第5章 文件管理

所有的应用程序在运行过程中都要存取信息，对信息的存取有三个基本要求：能够存储大量信息；使用信息的进程终止时，信息仍然存在，即信息能够被长期保存；多个进程能够并发存取有关信息，即实现信息的共享。

解决以上这些问题的最好方法就是把信息以一种单元，即文件的形式存储在可长久保存信息的磁盘或其他外部介质上。文件是由操作系统管理的，有关文件的组织、命名、存取、使用、保护等问题都是操作系统设计的主要内容。因此，将操作系统中处理文件的那部分称为文件系统(File System)。文件系统也是操作系统对用户最可见的部分。

从用户的角度来看，一个文件系统的主要内容应当包括文件如何进行组织，如何命名，如何保护，以及允许对文件执行什么操作，等等。对文件系统的设计者来说，怎样为文件分配存储空间、怎样记录和管理空闲存储区等内容则是相当重要的。本章主要从以上两个层面的内容分别进行详细论述。

5.1 文件

5.1.1 文件命名

文件(File)是具有名字的相关信息的集合。文件通常存放在外存上，它所表示的对象很广泛，一个源程序、一个可执行程序、一个文档、图形图像、学校的学生档案记录等，都可以构成一个文件。

当创建一个文件时，要给它命名。文件命名的规则随系统的不同而异，但几乎所有的现代操作系统都允许用1～8个字符作为合法的文件名。通常，文件名也允许有数字和一些特殊字符，像8、Fig7-1等都是合法的文件名；但是有些系统也规定了一些不能出现在文件名中的特殊字符，例如/、\、:、*、?、<、>等字符。许多文件系统支持长达255个字符的文件名。有的系统对文件名区分大小写字母，如UNIX系统；有的系统则不区分，如MS-DOS、Windows等。

大部分操作系统所支持的文件名都由两部分组成：文件名和扩展名，两者之间用圆点分开。扩展名通常用于指示文件的类型，例如，图片文件常常以JPEG格式保存并且文件扩展名为.jpg。一些常用的文件扩展名及其含义如图5-1所示。当双击某个文件时，就会启动与该扩展名相对应的应用程序。例如，双击student.xls文件，系统就会启动Microsoft

Excel 程序，并以 student.xls 作为待编辑的初始文件。

扩展名	含　义
exe	可执行文件
o,obj	目标文件
c,java	用 C 语言或 Java 语言编写的源文件
html	WWW 超文本标记语言文档
pdf	PDF 格式的文档
txt	记事本文档
doc	Word 文档
xls	Excel 文档
mp3	符合 MP3 音频编码格式的音乐文件
gif	符合图形交换格式的图像文件
jpg	符合 JPEG 编码标准的静态图片
hlp	帮助文档
mpg	符合 MPEG 编码标准的电影
zip	压缩文件

图 5-1　一些典型的文件扩展名

5.1.2　文件结构

文件结构就是文件的组织形式，通常用三种方式构造文件：字节序列文件、记录式文件和树状文件，如图 5-2 所示。

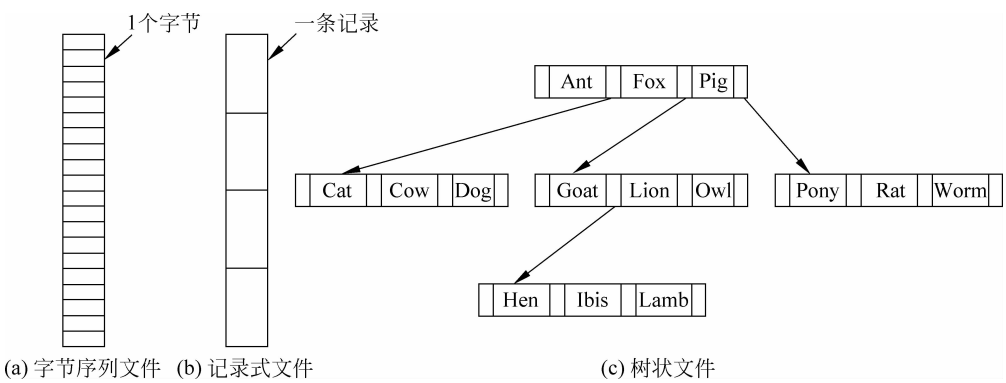

图 5-2　三种文件结构

1. 字节序列文件

字节序列文件也叫**流式文件**，是指用户创建文件时对信息不再划分单位，而是由一组相关信息组成的有序字符流来构成的文件。其长度就是它所包含的字节个数，如图 5-2(a)所示。大量的源程序、可执行文件、库函数等都采用流式文件的形式。在 UNIX 和 Windows 系统中，所有的文件都被看成流式文件。对于这种文件，操作系统不知道也不关心文件中存放的内容是什么，它所见到的就是一个一个的字节，其信息含义由用户程序解释。

把文件看成字符流，为操作系统提供了很大的灵活性。用户可以根据需要在自己的文

件中加入任何内容，不用操作系统提供任何额外的帮助。

2. 记录式文件

记录式文件在逻辑上是由若干条相关记录组成的，每条记录都有其内部结构，并且按照记录出现的逻辑顺序进行编号，依次为记录 0，记录 1，…，记录 n。每个记录都由一组相关的数据组成，这些数据通常描述一个对象的某些属性，如姓名、性别、年龄、专业、家庭住址等，如图 5-2(b)所示。对该类文件的操作是以记录为单位的：读操作返回一个记录，而写操作重写或追加一个记录。

记录式文件又可分为定长记录式文件和变长记录式文件两种。

(1) 定长记录式文件。文件中所有记录的长度都相同。文件的长度等于记录个数乘以每条记录的长度。这种文件开销小，处理方便，被广泛应用在数据处理中。

(2) 变长记录式文件。文件中各记录的长度不相同。一个记录中所包含的数据项的数目可能不同，例如论文中的关键词、书的作者等；数据项本身的长度也可能不同，例如会议记录中的摘要、病例中的病因等。但是不管哪一种，在处理之前每个记录的长度是可知的。变长记录式文件的长度等于各条记录的长度之和。

3. 树状文件

树状文件由一棵记录树组成，如图 5-2(c)所示，每个记录的长度可以不同。在记录的固定位置上有一个关键字字段，这棵树按照该字段进行排序，从而可以对特定关键字进行快速查找。

对文件中“下一个”记录的存取实际上是获得具有特定关键字的记录。例如，图 5-2(c)中的文件 zoo，用户可以要求系统取关键字为 Hen 的记录，而不必关心记录在文件中的确切位置。另外，也可以在文件中添加新记录，但把记录加在文件的什么位置是由操作系统而不是用户决定的。

由此可见，这种文件结构与 UNIX 和 Windows 中采用的无结构字节流文件有明显不同，它被广泛应用于某些商业数据处理的大型计算机中。

5.1.3 文件类型

文件按照不同的标准可以进行多种分类。

1. 按用途分类

(1) 系统文件：由系统软件构成的文件。这类文件对用户不直接开放，用户只能通过系统调用或系统提供的专用命令来执行它们，不允许对其进行修改。

(2) 库文件：由系统为用户提供的实用程序、标准子程序及动态重链接库等组成的文件。这类文件允许用户使用，但用户不能修改它们。例如：Java 函数库、C 语言子程序库等。

(3) 用户文件：由用户的信息(程序或数据)所组成的文件，只有文件的所有者或所有者授权的用户才能使用，他们可以对文件进行读、写或其他操作。该类文件主要由用户的源程序源代码、可执行目标程序的文件和用户数据库数据等组成，例如：*.cpp，*.java，*.exe。

2. 按存取权限分类

(1) 只读文件：只允许对其进行读操作，不允许进行写操作。

(2) 可读/写文件：允许文件的所有者或被授权的用户对其进行读或写操作。

(3) 可执行文件：允许被授权用户执行它，但通常不能对其进行读或写。

3. 按保存时间分类

(1) 临时文件：在一次工作过程中产生的中间文件，一般存放在系统暂存目录中，如temp 目录。正常工作情况下，临时文件在工作完毕会被自动删除；在异常情况出现时系统不能对某些临时文件及时清理，系统暂存目录中往往会残留不少临时文件，需要用户手动清理。

(2) 永久文件：长期保存的有价值的文件，以供用户经常使用。这些文件通常存放在软盘、硬盘或光盘等外存上，在用户没有发出撤销该文件的命令前，一直需要保存。

4. 按文件中的数据形式分类

(1) 源文件：用汇编语言或高级语言编写的源代码所形成的文件，例如 *.c，*.cpp，*.java 等文件。

(2) 目标文件：源程序经过相应的语言编译程序编译后，尚未经过连接处理的目标代码所形成的文件，它属于二进制文件，能够被 CPU 直接识别。

(3) 可执行文件：经过编译、连接之后所形成的可执行目标文件。

5. 按文件在系统内的组织形式分类

(1) 普通文件：用于存储信息的一般文件。这类文件既包括用户的各种文件，也包括系统文件、库文件、应用程序文件。

(2) 目录文件：由文件目录构成的一类用来维护文件系统结构的文件。通过目录文件，系统可以检索普通文件。目录文件也是由字符序列组成的，因此对它的处理(包括读、写或执行)与普通文件相同。

(3) 特别文件：有特定用途的文件。例如，在 UNIX 系统中，特别文件是指用于管理外部设备的文件。为了便于统一管理，系统把所有 I/O 设备都作为文件来对待，按文件格式提供给用户使用。例如目录查找、存取权限验证等方面与普通文件类似；而在具体读、写操作上，要针对不同设备的特性进行相应处理。特别文件分为字符特别文件和块特别文件。字符特别文件用于串行的 I/O 设备，如打印机、网络等。块特别文件用于磁盘类设备。

普通文件一般又分为 ASCII 文件和二进制文件。ASCII 文件又称文本文件，由多行正文组成，每行正文以回车符或换行符结束，各行的长度可以不同。ASCII 文件的最大特点是可以直接显示和打印，可用普通文本编辑器进行编辑，常用来存储程序源代码和文本数据。

二进制文件不能直接在终端上显示出来，打印出来的二进制文件是一张乱码表，无法被直接理解。它有一定的内部结构，只有使用该文件的程序才能了解其结构。图 5-3(a)是 UNIX 系统中的一个可执行的二进制文件，它有 5 个段：文件头、正文、数据、重定位标志及符号表。文件头由魔数(标志可执行文件的特征)、正文段长度、数据段长度、存放未初始化

的数据(Block Started by Symbol,BSS)段长度、符号表长度、入口单元及各种标志组成。文件头后面是程序本身的正文和数据,这些被装入内存,并使用重定位标志来重新定位。符号表用于调试程序。

二进制文件的另一个例子是**存档文件**,如图 5-3(b)所示。在 UNIX 系统中,它由已编译但未经连接的库过程(模块)集合组成。每个存档文件以模块头开始,其中记录了模块名、创建日期、文件拥有者、保护代码和文件长度。该模块头与可执行文件一样,也都是二进制数字,打印输出它们毫无意义。

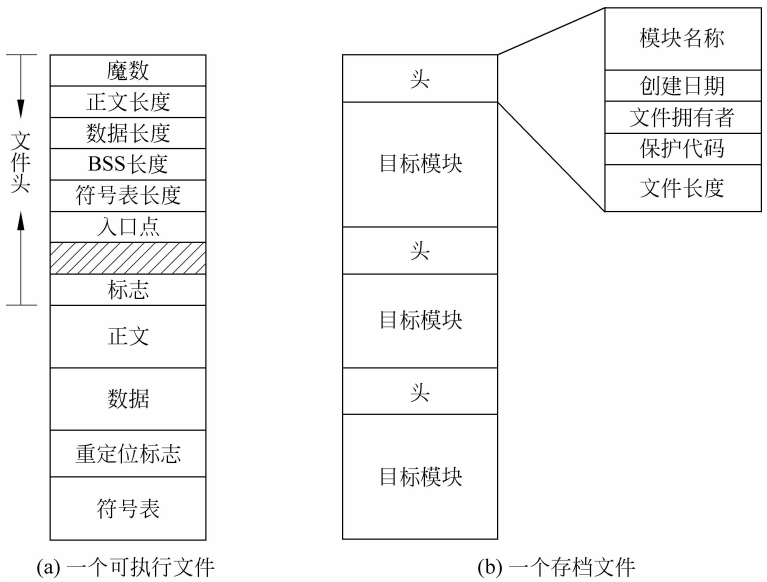

图 5-3 可执行文件和存档文件的内部结构

所有操作系统都必须至少识别一种它自己的可执行文件的类型,有些操作系统还可以识别多种文件类型。由于不同操作系统所识别的文件类型是不一样的,所以对文件进行操作时一定要注意其类型。

5.1.4 文件存取

文件的存取方法是由文件的性质和用户使用文件的情况决定的,它取决于用户以什么方式对文件中的信息进行定位。按照对信息存取顺序的不同,分为顺序存取和随机存取两种方法。

1. 顺序存取

顺序存取(Sequential Access)是严格按照字节出现的先后次序或记录排列顺序依次存取的一种方法。对文件的存取都是在前一次存取的基础上进行的,不允许跳过一些内容不按顺序存取。为了顺序存取,需要设置一个能自动前进的读/写指针,以动态指示当前读/写位置,根据要读/写的字节个数或记录长度,系统自动修改指针位置。在以磁带作为存储介质时,顺序存取文件是很方便的。

