文件管理

笔记源文件：Markdown，长图，PDF，HTML

1. 文件系统: 信息的组织/存储/访问

1.1. 文件

1.1.1. 文件的概念

1️⃣文件的含义：具有一定功能的程序/数据集合，通过文件名(用户访问接口)访问

2️⃣文件逻辑分类

1. 流文件：中是指文件内信息不再划分单位，只是一串字符流，如exe文件
2. 记录文件：按逻辑把文件独立划分位多个信息单位(aka记录，可定长/不定长)，每个记录都有数据项

2️⃣文件的组成结构

1. 数据项：基本数据项(文件系统中最小/不可分的原子数据单位)+组合数据项(多个基本数据项)
2. 记录：描述对象某方面的属性
3. 文件：创建者定义的有关数据项集合，分为有结构/记录式文件+无结构/流式文件

举个例子：张三(基本数据项)→张三+18岁(组合数据项)→张三的所有有关属性(记录)→有结构文件(张三李四王五构成的学生信息文件)

1.1.2. 文件的属性

1️⃣内容：名称(唯一)+标识符+文件类型+文件位置+其他(文件大小/建立时间/用户标识)

PS1: 标识符——OS层面文件的唯一标签，对用户透明

PS2: 文件位置——指向文件的指针

2️⃣属性信息，放在目录中，存储在硬盘中

1.1.3. 文件的分类

1️⃣按用途：系统文件(只允许用户调用)+库文件(只允许用户调用)+用户文件(所有者/授权用户访问)

2️⃣按保护级别：只读+读写+执行文件(读/写/执行是三种分立的操作)+不保护文件(随你怎么搞)

3️⃣按流向：输入文件(如光盘/键盘中的文件)+输出文件(如打印机上的文件)+IO文件(磁盘)

4️⃣按数据形式：源文件(你写的代码文本&数据)+目标文件(源文件编译后)+可执行文件(编译后再链接)

1.1.4. 文件的操作

1.1.4.1. 基本操作

1️⃣ 创建文件：分配空间+目录中建立目录项+建立新的FCB，返回一个文件描述符

2️⃣ 删除文件：先删除目录项+回收存储空间

3️⃣ 读/写：查找目录→找到指定目录项→得到被读/要写文件的外存地址(指针)→开始读写

4️⃣ 截断文件：将文件缩减到指定大小，但保留文件属性

5️⃣ 设置文件读写位置：其实就是调整文件指针，让读写不再从头开始

1.1.4.2. 打开/关闭文件

1️⃣ 打开文件

1. OS把文件属性复制到内存，给用户返回一个索引
2. 用户通过这个索引向OS请求操作(打开)这个文件

2️⃣ 关闭文件

1. 撤销之前建立的索引信息，切断用户与该文件的联系
2. 若文件打开期间修改了，则将其写回外存

1.2. 文件的逻辑结构：用户对文件的组织

1.2.1. 概述

1️⃣ 含义：用户所观察到&处理的文件组织形式

2️⃣ 分类：有结构的记录文件(顺序/索引/索引顺序)+无结构的流文件

3️⃣ 文件访问方式

1. 顺序访问：按文件逻辑顺序依次存取
2. 随机存取：根据记录编号直接存取文件中的任意一个记录(可忽略前面的)

1.2.2. 有结构文件分类：由OS/程序决定

1.2.2.1. 顺序文件/连续结构

1️⃣ 含义：将逻辑文件信息连续存放

2️⃣ 分类：定成/不定长(记录长度是否固定)+串结构(记录顺序不按关键字排)/顺序结构(按关键字)

3️⃣ 优缺点：常用于批量记录读取，访问某个记录请求性能不佳，会有碎片/无法动态扩充

1.2.2.2. 索引文件(好比一个目录)

1️⃣ 索引表：

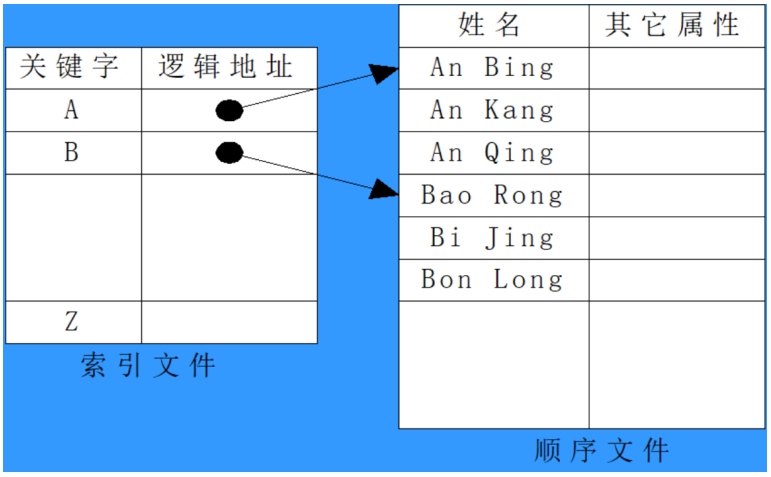
1. 存储逻辑文件记录信息的表，其本身就是一个定长文件
2. 每个表目指出一个文件记录在逻辑文件中的位置+长度

2 索引文件：索引表+逻辑文件

3 优缺点：随机访问(而不是从头开始)/删除修改添加高效；索引表本身占空间/查找策略影响大

1.2.2.3. 索引顺序文件

1 含义：将顺序文件中所有记录分组，每组第一个记录在索引表有索引项(关键字+该记录的指针)



2 检索操作：根据关键字检索索引表→找到组中第一个记录→顺序查找改组直到找到记录

3 优缺点：存取快，担任有额外存储开销(索引表)

1.2.2.4. 直接文件(散列HASH文件)

