

Lecture 8: 知识 - 文件

考试大纲

四、文件管理

- (一)文件
 - 1. 文件的基本概念
 - 2. 文件元数据和索引节点 (`inode`)
 - 3. 文件的操作
 - 建立, 删除, 打开, 关闭, 读, 写
 - 4. 文件的保护
 - 5. 文件的逻辑结构
 - 6. 文件的物理结构
- (二)目录
 - 1. 目录的基本概念
 - 2. 树形目录
 - 3. 目录的操作
 - 4. 硬链接和软链接
- (三)文件系统
 - 1. 文件系统的全局结构 (`layout`)
 - 2. 文件系统在外存中的结构, 文件系统在内存中的结构 2. 外存空闲空间管理办法
 - 3. 虚拟文件系统
 - 4. 文件系统挂载 (`mounting`)

文件的基本概念

文件元数据

概念: 文件的属性

- 文件名
 - 由创建文件的用户决定文件名, 主要是为了方便用户找到文件, 同一目录不允许有重名文件;
- 标识符
 - 一个系统内的各**文件标识符唯一**, 对用户来说毫无可读性, 因此标识符只是操作系统用于区分各个文件的一种内部名称;
- 类型
 - 指明文件的类型;
- 位置
 - 文件存放的路径 (让用户使用), 在外存中的地址 (操作系统使用, 对用户不可见);
- 创建者

- 所有者
- 保护信息
 - 对文件进行保护的访问控制信息 (权限);
 - 不同分组、不同用户的读写执行权限;

概念：文件控制块

- 概念：
 - 文件控制块 (File Control Block, FCB) 是用来存放控制文件需要的各种信息的数据结构, 以实现按名存取;
 - 文件与 FCB 一一对应, FCB 的有序集合称为文件目录, 一个 FCB 就是一个文件目录项;
 - 通常, 一个文件目录也被视为一个文件, 称为目录文件;
 - 每当创建一个新文件, 系统就要为其建立一个 FCB, 用来记录文件的各种属性
- 图示:

文件名
类型
文件权限 (读, 写)
文件大小
文件数据块指针

图 4.1 一个典型的 FCB

文件名	索引节点编号
文件名1	
文件名2	
⋮	
⋮	

图 4.2 UNIX 的文件目录结构

索引结点

概念：索引节点

- 概念：
 - 文件目录通常存放在磁盘上, 当文件很多时, 文件目录会占用大量的盘块;
 - 在查找目录的过程中, 要先将存放目录文件的第一个盘块中的目录调入内存, 然后用给定的文件名逐一比较, 若未找到指定文件, 就还需要不断地将下一盘块中的目录项调入内存, 逐一比较;
 - 我们发现, 在检索目录的过程中, **只用到了文件名**, 仅当找到一个目录项 (其中的文件名与要查找的文件名匹配) 时, 才需从该目录项中读出该文件的物理地址;
 - 也就是说, 在检索目录时, 文件的其他描述信息不会用到, 也不需要调入内存。因此, 有的系统 (如 UNIX) 便采用了 **文件名和文件描述信息分离** 的方法, 使文件描述信息单独形成一个称为索引节点的数据结构, 简称 i 节点 (inode);
 - 在文件目录中, 每个 **目录项仅由文件名和相应的索引节点号** (或索引节点指针) 构成;
- 分析: 索引节点号与文件容量的关系
 - 假设一个 FCB 为 64B, 盘块大小是 1KB, 则每个盘块中可以存放 16 个 FCB (FCB 必须连续存放);
 - 若一个文件目录共有 640 个 FCB, 则查找文件平均需要启动磁盘 20 次;
 - 而在 UNIX 系统中, 一个目录项仅占 16B, 其中 14B 是文件名, 2B 是索引节点号;

- 在 1KB 的盘块中可存放 64 个目录项，这样可使查找文件的平均启动磁盘次数减少到原来的 1/4，大大节省了系统开销；
- 分类：两种索引节点
 - 第一类：磁盘索引结点

是指存放在磁盘上的索引节点。每个文件有一个唯一的磁盘索引节点，主要包括以下内容：

 - 文件主标识符，拥有该文件的个人或小组的标识符。
 - 文件类型，包括普通文件、目录文件或特别文件。
 - 文件存取权限，各类用户对该文件的存取权限。
 - 文件物理地址，每个索引节点中含有 13 个地址项，即 `iaddr(0)~iaddr(12)`，它们以直接或间接方式给出数据文件所在盘块的编号（见 4.1.6 节）。
 - 文件长度，指以字节为单位的文件长度。
 - 文件链接计数，在本文件系统中所有指向该文件的文件名的指针计数。
 - 文件存取时间，本文件最近被进程存取、修改的时间及索引节点最近被修改的时间。
 - 第二类：内存索引结点

是指存放在内存中的索引节点。当文件被打开时，要将磁盘索引节点复制到内存的索引节点中，便于以后使用。在内存索引节点中增加了以下内容：

 - 索引节点号，用于标识内存索引节点。
 - 状态，指示 i 节点是否上锁或被修改。
 - 访问计数，每当有一进程要访问此 i 节点时，计数加 1；访问结束减 1。
 - 逻辑设备号，文件所属文件系统的逻辑设备号。
 - 链接指针，设置分别指向空闲链表和散列队列的指针。

文件名与文件描述符

概念：文件描述符

- 概念：
 - 文件名不必是打开文件表的一部分，因为一旦完成对 FCB 在磁盘上的定位，系统就不再使用文件名；
 - 对于访问打开文件表的索引号，UNIX 称之为文件描述符，而 Windows 称之为文件句柄；
 - 因此，只要文件未被关闭，所有文件操作都是通过文件描述符（不是文件名）来进行；
- 注意：
 - 只要完成了文件打开 `open()` 系统调用，后面再使用 `read()`、`write()`、`lseek()`、`close()` 等文件操作的系统调用，就不再使用文件名，而使用文件描述符；

概念：打开中的文件的关联信息

- 文件指针：
 - 系统跟踪上次的读写位置作为当前文件位置的指针，这种指针对打开文件的某个进程来说是唯一的，因此必须与磁盘文件属性分开保存；
- 文件打开计数：
 - 计数器跟踪当前文件打开和关闭的数量；
 - 因为多个进程可能打开同一个文件，所以系统在删除打开文件条目之前，必须等待最后一个进程关闭文件；
- 文件磁盘位置：

- 大多数文件操作要求系统修改文件数据；
- 查找磁盘上的文件所需的信息保存在内存中，以便系统不必为每个操作都从磁盘上读取该信息；
- 访问权限：
 - 每个进程打开文件都需要有一个访问模式（创建、只读、读写、添加等）；
 - 该信息保存在进程的打开文件表中，以便操作系统能够允许或拒绝后续的 **IO** 请求；

文件的大小与访问速度

概念：文件的大小

- 文件的总数：
 - 和索引节点总数有关；
 - 一个文件对应一个索引结点，故索引结点的总数等于文件总数，而与单个文件长度无关；
 - 并且索引结点的位数，决定了文件系统能够拥有的最大文件数量；
- 影响文件长度的因素：
 - （1）间接地址索引的级数：决定了存放单个文件地址的索引块最大数目；
 - （2）地址项的个数：决定单个文件可占用文件块数目；
 - （3）文件块大小：地址项的个数与文件块大小相乘即为文件长度；

