大容量存储器和文件系统

江学强，PB16120100

1. 大容量存储器

我们知道现代存储器具有明显的层次结构，层次结构主要可以分为 CPU 缓存、主存以及由大容量存储器组成的次级、三级存储器（也可称为外存）。磁盘是组成现代计算机大容量存储器最常见的物理设备。磁盘片两面涂有磁质材料，通过在磁质材料上进行磁记录可以保存和修改信息。磁盘的物理结构不多赘述，这里只叙述与后面磁盘调度有关的结构。读写头可以在磁盘面上移动用于读写信息，磁盘片的表面被划分为了一个个环形的磁道（track），磁道再进一步划分为扇区(sector)。磁盘上有磁盘控制器。主机控制器将读写命令传送给磁盘控制器，磁盘控制器操纵磁盘驱动器来执行命令。为了加快速率，磁盘控制器中通常置有内置缓存。

为了更有效地使用磁盘硬件，获得更快地访问速度和更宽地磁盘带宽，需要进行磁盘地调度。对磁盘的访问时间主要包括寻道时间和旋转延迟，寻道时间指磁臂将磁头移动到目标扇区的柱面所需的时间，旋转延迟时间指磁盘把目标扇区旋转到磁头下的时间。进程通过向操作系统发送系统调用来请求磁盘的 I/O 操作，这个系统调用包含一些有用的信息，如果所需的磁盘驱动器和控制器空闲，就会立刻处理请求，如果磁盘驱动器和控制器忙，就会把新的服务请求加入到等待队列中，因为等待队列可以有多个待处理的请求，所以就涉及对请求处理的调度，下面主要叙述几个典型的调度算法。

FCFS 调度：这是最简单的调度算法，先请求的先服务；

SSTF 调度（shortest-seek-time first）：最短寻道时间优先算法，意思是先处理靠近当前磁头位置的请求；

SCAN 调度：SCAN 算法是指磁臂从磁盘的一端向另一端移动，磁头移过每个柱面时处理该柱面上的服务请求，当到达另一端时，磁头改变方向，反向扫描。来回如此；

C-SCAN 调度：是 SCAN 调度的变种，C-SCAN 将磁头从一端移动到另一端时不反向移动，而是返回磁盘开始再次扫描；

LOOK 调度：SCAN 和 C-SCAN 调度都是扫描整个磁盘，但事实上磁头只移动到最远的请求为止而不是到磁盘的尽头，这样的调度算法相应的叫 LOOK 调度和 C-LOOK 调度。

操作系统还要负责管理磁盘其他方面的东西，下面叙述磁盘的初始化、磁盘引导。

新的磁盘可以看成是一个白板，只是含有磁性材料的盘子。在使用之前需要把磁盘划分为扇区，这个过程称为低级格式化。每个扇区通常由头、数据区域（通常 512B）和尾部组成。头部和尾部包含一些磁盘控制器所需要的信息，如扇区号码和 ECC。为了使用磁盘存储文件，操作系统还需要把磁盘分为一个或多个柱面组成的分区，在分区上挂载文件系统。

计算机启动时需要运行 bootstrap 程序，bootstrap 程序初始化系统，一般保存在 ROM 中，此外在磁盘上还有一个更完整的启动程序，拥有这个启动程序的磁盘分区称为启动盘或者系统盘。例如 windows 2000 的 MBR（主引导记录）,是磁盘上的第一个扇区

磁盘驱动器越来越小也越来越便宜，现在一台计算机系统上装有大量磁盘已经可行了，大量磁盘通过有效的组织可以有效改善读写速度和数据可靠性，这种组织技术称为 RAID 技术。下面叙述怎样通过 RAID 技术提高磁盘的读写性能和可靠性。两个主要思想是通过数据的冗余来提高可靠性以及通过数据读写的并行提高性能。

