《操作系统》期末速通教程

4. 文件管理

4.1 文件

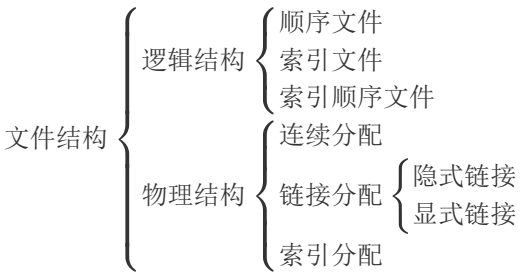
[**文件**]

(1) 定义: **文件**是具有文件名的若干相关元素的集合.

(2) 组成: 自底向上依次为, 数据项、记录、文件.

(3) 属性: 类型、长度、物理位置、建立时间等.

(4) 结构:



① 文件的**逻辑结构**: 即文件的组织结构.

② 文件的**物理结构**: 即文件的存储结构, 是文件在外存上的存储和组织形式.

(5) 分类:

① 按逻辑结构分类: 有结构文件、无结构文件.

② 按用途分类: 系统文件、用户文件、库文件.

③ 按数据形式分类: 源文件、目标文件、可执行文件.

④ 按存取属性分类: 只执行、只读、可读/写.

[**注 1**] 文件的物理结构和逻辑结构间无明显的制约或关联关系, 但若物理结构选择不慎, 很难体现出逻辑结构的优点.

[**注 2**] 对文件的访问受用户访问权限和文件属性共同限制. 注意用户优先级不是限制因素, 其作用是规定多用户同时请求该文件时应先满足谁.

[**注 3**] 文件只是一个数据结构, 本身不能提高磁盘使用效率.

[**注 4**] 文件的逻辑结构为方便用户而设计, 而文件的物理结构取决于文件系统的设计者针对硬件结构而采取的策略.

[**注 5**] 记录是文件进行存取的单位.

[**有结构文件**]

(1) 定义: 文件由若干个相关的记录组成.

(2) 例: 各种数据结构、数据库等.

(3) 分类: 顺序文件、索引文件.

[**无结构文件**]

(1) 定义: 文件是一个字符流.

(2) 长度以字节为单位.

(3) 访问时, 用读写指针指向下一个要访问的字符.

(4) 例: 源程序、可执行文件、库函数等.

4.2 文件的逻辑结构

[**顺序文件**]

(1) 策略: 文件中的记录一个接一个顺序排列, 记录可为定长记录或变长记录.

(2) 结构:

① 串结构:

(i) 策略: 记录按存入时间先后排序.

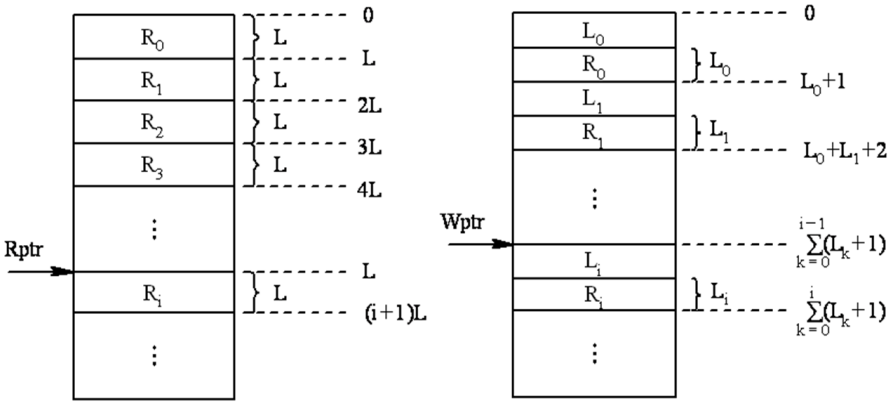
(ii) 缺点: 记录顺序与关键字无关, 检索时需顺序查找, 检索效率低.

② 顺序结构:

(i) 策略: 记录按关键字排序.

(ii) 优点: 检索时可折半查找, 检索效率高.

(3) 示意图:



(a) 定长记录文件　　(b) 变长记录文件

(4) 优点:

① 批量操作记录, 即每次读/写大批记录时, 顺序文件效率最高.

② 顺序存储设备 (如磁带) 中, 只有顺序文件能被存储并有效地工作.

(5) 缺点:

① 不方便增删改.

② 文件中的记录变长时, 检索效率低.

改进: 为可变长记录文件建立索引表, 即索引文件.

[索引文件]

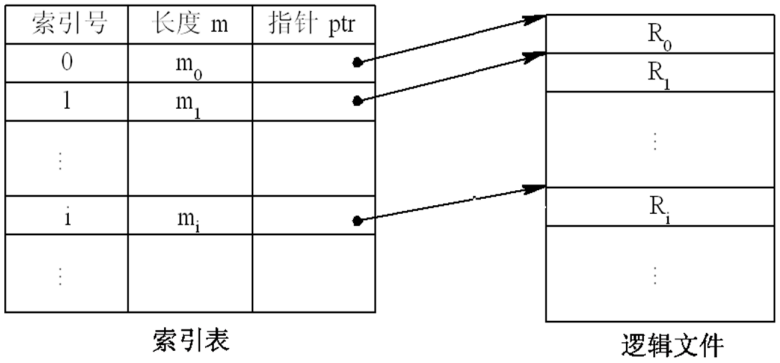
(1) 策略:

① 为可变长记录文件建立**索引表**, 索引按关键字排序.