概念：文件访问速度

- **真题：** 2018年
- （1）提前读：
 - 读取当前盘块时，和之后的一些盘块一起读取并缓存，之后的盘块内容可能会在之后被请求读取，加快了访问速度；
- （2）为文件分配连续的簇：
 - 为文件分配连续的簇，在读文件时可以顺序的读取，不必查找下一个盘块位置；
- （3）延迟写：
 - 延迟写是 **将数据写入缓冲区**，并在缓存首部设置**延迟写标志**；
 - 然后将此缓冲区放入 **空闲缓冲区链表** 的尾部；
 - 当有进程申请空闲缓冲区而得到该缓冲区时，才将该缓冲区内容写入磁盘，从而减少了不必要的访问磁盘操作；
- （4）访问磁盘高速缓存：
 - 访问磁盘高速缓存，使得访问磁盘速度快

文件的基本操作

创建文件

概念： **Create** 系统调用

- 参数
 - 1. 所需的外存空间大小
 - 2. 文件存放路径

- 3. 文件名
- 实现
 - 1. 在外存中找到文件所需的空間，在外存中分配所需的空間；
 - 2. 根据文件存放路径的信息找到该目录对应的目录文件，在目录文件中**创建该文件对应的目录项**；
 - 目录项中包含了文件名，文件在外存中的存放位置等信息；

删除文件

概念： Delete 系统调用

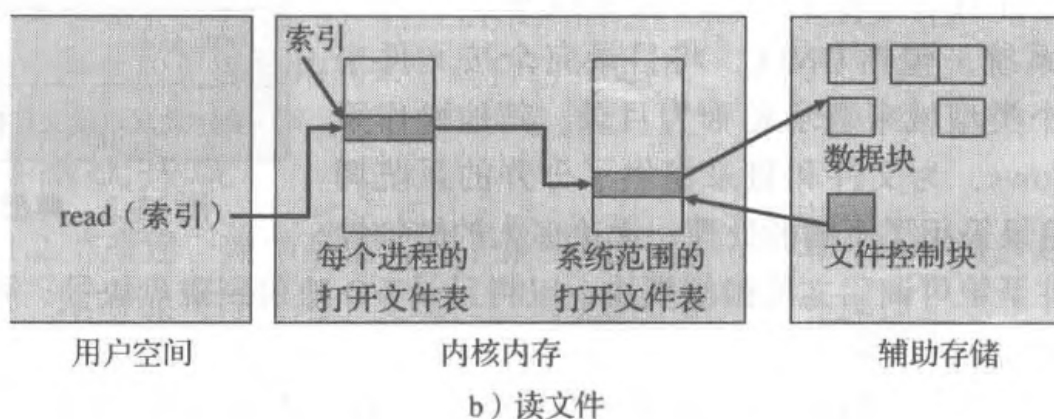
- 参数
 - 1. 文件存放路径
 - 2. 文件名
- 实现
 - 1. 根据文件名查找目录，根据存放路径找到相应的目录文件，**从目录中找到文件名对应的目录项以及文件控制块**；
 - 2. 根据该目录记录的文件在外存的存放位置、文件大小等信息，回收文件占用的磁盘块；
 - 3. 删除指定的文件控制块；
 - 4. 从目录表中删除文件对应的目录项；
- 删除的内容：
 - (1) 删除与文件相关的**目录项**；
 - (2) 删除与此文件对应的**文件控制块 (FCB)**；
 - (3) 释放与此文件相关的**内存缓冲区**；
 - (4) 释放文件占用的**磁盘空间**；
- 注意：
 - (1) 删除文件时，不会删除文件所在目录，不会一起删除文件的快捷方式；
 - (2) 删除文件**不等于**关闭文件，注意区分两者概念；

读文件

概念： Read 系统调用

- 参数
 - 1. 文件描述符 `fd`；
 - 2. `buf` 缓冲区首地址；
 - 3. 读入的字节数大小 `n`；
- 实现
 - 从读指针指向的外存中，将用户指定大小的数据读入用户指定的内存区域中；
 - **真题：** 2012年
 - 若该文件的数据不在内存中，就会产生缺页中断，则该进程进入**睡眠等待状态** (就是阻塞态)，等到所需数据从磁盘读入内存，进程才被唤醒，变为就绪态；

- 注意：
 - 读/写文件用文件描述符，即可指明文件不再需要用到文件名；
- 图示：



写文件

概念： Write 系统调用

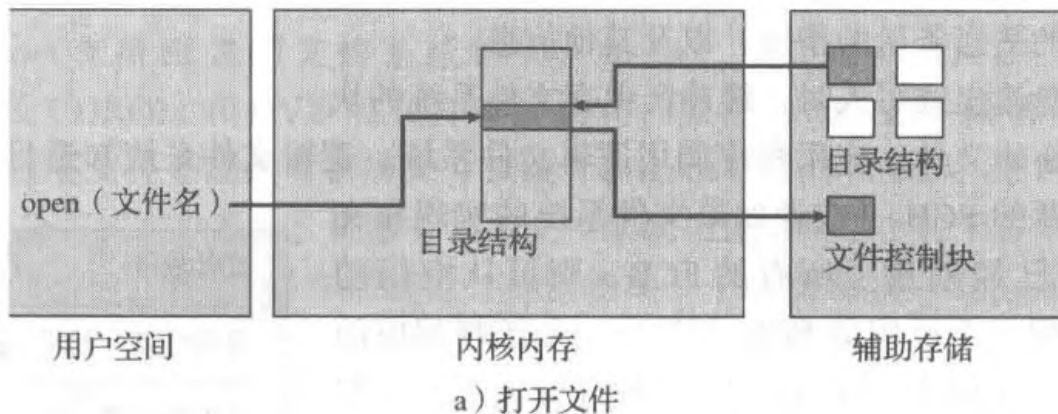
- 参数
 - 1. 哪个文件
 - 2. 写出多少数据
 - 3. 写回外存的数据放在内存中的什么位置
- 实现
 - 从用户指定的内存区域中，将指定大小的数据写回写指针指向的外存；
 - **真题：** 2014年
 - 操作系统会将文件的**数据缓冲区**的首指针返回给用户进程；
- 分析：多进程同时访问文件
 - 多个进程可以同时以“读”或“写”的方式打开文件，操作系统并不保证写操作的互斥性，进程可通过系统调用对文件加锁，保证互斥写（读者-写者问题）；

打开文件

概念： open 系统调用

- 参数
 - 1. 文件存放路径
 - 2. 文件名
 - 3. 要对文件的操作类型（如：r 只读，rw 读写等）
- 原因
 - 当用户对一个文件实施多次读/写等操作时，每次都要从检索目录开始；
 - 为了避免多次重复地检索目录，大多数操作系统要求，当用户首次对某文件发出操作请求时，须先利用系统调用 open 将该文件打开；
- 实现
 - 第一步：搜索系统的打开文件表

