操作系统作业六

PB18151866 龚小航

1. Consider a RAID organization comprising five disks in total, how many blocks are accessed in order to perform the following operations for RAID-5 and RAID-6?
   a. An update of one block of data
   b. An update of seven continuous blocks of data. Assume that the seven contiguous blocks begin at a boundary of a stripe.

解： RAID-5 将数据以块为单位分布到各个硬盘上。RAID-5 不对数据进行备份，而把数据和与其相对应的奇偶校验信息存储到组成 RAID-5 的各个磁盘上，并且奇偶校验信息和相对应的数据分别存储于不同的磁盘上。当 RAID-5 的一个磁盘数据损坏后，利用剩下的数据和相应的奇偶校验信息恢复被损坏的数据。

RAID-6 是在 RAID-5 基础上把校验信息由一位增加到两位的 RAID 级别。

RAID-6 和 RAID-5 一样对逻辑盘进行条带化然后存储数据和校验位，只是对每一位数据又增加了一位校验位。这样在使用 RAID-6 时会有两块硬盘用来存储校验位，增强了容错功能，同时必然会减少硬盘的实际使用容量。以前的 RAID 级别一般只允许一块硬盘坏掉，而 RAID-6 可以允许坏掉两块硬盘，因此，RAID-6 要求至少 4 块硬盘。

本题的 RAID 由 5 块硬盘构成：

- **当需要更新一块数据时：**

  RAID-5 先读取校验位所在块，再读取需要修改的数据所在块，然后计算出校验位再更新校验位，最后更新要写入的数据块，一共需要进行 4 次访问。

  对于 RAID-6，先读取两个校验位所在的块和需要修改的数据所在块，再计算出新的两个校验位，再更新写入两个新的校验位所在块，最后更新要写入的数据块，此时一共进行了 6 次访问。

- **当需要连续更新七块数据时：**

  对于 RAID-5，由于一共有 5 块磁盘并且有其中一块磁盘的信息是校验位，因此更新这连续七块的数据信息时一共会涉及到两个校验位的更改，这样一共需要读取 7+2=9 块的磁盘信息，并且在计算完校验位后再写入这 9 块磁盘的更新后的信息，这样一共进行了 18 次访问。

  对于 RAID-6，5 块磁盘中，2 块磁盘用于存储校验位，因此更新连续七块的数据信息会涉及到 6 位校验位的更新，这样一共需要读取 7+6=11 块的磁盘信息，并且在计算完校验位后还需再写入这 11 块磁盘的更新后的信息。共进行了 22 次访问。

2. Explain what open-file table is and why we need it.

解：打开文件表是操作系统用于维护所有打开文件的信息所用的表。大多数文件操作涉及搜索目录以得到命名文件的相关条目。为了避免这种不断的搜索，具有打开文件表的系统在首次使用文件之前进行系统调用 open()；当请求文件操作时，操作系统可以通过这张表的索引指定文件而不需要搜索。当文件最近不再使用时，进程关闭它，操作系统从打开文件表中删除它的条目。

3. Explain the concept of file and directory, and what does “755” mean for file permission?t

解： 解释定义：

- **文件：**文件是操作系统提供的信息存储的统一逻辑视图，是对存储设备的物理属性加以抽象得到的逻辑存储单位。文件是记录在外存上相关信息的命名组合，从用户角度看即是逻辑外存的最小分配单元。
- **目录：**目录是一种符号表，也是一种文件，可以将文件名转换成目录条目。目录一般包含了文件的基本信息，通过文件名称列出它所包含的文件。

