sự tham chiếu tệp tin (tên tệp tin, số tệp tin…) của bộ nhớ quảng đại. Đối với một hệ thống tệp tin hiệu nghiệm, nó thì khó thích hợp để tổ chức các chìa khóa thành các chỉ số thư mục theo sơ đồ khác nhau, mà sơ đồ này cho phép một sự tìm kiếm nhanh như là việc bổ sung vào hay lấy đi linh hoạt các chìa khóa.

Thêm vào đó, chúng ta thực hiện 2 thay đổi. Đầu tiên chúng tag hi chép mỗi chìa khoá vào một nút của cây thư mục chỉ một lần; điều này tiết kiệm được không

gian bộ nhớ và thời gian tìm kiếm. Sau đó, chúng ta sẽ đem lại một ý nghĩa khác cho kiểu này: Trong hình 4.9, đầu tiên, chúng ta phải quyết định trên một bậc, chìa khóa cần tìm để sắp xếp thì ở chỗ nào (?). Nếu chúng ta tìm thấy, thí dụ chìa khoá số 60, do đó, chúng ta nhận biết nó thuộc chỉ số bậc số 2, tức là chìa khoá này thì lớn hơn 48 và nhỏ hơn 99, và cho nên nó là loại được tiếp tục dẫn tới cho đến số 99. Tác vụ so sánh giữa các chìa khoá có tính nguyên tắc này được chúng ta sử dụng để khảo sát kết quả trong khoảng giới hạn này (60 tới 99) và không nối các cành vào khoảng giới hạn này, mà vào khoảng tự tạo. Bây giờ, cành này sẽ là một khoảng sơ đồ cây trung gian được kết nối giữa các chìa khoá và được dẫn tới các chìa khoá mà chúng nằm trong giá trị giữa 2 khoá nói trên. Mỗi nút chứa đựng nhiều nhất *m* sự tách nhánh và với *m-1* chìa khóa. Chúng ta nhận thấy các lá của cây (hay các bản ghi dữ liệu riêng lẻ) hình như *không thuộc cây* lúc đầu. Hình như 4.10 là một cây như thế, với nhiều nhất 3 chìa khóa trên mỗi hộp chìa khóa, mà ở đó sự kết nối tới các lá cây được biểu thị bởi các mũi tên.

Hình 4.10**********

Chúng ta có thể tạo lập một cách dễ dàng một cây thư mục như thế. Chúng ta xuất phát từ hộp chìa khóa bậc số 0 và điền đầy hộp đầu tiên này cho đến khi đầy tràn. Chìa khóa tiếp theo (thí dụ số 37) dịch lên phía trên với chỉ số của bậc số 1. Hộp chìa khóa tiếp theo lại được làm đầy và khi đó chìa khóa tiếp theo (thí dụ chìa khóa số 56) lại được dịch lên phía trên. Điều đó được thực hiện cho đến cuối bản ghi chìa khóa. Nếu chìa khóa cuối cùng là chìa khóa thứ m, do đó, nó sẽ không dịch lên bậc số cao hơn, khi đó, chúng ta tách chia m chìa khóa thành 2 hộp chìa khóa khác nhau, theo hướng nhìn xuống dưới. Bây giờ nếu trên mặt bằng kế cạnh cao hơn, hộp chìa khóa đầu tiên đã đầy tràn, do đó, tương tự như ở bậc thấp hơn, ở đây sẽ được sử trí: chìa khóa được đếm đầu tiên tới một hộp có chỉ số của bậc ở đó.

Với cấu trúc cây mới này, chúng ta đã đạt được một ưu điểm nữa để truy cập nhanh hơn: chúng ta có thể kết hợp một cách dễ dàng các chìa khoá bổ sung. Ở đây, chúng xuất phát từ phía trên rồi lần lượt đi qua cây và tìm khoảng nào mà chìa khóa tồn tại. Nếu chìa khóa chưa hề tồn tại, thì, chúng ta dừng lại một lá cây. Sauk hi chỉ định tại một lá cây như thế, bây giờ chúng ta kết hợp với chìa khóa đã có của mình. Nếu hộp chìa khoá đã đi theo qua, do vậy, chúng ta có thể thử nghiệm một lần, di chuyển các chìa khoá dư ra ở trong hộp theo hướng từ trái qua phải, mà ở đó, tất nhiên, chìa khoá ở bậc kế cạnh được dự kiến dịch chuyển tới. Tuy nhiên, độc lập với điều đó, một giải thuật kết hợp được thiết lập như sau:

+ Người ata phân chia hộp chìa khóa có m chìa khóa thành 2 hộp với các chìa khóa $S_1,\ldots,S_{[m/2]-1}$, $S_{[m/2]+1},\ldots,S_m$.

+Người ta di chuyển chìa khoá $S_{[m/2]}$ ở giữa lên trên ở bậc gần đó.

+ Trường hợp ở đó hộp chìa khóa đã chạy qua, thì người ta sẽ xử lý tại đây.

Tác vụ xoá bỏ các chìa khóa thì được phân tích ngược lại:

Bây giờ, chúng ta lưu ý rằng, chúng ta đã tạo ra được một cây mà lá cây đều ở trên một bình diện như nhau và số lượng các nhánh bị tách ra là như nhau. Do đó, chúng ta nhận thấy rằng, các hộp chìa khoá được điền đầy một cách thoả mãn với tối đa m nhánh và m-1 chìa khóa thì một nữa các nhánh [m/2] thực chất được sử dụng. Khi đó, ở gốc cây có thể xảy ra trường hợp: chỉ tồn tại 1 chìa khóa với 2 nhánh, tuy rằng con số thực chất là m nhánh.

Một loại cây cho cấu trúc dữ liệu như vậy có tên gọi là cây B (B-Tree) do D.Comer đề xướng (1979). Xem hình 4.10 ở trên, ta nhận thấy: Về hình thức, cây B là một cây được diễn giải như sau:

(1). Mỗi cành cũng như mỗi nút (tới gốc và lá) được toả ra tối thiểu k = [m/2] và tối đa k=m cành.

(2). Mỗi nút chứa đựng k-1 chìa khoá (hay chỉ số)

(3). Tất cả các lá đều ở trên một bình diện. Chúng ta lưu ý rằng, những là nào không chứa đựng chìa khóa thì không được quan tâm.

