文件系统

概念：是对磁盘的一种抽象。而磁盘主要是为了长期存储信息

- 能够存储大量信息
- 使用信息的进程终止时，信息仍旧存在
- 必须能够使多个进程并发访问有关信息

文件

概念：文件是一种抽象机制，它提供了一种在磁盘上保存信息而且方便以后读取的方法。或方法可以使用户不必了解存储信息的方法、位置和实际磁盘工作方式等有关细节

文件命名

- 所有现代操作系统都允许用1-8个字母组成的字符串作为合法的文件名
- 通常，文件名中也允许有数字和一些特殊字符
- 有些文件系统支持长达255个字符的文件名
- 有些文件系统区分大小写字母，有些则不区分

文件系统分类

- FAT-16(MS-DOS的文件系统)
- FAT-32(Windows 98 对FAT-16的扩展)
- NTFS(更先进的本地文件系统)

扩展名

在某些UNIX系统中，文件扩展名只是一种约定，操作系统并不强迫采用它

Windows关注扩展名且对其赋予了含义。用户或进程可以在操作系统中注册扩展名，并且规定那个程序拥有该扩展名。例如：双击file.docx启动Microsoft Word程序

文件结构

字节序列。事实上操作系统不知道也不关系文件内容是什么。操作系统所见的就是字节，其文件内容的任何含义只在用户程序中解释(所有UNIX版本，包括Linux和OS X以及Windows都采用这种文件模型)

记录序列。文件是具有固定长度记录的序列。每个记录都有其内部结构

树。每个记录不必具有相同的长度，记录的固定位置上有一个键字段，以便树按键字段进行排序，从而可以对特定键进行快速查找

文件类型

普通文件：包含有用户信息的文件

- ASCII 文件：由多行正文组成。在某些系统中，每行用回车符结束，其他系统则用换行符结束
  - 优势：可以显示和打印，还可以用任何文本编辑器进行编辑。多程序输入和输出通过管道交互
- 二进制文件：打印出来的二进制文件无法理解，充满高乱字符的一张表。通常二进制文件有一定的内部结构。使用该文件的程序必须了解这种结构

目录：管理文件系统结构的系统文件

字符特殊文件：和输入/输出有关，用于串行I/O

类设备，如终端、打印机、网络等

块特殊文件 —— 用于磁盘类设备

文件的访问

顺序访问 —— 进程在这些系统中可从头按顺序读取文件的全部字节或记录，但不能跳过某一些内容。也不能不按顺序读取。在存储介质是磁带而不是磁盘时，顺序访问文件是很方便的。每次read操作都给出开始读文件的位置

随机访问文件 —— 当用磁盘来存储文件时，可以不按顺序地读取文件中的字节或记录，或者按照关键字而不是位置来访问记录。UNIX和Windows使用非访问模式。特殊的seek操作设置当前位置

文件属性：与文件相关的信息，如文件创建的日期和时间、文件大小等，有些人称为元数据

文件操作：文件的目的是存储信息并方便以后的检索

- create:创建不包含任何数据的文件
- delete:删除该文件以释放磁盘空间
- open: 把文件属性和磁盘地址表装入内存，便于后续调用的快速访问
- close: 关闭文件以及释放内部表空间
- read: 在文件中读取数据，一般读取的数据来自文件的当前位置。调用者必须指明需要读取多少数据，并且提供存放数据的缓存区
- write: 向文件写数据，写操作一般从文件当前位置开始。如果当前位置是文件末尾，文件长度增加。如果当前位置在文件中间，则现有数据被覆盖，并且永远丢失
- append: 只能在文件末尾添加数据
- seek: 对于随机访问文件，要指定从何处开始获取数据。seek系统调用把当前位置指针指向文件中特定位置
- get attributes: 进程运行常需要读取文件属性
- set attributes: 某些属性是可由用户设置的，设置是在文件创建之后
- rename: 用户常常要改变已有文件的名字

目录

一级目录：在一个目录中包含所有的文件。根目录

层次目录系统：目录树，用户可以创建任意数量的子目录

路径名：用目录树组织文件系统时，需要有某种方法指明文件名

- 每个文件都赋予一个绝对路径名，它由根目录到文件的路径组成
- 相对路径名，常和当前工作目录一起使用
- 每个进程都有自己的工作目录，这样在进程改变工作目录并退出后，其他进程不会受到影响，文件系统中也不会有改变的痕迹
  - 对于进程而言，切换工作目录是安全的。知道程将工作目录切换到唯目录读取/usr/lib中的文件，从而使用相对路径
  - 但是如果改变了真正程的工作目录，并且工作完毕之后没有修改回去，则其他程序将可能无法正常工作