② 索引表项的组成: 索引号、长度、在逻辑文件中的始址.

(2) 索引表本身是定长记录的顺序文件.

(3) 示意图:



(4) 优点:

① 方便增删改查.

② 检索速度快.

(5) 缺点: 索引表增加存储开销.

[索引顺序文件]

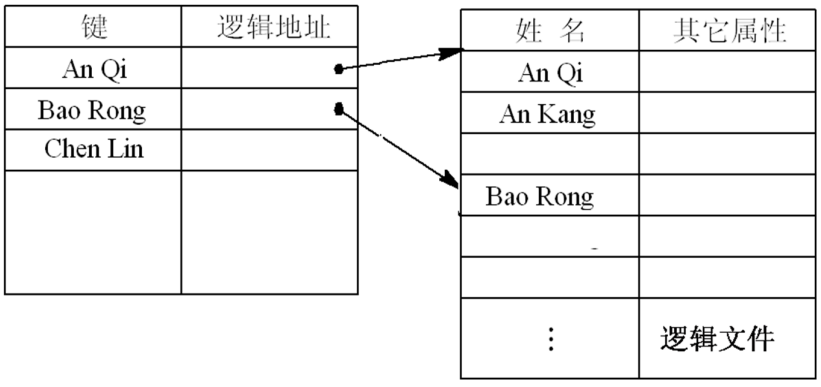
(1) 策略:

① 将记录分为若干组, 以每组的第一个记录为索引项, 建立索引.

② 索引项按记录的关键字排序, 一组内的记录可以无序, 即: 组间有序, 组内无序.

(2) 索引表本身是定长记录的顺序文件.

(3) 示意图:



(4) 优点:

① 检索速度快.

② 索引表较短, 节约存储空间.

(5) 缺点: 索引表增加存储开销.

4.3 文件的物理结构

[文件分配]

(1) 定义: **文件分配**对应文件的物理结构, 指文件如何分配磁盘块.

(2) 目标:

① 提高外存利用率.

② 提高文件访问速度.

(3) 分类:

① 连续分配.

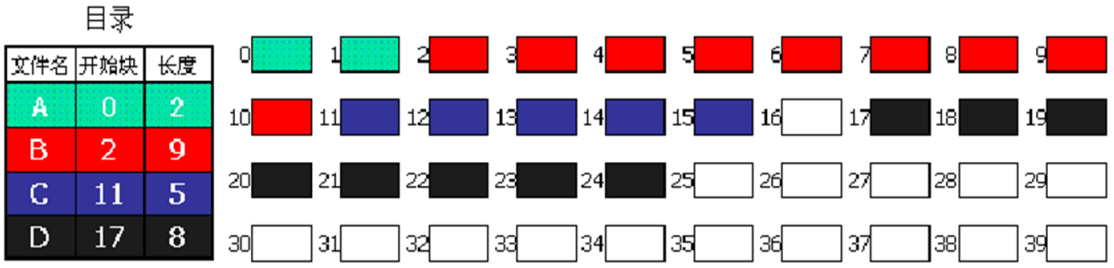
② 链接分配.

③ 索引分配.

[连续分配]

(1) 策略: 为文件分配一组相邻的盘块.

(2) 示意图:



(3) 优点:

① 支持顺序访问和随机访问.

② 顺序访问简单, 速度快.

原因: 文件占用的盘块位于一条或几条相邻的磁道上, 磁头移动距离最小,

即寻道数和寻道时间都最小.

(4) 缺点:

① 需有连续的存储空间.

② 最易产生外部碎片, 磁盘利用率低.

③ 需事先知道文件的长度, 也无法实现文件的动态增长.

④ 为保持文件的有序性, 插入或删除记录时, 需对相邻的记录作物理上的移动.

[链接分配]

(1) 策略: 采用离散分配方式, 将文件的各盘块通过指针链接.

(2) 优点:

① 无需连续的存储空间.

② 无外部碎片, 磁盘利用率高.

③ 无需事先知道文件的长度, 便于动态地为文件分配盘块.

④ 实现便于文件的增删改.

(3) 分类:

① 隐式链接.

② 显式链接.

[隐式链接]

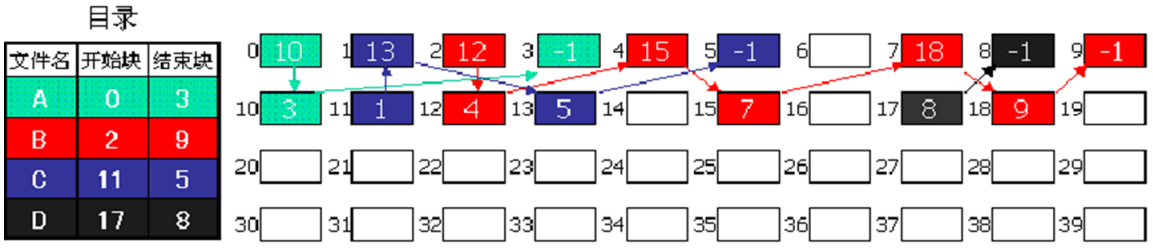
(1) 策略:

① 目录项包含文件的起始块号和结束块号.

