第5章 文件管理

本章学习目标

文件是存储在外存介质上的有序信息的集合。那么，这些有序信息的结构是怎样的？文件在外部介质上是如何存储的？在访问文件时，操作系统是如何实现文件按名访问的？本章学习关于文件管理的相关知识。通过本章的学习，读者应该掌握以下内容：

- 掌握文件系统的概念及功能；
- 掌握文件的逻辑结构；
- 掌握文件的访问方式；
- 掌握文件的物理结构；
- 掌握文件控制块及目录结构；
- 掌握操作系统对文件的操作；
- 掌握文件存储空间的管理；
- 了解 Linux 系统文件管理功能。

5.1 文件管理概述

5.1.1 文件的概念

文件的概念是在信息的物理存储和信息表示方式的基础上引入的。从用户使用处理的逻辑角度，文件定义为具有符号名而且在逻辑上具有完整意义的信息项的有序序列。另外，操作系统作为一个系统软件本身也需要信息管理功能提供支持。讨论文件时经常用到以下术语。

1. 数据项

数据项是描述一个对象的某种属性的字符集，是数据组织中可以命名的最小逻辑数据单位，即原子数据，又称为数据元素或字段。它的命名往往与其属性一致。例如，用于描述一个教职工的基本数据项有姓名、年龄、工资等。

2. 记录

记录是一组相关数据项的集合，用于描述一个对象在某方面的属性。例如，一个学生记

录有学号、姓名、性别、年龄、班级等。

3. 文件

文件是相关记录的集合，它通常存放在外存上，可以作为一个独立的单位存放和实施相应的操作。例如，用户编写的一个源程序、经编译生成的目标代码程序、系统中的库程序和各种系统程序、一批待加工处理的数据、一篇文章等，都可构成一个文件加以保存。

文件是由创建者定义的、具有文件名的一组相关信息的集合。一个文件包含有文件类型、文件长度、文件的物理位置、文件的创建时间、使用权限等属性。文件必须有文件名，通常由一串 ASCII 码字符或(和)汉字构成，名字的长度因系统不同而异。Linux 系统中，文件名最大长度由 NR-NAME-LEN 控制，默认值为 255 个字符。

注意：在 Linux 中，文件名是区分大小写的，如“a”与“A”代表不同的文件名。

5.1.2 文件系统

文件系统是操作系统中对文件进行管理和操作的软件机构与数据的集合，即文件管理系统。文件系统包含与文件管理相关的软件、被管理的文件和实施文件管理所需要的数据结构。

1. 文件系统需解决的问题

(1) 有效地分配存储器的存储空间。通常，一个文件存储器上的物理空间是以块为单位进行分配的。对于分页系统，块区的大小一般仅与页的大小有关，而与文件中的记录大小无关。

(2) 提供一种组织数据的方法。存储数据的存储器具有固定的物理特性，数据在辅存设备上的分布构成了文件的物理结构，但它对程序的使用是不相适应的。用户看到的应该是逻辑文件结构。文件系统负责实现逻辑特性到物理特性的转换，这实质上是实现了“按名存取”的功能。

(3) 提供合适的访问方法，以适应各种不同的应用。例如，用户不仅可以顺序地对文件进行操作，而且可以任意地对文件中的记录进行操作，即系统应提供顺序存取和直接存取方法。

(4) 提供一组服务，使用户能处理数据，以执行所需要的操作。这些操作包括创建文件、撤销文件、组织文件、读文件、写文件、传输文件和控制文件的访问权限等。

(5) 提供文件保护和共享功能。文件系统还允许多个用户共享一个文件副本。这一服务的目的是在辅存设备上只保留一个单一的使用程序和数据的副本，以提高设备利用率。这时，文件保护尤为重要，系统必须提供对文件的保护措施。

2. 文件系统的功能

文件系统的功能可以很简单，也可以很复杂，这是根据各种不同的应用环境而确定的。对于一个通用的操作系统来说，需要提供下列基本要求：

(1) 文件及目录的管理，如打开、关闭、读、写等。

(2) 提供有关文件自身的服务，如文件共享机制、文件的安全性等。

(3) 文件存储空间的管理,如分配和释放,主要针对可改写的外存,如磁盘。

(4) 提供用户接口。方便用户使用文件系统所提供的服务,称为接口。文件系统通常向用户提供两种类型的接口:命令接口和程序接口。不同的操作系统提供不同类型的接口,不同的应用程序往往使用不同的接口。

3. 文件系统的结构

文件系统的结构模型如图 5-1 所示。

1) 对象及其属性

文件系统管理的对象有:

(1) 文件。它作为文件管理的直接对象。

(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。

2) 对对象操纵和管理的软件集合

这是文件管理系统的核心部分,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。

3) 文件系统接口

为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:

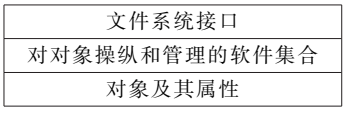

图 5-1 文件系统模型

(1) 命令接口。指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。指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。

5.1.3 文件的分类

为便于文件的控制和管理,通常把文件分成若干类型,但由于不同系统对文件的管理方式不同,因而对文件的分类方法有很大差异。下面介绍几种常用的分类方法。

1. 按文件的数据形式分类

(1) 源文件。由源程序和数据构成的文件,一般由 ASCII 码字符或汉字组成。

(2) 目标文件。由相应的编译程序编译而成的文件,由二进制数组成,扩展名为.obj。

(3) 可执行文件。由目标文件连接而成的文件,扩展名一般为.exe。

2. 按用途分类

(1) 系统文件。由系统进行管理及为用户提供基本服务的文件,这些系统有操作系统、编译系统、编辑系统、预处理系统等。这类文件对用户不直接开放,只能通过系统调用为用户服务。