Một cây như vậy được gọi là cây bậc m. Người ta lưu ý rằng, với tổng số chìa khóa ở trên cây là N thì chúng ta có N+1 lá ở bậc thấp nhất. (Vì sao? sẽ được giải thích như dưới đây). Cây B có thể đi qua một cách nhanh chóng: Gốc cây có tối thiểu hai cành và mỗi cành trong bậc kế cạnh lại có tối thiểu [m/2] cành, do vậy, ở bậc kế n tồn tại tối thiểu một số cành là:

$$2.[m/2].[m/2]\ldots = 2.[m/2]^{n-1}. \qquad (4.1)$$

Cho nên, đối với số lượng các bậc cần dùng (gọi là dãy tuần tự tìm kiếm) với N chìa khoá và N+1 lá cây thì sẽ dẫn tới điều kiện sau đây:

$$N+1 \geq [m/2]^{n-1} \qquad (4.2)$$

Hay: $\log_{[m/2]}(N+1)/2 \geq n-1$ (4.3)

Trong đó, người ta gọi N: số chìa khoá hay số lá cây; n: số bậc; m: số cành cây.

Điều đó đã đem lại một sự tiết kiệm thời gian rất nhiều, thí dụ: khi m=200 và $N=1{,}98.10^6$ thì theo điều kiện (4.3) chúng ta cần tới:

$n-1 <= \log_{100}(0{,}99.10^6) = \log_{100}(0{,}99) + \log_{100}(100^3) < 3.$

Hay : $3 \geq n-1$

Để thoả mãn bất phương trình cuối cùng, người ta nhận thấy vế trái của nó chỉ có thể là n-1 =2; khi đó bậc tối đa của cây B cho ví dụ này là n=3. Tóm lại, với chìa khóa đã cho và bậc tối đa của cây đã xác định, tệp tin sẽ được tìm thấy một cách dễ dàng! Điều đó cho thấy các cây B có khuynh hướng tự làm thật thấp xuống khi số lượng điều vào các dữ liệu gia tăng lớn lên.

Có rất nhiều hình thái khác nhau của cây B, bằng cách, người ta có thể chọn cho mỗi bậc giá trị m khác nhau. Về nguyên tắc, điều đó không làm thay đổi thuật toán, mà nó chỉ là vấn đề dẫn xuất cấu trúc cây. Sau đây, chúng ta muốn khảo sát một khả năng quan trọng khác, để làm gia tăng ý nghĩa thực tiễn: nó được các hệ điều hành và các ngân hàng dữ liệu áp dụng một cách rộng rãi.

Ý tưởng cơ bản để cải tiến cấu trúc cây được đơn giản nhờ lý thuyết cây B là ở chỗ: không chỉ sử dụng các hộp đầy chìa khóa (còn gọi là các phân đoạn), mà còn chia nhỏ một phân đoạn thành hai phân đoạn mới khi có các bản ghi dữ liệu bổ sung, và còn phản ảnh các thay đổi trong chỉ số của bậc kế cạnh. Các phân đoạn

được liên kết năng động với nhau qua bộ chỉ thị. Sự điền đầy của hai hộp chìa khóa mới có thể rất hạn hẹp. Để cải thiện điều này, người ta có thể xem xét tới các chìa khóa có ý nghĩa ở phía trên của các nút kế cạnh đã được điều đầy hơn, do đó, chúng ta đạt được một sự cân bằng và làm đầy hộp chìa khóa mới một cách hoàn hảo hơn. Ý tưởng này được trình bày trong một giải thuật duy nhất như sau:

- Người ta thâu tóm hộp chìa khóa có m-1 chìa khóa, hộp chìa khóa kế cạnh có m chìa khoá cũng như các chìa khóa khác thuộc bậc kế cạnh cao hơn tới một hộp duy nhất. Bấy giờ, hộp này chứa đựng các chìa khóa S_1, S_2,. .,S_{2n}.
- Người ta phân chia các chìa khóa trong 3 hộp với số các chìa khóa cho mỗi hộp [(2m-2)/3],[(2m-1)/3] và [2m/3]. Khi đó, hai chìa khóa S_A và S_B với các chỉ số A và B đi lên phía trên với bậc có chỉ số cao hơn. A và B được biểu diễn bởi các biểu thức:

$$A = [(2m-2)/3] +1 \qquad (4.4)$$

$$B = [(2m-1)/3] +1 + [(2m-1)/3] +1 \qquad (4.5)$$

Bấy giờ, mỗi một trong 3 phân đoạn thì không phải một nửa mà khoảng 2/3 được làm đầy bởi các chìa khóa; do đó, đối diện với cây B, người ta nhận được một cây cải tiến hơn; cây mới này được biểu thị là cây B^* (B^*- tree). Hình 4.11 ở dưới cho thấy kết quả của việc sử dụng chìa khóa số 41 ở trong đây được chỉ ra trong hình 4.10 ở trên: hình 4.11 (a) là cây B, còn hình 4.11(b) là cây B^*.

Hình 4.11**********

Một cách hình thức, theo định nghĩa của cây B và cây B* được pahan biệt qua điều kiện thay đổi (điều kiện 1 được nêu ở trên) như sau:

1). Mỗi nhánh cũng như mỗi nút (tới gốc và lá) được lẻ nhánh thêm tối thiểu k= (2m-1)/3 và tối đa m cành.

2). Gốc có tối thiểu 2 và tối đa 2[(2m-1)/3]+1 sự phân nhánh. Do đó, gốc cây có thể được phân chia thành 2 hộp chìa khóa với số chìa khóa của mỗi hộp [(2m-2)/3] (với lưu ý cộng thêm 1 chìa cho gốc mới).

Theo D.Comer (1979), còn có nhiều dạng khác nữa của cây B; với giáo trình nguyên lý hệ điều hành, một sự trình bày về cây B như vậy là vừa đủ.

Tệp tin đảo ngược (*inverted files*) **và đa danh sách** (*multi- liste*):

Ở cơ chế tệp tin này, những thông tin về các bản ghi dữ liệu đã được sắp xếp (một cách tuần từ) được sắp xếp lại theo các chìa khóa khác nhau. Đối với mỗi chìa khóa, các tham chiếu tới các bản ghi dữ liệu được viết thành một chỉ số ở đầu tệp tin; chìa khóa cũng được chứa đựng trong các bản ghi này. Hình 4.12 chỉ ra một tệp tin với đa danh sách.

Hình 4.12**********

Khi việc điền vào bắt đầu, các danh sách được sắp xếp theo chỉ số của bản ghi, xem chìa khóa 1 và 2 (hình 4.12 ở trên). Tuy nhiên, người ta cũng có thể sắp xếp theo chỉ số các giá trị tăng hay giảm của các chìa khóa, do đó, cấu trúc phân nhánh ở trong hình 4.12 thực ra vẫn chưa rõ ràng. Điều đó được chỉ ra với danh sách các bản ghi đối với chìa khóa m (thí dụ trong hình 4.12 là 1, 3, 2, 6, 5, 8).