② 每个文件对应一个盘块的链表.

③ 除文件的最后一个盘块外, 每个盘块都存有指向下一盘块的指针, 即指针在盘块中.

(2) 示意图:



(3) 缺点:

① 只支持顺序访问, 不支持随机访问.

② 可靠性差, 盘块中任一指针出错都会导致文件数据丢失.

③ 指向下一盘块的指针占用数据区, 导致数据区的大小可能非 2 的幂次, 不利于与内存页对应.

[**显式链接**]

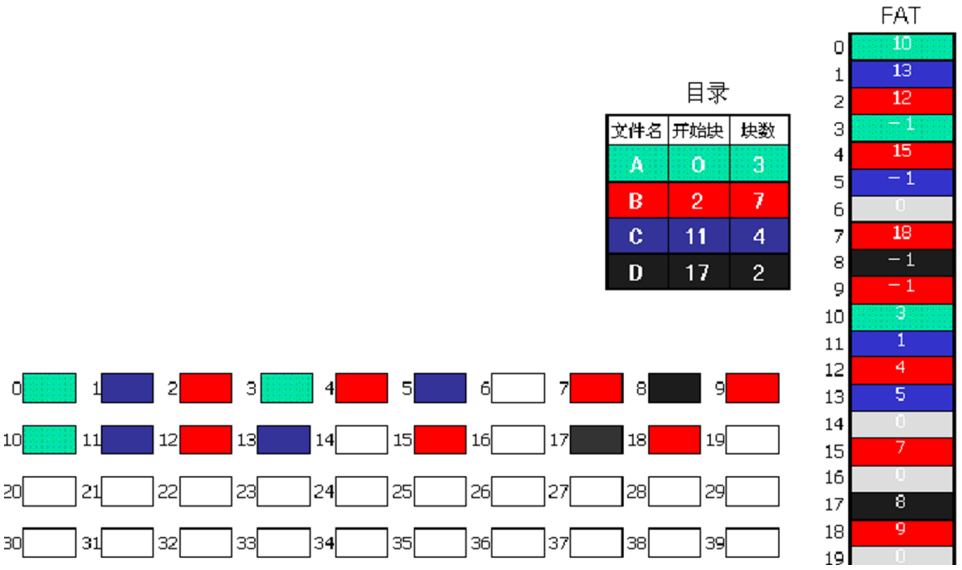
(1) 策略:

① 在磁盘中设置一张**文件分配表** (File Allocation Table, FAT) , 每个表项存放指向下一盘块的指针.

② 将链接文件的各盘块的指针显式地存放在 FAT 中.

③ 目录项包含文件的文件名和起始块号, 后续块号通过查 FAT 得到. (文件的块数不是必须的.)

(2) 示意图:



(3) 优点:

① 支持顺序访问和随机访问.

② FAT 在系统启动时即读入内存, 检索在内存中进行, 减少访存次数和访盘次数, 效率高.

(4) 缺点:

① FAT 占用额外的内存空间.

② 不支持高速随机存取.

改进: 注意到读某个文件时, 只需将该文件对应的盘块号调入内存, 无需将整个 FAT 调入内存, 故可将每个文件的盘块号集中存放, 即索引分配.

[**注**] FAT 必须用数组实现, 因为需要支持随机存取.

[索引分配]

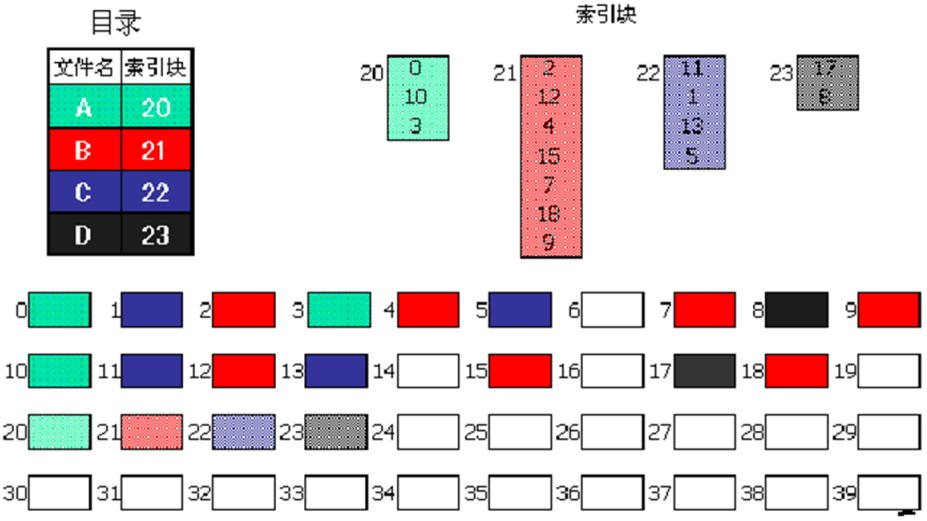
(1) 策略: 为每个文件建立索引表, 将文件的所有盘块记录在索引表中.

(2) 分类:

① 一级索引:

(i) 策略: 为每个文件建立一张索引表.