2. 随机存取

随机存取(Random Access)也叫直接存取(Direct Access),指允许以任意次序读取文件中的字节或记录,而不管上一次存取的是哪个字节或记录。例如,当前读记录 16,下次读记录 52,最后读记录 28。随机存取方式主要用于对大批信息的立即访问,如对大型数据库的访问。如果要支持用户以随机方式访问文件,文件必须存放在可以支持快速定位的随机访问存储设备中,例如磁盘。当接到访问请求时,系统计算出信息所在块的位置(可用专门的操作 seek 实现),然后直接读取其中的信息。

5.1.5 文件属性

文件有文件名和数据。此外,所有的操作系统还会保存与文件相关的一些信息。这些描述文件特征的信息称为文件属性,如文件的大小、文件创建的日期等。不同系统中所定义的文件属性不同。图 5-4 列出了一些常用的文件属性。需要说明的是,几乎没有任何一个系统会同时具有所有这些属性,但每种属性都会在某些操作系统中使用。

属　　性	含　　义
创建者	文件创建者的标识
所有者	当前文件的所有者
保护	谁可以存取文件,存取方式是什么
口令	存取文件需要的口令
只读标志	0 表示可读/可写;1 表示只读
隐藏标志	0 表示正常;1 表示不在列表中显示
系统标志	0 表示一般文件;1 表示系统文件
存档标志	0 表示已经备份;1 表示需要备份
ASCII/二进制标志	0 表示 ASCII 文件;1 表示二进制文件
随机存取标志	0 表示只能顺序存取;1 表示随机存取
临时标志	0 表示正常;1 表示进程结束时删除该文件
锁标志	0 表示未加锁;非 0 表示加锁
记录长度	一个记录的字节数
关键字的位置	每个记录中关键字的偏移量
关键字的长度	关键字字段中的字节数
创建时间	创建文件的日期和时间
最后存取时间	文件上一次存取的日期和时间
最后修改时间	文件上一次修改的日期和时间
当前长度	文件的字节数
最大长度	文件允许增加到的最大字节数

图 5-4　一些常用的文件属性

图中前 4 个属性与文件保护有关,指明谁有权存取该文件。中间的一些标志位用于启用或禁止某些特殊属性。例如隐藏文件不能在文件列表中出现;如果希望在进程结束时删除临时文件,则应将临时标志位置位等。记录长度、关键字位置、关键字长度等字段只能出现在用关键字查找记录的文件里,它们提供了查找关键字所需的信息。各个时间字段分别记录了文件的创建时间、最后一次存取时间以及最后一次修改时间,它们有各自的用途。例

如，目标文件生成后源文件被修改了，那么就需要重新编译源文件。当前长度字段指出了当前文件的大小。在一些老式的大型机操作系统中，当创建文件时，需要给出文件的最大长度，以便操作系统事先按照最大长度预留存储空间。工作站和个人操作系统则不需要这个属性。

5.1.6 文件操作

文件系统提供了对文件操作的各种命令，来实现对文件的存取。当然，不同的操作系统提供的命令也是不同的。下面是与文件有关的常用的系统调用。

1. 创建文件

创建(create)一个文件的主要功能是在文件系统中为该文件分配必要的空间，然后生成一个新的目录项，添加到相应的目录中。目录项中记录着文件的名字、文件类型、建立时间等有关文件属性的信息。新创建的文件不包含任何数据，大小为 0。

2. 删除文件

当不再需要某个文件时，可用命令将它删除(delete)，以便释放它所占用的磁盘空间。若要删除一个文件，首先需要根据文件的路径名找到指定的目录项，根据目录项回收文件占用的各个物理块，再将文件的目录项清空。此后，文件在系统中就不复存在了。

3. 打开文件

在使用文件之前，系统必须先打开(open)该文件。执行打开文件时，根据给定的文件名找到相应的目录项，将文件的目录项复制到主存的一个专门区域，并返回文件在该区域的索引，从而建立进程与文件的联系。打开文件的目的是把文件属性和磁盘地址表等信息装入主存，以便后续的系统调用能够快速地存取该文件，从而避免多次重复地检索文件目录。

4. 关闭文件

当进程对文件的存取结束后，不再需要文件属性和磁盘地址，这时应该关闭(close)文件以释放它所占用的内存空间。如果该文件的目录项在调入内存后被修改过，则还要回写到磁盘。文件关闭后，不能再使用；若要使用，需要再打开。

5. 读文件

从文件中读取(read)数据。一般来说，读取的数据来自文件的当前位置，命令中还应该指明要读取数据的数量以及存放这些数据的主存地址(缓冲区)。

6. 写文件

将数据写(write)到文件中。通常将要写的数据写到文件的当前位置，如果当前位置是文件末尾，则文件内容增加；如果当前位置在文件中间，则覆盖现有数据。

7. 追加文件

追加(append)调用限制了写文件的形式,只允许将数据添加到文件的末尾。如果系统只提供最小系统调用集合,则通常没有 append。很多系统对同一操作提供了多种实现方法,这些系统往往包含 append。

8. 随机存取文件

对于随机存取(seek)文件,必须提供一种方法说明从何处开始读/写文件。seek 系统调用就是用来将文件的当前位置指针重新定位到指定位置的。当该系统调用执行完后,就可以从那个位置读/写数据了。

9. 获得文件属性

进程在执行过程中通常需要了解文件的属性,例如文件的建立时间和最后修改时间等。系统根据调用者提供的文件名从目录中找到相应的目录项,再从目录项中获取该文件的相关属性。

10. 设置文件属性

文件的某些属性可以由用户设置,并且在文件创建后,用户可以通过 set attribute 系统调用修改文件属性。例如,修改文件的存取保护方式等。通常,修改文件属性的用户必须是文件的所有者或特权用户。系统根据给定的文件名从目录中找到相应的目录项,然后按照指定的文件属性修改原有的信息。

11. 重命名文件

当用户需要改变某一现存文件的名字时,可以使用该系统调用。实际上,这个系统调用并不是必需的,因为可以通过把老文件复制到一个新文件中,然后再删除老文件的方法达到重命名的目的。

5.2 文件系统的功能和结构

现代操作系统中都实现了比较完备的文件系统。所谓文件系统就是操作系统中提供管理文件的软件机构,它负责操纵和管理文件。从系统角度来看,文件系统是对文件存储器的存储空间进行组织、分配和回收,负责文件的存储、检索、共享和保护。从用户角度来看,文件系统主要是实现“按名取存”,用户只要知道所需文件的名字,就可存取文件中的信息,而不用考虑其信息存放在磁盘的哪个柱面、哪个磁道、哪个扇区上,也不必关心启动设备进行 I/O 操作的具体实现细节。从某种意义上来说,文件系统提供了用户与外存之间的界面。

5.2.1 文件系统的功能

一般来说,文件系统应该具备以下 5 种功能:

(1) 文件管理。按照用户的要求创建一个文件,对文件执行打开、关闭、读、写、删除、重命名等操作。

(2) 目录管理。为每个文件建立一个目录项,若干个目录项的集合构成一个目录文件。按照用户要求创建或删除目录文件,对用户指定的文件进行检索和权限验证等。

(3) 文件的共享和保护。一个文件可以通过设置共享来实现被其他用户所访问。为了防止用户对文件的非授权或越权访问,文件系统应该提供可靠的保护措施,如采用口令、加密、存取权限等。

(4) 文件存储空间的管理。记录哪些空间被占用、哪些空间空闲,以便用户创建文件时为其分配存储空间;修改或删除文件时,调整或回收相应空间。

(5) 提供方便的接口。为用户提供统一、方便的接口,主要是有关文件操作的系统调用,供用户编程时使用。

5.2.2 文件系统的结构模型

文件系统变化较大,不同的操作系统有不同的文件系统。但文件系统的大概模型是如图 5-5 所示的层次模型。

<table>
<tr><td colspan="2">文件系统接口</td></tr>
<tr><td rowspan="4">操纵并管理对象的软件集合</td><td>逻辑文件系统层</td></tr>
<tr><td>基本 I/O 管理程序层(文件组织模块)</td></tr>
<tr><td>基本文件系统层(物理 I/O 层)</td></tr>
<tr><td>I/O 控制层(设备驱动程序)</td></tr>
<tr><td colspan="2">对象(文件、目录及磁盘存储空间)及其属性说明</td></tr>
</table>

图 5-5 文件系统模型

1. 文件系统接口

为了方便用户使用文件系统,文件系统通常向用户提供两种类型的接口。

(1) **命令接口**:作为用户与文件系统交互的接口,用户可通过键盘终端输入命令,取得文件系统的服务。

(2) **程序接口**:作为用户程序与文件系统的接口,用户程序可通过系统调用来取得文件系统的服务。例如,用于打开一个文件的系统调用 open;用于删除一个文件的系统调用 delete 等。

2. 操纵并管理对象的软件集合

这是文件管理系统的核心部分,文件系统的功能大多是在这一层实现的,其中包括:对文件存储空间的管理、对文件目录的管理、用于将文件的逻辑地址转换为物理地址的机制、对文件读和写的管理,以及对文件的共享与保护等功能。在这里又分成几个子层。

(1) **逻辑文件系统层**:处理文件及记录的相关操作(访问、保护及目录操作)。

(2) **基本 I/O 管理程序层**:完成大量与磁盘 I/O 有关的工作(选择设备,逻辑块号到物理块号的转换,空闲空间管理等)。

(3) **基本文件系统层**:负责内存与磁盘间的数据块交换(在外存及内存缓冲区的

位置)。

(4) **I/O 控制层**：负责启动 I/O 操作及处理设备发来的中断信号。

3. 管理的对象及其属性

(1) **文件**：文件系统的直接管理对象。

(2) **目录**：为了方便用户对文件的存取和检索，在文件系统中必须配置目录。对目录的组织和管理是方便用户和提高对文件存取速度的关键。

(3) **磁盘(磁带)存储空间**：文件和目录必定占用存储空间，对这部分空间的有效管理，不仅能提高外存的利用率，而且能提高对文件的存取速度。

5.3 目录

文件系统为用户和程序提供了按名存取文件的机制，而将文件名转换为外存上的存储地址及对文件进行控制管理则需要通过目录来实现。在很多系统中，目录本身也是文件。

5.3.1 文件控制块和文件目录

1. 文件控制块

每当创建一个进程，便给它分配一个进程控制块(PCB)，用它记录与进程相关的各种信息。同样，对于文件也有相应的控制结构——文件控制块(File Control Block，FCB)，用它来对文件进行控制和管理。文件控制块是文件存在的标志，不同的操作系统中 FCB 包含的信息也不同，通常有：

(1) 文件名。用于标识一个文件的符号名，由文件的创建者定义，如 file. doc，score . xls 等。

(2) 文件类型。指明文件的类型，是普通文件、目录文件还是特别文件，是文本文件还是二进制文件，是用户文件还是系统文件等。

(3) 文件物理位置。指明文件在外存上的存储位置，包括存放文件的设备名、文件在外存的起始地址等。

(4) 文件的大小。当前文件的大小和文件允许的最大值。文件大小通常以字节、字或者块为单位。

(5) 文件的存取控制信息。标明对文件读、写及执行等操作的控制权限。

(6) 文件使用的信息。已打开该文件的进程数，文件被修改的情况。

(7) 时间。文件的创建日期、修改日期等。

(8) 口令。用于实现文件保护的，将用户设定的口令保存在文件控制块中以便系统核对，以此增加文件的安全性。

(9) 共享说明。文件可以被多个用户所共享，在共享说明中指出哪些用户可以共同使用该文件，文件的所有者及其授权者的用户名，有时还指出被授权用户共享该文件的使用权限。

上述内容多数是用户在建立文件时提供的，以后由相应操作也可以补充说明。

2. 文件目录

每个文件有一个文件控制块，文件与文件控制块一一对应，而文件控制块的有序集合即为文件目录（或者简称为目录）。换言之，一个文件控制块就是一个文件目录项，文件目录是需要长期保存的。为了实现文件目录的管理，通常将文件目录以文件的形式保存在外存中，这个文件就被称为目录文件。

文件目录能够实现文件名与磁盘块之间的映射，即将文件名转换成该文件在外存的物理位置，这也是文件目录所提供的最基本的功能。

5.3.2 目录结构

文件类似于图书馆的图书，目录就相当于图书书目表（一览表）。当图书馆的图书较少时，一本书目表就足以列出所有的图书。但是，当图书馆的图书逐渐增多时，就应该设置合理的书目表结构，例如在书目表中对图书进行分类，甚至为每类图书设立一本书目表。同样，计算机系统中的目录也存在多种结构，常用的目录结构有单级目录、二级目录和多级目录。

1. 单级目录结构

单级目录结构是指文件系统在每个存储设备上仅建立一个目录，即根目录，系统将所有的文件都放在这一个目录下。每当创建一个新文件时，就在目录表中找一个空的目录项，把文件的相关信息填在该目录项中。当删除一个文件时，从目录中找到该目录项，回收该文件占用的外存空间，并清空其所占用的目录项。图 5-6 是一个单级目录结构的例子，图中给出的是文件的所有者而不是文件名。该目录中有 4 个文件，其中用户 B 有两个文件，用户 A 和用户 C 各拥有一个文件。