若某个文件的权限属性为 755，下面解释它的含义：

权限属性为三位，由左至右，每一位分别表示所有者权限，同组用户权限，组外权限。

而每一位都是一个十进制数，不同的数表示不一样的权限：

将它们写成二进制形式：

$$7D = 111B;\quad 5D = 101B$$

二进制的从左至右三位分别表示 读权限，写权限，执行权限

因此，755 权限的含义是：

	读	写	执行
7	1	1	1
所有者	√	√	√
5	1	0	1
同组用户	√	×	√
5	1	0	1
组外用户	√	×	√

4. Explain the problems of using continuous allocation for file system layout and how to solve them.

解： 连续分配的一个难题是**为新文件找到空间**。用于管理空闲空间的系统决定了这个任务如何完成。找到新文件的算法都存在外部碎片的问题。为了解决这个问题，可以将整个文件系统复制到另一个磁盘，这样原来的磁盘完全变成空的，从而创建了一个大的连续空闲空间，然后再对这个大的连续空间采用连续分配方法将这些文件复制回来。但是这种合并的代价是时间，而且大硬盘的代价可能特别高。

连续分配的另一个难题是**确定一个文件需要多少空间**。一般来说，输出文件的大小可能难以估计。如果为文件分配的空间太小，则可能导致文件无法扩展，分配空间太大又会造成浪费。如果分配空间小了，可能会终止用户程序并给出适当的错误信息，这样用户必须分配更多的空间并再次运行该程序，或是必须找一个更大的空间，将文件内容复制到新空间，并释放以前的空间。为最小化这些缺点，有些操作系统采用连续分配的修正方案：起初分配一块空间。当这个数量不够时，会以指针链接添加另一块连续空间，称为扩展。但是块的大小不同，还是会产生内部或外部碎片的问题。

5. What are the advantages of the variation of linked allocation that uses a FAT to chain together the blocks of a file? What is the major problem of FAT?

解：FAT 的使用的优点是改善了链接分配随机访问的时间。因为通过读入 FAT 信息，磁头能找到任何块的位置。只要知道了文件首块的块号，就可以通过 FAT 一直找到文件的所有块号。为文件分配一个新块只要简单找到第一个值为 0 的 FAT 条目，用新块的地址替换前面文件结束值，用文件结束值代替 0。

FAT 也有其缺点：FAT 需要被缓存，否则可能导致大量的磁头寻道时间，磁头必须移到卷的开头读入 FAT，并找到所需块的位置，再移到块本身的位置，最坏情况下每块都要移动两次。此外 FAT 表的指针可靠性不足（链接分配的通病），如果指针丢失或出错，可能会导致搜索到另一个其他的文件上。

6. Consider a file system similar to the one used by UNIX with indexed allocation, and assume that every file uses only one block. How many disk I/O operations might be required to read the contents of a small local file at /a/b/c in the following two cases? Should provide the detailed workflow.
   a. Assume that none of the disk blocks and inodes is currently being cached.
   b. Assume that none of the disk blocks is currently being cached but all inodes are in memory.

解：两种情况分别说明访问顺序即可：

a. 当前没有缓存任何磁盘块和索引节点时：

访问根目录 → 访问目录a → 访问目录b → 访问目录c → 访问索引块 → 访问数据

可见需要六次访问磁盘。

b. 当前没有一个磁盘块被缓存，但是所有 inode 都在内存中时：

访问根目录 → 访问目录a → 访问目录b → 访问目录c → 通过索引块直接访问数据块

可见需要五次访问磁盘。

7. Consider a file system that uses inodes to represent files. Disk blocks are 8-KB in size and a pointer to a disk block requires 4 bytes. This file system has 12 direct disk blocks, plus single, double, and triple indirect disk blocks. What is the maximum size of a file that can be stored in this file system?

解：指针占用 4 字节，即4B，而一个块的大小为$8\text{KB} = 2^{13}\text{B}$

在多级索引中，需要$4\text{B} = 2^{2}\text{B}$空间来存放指针，因此块的有效大小为$2^{11}\text{B}$

因此文件系统最大可存储容量：

$$S_{max} = 12 \times 2^{13}\text{B} + (2^{11} \times 2^{13}\text{B}) + (2^{11} \times 2^{11} \times 2^{13}\text{B}) + (2^{11} \times 2^{11} \times 2^{11} \times 2^{13}\text{B}) = 64\text{TB}$$

即最大可存储文件约为64TB