第七章 文件系统

信息是计算机系统中的重要资源。操作系统中的一个重要组成部分，文件系统，就负责信息的组织、存储和访问。

文件系统的功能就是提供高效、快速和方便的信息存储和访问功能。本章的主要内容就是信息的组织。

7.1 引言

返回

7.1.1 文件管理的目的

- 方便的文件访问和控制：以符号名称作为文件标识，便于用户使用；
- 并发文件访问和控制：在多道程系统中支持对文件的并发访问和控制；
- 统一的用户接口：在不同设备上提供同样的接口，方便用户操作和编程；
- 多种文件访问权限：在多用户系统中的不同用户对同一文件会有不同的访问权限；
- 优化性能：存储效率、检索性能、读写性能；
- 差错恢复：能够验证文件的正确性，并具有一定的差错恢复能力；

返回

7.1.2 文件系统的基本概念

1. 文件

文件是具有符号名的数据项的集合。文件名是文件的标识符号。文件包括两部分：

- 文件体：文件本身的信息；
- 文件说明：文件存储和管理信息；如：文件名、文件内部标识、文件存储地址、访问权限、访问时间等；

返回

2. 文件系统

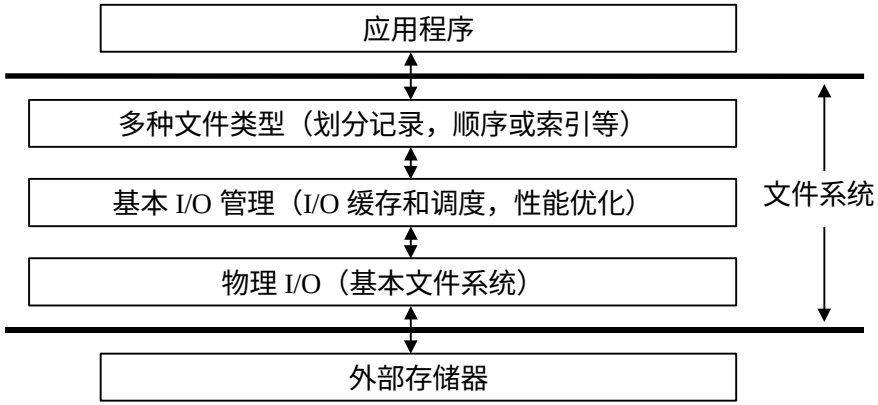
文件系统是操作系统中管理文件的机构，提供文件存储和访问功能。

3. 目录

目录是由文件说明索引组成的用于文件检索的特殊文件。

7.1.3 文件系统的结构和功能元素

1. 文件系统的结构



返回

2. 文件管理的服务功能元素

(文件系统向上层用户提供的服务)

- 文件访问：文件的创建、打开和关闭，文件的读写；
- 目录管理：用于文件访问和控制的信息，不包括文件内容
- 文件结构管理：划分记录，顺序，索引
- 访问控制：并发访问和用户权限
- 限额(quota)：限制每个用户能够建立的文件数目、占用外存空间大小等
- 审计(auditing)：记录对指定文件的使用信息（如访问时间和用户等），保存在日志中

3. 文件系统的实现功能元素

(文件系统要实现的功能模块)

- 文件的分块存储：与外存的存储块相配合
- I/O缓冲和调度：性能优化
- 文件定位：在外存上查找文件的各个存储块
- 外存存储空间管理：如分配和释放。主要针对可改写的外存如磁盘。
- 外存设备访问和控制：包括由设备驱动程序支持的各种基本文件系统如硬盘，软盘，CD ROM等

7.2 文件的组织(file organization)

文件组织讨论文件的内部逻辑结构，主要考虑因素是文件存储性能和访问性能。

返回

7.2.1文件的组织

文件的组织是指从用户观点出发讨论文件内部的逻辑结构(logical structure)或用户访问模式；它可以独立于在外存上的物理存储。

- 文件逻辑结构的设计要求：
  - 访问性能：便于检索；便于修改
  - 存储性能：向物理存储转换方便，节省空间
- 文件的不同组织层次：域、记录、文件

返回

7.2.2 文件的组织类型

1. 无结构文件

文件体为字节流，不划分记录，顺序访问，每次读写访问可以指定任意数据长度。当前操作系统中常用的文件组织。

2. 累积文件(pile)