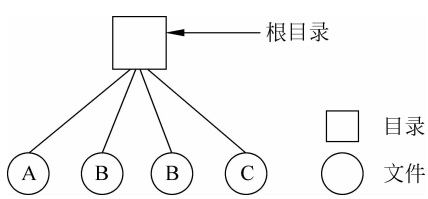

图 5-6 单级目录结构

单级目录结构的优点是结构简单，管理成本低，比较适合单用户文件管理的需要。但是它也有以下缺点：

（1）不允许文件重名。由于命名规则规定在一个目录中不允许两个不同的文件具有相同的名字，所以整个系统中不会有文件重名。这对于多用户的系统来说则是不方便的。例如，用户 A 创建了一个名为 prog. c 的文件，接着用户 B 也想创建了一个名为 prog. c 的文件，系统将不允许。

（2）查找速度慢。查找一个文件有时需要查找到整个目录中的所有目录项，开销大、速度慢。例如，在一个具有 n 个目录项的单级目录中查找一个目录项，平均需要查找 $n/2$ 个目录项。

（3）不便于文件共享。对于同一个文件，不同的用户可能有不同的命名习惯，而单级目录无法满足用户的这一需求，单级目录要求所有用户用同一名字来访问同一个文件。

2. 二级目录结构

解决文件重名问题的一个办法就是采用二级目录，即为每个用户提供一个私有目录。在这种方式中，各个用户对文件的命名彼此独立，互不影响，允许在两个或多个目录中存在相同名字的文件。如图 5-7 所示，系统中有 A，B，C 三个用户目录，假设在 A 目录下有一个文件名为 file. doc，那么在 B 目录下也可以有一个文件命名为 file. doc，同样在 C 目录下也可以有一个文件以 file. doc 命名。

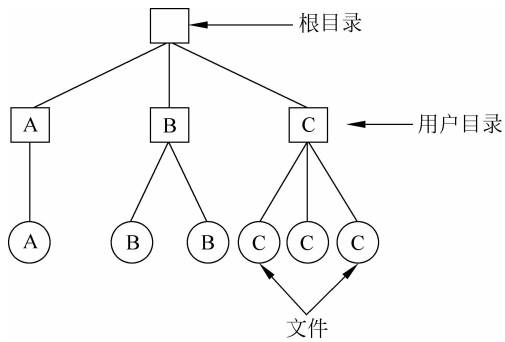

图 5-7　二级目录

在二级目录系统中，为每个用户所建立的目录叫做用户文件目录（User File Directory，UFD），只列出每个用户的文件。系统维护一个主文件目录（Master File Directory，MFD），记录各个用户名字及该用户文件目录所在的物理地址。二级目录结构如图 5-8 所示。

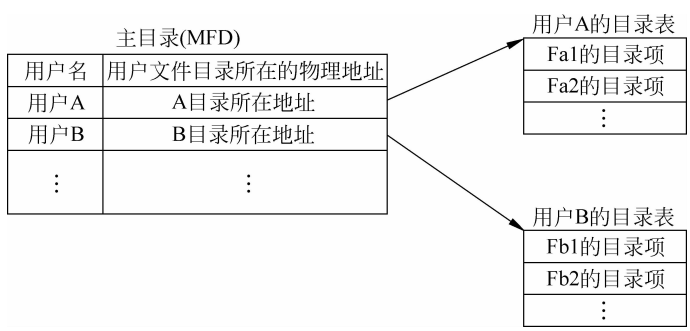

图 5-8　二级目录树结构

当新创建一个用户时，系统先为该用户分配一个记录用户文件的目录块，然后在主文件目录表（MFD）中找一个空的表项分给该用户，并记录该用户名和用户文件目录块地址。此后，当用户新创建一个文件时，文件系统根据用户名索引主目录找到该用户所对应的文件目录的起始地址，然后在用户文件目录表中为该文件建立一个目录项，记录文件名及文件的其他信息。当用户访问一个文件时，文件系统也是根据用户名索引主目录找到该用户所对应的文件目录位置，再根据文件名索引用户文件目录找到该文件的有关信息，最后对文件进行

相应操作的。当用户要删除一个文件时，也是通过主目录找到用户文件目录，再在用户文件目录中找到该文件，删除该文件的目录项的。想要删除某个用户时，只有当该用户的文件目录中的所有文件都被删除时，系统回收该文件目录块，并且撤销该用户在主目录表中的目录项，此后该用户在主目录中就没有记录信息了。

3. 多级目录结构

二级目录结构虽然解决了文件重名问题，但是当用户拥有大量不同类型的文件时，该结构使用起来还是不方便。用户通常希望把文件按照某种逻辑方式组织起来。例如，某位老师讲授多门课程，他希望将每门课程的教学文件组织在一起；同时，他还有工作文档、科研论文、电子邮件、电影等一系列文件。所以，需要提供某种方法使用户按照自己的方式组织文件。

现代操作系统向用户提供了多级目录结构，也叫层次式的目录结构。在这种结构中，每一级目录中可以包含下一级目录，也可以包含文件。这样，从根目录开始，一级一级地扩展下去，形成一个树状层次结构，如图 5-9 所示。每个目录的直接上一级目录称为该目录的父目录，而它的直接下一级目录称为子目录。在树状结构的文件系统中，只有一个根目录。除根目录外，每个目录都有父目录。有了多级目录结构，每个用户可以根据自己的需要创建多个目录，自由地组织自己的文件。

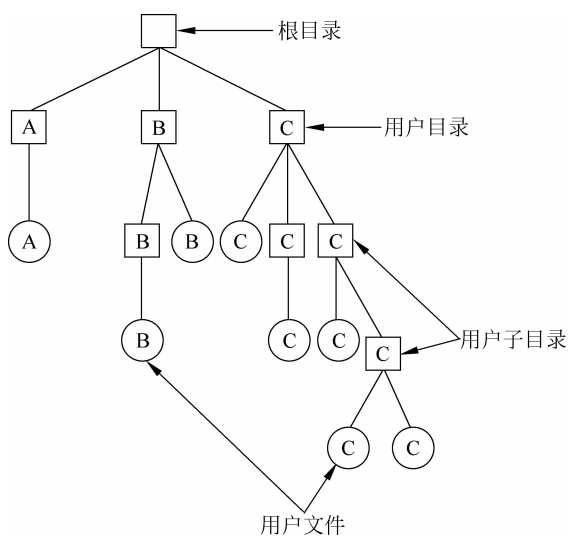

图 5-9　层次目录结构

在多级目录结构中，对文件的访问必须采用路径名。路径名有两种表达方式：绝对路径名(Absolute Path Name)和相对路径名(Relative Path Name)。

(1) 绝对路径名。**绝对路径名**是指从根目录出发最终到达文件的整个完整路径的名字。例如，图 5-10 中，路径名/doc/txt/score 表示根目录中有子目录 doc，doc 下又有子目录 txt，txt 下有个文件 score。绝对路径名一定是从根目录开始的，而且是唯一的。

在 UNIX 中，路径各部分之间用/分隔；而 Windows 中，分隔符是\；在 MULTICS 中，分隔符是>。因此，在这三个系统中，同样的文件其路径名分别表达为

```
UNIX          /doc/txt/score
Windows       \doc\txt\score
MULTICS       > doc > txt > score
```

由此可见，不管采用哪种分隔符，只要路径名的第一个字符是分隔符，那么这个路径就是绝对路径。

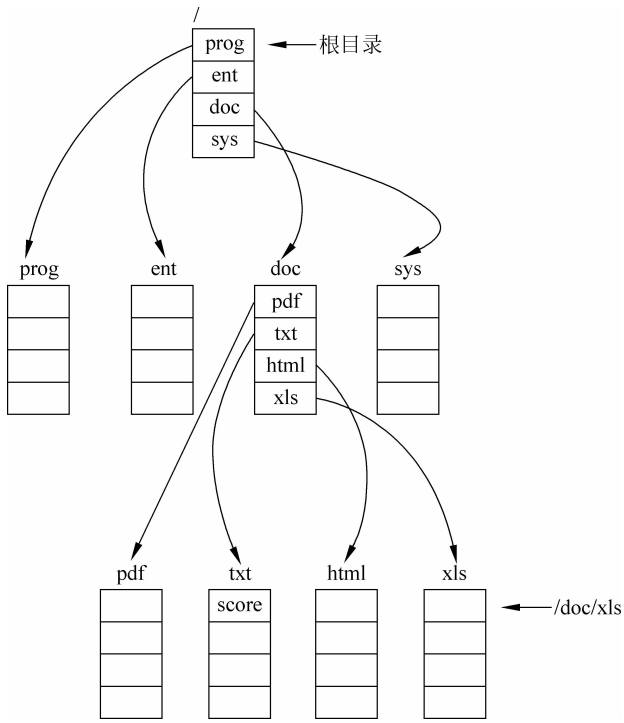

图 5-10 UNIX 目录树

(2) 相对路径名。在层次目录结构中，如果每次访问文件都从根目录开始检索，很不方便，而且会花费较长时间。因此，引入“当前目录”(又称工作目录)来解决这一问题。用户根据自己的需要指定一个目录作为当前目录，当访问某个文件时，就从当前目录开始向下检索。由于当前目录是在根目录下并且靠近常用文件的一个目录，所以会缩短检索路径，从而提高了处理速度。所有不从根目录开始的路径名都是相对于当前目录的。把从当前目录开始直到最终要访问的文件为止所构成的路径名称做**相对路径名**。例如，如果当前的工作目录是/doc/txt，则绝对路径名为/doc/txt/score 的文件可以直接用 score 来引用。也就是说，如果工作目录是/doc/txt，则在 UNIX 系统中的命令

```
cp /doc/txt/score /doc/txt/score.bak
```

和命令

```
cp score score.bak
```

具有相同的含义。

在层次目录结构中，大多数操作系统提供两个特殊的目录项：.和..，常读作 dot 和 dotdot。.指当前目录，..指其父目录。考虑图 5-10，一个进程的工作目录是/doc/pdf，它可采用..沿树向上回溯。例如，可用命令

```
cp ../txt/score .
```

把文件/doc/txt/score 复制到自己的工作目录下。第一个路径告诉系统向上回溯到 doc 目录，然后向下到 txt 目录，找到 score 文件。第二个参数.指定当前目录。当 cp 命令用一目录名(包括“.”)作为最后一个参数时，则把全部的文件复制到该目录下。

因此，对于上述的复制工作，输入

```
cp /doc/txt/score .
cp /doc/txt/score score
cp /doc/txt/score /doc/pdf/score
```

都完成同样的工作。

多级目录结构的优点是：层次结构清晰，便于文件的管理和保护；便于文件分类，可为每类文件建立一个子目录；有效地解决了文件重名问题；检索速度快，因为每个目录下的文件数目较少；可以实现文件共享。缺点是比较复杂。

5.3.3 目录查询技术

当访问一个已存在的文件时，系统首先利用用户提供的文件名对目录进行查询，找出该文件的文件控制块(FCB)；然后根据 FCB 中所记录的文件物理地址，计算出文件所在磁盘的物理位置；最后，再通过磁盘驱动程序，将所需文件读入内存。常用的目录查询方式有：线性检索法和 Hash 法。

1. 线性检索法

又称为顺序检索法，在单级目录中，利用用户提供的文件名，用顺序查找法直接从文件目录表中找到指名文件的目录项。

在单级目录结构中，依次扫描文件目录中的目录表项，将目录表项中的名字与目标文件名相比较，直到找到匹配的文件。树状目录结构则根据树状结构逐级查找，一般情况下从根目录查起，为了节省时间也可以从当前目录查起。例如，在图 5-10 中要查找用户定义的一个文件 score，其文件路径名为：/doc/txt/score 其查找过程说明如下：

首先从根目录查起，找出根目录中的各目录项与 doc 相比较直至找到目录 doc；再取出 doc 中各个目录表项与 txt 相比较，依次下去，直至找到 score 文件。如果在查找过程中，发现一个文件未能找到，则应停止查找并返回“文件未找到”信息。

为了提高查找速度，可以设置“当前目录”，即把工作目录从根目录或某级子目录改变到离查找目标较近的那个目录，从那儿开始查找。在上例中工作目录可以改为/doc/txt，就能非常快地找到文件 score。现代操作系统都设置有改变工作目录命令，这给用户在查找文件时提供了很大的方便。

2. Hash 法

所谓 Hash 法就是一种“散列法”或称“杂凑法”，是一种构造符号表、查询符号表常用的技术。其基本思想是，利用一个易于实现的变换函数(即 Hash 函数)，把每个符号名唯一地变换成符号表中的表目索引。

假定有效的文件符号名为 6 个字符，不足 6 个字符须用空格补足，多于 6 个字符时取前 6 个。一个简单的 Hash 函数就是把各个字符的 ASCII 代码“异或”在一起，把求得的 Hash 值作为符号文件目录的索引来用。例如，完全限定的文件名为 ANDING. SQRT，则由 ANDING 的 Hash 值作为主目录的索引号；而 SQRT 的 Hash 值作 ANDING 的符号文件目录 SFD 的索引号。Hash 函数值也称为 Hash 索引，ANDING 的 Hash 索引为

$$\mathrm{HINDEX(ANDING)=Hash(ANDING)=A\oplus N\oplus D\oplus I\oplus N\oplus G=(1011)_2=(11)_{10}}$$