(2) 库文件。由标准子程序及常用的应用程序组成的文件。这类文件允许用户调用,

但不允许用户修改。

(3) 用户文件。在权限范围内用户可以直接使用,进行读写和执行的文件,如源程序文件、目标程序文件以及由原始数据、计算结果等组成的文件。

3. 按存取权限分类

(1) 只读文件。允许授权用户读,但不准改写文件内容。

(2) 读写文件。允许授权用户读写,但禁止未授权用户读写的文件。

(3) 可执行文件。允许授权用户执行。

4. 按保存时间分类

(1) 临时文件。用户在一次解题过程中建立的中间文件,当用户撤离系统时,该文件往往也随之被撤销。

(2) 档案文件。只保存在作为档案的磁带上,以便考证和恢复用的文件,如日志文件。

(3) 永久文件。长期保存,以备用户经常使用的文件。它不仅在磁盘上有文件副本,而且在"档案"上也有一个可靠的副本。

5. 按对文件管理的方式分类

(1) 普通文件。由表示程序、数据或正文的字符串构成的文件,内部没有固定的结构。这种文件既可以是系统文件,也可以是库文件或用户文件。

(2) 目录文件。由文件目录构成的一类文件。对它的处理(读、写、执行)在形式上与普通文件相同。操作系统将目录作为文件进行管理。

(3) 特别文件。也叫设备文件,特指各种外部设备。为了便于管理,在 UNIX、Linux 和 DOS 中,把所有输入输出设备都按文件格式供用户使用。这类文件对于查找目录、存取权限验证等的处理与普通文件相似,而其他部分的处理要针对设备特性要求做相应的特殊处理。

应该指出,采取不同的分类方式将导致不同的文件系统。

5.1.4 文件存取方式

文件存取方式是指用户对文件的逻辑存取方式,是由文件的性质和用户使用文件的情况决定的。常用的存取方式有顺序存取方式、随机存取方式和按键存取方式。

1. 顺序存取方式

顺序存取是指按照文件的逻辑地址依次存取。对记录式文件,是按照记录的排序顺序依次存取。顺序文件即顺序存放的文件,物理记录的顺序和逻辑记录的顺序是一致的。图 5-2 给出了三种逻辑结构文件的组织形式。

图 5-2(a)是流式文件,可认为该字符流式文件是由一个记录长度为 m(m 为字符流长度)的单记录式文件;图 5-2(b)表示由若干个定长记录组成的一个顺序文件;图 5-2(c)表示由若干个不同长度的记录组成的顺序文件,其中,L_i 为第 i 条记录的长度,R_i 为第 i 条记录的指针。定长记录文件记录的大小为 L,则它的第 i 条记录的逻辑地址为 $i * L$;变长记

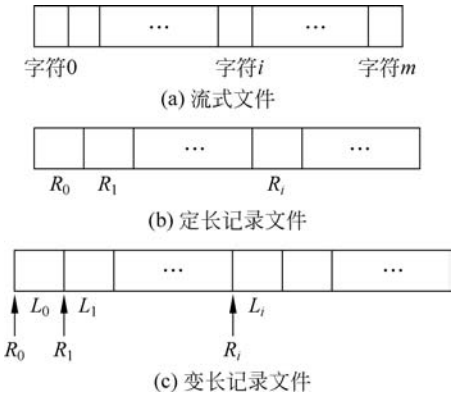

图 5-2 顺序存取方式

录文件,每条记录的大小为L_i($i=0,1,2,\cdots$),则它的第 i 条记录的逻辑地址为$L_0+L_1+\cdots+L_{i-1}$。因为变长记录文件中每个记录的长度都需要记录,所以变量记录的文件确定逻辑位置开销较大,现在这种结构用得较少。

顺序存取方式适用于整个文件只需要顺序读或顺序写的只读或只写文件,但对于某些文件,用户希望能以任意次序直接得到某个记录,采用随机存取方式较为合适。

2. 随机存取方式

随机存取方式又称为直接存取方式,是按照记录的编号或地址来存取文件中的任一记录。对于定长记录文件,随机存取是把一个文件视为若干编上号的块或记录,每块的大小是相同的。随机存取允许随意读入块写入块。因而,对文件的随机存取是没有限定顺序的。当接到访问请求时,计算出记录的逻辑地址,然后存取该记录。

对于变长记录文件,用计算从头至指定记录长度的方法来确定读写位移的方式是很不方便的,通常采用索引表组织方式。

存取文件的步骤如下:

(1) 以记录号为索引,读出索引表中的相应表目。

(2) 根据此表目指针指出的逻辑地址去存取记录。

在无结构的流式文件中,随机存取法必须事先用必要的命令把读写指针移到要进行读写的信息开始处,然后再进行读写。

3. 按键存取方式

按键存取指按逻辑记录中的某个数据项值(称为关键字)作为索引而进行存取。按键存取方式实质上属于随机存取方式。

5.2 文件的逻辑结构

对于文件的组织形式,可以从用户观点和实现观点两种不同的视角来研究,形成两种不同的文件结构,即逻辑文件结构和物理文件结构。

文件的逻辑结构是用户可见的结构，即从用户的角度所观察到的文件结构。文件的逻辑结构分为两种，即无结构文件(流式文件)和有结构文件(记录文件)。

5.2.1 流式文件

流式文件又称无结构文件，是由字符序列组成的文件，其文件内部不再划分记录，文件长度直接按字节来计算，如大量的源程序、可执行文件、库函数等都是无结构文件形式。在Linux系统中，所有文件都被看作流式文件，系统不对文件进行格式处理。