文件体为无结构记录序列，通过特定分隔符来划分记录，各记录大小和组成可变。新记录总是添加到文件末尾。如日志log，或电子邮件的邮箱文件(mailbox)。检索必须从头开始。

返回

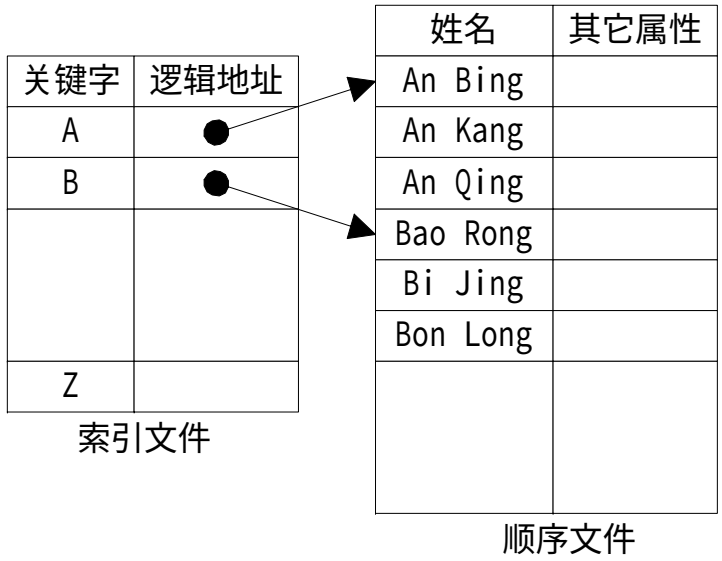
3. 顺序文件(sequential file)

文件体为大小相同的排序记录序列。它由一个主文件和一个临时文件组成。记录大小相同，按某个关键字域(key field)排序，存放在主文件(master file)中。新记录暂时保存在日志或事务文件(log file or transaction file)中，定期归并入主文件。

4. 索引顺序文件(indexed-sequential file)

在顺序文件（主文件main file）的基础上，另外建立索引(index)和溢出文件(overflow file)。这样做的目的是加快顺序文件的检索速度。

- 在索引文件中，可将关键字域中的取值划分若干个区间（如A~Z可以划分为A到Z共26个区间），每个区间对应一个索引项，后者指向该区间的开头记录。新记录暂时保存在溢出文件中，定期归并入主文件。
- 通过划分层次，在记录数量较大时，比顺序文件大大缩短检索时间。顺序文件是N/2(这时可使用折半查找)，而索引顺序文件（一级索引）是i/2 + N/(2*i)，其中i为索引长度。索引还可以是多级的。如：有1000,000条记录的顺序文件的平均检索长度为500,000，而在添加一个有1000条索引项的索引文件后，平均检索长度为1000。



索引顺序文件

5. 索引文件(indexed file)

记录大小不必相同，不必排序，存放在主文件(primary file)中。索引文件与索引顺序文件的区别在于主文件不排序。另外建立索引，每个索引项指向一个记录，索引项按照记录中的某个关键字域排序。对同一主文件，可以针对不同的关键字域相应建立多个索引。索引文件的记录项通常较小，查找速度快，便于随机访问(random access)。

6. 哈希文件或直接文件(hashed file or direct file)

记录大小相同。由主文件和溢出文件组成。记录位置由哈希函数确定。检索时给出记录编号，通过哈希函数计算出该记录在文件中的相对位置。访问速度快，但在主文件中有空闲空间。

7.3 文件目录

目录是由文件说明索引组成的用于文件检索的特殊文件。文件目录的内容主要是文件访问的控制信息（不包括文件内容）。

返回

7.3.1 目录内容

目录的内容是文件属性信息(properties)，其中的一部分是用户可获取的。

1. 基本信息

- 文件名：字符串，通常在不同系统中允许不同的最大长度。可以修改。有些系统允许同一个文件有多个别名(alias)；
- 别名的数目；
- 文件类型：可有多种不同的划分方法，如：
  - 有无结构（记录文件，流式文件）
  - 内容（二进制，文本）
  - 用途（源代码，目标代码，可执行文件，数据）
  - 属性attribute（如系统，隐含等）
  - 文件组织（如顺序，索引等）