1 直接文件：由关键字值找到记录的物理地址

2 散列文件：典型直接文件，关键字$\xrightarrow{HASH函数}$物理地址，存取快但是有冲突

1.3. 物理结构：文件在外存的存储结构

1.3.1. 概述

1 用户按逻辑结构使用文件$\overset{转换}{\longleftrightarrow}$文件系统按物理结构管理文件

2 含义：**文件在外存的存放组织形式**

3 物理结构的类型：取决于外存方式——连续/链接/缩引分配↔连续/链接/缩引结构

4 文件存取方式：取决于存储设备特性(磁带是顺序存取，磁盘是直接存取)

5 物理块与逻辑记录

1. 物理块大小固定，逻辑记录大小可变，一个逻辑记录可只占物理块一部分也可跨越多个物理块
2. 逻辑块：OS会将文件信息分为与物理块大小相等的逻辑块

1.3.2. 连续/链接/缩引分配：见2.3. 文件(在存储器的)实现

1.4. 目录结构

1.4.1. 文件控制块(FCB)

1️⃣文件=FCB+文件体

2️⃣含义：用于描述和控制文件的数据结构

3️⃣内容：至少有文件名+文件物理地址，此外还有——

1. 文件结构：物理的(顺序/索引/索引)，逻辑的(记录/流)
2. 文件物理位置：存放设备名+存储位置+文件长度
3. 存取控制信息：存取权限，分为文件主or其他用户
4. 管理信息：文件建立/上次存取日期，当前使用状态，共享计数

1.4.2. 索引结点

1️⃣含义：将文件名和文件描述分开→文件名+索引结点(包含文件描述)

2️⃣存储特点：索引结点在外存，文件名+索引结点指针在内存(FCB)

3️⃣磁盘索引结点内容

文件的：主标识+类型+存取权限+物理地址+长度+链接计数+存取时间

1. 文件主标识：标识了文件大小/小组的标识符
2. 文件链接计数：文件系统中指向该文件名的指针数

4️⃣内存索引结点：打开文件时，磁盘索引结点$\xrightarrow{\text{复制}}$内存索引结点，有如下新增内容

1. 索引结点编号：标识内存索引结点
2. 状态：标识该节点是否上锁/修改
3. 访问计数：正在访问该文件的进程数
4. 逻辑设备号：文件所在文件系统的逻辑设备号
5. 链接指针：指向空链表/散列队列的指针

1.4.3. 目录文件&文件目录

1️⃣文件目录：FCB的有序集合

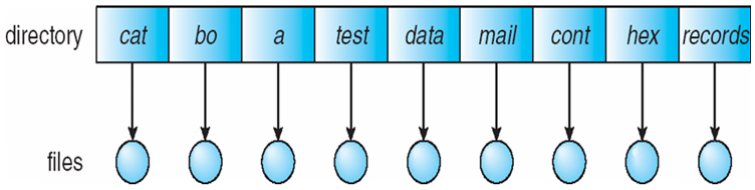
1. 用途：检索文件，实现按名存取，防止冲突
2. 目录项：就是FCB
3. 目录结构会影响到：文件存取速度/共享性/安全性

2️⃣目录文件：文件目录以文件形式保存在外存，这个文件就叫目录文件

3️⃣目录操作：建立/寻找/删除一个文件，列出目录列表，重命名文件，遍历文件系统

1.4.4. 目录类型

1.4.4.1. 单级目录结构：仅一表，每个文件占一表项



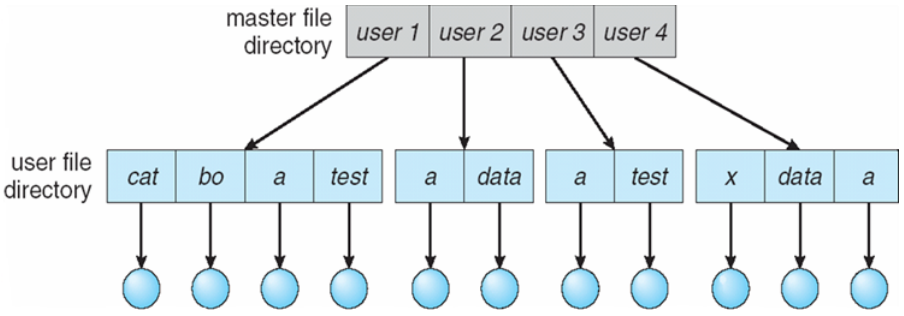
1 动态操作

1. 建立文件：验证不重名后→找出一个空表目然后将信息填入
2. 删除文件：找到对应目录项→找到物理地址→回收物理空间→回收目录项
3. 访问文件：检索到文件名→找出物理地址→然后操作文件

2 缺点：只有一个目录，不允许文件重名，查找慢

1.4.4.2. 二级目录

1 结构：主文件目录+用户文件目录



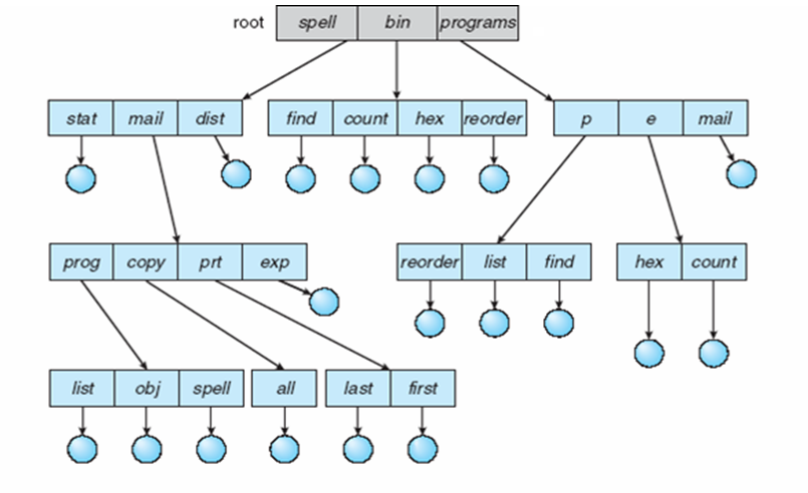
1. 主文件目录MFD：给出用户名+用户子目录指针
2. 用户子目录UFD：给出用户所有文件的FCB

