File System - Hệ thống quản lý tập tin.

Trong máy tình, file system hoặc filesystem được sử dụng để kiểm soát dữ liệu, lưu trữ và lấy lại. Nếu không có file system, thông tin được lưu trong nhưng khối lớn sẽ không có cách nào để tìm thấy vị trí bắt đầu và vị trí kết thúc, vị trí tiếp theo của dữ liệu. Bởi dữ liệu được chia thành các mảnh và mỗi mảnh được đặt tên, vì thế thống tin dễ dàng được đánh dấu và xác thực. File system được bắt nguồn từ hệ thống lưu trữ trên giấy, nhóm các dữ liệu được gọi là “file”. Cấu trúc và luật logic được sử dụng để quản lý các nhóm thống tin và chúng được gọi là “file system”.

Có rất nhiều loại file system. Mỗi loại đều có cấu trúc và logic khác nhau, tính chất về tốc độ, tính linh hoạt, bảo mật, size và .. 1 số file system được thiết kể để sử dụng cho 1 số ứng dụng đặc biệt. Ví dụ, ISO 9660 file system được thiết đặc biệt cho đĩa quang.

File system có thể sử dụng nhiều trên loại thiết bị lưu trữ và các loại phương tiên truyền thống khác nhau. Nội bật và thông thương nhất là ổ đĩa cứng. Bên cạnh đó là bộ nhợ flash, đĩa quang. Trong 1 số trường hợp, như tmpfs, computer main memory (RAM) sử dụng cả file đệm, sử dụng trong thời gian ngắn.

1 số file system được sử dụng cho local data storage devices, bên cạnh đó cung cấp truy chế truy cập file thông qua giao thức mạng (NFS, SMB, 9P). Một số file system là ảo, có nghĩa cung cấp “file” ảo sử dụng cho những yêu cấu tính toán hoặc ánh xạ vào nhưng file system khác. File system quản lý truy cập trên cả nội dung của file và metadata của nhưng file này. Nó chịu trách nhiệp sắp xếp không gian lưu trữ đảm bảo, tin cậy, rõ ràng, có hệ thống.

Kiến trúc:

File system bao gồm 2 hoặc 3 lớp. Đôi khi các lớp được chia rõ ràng, đội lúc được kết hợp lại.

The “logical file system” chịu trách nhiệm tương tác với ứng dụng người dùng. Nó cung cấp API cho các hoạt động cơ bản – Mở, đóng, đọc, .. và truyền các hoạt động yêu cầu xuống lớp dưới cho việc xử lý. “Logical file system” quản lý việc mở các đối tượng file table và per-process file descriptors. Lớp này cung cấp “file access, directory operations, [and] security and protection”.

Lớp thứ 2 (không bắt buộc) là virtual file system. “Giao diện cho phép hỗ trợ đồng thời nhiều loại file system vật lý, còn được gọi là thực thi file system”.

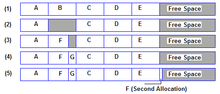
Lớp thứ 3 là physical file system. Đây là lớp liên quan đến hoạt động vật lý của thiết bị lưu trữ (disk). Nó xử lý các khối vật lý cho việc đọc hoặc ghi. Nó xử lý các buffer, memory management và chịu trách nhiệm bố trí các khối vật lý trong những ví trí được chỉ định. Physical file system tương tác với device drivers hoặc các kênh tới thiết bị lưu trữ.

Các khía cạnh của file system (Chỉ áp dụng với storage devices)

File system chỉ định, tạo các không gian lưu trữ chi tiết. File system chịu trách nhiệm tổ chức file và directories, chỉ định các phân vùng để lưu trữ hay không lưu trữ.

Kết quả của việc dữ thừa các không gian lưu trữ mà không thể tận dụng được gọi là “slack space”. VD, 512 byte được cấp phát, trung bình 256 b không được sử dụng, hay 64KB cluster thì trung bình 32KB sẽ không được sử dụng. Độ lớn cấp phát chi unit dược chọn khi file system được tạo. Việc chọn độ lớn cấp phát phụ thuộc vào độ lớn trung bình mà file cần cho việc lưu trữ. Thông thướng sẽ cấp phát hợp lý nhất cho việc lưu trữ.

Khái niệm phân mảnh xảy ra khi các không gian không lữu trữ hoặc các file đơn không tiếp giáp. Việc này xuất hiện trong quá trình sử dụng, tạo, sửa, xóa file. Khi file được tạo mới cần không gian lớn hơn các phân mảnh hiện có(hợp lý nhất cho việc lưu trữ). Các file được xóa nhưng không gian đang sử dụng lại không được cấp phát cho việc lưu trữ



Filenames

Filename được sử dụng để xác thực vị trí lưu trữ trong file system. Hầu hết file system có hạn chế về độ dài, quy tắc đặt tên filenames.