5.2.2 记录文件

记录文件又称为有结构文件，它把文件内的信息划分为多个记录，用户以记录为单位组织信息，即在逻辑上可被看成是一组连续顺序的记录的集合。每个记录是一组相关的数据项集合，每个数据项用于描述一个对象某个方面的属性，如姓名、年龄、性别、工资等。文件有以下分类方式：

1. 按记录的长度分类

有结构文件按其记录的长度是否相同，可分为定长记录文件和不定长记录文件两种。

(1) 定长记录文件是指文件中所有记录的长度都相同。文件的长度可用记录的数目来表示。

(2) 不定长记录文件是指文件中几个记录的长度不相同，如姓名、家庭住址、备注等，有长有短。在处理之前每个记录的长度是已知的。

2. 按记录的组织形式分类

对于记录类型的文件，操作系统根据用户和系统管理的需要，可采用多种方式来组织这些记录，形成下述几种文件。

1) 顺序文件

这是由一系列记录按某种顺序排列所形成的文件。其中的记录通常是定长记录，文件中的所有记录按关键字(词)排列。可以按关键词的长短从小到大排序，也可以从大到小排序，或按其英文字母顺序排序。

对顺序结构文件可有更高的检索效率。在检索串结构文件时，每次都必须从头开始，逐个记录地查找，直至找到指定的记录，或查完所有的记录为止。而对顺序结构文件，则可利用某种有效的查找算法，如折半查找法、插值查找法、跳步查找法等方法来提高检索效率。

顺序文件中的记录可以是定长的，也可以是变长的。

(1) 对于定长记录的顺序文件，如果已知当前记录的逻辑地址，便很容易确定下一个记录的逻辑地址。在读一个文件时，可设置一个读指针 Rptr，令它指向下一个记录的首地址，设每条记录长度为 L，则每当读完一条记录时，便执行

```
Rptr := Rptr + L
```

操作，使之指向下一个记录。

(2) 对于变长记录的顺序文件，在顺序读写时的情况相似，但应分别为它们设置读指针

或写指针,在每次读写完一个记录后,须将读写指针加上 L_i。L_i 是刚读写完的记录的长度。

顺序文件的最佳应用场合是在对诸记录进行批量存取时,即每次要读或写一大批记录时。此时,对顺序文件的存取效率是所有逻辑文件中最高的。磁带机只能存储顺序文件。

在交互应用的场合,如果用户(程序)要求查找或修改单个记录,为此系统需要逐个查找记录。这时,顺序文件所表现出来的性能就可能很差,尤其是当文件较大时,情况更为严重。例如,有记录的顺序文件,如果对它采用顺序查找法去查找一个指定的记录,则平均需要查找 1M 个记录;如果是变长记录的顺序文件,则为查找一个记录所需付出的开销将更大,这就限制了顺序文件的长度。

顺序文件的另一个缺点是,如果想增加或删除一个记录都比较困难。

2) 索引文件

当记录为可变长度时,通常为之建立一张索引表,并为每个记录设置一个表项,以加快对记录检索的速度。

可见,对于定长记录,除了可以方便地实现顺序存取外,还可较方便地实现直接存取。然而,对于变长记录就较难实现直接存取了,因为用直接存取方法来访问变长记录文件中的一个记录是十分低效的,其检索速度也很难令人接受。为了解决这一问题,可为变长记录文件建立一张索引表,对主文件中的每个记录,在索引表中设有一个相应的表项,用于记录该记录的长度 L 及指向该记录的指针(指向该记录在逻辑地址空间的首址)。由于索引表是按记录键排序的,索引表本身是一个定长记录的顺序文件,从而也就可以方便地实现直接存取。图 5-3 示出了索引文件(Index File)的组织形式。

3) 索引顺序文件

这是上述两种文件构成方式的结合。它为文件建立一张索引表,为每一组记录中的第一个记录设置一个表项。

索引顺序文件(Index Sequential File)可能是最常见的一种逻辑文件形式。它有效地克服了变长记录文件不便于直接存取的缺点,而且所付出的代价也不算太大。前已述及,它是顺序文件和索引文件相结合的产物。它将顺序文件中的所有记录分为若干个组(例如,50个记录为一组);为顺序文件建立一张索引表,在索引表中为每组中的第一个记录建立一个索引项,其中含有该记录的键值和指向该记录的指针。索引顺序文件的组织形式如图 5-4 所示。

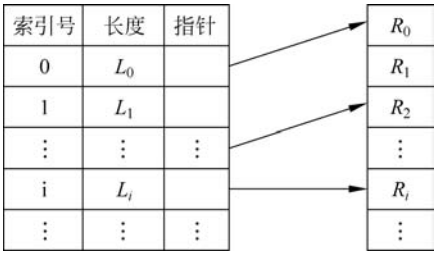

图 5-3 索引文件的组织形式

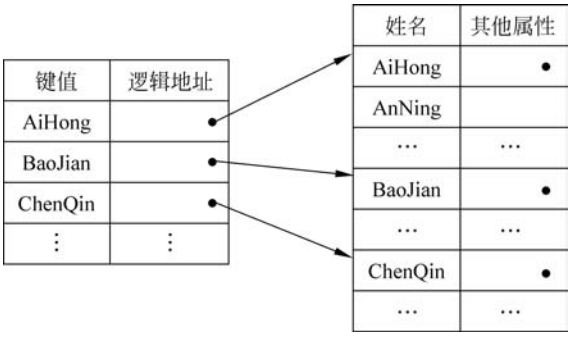

图 5-4 索引顺序文件的组织形式

在对索引顺序文件进行检索时，根据用户所提供的关键字以及某种查找算法去检索索引表，找到该记录所在记录组中第一个记录的表项，从中得到该记录组第一个记录在主文件中的位置；然后，利用顺序查找法查找主文件，从中找到所要求的记录。