返回

2. 地址信息

- 存放位置：包括哪个设备或文件卷volume，以及各个存储块位置；
- 文件长度（当前和上限）：以字节、字或存储块为单位。可以通过写入或创建、打开、关闭等操作而变化。

3. 访问控制信息

- 文件所有者（属主）：通常是创建文件的用户，或者改变已有文件的属主；
- 访问权限（控制各用户可使用的访问方式）：如读、写、执行、删除等；

4. 使用信息

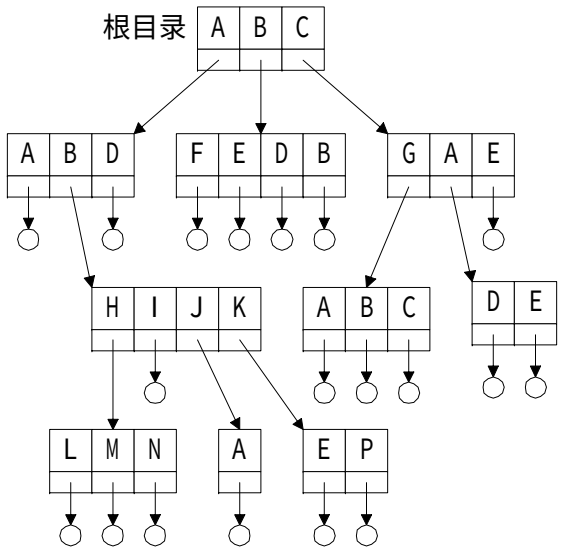
- 创建时间
- 最后一次读访问的时间和用户
- 最后一次写访问的时间和用户

7.3.2 目录结构类型

目录结构讨论目录的组织结构，设计目标是检索效率。

- 一级目录：整个目录组织是一个线性结构，系统中的所有文件都建立在一张目录表中。它主要用于单用户操作系统。它具有如下的特点：
  - 结构简单；
  - 文件多时，目录检索时间长；
  - 有命名冲突：如重名(多个文件有相同的文件名) 或别名(一个文件有多个不同的文件名)
- 二级目录：在根目录下，每个用户对应一个目录（第二级目录）；在用户目录下是该用户的文件，而不再有下级目录。适用于多用户系统，各用户可有自己的专用目录。