(ii) 示意图:



(iii) 优点:

i) 支持顺序访问和随机访问.

ii) 无外部碎片.

(iv) 缺点:

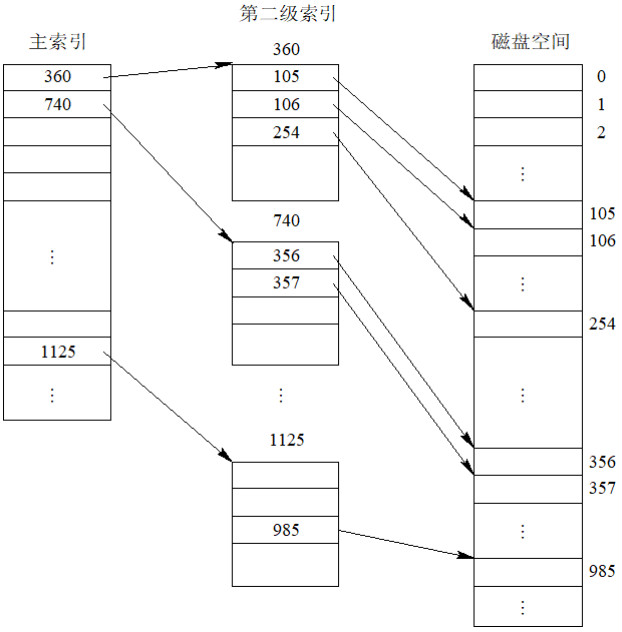
i) 占用额外的存储空间.

ii) 大量的小文件使用大量索引, 浪费空间.

② 多级索引:

(i) 策略: 文件较大而索引块较多时, 为索引块建立索引.

(ii) 示意图:



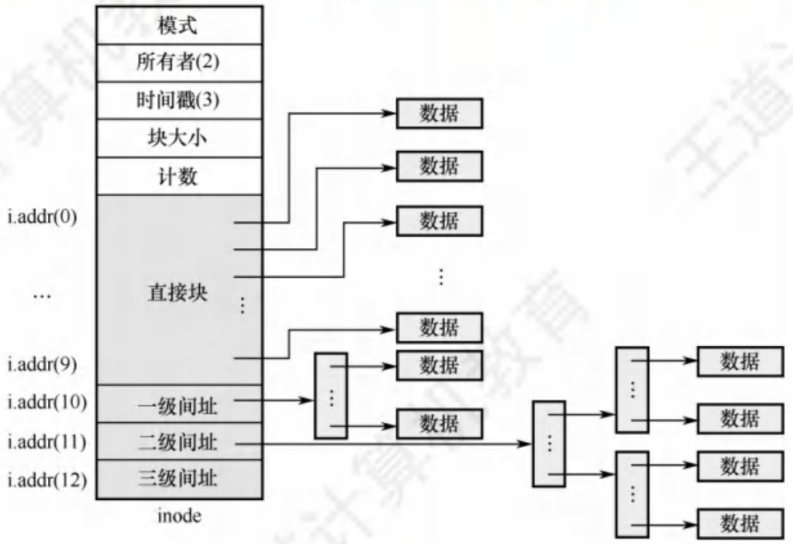
(iii) 优点: 加快大文件的检索速度.

(iv) 缺点: 访问一个盘块时, 启动磁盘的次数随索引级数的增加而增加.

③ 混合索引:

(i) 策略: 将索引分为直接地址、一次间接地址和多次间接地址.

(ii) 示意图:



(iii) 优点: 照顾到不同大小的文件.

[**注**] 采用索引分配时, 单个文件长度的决定因素: 地址项的个数、间接地址级数、文件块大小.

[**例**] 某记录文件采用隐式链接分配, 其逻辑记录的固定长度为 $100\ \mathrm{B}$, 在磁盘上存储时采用记录成组分解技术 (即将若干条逻辑记录存入一个块中, 一条逻辑记录不能横跨两块) , 盘块大小为 $512\ \mathrm{B}$. 若该文件的目录项已读入内存, 求修改 22 号 (下标从 1 开始) 逻辑记录的访盘次数.

[**解**] 每个盘块存 $\left\lfloor \dfrac{512\ \mathrm{B}}{100\ \mathrm{B}} \right\rfloor = 5$ 条逻辑记录, 剩下的 $512\ \mathrm{B} - 5 \times 100\ \mathrm{B} = 12\ \mathrm{B}$ 存放指向下一盘块的指针.

22 号逻辑记录在 $\left\lceil \dfrac{22}{5} \right\rceil = 5$ 号盘块中, 采用链接分配, 需顺序查找 5 次, 访盘 5 次.

修改该盘块后, 需将修改内容写回磁盘, 而该文件的目录项已读入内存, 即已知该块在磁盘上的位置,

故还需访盘 1 次.

综上, 访盘 6 次.

[**例**] 某文件有 $L_1 \sim L_8$ 的 8 个记录, 采用隐式链接分配, 每个记录和链接指针占一个盘块, 主存中的磁盘缓冲区的大小等于磁盘块的大小. 若该文件的目录已读入内存, 求在 L_5 和 L_6 间插入一个记录 L_x (已在内存中) 的读盘次数和写盘次数.

[**解**] 先依次读 $L_1 \sim L_5$ 盘块, 并根据 L_5 盘块的指针得到 L_6 盘块的地址, 读盘 5 次.