SQRT 的 Hash 索引为

$$\mathrm{HINDEX(SQRT)=Hash(SQRT\_\_\_)=(100)_2=(4)_{10}}$$

其中␣为空格符。在构造符号文件目录时，同样根据相应符号名的 Hash 索引来确定它在目录中的位置。

这种简单的 Hash 技术存在着一个问题，就是几个符号名有可能被变换成同一个 Hash 索引，即所谓“冲突”。Hash 法是一种处理文件名冲突的有效方法，解决的办法有三种：

第一种办法是，把符号文件目录做成二维表，对求得的每个 Hash 索引均设立若干列(例如 4 列)用来存放实际表目，如图 5-11 所示。这样，在建立表目时，同一 Hash 索引的符号名及其标识符 ID 可存入同一行的不同列上。当存取文件进行查表时，在相应的行上顺序查找匹配的文件名。

Hash 索引	0	1	2	3
0	表目$_{0,0}$	表目$_{0,1}$	表目$_{0,2}$	表目$_{0,3}$
1	表目$_{1,0}$	表目$_{1,1}$		
2	表目$_{2,0}$			
255	表目$_{255,0}$			表目$_{255,3}$

图 5-11　二维符号文件目录

第二种办法和二维目录类似，就是每当发现了不匹配或非空表目时，给 Hash 索引加上一个位移常数。这个常数与目录尺寸 N(即表目数)互质。这种技术允许在计算 Hash 索引后对符号文件目录进行线性查找，其查法如下：

(1) 在利用 Hash 法索引查找目录时，如果目录表中相应的目录项是空的，则表示系统中并无指定文件；

(2) 如果目录项中的文件名与指定文件名相匹配，则表示该目录项正是所要寻找的文件所对应的目录项，故而可从中找到该文件所在的物理地址；

(3) 如果在目录表的相应目录项中的文件名与指定文件名并不匹配，则表示发生了“冲突”，此时须将其 Hash 值再加上一个常数(该常数应与目录的长度值互质)，形成新的索引

值，再返回到第一步重新开始查找。

例如，目录尺寸为 256，位移常数为 51，用户名 ANDING 的初始索引为 11，如果发现该表目中的名字字段不为 ANDING 时，则可进一步查找索引为 62、113、164 等的表目，直至找到匹配的表目或新的索引号又为 11 为止。

第三种办法是采用溢出处理技术，如图 5-12 所示。即把具有相同 Hash 索引的文件说明存放在同一物理块中，当其发生溢出时，便向系统另申请一物理块，并把它们相应地链在一起。

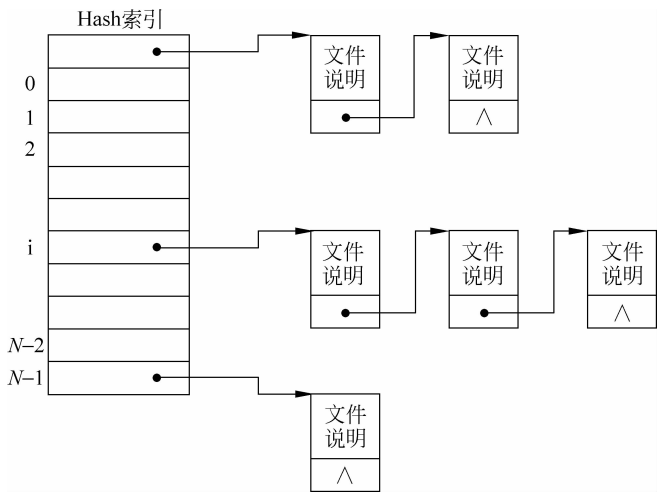

图 5-12　Hash 索引目录文件的溢出处理

5.3.4　目录操作

与文件操作类似，操作系统也提供了一组系统调用来实现对目录的操作。不同操作系统中对目录操作的系统调用的差别很大。下面以 UNIX 系统为例，描述有关目录的系统调用。

1. 创建目录

被创建(create)的新目录中除了目录项.(表示该目录本身)和..(表示父目录)以外，目录内容为空。目录项.和..是系统自动放在目录中的。系统首先根据路径名检索目录，如果存在同名目录文件，则提示出错信息；否则，为新目录文件分配必要的磁盘空间和控制结构，并进行初始化；然后将新目录文件对应的目录项添加到其父目录中。

2. 删除目录

删除(delete)目录有两种策略：

(1) 不删除非空目录。当目录中有文件时不能将其删除，而为了删除一个非空目录，必须先删除目录中的所有文件，使之先成为空目录后才可以删除。如果目录中还包含子目录，还必须采取递归方式来将其删除，在 MS-DOS 中就是采用这种删除策略的。

(2) 可删除非空目录。当要删除一个目录时,如果在该目录中还包含文件,则目录中的所有文件和子目录也同时被删除。目前大多数操作系统都采用这种策略。

3. 打开目录

目录中的内容可以被读取。与打开和读文件类似,在读目录之前,必须先打开目录。例如,要列出一个目录中的所有文件名,程序必须先打开该目录(opendir),然后读其中全部文件的文件名。

4. 关闭目录

在读取目录之后,应该关闭目录(closedir),从而释放其所占用的内部表空间。

5. 读目录

该调用返回打开目录的下一个目录项。以前系统利用读文件的系统调用(read)来读目录(readdir),但这个方法有一个缺点:程序员必须了解目录的内部结构。而 readdir 总是以标准格式返回一个目录项,这样,程序员就不用关心所用的目录结构了。

6. 重命名目录

与文件重命名(rename)类似,也可以对目录重新命名。

7. 链接文件

链接(link)允许一个文件在多个目录中出现,通过多条不同的路径存取同一个文件,从而实现对该文件的共享。link 这个系统调用指定一个存在的文件和一个路径名,并建立从该文件到该路径名之间的链接。这种类型的链接增加了源文件的链接计数值,有时将这种链接称为**硬链接**。

8. 解除链接文件

当不需要对某个文件链接共享时,可以解除链接(unlink)。通常情况下,如果被解除链接的文件只出现在一个目录中,则将该文件从文件系统中删除。如果它出现在多个目录中,则只删除指定路径名,其他的路径名仍然保留。

5.4 文件系统的实现

上面主要从用户的角度考察了文件系统,用户所关心的是文件是如何命名的、可以进行哪些操作,目录的结构是什么样的以及相应的操作。下面将目光转向系统设计者,系统的设计者所关注的是怎样为文件和目录分配存储空间、磁盘空间是如何管理的,以及怎样使系统有效可靠地工作等问题。

5.4.1 文件系统的格式

如前所述,大多数磁盘在使用之前被划分成一个或多个分区,每个分区都有一个独立的

文件系统。磁盘的0号扇区存放着磁盘的分区信息，称为主引导记录(Master Boot Record，MBR)，用它来引导计算机。在MBR的末尾是分区表，给出每个分区的起始地址和结束地址，并且表中有一个分区被标记为活动分区(默认引导分区，该分区中安装有一个操作系统)。MBR中包含一小段程序，当计算机上电启动时，BIOS读入并执行MBR，执行MBR的结果是读入分区表并确定系统的活动分区，读入活动分区的第一个块——引导块(Boot Block)，并执行之。引导块中的程序把该分区中的操作系统装入内存，从而系统启动完成。为了统一，为每个分区的起始处都保留一个引导块。即使该分区中目前不包含可引导的操作系统，但是并不意味着该分区中永远不安装操作系统，将来也许就会在这个分区中安装一个操作系统。

除了引导块之外，不同文件系统的分区格式差别很大。通常，文件系统的格式包含如图5-13所示的内容。超级块(Super block)中存放该文件系统的关键参数，包括标识文件系统类型的幻数、文件系统中数据块的数量、修改标识及其他关键管理信息等。当计算机启动时，或者在该文件系统首次使用时，把超级块中的信息读入内存。

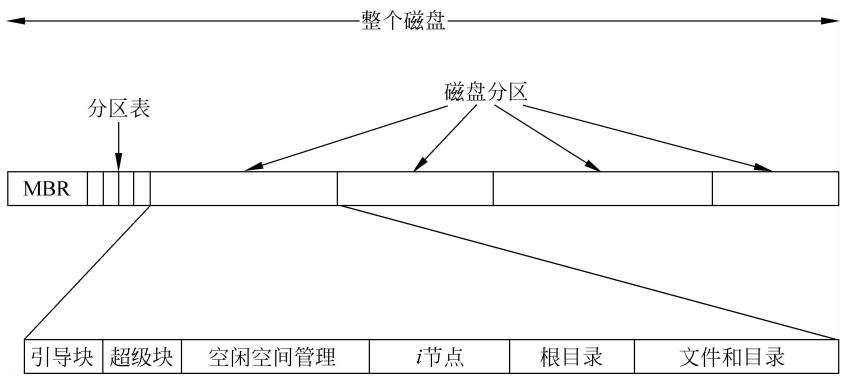

图5-13 一种可能的文件系统的格式

超级块之后是有关空闲块的信息，可以用指针链表形式表示，也可以用位图形式表示。后面跟的是i节点(i-node)，这是一个数据结构数组，每个文件都有一个i节点，其中记录有关该文件的所有管理信息。接着是根目录，它存放文件系统目录树的根部。最后，分区的其余部分存放除了根目录之外的所有其他的目录和全部文件。

5.4.2 文件存储空间的分配

文件存储在磁盘上，从而满足文件长期存储的需求。到底将文件存储在哪些物理磁盘块上，换言之，系统为文件分配哪些磁盘空间，这是文件系统要解决的一个问题。所有文件在逻辑上都是连续的，但是在物理介质上存放时却不一定连续。不同的操作系统采用不同的分配方法，其分配方法的优劣，将直接影响对文件的操作和文件系统的性能。文件存储空间的分配涉及以下几个方面的问题：

(1) 静态分配还是动态分配：静态分配是指创建一个文件时就给文件一次性分配所需的最大文件存储空间；动态分配是指随文件动态增长动态分配所需的文件存储空间。

(2) 分区大小：分区大小的选择不仅应该考虑单个文件的效率，而且还要考虑整个系

统的效率。一般来说有两种选择：一种是可变的连续分区；另一种是定长的分区。

(3) 文件空间的管理：指采用什么数据结构来描述分配给文件的磁盘块信息，一般采用文件分配表(File Allocation Table，FAT)来进行管理。

(4) 文件分配方法：在实现文件存储中最重要的问题是如何记录各个文件分别用到哪些磁盘块，这种记录各个文件分别用到哪些磁盘块的方法就称为文件分配方法。

总之，如何才能有效地利用文件存储空间、提高对文件的访问速度、记录各个文件分别用到哪些磁盘块是文件存储空间分配时要考虑的主要问题。不同操作系统采用不同的方法，下面讨论其中一些常用的方法。

1. 连续分配

连续分配是最简单的一种分配方案，其思想就是给一个文件分配一组连续的磁盘块，即使逻辑上连续的信息文件在物理介质上也是连续存放的。如图 5-14 所示，从磁盘开始处连续的 5 个磁盘块分给文件 A，即文件 A 所占用的磁盘块依次是 0，1，2，3 和 4；紧接着，在文件 A 的结尾处开始写入文件 B，占用了连续的 4 个块，即磁盘块 5，6，7 和 8 分给了文件 B。以此类推，共写入了 6 个文件。需要注意的是，每个文件都是从一个新的块开始写入的，假设即使文件 A 的最后一个块只用了 1/2，那么剩下的 1/2 也不能分给另外一个文件，即一个物理块只能分给一个文件使用，以块为单位进行分配。

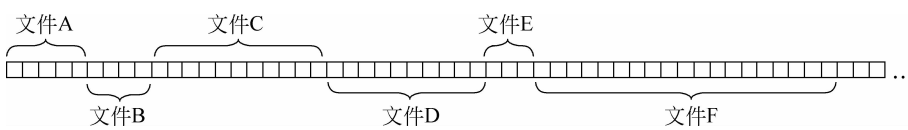

图 5-14 连续分配

连续磁盘空间分配方案有三个优点。第一，实现简单，只要记住文件第一块的磁盘地址和文件所占的块数即可。第二，支持顺序存取和随机存取。只要找到第一个磁盘块，顺着往后就可以依次存取后面的所有块。另外，也很容易直接存取文件中的任意一块，即给定了第一块的编号，只要做一个简单的加法运算就可以找到任何其他块的编号。例如，文件的起始块的编号是 m，则访问该文件的第 n 块的编号就是 $m+n-1$。第三，存取速度快，只要访问一次文件的管理信息，就可以方便地存取到任一记录。

但是，连续磁盘空间分配方案也有明显的缺点。第一，不灵活。要求在文件创建时，就给出文件需要的长度，这往往很难实现。第二，不便于文件的动态扩充。例如，作为输出结果的文件往往随执行过程而不断增加新内容，它的长度是要动态增加的。当该文件需要扩大空间而其后面的磁盘块已被分配给其他文件时，就必须另外寻找一个足够大的空间，把原空间的内容和新增加的内容复制进去。这种文件的搬移是很费时间的。第三，容易产生碎片。随着文件的写入和删除，可能会出现许多小的无法利用的空洞，虽然可以采用磁盘紧缩的方法解决，但由于代价高很少采用。