- 系统调用 `open` 将文件名传递到逻辑文件系统。系统调用 `open` 首先搜索整个系统的打开文件表，以便确定这个文件是否已被其他进程使用；
- 第二步：确定是否已被加载
 - 如果已被其他进程使用，则在当前的单个进程的打开文件表中创建一个条目，并让其指向现有整个系统的打开文件表；
 - 如果这个文件尚未打开，则根据给定的文件名来搜索目录结构；
- 第三步：如果未被其他进程加载，自己载
 - 部分的目录结构通常缓存在内存中，以加速目录操作；
 - 如果这个文件尚未打开，则根据给定的文件名来搜索目录结构，在找到文件后，它的 `FCB` 会复制到内存的整个系统的开放文件表中。
 - 该表不但存储 `FCB`，而且还跟踪打开该文件的**进程的数量**
- 第四步：返回指针
 - 接下来，在单个进程的打开文件表中会创建一个条目，指向整个系统打开文件表的条目的一个指针，以及其他一些域；
 - 这些域可能包含文件的当前位置的指针（用于接下来的 `read` 或 `write` 操作）和打开文件的访问模式。
 - 即：系统调用 `open` 通常返回一个指针，以**指向打开文件表的对应条目**，这个指针而不是实际的文件名，会用于所有 `IO` 操作，以避免任何进一步搜索，并简化系统调用接口；
 - 打开文件表的条目有多种名称。`UNIX` 称之为文件描述符 (`file descriptor`)，`Windows` 称之为文件句柄 (`file handle`)；
- 即：
 - 系统检索到指定文件的目录项后，将该**目录项**从外存复制到内存中的**打开文件表**的一个表目中，并将该表目的索引号（也称**文件描述符**）返回给用户；
- 当用户再次对该文件发出操作请求时，可**通过文件描述符在打开文件表中查找到文件信息**，从而节省了大量的检索开销；
 - 真题：2014年
 - 文件被用户进程首次打开的时候，执行了 `open` 系统调用，`open` 系统调用先根据传入的文件名查找相应文件，找到后将文件的 `FCB` 调入内存；
 - 当进程希望获取文件内容时，操作系统才会将文件内容读入内存；
- 当文件不再使用时，可利用系统调用 `close` 关闭它，则系统将会从内存的打开文件表中删除这一表目；
- 注意：
 - 打开文件**不会把文件数据直接读入内存**。而是将目录项中的信息复制到内存中的**打开文件表**中，并将打开文件表的**索引号 (文件描述符)**返回给用户；
- 图示：



• 真题：2024年

• 题目：

下列系统调用的实现中，包含文件按名查找功能的是()。

- A. open()
- B. read()
- C. write()
- D. close()

• 答案：

- A

• 分析：

- read、write、close 都是使用的 open 返回的文件描述符来操作文件，只有 open 是按名查找；

• 图示：打开文件表：

- 系统的打开文件表（整个系统只有一张）
- 用户进程 A 的（打开文件表）

编号	文件名	读写指针	访问权限	系统表索引号
1	...	...	...	...
2	test.txt	...	只读	k
3	...	...	...	...

用户进程A的“打开文件表”

编号	文件名	...	外存地址	打开计数器
1	...	...	...	...
...	...	...	...	...
k	test.txt	...	...	2
...	...	...	...	...

系统的打开文件表（整个系统只有一张）

编号	文件名	读写指针	访问权限	系统表索引号
1	test.txt	...	读和写	k

用户进程B的“打开文件表”

可以方便实现某些文件管理的功能。例如：在Windows系统中，我们尝试删除某个txt文件，如果此时该文件已被某个“记事本”进程打开，则系统会提示我们“暂时无法删除该文件”。其实系统在背后做的事就是先检查了系统打开文件表，确认此时是否有进程正在使用该文件。

关闭文件

概念：Close 系统调用

• 实现

- 1. 将进程的打开文件表相应表项删除（打开计数器为 0 时）
- 2. 回收分配给该文件的内存空间等资源
- 3. 系统打开文件表的打开计数器 count 减 1，若 count = 0，则删除对应表项；

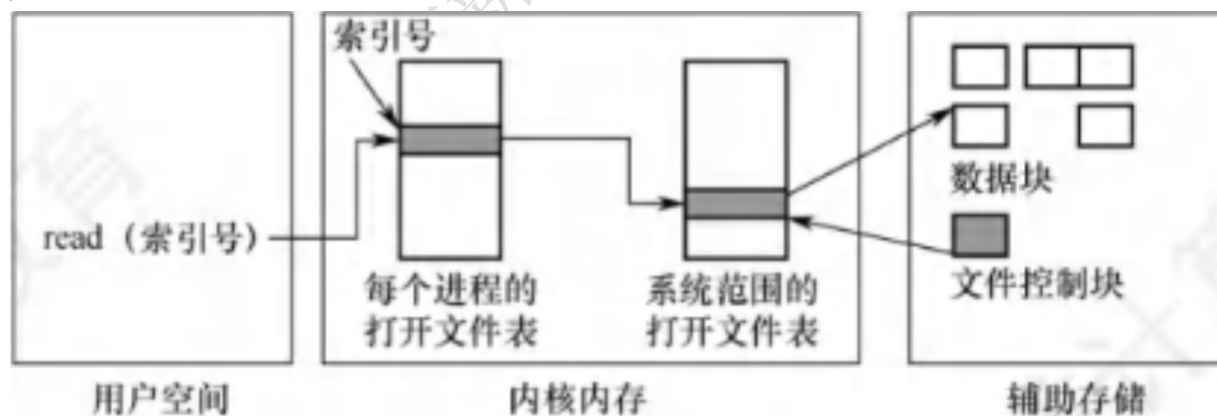
- 解释:

- 在多个进程可以同时打开文件的操作系统中，通常采用两级表：一个**系统表**和多个**进程表**；
- 整个**系统**的打开文件表包含与进程无关的信息，如**文件在** 磁盘上的位置、 访问日期 和 文件大小；
- 每个**进程**的打开文件表保存的是进程对文件的使用信息，如文件的当前 读写指针（文件描述符）、 文件访问权限；
- 进程打开了一个文件，系统表就包含该文件的条目，并包含指向系统表中适当条目的指针；
- 一旦有当另一个进程执行调用 `open` 时，只不过是**在其打开文件表中增加一个条目**，并指向系统表的相应条目；
- 通常，系统打开文件表为每个文件关一个打开计数器（`Open Count`），以记录多少进程打开了该文件；
- 当文件不再使用时，利用系统调用 `close` 关闭它，会删除单个进程的打开文件表中的相应条目，系统表中的相应打开计数器也会递减；
- 当打开计数器为 `0` 时，表示该文件不再被使用，并且可从系统表中删除相应的条目；

- 真题：2023年

- 若某文件 F 被进程 P 打开，则当进程 P 关闭文件 F 时，操作系统需要将文件的内存索引节点中的链接计数器减一，并释放 F 的索引节点所占的内存空间；
- 错误选项：
 - 释放文件 F 所占的外存空间：错误，这是删除文件所做的事情；
 - 将文件的外存索引节点中的链接计数器减一：错误，是对内存中的计数器进行减一；

- 图示:



文件保护

访问控制

概念：访问控制

- 概念：访问控制
 - 解决访问控制最常用的方法是根据用户身份进行控制，而实现基于身份访问的最为普通的方法，是**为每个文件和目录增加一个访问控制列表**（`Access-Control List, ACL`），以规定每个用户名及其所允许的访问类型；

- 这种方法的优点是可以使用复杂的访问方法，缺点是长度无法预计，并且可能导致复杂的空间管理，使用精简的访问列表可以解决这个问题；
- 在每个文件的 FCB (或索引结点)中增加一个访问控制列表 (Access-Control List, ACL)，该表中记录了各个用户可以对该文件执行哪些操作。
- 概念：精简的访问控制表
 - 引入：
 - 有的计算机可能会有很多用户，因此访问控制列表可能会很大，可以用精简的访问列表解决这个问题；
 - 分类：
 - 1)拥有者：创建文件的用户；
 - 2)组：一组需要共享文件且具有类似访问的用户；
 - 3)其他：系统内的所有其他用户；
 - 解释：
 - 每项占用一个二进制位，这样只需用 3×4 位的矩阵即可描述这三类用户的访问权限；
 - 创建文件时，系统将文件拥有者的名字、所属组名记录在该文件的 FCB 中。用户访问该文件时，若用户是文件主，按照文件主所拥有的权限访问文件。若用户和文件主在同一个用户组，则按照同组权限访问，否则只能按其他用户权限访问；