将 L_x 写入一个空闲盘块, 并将该盘块的指针指向 L_6 盘块, 写盘 1 次.

修改 L_5 盘块的指针指向 L_x 盘块, 写盘 1 次.

综上, 读盘 5 次, 写盘 2 次.

[**例**] 某 OS 采用混合索引方式为文件分配外存, 索引节点含 13 个地址项, 其中直接地址 10 个, 一次间接地址 1 个, 二次间接地址 1 个, 三次间接地址 1 个. 设盘块大小为 $4\ \mathrm{KB}$, 每个盘块号占 $4\ \mathrm{B}$. 求存储一个 $5\ \mathrm{GB}$ 的文件实际占用磁盘空间的大小.

[**解**]

(1) 求各级索引支持的最大文件大小.

每个索引块中存放 $\dfrac{4\ \mathrm{KB}}{4\ \mathrm{B}} = 1\ \mathrm{K}$ 个盘块号.

直接索引支持 $10 \times 4\ \mathrm{KB} = 40\ \mathrm{KB}$.

一级索引支持 $1024 \times 4\ \mathrm{KB} = 4\ \mathrm{MB}$.

二级索引支持 $(1024)^2 \times 4\ \mathrm{KB} = 4\ \mathrm{GB}$.

三级索引支持 $(1024)^3 \times 4\ \mathrm{KB} = 4\ \mathrm{TB}$.

(2) 求文件本身占用的块数.

文件本身占用 $\dfrac{5\ \mathrm{GB}}{4\ \mathrm{KB}} = 1310720 = 1280\ \mathrm{K}$ 块.

(3) 求文件占用的各级索引块数.

① $5\ \mathrm{GB} > 40\ \mathrm{KB}$, 需占用 10 个直接索引中的所有盘块, 即 $\dfrac{10 \times 1}{1\ \mathrm{K}} \approx 0$ 块.

同理一级索引、二级索引、三次索引本身都可认为占用 0 块.

或认为这部分占索引节点的大小, 不额外计入占用磁盘大小.

② $5\ \mathrm{GB} - 40\ \mathrm{KB} > 4\ \mathrm{MB}$, 需占用 1 个一级索引中的所有直接索引, 即 $\dfrac{1 \times 1024}{1\ \mathrm{K}} = 1$ 块.

③ $5\ \mathrm{GB} - 40\ \mathrm{KB} - 4\ \mathrm{MB} > 4\ \mathrm{GB}$, 需占用 1 个二级索引中的所有一级索引,

加上二级索引指向的 1 块一级索引, 共 $\dfrac{1 \times (1024)^2}{1\ \mathrm{K}} + 1 = 1025$ 块.

④ $s = 5\ \mathrm{GB} - 40\ \mathrm{KB} - 4\ \mathrm{MB} - 4\ \mathrm{GB} = 1\ \mathrm{GB} - 40\ \mathrm{KB} - 4\ \mathrm{MB} < 4\ \mathrm{TB}$,

需占用 1 个三级索引中的部分二级索引.

(i) 三级索引中的 1 个二级索引支持的最大文件大小 $4\ \mathrm{GB} > s$,

故需占用 1 个二级索引中的部分一级索引.

(ii) 二级索引中的 1 个一级索引支持的最大文件大小 $4\ \mathrm{MB} < s$,

故需占用 $\left\lceil \dfrac{s}{4\ \mathrm{MB}} \right\rceil = 255$ 块一级索引.

共 $1 + 1 + 255 = 257$ 块.

(4) 求文件实际占用空间.

总块数 $1280\ \mathrm{K} + 1 + 1025 + 257 = 1312003$ 块.

实际占用大小 $1312003 \times 4\ \mathrm{KB} \approx 5125\ \mathrm{MB} \approx 5.005\ \mathrm{GB}$.

[**例**] 某文件存放在 100 个块中. 若管理文件所需的 PCB 、索引块、索引信息等都已在内存中, 判断下列操作是否需要磁盘 I/O .

[1] 采用单级索引分配, 将最后一个数据块插入文件头.

[2] 采用隐式链接分配, 将最后一个数据块插入文件头.

[**解**]

[1] 因索引块已在内存中, 故无需磁盘 I/O .

[2] 链接分配只支持顺序访问, 故需先用磁盘 I/O 将文件读入内存后, 先找到最后一个数据块的位置, 再修改指针.

4.4 目录管理

[**文件控制块**, **File Control Block**, **FCB**]

(1) 定义: **文件控制块**是用于存放控制文件所需的各种信息的数据结构, 用于实现 "按名存取" .

(2) 一个文件与一个 FCB 一一对应, FCB 的有序集合称为**文件目录**, 一个 FCB 是一个**文件目录项**.

(3) 示意图:

文件名	扩展名	属性	备用	时间	日期	第一块号	盘块数

主要组成:

① 基本信息: 文件名、物理位置、逻辑结构、物理结构等.

② 存取控制信息: 存取权限等.

③ 使用信息: 文件的建立时间、上次修改时间等.

(4) 文件首次打开时, OS 将该文件的 PCB 读入内存.

(5) 问题: 文件很多时, 文件目录占用大量盘块, 检索效率低.