Nếu chúng ta kết hợp tất cả các bộ chỉ thị của một chìa khóa trong chỉ số bắt đàu của tệp tin và không viết chúng vào trong các bản ghi dữ liệu, do đó, điều này được biểu thị là *tệp tin được đảo ngược*: Khái niệm *đảo ngược* được rút ra từ viện chứng, rằng, chúng ta không khai thác giá trị chìa khóa từ bản ghi, ngược lại, bản ghi được khai thác từ chìa khóa.

Các tệp tin đảo ngược thích hợp đặc biệt tốt đối với việc đọc chọn có quan hệ với một chìa khóa xác định.

Một sự thực thi hiệu nghiệm cách tổ chức tệp tin như cấu trúc vừa nêu ở trên được thực hiện ở trong hệ điều hành các máy tính lớn. Từ lý do này, hệ điều hành Unix với tư cách là một hệ điều hành máy tính đứng vững lâu dài đối với các áp dụng về ngân hàng dữ liệu; vì trong các chương trình ứng dụng, người ta không cần thiết phải tạo lập trên những tác vụ cao hơn. Các lớp trung gian giữa các tác vụ liên hiệp (complex) và cơ chế đơn giản của các hàm tuỳ chọn ReadFile() và WriteFile() của hệ điều hành phải được mỗi người sử dụng tự viết lấy, mà với điều đó, sự cảnh giới của ngân hàng dữ liệu che phủ lên hệ điều hành Unix với mức độ cao. Trong khi đó, sự tư duy kiểu này còn có vai trò đáng kể trong nhiều áp dụng đặc biệt, vì những bộ vi xử lý với ấn bản mới có tốc độ truy cập nhanh và vì các ổ đĩa đời mới đã làm cân bằng tổn hao của một lớp trung gian rất nhiều.

4.4.5. Các tệp tin được ảnh xạ bộ nhớ (*memory mapping files*)

Các tệp tin với việc truy cập tuỳ chọn thường được làm việc mạnh mẽ trên bộ nhớ chính (RAM). Người ta có thể nghiên cứu tiếp sự phân tích như sau: Người ta có thể xem xét các tệp tin ở trên bộ nhớ quảng đại như là một sự kế tục của bộ nhớ chính trên bộ nhớ quảng đại. Cho nên, điều đó đặt ra, phải thực hiện bước tiếp theo, và một cách đúng mức (trước sau như một) phải tái tạo một sự kết nối trực tiếp một tệp tin với một khoảng của bộ nhớ chính, mà tệp tin được tách chia thành những khoảng có chiều dài bằng một trang. Khi đó, người ta gọi là tệp tin ảnh xạ bộ nhớ; tương tự, bộ nhớ cũng được phân chia thành các trang.

Một kiểu kết nối như thế sẽ đạt được bởi gọi hệ thống; trong đó, các trang của một tệp tin được ảnh xạ như là một sự thay thế các trang bộ nhớ chính ở trong không gian địa chỉ ảo của một tiến trình. Hình 4.13 minh hoạ điều này một cách đầy đủ.

Việc thực thi một cơ chế như vậy thì tương đối đơn giản: Khi một miền mà trên đó các trang của một tiến trình được nạp, do đó, miền của tệp tin được chỉ ra ở trên bộ nhớ quảng đại. Điều đó, mang lại những lợi thế khác nhau như sau:

+ *Độ nhanh nhạy*:

Trang của tệp tin được đọc, nếu nó được sử dụng; việc sao chép không cần thiết được loại bỏ.

+ Ghi vào bộ đệm tự động:

Ở hầu hết các chương trình sử dụng các tệp tin, một bộ đệm logic được viết để cần thiết thực hiện vài tác vụ đọc/ viết. Điều đó thì khác biệt với ảnh xạ bộ nhớ: Với cơ chế trang (paging mechanismus), việc ghi vào bộ đệm được thực hiện một cách tự động và hiệu nghiệm bởi hệ điều hành.

Hình 4.13**********

Các tác vụ cờ hiệu cho phép sử dụng cùng nhau các khoảng bộ nhớ với nhiều tiến trình. Từ lý do này, có một cơ chế như thế ở trong hệ điều hành. Tuy nhiên, cơ chế này còn tồn tại một giới hạn, do đó, một tệp tin đang tồn tại không chỉ được đọc và viết, mà còn thay đổi chiều dài của nó. Khi đó, với ảnh xạ bộ nhớ, vấn đề vừa nói đã được giải quyết.

Thí dụ về tệp tin ảnh xạ bộ nhớ ở Unix:

Để tạo các tệp tin ảnh xạ, hệ điều hành Unix có các gọi hệ thống quan trọng:

mmap() để thiết đặt ảnh xạ ở không gian địa chỉ ảo trong phạm vi một tệp tin;
munmap() để kết thúc ảnh xạ. Nếu nội dùng bộ nhớ bị thay đổi, do đó các trang đã được thay đổi sẽ được viết trở lại trên tệp tin;
msync() để thực hiện tệp tin từ bộ nhớ

Thêm vào đó, còn có một vài gọi hệ thống khác cũng thường hay được dùng:

msem_init() để tạo lập các cờ hiệu cho phạm vi bộ nhớ;
msem_lock() để giải phóng các khoảng bộ nhớ;
msem_unlock() để giải phóng các khoảng bộ nhớ;
msem_remove() để rời khỏi bộ nhớ

Tệp tin ảnh xạ bộ nhớ ở Windows NT:

Đối với việc ảnh xạ giữa bộ nhớ và hệ thống các tệp tin, thì bộ điều hành I/O và bộ điều hành cơ cấu ảo được coi là cùng tác dụng với nhau. Nhờ hàm gọi CreateFileMapping(), một đối tượng (*object*) được tạo lập; đối tượng này có thể được mở bởi các tiến trình khác nhau dưới cái tên đã biết OpenFileMapping(). Theo đó, một khoảng bộ nhớ tuỳ ý (vài Byte đến 2 GBytes) được chuẩn bị. Các khoảng bộ nhớ riêng rẻ có thể được ảnh xạ với hàm MapViewOfFile() ở trong không gian địa chỉ ảo. Với hàm gọi hệ thống FlushViewOfFile(), phạm vị tệp tin được thực hiện và với hàm UnmapViewOfFile() nó được đóng lại.

Cơ chế này được sử dụng để trao đổi thông tin của các tiến trình; trong đó, người ta chọn tệp tin làm việc trên phạm vi bộ nhớ chia xẻ. Với cơ cấu ảnh xạ fie,

tệp tin này còn có thể được xử lý một cách trực tiếp bởi nhiều tiến trình. Đối với việc làm đồng bộ các kiểu truy cập này, không có chức năng đặc biệt xem xét, mà chỉ có việc sử dụng bình thường các cờ hiệu.