对于一个非常大的文件，采用索引顺序文件可以提高查询速度。

4）直接文件

根据给出记录的键值直接决定记录的物理地址。采用前述几种文件结构对记录进行存取时，都必须利用给定的记录键值，先对线性表或索引表进行检索，以找到指定记录的物理地址。对于直接文件，则可以根据给定的记录键值，直接获得指定记录的物理地址。也就是说，记录键值本身就决定了记录的物理地址。这种由记录键值到记录物理地址的转换被称为键值转换(Key to Address Transformation)。组织直接文件的关键在于用什么方法进行从记录值到物理地址的转换。

5）哈希(Hash)文件

这是一种最为广泛使用的直接文件，使用 Hash()函数把键值转换为相应记录的地址。但为了能实现文件存储空间的动态分配，通常由 Hash()函数所求得的并非是相应记录的地址，而是指向一目录表相应表目的指针，该表目的内容指向相应记录所在的物理块，如图 5-5 所示。例如，若令 K 为记录键值，用 A 作为通过 Hash()函数 H 的转换所形成的该记录在目录表中对应表目的位置，则有关系 $A=H(K)$。通常，把 Hash()函数作为标准函数保存于系统中，供存取文件时调用。

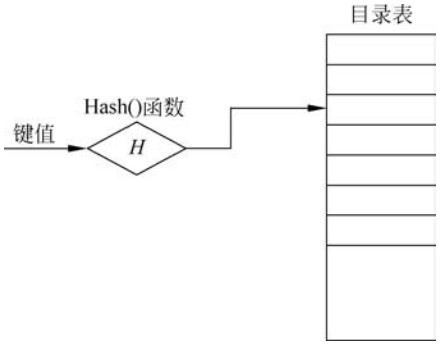

图 5-5 Hash 文件的逻辑结构

5.3 外存分配方式


视频讲解

一个磁盘上会存放许多文件。外存的分配方式是指将磁盘空间分配给文件的方式。它解决的是如何为一个文件分配磁盘空间，以便磁盘空间能够得到有效利用，以及如何提高对文件的访问速度的问题。

外存的分配方式决定了文件的物理结构。文件的物理结构是指文件在外部存储器上的存储方式，以及它与文件逻辑结构之间的对应关系，即文件的存储结构。通常，文件的物理结构有连续文件结构、链接文件结构和索引文件结构等。

5.3.1 连续分配方式

一个文件顺序存放在外存的若干个连续物理块中,这种存放文件的方式是连续分配方式,称这种文件为连续文件。连续文件保证了逻辑文件中的记录顺序与存储器中文件占用盘块的顺序是一致的。在连续文件的文件控制块中记录着文件所占用的起始物理块号和物理块数。图 5-6 所示为连续文件的结构。连续文件的文件控制块(File Control Block,FCB)是用于描述和控制文件的数据结构,图 5-7 给出了连续文件目录的一部分。

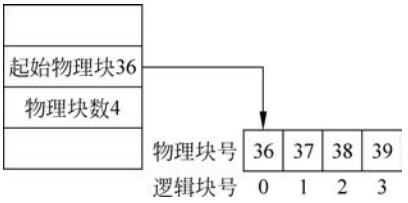

图 5-6 连续文件结构

目录

file	start	length
count	0	2
tr	15	3
mail	21	6
list	29	3
f	7	2

图 5-7 连续文件目录

连续文件的主要优点是:顺序访问容易,访问一个占有连续空间的文件非常容易,同时连续分配也支持直接存取;顺序访问速度快,因为由连续分配装入的文件所占用的盘块可能是位于一条或几条相邻的磁道上,所以磁头的移动距离最少,访问速度快,常用于存放系统文件等固定长度的文件,如编译程序文件、操作系统文件和由系统提供的实用程序文件等。

连续文件也存在以下缺点:首先要有连续的存储空间;要求建立文件时就确定它的长度,依此来分配存储空间,往往难以实现;不利于文件长度的动态增加;反复删除记录后,易产生碎片,导致外存空间利用率低。

5.3.2 链接分配方式

把一个逻辑上连续的文件离散地存放在不连续的物理块中,为了表示其对应的逻辑块次序,对各物理块设置一个指针(称为链接字),指向下一个逻辑块对应的物理块,从而使存放同一文件的物理块链接成一个串联队列,这样形成的物理文件称为串联文件,又称为链接文件。串联文件的优点是支持离散分配,因而消除了碎片,提高了存储空间的利用率;能够实现按需分配且无须事先知道文件的长度,支持文件的动态增长,并方便对文件增、删、改等操作,链接分配方式具有以下两种链接结构。

1. 隐式链接

把一个逻辑文件分为若干个逻辑块,每个块的大小与物理块的大小相同,并为逻辑块取从 1～n 的编号,再把每一个逻辑块存放到对应的物理块中。在每个物理块中设置一个链接指针,通过这些指针把存放了该文件的物理盘块链接起来,由于链接指针只存放在文件的物理块中,所以称为隐式链接,其结构如图 5-8 所示。