- 图示：

访问类型 用户类型	读 取	写 入	删 除	执 行
拥有者	1	1	1	1
组	1	0	0	0
其他	0	0	0	0

- 举例：

- 问题：

【2017 统考真题】某文件系统中，针对每个文件，用户类别分为 4 类：安全管理员、文件主、文件主的伙伴、其他用户；访问权限分为 5 种：完全控制、执行、修改、读取、写入。若文件控制块中用二进制位串表示文件权限，为表示不同类别用户对一个文件的访问权限，则描述文件权限的位数至少应为（ ）。

- A. 5 B. 9 C. 12 D. 20

- 解析：

- 因为一共有四类用户，分别对应五类权限，因此需要有 20 位来存储访问控制信息；
- 注意：文件控制信息不是存储在某个用户中的，而是存在文件中的，因此是以二维矩阵的形式、存储各个类别的访问权限和类别的对应信息；

访问权限 用户类别	完全控制	执行	修改	读取	写入
安全管理员	1	1	1	1	0
文件主	1	1	1	1	1
文件主的伙伴	0	1	0	1	0
其他用户	0	0	0	1	0

- 概念：访问类型

- 分类：

- 读

- 从文件中读数据

- 写

- 向文件中写数据

- 执行

- 将文件装入内存并执行

- 添加

- 将新信息添加到文件结尾部分

- 删除

- 删除文件，释放空间

- 列表清单

- 列出文件名和文件属性

- 特点：

- 实现灵活，可以实现复杂的文件保护功能

- 图示：

用户	读	写	执行	添加	删除	列表清单
father	1	1	1	1	1	1
mother	1	0	1	0	0	1
son	0	0	0	0	0	0

	完全控制	执行	修改	读取	写入
系统管理员	1	1	1	1	1
文件主	0	1	1	1	1
文件主的伙伴	0	1	0	1	0
其他用户	0	0	0	0	0

口令保护

概念： 口令保护

- 概念
 - 为文件设置一个口令，用户请求访问该文件必须提供口令
 - 口令一般存放在文件对应的 FCB 或索引结点中；
 - 用户访问文件前需要先输入口令，操作系统会将用户提供的口令与 FCB 中存储的口令进行对比，如果正确，则允许该用户访问文件。
- 特点
 - 优点：
 - 保存口令的空间开销不多；
 - 验证口令的时间开销也很小；
 - 缺点：
 - 正确的口令存在在系统内部，不够安全；

加密保护

概念：加密保护

- 定义
 - 使用某个密码对文件进行加密，在访问文件时需要提供正确的密码才能对文件进行正确的解密；
 - 如果使用错误的解压密码时，解密出来的文件是错误的；
- 特点
 - 优点：
 - 保密性强，不需要在系统中存储密码
 - 缺点：
 - 编码/译码，或者说加密/解码需要花费一定时间；
- 图示：

Eg: 一个最简单的加密算法——异或加密
假设用于加密/解密的“密码”为“01001”

文件的原始数据:	0	0	1	0	1	0	1	1	0	0	0	1	1	1	0	1	0	0	0	1	...
加密结果:	0	1	1	0	0	0	0	1	0	1	0	0	1	1	1	1	1	0	0	0	...
解密密码:																0	1	0	0	1	
解密结果:	0	0	1	0	1	0	1	1	0	0	0	1	1	1	0	1	0	0	0	1	...

文件的逻辑结构

无结构文件

概念：无结构文件

- 概念：
 - 文件内部的数据就是一系列二进制流或字符流组成。
 - 又称为流式文件；

- 例如：
 - Windows 操作系统中的 .txt 文件；

有结构文件

概念：有结构文件

- 概念：
 - 由一组相似的文件记录组成，又称记录式文件。
 - 每条记录又若干个数据项组成。
- 分类：
 - 定长记录
 - 可变长记录
- 逻辑结构
 - 顺序文件
 - 索引文件
 - 索引顺序文件

顺序文件

概念：顺序文件

- 定义
 - 文件中的记录一个接一个地顺序排列（逻辑上），记录可以是定长的或可变长的；
- 存储形式
 - 各个记录在物理上可以顺序存储或链式存储；
- 结构
 - 串结构
 - 记录之间的顺序与关键字无关；
 - 通常按照记录存入的时间决定记录的顺序；
 - 顺序结构
 - 记录之间的顺序**按关键字顺序排列**；

概念：顺序文件存储

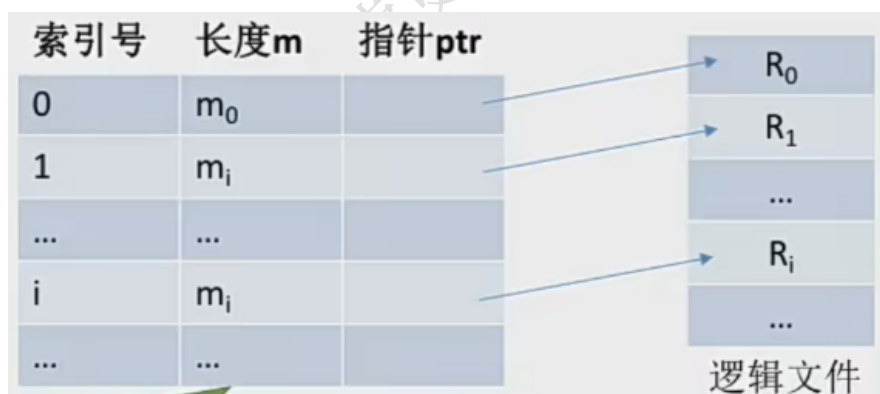
- 链式存储
 - 无论是定长/可变长记录，都**无法实现随机存取**，每次只能从第一个记录开始依次往后查找；
- 顺序存储
 - 可变长记录
 - **无法实现随机存取**。每次只能从第一个记录开始依次往后查找；
 - 定长记录
 - **可实现随机存取**。记录长度为 L ，则第 i 个记录存放的相对位置是 $i*L$
 - 若采用串结构，无法快速找到某关键字对对应的记录
 - 若采用顺序结构，可以快速找到某关键字对应的记录；

- 缺点：
 - 增加、删除一个记录困难（除了串结构）
- 结论：
 - 定长记录的顺序文件，若物理上采用顺序存储，则可实现随机存取；
 - 若能再保证**记录的顺序结构**，则可实现快速检索（即根据关键字快速找到对应记录 QS）；
- 注意：
 - 一般来说，考试题目中所说的“顺序文件”指的是物理上顺序存储的顺序文件；

索引文件

概念：索引文件

- 概念：
 - 每个文件建立一个索引表，建立一张索引表以加快文件检索速度；
 - 每条记录对应一个索引项；
 - 索引表中的内容**连续排列存放**，但索引表记录的文件可以**离散的存放**；
 - 每当要增加/删除一个记录时，需要对索引表进行修改；
- 场合：
 - 由于索引文件有很快的检索速度，因此主要用于**对信息处理的及时性要求比较高的场合**；
- 查找：
 - **索引表本身是定长记录的顺序文件**，因此可以**快速找到**第 i 个记录对应的索引项；
 - 可将关键字作为索引号内容，若按关键字顺序排列，则还可以支持按照关键字**折半查找**；
- 缺点：
 - 每个记录对应一个索引表项，因此索引表可能会很大；
- 图示：