4.4.6. Các tệp tin đặc biệt (*special files*)

Có những cơ chế khác nhau của hệ điều hành, mà với sự trợ giúp của hệ thống tệp tin, các cơ chế này có thể được chuyển đổi một cách hài hoà; mà không cần phải lý giải hay tạo lập những tệp tin đích thực.

Một trong những phương pháp mở rộng nổi tiếng nhất của hệ điều hành Unix là việc mô hình hóa các thiết bị vật lý như những *tệp tin đặc biệt*. Mỗi cái tên của một tệp tin được thu xếp là một thiết bị vật lý; dĩ nhiên, các tệp tin này được biểu thị với trạng thái của một *tệp tin đặc biệt*. Tất cả các việc truy cập và các thay đổi trạng thái đối với tệp tin biểu trưng này (*symbol file*) đều có tác dụng trực tiếp lên thiết bị, chứ không phải tác dụng lên tệp tin. Nếu tệp tin được viết lên một thiết bị như thế, do đó, khi thực hiện, tệp tin sẽ được dịch chuyển tới thiết bị (chẳng hạn tới một thiết bị đầu cuối nào đó) và khi đó, chúng được thể hiện trên màn hình. Nếu một tiến trình mở tệp tin đặc biệt của thiết bị đầu cuối để đọc, do đó, nó sẽ cảm nhận tất cả các ký tự được viết trên bàn phím.

Lợi ích của việc mô hình hóa như vậy nằm trong những điều có thật, rằng một mặt, nó đã tạo ra khả năng cho người sử dụng truy cập trực tiếp và hiệu quả lên các tính chất vật lý của thiết bị; mặt khác, các cơ chế quản lý và bảo vệ của hệ điều hành hoạt động rất hiệu quả; và do đó, việc truy cập được điều chỉnh và điều khiển một cách hợp lý.

Tệp tin đặc biệt ở Unix:

Trong hệ điều hành Unix, có một thư mục tồn tại với đường dẫn /dev; ở đó, các tệp tin tồn tại với các thiết bị khác nhau. Có hai kiểu tệp tin thiết bị: Loại thứ nhất là các tệp tin tuần tự, gọi một cách đầy đủ là các tệp tin đặc biệt hướng ký tự (*character oriented special files*); chúng được bổ sung cho tất cả các thiết bị hướng ký tự như các thiết bị đầu cuối, các ổ đĩa từ tính, máy in… Loại thứ hai là các tệp tin tuỳ chọn, gọi đầy đủ là các tệp tin đặc biệt hướng tổng thể (*block oriented special files*); khi đó, những khoảng bộ nhớ bấy kỳ (block) có thể được xem là các tuỳ chọn, thí dụ các ổ đĩa cứng, các đĩa mềm…Mỗi một loại tệp tn đặc biệt được tạo ra bởi sự trợ giúp của gọi hệ thống mknod() và ở tại các gọi hệ thống close(), read(), write() tệp tin này thích hợp cho các thủ tục đặc biệt, mà các thủ tục này lại được các bộ kích thích thiết bị của hệ điều hành sử dụng. Thí dụ, với đường dẫn /dev/tty, các ký tự được xuất ra và được đọc trên thiết bị đầu cuối.

Người ta lưu ý rằng, với cơ chế vừa nêu, một thiết bị như thế có thể được vay mượn dưới những tên tệp tin đặc biệt khác nhau. Thí dụ, một ổ đĩa từ tính có thể được coi như một tệp tin đặc biệt, chẳng hạn tệp tin tuần tự liên tục. Hoặc giả, dưới một cái tên khác, người ta có thể coi thiết bị này như là một tệp tin đặc biệt

hướng tổng thể (khối) và với cái đó, để truy cập trực tiếp trên một trong các khối lưu trữ tuần tự; chẳng hạn, một khối với số 15 được đọc trước một khối số 5. Với tác vụ này, các lệnh định vị bổ sung ở trong bộ kích thích thiết bị phải được sử dụng.

Ở một vài thiết bi như máy in và thiết bị đầu cuối thì không thể lẫn lộn với thiết bị khối (tổng thể), do đó, nó thì có điều kiện để tất cả các thiết bị đều có thể xưng hô là tệp tin đặc biệt hướng ký tự cũng như tệp tin đặc biệt hướng tổng thể (khối). Vì giao diện để truy cập trên các dữ liệu thuần khiết thì cho phép không có cấu trúc và không cần thông tin quản lý, do đó, các tệp tin đặc biệt này còn được biểu thị là những thiết bị nguyên sơ.

Để sử dụng hệ thống tệp tin tồn tại trên một ổ đĩa, với lệnh mount(), nút gốc của cây phải được ảnh xạ trên một thư mục của hệ thống tệp tin đang tồn tại. Tất cả các tệp tin trước đó đã ở trong thư mục thì sẽ được nạp lần đầu tiên vào một hệ thống tệp tin mới.

Trong hệ điều hành Unix, tệp tin đặc biệt được phô bày những khả năng phi thường do quan điểm của người sử dụng khi truy cập trên thiết bị vật lý. Tất cả sự thay đổi trạng thái (như thay đổi tốc độ truyền đạt, thay đổi kiểu khi chịu tải nối tiếp…) phải được thực hiện nhờ hàm gọi hệ thống đặc biệt IOCTL() với sự biểu thị thích hợp tệp tin đặc biệt. Nếu người ta muốn sử dụng một tính chất phần cứng đặc biệt, thí dụ: một mật độ cao khi viết vào các đĩa mềm hay một sự cuộn lại tự động theo gọi hệ thống close(), do đó, người quản lý trong hệ điều hành Unix tạo ra một tệp tin đặc biệt mới bằng gọi hệ thống mknod(); đồng thời anh ta cũng chuyển giao cho bộ kích thích hệ điều hành những thông số thích hợp khi gọi hệ thống.

***Tệp tin đặc biệt ở Windows NT*:**

Các nhà thiết kế hệ điều hành Windows NT đã học tập từ các hệ điều hành đương thời và đã đảm nhận cơ chế của các tệp tin đặc biệt của hệ điều hành Unix. Thêm vào đó, họ đã phân bổ việc quản lý cây thư mục cho những người quản lý khác nhau. Việc điều hành các đối tượng tệp tin đặc biệt cho phép người quản lý đối tượng có toàn quyền trên các hệ thống tệp tin và bắt buộc phải quan tâm tới việc quản lý tệp tin và quản lý I/O của nhân hệ điều hành. Mỗi thiết bị phải phù hợp cho một tệp tin ảo (virtual file), tệp tin này đã xác định các đối tượng và có những phương pháp đã khẳng định. Những phương pháp này sẽ được người ta nói tới và làm việc nhiều trên các thủ tục của bộ kích thích thiết bị ở chương sau.