隐式链接结构的优点是:克服了连续文件的缺点,但是又带来了新的问题,如只适合顺

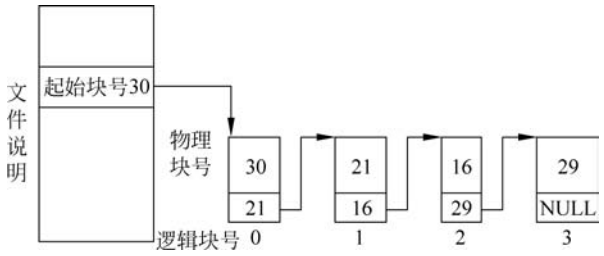

图 5-8 隐式链接结构

序访问，对随机存取极其低效；由于仅通过链接指针来实现各离散盘块的链接，所以只要其中任何一个指针出现问题，都会导致整条链的断开，因而可靠性较差。为了提高检索速度，可将几个盘块组成一个簇，以簇为单位进行盘块分配，但又会带来磁盘簇内碎片增大的缺点。

2. 显式链接

把用于链接文件各物理块的指针显式地放在一张链接表中，该表在整个磁盘上仅设置一张，如图 5-9 所示，表的序号是物理块号。在每个表项中存放链接指针，指向属于同一文件的下一个物理块。文件的第一个物理块号存放在其 FCB 中。由于此表展示了文件在外存上的存储情况，所以称此表为文件分配表(File Allocation Table，FAT)。FAT 实际上就是文件的一张显示链接表。MS-DOS、Windows 都采用了 FAT 文件物理结构。

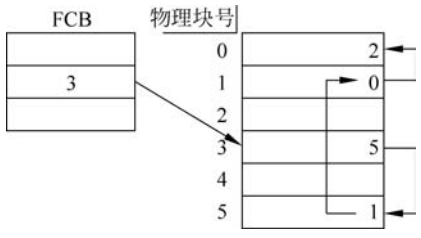

图 5-9 显式链接结构

优点：因为只需将 FAT 一次性装入内存，就可查找所有文件物理块的磁盘地址，所以与隐式链接结构相比，显式链接结构的文件易于采用随机存取，因此随机存取速度快。

链接分配文件方式虽然解决了连续文件方式存在的问题，但又出现了另外两个新问题：不能支持高效的直接存取，因为若对一个较大的文件进行直接存取，必须首先在文件分配表中顺序地查找许多盘块号；文件分配表需占用较大的内存空间。

实例：MS-DOS 的文件物理结构。

磁盘文件卷结构如图 5-10 所示。

图 5-10 MS-DOS 的磁盘文件卷结构

文件卷(Volume)信息：记录在引导记录的扇区中，包括簇大小、根目录项数目、FAT 大小、磁盘参数(每道扇区数、磁头数等)、文件卷中的扇区总数、簇编号长度等。簇(Cluster)：由若干扇区组成，通常一个簇的大小为 2^n 个扇区大小。在一个文件卷中从 0 开始对每个簇编号。

对簇大小的讨论：文件卷容量越大，若簇号所需位数保持不变，那么簇的总数保持不变，只能是簇增大。缺点：簇越大意味着簇内碎片浪费越多，文件卷容量越大，在文件卷容

量不变的前提下,若簇大小不变,则簇总数越多,相应簇编号所需位数越多,可以是 12、16、32 个二进制位,即 FAT12、FAT16 和 FAT32。

FAT:两个镜像,互为备份。文件卷中的每个簇均对应一个 FAT 项,文件分配采用显式分配方法。每个 FAT 项所占位数是簇编号的位数。一个磁盘文件卷有多少簇,则在 FAT 中就有多少表项。因此,如果已知文件卷大小,就可以设计并计算 FAT 的位数以及 FAT 区的大小。例如一个 16 位 FAT 的结构如图 5-11 所示(用工具显示时不带其中的逗号)。

…332A,23BC,55A6,FFFF,76CC,890A,03AC,FFFF,120A,…

图 5-11 一个 16 位的 FAT

FDT:即文件目录分配表,用于存放该文件卷上的根目录文件。子目录文件放在文件区。每个目录项大小为 32B,其内容包括:文件名(8+3 个字符)、属性(包括文件、子目录和文件卷标识)、最后一次修改的时间和日期、文件长度、第一个簇的编号。在目录项中,一开始是文件名,因此第一个字节为文件名的首字节。若该字节为 E5h,则表示空目录项或表示该目录项已被删除。

在访问一个文件或其属性时,通过文件名检索 FDT,找到该文件的目录项,通过 FDT 中的首簇号就可以引导该文件在 FAT 区的其他簇号,从而可以访问该文件,如图 5-12 所示。

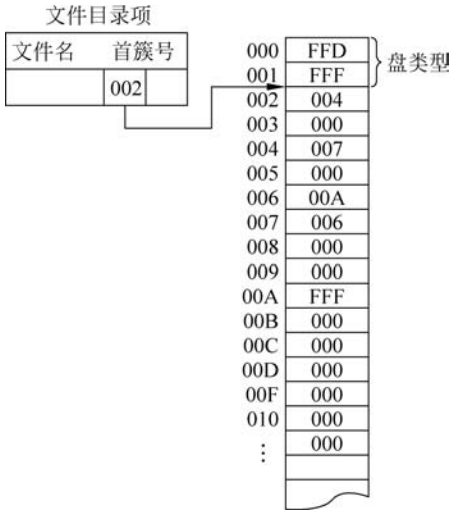

图 5-12 FDT 引导 FAT

例 5-1 对于一个 1.2MB 的磁盘,盘块大小为 1KB,共有盘块 1.2K 个,FAT 中的每个分配表项占 16 位,即 2B,所以共需 2.4KB 的存储空间。对于 540MB 的硬盘,共有盘块 540MB/1KB=540K∈(2^{19}, 2^{20}),故文件分配表表项应取 20 位,即 2.5B,所以其文件分配表需占用存储空间为 540K×2.5B=1350KB。