索引顺序文件

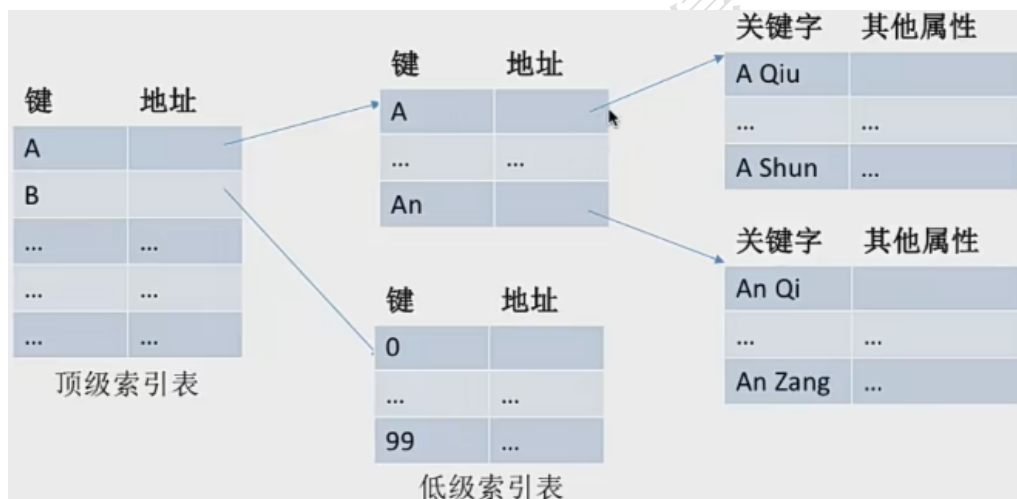
概念：索引顺序文件

- 概念：
 - 为文件建立一张索引表，但其不是每个记录对应一个索引表项，而是一组记录对应一个索引表项；
 - 索引顺序文件的索引项也不需要按关键字顺序排列，这样可以极大地方便新表项的插入；
 - 分组的**组间关键词需要有序**，但是分组内的关键词无需有序；
- 举例：

- 问题：
 - 若一个顺序文件有 10000 个记录，则根据关键字检索文件，只能从头开始顺序查找（这里指的并不是定长记录、顺序结构的顺序文件），平均须查找 5000 个记录；
- 分析：
 - 若采用索引顺序文件结构，可把 10000 个记录分为 100 组，每组 100 个记录；
 - 则需要先顺序查找索引表找到分组（共 100 个分组，因此索引表长度为 100，平均需要查 50 次），找到分组后，再在分组中顺序查找记录（每个分组 100 个记录，因此平均需要查 50 次）；
 - 采用索引顺序文件结构后，平均查找次数减少为 $50+50=100$ 次；

概念：多级索引顺序文件

- 概念：
 - 为了进一步提高检索效率，可以为顺序文件建立多级索引表
- 举例：
 - 对于一个含 10^6 个记录的文件，可先为该文件建立一张低级索引表，每 100 个记录为一组，故低级索引表中共有 10000 个表项（即 10000 个定长记录），再把这 10000 个定长记录分组，每组 100 个，为其建立顶级索引表，故顶级索引表中共有 100 个表项
- 图示：



文件的物理结构

物理结构

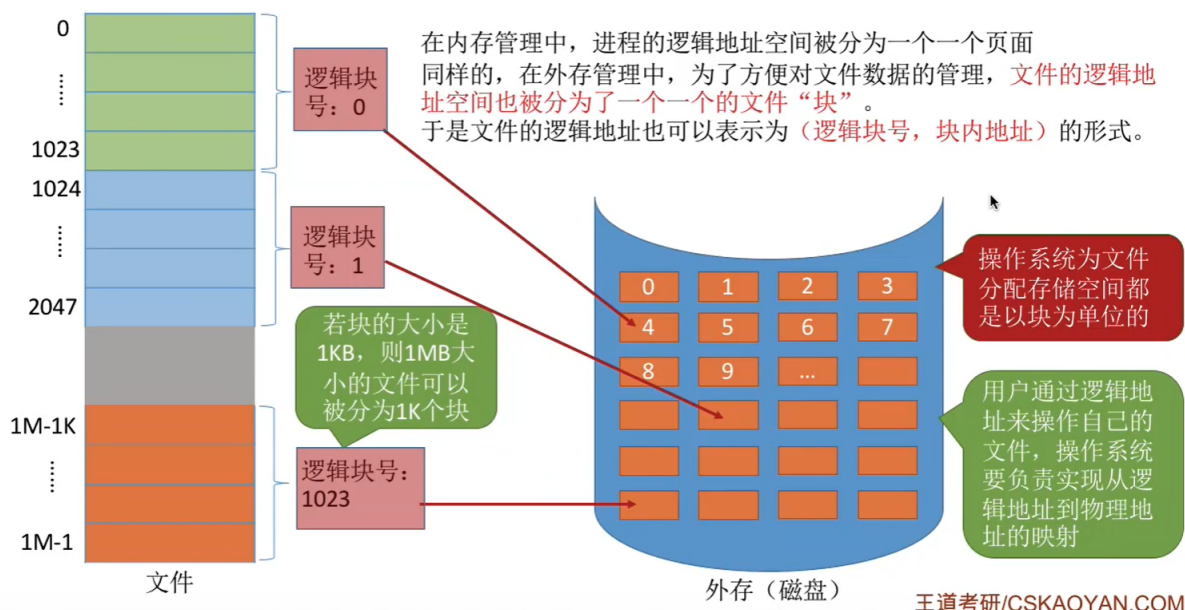
概念：文件存储在外存

- 概念：
 - 外存会分为一个个“块/磁盘块/物理块”；
 - 很多操作系统中，磁盘块的大小与内存块、页面的大小相同；
 - 每个磁盘块的大小是相等的，每块一般包含 2 的整数幂个地址；
 - 操作系统同样需要将逻辑地址转换为外存的物理地址（物理块号，块内地址）的形式，块内地址的位数取决于磁盘块的大小；
- 文件块：
 - 文件的逻辑地址空间被分为了一个一个的文件“块”；

- 文件的逻辑地址也可以表示为 (逻辑块号, 块内地址) 的形式;

- **文件系统分配空间也是以块为单位进行的;**

- 图示:



连续分配

概念: 连续分配

- 定义

- 连续分配方式要求每个文件在磁盘上占有一组连续的块;

- 地址:

- 用户给出要访问的逻辑块号, 操作系统找到该文件对应的目录项 (FCB), 计算:

- $$\text{物理块号} = \text{起始块号} + \text{逻辑块号}$$

- 当然, 还需要检查用户提供的逻辑块号是否合法 (逻辑块号 $\geq$ 长度: 不合法)