2. 链表分配

为了避免连续分配中磁盘碎片的问题，可以采用链表分配方法。如图 5-15 所示，链表分配方法是为每个文件构造一个磁盘块链表，每个块的第一个字作为指向下一个块的指针，

块的其余部分用来存放数据。文件的最后一个磁盘块的指针通常为0,以指示该块是链尾。

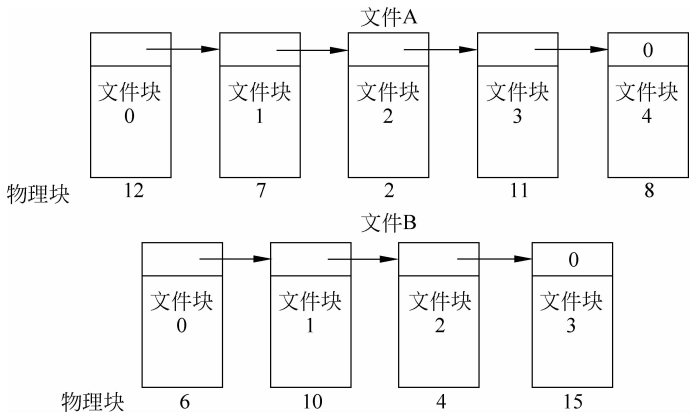

图 5-15 以磁盘块链表的形式存储文件

因此,链表分配方法将逻辑上连续的文件分散存储在不连续的磁盘块中,从而有效地利用每个磁盘块,不会因为磁盘碎片而浪费存储空间。在这种分配方法中,目录项只需要存放第一个块的磁盘地址,文件的其他块就可以顺着这个首地址依次查找到。同时,该方案允许文件动态增长,增加了使用的灵活性。

但是,这种方案也存在以下的缺点:

(1) 顺序访问文件比较方便,但不利于对文件的随机存取。为了存取一个文件的第 n 块的信息,必须从头开始,先读前面的 $n-1$ 块之后,才能找到第 n 块,效率很低。

(2) 由于每个块中指针占去了一些字节,致使每个磁盘块存储数据的字节数不再是2的整数次幂。然而,许多程序都是以长度为2的整数次幂来读写磁盘块的,所以要读出一个完整的块的数据内容,就需要从两个磁盘块中获得并拼接信息,这样由于复制又引发了额外的开销。

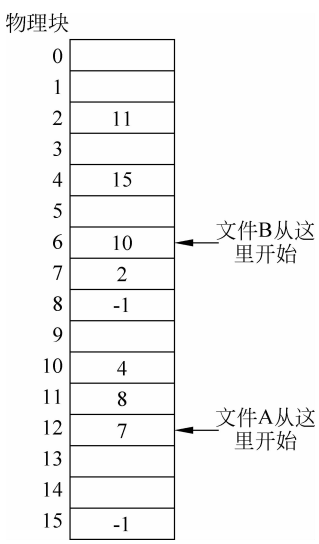

图 5-16 在内存中使用文件分配表的链表分配

3. 在内存中采用表的链表分配

为了克服上述链表分配的两个缺点,可以将指针字从每个磁盘块中取出来,放在内存的一个表中。内存中的这样一个表格称为文件分配表(File Allocation Table,FAT)。表的长度就是磁盘能够划分的物理块数,表用物理块做索引,表中的每一项记录着下一个物理块号,链以一个不属于有效磁盘编号的特殊标记(-1)表示结束。图5-16给出了用图5-15的例子建的表,例子中,文件A依次使用了磁盘块12,7,2,11和8,文件B依次使用了磁盘块6,10,4和15。从图5-16中的表中可以看出,文件A从第12块开始,顺着链依次是7,2,11和8。依此方法也能找到文件B的全部磁盘块。

使用这种方式分配文件,优点是:第一,整个块都可

以存放数据；第二，随机存取容易，虽然存取任意磁盘块仍要顺着链查找其编号，但是整个链表都存放在内存中，因此不需要任何磁盘引用；第三，与前面的方法相同，在目录项中只需要记录一个文件的起始块号，依据起始块号查找 FAT 就可以找到文件的全部磁盘块。

这种方法的主要缺点是在系统工作期间，整个表必须常驻内存。假设磁盘容量为 20GB，每块的大小为 1KB，那么这个磁盘有 2000 万个块，则这个表需要 2000 万项。每项至少需要 3 个字节，为了提高查找速度，有时需要 4 个字节，则这张表就要占用 60MB 或 80MB 的内存空间。如果系统采用的是页式内存管理方法，那么该表将要占用大量的虚拟存储器和磁盘空间，而且还会产生额外的调页流量问题。

4. i 节点

为了克服 FAT 占用内存空间的问题，可以给每个文件分配一个称为 i 节点(Index-node)的数据结构，其中列出了文件属性和文件块的磁盘地址。所以，通过查找文件的 i 节点，就可以找到文件的所有磁盘块。图 5-17 给出了一个简单的 i 节点的示例。

相对于文件分配表而言，采用 i 节点分配方法所占据的内存空间小，这是因为只有在对应文件打开时，其 i 节点才在内存中。如前所述，采用文件分配表方法，如果磁盘有 m 块，则该表需要 m 个表项。若磁盘容量变大，则该表也随之线性增长。即文件分配表的大小正比于磁盘自身的大小。但是，i 节点机制需要在内存中的数组的大小正比于可能要同时打开的最大文件个数，它与磁盘的大小无关。如果每个 i 节点占用 n 个字节，最多有 k 个文件同时打开，那么为了打开文件而保留 i 节点的数据所占据的内存空间仅仅是 $k \times n$ 个字节，它远远小于文件分配表所占用的内存空间。

在使用 i 节点方法时需要考虑的一个问题是，如果每个 i 节点只能存储固定数目的磁盘块地址，那么当一个文件所占用的磁盘块数超出了 i 节点所能容纳的数目怎么办？一个解决办法是让最后一个“磁盘地址”不指向数据块，而是指向一个包含磁盘块地址的块的地址，如图 5-17 所示。进一步扩展，可以有两个或更多个包含磁盘地址的块，或者指向其他存放地址的磁盘块的磁盘块。

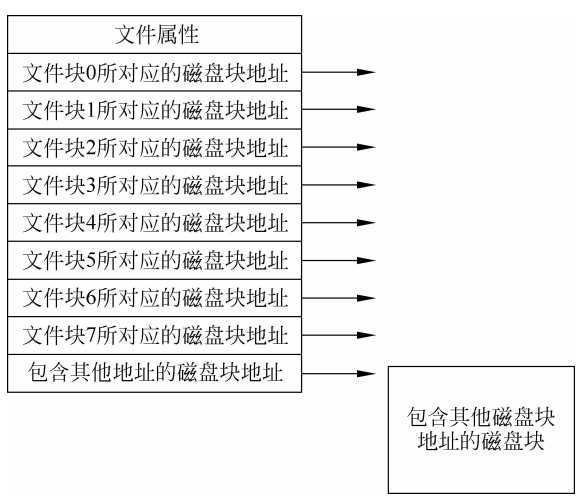

图 5-17 i 节点的例子

5.4.3 磁盘空间管理

当文件新被创建时，要为它分配相应的磁盘空间；当文件被删除时，要回收其所占用的磁盘块。那么，怎样跟踪记录哪些块是空闲的，这是文件系统要考虑的问题。将该问题称为磁盘的空间管理。跟踪记录空闲块的方法主要有空闲表法、位图法和磁盘块链表法等。

1. 空闲表法

为了说明问题的方便，把磁盘中一个连续的未分配区域称为“空闲区”。为了记录哪些磁盘块是空闲的，系统为所有空闲区建立一张空闲表，每一个空闲区对应表中的一个表目。表目的内容包括：空闲区第一个盘块号、该空闲区的盘块总数等信息。将所有空闲区按第一个盘块号的大小排列，如图 5-18 所示，第一个空闲区包含 7 个空闲块，物理块号分别是 5，6，7，8，9，10，11。

序号	首块号	空闲块数
1	5	7
2	18	8
3	54	15
…	…	…

图 5-18 空闲表法

空闲区的分配与内存的动态分配类似，同样可采用首次适应算法、下次适应算法、最佳适应算法和最坏适应算法。当用户提出请求分配 n 个磁盘块空间时，系统按照相应的算法扫描该空闲表的各表目中的空闲块总数，直到找到一个能满足要求的空闲区为止。如果请求的盘块数正好等于表目中的空闲块总数，则把该表目从空闲表中删除；如果表目中的总数多于请求的盘块数，则把分配后剩余的块号留在原表目中，并修改相应的数值。

当用户删除一个文件时，系统回收该文件所占用的磁盘块，且把相应的空闲块信息填回空闲表中。如果释放的磁盘块与原有的空闲区相邻，则把它们合并成一个大的空闲区，并修改相应的表目内容。

这种将若干连续的空闲块组合在一起形成一个空闲表项的方法，比较适合于文件的连续分配。但是，如果存储空间中有大量的小空闲区时，空闲表会变得很大，使检索效率降低。

2. 位图法

位图法是使用较多的一种磁盘块管理方法，其基本思想是利用一个二进制位来表示一个磁盘块的使用情况。如图 5-19 所示，当某位的值为 0 时，表示相应的磁盘块为空闲；当为 1 时，表示已经分配(或者反之)。位图大小由磁盘块总数确定，对于 n 个块的磁盘来说，需要 n 位的位图。例如，对于 32GB 的磁盘，块大小为 2KB，则有 2^{24} 个块，因此位图需要 2^{24} 个位来表示，即需要 1024 个块。

当需要分配一个磁盘块时，就从位图中找到一个值为 0 的二进制位，将该位所对应的磁盘块分给某文件，同时将该位置为 1；当回收一个磁盘块时，将该磁盘块所对应的位置为 0，

表示该块空闲。

1000110000110000
0011110100001110
0111010110010001
1100111001111110
1111000000110101
0010110000000111
1101100111101101
0101000011101111
…
1101001011101111
0000111000010110

图 5-19　磁盘空间的位图法

由于位图所占空间小，所以可以复制到内存中，从而可高速地实现文件的分配和回收。但值得注意的是，当关机或文件信息转储时，位图信息需要完整地在保留在磁盘上。

3. 磁盘块链表法

磁盘块链表法的思想是将一组空闲的磁盘块号记录在一个磁盘块中（每个磁盘块中包含尽可能多的空闲磁盘块号），该磁盘块中有一个指针指向下一个记录空闲磁盘块号的磁盘块，然后将这些磁盘块链接在一起，如图 5-20 所示。假设磁盘块的大小为 1KB，每个磁盘块号用 32 位表示，则每个磁盘块理论上可以记录 256 个空闲磁盘块号，但需要有一个位置存放指向下一个块的指针，所以实际上每个磁盘块中只记录 255 个空闲磁盘块号。则对于 16GB 的磁盘来说，最多需要 65 794 个块的链表来存放全部 2^{24} 个磁盘块号。

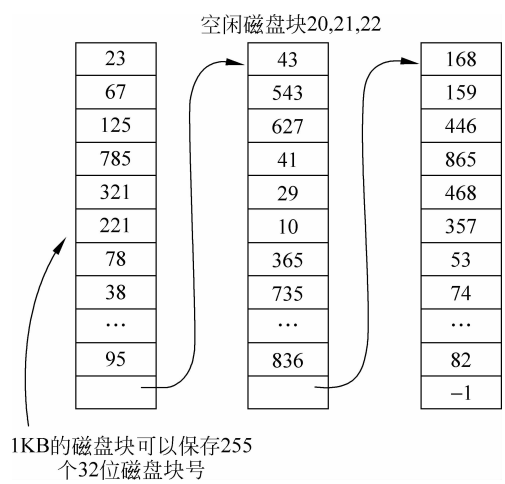

图 5-20　空闲磁盘块链表示意图

与位图法相比，空闲磁盘块链表法只有在磁盘快满时（几乎没有空闲块时）所占的磁盘块才少于位图法所占的磁盘块；否则，由于位图法中每块只用一位二进制标识，而空闲磁盘块链表法中每块用 32 位标识，所以，位图法所需的空间较少。

5.5 文件的共享与保护

现代操作系统大多数支持多用户，在多用户系统中，文件共享是文件系统提供的重要功能之一。文件共享是指一个文件可以被不同的用户或进程使用。例如，层次结构中的某一用户，希望共享其他用户所拥有的文件；计算机网络中，连入网络中的计算机共享网络上的各种文件等。因此，文件共享可以节省大量的存储空间，减少输入/输出操作，进程间也可以通过文件交换信息，为用户的合作提供便利条件。

但是，文件共享并不意味着用户可以不加限制地随意使用文件，出于安全性的考虑，文件是需要保护和加以控制的。文件保护是指防止数据丢失和被无权使用的用户窃取；文件的存取控制是指当多个用户共享同一个文件时，规定“谁能访问该文件以及可以进行哪些访问”，是对文件共享的限制。