注意到查找目录时只涉及文件名, 故只需将文件名调入内存.

改进: **索引节点**.

[**注 1**] 文件首次打开用 open() 系统调用, 其参数包含文件名. 此后对文件的访问用 read() 系统调用, 其参数包含 open() 返回的文件标识符, 不再使用文件名作为参数.

[**注 2**] 多个进程打开同一文件时, 得到的进程标识符相互独立.

[**注 3**] 文件系统可存储的文件数受限于 FCB 数.

[**索引节点**]

(1) 策略:

① 将文件名和**文件描述信息**分离, 文件描述信息单独形成一个**索引节点**.

② 文件目录中的每个目录项由文件名和索引节点编号组成.

(2) 示意图:

文件名	索引节点编号
文件名1	
文件名2	
⋮	
⋮	

(3) 分类:

① 磁盘索引节点:

(i) 定义: 存放在磁盘上的索引节点.

(ii) 每个文件有唯一的磁盘索引节点.

② 内存索引节点:

(i) 定义: 存放在内存中的索引节点.

(ii) 打开文件时, 将磁盘索引节点复制到内存索引节点中.

[**注**] 系统能创建的文件数量上限为索引节点数.

[**例**] 某系统的盘块大小为 $1\ \mathrm{KB}$, FCB 大小为 $64\ \mathrm{B}$. 某目录中包含 640 个文件. 求分别采用如下策略时的平均访问盘块数.

[1] 无索引节点.

[2] 有索引节点, 文件目录中的每个目录项大小为 $16\ \mathrm{B}$.

[**解**] 每个盘块存 $\dfrac{1\ \mathrm{KB}}{64\ \mathrm{B}} = 16$ 个 FCB , 640 个文件占 $\dfrac{640}{16} = 40$ 个盘块.

[1] 无索引节点时, 平均访问 $\dfrac{40}{2} = 20$ 个盘块.

[2] 有索引节点时, 每个盘块存 $\dfrac{1\ \mathrm{KB}}{16\ \mathrm{B}} = 64$ 个目录项,

则 640 个文件占 $\dfrac{640}{64} = 10$ 个盘块, 平均访问 $\dfrac{10}{2} = 5$ 个盘块.

[**目录管理**]

(1) 工作:

① 实现 "按名存取" .

② 提高目录检索速度.

③ 实现文件共享.

④ 允许文件重名: 允许不同用户对不同文件使用相同的文件名.

(2) 分类:

① 单级目录.

② 两级目录.

③ 多级目录.

(3) 目录检索的性能的决定因素:

① 目录项数: 影响平均比较次数.

② 目录项大小: 影响占用盘块数, 进而影响磁盘 I/O 时间.

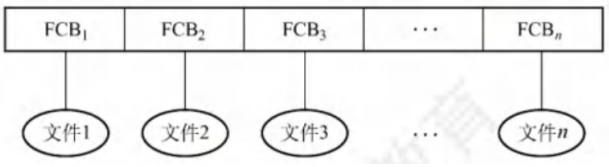
③ 目录项在目录中的位置: 影响检索路径.

[**注**] 文件检索得到文件的逻辑地址.

[**单级目录**]

(1) 策略: 在文件系统中只建立一张目录表, 每个文件占一个目录项.

(2) 示意图:



(3) 优点:

① 实现简单.

② 实现 "按名存取" .

(4) 缺点:

① 查找速度慢.

② 不允许文件重名.

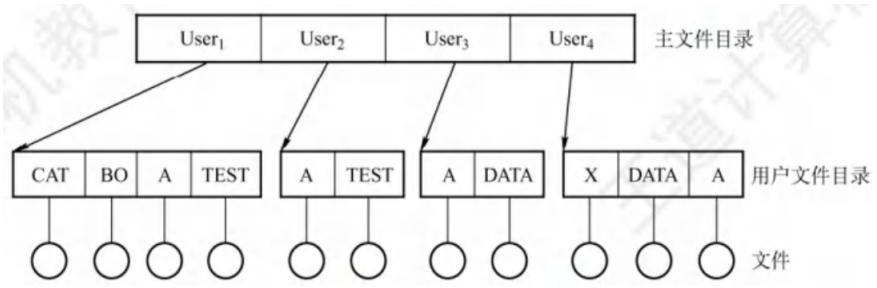
③ 不便于实现文件共享.

④ 不适用于多用户 OS .

[两级目录]

(1) 策略: 为每个用户建立单级目录.

(2) 示意图:



(3) 优点:

① 提高检索速度.

② 允许文件重名, 即不同用户目录下可使用相同的文件名.

③ 便于文件共享.

④ 可在目录上实现访问限制.