下面简单叙述一下 RAID 级别：

RAID0：按块分散的磁盘阵列，没有冗余，所以无法保证可靠性也并未提高读写性能；

RAID1：设置一份（不是一个）镜像磁盘，有冗余，保证了数据可靠性；

RAID2：称为内存方式的差错纠正码结构，实现了基于奇偶校验位的检错系统，每个字节的奇偶位记录字节中置为 1 的 bit 数是偶数还是奇数，如果字节的一个位损坏，字节的奇偶位将改变。差错纠正方案会储存两个或者多个额外的位，当单个位出错时可用来恢复数据。RAID2 相对于 RAID1 减少了冗余磁盘数，但可靠性不如 RAID1；

RAID3: 基于位交织奇偶结构，相对于 RAID2 做了改进，与内存系统不同，这里磁盘控制器能够检测到一个扇区是否正确读取，这样单个奇偶位就能用于差错检测和差错纠正，RAID3 和 RAID2 在可靠性上同样好，但是 RAID3 节省了磁盘，此外由于同一字节的不同位存储在不同的磁盘，因此可以并行读取，相对 RAID1 也提高了读写性能；

RAID4：采用块交织奇偶结构，采用块分散，另设一独立磁盘保存数据块的奇偶块；

RAID5：块交织分布奇偶结构，与 RAID4 不同，它是把数据和奇偶校验分布在所有 N+1 块磁盘中；

RAID6：RAID6 又称为 P+Q 冗余方案，与 RAID5 类似，但是保存额外冗余信息防止多个磁盘出错。并未采用奇偶校验，而是采用了差别纠正码如 Read-Solomon 码；

RAID0+1 和 RAID1+0:RAID0+1 和 RAID1+0 是 RAID0 和 RAID1 的组合，只不过 RAID0+1 是先分散再镜像，RAID1+0 是先镜像再分散；

RAID 的选择很多，在真正设计系统时要权衡性能、可靠性以及重建数据的代价等，此外需要注意 RAID 不能保证数据的可用性，例如执行文件的指针或者文件结构内的指针错误等,RAID 结构并不能解决这些错误带来的问题。

除了磁盘还要其他外存，但磁盘最为典型也最为重要，其他存储器不再赘述。

2. 文件系统接口

文件系统接口主要是指文件系统呈现给用户的部分内容，下面主要讨论文件和目录的一些概念以及文件系统的一些功能。

文件是一种抽象的逻辑概念，操作系统对存储设备的各种属性进行抽象，定义文件这一逻辑单元，再把文件映射到物理设备中。从用户角度看，文件是逻辑外存地最小分配单元，数据保存到外存都是通过把数据存储到文件来实现的。文件有很多属性，文件有名称以及文件系统内唯一的文件标识符（一般为数字），以及文件类型，文件的位置，文件大小，文件权限等。文件还有几个基本的操作，最基本的文件操作是创建文件、读写文件、在文件中进行文件指针的重定位以及删除文件和截短文件等。文件系统会维护一个

open-file-table 来记录打开了哪些文件。此外，每个打开的文件也需要包含一些信息，例如文件指针，文件打开计数器，文件磁盘位置以及访问权限。文件系统要识别和支持各种文件类型，接着才能对相应的文件进行相应的操作，现在最常见的标志文件类型的方式就是采用文件拓展名的方式，显然，不同类型的文件内部有不同的结构（但在物理层面上没有差别）。文件的内容在使用时必须要加载到内存中，文件可以有多种访问的方式，最常见的两种访问方式是顺序访问和直接访问。顺序访问是指文件信息按照顺序，一个记录一个记录地处理，直接访问是指允许按照任意顺序进行随机读和写。当然也有其他拓展的方式，如利用索引等。

文件系统除了文件这一概念外还有一个非常重要的概念那就是目录，目录事实上是一种特殊的文件。目录也有一些基本的操作，基本的目录操作有搜索文件、创建新文件并加到目录中、删除目录中的文件、遍历目录、重命名文件以及跟踪文件系统。目录可以有多种目录结构：单层文件目录结构，是所有文件都包含在同一目录中，这显然会有很多问题；树状目录结构是最常见的目录结构，此外还有无环图状的目录结构。

下面简单叙述一下文件系统的安装。文件系统在被进程使用之前必须要被安装，也成为挂载。挂载文件系统需要一个一个目录，通常是空目录，称为挂载点，挂载之后一般操作系统会验证文件设备的有效性。