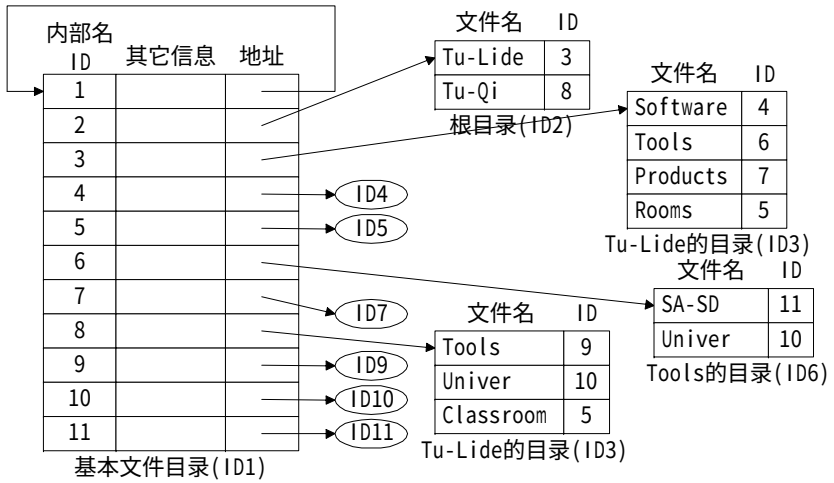
返回



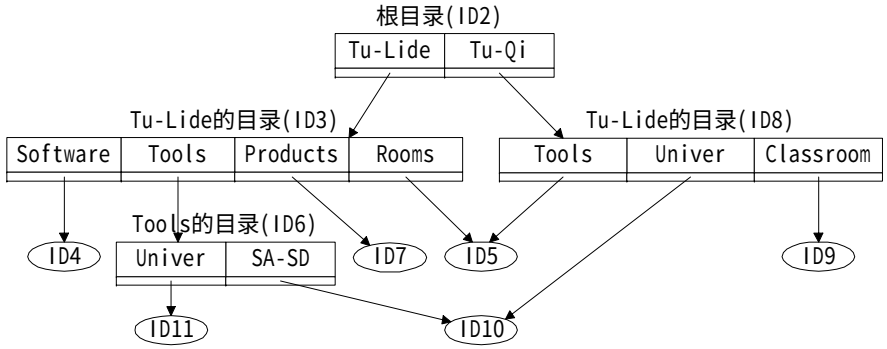
多级目录组织

- 多级目录：或称为树状目录(tree-like)。在文件数目较多时，便于系统和用户将文件分散管理。适用于较大的文件系统管理。目录级别太多时，会增加路径检索时间。
  - 目录名：可以修改。
  - 目录树：中间结点是目录，叶子结点是目录或文件。
  - 目录的上下级关系：当前目录(current directory, working directory)、父目录(parent directory)、子目录(subdirectory)、根目录(root directory)等；
  - 路径(path)：每个目录或文件，可以由根目录开始依次经由的各级目录名，加上最终的目录名或文件名来表示；

- 改进的多级目录：为了提高目录检索速度，可把目录中的文件说明（文件描述符）信息分成两个部分：
  - 符号文件目录：由文件名和文件内部标识组成的树状结构，按文件名排序；
  - 基本文件目录（索引节点目录）：由其余文件说明信息组成的线性结构，按文件内部标识排序；



基本文件目录



符号文件目录的层次结构

7.3.3 文件别名的实现

提供文件共享的方法有两种：各用户通过唯一的共享文件的路径名访问共享文件（该方法的访问速度慢，适用于不经常访问的文件共享），或利用多个目录中的不同文件名来描述同一共享文件（即文件别名，该方法的访问速度快，但会影响文件系统的树状结构，适用于经常访问的文件共享，同时存在一定的限制）。文件别名的实现方法有以下两种：

- 基于索引结点
- 基于符号链接

返回

1. 基于索引结点(index node)的文件别名

也称为硬链接（hard link）；基于改进的多级目录结构，将目录内容分为两部分：文件名和索引结点。前者包括文件名和索引结点编号，后者包括文件的其他内容（包括属主和访问权限）。通过多个文件名链接(link)到同一个索引结点，可建立同一个文件的多个彼此平等的别名。别名的数目记录在索引结点的链接计数中，若其减至0，则文件被删除。

- UNIX举例："ln source target ; rm source"则该文件还存在，文件名为target；
- 限制：不能跨越不同文件卷；通常不适用于目录（在UNIX中只对超级用户允许），否则由树状变为网状。

2. 基于符号链接(symbolic link, shortcut)的文件别名

它是一种特殊类型的文件，其内容是到另一个目录或文件路径的链接。建立符号链接文件，并不影响原文件，实际上它们各是一个文件。可以建立任意的别名关系，甚至原文件是在其他计算机上。

- UNIX举例："ln -s a b ; rm a"则文件a不存在，b能被控制但无法访问。若a是目录，"ln -s /user/a /tmp/b"则"cd /tmp/b ; cd .."是进入目录"/user"而不是"/tmp"；
- 缺点：空间和时间开销更大。如果设置不当，上下级目录关系可能会形成环状。

7.4 文件和目录的使用

这一部分讨论操作系统提供的与文件系统相关的API。

返回

7.4.1 文件访问

文件访问是指围绕文件内容读写进行的文件操作。

- 打开open：为文件读写所进行的准备。给出文件路径，获得文件句柄(file handle)，或文件描述符(file descriptor)。需将该文件的目录项读入到内存中。
- 关闭close：释放文件描述符，把该文件在内存缓冲区的内容更新到外存上。
- 复制文件句柄dup：用于子进程与父进程间的文件共享，复制前后的文件句柄有相同的文件名、文件指针和访问权限；

返回

- 读read、写write和移动文件读写指针lseek：系统为每个打开文件维护一个读写指针(read-write pointer)，它是相对于文件开头的偏移地址(offset)。读写指针指向每次文件读写的开始位置，在每次读写完成后，读写指针按照读写的数据量自动后移相应数值。
- 执行exec：执行一个可执行文件；
- 修改文件的访问模式（fcntl和ioctl）：提供对打开文件的控制，如：文件句柄复制、读写文件句柄标志、读写文件状态标志、文件锁定控制、流（stream）的控制；