Directories

Filesystem còn có dạng directory, cho phép user nhóm các file riêng lẻ thành 1 tập hợp. Có thể được thực hiện bằng cách gán file với với số thứ tự trong table of content hoặc inode trong hệ điều hành nhân Unix. Directory có thể cho phép tổ chức dạng phân cấp chứ nhiều file và direc bên trong.

Metadata

Thông tin đi kèm với mỗi file system. Như độ dài của dữ liệu file được lưu với số block được phân phát hoặc dung lượng. Thời gian chỉnh sửa cuối cùng. Thời gian tạo, khối, character, socket, .. Chủ sở hữu userID và group ID, quyển truy cập

Multiple file systems within a single system

Thông thường, nhà phân phối thương cấu hình với 1 file system duy nhất trên toàn thiết bị lưu trữ.

Bên canh phương pháp chia ổ, 1 số file system với thuộc tính khác nhau có thể cùng được sử dụng. VD, browser cache, được cấu hình để lưu trong nhưng bộ đệm cấp phát nhỏ, cho phép xóa, tạo mới liên tục mà không ảnh hưởng đến hệ thống lưu trữ.

Bên cạnh đó là 1 số hệ thống đám mây, sử dụng "disk images" sử dụng file system khác, với cùng thuộc tính hoặc khác thuộc tính. Ví dụ dễ thấy nhất là ảo hóa, user có thể chạy định dạng ext4 của Linux trên máy ảo, mà máy ảo đó được lưu trữ trên định dạng NTFS windows. Ext4 file sustem được định dạng lại disk image, mà disk image được lưu trên NTFS host.

Types of file systems

Disk file systems

Examples include FAT (FAT12, FAT16, FAT32), exFAT, NTFS, HFS and HFS+, HPFS, APFS, UFS, ext2, ext3, ext4, XFS, btrfs, ISO 9660, Files-11, Veritas File System, VMFS, ZFS, ReiserFS and UDF. Some disk file systems are journaling file systems or versioning file systems.

Database file systems

Another concept for file management is the idea of a database-based file system. Instead of, or in addition to, hierarchical structured management, files are identified by their characteristics, like type of file, topic, author, or similar rich metadata.[11]

IBM DB2 for i [12] (formerly known as DB2/400 and DB2 for i5/OS) is a database file system as part of the object based IBM i [13] operating system (formerly known as OS/400 and i5/OS), incorporating a single level store and running on IBM Power Systems (formerly known as AS/400 and iSeries), designed by Frank G. Soltis IBM's former chief scientist for IBM i. Around 1978 to 1988 Frank G. Soltis and his team at IBM Rochester have successfully designed and applied technologies like the database file system where others like Microsoft later failed to accomplish.[14] These technologies are informally known as 'Fortress Rochester'[citation needed] and were in few basic aspects extended from early Mainframe technologies but in many ways more advanced from a technological perspective[citation needed].

Chuyển đổi file system

Cho phép chuyển đổi định dạng từ loại filesystem a thành 1 loại filesystem khác, cùng hoặc khác thuộc tính.

Network file systems

A network file system is a file system that acts as a client for a remote file access protocol, providing access to files on a server. Programs using local interfaces can transparently create, manage and access hierarchical directories and files in remote network-connected computers. Examples of network file systems include clients for the NFS, AFS, SMB protocols, and file-system-like clients for FTP and WebDAV.

Shared disk file systems

A shared disk file system is one in which a number of machines (usually servers) all have access to the same external disk subsystem (usually a SAN). The file system arbitrates access to that subsystem, preventing write collisions. Examples include GFS2 from Red Hat, GPFS from IBM, SFS from DataPlow, CXFS from SGI and StorNext from Quantum Corporation.

Unix and Unix-like operating systems

Linux

Linux supports numerous file systems, but common choices for the system disk on a block device include the ext\* family (ext2, ext3 and ext4), XFS, JFS, ReiserFS and btrfs. For raw flash without a flash translation layer (FTL) or Memory Technology Device (MTD), there are UBIFS, JFFS2 and YAFFS, among others. SquashFS is a common compressed read-only file system.

Solaris

Solaris in earlier releases defaulted to (non-journaled or non-logging) UFS for bootable and supplementary file systems. Solaris defaulted to, supported, and extended UFS.

Support for other file systems and significant enhancements were added over time, including Veritas Software Corp. (Journaling) VxFS, Sun Microsystems (Clustering) QFS, Sun Microsystems (Journaling) UFS, and Sun Microsystems (open source, poolable, 128 bit compressible, and error-correcting) ZFS.