5.5.1 文件共享

文件共享可以通过将共享文件复制在多个需要共享的子目录中，在每个子目录中都用共享文件副本的方法来实现，但是很显然这种方法会浪费磁盘空间。因此，可以将共享文件链接到多个用户的目录中，使共享文件在磁盘中只保留一个副本，只在有共享需要的目录中对这个共享文件建立一个链接指针，通过指针可以实现对同一个文件的访问，这样比较经济有效。如图 5-21 所示，C 目录下的一个文件现在也出现在 B 的目录下，B 称为该共享文件的一个链接（Link）。但是，这种链接方式实现文件共享时，文件系统已不是一棵树，而形成了一个有向无环图（Directed Acyclic Graph，DAG）。这种变化无疑增加了文件系统管理的复杂程度，为了实现通过不同的路径查找到同一个文件，需要在共享的节点上知道文件在磁盘中的物理位置，而文件在磁盘中存放地址的描述方式决定了共享处理方式的差异。下面描述几种共享文件的管理方式。

1. 基于文件磁盘地址的共享方式

这种方法是利用文件在磁盘中的地址实现共享管理的，即在目录中包含文件的物理地址，当链接文件时，将文件的物理地址复制到需要共享的目录中。在图 5-21 中，C 目录中的一个文件被共享给 B 目录，这样就需要将该共享文件的物理地址复制到 B 目录中去，当 B 目录需要访问该文件时就可以从自己的目录中找到文件的地址信息从而方便地访问到该文件。但是这种共享管理方式也存在一定的缺点，如果 B 或 C 随后又往该文件中添加内容，则新的数据块将只会出现在进行添加操作的用户目录中，其他的用户对此改变是不可见的，即新增加的这部分内容是不能被共享的，这显然不符合共享的原则。

2. 基于索引节点的共享方式

为了解决上述问题，可以使用索引节点方式。在这种方式中，为每个文件建立一个索引

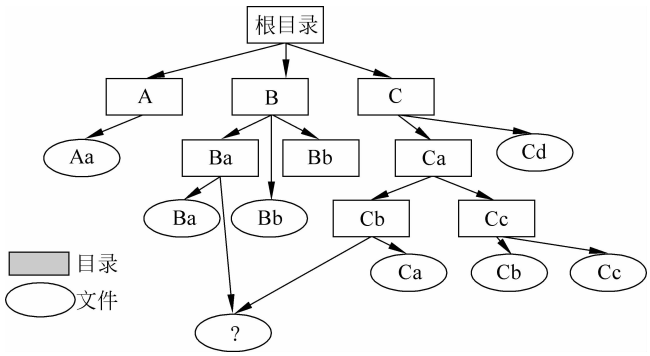

图 5-21 包含共享文件的文件系统

节点 i_node，将文件属性及文件的物理地址等信息放在索引节点中，而不再放在目录项中，同时在每个文件的目录项中建立一个指向该索引节点的指针。这样，文件目录中的每个目录项仅由文件名及指向相应索引节点的指针构成，如图 5-22 所示。此时，任何用户对文件的修改都会反映在索引节点中，其他用户可以通过索引节点存取文件，因此文件的任何变化对于所有共享它的用户都是可见的。

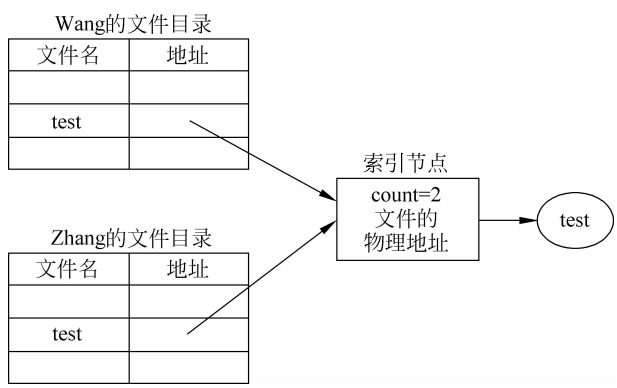

图 5-22 基于索引节点的共享方式

在索引节点中设有一个链接计数字段 count，用于表示链接到本索引节点的目录项的数目。当 count＝2 时，表示有两个目录项链接到本文件上。例如，在图 5-23 中，当用户 C 创建一个新文件时，他是该文件的所有者，此时 count 值为 1。当用户 B 希望共享此文件时，应在用户 B 的目录中增加一个目录项，并设置指针指向该文件的索引节点，此时文件的所有者仍然是 C，但索引节点的链接计数应加 1，即此时 count 值为 2。如果以后用户 C 不再需要该文件而将其删除，此时系统只是删除用户 C 的目录项而保留文件的索引节点（若删除该文件的索引节点，将使 B 的指针悬空），并将 count 减 1，即此时 count 值为 1。此时，只有 B 拥有指向该文件的目录项，而该文件的所有者仍然是 C，如果系统要对该文件记账收费，则 C 将继续为该文件付账直到 B 将它删除。如果 B 不再需要它，该文件才被删除，此时 count 值为 0。

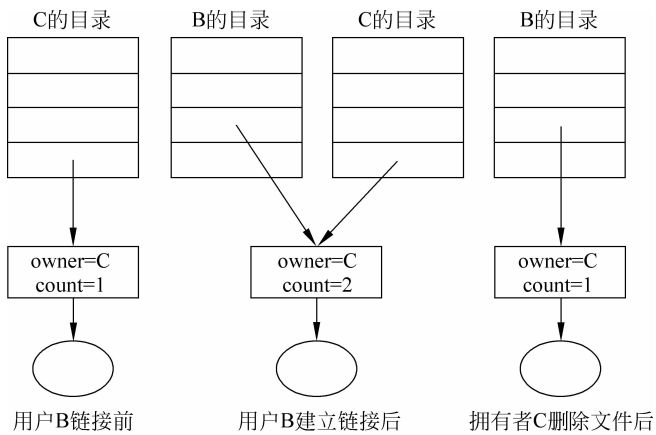

图 5-23 进程 B 链接前后的情况

3. 基于符号链接法的文件共享

利用符号链接也可以实现文件共享。符号链接法是指当需要共享文件时，就在本目录中建立一个新的文件，这个文件是 link 类型的链接文件，该文件中只包含一个指向共享文件的指针。例如，用户 B 为了共享用户 C 的一个文件 M，这时可以由系统创建一个 link 类型的新文件，并把新文件添加到 B 的目录中，以实现 B 的一个目录与文件 M 的链接。新文件中只包含被链接文件 M 的路径名，称这种链接方式为**符号链接**(Symbolic Linking)。当用户 B 要访问被链接的文件 M 时，操作系统发现要读的文件 M 是 link 类型的文件，则由操作系统根据 link 文件中的路径名去读该文件，从而实现了用户 B 对文件 M 的共享。

在利用符号链接实现文件共享时，因为只有文件所有者拥有指向其索引节点的指针，共享该文件的用户只有其路径名，而没有指向索引节点的指针，因此，当文件所有者删除文件后，其他用户如果试图通过符号链接访问该文件，则会因系统找不到该文件而导致访问失败，于是系统会将该符号链接删除掉。

符号链接方式实现共享的优点是只要提供一个机器的地址以及文件在该机器上的驻留路径，就可以链接全球任何地方的机器上的文件。其不足是需要较大的开销。当其他用户去读共享文件时，系统是根据给定的文件路径名逐个分量进行查找的，直到找到索引节点，这些操作需要多次访问磁盘，这使共享文件的访问开销很大。另外，符号链接需要配置索引节点以及一个磁盘块用于存储路径，这也要消耗一些磁盘空间。

5.5.2 文件的访问保护

若要防止系统中的文件被他人破坏、窃取，以及在共享过程中对文件进行未授权的操作，就必须对文件采取有效的保护措施。文件有多种保护方法，不同系统采用了不同的方法。

1. 口令保护

口令保护方式大致有两种。一种是当用户登录系统时，系统登录程序要求用户输入登

录系统的用户名和密码。如果用户输入的口令与原来设置的口令不一致的话，该用户将被拒绝登录系统。另一种口令保护方式是，用户在创建文件时，为每一个创建的文件设置一个口令，且将其置于文件说明中。当任何一个用户想使用该文件时，都必须首先提供口令。只有口令正确了才允许访问。当然，口令只有设置者自己知道，如果他想允许其他用户使用自己的文件，则可以将其口令告诉给其他用户。这样，既可以做到文件共享，又可做到保密。而且，由于口令较为简单，占用的内存单元以及验证口令所费的时间都较少。

但是，口令保护方式的显著缺点是保密性较差。对于前一种方式，如果登录系统的口令被窃取或被破解，那么任何人都可以轻松登录系统。对于后一种方式，口令一旦被别人获得，就可以获得同文件所有者同样的权利而没有任何等级差别，这就使得文件失窃的可能性大大增加。

因此，为了防止无权用户使用系统和文件中的口令被窃取，系统一方面要求用户要经常修改口令，另一方面应限制无权用户在一个终端上登录的次数。

2. 加密保护

为了防止文件泄密，可以对一些关键文件进行加密。加密保护方式在用户创建源文件并将其写入存储设备时对文件进行编码加密，在读出文件时对其进行译码解密。只有能够进行译码解密的用户才能读出被加密的文件信息。所以即使文件被别人窃取，但由于不能解密文件而无法获得文件的信息内容，从而起到保护文件的作用。

目前，加密方式有很多，但基本思想都类似，即用户加密一个文件时，提供一个代码键，加密程序根据这个代码键对文件进行变换，从而得到其相应的加密文件；在存取和执行文件时，解密程序再使用对应的代码键对文件进行解密，还原成源文件后，方可使用。

加密方式相对于口令保护方式具有保密性强的优点，主要是因为代码键没有存放在系统中，而是由用户自己掌握的。但是，由于编码解码工作要耗费大量的处理时间，因此，加密技术是以牺牲系统开销为代价的。

3. 访问控制

“口令”和“密码”保护方式只是防止文件被他人存取和窃取，并没有实现用户对文件的访问权限的控制。若想实现文件对不同用户进行不同的保护，则要对文件实施访问权限的控制。

1）存取控制矩阵

存取控制矩阵是一个二维矩阵，其中一维列出使用该文件系统的全部用户；另一维列出存入系统中的全部文件。设有 m 个用户，n 个文件，则矩阵为 $m \times n$，其矩阵元素 B_{ij} = right，其中 $i \in \{1,2,\cdots,m\}$，$j \in \{1,2,\cdots,n\}$，right = {R，W，E，A，M，D} 的合法子集。

right 表示权限（R 表示可读，W 可写，E 可执行，A 可添加，M 可修改，D 可删除）。

访问控制矩阵有时又称为用户权限表，矩阵中的每个元素用来表示某个用户对某个文件的存取权限，存取权限可以为读、写、执行、添加、修改、删除以及它们的任意组合。表 5-1 给出了一个存取控制矩阵的例子，例如表中用户 Zhao 对文件 A 可以进行读、写和执行操作。

表 5-1 存取控制矩阵

用户 / 文件	用户 Zhao	用户 Zhang	用户 Wang	…
文件 A	RWE	E	RWE	
文件 B	RW	R	RWE	
文件 C	R	W	WE	
文件 D	R	W	RW	
文件 E	W	RW	RWE	
⋮	⋮	⋮	⋮	

当用户的进程请求访问文件时，操作系统便根据该矩阵内容对本次存取要求进行比较，如果匹配成功则允许访问；否则，系统拒绝此次用户对文件的访问。存取控制矩阵的方法虽然在概念上比较简单，但是，当文件和用户较多时，存取控制矩阵将变得非常庞大，这无论是在占用内存空间的大小上，还是在为使用文件而对矩阵进行扫描的时间开销上都是不合适的。例如，如果系统有 200 个用户，他们共有 50 000 个文件，那么这个存取控制矩阵就有 200×50 000＝10 000 000 个元素，它们将占据大量的存储空间，而且查找某个元素也很耗时。因此，这种方法只有在较小规模的系统上才具有使用价值。

2）存取控制表

存取控制表以文件为单位，把用户按某种关系划分为若干组，同时规定每组用户对该文件的存取权限。这样，所有用户组对文件权限的集合就形成了该文件的存取控制表，如表 5-2 所示。

表 5-2 文件存取控制表

文件 / 用户组	文件 A
用户组 A	E
用户组 B	RE
文件拥有者	RWE
其他	none
⋮	⋮

从表 5-2 可以看出，文件拥有者可以对文件 A 进行读、写、执行操作，用户组 A 只能执行文件 A，用户组 B 既能读也能执行文件 A。

每个文件都有一张存取控制表。在实现时，该表存放在文件说明中，文件被打开时，由于存取控制表也相应地被复制到了内存活动文件中，因此，存取控制验证能高效进行。UNIX 操作系统就是用该方法实现文件访问控制的。

5.6 文件系统的可靠性

计算机在使用过程中难免会出现故障。如果计算机由于物理原因（火灾、短路等）而造成 CPU、内存、终端等部件受到损坏的话，可以花钱更换部件或再买一台机器取代它继续工作。但是，如果计算机的文件系统被破坏了则是一件很头疼的事，即使有时能够利用数据恢复工具恢复部分信息，但也是很困难的。因此，文件系统必须采取有效的保护措施，预防这种事件的发生，这就是文件系统的可靠性问题。