5.3.3 索引分配方式

1. 直接索引

索引文件是实现非连续分配的另一种方案,系统为每个文件建立一张索引表,索引表是

文件逻辑块号和磁盘物理块号的对照表。此外，在文件控制块中设置了索引表指针，它指向索引表的起始地址，索引表存放在盘块中，如图 5-13 所示。

索引文件克服了连续文件和串联文件的不足，既能方便、迅速地实现随机存取，又能满足文件动态增长的需要。由于它的检索速度较快，所以主要用于对信息处理及时性要求较高的场合，但是增加了索引表带来的存储空间开销。在存取文件时，需要先取出索引表，然后再查表，得到物理块号，这样增加了存取时对存储器的访问次数，降低了文件的存取速度，加重了输入输出的负担。一种改进办法是将索引表部分或全部放入内存，以内存空间为代价换取存取速度的改善。

2. 多重索引

若文件很大，那么不仅存放文件信息需要大量盘块，而且相应的索引表也必然很大。

例如，若盘块大小为 1KB，那么长度为 1000KB 的文件就需要 1000 个盘块，索引表项要有 1000 项。若盘块号用 4B 表示，则索引表至少占用 4000B。显然，把索引表全部放入内存是不合适的，而且不同文件其大小也不同，文件在使用过程中很可能需要扩充空间。采用单重索引文件结构无法满足灵活性和节省内存的要求，为此引入了多重索引文件结构，其结构如图 5-14 所示。在这种结构中采用了间接索引方式，即由最初索引项中得到某一盘块号，该块中存放的是另一组盘块号，后者每一盘块中又可存放下一组盘块号，最后的盘块中存放的一定是文件内容。

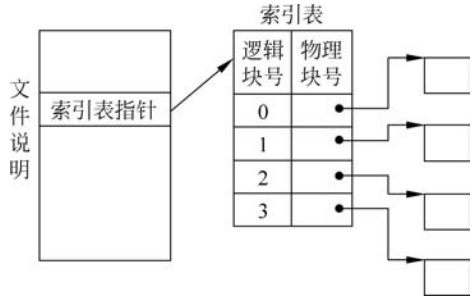

图 5-13 索引文件结构

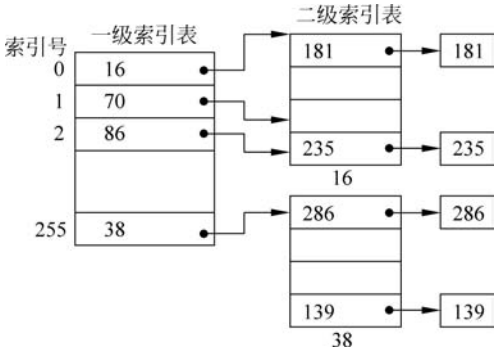

图 5-14 多重索引文件结构

3. 混合索引

从物理结构上看，Linux 采用的是混合索引文件结构，即将文件所占用盘块的盘块号，直接或间接地存放在该文件索引节点的地址项中。在查找文件时只要找到该文件的索引节点就可以用直接或间接的寻址方式获得指定文件的盘块号。图 5-15 所示为 Linux 的混合索引文件结构。

1）直接寻址方式

Linux 系统中的作业以中小型为主，为了提高对文件的检索速度，宜采用直接寻址方式。在索引节点中建立 12 个地址项用来直接存放该文件所在的盘块号，相应的盘块称为直接块。例如，设盘块大小为 1KB，某进程要访问字节偏移量为 7000B 处的数据。首先将

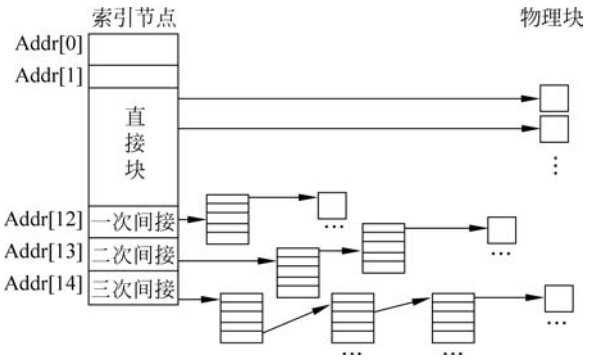

图 5-15 Linux 的混合索引文件结构

7000B 转换为文件逻辑块号 7000/1024＝6，块内位移量 7000％1024＝856(B)。其逻辑块号小于 12，所以该块为直接块，从 addr[6]中读出对应的物理块号，在该物理块第 856B 处即为文件的第 7000B。

2）一次间接寻址方式

当文件较大时，Linux 系统提供了一次间接寻址方式。在这种寻址方式中，一次间接地址项中所对应的盘块(间接块)存放的不是文件所在的物理盘块号，而是直接块的块号表。为了通过间接块读取文件数据，关键是要先读出间接块，找到相应的直接块项，然后从直接块中读取数据。

当计算出的文件逻辑块号大于或等于 12 而小于 268 时，采用一次间接寻址方式。将逻辑块号转换为物理块号的方法是从一次间接项中得到一次间接的盘块号，根据该间接块的内容计算一次间接块中的地址下标，即将文件的逻辑块号减 12，从相应下标的地址项中得到物理块号。

