Review

Objectives:

- Bố cục hệ thống tệp

- Triển khai tệp

- Thư mục triển khai

- Những tệp đã chia sẻ

- Nhật ký hệ thống tệp có cấu trúc

- Hệ thống nhật ký tệp

- Hệ thống tệp ảo

File system implementation:

- Làm thế nào mà các tệp và thư mục được lưu trữ?

- Làm thế nào mà ổ đĩa được quản lý?

- Làm thế nào mà mọi thứ có thể hoạt động trơn tru và mượt mà?

File system layout (slide 4 và 5)

Hệ thống tệp được lưu trữ trên đĩa.

Đĩa có thể được chia thành một hoặc nhiều phân vùng, với các hệ thống tệp độc lập trên mỗi phân vùng

- Khối đầu tiên trong mỗi phân vùng được gọi là khối khởi động tồn tại ngay cả khi phân vùng không chứa HĐH có khả năng khởi động

- Khối thứ 2 là khối siêu:

+ Chứa tất cả các tham số chính về hệ thống tệp bao gồm số ma thuật (kiểu hệ thống tệp), số khối trong hệ thống tệp và các thông tin quản trị quan trọng khác

+ Được đọc vào bộ nhớ khi máy tính được khởi động hoặc lần đầu tiên mở vào hệ thống tệp

Khối tiếp theo là các khối tự do (ví dụ: dạng bitmap hoặc danh sách con trỏ)

Khối sau là i-node, một cấu trúc dữ liệu mảng, một trên mỗi tệp, cho biết tất cả về tệp

Sau đó có thể đến thư mục gốc, chứa phần trên cùng của cây hệ thống tệp

Cuối cùng, phần còn lại chứa tất cả các thư mục và tệp khác

Sector 0 của đĩa được gọi là Master Boot Record (MBR) được sử dụng để khởi động máy tính.

Phần cuối của MBR chứa bảng phân vùng:

+ Cung cấp địa chỉ bắt đầu và địa chỉ kết thúc của mỗi phân vùng

+ Một trong các phân vùng trong bảng được đánh dấu là đang hoạt động

Cơ chế khởi động máy tính:

+ Máy tính được khởi động, BIOS đọc và thực thi MBR

+ MBR định vị phân vùng hoạt động, đọc trong khối khởi động của nó và thực thi nó

+ Chương trình trong khối khởi động tải hệ điều hành có trong phân vùng đó

(Hình mô phỏng cấu trúc hệ thống tệp máy tính)

Các cách triển khai tệp (slide 7,

Phân bổ liên tục: Lưu trữ từng tệp dưới dạng một loạt khối đĩa liền nhau

Thuận lợi:

+ Đơn giản để thực hiện

+ Theo dõi nhu cầu khối của tệp địa chỉ đĩa của khối đầu tiên số lượng khối được phân bổ

+ Hiệu suất cao: lấy toàn bộ tệp từ đĩa trong một thao tác (chỉ tìm kiếm khối đầu tiên)

Nhược điểm: Phân mảnh (bên trong và bên ngoài) và Khó tăng kích thước tệp .

(Hình mô phỏng Phân bổ liên tục slide 8)

Phân bổ danh sách được liên kết:

+ Lưu trữ tệp là giữ mỗi tệp dưới dạng danh sách liên kết của các khối đĩa

+ Từ đầu tiên của mỗi khối được sử dụng làm con trỏ đến khối tiếp theo và phần còn lại của khối là dữ liệu

Thuận lợi:

+ Không có phân mảnh bên ngoài (vẫn phân mảnh bên trong)

+ Đủ để mục nhập thư mục chỉ lưu trữ địa chỉ đĩa của khối đầu tiên

Nhược điểm: Truy cập ngẫu nhiên rất chậm, Dung lượng lưu trữ dữ liệu trong một khối không phải là lũy thừa của hai vì con trỏ chiếm một vài byte.

(Hình 4.11 Mô phỏng danh sách được liên kết)

Phân bổ danh sách được liên kết bằng cách sử dụng một bảng trong bộ nhớ:  
+ Cả hai nhược điểm của phân bổ danh sách liên kết có thể được loại bỏ bằng cách lấy từ con trỏ từ mỗi khối đĩa và đặt nó vào một bảng trong bộ nhớ (Bảng phân bổ tệp - FAT)

Thuận lợi:

+ Các lợi ích tương tự như phân bổ danh sách liên kết

+ Toàn bộ khối có sẵn cho dữ liệu

+ Truy cập ngẫu nhiên tương đối nhanh

Nhược điểm

+ Toàn bộ bảng phải được giữ lại để có hiệu quả trong bộ nhớ

Ví dụ: đối với ổ cứng HDD 200GB, Kích thước khối 1KB, 4 byte cho mục nhập  200 triệu mục nhập  600MB - 800MB bộ nhớ dành cho bảng phân bổ FAT.