当操作系统支持多用户时，就需要考虑文件共享、文件保护的问题了，因而多用户系统要比单文件系统维护更多的文件和目录属性。例如 UNIX 系统中文件的拥有者和拥有者的组内成员以及其他成员就对文件可能有不同的访问权限。

上面所述只是基本的文件系统的一些概述，现在还发展出了很多其他的文件系统；例如分布式文件系统和网络文件系统等等。

3. 文件系统的实现

在深入文件系统实现的细节之前首先得理解文件系统的分层设计，进程直接相关的是所谓逻辑文件系统，逻辑文件系统之下是文件组织系统或者虚拟文件系统，然后是基本文件系统最后才是 I/O 设备和磁盘等硬件设备。

I/O 控制是最底层的控制模块，由设备驱动程序和中断处理程序组成，实现内存和磁盘之间的信息连接；基本文件系统只需要向设备驱动程序发送命令就可以对磁盘上的物理块进行读写，每个物理块有数值地址来标识；文件组织模块连接管理文件的逻辑块和物理块；逻辑文件系统管理元数据，而不包括真正的文件数据内容，逻辑文件系统管理目录结构，并提供给文件组织模块必要额信息，文件控制块（FCB）包含文件的信息，如拥有着、权限、文件内容的位置。

实现文件系统要使用多个磁盘和内存结构，这些结构有一些通用规律。在磁盘上，文件系统可能包含以下信息：如何启动磁盘中所存储的操作系统（引导控制块）、总的存储的块数、空闲块的数目和位置、目录结构以及各个具体文件等。引导控制块包含分区的详细信息，如分区的块数、块的大小、空闲块的数量和指针、空闲 FCB 的数量和指针等等。引导控制块在有的文件系统中也称为超级快（superblock）或者主控文件表（Master File Table）；每个文件系统的目录结构用来组织文件，在超级块（以超级块为例）包含文件名和相关的索引结点（inode）号；每个文件的 FCB 包含包含该文件的一些详细信息，如文件权限、拥有者，在有的文件系统类似 FCB 的功能的叫做 inode（索引结点）系统范围维护一个整个系统的打开文件表，包含每个打开文件的 FCB 副本和其他信息，单个进程的打开文件表包含一个指向系统范围内已经打开文件表合适条目的指针和其他信息。

创建一个新文件，应用程序调用逻辑文件系统，逻辑系统需要分配一个新的 FCB，然后系统把相应的目录信息读入内存，用新的文件名更新该目录和 FCB，并把结果写回磁盘。

很多操作系统，例如 UNIX，把目录按照文件来处理，用一个标志位来表明是否为目录，但是也有操作系统为文件和目录提供了分开两套的系统调用。文件被创建之后，被打开之后就能用于 I/O 操作，系统调用 open（）会首先搜索系统范围内的打开文件表确定这个文件是不是被其他进程使用中，如果是，就在单个文件的打开文件表新创建一项，并指向系统范围的打开文件表，如果没有的话就新建一个这个文件的打开文件表，之后 open()会返回一个指向单个进程的打开文件表的指针，之后文件的操作都是基于这个指针进行的。文件名不必是打开文件表内容的一部分，而是常常用文件描述符来起到标识文件的作用，当一个进程关闭文件时就删除单个进程打开文件表的相应条目，系统范围内的打开文件表也会调整。

现代操作系统必须要支持多个文件系统，操作系统要是为每个文件系统都设计一套机制，这显然很麻烦，所以采用的是虚拟文件系统的方式。

虚拟文件系统通过定义一个清晰的 VFS 接口，把文件系统的通用操作和具体实现分开，多个虚拟文件系统的接口实现可以共存在同一台机器上。下面简单叙述不一下 Linux VFS 的一些内容，Linux VFS 定义了四种主要的对象，索引结点对象（inode）表示一个单独的文件、文件对象（file object）表示一个打开的文件，超级块对象（superblock）表示整个文件系统，目录条目对象（dentry object）表示一个单独的目录条目。VFS 对每种对象都定义了一组必须实现的操作，这些类型的每个对象都包含一个指向函数表的指针，而在特点文件类型的文件对象的实现必须实现对象定义的每个函数，VFS 通过调用对象函数表中的函数来实现文件操作。