2 动态操作

1. 访问：主目录中查找用户名→进入用户目录查找文件名→得到物理地址
2. 新用户建立文件：主目录中建立表目→给新用户目录分配空间→用户目录中添加表目→填入信息
3. 删除：直接删用户目录的目录项，用户目录为空就把主目录对应项删除

5 优劣：文件可重名，查找块，但灵活性差，不利于共享

1.4.4.3. 树形(多级)目录结构



1 结构：根目录=第一级的目录，非叶结点=均为目录文件(子目录)，叶=文件(每个文件都有唯一内部标识)

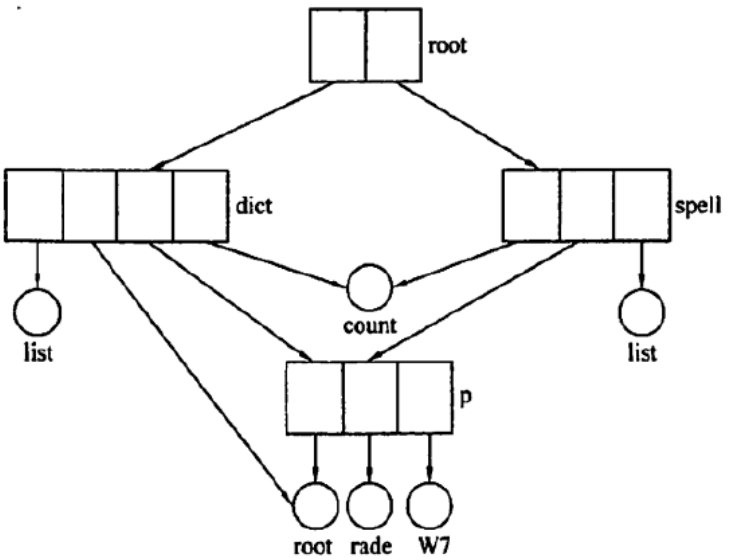
2 有关概念

1. 路径名：一个字符串，文件的唯一标识，按照`\`级联，分文绝对(根目录开始)+相对目录
2. 当前目录(工作目录)：选取某目录为当前目录，进程访问文件都是以这个目录为对照

   PS: 某目录的上一级用`..`标识

3 优劣：可重名/检索快/有利于分类，查找文件要多次访问目录文件(在外存)

1.4.4.4. (无环)图形目录：相比树形更便于共享



1 结构：树形目录+指向同一结点的有向边

2 共享计数器：

1. 向该节点增/减共享链时计数器+1/-1，同时增/减指针
2. 当共享计数器为0时结点才算真正删除，删除文件本身

1.4.5. 文件目录改进：分解FCB为两部分

1. 符号文件目录：文件名+文件内部标识→树状结构，按文件名排序
2. 基本文件目录：其余文件说明信息→线性结构，按文件内部标识排序

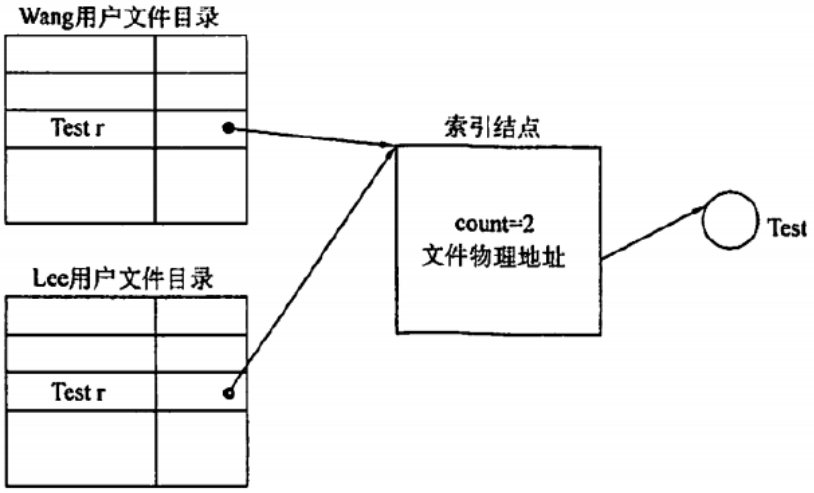
可以加快检索

1.5. 文件共享：不同用户使用一个文件

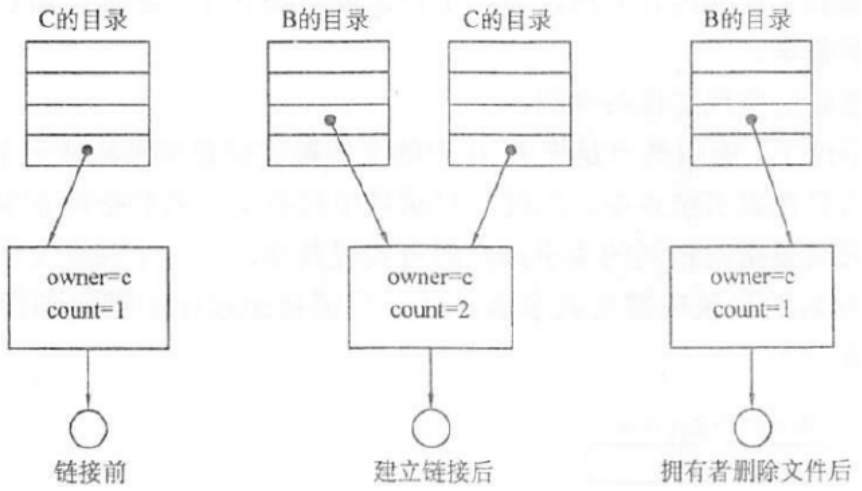
核心问题：如何实现共享+共享用户的存取控制

1.5.1. 基于索引结点的共享(硬链接)

1. 两个/多个目录项指向相同索引结点
2. 索引结点有一计数器，统计其对应目录项个数(为1时才可删除节点，大于1时只减1)



1.5.2. 利用符号链实现文件共享(软链接)