7.4.2 文件控制

文件控制是指围绕文件属性控制进行的文件操作。

- 创建（creat和open）：给出文件路径，获得新文件的文件句柄；
- 删除unlink：对于symbolic link和hard link，删除效果是不同的；
- 获取文件属性（stat和fstat）：stat的参数为文件名，fstat的参数为文件句柄；
- 修改文件名rename；
- 修改文件属主chown，修改访问权限chmod：与相应系统命令类似；
- 文件别名控制：创建symlink或link，读取链接路径readlink；

返回

7.4.3 目录管理

目录管理是指目录访问和目录属性控制。

- 进行文件访问和控制时，由操作系统自动更新目录内容
- 目录创建mkdir，删除rmdir，修改目录名rename。只适用于超级用户：mknod（建立文件目录项）和unlink（删除目录项）
- 修改当前目录chdir；

返回

7.4.4 伪文件(pseudo file)

伪文件是指具有文件某些特征的系统资源或设备，它们的访问和控制方式与文件类似。

- 特点
  - 内容并不保存在外存上，而是在其他外部设备上或内存里
  - 随文件类型的不同，适用于某些文件访问和控制的系统调用，如：open, read, write, close, chmod, chown。
  - 创建时使用特定的系统调用，如：创建管道pipe，创建管套socket，创建设备文件mknod
- 类型
  - 设备：字符设备或块设备，可以直接访问设备中的字节数据或数据块。如终端、硬盘、内存等。在UNIX中称为特殊文件(special file)。
  - 进程间通信：本计算机或通过网络。如：管道，管套等。

返回

7.5 文件共享

返回

7.5.1 文件的访问权限

设置文件访问权限的目的是为了在多个用户间提供有效的文件共享机制；

- 文件访问类型：
  - 读read：可读出文件内容；
  - 写write（修改update或添加append）：可把数据写入文件；
  - 执行execute：可由系统读出文件内容，作为代码执行；
  - 删除delete：可删除文件；
  - 修改访问权限change protection：修改文件属主或访问权限

返回

- 用户范围类型：
  - 指定用户
  - 用户组
  - 任意用户
- 访问类型和用户范围的组合：
  - 访问矩阵：矩阵的一维是每个目录和文件，另一维是用户范围，每个元素是允许的访问方式
  - 访问策略(policy)：每种文件访问方式，所允许或禁止的用户范围。可以将文件访问方式推广到其他操作如用户管理，备份，网络访问等。

7.5.2 文件的并发访问

文件并发访问控制的目的是提供多个进程并发访问同一文件的机制。

- 访问文件之前，必须先打开文件：如果文件的目录内容不在内存，则将其从外存读入，否则，仍使用已在内存的目录内容。这样，多个进程访问同一个文件都使用内存中同一个目录内容，保证了文件系统的一致性。
- 文件锁定(file lock)：可以协调对文件指定区域的互斥访问
  - Solaris 2.3中lockf的锁定方式：
    - F_UNLOCK：取消锁定；
    - F_LOCK：锁定；如果已被锁定，则阻塞；
    - F_TLOCK：锁定；如果已被锁定，则失败返回
    - F_TEST：锁定测试；
- 利用进程间通信，协调对文件的访问；

返回

7.6 外存存储空间管理

讨论如何高效地进行数据存储

返回

7.6.1 文件存储空间分配(file allocation)

1. 新创建文件的存储空间（文件长度）分配方法

- 预分配(preallocation)：创建时(这时已知文件长度)一次分配指定的存储空间，如文件复制时的目标文件。
- 动态分配(dynamic allocation)：需要存储空间时才分配（创建时无法确定文件长度），如写入数据到文件。