例如，盘块大小为 1KB，某进程要访问字节偏移量为 28 000B 处的数据。先将 28 000B 转换为文件逻辑块号 28 000/1024＝27，块内位移量为 28 000％1024＝352(B)。由于逻辑块号大于 12 而小于 268，所以采用一次间接寻址方式。先从一次间接项中得到一次间接的盘块号，再从一次间接块的地址下标为 15(即 27－12)的地址项中得到其物理块号，在该物理块中的第 352B 即为要读取的数据。

3）多次间接寻址方式

对于大型和巨型的文件，Linux 系统又引入了二次间接寻址和三次间接寻址。二次间接项中存放的是一次间接块号表，三次间接项所对应的盘块中存放有二次间接块号表。

在 Linux 系统中，采用混合索引文件结构的优点是：对于小文件，访问速度快；对于大中型文件，其文件系统也能很好地支持。


视频讲解

5.4 文件目录管理

文件系统要解决的核心问题，就是把文件信息的逻辑结构映像成设备介质上的物理结构，把用户的文件操作转换为相应的输入输出指令。操作系统转换过程中所使用的主要数据结构是文件目录。文件目录将每个文件的文件名和它在外存空间的物理地址以及文件属

性的说明信息建立了联系，这样，用户只需要向系统提供一个文件名字符串，系统就能准确地找出需要的文件，这就是文件系统的按名访问功能，也是文件管理各个功能的核心问题。因此，目录的构建应以如何才能够准确地定位到所需的文件物理地址为原则，选择查找目录的方法应该以查找速度快为目标。

文件目录是记录系统中所有文件的文件名及其存放地址的目录表，表中还包含文件属性相关信息，如文件的描述信息和文件的控制信息等。

文件目录用于查看和读取外存中所存放文件名其属性、对文件进行描述和文件控制、实现按名存取和文件的共享与保护。文件目录项随着文件的建立而创建，随着文件的删除而消亡。在很多操作系统中，对目录采用文件的方式进行管理。

5.4.1 文件控制块和索引节点

1. 文件控制块

为了便于对文件进行控制和管理，必须为文件设置用于描述和控制文件的数据结构，这种数据结构称为文件控制块，文件与文件控制块一一对应。一个文件控制块可以是一个文件目录项，完全由目录项构成的文件称为目录文件。文件控制块通常由文件的基本信息、存取控制信息和文件使用信息组成。

1）基本信息

文件名：用于标识一个文件的名字。每个文件必须有唯一的文件名，用户可根据文件名对文件进行操作。

文件类型：指明文件属性是系统文件还是用户文件，是普通文件还是目录文件，或是特别文件等。

文件物理位置：指明文件在外存上存放的物理位置和范围，包括文件的设备名、文件在外存的起始地址、文件长度等。

文件的逻辑结构：指明文件是记录文件还是流式文件，若是记录文件，需指明记录个数及记录是变长记录还是定长记录等。

文件的物理结构：指明文件是连续文件、链接文件或是索引文件。

2）存取控制信息

文件所有者（属主）：通常是创建文件的用户，或者改变已有文件的属主。

访问权限（控制各用户可使用的访问方式）：为了防止用户有意或无意地破坏文件，对文件设立保护，规定其允许访问的操作类型，如读、写、执行、删除等。

3）文件使用信息

日期和时间：文件创建和上一次修改的日期和时间。

当前使用的信息：包括有多少个进程正在使用该文件、是否被其他进程锁住等信息。

应该说明，对于不同操作系统的文件系统，有不同的文件目录。也就是说，不同的操作系统其文件控制块中所包含的信息也不同，可能只含有上述信息中的某些部分。

例 5-2 MS-DOS 系统中的文件控制块的长度为 32B，它含有文件名及文件扩展名共 11B，包括文件所在的首块号、文件属性、文件大小及文件建立和修改的日期等。利用文件所在的首块号作为物理块链接表的索引，在 FAT 中按索引链向下查找，可找到文件占用的

所有盘块号。图 5-16 所示是 MS-DOS 的文件控制块示意图。

0 7	8 A	B	C F	10 15	16 17	18 19	1A 1B	1C 1F
文 件	扩展名	属性	保留	时间	日期	首簇号	大小	大小

图 5-16 MS-DOS 的文件控制块

DOS 系统的文件目录的构成是：文件名＋文件控制块。

由于 DOS 系统的文件属性(即文件控制块)结构较小，占用磁盘空间小，因此这种结构在实现文件的按名访问时，查询目录速度较快。

2. 索引节点

1) 索引节点的引入

我们通常把文件目录存放在磁盘上，当文件很多时，文件目录可能需要占用大量的盘块。在查找目录的过程中，先将目录文件中第一个盘块中的 FCB 调入内存，然后把用户给定的文件名与 FCB 中的文件名进行比较，若未找到，则将下一个盘块调入内存，直到找到或确定没有与之匹配的文件名为止。设文件目录占用的盘块数为 M，按此方法查找，则查找一个目录项，平均需调入盘块$(M+1)/2$ 次。例如，一个 FCB 长度为 128B，盘块大小为 1KB，则一个盘块可以存放 8 个 FCB，若一个文件目录共有 640 个 FCB，则需占用 80 个盘块，平均查找一个文件需要启动磁盘 40 次。