4.5. Việc thực thi cơ chế tệp tin

4.5.1. Biểu diễn bộ nhớ liên tục

Cấu trúc cơ bản của một phương pháp như thế thì tương đối đơn giản: Bắt đầu không gian bộ nhớ là một chỉ số hay một thư mục, mà trong đó các tệp tin được

liệt kê vào; và không gian còn lại để ghi tiếp các tệp tin. Một tệp tin thì được viết liên tục với nhau trong một đoạn. Ý tưởng vừa nêu trên phát triển theo thời gian, trong đó mỗi tệp tin chiếm một vị trí của đĩa từ và các thư mục là sự khái quát các ngăn xếp với các dãy ở trong đĩa từ. Vì ý tưởng này thì quá cứng nhắc để thay đổi tệp tin, do đó, nó chỉ được sử dụng để thu xếp chỗ cho nhiều tệp tin nhỏ (gọi là các đối tượng) ở trong một tệp tin lớn (gọi là thư viện các tệp tin).

4.5.2. Biểu diễn bộ nhớ kiểu danh sách

Việc quyết định cho một khoảng bộ nhớ liên tục là ở chỗ: bộ nhớ phải được phân ra những khoảng có độ lớn bằng nhau (block) và tất cả các block của một tệp tin phải được kết nối trong một danh sách. Ưu điểm của phương pháp này là ở chỗ, tại một thư mục hỏng (bị xoá nhầm lẫn…) được biểu thị gấp đôi và các thông tin tệp tin (tên, các quyền truy cập…) được chứa đựng ở đầu tệp tin.

Nhược điểm của phương pháp này là việc truy cập tệp tin kém hiệu nghiệm: Nếu người ta muốn đọc block số 123, do đó, người ta phải đọc trước đó tất cả 123 blocks để có thể lướt qua danh sách cho đến block số 123. Hình 4.14 chỉ ra một phương pháp như thế.

Hình 4.14**********

4.5.3. Biểu diễn bộ nhớ được biểu thị qua chỉ số tập trung

Người ta có thể tránh được sự bất lợi của kiểu danh sách tệp tin, bằng cách, người ta không phân bổ trong môi trường bộ nhớ các thông tin về dãy tuần tự số khối cũng như danh sách các khối liên tục với nhau, mà người ta kết hợp chúng trong một khối tập trung. Phương pháp này được gọi là cấu trúc chỉ số tập trung, nó chính là một danh sách, mà trong đó, mỗi sự điền vào chứa đựng số của khối tiếp tục trên đó.

Hình 4.15**********

Bảng cấp phát các tệp tin (FAT) ở MS-DOS:

Với hệ điều hành MS-DOS, mỗi đĩa từ được phân chia thành nhiều cung đoạn (*sectors*), mỗi sectors có độ lớn khoảng 512 Byte. Bắt đầu là sector số 0 được bố trí cho cung đoạn khởi động (*bootstrap-sector*), còn bảng cấp phát tệp tin (file allocation table: FAT) được bố trí các sector số 1 đến số 5; các bản sao bảo vệ tồn tại ở các sector số 6 đến số 10. Bảng FAT thì bao gồm các sự điền vào với chiều dài 12 bits hay 3 số thập lục phân với các giá trị 000_H tới FFF_H, cho nên, bảng FAT có một không gian lớn nhất 512*5 = 2560 Byte ứng với khoảng 1706 sự điền vào. Mỗi sự điền vào thì tương ứng một đơn vị bộ nhớ của đĩa từ. Mỗi đơn vị bộ nhớ được gọi là một sluster (tức một cụm sector) ứng với 1024 Byte, do đó, một

đĩa từ trong MS-DOS có thể đạt được nhiều nhất một dung lượng khoảng 1,7 MB và mỗi tệp tin đạt tối thiểu 1kByte.

Mỗi một sự điền vào bảng FAT có ý nghĩa như sau:

000_H các cluster (cụm) còn trống;

001_H…$FF0_H$ cluster đã đầy; sự điền vào biểu thị bởi số của cluster kế tiếp của tệp tin;

$FF1_H$…$FF7_H$ các cluster bị khuyết tật không thể sử dụng do lỗi bề mặt vật liệu;

$FF8_H$…FFF_H các cluster cuối cùng của tệp tin.

Vì hai sự điền vào đầu tiên được sử dụng cho việc quản lý, còn các tệp tin bắt đầu với sector số 18, do đó, số sector đầu tiên của một cluster được dẫn ra bởi biểu thức:

(số của sector) = (chỉ số của điền vào cluster -2)*2 +18. (4.6)

Bình thường việc điền vào bảng FAT tạo điều kiện để xác định cluster tiếp theo của tệp tin một cách độc lập với việc che phủ, do đó, việc biểu thị một cluster khởi động đủ để đọc một tệp tin của đĩa từ một cách đầy đủ. Nó thì cần thiết, phải kết nối việc điền vào với thông tin đã phủ tĩnh lại. Tuy nhiên, việc mã hoá này giới hạn số lượng các cluster đã quản lý; các bảng FAT có độ lớn tuỳ ý được kéo dài quá 5 sector, mà ở đó, người ta có thể chỉ quản lý $16^3 = 2^{12} = 4096$ cluster ứng với khoảng 4 MB.

Trên cơ sở này, ở ổ đĩa cứng, kích cỡ bảng FAT trong MS-DOS sẽ được thay đổi: chỉ số của cluster là một số dài 16bit. Điều đó cũng có nghĩa rằng, với $2^{16} = 65536$ cluster, thì không gian của đĩa từ phải được che phủ. Khi một ổ đĩa từ có dung lượng 2 GB- đó là một độ lớn tiện lợi cho các máy tính, điều đó có nghĩa một cluster có độ lớn $2^{31}/2^{16} = 2^{15} = 32168$ Byte. Vì hầu hết các tệp tin thì ở trong khoảng 1kByte, nghĩa là phải biếu mất 31 đến 32 kByte cho việc quản lý hệ thống, khi đó hiệu suất sử dụng quá thấp. Cho nên ở trong MS-DOS, một ổ đĩa từ vật lý như thế được phân làm nhiều ổ đĩa logic, mà với một hệ thống tệp tin riêng lẻ, khi đó, chúng chỉ cần một độ lớn cluster nhỏ thôi.

Ngược lại, phương pháp này vẫn còn tồn tại vấn đề, rằng khi các hệ thống tệp tin lớn thì danh sách tập trung cũng rất lớn. Vì thư mục chung phải ở trong bộ nhớ chính để có thể nhìn lướt qua, do đó, không gian bộ nhớ chính chắc chắn được che phủ…

Thí dụ: Giả thử một ổ đĩa cứng có dung lượng 2 GByte = 2^{31} Byte và mỗi block có dung lượng 1 kByte = 2^{10} Byte; một danh sách chỉ số thư mục có tối thiểu $2^{31} / 2^{10} = 2.10^6$ lần điền vào được dẫn ra. Mỗi lần điền vào phải đón nhận tối thiểu một con số tuỳ ý 0…2^{20}, do đó, tối thiểu 20 bit hay 3 Byte cho mỗi lần điền vào được sử dụng. Điều đó, có nghĩa rằng, đối với một danh sách chỉ số tập trung, có khoảng 6 MByte bộ nhớ chính luôn luôn phải được dự trữ, và lưu ý rằng, không phải chỉ có sự điền vào sử dụng hết dung lượng đó!