目录的实现对文件系统的效率、性能和可靠性有很大影响。下面讨论两种最常见的目录的实现方法。一种是线性列表，顾名思义，要创建新文件必须在线性表中搜索有没有相同名字的文件，如果没有，则在表的末尾添加一个元素，删除文件则有想反的过程，显然这种方法效率很低，而且随着文件的变多，效率会越来越低；还有一种常见的方法就是利用哈希表这种数据结构，采用这种方法，哈希表根据文件名得到一个值，并返回一个指向线性列表中元素的指针，因此能够大大地减少搜索时间，但是利用哈希表的困难就是要解决冲突的问题。

为了在利用文件系统访问磁盘时更加有效率，必须要采用有效的磁盘空间分配的方法，常见的分配方法有连续分配、利用链接分配以及索引。

连续分配的方式是指每个文件在磁盘上占有一块连续的块，文件的连续分配可以用文件占据的第一块的磁盘地址和连续块的数量来定义，对连续分配文件的访问很容易，根据初始地址和数量连续访问即可。连续分配一个很大的问题就是这种方法可能导致大量的外部碎片，因为随着文件的分配和删除，磁盘空间被分割成一个个小片，这就很可能导致创建大的文件时无法利用这些碎片，造成了磁盘空间的浪费；还有一个主要的问题是这种分配方式导致文件很难拓展。

链接分配是指每个数据块上维护一个指针，指向下一个数据块，这就意味着同一个文件得数据块可以分布在磁盘得任何地方，目录中包含第一个数据块的指针和最后一个数据块的指针，这种方式解决了连续分配的所有问题，但是这种方式导致访存会导致不停地寻道，导致访存地效率比较低，此外指针地维护也需要耗费磁盘空间。FAT 是链接分配方式地变种，FAT 是文件分配表地简称，每个分区地开始存储 FAT，目录条目含有文件块地首地址，根据块号码，FAT 条目中包含文件下一块的块号码，FAT 中标志未用的数据块用 0 来标记。为文件分配新的块只要找到 FAT 中标记为 0 的条目即可。

索引分配是指把所有的指针放在一起，形成索引快，索引块的第 i 个条目指向文件的第 i 个数据块，有时候文件过大，一个索引快可能不够用，这就需要额外的机制，这里可以采取多种方式，例如采取多层索引或者让索引块的最后一个指针指向下一个索引块，当然也可以兼用两种方法，例如 UNIX 文件系统的 inode 就是同时包含直接索引块和间接索引块。

上面所说的三种分配方式都各有优缺点，在不同的应用场景下要采取不同的分配方式。

为了充分利用磁盘的空间，文件释放之后，我们还需要利用这些空间，这就需要考虑要怎么样管理空闲的磁盘空间。一种方式是在内存中维护一个位表来标识某一个数据块是不是已被使用，但是这有个问题就是当磁盘空间太大时对内存空间的消耗有点大，还有一种方式就是利用指针把所有空闲的数据块连接起来，只需要维护一个起始地指针即可，还有一种方式是对直接空闲链表的改进，是指把 n 个空闲块地地址存在第一个空闲块中，最后一块又包含下一个 n 块空闲数据的地址。

即便是选择好了文件系统，也有很多可以方式来提高文件系统的性能，例如有的系统有一块独立内存用做缓冲储存，也有系统采用页面缓存，也有些系统结合两种方式采用双重缓存，用来缓存文件数据。

现代的文件系统还需要考虑文件恢复的问题，因为磁盘有时候会出错导致数据丢失，所以现在见到的操作系统大多是日志型文件系统，这些操作系统维护一些数据用来检查数据是否有损坏缺失以及文件的恢复所需要的信息。

参考文献格式:

[1]Operating System Concepts Essentials. Abraham Silberschatz. Wilay