- 特点

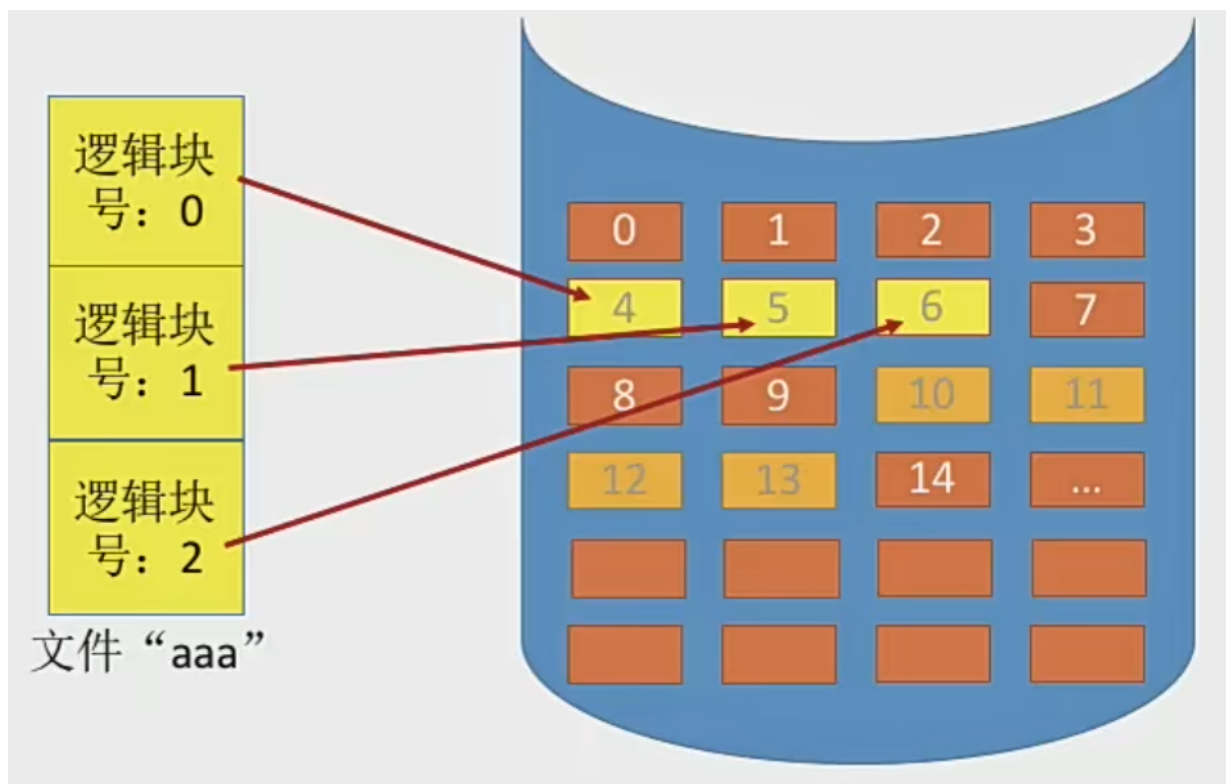
- 优点

- 1. 可以直接算出逻辑块号对应的物理块号;
- 2. 连续分配支持**顺序访问和直接访问 (随机访问)**;
- 3. 连续分配的文件在**顺序读/写**时速度最快;

- 缺点

- 1. 物理上采用连续分配的文件**不方便拓展**;
 - 因为要连续的空间, 因此经常会发生数据整体迁移, 不方便拓展;
- 2. 存储空间利用率低, 会产生难以利用的磁盘碎片
 - 可以用**紧凑**技术来处理碎片, 但是需要耗费很大的时间代价;

- 图示:



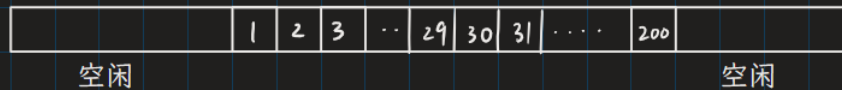
分析：连续分配结构下的文件插入

• 问题：

- 文件 F 由 200 条记录组成，记录从 1 开始编号。用户打开文件后，欲将内存中的一条记录插入文件 F，作为其第 30 条记录；
- 请回答下列问题，并说明理由；
- 1. 若文件系统采用连续分配方式，每个磁盘块存放一条记录，文件 F 存储区域前后均有足够的空闲磁盘空间，则完成上述插入操作最少需要访问多少次磁盘块？
- 2. F 的文件控制块内容会发生哪些改变？

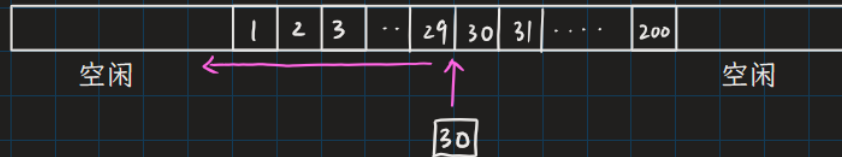
• 解析：

注意当前是以及创建并分配完成的文件F，此文件F的200条记录以及分配到了内存当中，如下：

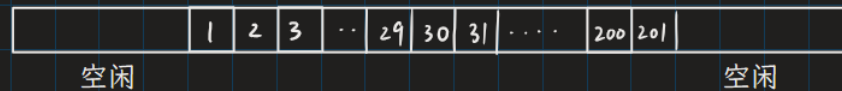


若要往其中插入记录作为第30号记录，则此时可以选择将该文件的前29个文件将前移动移动1块，并且文件块的移动需要两步：

- 1. 将目标块读入内存；2. 将原位置释放；3. 将目标块写入新磁盘位置



由于每移动一块磁盘块需要进行一次读入、一次写回，因此29个磁盘块共需要 $29 \times 2 = 58$ 此磁盘访问；并且还需要写入第30号磁盘块，因此功需要 $58 + 1 = 59$ 次磁盘访问



链接分配

概念：链接分配

- 定义：
 - 采用离散分配的方式，可以为文件分配离散的磁盘块。分为隐式链接和显示链接两种。
- 注意：
 - 考试题目中遇到未指明隐式/显式的“链接分配”，**默认指的是隐式链接**的链接分配；
- 隐式链接：
 - 实现
 - 除文件的最后一个盘块之外，每个盘块中都存有**指向下一个盘块的指针**；
 - 文件目录包括文件**第一块**的指针和**最后一块**的指针；
 - 特点
 - 优点：
 - 方便文件拓展；
 - 不会有碎片问题；
 - 外存利用率高；
 - 缺点：
 - **只支持顺序访问**，不支持随机访问，查找效率低；
 - 指向下一个盘块的指针也需要耗费少量的存储空间；

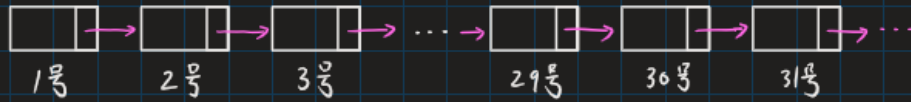
分析：链接分配插入过程分析

- 问题：
 - 文件 F 由 200 条记录组成，记录从 1 开始编号。用户打开文件后，欲将内存中的一条记录插入文件 F，作为其第 30 条记录；

- 请回答下列问题，并说明理由；
- 若文件系统采用链接分配方式，每个磁盘块存放一条记录和一个链接指针，则完成上述插入操作需要访问多少次磁盘块？若每个存储块大小为 1KB，其中 4B 存放链接指针，则该文件系统支持的文件最大长度是多少？