1. 符号链接(link类型文件)：包含被链接(共享)文件的**路径名**
2. 链接和指针：文件拥有者才拥有指针+其他用户路径名，共享给的用户只拥有链接(路径名)
3. 动态操作：将C的文件，共享给用户

    1. 共享：B目录中，建立符号链接指向文件 C
    2. 访问：检索目录，若检索到的是链接，OS自动忽略

1.6. 文件保护

1. 限制文件访问类型：读/写/执行/添加/删除/列出属性/重命名/复制/粘贴
2. 针对不同用户访问控制
   1. 含义：不同权限用户访问同一文件，采取不同访问类型
   2. 方式：访问控制矩阵/访问控制表/用户权限表(用数据结构记录用户的操作权限)+口令密码

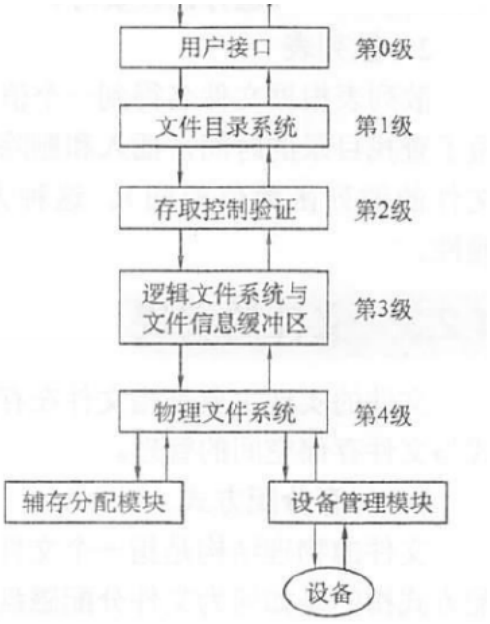
2. 文件系统的实现

2.1. 文件系统概述

2.1.1. 文件系统概念

1. 定义：OS中，(以文件方式)管理软件资源的软件+被管理的文件+被管理的数据结构
2. 文件系统的作用
   1. OS角度的文件系统：**对存储文件的空间**组织/分配/回收，**对文件**存储/检索/共享/保护
   2. 用户角度的文件系统：实现按名存取(用户知道文件名就可以获取文件信息)

2.1.2. 层次结构



1. 用户接口：图形窗口/命令行→连接现实与虚拟，用户通过用户接口向OS发出文件操作命令
2. 文件目录系统：查找目录，找到文件F索引信息
3. 存取控制验证：即使有索引信息，也要看你有无访问权
4. 逻辑文件系统与文件信息缓冲区：确定拥有访问权后，先找到逻辑地址
5. 物理文件系统：再由逻辑地址找到物理地址，这一层分为外存分配管理+设备管理

2.2. 目录的实现

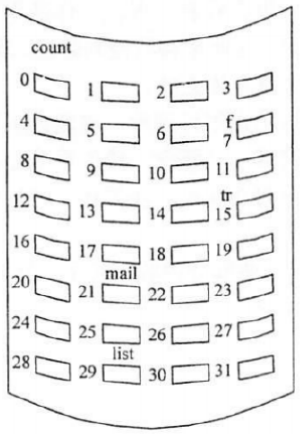
1. 线性表：按顺序存放文件信息(文件名+数据块)，创建删除文件就是线性表的插入和删除，创建是还要遍历检查文件是否已存在(因此低效)

2 散列表：(查找/删除时)将HASH函数将文件名转化为数字，用这个数字找到文件信息位置。但需要一个好的冲突解决策略

2.3. 文件在存储器的实现(文件的物理结构)