返回

2. 文件存储单位：簇（cluster）

文件的存储空间通常由多个分立的簇组成，而每个簇包含若干个连续的扇区(sector)。

- 簇的大小
  - 两个极端：大到能容纳整个文件，小到一个外存存储块；
  - 簇较大：提高I/O访问性能，减小管理开销；但簇内碎片浪费问题较严重；
  - 簇较小：簇内的碎片浪费较小，特别是大量小文件时有利；但存在簇编号空间不够的问题（如FAT12、16、32）；
- 簇的分配方法：两种
  - 簇大小可变，其上限较大：I/O访问性能较好，文件存储空间的管理困难（类似于动态分区存储管理）
  - 簇大小固定，较小：文件存储空间使用灵活，但I/O访问性能下降，文件管理所需空间开销较大
- 文件卷容量与簇大小的关系
  - 文件卷容量越大，若簇的总数保持不变即簇编号所需位数保持不变，则簇越大。缺点：簇内碎片浪费越多
  - 文件卷容量越大，若簇大小不变，则簇总数越多，相应簇编号所需位数越多。如簇编号长度为12、16、32二进制位，即构成FAT12、FAT16、FAT32。

3. 文件存储分配数据结构

采用怎样的数据结构来记录一个文件的各个部分的位置。

- 连续分配(contiguous)：只需记录第一个簇的位置，适用于预分配方法。可以通过紧缩(compact)将外存空闲空间合并成连续的区域。
- 链式分配(chained)：在每个簇中有指向下一个簇的指针。可以通过合并(consolidation)将一个文件的各个簇连续存放，以提高I/O访问性能。
- 索引分配(indexed)：文件的第一个簇中记录了该文件的其他簇的位置。可以每处存放一个簇或连续多个簇（只需在索引中记录连续簇的数目）。

7.6.2 外存空闲空间管理(free space management)方法

外存空闲空间管理的数据结构通常称为磁盘分配表(disk allocation table)，分配的基本单位是簇。文件系统可靠性包括检错和差错恢复。空闲空间的管理方法：三种，均适用于上述几种文件存储分配数据结构；

- 位示图(bitmap)：每一位表示一个簇，取值0和1分别表示空闲和占用。
- 空闲空间链接(chained free space)：每个空闲簇中有指向下一个空闲簇的指针，所有空闲簇构成一个链表。不需要磁盘分配表，节省空间。每次申请空闲簇只需取出链表开头的空闲簇即可。
- 空闲空间索引(indexed free space)：在一个空闲簇中记录其他几个空闲簇的位置。

注：可以上述方法结合，应用于不同的场合。如：位示图应用于索引结点表格，链接和索引结合应用于文件区的空闲空间。

返回

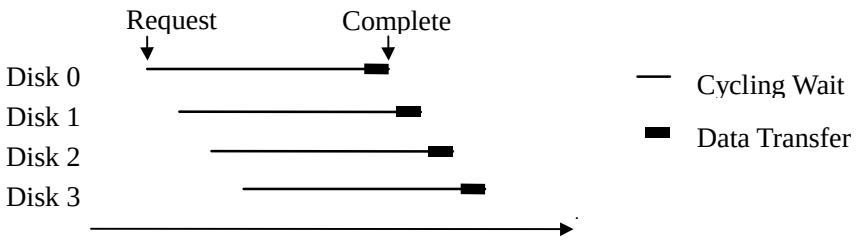
7.6.3 文件卷

- 磁盘分区(partition)：通常把一个物理磁盘的存储空间划分为几个相互独立的部分，称为"分区"。一个分区的参数包括：磁盘参数（如每道扇区数和磁头数），分区的起始和结束柱面等。
- 文件卷(volume)：或称为"逻辑驱动器(logical drive)"。在同一个文件卷中使用同一份管理数据进行文件分配和外存空闲空间管理，而在不同的文件卷中使用相互独立的管理数据。
  - 一个文件不能分散存放在多个文件卷中，其最大长度不超过所在文件卷的容量。
  - 通常一个文件卷只能存放在一个物理外设上（并不绝对），如一个磁盘分区或一盘磁带。

返回

- 格式化(format)：在一个文件卷上建立文件系统，即：
  - 建立并初始化用于进行文件分配和外存空闲空间管理的管理数据。
  - 通常，进行格式化操作使得一个文件卷上原有的文件都被删除。
- 扩展文件卷集(extended volume set)：一个文件卷由一个或几个磁盘上的多个磁盘分区依次连接组成。可以容纳长度大于磁盘分区容量的文件。
  - 实例：Windows NT中的扩展文件卷集。