为了提高文件系统的可靠性，通常采用以下一些常用的方法。

5.6.1 文件备份

人们越来越认识到保护数据免受损坏的重要性，所以，越来越多的企业、单位和个人都对文件进行了备份（即为一个文件复制多个副本），以便一旦发生故障时可以从中恢复数据。但是，为文件做备份既浪费时间又需要大量的存储空间，所以应当尽量使备份工作高效简便。因此，在备份时需要考虑以下几个问题：

（1）备份的范围，是备份整个文件系统还是只备份其中的一部分？实际上，系统中很多可执行程序（二进制代码）保存在文件系统树的有限部分，这些文件能直接从销售商提供的CD-ROM盘上得到，重新安装一遍就行了，所以没必要备份这部分文件。此外，在UNIX系统中，所有的特殊文件（即I/O设备）都放置在/dev目录下，对这个目录做备份不仅没有必要而且还十分危险，因为一旦进行备份的程序试图读取其中的文件，备份程序就会永久挂起。因此，合理的做法是只备份特定目录及其下的全部文件，而不是备份整个文件系统。

（2）对从最近一次备份以来没有修改过的文件做备份是一种浪费，因而产生了**增量转储**的思想。**增量转储**就是周期性地（每月一次或每周一次）做全面的转储，而每天只对当天修改的数据做备份，或只备份自最近一次转储以来更改过的文件。这种做法大大地缩短了转储时间，但是在做数据恢复时操作比较复杂。

（3）由于转储的往往是海量数据，所以在转储之前可以对文件进行压缩。但是，对于许多压缩算法，备份介质上的单个坏点就能破坏压缩算法，导致整个文件甚至整个备份介质都无法阅读。所以，是否对备份文件进行压缩需要慎重考虑。

（4）对活动的文件系统做备份是很难的。因为在转储过程中添加、修改或删除文件和目录可能会导致文件系统的不一致性。因此，在晚上让文件系统脱机转储是较好的方法。

在考虑以上问题之后，重点看一看备份介质、备份策略和转储方式的选择。

1. 备份介质

目前，比较常用的备份介质有磁带、光盘和硬盘等。磁带适合大量数据备份，允许用户在无人干涉的情况下进行备份与管理；但是磁带机的缺点是速度慢，只支持顺序存取，不支持随机存取。光盘携带方便，可靠性好，不过这些介质的写入次数有限。硬盘适合各种数据类型的备份，支持随机存取，不过其价格较贵。

2. 备份策略

通常有以下三种备份策略：

1）完全备份

完全备份是指在某个时间点上对整个系统做一次完全拷贝，实际应用中就是用一盘磁带对整个系统进行完全备份，包括其中的系统和所有数据。这样在备份期间如果出现数据破坏或丢失，就可以使用上一次的备份数据将系统恢复到上一次备份时的状态。

完全备份是最基本的系统备份方式。这种备份策略的好处是：当发生数据丢失或破坏性的灾难时，只要用一盘磁带（即灾难发生前所备份的磁带），就可以恢复丢失的数据。因此大大加快了系统或数据的恢复时间。然而它也有不足之处。首先，由于每次都对整个系统进行完全备份，造成备份的数据大量重复。这些重复的数据占用了大量的磁带空间，这无疑

增加了成本。其次，由于需要备份的数据量较大，因此备份所需的时间也就较长。对于那些业务繁忙、备份时间有限的单位来说，选择这种备份策略是不明智的。

2）增量备份

在这种备份策略中，只备份上一次备份后发生变化的数据。通常，首先进行一次完全备份，然后每隔一段时间备份一次在这段时间间隔内修改过的数据。当经过一段较长的时间间隔后，再进行一次完全备份。依照这样的周期反复执行。

例如，可将一周作为备份周期，星期日进行一次完全备份，然后在接下来的 6 天里只对当天新的或被修改过的数据进行备份。这种备份策略的优点是节省了磁带空间，缩短了备份时间。但它的缺点在于，当灾难发生时，数据的恢复比较麻烦。例如，系统在星期三的早晨发生故障，丢失了大量的数据，那么现在就要将系统恢复到星期二晚上时的状态。这时系统管理员就要首先找出星期日的那盘完全备份磁带进行系统恢复，然后再找出星期一的磁带来恢复星期一的数据，然后找出星期二的磁带来恢复星期二的数据。很明显，这种方式很烦琐。另外，这种备份的可靠性也很差。在这种备份方式下，各盘磁带间的关系就像链子一样，一环套一环，其中任何一盘磁带出了问题都会导致整条链子脱节。在上例中，如果星期二的磁带出了故障，那么管理员最多只能将系统恢复到星期一晚上时的状态。

3）差分备份

差分备份是针对完全备份而言的：备份上一次的完全备份后发生变化的所有文件。这种备份方法与增量备份相似。首先每隔一段时间进行一次完全备份，然后每天对修改过的数据进行备份。但是两者不同的是：增量备份只备份当天更改的数据，而差分备份则备份从上次进行完全备份后至今更改的全部数据文件。因此，一旦发生数据丢失，就可以首先恢复前一个完全备份，再使用前一个差分备份即可恢复到前一天的状态。

显然，每次做差分备份的工作量要比增量备份大。但其好处在于，增量备份每天都保存当天的备份数据，需要较多的备份文件；而差分备份只需要保存一个完全备份和一个差分备份就可以了，备份文件少。同样，在进行数据恢复时，增量备份需要按顺序进行多次备份的恢复，而差分备份只需要恢复两次。因此，差分备份的恢复工作相对较为简单。

在实际应用中，备份策略通常是以上三种的结合。例如每周一至周六进行一次增量备份或差分备份，每周日进行完全备份，每月底进行一次完全备份，每年底进行一次完全备份。

3. 转储方式

将磁盘上的数据转储到磁带上有物理转储和逻辑转储两种方式。

物理转储是从磁盘的第 0 块开始，将全部的磁盘块按序写到磁带上，直到将最后一块复制完，转储结束。这种方式实现起来简单，可以确保万无一失，而且速度较快，基本上可以以磁盘的速度运行；缺点是无法跳过指定的目录，无法执行增量转储，也无法根据需要恢复单个文件。

逻辑转储是从一个或多个指定的目录开始，递归地转储自某个日期以来更改过的全部文件和目录。在逻辑转储方式中，转储磁带上会有一系列精心标识的目录和文件，这样可以方便满足恢复特定文件或目录的请求。

5.6.2 文件系统的一致性

文件系统的一致性是影响文件系统可靠性的另一个问题。当访问文件时，文件系统首先将磁盘块内容读到内存中，在内存修改它们之后再写回磁盘。如果在将修改过的内容全部写回磁盘之前系统崩溃了，那么，文件系统则处于不一致状态。如果一些未被写回的是i节点块、目录块或者是包含空闲表的块时，这个问题尤显严重。

为了解决文件系统的不一致性问题，多数计算机系统都有一个用来检查文件系统一致性的实用程序。例如，Windows系统的scandisk，UNIX系统的fsck，每当系统启动时，特别是崩溃之后，都要运行它。

UNIX系统中的一致性检查分为：**磁盘块的一致性检查**和**文件的一致性检查**。

1. 磁盘块的一致性检查

在检查磁盘块的一致性时，检查程序建立两张表——**使用表**和**空闲表**。每个磁盘块在两个表中各对应一项。各表项相当于一个计数器，初值都置成0。使用表记录各个磁盘块在文件中出现的次数，而空闲表则记录各个磁盘块在空闲块链表(或空闲块位图)中出现的次数。

接着，检查程序读取全部的i节点，从而明确相应文件所使用的磁盘块号。每当读到一个磁盘块号，便将使用表中的对应项加1。然后，该程序检查空闲块链表或位图，找出全部未使用的磁盘块；每找到一个，便将空闲表中的对应项加1。

经过检查之后，如果文件系统是一致的，则每一个块要么在使用表中的值为1(表示该块在文件中使用)，要么在空闲表中的值为1(表示该块处于空闲)，即每个盘块在两个表中对应项值之和为1，如图5-24(a)所示。

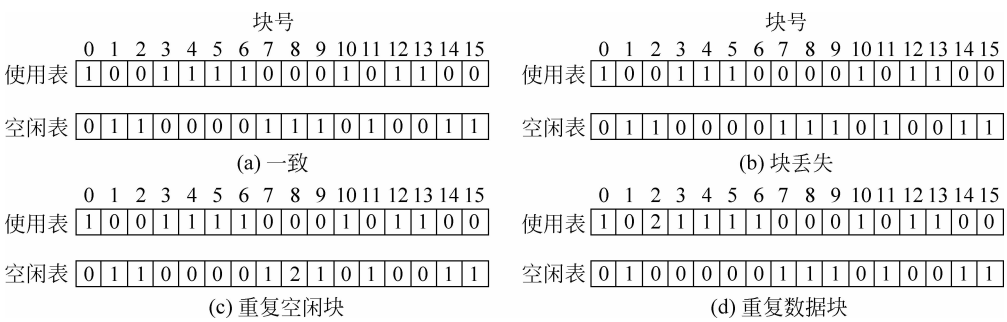

图5-24 文件系统状态

如果文件系统崩溃，会造成磁盘块丢失现象，这两张表可能如图5-24(b)所示，其中磁盘块6在两个表中都未出现。尽管块丢失不会造成实际的损害，但是它浪费了磁盘空间，减少了磁盘容量。其实，块丢失的解决办法很简单：把丢失的磁盘块添加到空闲链表(或空闲块位图)中即可。

有可能出现的另外一种情况如图5-24(c)所示。其中磁盘块8在空闲表中出现两次(在空闲块链表中可能会发生这种情况，而在位图中不会发生此类情况)。其解决办法是：重新建立空闲块链表。

最糟糕的情况是同一磁盘块在两个或多个文件中出现，如图 5-24(d)所示，磁盘块 2 在使用表中的计数值为 2。如果删除其中一个文件，则该磁盘块就同时出现在两个表中，即处于使用和空闲两种状态。如果删除这两个文件，则它在空闲表中的计数就是 2。检查程序对此种情况可以采取的办法是让系统分配一个空闲块，把磁盘块 2 的内容复制到该空闲块中，并且把它插入其中一个文件中。这样，文件的内容虽未改变，但至少保证了文件系统的一致性。然后检查程序应该报告出错，由用户检查错误的原因。

2. 文件的一致性检查

除了检查每个块的计数正确与否之外，文件系统检查程序还要检查目录系统。此时也要用到计数器表，不过这时不是一个块而是一个文件对应一个计数器。程序从根目录开始，沿目录树递归向下查找，检查文件系统中的每个目录。对于每个目录中的每个文件，其 i 节点对应的计数器值加 1。需要注意的是，由于存在硬链接，一个文件可能出现在两个或多个目录中。而符号链接是不计数的，不会引起对目标文件的计数器加 1。

当全部检查完毕后，得到一张以 i 节点号为下标的列表，说明每个文件包含在多少个目录中。然后，把这些数目与存放在 i 节点中的链接计数进行比较。(当文件被创建时，这些计数器从 1 开始，随着每次对文件的一个硬链接的产生，对应计数器加 1。)如果文件系统是一致的，则这两个值应该相等；否则，出现两种错误，即 i 节点中的链接计数太大或太小。

如果 i 节点中的链接计数大于目录项个数，即使相关目录中的所有文件都被删除，该计数也不会等于 0，从而无法释放 i 节点。这个错误并不太严重，只是因为存在不属于任何目录的文件而浪费了磁盘空间。解决办法是将 i 节点的链接计数设成正确值。如果正确值为 0，则应删除该文件。

如果该计数太小，例如同一个文件链接到两个目录项，但其 i 节点链接计数只为 1，只要其中有一个目录项删除，对应 i 节点链接计数就变为 0。此时，文件系统标记该 i 节点为“不可用”，并释放其全部磁盘块。结果导致还有一个目录项指向不可用的 i 节点，它的磁盘块可能分给另外的文件，这就造成了严重的后果。解决办法是，把 i 节点链接计数设为实际的目录项个数。

出于效率上的考虑，可把上述磁盘块的一致性检查和文件的一致性检查这两种操作结合起来进行。当然，文件系统还可能出现其他不一致现象，例如文件放在普通用户目录中却被特权用户拥有，不允许文件主访问而其他用户却可以读写此文件等，对此要具体情况具体分析、处理。

5.7 小结

(1) 文件及命名。

文件(File)是具有符号名的相关信息的集合。文件名字可以是一些字符和数字的组合，由文件名和扩展名组成。

(2) 文件结构。

① 字节序列文件：也叫流式文件，是指文件内部不再划分记录，而是由一组相关信息组成的有序字符流，其长度就是它所包含的字节个数。

② 记录式文件：它在逻辑上可被看成一组连续记录的集合，即文件是由若干相关记录组成的，并且按照记录出现的逻辑顺序进行编号。

③ 树状文件：由一棵记录树组成，每个记录的长度可以不同。在记录的固定位置上有一个关键字字段，这棵树按照该字段进行排序，从而可以对特定关键字进行快速查找。

(3) 文件类型。

按用途分类：系统文件、用户文件、库文件。

按存取权限分类：只读文件、可读/写文件、可执行文件。

按保存时间分类：临时文件、永久文件。

按文件中的数据形式分类：源文件、目标文件、可执行文件。

按文件的内部构造和处理方式分类：普通文件、目录文件、特别文件。

(4) 文件存取。

① 顺序存取：是指对文件的存取严格按照字节出现的先后次序或记录排列顺序依次存取的一种方法，不允许跳过一些内容不按顺序存取。

② 随机存取：也叫直接存取，指允许以任意次序读取文件中的字节或记录，而不管上一次存取的是哪个字节或记录。

(5) 文件操作。

不同的文件系统提供了对文件操作的各种命令，与文件有关的常用的系统调用有：创建(create)文件，删除(delete)文件，打开(open)文件，关闭(close)文件，读(read)文件，写(write)文件，追加(append)文件，随机存取(seek)文件，获得文件属性(get attribute)，设置文件属性(set attribute)，重命名(rename)文件等。