Kernel extensions were added to Solaris to allow for bootable Veritas VxFS operation. Logging or Journaling was added to UFS in Sun's Solaris 7. Releases of Solaris 10, Solaris Express, OpenSolaris, and other open source variants of the Solaris operating system later supported bootable ZFS.

Logical Volume Management allows for spanning a file system across multiple devices for the purpose of adding redundancy, capacity, and/or throughput. Legacy environments in Solaris may use Solaris Volume Manager (formerly known as Solstice DiskSuite). Multiple operating systems (including Solaris) may use Veritas Volume Manager. Modern Solaris based operating systems eclipse the need for Volume Management through leveraging virtual storage pools in ZFS.

VÀI KHÁI NIỆM TRONG FILE SYSTEM

1. File và inode

Một file là một khối dữ liệu được lưu trữ liên tục hoặc không liên tục (tình trạng dữ liệu bị phân mảnh) trên thiết bị lưu trữ như ổ cứng, ổ mềm, ổ flash,… Người dùng bình thường sẽ nhận dạng một file dựa trên tên của nó (file name).

Mỗi file được liên kết với một inode mà chứa các thuộc tính của file đó như là định dạng (text, binary,…), kích thước, ngày khởi tạo, vị trí trên thiết bị lưu trữ, chủ sở hữu, quyền truy cập,… Thông tin về file mà inode nắm giữ thường được gọi là metadata, đặc biệt inode không chứa tên file và nội dung thật sự của file.

Mỗi inode được xác định bởi một con số (inode number) , có một bảng chỉ mục (inode table) gồm inode number – vị trí inode trên thiết bị lưu trữ. Với inode number có được kernel sẽ tìm trong bảng chỉ mục này và truy cập tới nội dung của inode, bao gồm con trỏ dữ liệu từ đó truy cập tới nội dung của file mà liên kết với inode đó.

Khi một file được tạo, nó được gán một file name + một inode number là số nguyên duy nhất trong một file system. Các file name và inode number tương ứng được lưu trữ thành các mục (entry) trong thư mục, tức là thư mục thực ra chỉ là danh sách các liên kết giữa file name và inode number mà thôi.

Khi người dùng hoặc chương trình sử dụng file name để tham khảo tới một file, hệ điều hành sẽ sử dụng tên này để tìm kiếm inode tương ứng bằng cách tra cứu trong inode table rồi từ đó biết được thông tin và vị trí của file trên thiết bị lưu trữ để phục vụ cho các thao tác về sau (như chỉnh sửa nội dung của file, cung cấp thuộc tính của file,…).

Sử dụng lệnh ls -i để biết inode number của 1 file, và ls -l để biết metadata của file chứa trong inode.

Trên nhiều kiểu file system thì số lượng các inode có thể sử dụng được cố định tại thời điểm khởi tạo file system, dẫn tới việc giới hạn số lượng file mà hệ thống file có thể nắm giữ, quản lý.

2. Directory

Directory (hay folder – thư mục) trong \*nix là một loại file đặc biệt, chứa danh sách các liên kết giữa tên đối tượng (file, folder, soft/hard link…) và inode number tương ứng với đối tượng đó.

Current Directory

Hay working directory (thư mục hiện hành) là thư mục mà hiện tại người dùng, chương trình đang “đứng”, tham khảo, làm việc tại đó.

Mọi user luôn luôn đang làm việc bên trong một thư mục nào đó. Chính xác hơn thì mỗi tiến trình (process) có một WK được liên kết linh động với nó. Trong Windows thì process gọi hàm GetCurrentDirectory để xác định vị trí của WK và gọi hàm SetCurrentDirectory để thay đổi WK.

Khi user hoặc process chỉ định một file mà đơn giản chỉ sử dụng tên file hoặc đường dẫn tương đối (relative path) của file thì việc tìm kiếm tới file này bắt đầu từ WK. Ví dụ trong Linux, khi mới mở Shell CLI lên để gõ lệnh thường thì ta đang đứng ở thư mục có đường dẫn tuyệt đối là

/home/user\_name thì thư mục user\_name là WK. Khi gõ vào

# rm foo.txt

thì lệnh này sẽ thực hiện việc xóa file foo.txt trong thư mục chủ của người dùng.

Thường có 2 cách để thể xác định WK.

Một là nhìn vào dấu nhắc lệnh (command prompt). Với bash thì dấu nhắc lệnh chứa tên người dùng, máy tính và thư mục hiện hành, ví dụ:

[uit@localhost bluesky]#

thì uit là user name, localhost là computer name và bluesky là working directory, # chỉ ra rằng đây là user root.