4.5.4. Biểu diễn bộ nhớ biểu thị chỉ số phân bổ

Một quyết định quan trọng để lựa chọn tệp tin chỉ số tập trung đối với bộ nhớ quảng đại thì bao gồm việc dẫn ra một danh sách chỉ số riêng lẻ đối với mỗi tệp tin và danh sách này phải được lưu trữ trong một block, hãy so sánh trong hình 4.16 ở dưới. Nếu chỉ có số tệp tin phù hợp được làm hài lòng thực thụ, do đó, khối chỉ số (*index block*) của nó sẽ phải được hoàn trả lại.

Hình 4.16**********

Tất nhiên, các danh sách chỉ số sẽ dài hơn, khi đó, các tệp tin thuộc danh sách đó cũng dài hơn. Bấy giờ, trong các hệ thống tệp tin đang tồn tại, người ta lưu ý rằng, có rất nhiều tệp tin tồn tại với vài block và cũng có ít tệp tin tồn tại trong nhiều block. Với lý do này, người ta có thể dẫn giải ra một block chỉ số chính, mà đối với hầu hết các tệp tin, block này đã đạt yêu cầu. Đối với một vài tệp tin cần tới nhiều không gian cho chỉ số, chúng ta phải đặt vào một block chỉ số tiếp theo,mà địa chỉ logic của nó đứng ở cuối danh sách chỉ số. Nếu các tệp tin quá lớn, thì điều xẩy ra rất chậm, vì khi đó, tất cả các block chỉ số phải được đọc một cách tuần tự vào đĩa từ để tìm kiếm địa chỉ vật lý của block cần tìm. Điều đó nhằm mục đích để tổ chức một cách hệ thống các danh sách chỉ số đã được dẫn ra, và do đó, nhằm tăng tốc độ chuyển đổi (*speed of translation*) các địa chỉ logic của tệp tin tới các địa chỉ vật lý của thiết bị. Hình 4.17 ở dưới đây sẽ làm sáng tỏ nguyên tắc đã được trình bày trong mục 3.3 ở chương trước về chuyển đổi nhiều bậc từ không gian địa chỉ ảo thành không gian địa chỉ vật lý.

Hình 4.17**********

4.5.5. Thực thi hệ thống tệp tin ở trong Unix

Hệ điều hành tiếng tăm nhất sử dụng phương pháp vừa trình bày ở trên, đó là hệ điều hành Unix. Ở đây, trong những ứng dụng, một sự biến đổi đặc biệt được thử nghiệm để giữ cố định những thông tin xác đáng đối với các tệp tin ngắn cũng như dài ở trong một block chỉ số.

Về điều này, thì nút chỉ số đầu tiên (*first index-node*) được tạo bởi một phần khái quát, mà trong đó chứa đựng các tên, các địa chỉ đầu tiên của các block dữ liệu, và các địa chỉ của một cây chuyển đổi gián tiếp một bậc, hai bậc và ba bậc. Hình 4.18 chỉ ra một thí dụ tổng quát về điều đó.

Nội dùng được chỉ ra ở trên của một nút chỉ số đã được tu chỉnh trong các hệ thống cụ thể và được mở rộng bởi các sự biểu thị thực chất ở bên trong, thí dụ ở Unix System V, với các thông tin cờ hiệu để ngăn hãm tệp tin, chúng đã mở rộng không gian cho bộ chỉ thị (khi không gian bộ nhớ còn trống, nút chỉ số đầu tiên sẽ được treo vào danh sách bậc hai kết hợp), các sự điền vào đối với bảng FAT với hàm mount() và các thông tin trạng thái đối với việc truy cập ở trong mạng máy

tính… Ở trong Unix System V không tồn tại bảng FAT gián tiếp hai bậc và ba bậc; tuy nhiên, nút chỉ số vẫn có thể tồn tại ở các block tiếp theo.

Mỗi tệp tin được thu xếp một cách chính xác một nút chỉ số; tương tự, thư mục tệp tin cũng tự thu xếp như là một tệp tin lưu trữ. Các số của nút chỉ số sẽ được trao một cách tuần tự; còn vị trí giao dịch hiện hành ở trên bộ nhớ quảng đại (số của block vật lý) được kết hợp thành một block đặc biệt, còn gọi là nút đặc biệt (super node). Nếu super node này bị phá hỏng, do đó, hệ thống tệp tin không sử dụng được.

Hình 4.18**********

Tham chiếu chìa khóa của một tệp tin là số của nút chỉ số, còn nếu có nhiều tham chiếu (nhiều tên tệp tin) đối với tệp tin này thì chúng sẽ tồn tại trong các thư mục khác nhau.

Một thư mục sẽ được tạo lập một cách đơn giản: Nó chỉ chứa đựng số của nút chỉ số của tệp tin và tên (như chiều dài tên, chiều dài của sự điền vào…), ngoài ra không cần gì thêm. Tất cả những cái khác như quyền truy cập…thì được lưu trữ ở block đầu tiên của tệp tin. Vì mỗi thư mục chiếm một khoảng bộ nhớ, mà khoảng này là đa bậc của độ lớn block (tức là một đơn vị bộ nhớ được chuyển đổi nhanh), do đó, các thư mục có thể được đọc hay được lưu trữ nhanh và hiệu quả. Việc biểu diễn bổ sung chiều dài tên hay độ lớn của sự điền vào (tức chiều dài tên được mở rộng trên giới hạn 1 từ) thì cho phép lưu trữ một sự điền vào tên ngắn hơn ở trong một thư mục.

4.5.6. Thực thi hệ thống tệp tin ở Windows NT:

Ở trong hệ điều hành Windows NT, đơn vị cơ bản để quản lý tệp tin là một volume (dung lượng). Đó là một đơn vị bộ nhớ logic (ở trên bộ nhớ quảng đại), nó có thể bao gồm một hay nhiều đĩa từ riêng lẻ, mà trên đó còn tồn tại những khoảng bộ nhớ được liên kết trong một đơn vị logic.