+ Không thực tế và không sử dụng quy mô tốt cho các ổ đĩa lớn.

(Hình mô phỏng)

inodes (hết slide 15)

Là cặp của các nút chỉ mục

Một cấu trúc dữ liệu (bao gồm 2 phần) liên kết với mọi tệp

+ Phần 1: Liệt kê các thuộc tính

+ Phần 2: Địa chỉ đĩa của khối tệp. Nó chia tiếp ra 2 phần phụ

~ Phần phụ 1: chứa trực tiếp địa chỉ của khối đĩa thứ i

~ Phần phụ 2: chứa địa chỉ của khối con trỏ được sử dụng với tệp lớn (chỉ mục đa cấp)

Với nút i có sẵn, có thể tìm thấy tất cả các khối của tệp.

Thuận lợi:

+ Không gian đĩa được để dành riêng vì i-node chỉ cần có trong bộ nhớ khi tệp tương ứng được mở

+ Cho phép số lượng tệp tối đa có thể được mở cùng một lúc

+ Nhỏ hơn không gian bị chiếm bởi bảng tệp

Nhược điểm:

+ Mỗi nút i chỉ có chỗ cho một số lượng địa chỉ đĩa cố định (không linh hoạt) → để dành đĩa cuối cùng không phải cho khối dữ liệu, nhưng nó được sử dụng để chứa nhiều địa chỉ khối đĩa hơn.

Ví dụ: Một i-node chứa 10 địa chỉ trực tiếp và 1 địa chỉ gián tiếp duy nhất, mỗi địa chỉ 4 byte và tất cả các khối đĩa là 1KB. Tệp lớn nhất có thể là bao nhiêu?

10 địa chỉ trực tiếp với 1KB mỗi khối: 10 \* 1KB = 10KB

1 địa chỉ gián tiếp duy nhất

Kích thước 1KB = 1024 byte

Con trỏ là 4 byte → 1024/4 = 256 con trỏ có thể trỏ địa chỉ khối 256 với 1K

→ 256 \* 1KB = 256KB

→ Tệp lớn nhất có thể là 10 + 256 = 266 KB

Implementing Directories (slide 16)

Chức năng chính của hệ thống thư mục là ánh xạ tên ASCII của tệp thành thông tin cần thiết để định vị dữ liệu của tệp đó.

Khi một tệp được mở, Hệ điều hành sử dụng tên đường dẫn để định vị mục nhập thư mục chứa:

+ Thông tin cần thiết để tìm đĩa bị chặn, chẳng hạn như địa chỉ đĩa của toàn bộ tệp, hoặc số khối đầu tiên hoặc số nút i

+ Một số thuộc tính tệp

Trong thiết kế đơn giản, một thư mục bao gồm:

+ Danh sách các mục nhập có kích thước cố định, mỗi mục nhập một tệp, chứa tên tệp (độ dài cố định), cấu trúc thuộc tính tệp và một hoặc nhiều địa chỉ đĩa

+ Hoặc chỉ một tên tệp và số nút i

(slide 18)

Tên tệp dài và có độ dài thay đổi

- Cách tiếp cận đầu tiên (đơn giản nhất):

+ Đặt giới hạn về độ dài tên tệp, thường là 255 ký tự

+ Nhược điểm: lãng phí rất nhiều không gian thư mục

- Cách tiếp cận thứ hai: Tất cả các mục nhập thư mục có cùng kích thước chứa

+ Bắt đầu với độ dài của mục nhập

+ Theo dõi với các thuộc tính tệp

+ Kéo dài là tên tệp thực tế có thể dài (mỗi tên tệp được kết thúc bằng ký tự đặc biệt)

+ Để cho phép mỗi mục nhập thư mục bắt đầu trên ranh giới từ, mỗi tên tệp được điền vào một số từ tích hợp

+ Nhược điểm:

> Khi một tệp bị xóa, một khoảng trống có kích thước thay đổi được đưa vào thư mục mà tệp tiếp theo được nhập vào có thể không vừa → phân mảnh

> Một mục nhập thư mục có thể kéo dài nhiều trang → lỗi trang xảy ra khi đọc tên tệp

(slide 20)

Tên tệp dài và có độ dài thay đổi (tiếp tục)

- Cách tiếp cận thứ ba (Cải tiến)

+ Tự tạo các mục trong thư mục có độ dài cố định và giữ các tên tệp cùng nhau trong một nhóm ở cuối thư mục

- Thuận lợi

+ Khi một mục nhập bị xóa, tệp tiếp theo được nhập sẽ luôn phù hợp ở đó

+ Không có ký tự điền sau tên tệp

(slide 21)