2.3.1. 外存分配方式: 如何为文件分配磁盘

2.3.1.1. 顺序分配：文件分配连续的磁盘区域



目录

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

1 文件创建过程：用户进程说明文件所需空间，OS查找空闲区管理表，若有足够空间就为文件分配，否则等待

2 结构：逻辑相邻可以是物理相邻

3 优劣：查找快，但是会产生外碎片，不适合文件大小动态变化

2.3.1.2.链接分配

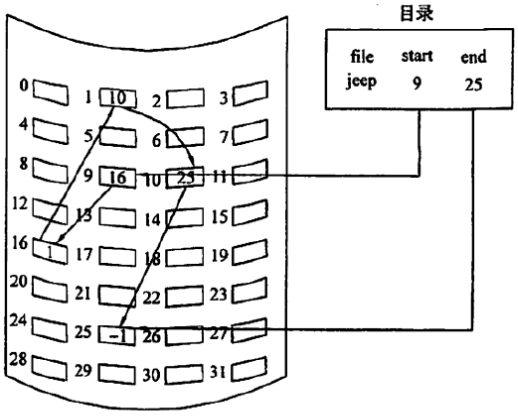


图 4-9 隐式链接

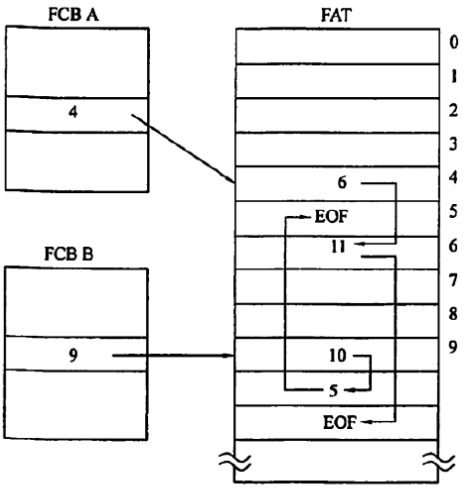


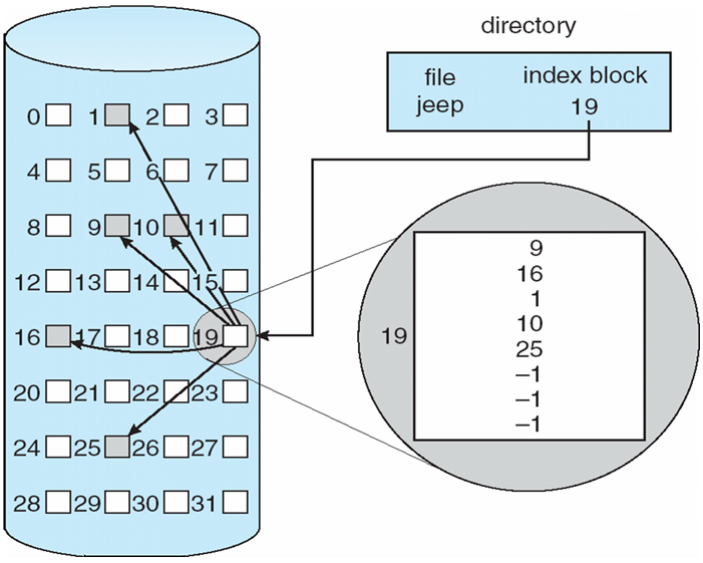
图 4-10 显式链接

1 隐式链接：

1. 含义：文件信息再若干不连续块中，每个块都指向文件的下一块，读文件只需找到第一块
2. 缺点：存取速度慢(不适于随机存取)，可靠性差(一个指针烂了后面都没用了)

2 显式连接：所有指针被放在**文件分配表FAT**(整个磁盘只有一张)，列出了所有块及其位置。缺点是仍需从表顶开始找

2.3.1.3. 索引分配



1 结构：每个文件分配一个索引块，块中放索引表，表中每个表项分给该文件一个物理块

PS：索引表——本质上是磁盘块地址数组，其中第i个条目指向文件第i块

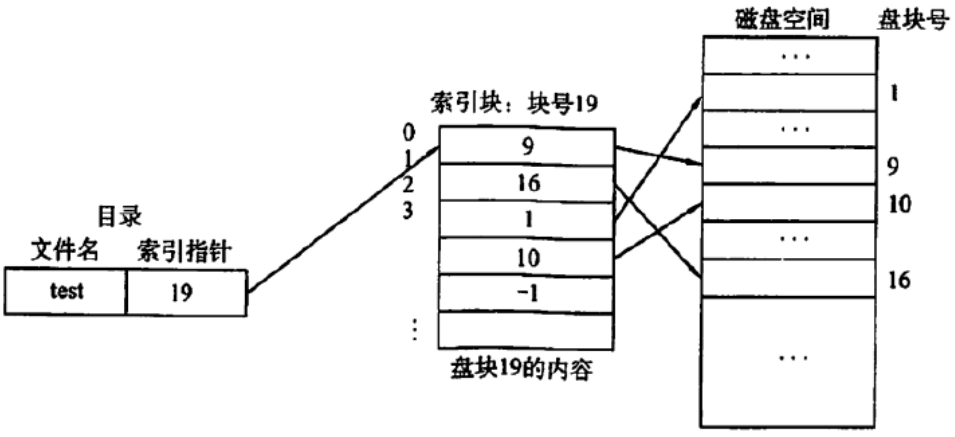
2 优点：能够随机存取，无外碎片

3 缺点：索引块占空间(其大小也很重要)，存取需要两次访问(但可在访问文件前就把索引表调入内存)

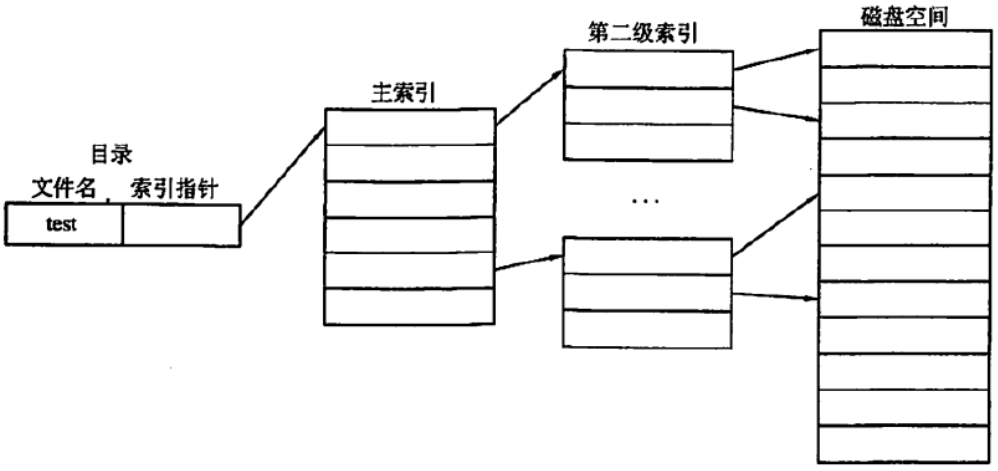
4 多级索引表：如果文件很大索引表也会很大，大到超过一块时，可以把索引表当作文件再为其建立索引表

2.3.1.4. 番外：多级索引分配补档

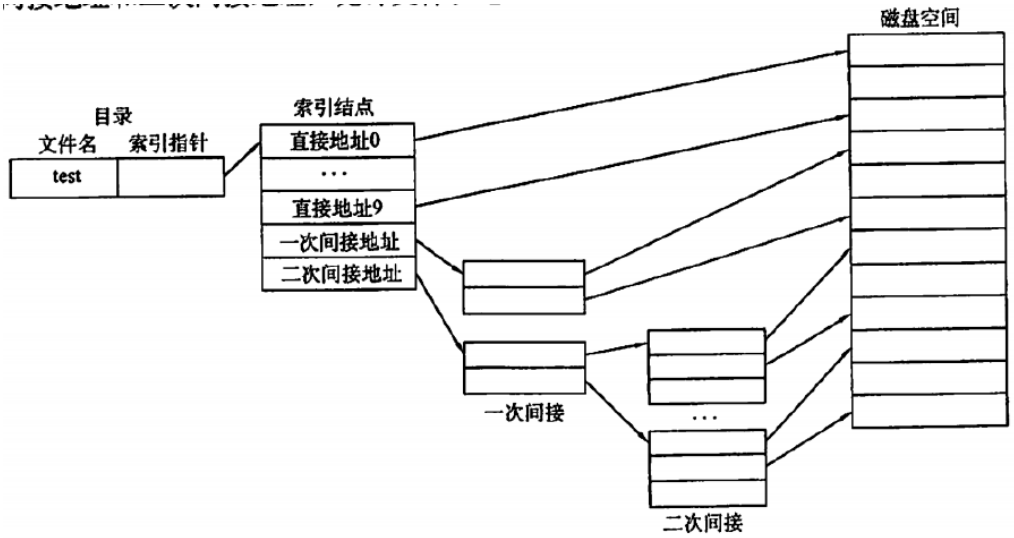
1 单级索引分配：文件名为test，索引表放在19块中，索引表指出的文件结构(块序)为9,16,1,10