(4) 缺点:

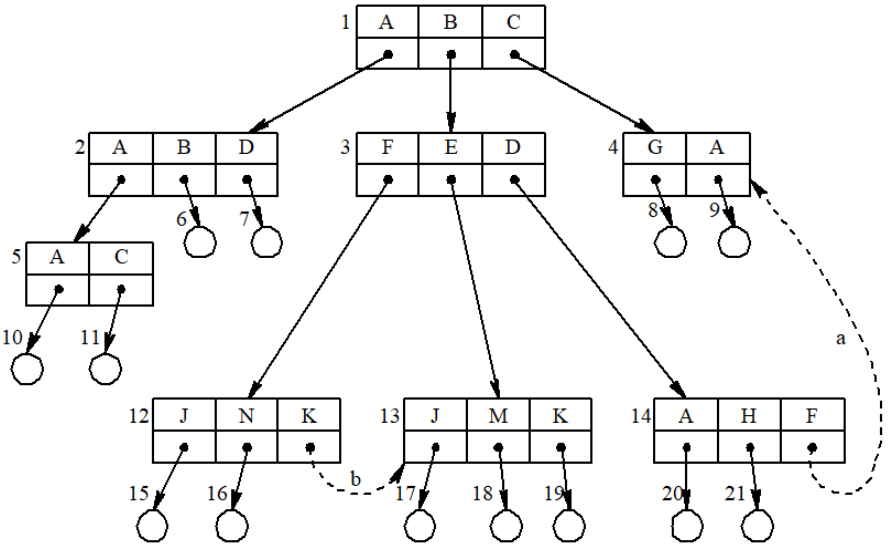
① 缺乏灵活性.

② 不能对文件分类.

[多级目录]

(1) 策略: 构造目录树.

(2) 示意图:



(3) 目录查询技术:

① 线性检索, 即遍历树.

② 哈希.

4.5 外存管理

[**文件系统**]

(1) 定义: OS 中负责管理和存储文件信息的软件机构.

(2) 功能: 管理外存上的文件, 为用户提供文件的存取、共享、保护等功能.

(3) 主要目标: 提高外存利用率.

(4) 组成: 管理程序、数据.

(5) 提高文件系统性能的方法:

① 目录项分解: 将目录项分为**符号目录项**和**基本目录项**, 查找文件时只需查符号目录项.

② 文件高速缓存.

③ 采用高效的磁盘调度算法.

[**注 1**] 注意异步 I/O 提高 CPU 利用率, 不能减少磁盘 I/O 次数, 故不能提高文件系统的性能.

[**注 2**] 管理文件的存储空间的本质是管理外存空闲区.

[**注 3**] 文件系统与磁盘的关系:

① 文件系统的组织信息存在磁盘上, 这些信息和代码共同组成文件系统.

② 文件系统只占用磁盘的一部分空间, 一个磁盘上可有多个文件系统, 也可无文件系统.

[**注 4**] 文件系统未必依赖磁盘, 也可存在其它的存储介质, 如光盘、闪存等.

[**外存管理方法**]

① 空闲表法.

② 空闲链表法.

③ 位示图法.

④ 成组链接法.

[**注**] 还可以用 FAT 管理空闲盘块, 即用特殊的数值表示盘块的分配情况.

[**空闲表法**]

(1) 策略:

① 为外存上的所有空闲分区建立**空闲表**, 每个空闲区对应一个**空闲表项**.

② 将空闲区按起始块号升序排列.

③ 盘块的分配与回收:

(i) 分配盘块: 类似于内存的动态分配.

(ii) 回收盘块: 类似于内存的回收.

(2) 属于连续分配方式.

(3) 优点:

① 分配速度快.

② 减少访问磁盘的 I/O 频率.

③ 适用于空间对换或小文件系统.

[**空闲链表法**]

(1) 策略: 用链表维护空闲盘区.

(2) 分类:

① **空闲盘块链**:

(i) 策略:

i) 将空闲盘区以盘块为单位连成链表.

ii) 空间分配: 从链首开始分配盘块.

iii) 空间回收: 将释放的盘块挂回链尾.

(ii) 优点: 分配和回收实现简单.

(iv) 缺点:

i) 为一个文件分配盘块时可能需操作多次, 效率低.

ii) 空闲盘块链很长.

② **空闲盘区链**:

(i) 策略:

i) 将空闲盘区以盘区为单位连成链表.

ii) 空间分配: 沿链表搜索, 采用首次适应算法.

iii) 空间回收: 将释放的盘区插入链表的合适位置, 可能需合并相邻盘区.

(ii) 优点:

i) 分配和回收效率高.

ii) 空闲盘区链短.

(iii) 缺点: 分配和回收实现复杂.

[**位示图法**]

(1) 策略: 用一个二进制位表示一个盘块的使用情况, 0 表示空闲, 1 表示已分配.

(2) 示意图:

	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																

(3) 分配盘块步骤:

① 扫描**位示图**, 依次找到值为 0 的盘块并分配.

② 位示图第 i 行、第 j 列的位对应的盘块号 b 的计算: (n 是每行的位数)

(i) 若盘块号和位示图的位号都从 0 开始, 则 $b = n \cdot i + j$.

(ii) 若盘块号和位示图的位号都从 1 开始, 则 $b = n \cdot (i - 1) + j$.

③ 修改位示图.

(4) 回收盘块步骤:

① b 号盘块对应位示图中的位置 (i, j) 的计算: (n 是每行的位数)

(i) 若盘块号和位示图的位号都从 0 开始, 则 $\begin{cases} i = \left\lfloor \dfrac{b}{n} \right\rfloor \\ j = b \bmod n \end{cases}$.

(ii) 若盘块号和位示图的位号都从 1 开始, 则 $\begin{cases} i = \left\lfloor \dfrac{b-1}{n} \right\rfloor + 1 \\ j = (b-1) \bmod n \end{cases}$.