(6) 文件系统。

所谓文件系统就是操作系统中提供管理文件的软件机构，它负责操纵和管理文件，实现文件的保护和共享，方便用户“按名存取”。

文件系统的主要功能有：文件管理、目录管理、文件的共享和保护、文件存储空间的管理、提供方便的接口。

(7) 目录结构。

单级目录、二级目录、多级目录。

(8) 路径名。

① 绝对路径名：是指从根目录出发最终到达文件的整个完整路径的名字。

② 相对路径名：是指从当前目录出发最终到达文件所经过的路径的名字。

(9) 目录操作。

创建目录(create)、删除目录(delete)、打开目录(opendir)、关闭目录(closedir)、读目录(readdir)、重新命名目录(rename)、链接文件(link)、解除链接文件(unlink)。

(10) 文件的分配。

文件的分配是指系统怎样为文件分配存储空间，常用的分配方法有：

① 连续分配。给一个文件分配一组连续的磁盘块。

② 链表分配。为每个文件构造一个磁盘块链表，每个块的第一个字作为指向下一个块的指针，块的其余部分用来存放数据。文件的最后一个磁盘块的指针通常为0，以指示该块是链尾。

③ 在内存中采用表的链表分配。将指针字从每个磁盘块中取出来，放在内存的一个表中。内存中的这样一个表格称为文件分配表(File Allocation Table，FAT)。表的长度就是磁盘能够划分的物理块数，表用物理块做索引，表中的每一项记录着下一个物理块号，链以一个不属于有效磁盘编号的特殊标记(－1)表示结束。

④ i 节点。给每个文件分配一个称为 i 节点的数据结构，其中列出了文件属性和文件块的磁盘地址。

(11) 磁盘空间管理。

磁盘空间管理解决的是系统怎样记录空闲块。常用的方法有：

① 空闲区表：为了记录哪些磁盘块是空闲的，系统为所有空闲区建立一张空闲表，每一个空闲区对应表中的一个表目。表目的内容包括：空闲区第一个盘块号、该空闲区的盘块总数等信息。

② 位图法：利用一个二进制位来表示一个磁盘块的使用情况。当某位的值为“0”时，表示相应的磁盘块为空闲；当为“1”时，表示已经分配(或者反之)。

③ 磁盘块链表法：磁盘块链表法的思想是将一组空闲的磁盘块号记录在一个磁盘块中，该磁盘块中有一个指针指向下一个记录空闲磁盘块号的磁盘块，然后将这些磁盘块链接在一起。

(12) 文件的共享与保护。

文件共享是指一个文件可以被不同的用户或进程使用。包括基于索引节点的文件共享和符号链接法文件共享两种方式。

文件的保护是指防止系统中的文件被他人破坏、窃取，以及在共享过程中对文件进行未授权的操作所采取的有效措施。保护方法包括口令保护、密码保护、访问控制等。

(13) 文件备份的策略包括：

① 完全备份。

完全备份也称简单备份，即每隔一定时间就对系统做一次全面备份，这样在备份间隔期间出现数据丢失或破坏，可以使用上一次的备份数据将系统恢复到上一次备份时的状态。

② 增量备份。

在这种备份策略中，首先进行一次完全备份，然后每隔一个较短的时间段进行一次备份，但仅仅备份在这段时间间隔内修改过的数据。当经过一段较长的时间后，再重新进行一次完全备份。依照这样的周期反复执行。

③ 差分备份。

这种备份方法与增量备份相似。首先每隔一段时间进行一次完全备份，然后每天进行一次更新数据的备份。不同的是：增量备份是备份当天更改的数据，而差分备份是备份从上次进行完全备份后至今更改的全部数据文件。一旦发生数据丢失，首先可以恢复前一个完全备份，然后再使用前一个差分备份恢复到前一天的状态。

(14) 将磁盘上的数据转储到磁带上有物理转储和逻辑转储两种方式。

物理转储是从磁盘上第 0 块开始，把所有的盘块按照顺序写到磁带上；当复制完最后一块时，转储结束。这种方式实现起来简单，主要缺点是无法跳过给定的目录，以便执行增量转储，也无法根据需要恢复单个文件。

逻辑转储是从一个或多个指定的目录开始，递归地转储自某个日期以来被修改的所有

文件和目录。利用逻辑转储方式可在转储带上得到一系列精心标识的目录和文件，从而根据需要，很容易恢复一个特定文件或目录。

课后习题

1. 单项选择题

(1) 文件系统的主要目的是(　　)。

A. 实现对文件的按名存取　　B. 实现虚拟存储

C. 提高外存的读写速度　　D. 用于存储系统文件

(2) 为了解决不同用户文件的“命名冲突”问题，通常在文件系统中采用(　　)。

A. 多级目录　　B. 约定的方法　　C. 索引　　D. 路径

(3) 使用绝对路径名访问文件是指从(　　)开始按目录结构访问某个文件。

A. 当前目录　　B. 用户主目录　　C. 根目录　　D. 父目录

(4) 由字符序列组成，文件内的信息不再划分结构，这是指(　　)。

A. 流式文件　　B. 记录式文件　　C. 顺序文件　　D. 有序文件

(5) 位图可用于(　　)。

A. 文件目录查找　　B. 文件保护

C. 主存空间共享　　D. 磁盘空间和主存空间的管理

(6) 打开文件的具体含义是(　　)。

A. 在指定的磁盘地址上建立一个文件

B. 撤销指定文件的目录

C. 将指定的文件目录内容复制到主存的活动文件表中

D. 修改指定文件的内容

(7) 以下哪个文件操作能将数据加到文件末尾？(　　)

A. Open　　B. Append　　C. Write　　D. Read

(8) (　　)是从磁盘上第0块开始，把所有的盘块按照顺序写到磁带上；当复制完最后一块时，转储结束。

A. 逻辑转储　　B. 增量转储　　C. 更新转储　　D. 物理转储

(9) 文件系统中，文件访问控制信息存储的合理位置是(　　)。

A. 文件控制块　　B. 文件分配表

C. 用户口令表　　D. 系统注册表

(10) 设置当前工作目录的主要目的是(　　)。

A. 节省外存空间　　B. 节省内存空间

C. 加快文件的检索速度　　D. 加快文件的读/写速度

2. 简答题

(1) 文件逻辑结构有哪几种？它们各自的特点是什么？

(2) 在UNIX等系统中，按文件的内部构造和处理方式，文件分为哪几类？

(3) 对文件可以进行哪些操作?

(4) 使用文件系统时,通常要显式地进行 OPEN 与 CLOSE 操作。试问:

① 这样做的目的是什么?

② 能否取消显式的 OPEN 与 CLOSE 操作?应如何做?

③ 取消显式的 OPEN 与 CLOSE 操作有什么不同?

(5) 文件的顺序存取和随机存取有何不同?

(6) 文件目录和目录文件各起什么作用?目前广泛采用的目录结构形式是哪种?它有什么优点?

(7) 如果当前工作目录是/usr/ast,则相对路径名为../bin/x 的绝对路径名是什么?

(8) 为文件分配存储空间的方法有哪些?它们各自的特点是什么?

(9) 设某文件是链接文件,由 5 个逻辑记录组成,每个逻辑记录大小与磁盘块大小相等,都为 1024 字节,并依次存放在 55、58、76、80、122 号磁盘块上。若要存取文件的第 1659 逻辑字节处的信息,问要访问哪一个磁盘块?

(10) 文件分配表(FAT)如图 5-25 所示,文件 A 和文件 B 各使用了哪些磁盘块?

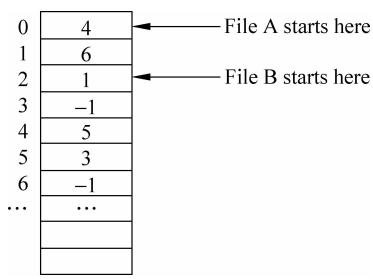

图 5-25 FAT

(11) 磁盘空间的管理方法有哪些?它们各自的特点是什么?

(12) 空闲磁盘空间可用空闲表或位图来跟踪。假设磁盘地址需要 D 位,一个磁盘有 B 个块,其中有 F 个空闲。在什么条件下,空闲表采用的空间少于位图?设 D 为 16 位,请计算空闲磁盘空间的百分比。

(13) 一个空闲空间位图开始时和磁盘分区首次格式化类似,如 1000 0000 0000 0000(首块被根目录使用),系统总是从最小编号的块开始寻找空闲块的,所以在有 6 块的文件 A 写入之后,该位图为 1111 1110 0000 0000。请说明在完成下列每一个附加动作之后位图的状态:

① 写入有 5 块的文件 B;

② 删除文件 A;

③ 写入有 8 块的文件 C;

④ 删除文件 B。

(14) 考虑这样一个文件系统,其中文件可能被删除,并且在指向它的链接仍然存在的情况下可重新使用其磁盘空间。在同一个磁盘空间建立一个新文件,将会出现什么问题?请给出方法来避免该问题。

(15) 文件的共享方法有哪些?

(16) 为了保证文件系统的安全性,可以采取哪些措施?

(17) 备份的策略有哪些?

(18) 文件存储器的管理与内存管理有何异同点?

(19) 设某系统磁盘共有 600 个块,块号为 0～599。如果采用位图法管理这 600 个块的盘空间,当字长为 32 位时,问:

① 位图需要多少个字?

② 第 i 字第 j 位对应的块号是多少?

英文习题

1. Select the correct answer

(1) (　　) operation can add data to the end of the file.

A. Open　　B. Append　　C. Write　　D. Read

(2) A (　　) dump starts at block 0 of the disk, writes all the disk blocks onto the output tape in order, and stops when it has copied the last one.

A. logical　　B. full　　C. incremental　　D. physical

(3) File can be structured in the following ways excepted for (　　).

A . byte sequence　　B. tree　　C. record sequence　　D. graph

(4) A (　　) dump starts at one or more specified directories and recursively dumps all files and directories found there that have changed since some given base date.

A. logical　　B. full　　C. incremental　　D. physical

(5) (　　) operation allows a file to appear in more than one directory.

A. Read　　B. Link　　C. Write　　D. Append

(6) In OS, the part of dealing with files is known as (　　).

A. database system　　B. file system

C. searches system　　D. data storage system

(7) (　　) are system files for maintaining the structure of the file system.

A. Block special files　　B. Character special files

C. Regular files　　D. Directories

(8) The OS provides two ways of file access: random access and (　　) access.

A. streaming　　B. series　　C. sequence　　D. sequential

(9) In the following file operations, the purpose of (　　) operation is to allow the system to fetch the attributes and list of disk addresses into main memory for rapid access or later calls.

A. open　　B. create　　C. write　　D. read

(10) The beginning of the bitmap of a disk is 0001 1110 0000 0011(0 is on behalf of free). With contiguous allocation, the bitmap can't be (　　) after writing file A which uses 4 blocks.

A. 1111 1111 0000 0011　　B. 0001 1110 1111 0011

C. 0001 1110 0011 1111　　D. 0001 1111 1110 0011

(11) A (　　) stores information in fixed-size blocks, each one with its own address.

A. character device　　B. block device

C. device controller　　D. clock

(12) Which of the following is not a file operation? (　　)

A. Opendir　　B. Create　　C. Delete　　D. Rename

2. Answer the following questions

(1) If /usr/ast is the working directory, for the file whose relative path name is ../lib/directory, the absolute path name is _____.

(2) The beginning of a free space bitmap looks like this after the disk partition is first formatted: 1000 0000 0000 0000 (the first block is used by the root directory). The system always searches for free blocks starting at the lowest numbered block. Show the bitmap after each of the following additional actions:

① File A is written, using 6 blocks, the bitmap looks like _____.

② File B is written, using 5 blocks, the bitmap looks like _____.

③ File A is deleted, the bitmap looks like _____.

④ File C is written, using 7 blocks, the bitmap looks like _____.

(3) Using the File Allocation Table (FAT), file A uses disk blocks 2, 7, 4, 6 and 3, in that order. What does the FAT look like? Please fill in the FAT (Figure 5-26). You should also point out the start disk block.

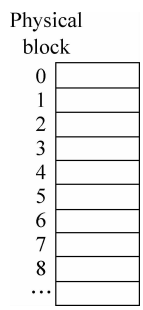

Figure 5-26　FAT

思考题

1. 现代操作系统中为什么要设置文件管理系统?

2. 硬盘的分区信息和引导区信息在什么位置? 结构是怎样的?

3. 一个 320GB 的硬盘, 空闲空间为多少的时候, 用链表管理比位图管理更节省空间?