2 两级索引分配：文件名test，索引到主索引的块号(每块含有二级索引块号)，又索引到第二级索引块号(每块包含文件块号序列)，索引到文件块号



3 混合索引分配：假设每个盘块大小4K



1. 直接地址：主要是为提高检索速度，此处有10个直接地址项，可索引10块共40K地址。文件小于40K时直接到这里来
2. 一次间接地址：就是一级索引分配，其一次可索引1K个盘块，每块4KB，故允许文件长4MB

   为什么是1K呢，因为一个表项4B，一块4KB，所以一块中就有4K个表项，一个表项索引到一块
3. 二次间接地址：两级索引分配，索引一次1K块，每块又索引1K块，总大小4MB*1K=4GB
4. 三次间接寻址：以此类推是4TB

2.3.2. (空闲)文件存储空间管理

2.3.2.1. 空闲文件表法

1 空闲文件/空白文件/自由文件：存储设备的一个连续空闲区

2 空闲盘块表：一个空闲文件占据一表项，表项包含第一空闲块号+空闲块号数+物理块号

序号	第一个空闲块号	空闲块数目	物理块号
1	5	3	(5，6，7)
2	13	5	(13，14，15，16，17)
3	20	6	(20，21，22，23，24，25，)
4	—	—	—

3 动态操作：

1. 请求存储空间：扫描空闲目录然后请求，请求=空闲直接分配，请求<空闲则留下空余项并修改表项，请求>空闲就不请求
2. 撤销文件：扫描空闲目录，将释放空间的第一个物理块号填到表中

   释放块与其他空闲块相邻则直接合并

2.3.2.2. 空闲块链表法

1 概述：将所有空闲块连接成一个链表+一个头指针

2 动态操作

1. 建立文件：从链首依次取下几个空闲块给文件
2. 撤销文件/回收存储空间：将要回收的空闲块依次链接入空闲块链表

3 空闲盘区链：

1. 将链表元素从空闲盘块改为空闲盘区
2. 每盘区有：若干空闲盘块+指向下一盘区指针+本盘区大小指示

2.3.2.3. 位视图法

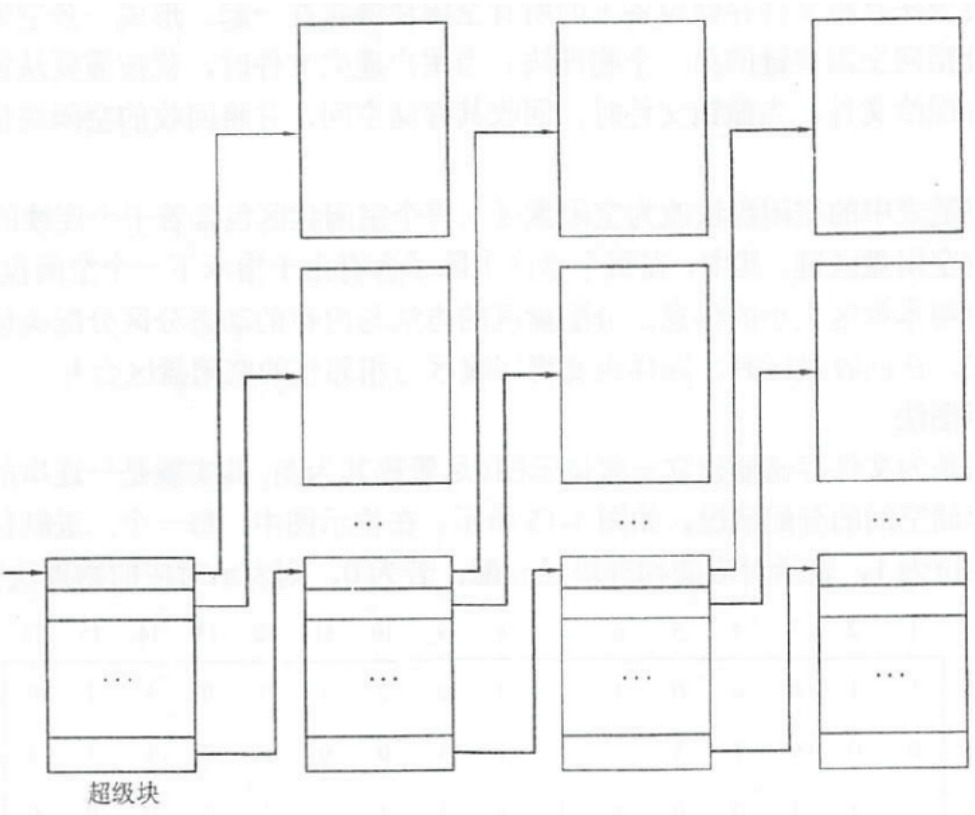
1 概述：文件存储器在主存中建立一张表，每位都对应一物理块，1=已分配，0=空闲

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
3	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
4																
⋮																
16																

2 动态操作：请求分配时OS先要扫描这张表，找出一组0，得到其盘号，然后改为1；回收正好相反

2.3.2.4. 成组链接法(UNIX)