Tìm kiếm thư mục

- Tìm kiếm tuyến tính từ đầu đến cuối khi tên tệp phải được tra cứu → chậm với các thư mục cực dài

- Để tăng tốc, bảng hash được sử dụng:

+ Ưu điểm: tra cứu nhanh hơn

+ Nhược điểm: quản trị phức tạp hơn

Áp dụng cho các thư mục lớn, kết quả tìm kiếm được lưu vào bộ nhớ đệm:

- Trước khi bắt đầu tìm kiếm, việc kiểm tra đầu tiên được thực hiện trong bộ nhớ cache

- Nếu vậy, nó có thể được định vị ngay lập tức

- Nếu không, việc tìm kiếm sẽ được bắt đầu

(slide 22) Shared files

Chia sẻ tệp rất hữu ích khi làm việc cùng nhau trong một dự án.

Một tệp được chia sẻ xuất hiện đồng thời trong các thư mục khác nhau thuộc về những người dùng khác nhau.

Các vấn đề:

- Việc thay đổi tệp được chia sẻ được liệt kê trong thư mục của người dùng đang thực hiện (không hiển thị với những người dùng khác)

Giải pháp đầu tiên (truyền thống)

- Các khối đĩa không được liệt kê trong thư mục, nhưng trong một cấu trúc dữ liệu nhỏ (i-node) được liên kết với chính tệp. Các thư mục sau đó sẽ chỉ đến cấu trúc dữ liệu nhỏ

Thuận lợi:

- Một nút i

- Không có khối mới được phân bổ

Nhược điểm:

- Có thể dẫn đến sự không nhất quán

- 1 người dùng có thể bị lập hóa đơn cho "tệp đã xóa"

(slide 26) Log- structured File Systems (LFS)

Vấn đề:

Trong hầu hết các hệ thống tệp, việc ghi được thực hiện là những phần rất nhỏ (rất kém hiệu quả)

Trong UNIX:

- Để ghi tệp mới, tất cả phải ghi nút i cho thư mục, khối thư mục, nút i cho tệp và bản thân tệp.

- Nếu một sự cố xảy ra trước khi quá trình ghi hoàn tất, hệ thống tệp có vấn đề nghiêm trọng về tính nhất quán. Do đó, tất cả các tác vụ ghi có thể được trì hoãn.

(slide 27)

Giải pháp: sử dụng LFS

- Cấu trúc toàn bộ đĩa như một bản ghi

- Tất cả các lần ghi ban đầu được lưu vào bộ nhớ đệm và theo định kỳ, tất cả các lần ghi được ghi vào bộ đệm vào đĩa trong một phân đoạn duy nhất ở cuối nhật ký

- Một phân đoạn duy nhất:

+ Chứa i-node, khối thư mục và khối dữ liệu, tất cả được trộn với nhau

+ Ở đầu mỗi phân đoạn là một bản tóm tắt phân đoạn cho biết những gì có thể tìm thấy

- Các nút i nằm rải rác trên toàn bộ nhật ký (các khối được đặt theo cách thông thường khi một nút i được đặt)

- Tìm nút i khó hơn nhiều, vì địa chỉ của nó không thể đơn giản tính từ số i của nó;

+ Do đó, một bản đồ i-node, được lập chỉ mục bởi i-number được duy trì

+ Mục nhập i trong bản đồ này trỏ đến nút i trên đĩa

+ Bản đồ được lưu trên đĩa, nhưng nó cũng được lưu vào bộ nhớ đệm → những phần được sử dụng nhiều nhất sẽ nằm trong bộ nhớ hầu hết thời gian

- Mở tệp bao gồm việc sử dụng bản đồ để xác định vị trí nút i cho tệp. Khi nút i đã được định vị, địa chỉ của các khối có thể được tìm thấy từ nó. Tất cả các khối sẽ tự nằm trong các phân đoạn ở đâu đó trong nhật ký.

(5 cái hình)

(slide 33)

Các vấn đề:

Các đĩa có dung lượng rất lớn, do đó khi nhật ký chiếm toàn bộ đĩa, không có phân đoạn mới nào có thể được ghi.

Nhiều phân đoạn hiện tại có thể có các khối không còn cần thiết nhưng chúng vẫn chiếm không gian là các phân đoạn đã được viết trước đó.