目录操作

- create：创建目录。除了目录项“.”和“..”外，目录内容为空
- delete：删除目录。只有空目录可删除。只包含目录项“.”和“..”的目录被认为是空目录
- opendir：目录内容可被读取。为列出目录中全部文件，程序必须先打开该目录，然后读取全部文件的文件名
- closedir：关闭目录以释放内部表空间
- readdir：返回打开目录的下一个目录项
- rename：目录重命名
- link：链接技术允许在多个目录中出现同一个文件。这个系统调用指定一个存在的文件和一个路径名，并建立从该文件到路径名所指定的链接
- unlink：删除目录项。如果被解除连接的文件只出现在一个目录中，则将已从文件系统中删除。如果出现在多个目录中，则只删除指定路径名的连接，软件保留其他路径名的连接

目录操作

- 文件系统布局
  - 文件系统存放在磁盘上。多数磁盘划分为一个或多个分区，每个分区中有一个独立的文件系统
  - 磁盘的0号扇区称为主引导记录(MBR)，用来引导计算机。在MBR的结尾是分区表。该表给出了每个分区的起始和结束地址
  - 分区表中的一个分区被标记为活动分区。在计算机被引导时，BIOS读入并执行MBR。MBR做的第一件事是确定活动分区，读入它的第一个块，称为引导块(boot block)，并执行
  - 引导块中的程序将装载该分区的操作系统
  - 磁盘分区的布局
    - 随文件系统的不同而变化
    - 超级块，包含文件系统的所有关键参数，在计算机启动时，或者在该文件系统首次使用时，超级块会被读入内存。超级块中的典型信息包括：确定文件系统类型的魔数、文件系统中块的数量以及其他重要的管理信息
- 文件的实现
  - 目的：为了记录各个文件分别用到那些磁盘块
  - 1. 连续分配
    - 把每个文件作为一连串数据块存储在磁盘上
    - 优点
      - 实现简单，记录每个文件用到的磁盘块simple为只需记住两个数字即可，第一块的磁盘地址和文件块数
      - 读操作性能好
    - 缺点 —— 随着时间的推移，磁盘会变得零碎
  - 2. 链表分配
    - 为每个文件构造磁盘块链表
    - 每个块的第一个字作为指向下一块的指针，块的其他部分存放数据
    - 优点
      - 不会因为磁盘碎片而浪费存储空间
      - 只要存放第一块的磁盘地址，文件的其他块就可以从这个首地址查找到
    - 缺点
      - 随机访问相当缓慢
      - 由于指针占去了一些字节，每个磁盘块存储数据的字节数不再是2的整数次幂
  - 3. 采用内存中的表进行链表分配
    - 取出每个磁盘块的指针字，把它们放在内存的一个表中
    - 内存中的这样一个表格称为文件分配表(FAT)
    - 优点 —— 整个链都放在内存中，不需要任何磁盘引用
    - 缺点 —— 必须把整个表都存放在内存中
  - 4. i节点
    - 给每个文件赋予一个称为i节点的数据结构，其中列出了文件属性和文件块的磁盘地址
    - 优点 —— 只有在对应的文件打开时，其节点才在内存中
    - 缺点 —— 每个节点只能存储固定数量的磁盘地址，当文件所含的磁盘块的数目超过了i节点所能容纳的数目时，需要将最后一个磁盘地址指向一个包含额外磁盘块地址的块的地址
- 文件系统的实现
  - 读取文件必须打开文件，根据路径名找到对应的目录项。目录项中提供了查找文件磁盘块所需要的信息(根据文件的实现方式不同，存储的信息也不同)
  - 目录系统的主要功能是把ASCII文件名映射成定位文件数据所需的信息
- 目录的实现
  - 存放在目录项中
    - 目录中有一个固定大小的目录项列表，每个文件对应一项
    - 其中每个文件对应一项，其中包含一个固定长度的文件名、一个文件属性的结构体以及用以说明磁盘块位置的一个或多个磁盘地址
  - 对于采用i节点的系统，把文件属性存放在i节点中而不是目录项中
  - 文件属性在哪里存储
    - 可变长度的长文件名如何处理
      - 1. 给文件名一个固定的长度，如255个字符，但是浪费空间(并不是所有的文件都很长)
      - 2. 所有目录项大小一样，每个目录有一个固定部分，以目录项的长度开始，后面是固定格式的数据(所有者、创建时间、保护信息以及其他属性)，这个固定长度的头的后面是一个任意长度的供应文件名
      - 3. 使目录项自身都有固定长度，而将文件名放置在目录后面的堆中(解决了3中，一个文件目录项被移走后，引起的长度可变的空隙问题)。但即使在子需要对堆进行管理
    - 查找文件在一般都是线性地从头到尾对目录进行搜索。对非常长的目录，线性查找很慢，可以使用散列表
- 共享文件
  - 使用文件链接实现
  - 出现的问题，文件链接时，共享的目录将被复制到用户的目录下，如果目录中包含了磁盘地址，那一个用户修改了文件，其他共享用户不知道(信息仅仅有更新)
    - 解决方案
      - 一、磁盘块不列入目录，而是列入一个与文件本身关联的小型数据结构中，目录所指向这个小型数据结构这个小数据结构共享，多个共享用户需要读这个入口)
      - 二、让系统建立一个类型为LINK的新文件，共享文件在用户之间存在一个链接，新的LINK文件只包含它所链接的文件的路径名(将号链接对相当于还是保证了数据的唯一性)
- 日志结构文件系统 —— 为了解决文件系统的性能问题。因为CPU越来越快、磁盘容量越来越大，而磁盘的寻道时间却没有变化(因态磁盘没有寻道时间)
- 日志文件系统