1 基本结构



1. 将所有空闲块，100块一组分很多组
2. 每组需要记录两个数据：组中块数，所有块的编号。这两个数据放在前一组第一块中
3. 第一块的这两个数据放在超级块中
4. 总体结构：每组第一块链接成链表，组内多块构成堆栈

2 分配空闲盘块的方法

1. OS要为文件分配空闲块，先检查第一组中的块数
2. 若第一组块数>1，则超级块中记录的空闲块数-1，第一组中栈顶的块分配给文件
3. 重复2的操作，直到第一组中只剩一块了，这一块是来记录下一组的块号和块数的
4. 则将第二组的信息读入超级块，然后继续分配
5. 一直读到最后一组，栈顶盘块有结束标记0，代表所有块都用完了，停止分配

3 空闲盘块的回收

1. 若一组中不足100块，则直接把回收的块塞进去，然后超级块中放入该块号&空闲块数+1
2. 若一组已经100块了，则先将第一组的块号+块数写入回收块
3. 由此这个回收块就变成了新的第一组了(一组仅这一块)
4. 最后跟新超级块，块数变为1，设置块号为回收块的块号

3. 外存储器

3.1. 磁盘组：多磁盘绕中轴高速旋转

3.1.1. 物理记录定位

1. 柱面：同一垂直位置，各盘面相应磁道的集合，编号从外向内0, 1, 2, ..., L
2. 磁头：每个盘面一个磁头，编号从上而下0, 1, 2, ..., H，表示有效盘面
3. 扇区：盘面被分为若干大小相同的扇区，编号为0, 1, 2, ..., n

3.1.2. 磁盘编址

1. 逻辑块：传输数据的最小单位
2. 映射方式：逻辑块的一维数组$\overset{\text{映射}}{\longleftrightarrow}$磁盘相连扇区，磁道中扇区按顺序映射到逻辑块数组
3. 映射的顺序：某一磁道$\xrightarrow{\text{一个磁道映射完了}}$同柱面其他磁道$\xrightarrow{\text{柱面都用完了}}$从外向里其他柱面

3.1.3. 盘块物理地址&盘块号

1. 物理地址组成：[柱面号L] [磁头/盘面号M] [扇区号N]
2. 物理块→唯一盘块号：$B=(i\times M\times N)+(j\times N)+k$

范围：$i=0\ldots,L-1;j=0,\ldots,M-1;k=0,\ldots,N-1$

1. $(i\times M\times N)$：达第 i个柱面之前的总扇区数
2. $(j\times N)$：当前第i柱面，到达第 j个盘面之前的扇区数
3. k：当前盘面上的扇区号

3. 块号→物理块(一下计算自动强制转换int)

M×N=每柱面的扇区总数→B/(M×N)=之前经历了几个完整柱面

B%(M × N)=当前柱面中扇区的偏移数→[B%(M × N)]/N=当前柱面中经历了几个完整盘块

- 柱面号i=B/(M×N)
- 盘面号j=[B%(M × N)]/N
- 扇区号k=B%N

3.2. 磁盘访问时间

总时间$T_a=T_S+T_r+T_t$

1. 寻道时间$T_S=mn+s$：移动磁头到指定磁道所需时间

1. m: 磁头移动一个磁道所需时间，一般磁盘=0.2/高速磁盘≤0.1
2. n：移动磁道数
3. s：磁臂启动时间，一般为2ms

2. 旋转延迟时间$T_r=\frac{1}{2r}$：磁头到达了磁道后，让扇区移到磁头所需时间

1. r：磁盘的转速，公式基于假设——平均需要转0.5圈找到扇区

3 传输时间$T_t = \frac{b}{rN}$：读出或写入数据所需的时间

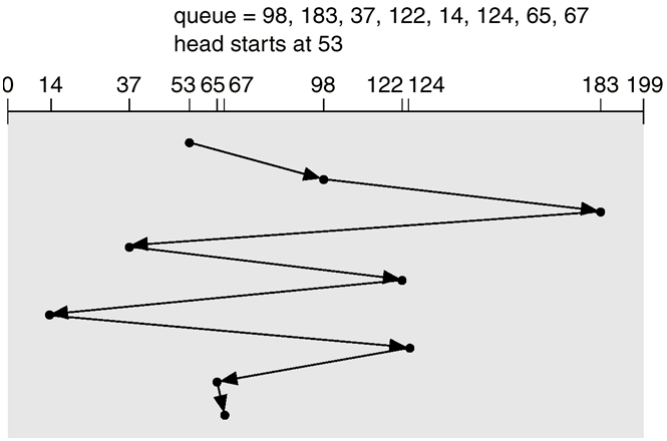
1. b：所读/写的字节数
2. N：一条磁道上的字节数→b/N=需要读多少个磁道(转多少圈)