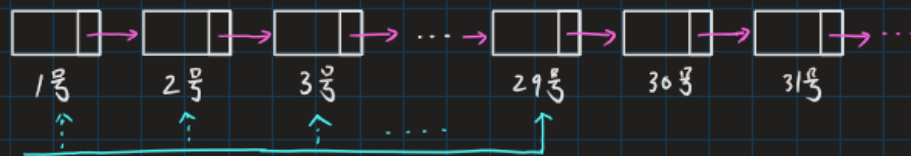
• 解析：

在插入文件之前，由于采取链接分配，文件的结构如下：



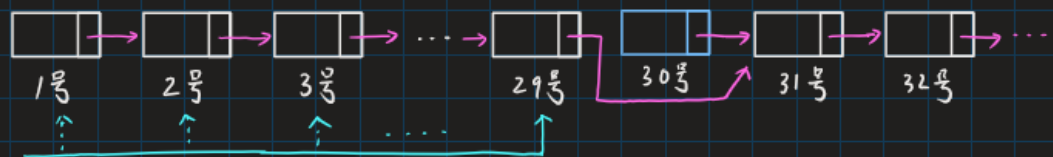
插入文件的基本步骤如下：

- 1. 访问链表，找到目标块的位置，返回目标块的前一块的地址：

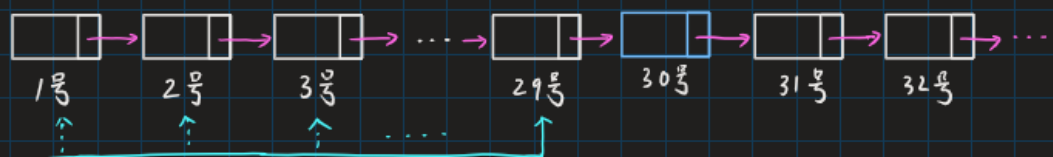


连续访问了29个磁盘块，找到第30块的前一块，即第29块，此过程共访问了29个磁盘块；

- 2. 根据第29块中链接地址部分数据，返回给CPU，CPU知道了第30块的地址后，将目标块的数据存入第30块，此过程涉及一次磁盘块访问；



- 3. 访问第29条记录的链接地址部分，将其指向第30号记录，此过程涉及一个磁盘块访问；



4B 共 32 位，可以寻址 2^{32} 块存储块，每块的大小为1KB，即 1024B，其中下块地址部分占 4B，数据部分占1020B，因此该系统的文件最大长度是 $4G * 1020B = 4080GB$

★ 注意：文件大小不包括链接指针的大小，只包括数据的部分大小

只含有数据部分。

FAT 分配

概念：FAT 分配

- 实现

- 真题：2009年
- 把用于链接文件各物理块的指针显式地存放在一张表中，即文件分配表（FAT，File Allocation Table）；
- 一个磁盘仅设置一张 FAT，开机时，将 FAT 读入内存，并常驻内存；
- FAT 的各个表项在物理上连续存储，且每一个表项长度相同，因此物理块号字段可以是隐含的；
- 特点
 - 优点
 - 很方便文件拓展，不会有碎片问题，外存利用率高，并且支持随机访问；
 - 逻辑块号转换成物理块号的过程不需要读磁盘操作；
 - 相比于隐式链接来说，地址转换时不需要访问磁盘（只需要读内存中的 FAT 表），因此文件的访问效率更高；
 - 缺点
 - 文件分配表的需要占用一定的存储空间；
- 总结：
 - 采用链式分配（显式链接）方式的文件，支持顺序访问，也支持随机访问（想访问 i 号逻辑块时，并不需要依次访问之前的 0~i-1 号逻辑块）；
 - 由于块号转换的过程不需要访问磁盘，因此相比于隐式链接来说，访问速度快很多；
 - 而且也不会产生外部碎片；
- 图示：



索引分配

概念：索引分配

- 概念：
 - 允许文件离散地分配在各个磁盘块中，系统会为每个文件建立一张索引表，索引表中记录了文件的各个逻辑块对应的物理块；

- 索引表存放的磁盘块称为**索引块**。文件数据存放的磁盘称为**数据块**；
- 目录中需要记录文件的索引块是几号磁盘块；
- 注意：
 - 在显式链接的链式分配方式中，文件分配表 **FAT** 是一个磁盘对应一张；
 - 而在索引分配方式中，索引表是一个文件对应一张；
 - 逻辑块号可以是隐含的；
- 实现：
 - 从**目录项**中可知索引表存放位置；
 - 将索引表从**外存**读入**内存**；
 - 查找**索引表，得到 **i** 号**逻辑块**在外存中的存放位置；
- 特点：
 - 索引分配方式可以支持随机访问；
 - 索引分配中文件拓展也很容易实现 (只需要给文件分配一个空闲块，并增加一个索引表项即可)；
- 大文件处理：
 - 链接方案；
 - 多层索引；
 - 混合索引；
- 图示：

文件名	...	索引块
aaa	...	7
bbb	...	23

逻辑块号	物理块号
0	2
1	5
2	13
3	9

概念：索引分配方案 - 链接方案

- 实现
 - 如果索引表太大，一个索引块装不下，那么可以**将多个索引块链接起来存放**；
 - 比如当前索引块最多可以有 **256** 个索引项，此时若文件的大小超过 256 个索引块，再分配一块索引块，用指针链接在一起；
 - 想要找到 **i** 号索引块，必须先依次读入 **0~i-1** 号索引块；

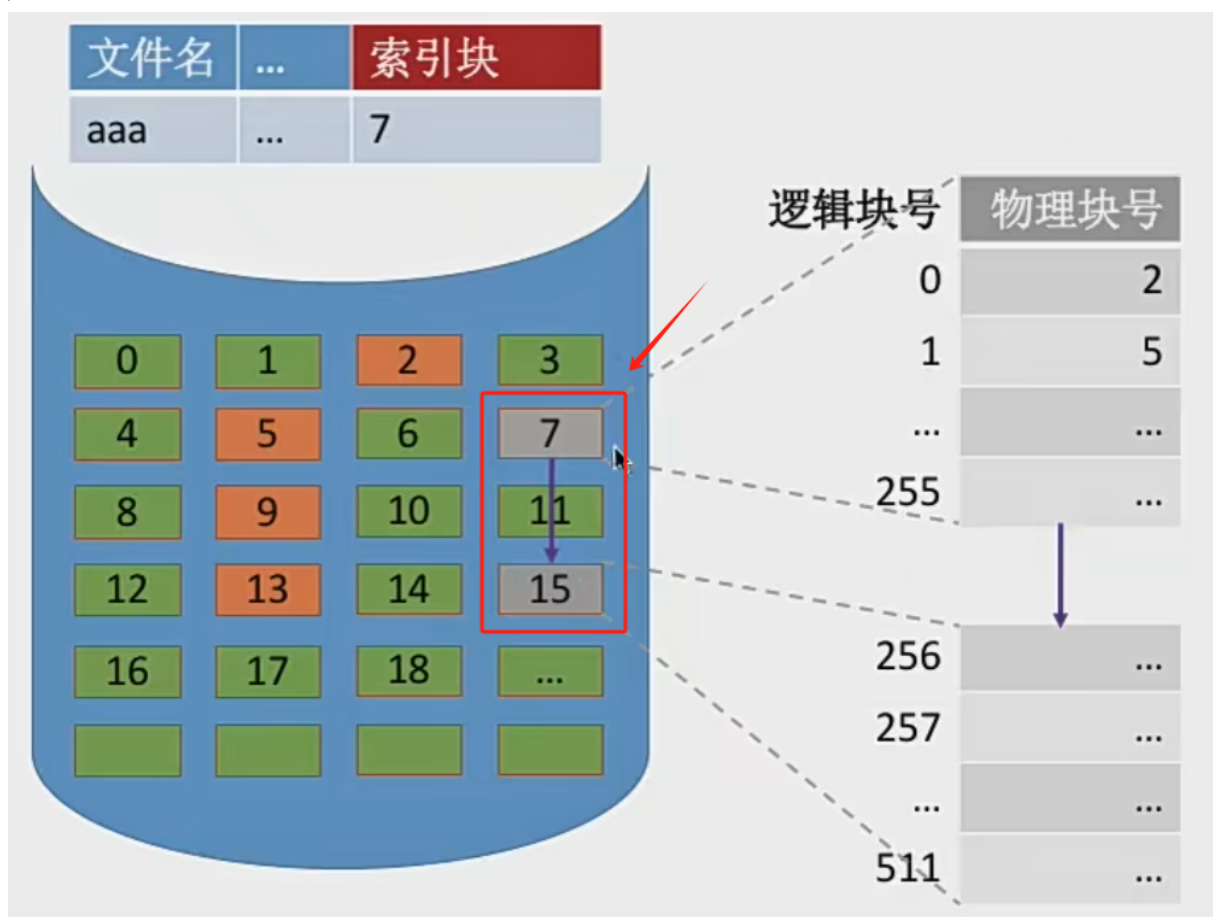
- 比如若想要访问文件的最后一个索引块，就必须找到最后一个索引块 (第 256 个索引块)，而各个索引块之间是用指针链接起来的，因此必须先顺序地读入前 255 个索引块