② 修改位示图.

(5) 优点:

① 易找到一组相邻接的空闲盘块.

② 位示图小, 可保存在内存, 进而节省启动磁盘的开销.

(6) 缺点: 位示图的大小随磁盘容量的增大而增大, 常用于小型计算机.

[**例**] 某容量为 $10\ \mathrm{GB}$ 的磁盘以簇为单位, 簇的大小为 $4\ \mathrm{KB}$. 采用位示图法管理该分区的空闲空间, 求存放位示图所需的簇数.

[**解**] 磁盘有 $\dfrac{10\ \mathrm{GB}}{4\ \mathrm{KB}} = 2.5\ \mathrm{MB}$ 簇, 则需 $2.5\ \mathrm{Mbit}$ 个二进制位,

即 $\dfrac{2.5\ \mathrm{M}}{8} = 320\ \mathrm{KB}$, 需 $\dfrac{320\ \mathrm{KB}}{4\ \mathrm{KB}} = 80$ 簇.

[**例**] 某文件系统用位示图法表示磁盘空间的分配情况, 位示图位于磁盘的 $32 \sim 127$ 号块, 每个盘块的大小为 $1024\ \mathrm{B}$, 盘块和块内字节号都从 0 开始编号. 求释放 409612 号盘块时, 位示图中需要修改的位所在的盘块号和块内字节号.

[**解**] 块号偏移 $= \left\lfloor \dfrac{409612}{1024 \times 8} \right\rfloor = 50$, 块号 $=$ 起始块号 $+$ 块号偏移 $= 32 + 50 = 82$.

要求块内字节号, 需再除以 8 , 即 块内字节 $= \left\lfloor \dfrac{\text{块号} \bmod (1024 \times 8)}{8} \right\rfloor = 1$.

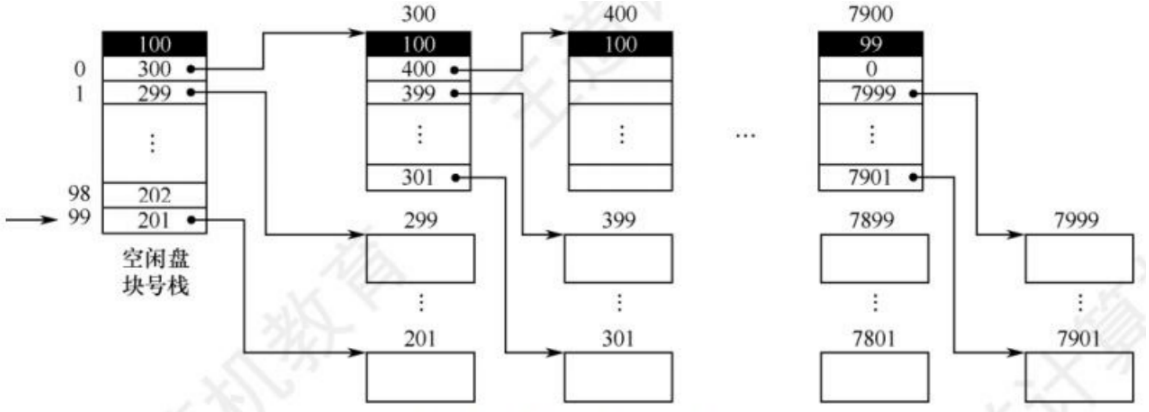
[**成组链接法**]

(1) 策略:

① 将空闲盘块分为若干组, 每组最后一个盘块存放下一组的空闲盘块数和空闲盘块号.

② 用**空闲盘块栈**存放当前可用的一组空闲盘块号.

(2) 示意图:



(3) 优点:

① 适用于大型文件系统.

② 空闲盘块栈短, 可保存在内存.

③ 盘块的分配和回收均匀, 减少磁道磨损.

[**注 1**] 分配时, 每组的最后一个盘块将下一组的空闲盘块压入空闲盘块栈后, 本身也作为空闲盘块分配.

[**注 2**] 空闲链表法比成组链接法更难得到连续的空闲区, 原因:

① 空闲链表法适用于离散分配.

② 成组链接法是连续分配和离散分配的结合.

[**例**] 某系统采用成组链接法管理磁盘空闲空间. 设空闲盘块栈长度为 5 , 每 4 个盘块为一组. 某时刻空闲盘块栈的内容为 3 (栈长)、91 、81 、45 , 其中 91 号块的管理信息为 4 (栈长)、21 、35 、77 、68 .

[1] 现分配 5 个空闲块, 写出被分配的块号.

[2] 在 [1] 的基础上, 回收三个块 189 、55 、62 , 写出空闲盘块栈的变化过程.

[**解**]

[1] 先分配 45 号块和 81 号块, 然后将下一组压入栈中, 再分配 91 号块、68 号块和 77 号块.

[2] 将 189 号块和 55 号块压入栈, 然后将此时的栈信息 $(4, 21, 35, 189, 55)$ 打包到 62 号块, 清空栈, 再压入 62 号块.

4.6 文件共享

[**文件共享**]

分类:

(1) **基于索引节点的共享 (硬链接)** : 文件共享索引节点.

(2) **基于符号链的共享 (软链接)** : 类似于快捷方式.