虚拟文件系统

Windows有一个主要的NTFS文件系统，但也有一个包含老的但仍使用的FAT-32或FAT-16驱动器或分区。Windows通过指定不同的盘符来处理这些不同的文件系统，比如：“C:”“D:”

UNIX将多种文件系统整个到一个统一的结构中

Linux系统可以用ext2作为根文件系统，ext3分区装载在/usr下，另一块采用ReiserFS文件系统的硬盘装载在/home下，以及一个ISO 9660的CD-ROM临时装载在/mnt下

- 绝大多数UNIX操作系统使用虚拟文件系统(Virtual File System，VFS)，尝试将多种文件系统统一成一个有序的结构
  - 关键思想是抽象出所有文件系统都共有的部分，并将这部分以前提在单独的一层，该层调用底层的实际文件系统来具体管理数据
  - 所有和文件相关的系统调用在最初的处理上都指向虚拟文件系统。这些来自用户进程的调用，都是标准的POSIX系统调用，比如open，read，write和seek等
- VFS有两个不同的接口：上层给用户进程的接口和下层给实际文件系统的接口
  - VFS如何工作
    - 系统启动时，根文件系统在VFS中注册。另外，当装载其他文件系统时，不管是自动还是操作过程，它们必须在VFS中注册
    - 文件系统注册时，最基本的工作就是提供一个包含VFS所需要的函数地址的列表(可以使一个或多个长的调用实参表，每个VFS对象一个)
    - 装载一个文件，搜索已经装载文件系统的超块表来确定超块，找到根目录，查找其路径的文件
    - VFS创建一个节点并调用实际文件系统，返回文件i节点的信息，把后信息被复制到一节点中(在RAM中)，此时，v节点操作的函数表的指针中包含了read、write等
    - v节点被创建后，为了进程调用，VFS在文件描述符表汇总创建一个表项，并将它指向新的v节点。最后，VFS向调用者返回文件描述符，调用者可以用它去读、写或者关闭文件

文件系统管理和优化

磁盘空间管理

几乎所有的文件系统都把文件分割成固定大小的块来存储，各块之间不一定相邻

- 块大小
  - 从历史观点上来说，文件系统将大小设在1-4KB之间。但随着磁盘超过了1TB，可以将块的大小提升到64KB
- 如何跟踪空闲块
  - 磁盘链表
  - 位图
- 磁盘配额
  - 强制性磁盘配额机制。系统管理员分给每个用户拥有文件和块的最大数量

文件系统备份

- 回收站

文件系统的一致性

计算机带有实用程序检查文件系统的一致性，UNIX有fsck，Windows的scandisk。系统启东时，也别是磁盘之后的重新启动，可以进行该实用程序

- 一致性检查分为两种
  - 块的一致性检查：程序构造两张表。每张表中为每个块设立一个计数器，都初始为0。第一个表中的计数器跟踪该块在文件中的出现次数。第二个表中的计数器跟踪该块在空闲表或空闲位图中出现的次数
  - 文件的一致性检查

文件系统性能

- 高速缓存，块高速缓存
  - 管理高速缓存的软算法，检查全部的读请求，查看在高速缓存中是否有所需要的块。如果存在，可执行操作而无需访问磁盘。如果不存在，首先把它读入高速缓存，在复制到所需的地方
  - 快速的方法确定所需要的块是否存在 —— 常见的方法是将设备和磁盘地址进行散列操作
  - 高速缓存已满，此时需要调入新的块，则把原来的某一块调入高速缓存
- 块提前读
  - 只适用于实际顺序读取的文件。对随机访问文件，提前读块毫不起作用
- 减少磁盘臂运动
  - 把有可能顺序访问的块放在一起，当然最好是在同一个柱面，从而减少磁盘臂的移动次数

磁盘碎片整理

移动文件使它们相邻，并把所有的空闲空间放在一个或多个大的连续的区域内。Windows有一个程序defrag就是从事这个工作