➕磁盘IO提速小妙招——提高磁盘性能/设置Cache/选择好的调度算法

3.3. 磁盘调度

3.3.1. FCFS

如图磁头共移动了640个柱面距离。

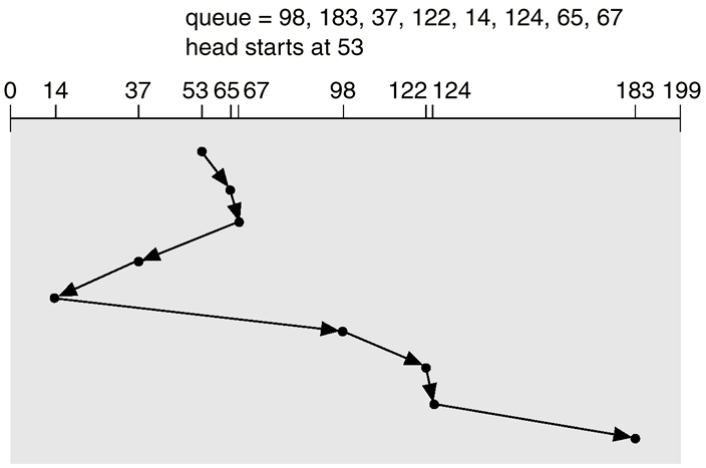


3.3.2. SSTF最短寻道时间优先

1 含义：选磁头当前位置，所需寻道时间最短的请求

2 特点：实质上是SJF(短作业优先)，会引起某些请求饥渴

3 示例：磁头移动总距离236柱面



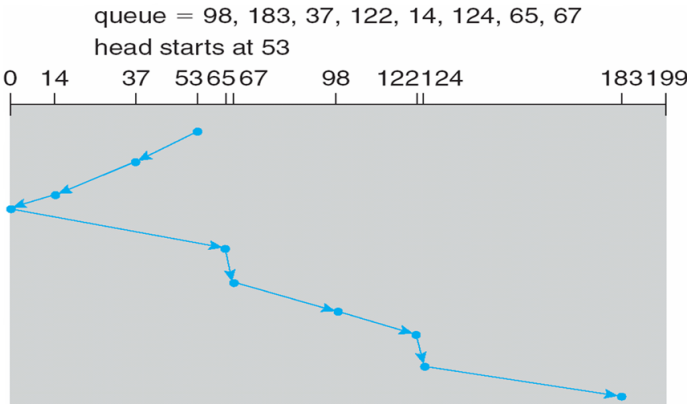
3.3.3. 各种SCAN算法

3.3.3.1. SCAN算法

1 含义：选择与磁头当前的移动方向一致且距离最近的请求

2 特点：磁头只会出现一次转向，转向时要先回到磁盘的最内/最外端

3 示例：磁头移动总距离236柱面

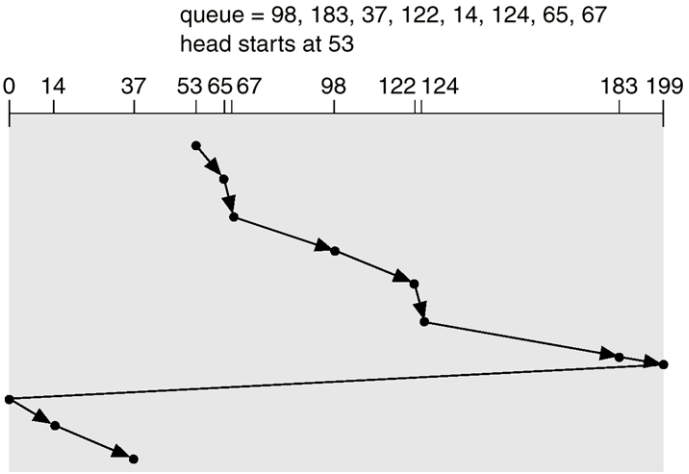


3.3.3.2. C-SCAN算法

1. 含义：磁头往同一方向移动→沿途响应最近请求→到顶后返回磁头开始处→再往原方向响应请求

2. 特点：把所有柱面看成一个循环，最后一个柱面接第一个柱面

3. 示例



3.3.3.3. 分步扫描算法

1. 步骤：

    1. 将请求队列分成若干个长度为N的子队列，按照FCFS处理这些子队列
    2. 在每个子队列中，按SCAN算法处理
    3. 处理某一子队列时，若又有新I/O请求，便将请求进程放入其他子队列

2. 性能：N极大时接近SCAN，N=1时就是FCFS

3. 优点：I/O请求等待时间不至于过长，不容易饿死

3.3.3.4. F-SCAN算法

只分两个子队列

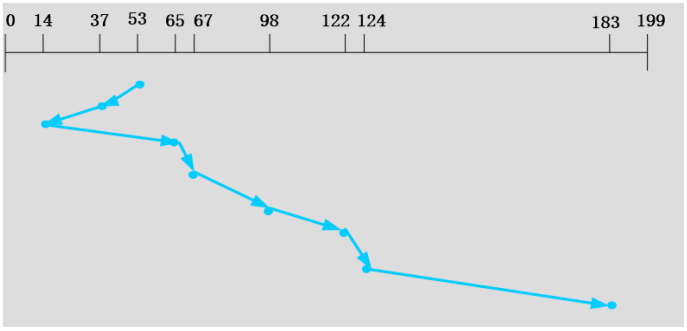
1. 当前队列：所有请求磁盘I/O的进程，按SCAN处理
2. 等待队列：SCAN期间新出现的请求进程

3.3.4. LOOK&C-LOOK算法

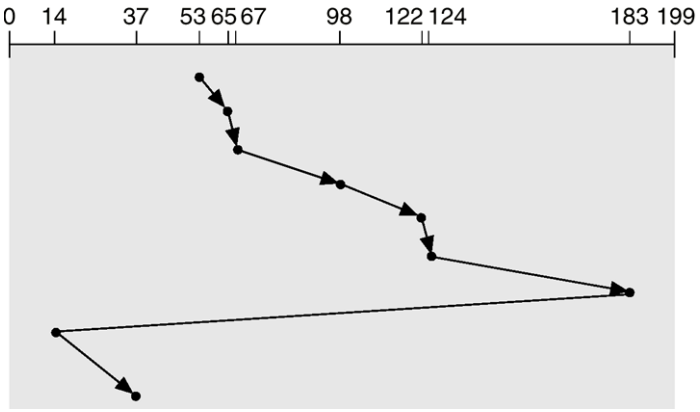
1. 含义：区别于SCAN算法，转向/返回的时候不需要回到磁盘最内/最外端

2. 示例

·Queue=98,183,37,122,14,124,65,67
·head stars at 53
·total head movement of 208 cylinders.



queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53



3.4. 磁盘高速缓存

1. 含义：用内存空间来暂存磁盘数据，物理属于内存，逻辑属于磁盘

2. 形式：

   1. 内存中开辟一个单独的**大小固定的**空间
   2. 把所有未利用内存空间变为一个**缓冲池**

3. 数据交付：如何将磁盘Cache数据给请求的进程

   1. 数据交付：Cache数据→请求进程的内存工作区
   2. 指针交付：将指向Cache某区域的指针，交给请求进程

4. 置换算法：LRU