- 磁盘交叉存储(disk interleaving)：将一个文件卷的存储块依次分散在多个磁盘上。如4个磁盘，则磁盘0上是文件卷块0, 4, 8, …，磁盘1上是文件卷块1, 5, 9, …。
  - 优点：提高I/O效率。如果需要访问一个文件的多个存储块，而它们分散在多个磁盘上，则可以并发地向多个磁盘发出请求，并可在此基础上提供文件系统的容错功能。关键：磁盘访问时间大部分由旋转等待时间组成。
  - 需要相应硬件设备：如多个硬盘连接在同一个或不同的SCSI接口上，或者两个硬盘连接在一个或不同的IDE接口上（两个硬盘连接在同一个IDE接口上，不能提高I/O效率）
  - 实例：Windows NT中的条带卷(stripe set)，每个文件卷块的大小是64KB。
  - 类似例子：在虚拟存储器中建立多个交换区，分散在多个磁盘上



多个磁盘上的交换区访问

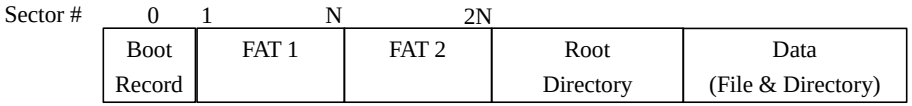
7.7 文件系统举例

返回

7.7.1 MS DOS的文件系统

多级目录，不支持文件别名，无用户访问权限控制

1. 磁盘文件卷结构



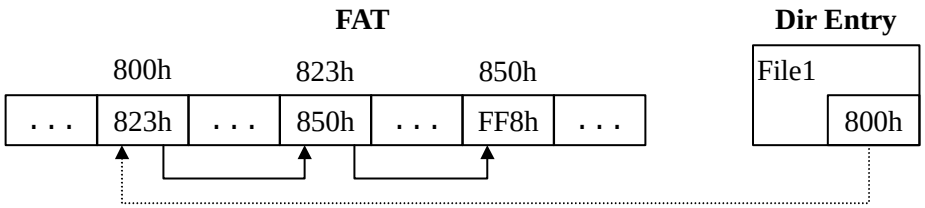
Volume Structure in MS DOS

返回

- 文件卷(volume)信息：记录在引导记录的扇区中。包括：簇大小，根目录项数目，FAT表大小，磁盘参数（每道扇区数，磁头数），文件卷中的扇区总数，簇编号长度等
  - 逻辑扇区号：三元组（柱面号，磁头号，扇区号）->一个文件卷中从0开始对每个扇区编号，优点：屏蔽了物理磁盘参数的不同
  - 允许同时访问的文件卷数目上限可以由config.sys文件中的LASTDRIVE= 语句指定
  - 簇(cluster)：由若干个扇区组成。在一个文件卷中从0开始对每个簇编号。

- FAT表：两个镜像，互为备份。文件卷中的每个簇均对应一个FAT表项，文件分配采用链式分配方法。
  - 每个FAT表项所占位数是簇编号的位数，其值是（以FAT12为例）：
    - 0：表示该簇空闲
    - FF7h：物理坏扇区
    - FF8h~FFFh：表示该簇是文件的最后一个簇
    - 其他值：表示该簇被文件占用，而且表项中的值是文件下一个簇的编号。
  - FAT表大小占文件卷容量的比例：
    - 簇编号位数/ (8*512*每个簇的扇区数)

- 目录：是目录项的顺序文件(即大小相同的排序记录序列)，不对目录项排序。
  - 若目录中包含的文件数目较多，则搜索效率低。
  - 每个目录项大小为32字节，其内容包括：文件名（8+3个字符），属性（包括文件、子目录和文件卷标识），最后一次修改时间和日期，文件长度，第一个簇的编号。
  - 在目录项中，若第一个字节为 E5h，则表示空目录项；若为 05h，则表示文件名的第一个字符为 E5h。
  - 文件名不区分大小写



2. 打开文件管理

- 系统文件表(SFT, System File Table)和任务文件表(JFT, Job File Table)：
  - SFT包含系统的所有打开文件，可以由几个表项依次连接组成。
  - JFT包含该任务（进程）的所有打开文件。JFT表项内容是到SFT表项的索引。
  - SFT的表项数目可由 config.sys文件中的 FILES= 来语句指定，默认是8。

7.7.2 Windows NT的文件系统

2

返回

7.7.3 UNIX的文件系统

3

返回