Các giải pháp:

- Chuỗi Cleaner dành thời gian quét nhật ký theo hình tròn để thu gọn nó

+ Nó bắt đầu bằng cách đọc tóm tắt của phân đoạn đầu tiên trong nhật ký để xem i-node và tệp nào ở đó

+ Sau đó, nó sẽ kiểm tra bản đồ i-node hiện tại để xem liệu i-node có còn hiện tại hay không và các khối tệp vẫn đang được sử dụng → được viết trong phân đoạn tiếp theo. Nếu không, thông tin đó sẽ bị loại bỏ

+ Cleaner di chuyển dọc theo nhật ký, xóa các phân đoạn cũ ở phía sau và đưa bất kỳ dữ liệu trực tiếp nào vào bộ nhớ để viết lại trong phân đoạn tiếp theo

→ đĩa là một bộ đệm tròn lớn, với luồng người viết thêm phân đoạn mới vào phía trước và luồng sạch hơn loại bỏ các phần cũ ở phía sau

→ phức tạp (theo lý thuyết)

(slide 34) Journaling File System

Giữ nhật ký về những gì hệ thống tệp sẽ làm trước khi nó thực hiện:

- Nếu hệ thống gặp sự cố trước khi nó có thể thực hiện công việc đã định

- Khi khởi động lại hệ thống có thể xem nhật ký để xem điều đó đang diễn ra tại thời điểm xảy ra sự cố và hoàn thành công việc

Đã áp dụng hệ thống tệp NTFS và Linux ext3

Các thao tác cần thiết để xóa tệp trong UNIX:

+ Xóa tệp khỏi thư mục của nó.

+ Giải phóng nút i vào nhóm các nút i miễn phí.

+ Trả lại tất cả các khối đĩa vào nhóm các khối đĩa trống.

Các vấn đề:

+ Bước đầu tiên đã hoàn thành và hệ thống gặp sự cố ( Các khối tệp và nút i sẽ không thể truy cập được từ bất kỳ tệp nào, nhưng cũng sẽ không có sẵn để gán lại)

+ Bước thứ hai đã hoàn thành và hệ thống gặp sự cố (Chỉ các khối bị mất)

+ Nếu thứ tự hoạt động bị thay đổi:

\*Nếu i-node được phát hành trước, thì sau khi khởi động lại, i-node có thể được chỉ định lại, nhưng mục nhập thư mục cũ sẽ tiếp tục trỏ đến nó, do đó đến tệp sai

\*Nếu các khối được phát hành trước, thì sẽ xảy ra sự cố trước khi i-node bị xóa (có nghĩa là mục nhập thư mục hợp lệ trỏ đến khối danh sách i-node bây giờ là nhóm lưu trữ miễn phí)

(slide 35) JFS

- Đầu tiên, viết một mục nhật ký liệt kê ba hành động cần hoàn thành

- Sau đó, mục nhập nhật ký được ghi vào đĩa

- Chỉ sau khi mục nhập nhật ký đã được viết, các hoạt động khác nhau mới bắt đầu

- Sau khi hoàn tất thành công, mục nhật ký sẽ bị xóa

- Do đó, nếu hệ thống bây giờ bị lỗi, khi khôi phục hệ thống tệp có thể kiểm tra nhật ký để xem có bất kỳ hoạt động nào đang chờ xử lý hay không → tất cả chúng có thể được chạy lại cho đến khi tệp được gỡ bỏ chính xác

- Yêu cầu:

+ Các hoạt động đã ghi phải là không cần thiết (chúng có thể được lặp lại thường xuyên nếu cần mà không gây hại)

+ Việc khôi phục sự cố có thể được thực hiện và an toàn

- Thực tế, JFS được áp dụng cho khái niệm DB với giao dịch nguyên tử

(2 hình gì đó)

(slide 38)Virtual File Systems

Các vấn đề

Nhiều hệ thống tệp khác nhau được sử dụng trên cùng một máy tính, thậm chí cùng một hệ điều hành.

Ví dụ: Windows Vista, XP được sử dụng NTFS, FAT32, FAT 16

Giải pháp là Hệ thống tệp ảo (Kleiman, 1986 - Sun Microsystems)

Cố gắng tích hợp nhiều hệ thống tệp vào một cấu trúc có trật tự

Ý tưởng chính là trừu tượng hóa rằng:

+ Một phần của hệ thống tệp là chung cho tất cả hệ thống tệp

+ Đặt mã đó trong một lớp riêng biệt gọi các hệ thống tệp cụ thể bên dưới để quản lý dữ liệu thực tế VFS là giao diện phía trên cho các quy trình của người dùng và giao diện phía dưới cho hệ thống tệp cụ thể

Tất cả các lệnh gọi hệ thống (đến từ các quy trình của người dùng) liên quan đến tệp đều được chuyển hướng đến VFS để xử lý ban đầu

Không biết hoặc không quan tâm nơi dữ liệu được lưu trữ hoặc hệ thống tệp cơ bản như thế nào Được thực hiện về cơ bản là hướng đối tượng

Hỗ trợ các loại đối tượng chính như siêu khối, v-node, thư mục, bảng gắn kết, một mảng các bộ mô tả tệp

Next slide