Mỗi volume thì bao gồm một sự sắp xếp theo một nguyên tắc nào đó các dữ liệu hay các tệp tin đã được đánh số. Mỗi tệp tin có một số tham chiếu, gọi là số tham chiếu tệp tin (*file reference number*) với khoảng 64 Bit, tức là nó bao gồm 48bit cho số tệp tin và 16bit cho số tuần tự. Số tuần tự sẽ được gia tăng theo mỗi khi xoá tệp tin thuộc số tệp tin, do đó, người ta có thể phân biệt giữa sự tham chiếu tới các số tệp tin trung gian đã bị xoá và một tệp tin hiện hành, nếu chúng tham chiếu tới các số tệp tin tương tự.

Ưu điểm của ý tưởng này là cung cấp cho người ta tương đối đầy đủ tất cả các thông tin về tệp tin:

+ Cơ cấu truy cập tệp tin thì kiên đình và đơn giản, kể cả các thông tin khởi động hệ thống;

+ Các thông tin về bảo vệ được tách bạch theo các thành phần quản lý, và với điều này, nó thì có thể thích hợp hơn ở các ứng dụng;

+ Khi các phần đĩa từ không thể sử dụng được, do đó, các thông tin quản lý (mà người sử dụng không thể nhìn thấy…) có thể được nạp trở lại trên các phần khác của đĩa từ.

Mỗi volume chứa đựng một bảng trung tâm gọi là bảng tệp tin chủ (master file table: MFT), mà trong đó, các tệp tin hệ thống được ghi chép rất đầy đủ.

Hình 4.19 ở dưới đây cho thấy, chính nó là một tệp tin bao gồm tất cả các dữ liệu về các thư mục cũng như các thông tin về việc tự khởi động của hệ thống (bootstrap). Tất cả 16 tệp tin đầu tiên này được giữ chặt và tạo thành các tệp tin hệ thống của hệ điều hành Windows NT (NT file-system: NTFS).

Tệp tin 0	Bảng tệp tin chủ (MFT)
Tệp tin 1	Bản sao bảo vệ bảng MFT đối với các tệp tin hệ thống, định vị đĩa từ.
Tệp tin 2	Tệp tin logic: Các tác vụ thay đổi cấu trúc NTFS, được biểu thị ở đây và đảm bảo các hoạt động chuyển đổi nhân tử.
Tệp tin 3	Tệp tin volume: Các thông tin trạng thái của volume như tên, ấn bản… Nếu Bit corrupted được đặt thì chương trình chkdsk phải được thực hiện; chương trình này kiểm tra hệ thống tệp tin thường trú và sửa chữa lỗi đã được khẳng định.
Tệp tin 4	Bảng định nghĩa thuộc tính: Ở đây biểu thị tất cả các kiểu thuộc tính tồn tại ở trong volume, như trạng thái
Tệp tin 5	Thư mục gốc: Tên của thư mục gốc, thí dụ ký tự "\"
Tệp tin 6	Tệp tin bitmap: Bảng che phủ của đơn vị bộ nhớ volume (sluster), xem thư mục 3.1.1
Tệp tin 7	Tệp tin boot: Tệp tin khởi động này được tạo ra nhờ chương trình format và chứa đựng mã bootstrap của Windows NT và địa chỉ vật lý của tệp tin MFT
Tệp tin 8	Tệp tin bad cluster: Thư mục của colume cluster không thể sử dụng
Tệp tin 9	…
	…
Tệp tin 16	Các tệp tin của người sử dụng bình thường và các thư mục

Hình 4.19. Dãy tuần tự của 16 tệp tin đầu tiên của hệ thống các tệp tin (NTFS) ở Windows NT

Cấu trúc logic của một volume được tạo thành như sau: Cấu trúc dữ liệu trung tâm của một hệ thống tệp tin là một bảng tệp tin chủ (master file table: MFT); bảng này đóng vai trò như một nút đặc biệt ở trong Unix. Bảng MFT chứa đựng một sự điền vào đối với mỗi tệp tin và mỗi thư mục. Sự điền vào này bao gồm dãy tuần tự các thuộc tính của tệp tin. Mỗi thuộc tính có thể hoặc tồn tại một cách đầy đủ, gọi là thuộc tính thường trú (*resident atribute*), hoặc là, nếu thuộc tính này quá dài (thí dụ các dữ liệu riêng lẻ tồn tại như là một thuộc tính bình thường), thêm vào đó, sự tham chiếu tới các block bộ nhớ tiếp theo (còn gọi là các cluster) được biểu thị là "runs". Mỗi thuộc tính có một sự kéo căng hai đầu (header); bên cạnh sự biểu thị "*resident*" (thường trú) hay "nonresident" (không thường trú), sự kéo căng này còn chứa đựng sự bắt đầu và cả chiều dài thuộc tính trong khi điền vào tệp tin. Nếu thuộc tính là không thường trú, do đó, nó chứa đựng sự bắt đầu và chứa đựng cả chiều dài của các khối bổ sung. Thêm vào đó, số tương đối của các khối 0…N, gọi là số của các cluster ảo (virtual cluster number: VCN), sẽ được biểu diễn ở trong tệp tin như là việc sắp xếp chúng từ số của cluster ảo thành số

của cluster logic (logic cluster number: LCN). Hình 4.20 biểu diễn một sự điền vào như vậy. Những thuộc tính của tệp tin được giải thích một cách đầy đủ ở trong mục 4.3.2 ở trên.

Nếu theo đó các block của bộ nhớ chỉ tồn tại toàn số 0, và nếu một thuộc tính chuẩn được biểu thị compress (nén lại) đối với tệp tin, do đó, các block trống của bộ nhớ sẽ không lưu trữ. Đáng lẽ là như vậy, nhưng khi đếm các cluster ảo, các khối cluster thích hợp sẽ được nhãy qua một cách đơn giản, do đó, ở trong dãy tuần tự VCN, các lỗ trống vẫn tồn tại.

Hình 4.20**********

Khi đọc các tệp tin, thì điều đó được các tệp tin hệ thống của Windows NT nhận biết và được đưa trở lại với các block khởi xướng là các số 0. Nếu các block của tệp tin nén không chứa đựng các số 0, do đó, một giải thuật nén tổng quát được đáp ứng; tuy nhiên, giải thuật này chỉ đạt được nén nhanh chứ không đạt được nén hiệu suất.

Ở một thư mục, dòng dữ liệu bao gồm một chỉ số với tên tệp tin, mà bộ đệm của nó được tổ chức thành dạng cây B* (thí dụ các tệp tin hệ thống của hệ điều hành OS/2), xem mục 4.4.4 ở trên.

Chỉ số này bao gồm 3 thành phần: phần thứ nhất là các tên được đánh chỉ số ở trong dãy tuần tự từ điển (gọi là chỉ số gốc: *index root*) và các chỉ dẫn của chúng; phần thứ hai là các bảng sắp xếp (VCN -> LCN) của các bộ đệm để cấp phát chỉ số (*index allocation*) cho các dữ liệu không thường trú; phần thứ ba là các bảng che phủ của các bộ đệm còn gọi là các bit ảnh xạ.