Cách thứ hai là sử dụng lệnh pwd (present working directory), lệnh này không có tùy chọn hay đối số để hiện đường dẫn đầy đủ của thư mục hiện hành.

WK thường được biểu thị bởi 1 dấu chấm “.” , còn 2 dấu chấm liên tiếp “..” thay cho thư mục cha của WK. Ta sẽ luôn thấy 2 mục bị ẩn này tồn tại trong mọi thư mục trên \*nix bằng cách sử dụng lệnh ls -a

Directory Tree

Gọi là cây thư mục, là hệ thống các thư mục được phân cấp và trong đó có duy nhất một thư mục được gọi là thư mục cha và tất cả các mức độ thư mục con của nó. Bất kỳ thư mục nào cũng có thể là điểm bắt đầu cho 1 cây thư mục của riêng nó nếu nó chứa ít nhất 1 thư mục con.

Hầu hết các hệ điều hành ngày nay đều sử dụng cấu trúc cây thư mục cho việc tổ chức file. Với các hệ \*nix chỉ có duy nhất một thư mục gốc (root directory, ký hiệu là /) mà từ đó các cây thư mục khác sinh ra từ đây. Con các hệ Microsoft Windows thì có nhiều thư mục gốc độc lập với nhau có các tên như C: , D: , E: ,…

Lệnh du (disk usage) trong Linux là một tiện ích thu thập thông tin về các cây thư mục, bao gồm tổng không gian đĩa mà một cây chiếm dụng, tên và kích thước mỗi nhánh hoặc file trong cây đó.

Hard Link

Hard link (HL) là tên gọi khác cho một file đang tồn tại trên file system.

Một file có thể có nhiều HL và mỗi HL cũng có thể có nhiều HL cho nó. Tuy nhiên, không thể tạo cho HL cho thư mục và cũng không thể tạo ra HL cho một file không nằm cùng phân vùng với HL. Ví dụ, không thể tạo HL trên phân vùng A cho một file nằm trên phân vùng B.

Hệ điều hành không phân biệt giữa file name ban đầu của file với các HL được tạo ra sau này cho file đó. Cả file name và các HL này đều trỏ tới cùng một inode, và vì mỗi inode có một số inode number là duy nhất trong một file system nên HL không thể làm việc chéo qua các phân vùng khác nhau.

Sử dụng lệnh ln để tạo HL. Ví dụ dưới đây sẽ tạo một HL có tên hlink1 cho một file có tên file1, cả 2 file này đều nằm trong cùng thư mục hiện hành.

$ ln file1 hlink1

File name ban đầu và tất cả các HL tới file đều chia sẻ chung inode, có thể thấy rõ điều này bằng cách sử dụng lệnh ls –i. Câu lệnh dưới đây sẽ cho thấy inode number của file1 và hlink1 là giống nhau:

$ ls -i file1 hlink1

Số lượng các HL cho 1 file được thể hiện trong cột thứ 2 trong output của lệnh ls –l. Con số này là tổng của filename ban đầu và HL và số này giống nhau cho target file và mỗi HL, ví dụ:

$ ls -l file1 hlink1

Sử dụng lệnh sau để tìm các file có nhiều hơn một HL

$ find -type f -links +1

Thử chỉnh sửa nội dung của target file và lưu lại. Sau đó mở HL của target file đó lên sẽ những thay đổi được giữ nguyên.

Lệnh rm thực sự làm gì?

Khi sử dụng lệnh rm để xóa file thì thực chất là làm giảm đi một HL. Khi số lượng HL giảm còn 0 thì không thể truy cập tới nội dung của file được nữa (mặc dù nội dung đó vẫn tồn tại trên thiết bị lưu trữ) vì hệ điều hành không còn cách nào để tham khảo tới file này. Dữ liệu của file chỉ thực sự bị mất khi vị trí của nó bị ghi đè bởi các file mới. Điều này giải thích tại sao ta vẫn có thể khôi phục dữ liệu vừa bị xóa và không có dữ liệu nào được tạo ra trên vị trí của dữ liệu cũ.

HL cũng như shortcut trong Windows là cho phép truy cập tới file, chương trình, script từ một vị trí khác thuận tiện hơn.

Symbolic link

SL thì hơi khác chút so với HL đó là có thể tạo SL cho 1 thư mục cũng như là tạo SL các file và thư mục trên các phân vùng khác. Tuy nhiên, hạn chế của SL so với HL là khi xóa target file thì SL không có tác dụng nữa. Sử dụng cú pháp lệnh sau để tạo SL

$ ln -s target\_file SL\_name

e