- 特点

- 缺点

- 若文件很大，索引表很长，就需要将很多索引表链接起来。
 - 想要找到 i 号索引块，必须先依次读入 $0 \sim i-1$ 号索引块，就导致磁盘 I/O 次数过多，查找效率低下。

- 图示：



概念：索引分配方案 - 多层索引

- 实现

- 建立多层索引 (原理类似多级页表)，使第一层索引块指向第二层的索引块；
 - 还可根据文件大小的要求再建立第三层，第四层索引块；
 - 采用 K 层索引结构，且顶级索引表未调入内存，则访问一个数据块需要 $K+1$ 次读磁盘操作；

- 特点

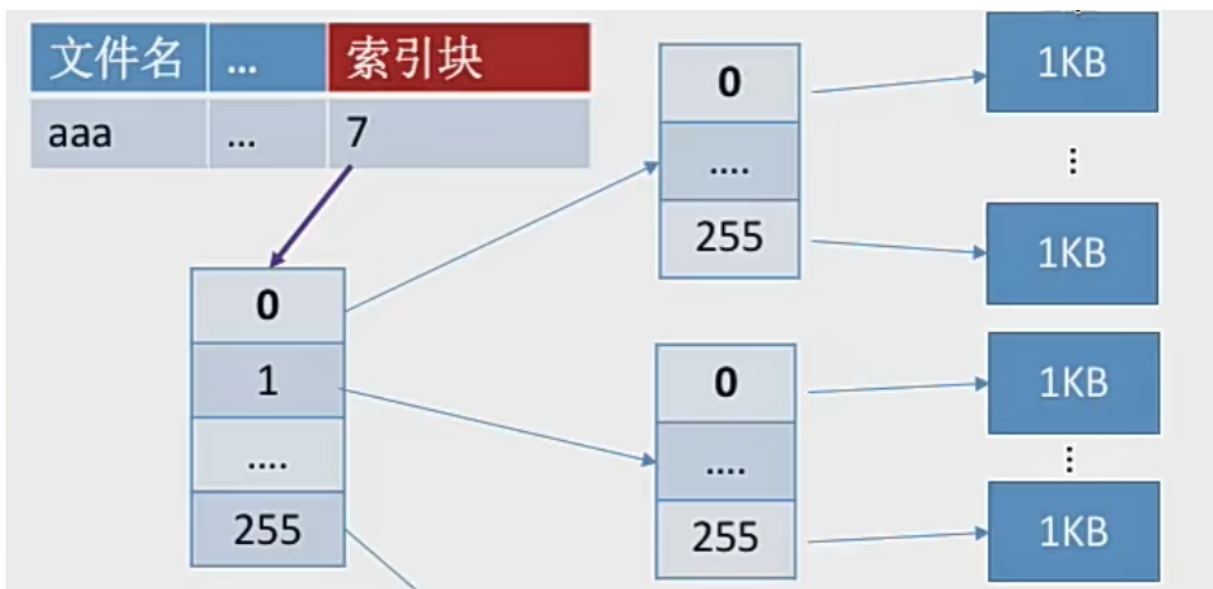
- 缺点：

- 即使是小文件，访问一个数据块依然需要 $K+1$ 次读磁盘。

- 注意：

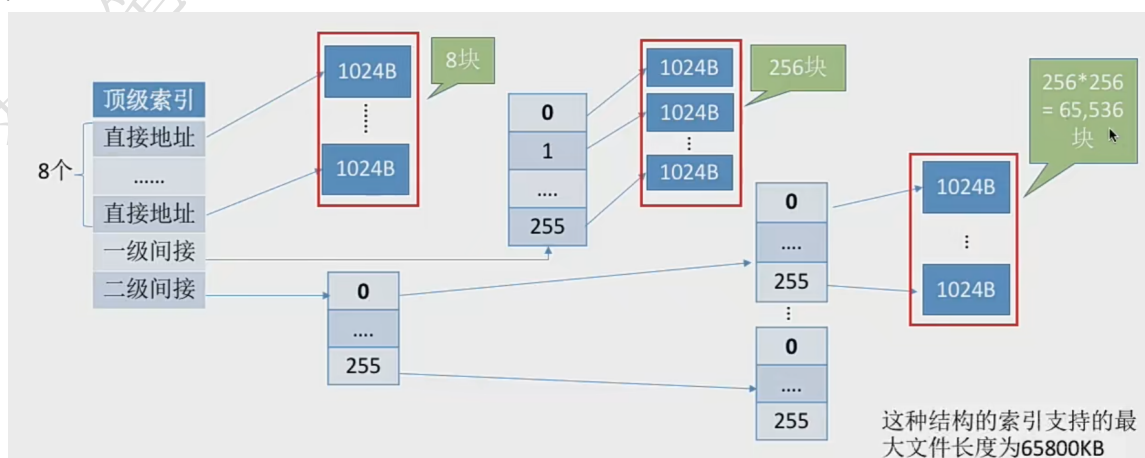
- 若采用多层索引，则各层索引表大小不能超过一个磁盘块；

- 图示：



概念：索引分配方案 - 混合索引

- 实现
 - 多种索引分配方式的结合；
 - 例如：一个文件的顶级索引表中，既包含直接地址索引（直接指向数据块），又包含一级间接索引（指向单层索引表），还包含两级间接索引（指向两层索引表）；
- 特点
 - 优点：对于小文件来说，访问一个数据块所需的读磁盘次数更少；
- 分析：混合索引下的磁盘访问次数
 - 描述：
 - 当前顶级索引表还没读入内存，且索引结构如图所示；
 - 分析：
 - 第 0~7 号逻辑块：两次读磁盘访问
 - 第 8~263 号逻辑块：三次读磁盘访问
 - 第 264~65799 号逻辑块：四次读磁盘
 - 图示：



总结

		How?	目录项内容	优点	缺点
顺序分配		为文件分配的必须是连续的磁盘块	起始块号、文件长度	顺序存取速度快，支持随机访问	会产生碎片，不利于文件拓展
链接分配	隐式链接	除文件的最后一个盘块之外，每个盘块中都存有指向下一个盘块的指针	起始块号、结束块号	可解决碎片问题，外存利用率高，文件拓展实现方便	只能顺序访问，不能随机访问。
	显式链接	建立一张文件分配表(FAT)，显式记录盘块的先后关系（开机后FAT常驻内存）	起始块号	除了拥有隐式链接的优点之外，还可通过查询内存中的FAT实现随机访问	FAT需要占用一定的存储空间
索引分配		为文件数据块建立索引表。若文件太大，可采用链接方案、多层索引、混合索引	链接方案记录的是第一个索引块的块号，多层/混合索引记录的是顶级索引块的块号	支持随机访问，易于实现文件的拓展	索引表需占用一定的存储空间。访问数据块前需要先读入索引块。若采用链接方案，查找索引块时可能需要很多次读磁盘操作。

获取笔记持续更新可加微信: WandD2277