Mỗi sự điền vào bao gồm số tham chiếu tệp tin, tên tệp tin, thời gian tạo lập và chiều dài tệp tin; tất cả những cái đó được bảng MFT sao chép lại ở tại đây. Tuy nhiên, một sự chi phí cho việc đồng bộ sẽ được bổ sung; chi phí này có lợi nhờ thời gian tìm kiếm ngắn hơn ở trong các thư mục. Tên tệp tin được đánh chỉ số ở trong dãy tuần tự từ điển. Mỗi tên tệp tin được chỉ dẫn trên một bộ đệm chỉ số và chứa đựng các sự điền vào với chỉ số nhỏ hơn. Một cây B* thường phát triển theo chiều rộng tốt hơn chiều sâu, vì nó cho phép các tác vụ tìm kiếm rất nhanh.

Các bộ đệm bổ sung chứa đựng những chỉ dẫn tệp tin trong dạng đóng gói; ở lần sử dụng đầu tiên (mount) của tệp tin, cấu trúc dữ liệu phải được đóng gói ở trong bộ nhớ. Nhờ đó, một sự lưu trữ với khoảng 15 tên tệp trên mỗi bộ đệm với 2kB hay 4 cluster (khoảng 512 Byte cho mỗi cluster) là có thể.

4.6. Các bài tập về quản lý tệp tin

4.6.1.Các bài tập về tệp tin

Bài tập 4.1. Về các thư mục cô lập

Giả sử thư mục Test là thư mục gốc. Bạn hãy lập thư mục con của thư mục Test. Trên thư mục vừa thiết lập, bạn hãy tạo ra một kết nối cứng (hard link). Bấy

giờ, nó tồn tại một kết nối quay vòng. Bạn hãy thử nghiệm xoá thư mục. Test từ đường dẫn gốc. Bạn đạt được cái gì ?

4.6.2. Các bài tập về tên tệp tin và thư mục

Bài tập 4.2. Về đổi tên tệp tin

Ở ví dụ đổi tên file, giải thuật cho Windows NT được mô tả để đổi một tên tệp tin dài thành tên ngắn và rõ ràng.

a). Tên các tệp tin có kích cỡ bao nhiêu để có thể mã hóa thành một tên ngắn và rõ ràng với 8 ký tự ? Với lưu ý: các ký tự của phần tên chính không được ít hơn 6.

b). Bạn có thể thay đổi sơ đồ như thế nào để mã hóa một số lượng lớn các tên tệp tin dài thành tên tệp tin rõ ràng với 8 ký tự theo mã ASCII ? Bạn hãy suy nghĩ vì sao các việc thực thi đã không chọn khả năng này ?

Bài tập 4.3. Về tên đường dẫn

Nhược điểm và ưu điểm của tên đường dẫn tương đối với tên đường dẫn tuyệt đối là gì ?. Ở đâu và khi nào người ta có thể thay thế tên đường dẫn nào cho phù hợp ? Bạn hãy lý giải điều này với sự trợ giúp của một hệ thống biên dịch!

Bài tập 4.4. Về quản lý tệp tin hướng đối tượng

Giả sử người ta cho phép việc quản lý tệp tin trong Unix một cách hướng đối tượng. Những phương pháp và những thuộc tính nào là cần thiết đối với một đối tượng thư mục và đối với một đối tượng tệp tin ở trong hệ điều hành Unix ?

Bài tập 4.5. Về danh sách điều khiển truy cập

Nhược điểm và ưu điểm của danh sách điều khiển truy cập là gì ? Khi điều đó được người sử dụng thiết đặt cho tất cả các tệp tin.

4.6.3. Các bài tập về các chức năng tệp tin

Bài tập 4.6. Về file mở

Một hệ điều hành có thể thực hiện các tác vụ tệp tin trên hai loại khác nhau:

+ Bình thường để truy cập tệp tin, trước hết người sử dụng phải mở tệp tin;

+ Hoặc điều này xảy ra một cách tự động khi truy cập tệp tin. Ở trường hợp thứ nhất, một sự truy cập tệp tin được mô tả như sau:

open()...read() ...I/O-Ops...Close()

Trong trường hợp thứ hai, người ta chỉ việc truy cập tệp tin (I/O- Ops); sau đó, tệp tin được đóng lại khi chương trình kết thúc. Với hai cách nói trên, tồn tại những ưu điểm và nhược điểm nào ?

Bài tập 4.7. Lệnh copy

Mỗi hệ điều hành thông thạo một số lệnh để quản lý các tệp tin. Bạn hãy đặt mình vào vị trí một nhà thiết kế hệ thống để viết một lệnhh copy.

a). Bạn hãy thực thi lệnh copy để sao một tệp tin của hệ điều hành với các chức năng thư viện về quản lý tệp tin (read, write...).

b). Những thay đổi nào phải được lựa chọn đối với lệnh move ? Nếu đầu tiên đạt được việc copy, thì khi đó, tệp tin ở thư mục nguồn có bị xóa không ? Xin lưu ý: Bạn hãy suy nghĩ tới cơ chế bảo vệ !

c). Lệnh copy phải được mô tả như thế nào để sao chép toàn bộ thư mục kể cả các thư mục con ? Bạn hãy thực thi lệnh này xem !

Bài tập 4.8. Về xuất- nhập

Tại sao trong Unix có sự khác nhau giữa xuất chuẩn và xuất lỗi chuẩn, nếu cả hai đều dẫn tới sự mặc định trên màn hình ?

Bài tập 4.9. Thực hiện lệnh echo bởi thiết bị đặc biệt

Cái gì sẽ xảy ra, nếu trong Unix bạn xuất ra một bài text với chương trình echo (dội lại) theo các đường dẫn *dev/tty* hay *dev/null* ?

4.6.4. Thực thi việc tổ chức tệp tin

Bài tập 4.10. Về i-node

Trong hệ điều hành Unix, các i-node chứa đựng 20 địa chỉ đối với các block dữ liệu, cũng như các địa chỉ của một block vô hướng của bậc đầu tiên, bậc 2 và bậc 3. Nếu một trong các block chứa đựng 256 địa chỉ ổ đĩa, thì tệp tin có thể sử dụng lớn nhất là bao nhiêu ? Với kích cỡ một block của ổ đĩa là 1 KB.

Bài tập 4.11. Về quản lý việc lưu trữ các tệp tin

Việc lưu trữ các tệp tin dẫn tới việc phân mảnh nhỏ ổ đĩa từ. Bạn hãy trình bày về việc phân mảnh nhỏ các vòng xuyến ở ngoài và ở trong gần tâm của đĩa từ ?