经过分析可以发现，在检索文件目录的过程中，只用到了文件名，只有目录项中的文件名与指定的文件名相匹配时，才需从 FCB 中读出该文件的物理地址等信息，而其他一些描述信息在检索目录时一概不用，显然，这些信息在检索目录时不需要调入内存。为此，在有些系统中便采用了文件名与文件描述信息分开的办法，即把文件描述信息单独形成一个数据结构，称为索引节点，简称为 I 节点。文件目录中的每个目录项仅由文件名及指向该文件所对应的索引节点的指针构成。

Linux 系统中的目录项由文件名和 I 节点号组成，每个文件对应唯一的 I 节点号，如图 5-17 所示。

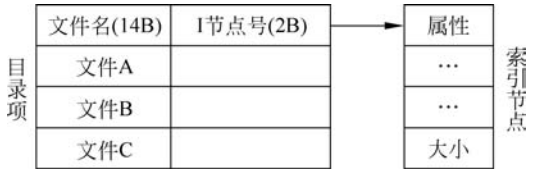

图 5-17 Linux 系统的文件目录

Linux 系统中，一个目录项共占 16B，其中文件名占用 14B，I 节点占用 2B。在大小为 1KB 的盘块中，可存放 64 个 FCB。在一个共有 640 个 FCB 的文件目录中查找一个文件时，平均只需启动磁盘 5 次，因此大大减少了系统开销。

Linux 系统使用文件名和索引节点号作为文件的目录，文件目录的大小被缩小，因此目录文件的长度也减小，检索文件目录所需访问的物理块数也就相应减少，所以加快了目录检索的速度。

2）磁盘索引节点

它是指存放在磁盘上的索引节点。每个文件有唯一的一个磁盘索引节点，它主要包括以下内容。

文件所有者标识号：指拥有该文件的文件主或同组的标识符。

文件类型：指明文件是普通文件、目录文件还是特别文件等类型。

文件物理地址：指出数据文件所在的物理块号。如在 Linux 系统中，通过 15 个地址项来表明文件所在的物理块号。

文件存取权限：用户对文件的操作类型，如读写、修改、执行等。

文件大小：文件所占有的字节个数。

文件链接计数：指明系统中共享该文件的进程个数。

文件存取时间：指出该文件最近被进程存取的时间、最近被修改的时间及索引节点最近被修改的时间等。

3）内存索引节点

它是指存放在内存的索引节点。当文件打开时，要将磁盘索引节点复制到内存索引节点中，便于以后使用。内存索引节点包括以下内容。

索引节点编号：标识内存索引节点。

索引节点状态：指示该节点是否已被修改或已被上锁。

访问计数：当进程访问该节点时，访问计数加 1，访问完再减 1。

链接指针：指向空闲链表和散列队列的指针。

逻辑设备名：含有该文件的文件系统的逻辑设备名。

5.4.2　文件目录结构

文件目录结构的组织关系到文件的存取速度、文件的共享性和安全性。因此，组织好文件的目录是设计文件系统的重要环节。下面介绍几种常用的目录结构组织形式。

1. 一级目录结构

一级目录结构是最简单的目录结构，如设备目录就是一级目录，它是指把系统中的所有文件都建立在一个目录下，每个文件占用其中的一个目录项，每个目录项中包含了文件名、文件物理地址和文件说明信息等，如图 5-18 所示。

目录项

文件名	物理地址	文件说明	状态位
文件名 1			
文件名 2			
…			

图 5-18　一级文件目录结构

一级目录结构主要用于单用户操作系统，它具有如下优点：

（1）结构简单，通过管理其目录文件便可实现文件信息的管理。

（2）实现按名存取。

同时,一级目录结构具有以下缺点:

(1) 文件较多时,目录检索时间长。

(2) 有命名冲突。简单的文件目录结构中,文件名和文件实体之间存在着一一对应的关系,即不允许两个文件具有相同的名字。在多道程序系统中,尤其是多用户的分时系统中,重名很难避免,这就很难准确地找到用户需要的文件。显然,如果用人工管理文件名注册,以避免命名冲突,会非常麻烦。

(3) 不便于实现文件共享。一级目录结构要求所有用户用相同的名字访问同一个文件。

2. 二级目录结构

在多用户系统中,为解决不同用户的文件重名问题,可以建立两级目录结构。在两级目录结构中,除了系统目录外,还在系统目录下为每个用户建立一个用户文件目录(第二级目录);在用户目录下是该用户的文件,而不再有下级目录。二级目录结构适用于多用户系统,各用户可有自己的专用目录,如图 5-19 所示。

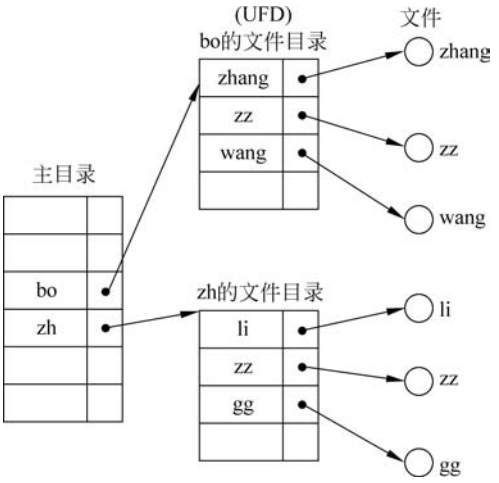

图 5-19 二级目录结构

二级目录结构基本上克服了一级目录结构的缺点而具有以下优点。

(1) 提高了检索目录的速度。

(2) 在不同的用户目录中可以使用相同的文件名。

(3) 不同用户可使用不同的文件名来访问系统中的同一个共享文件。

3. 多级目录结构

(1) 目录结构。为了给某些使用多个文件的用户提供检索方便,以及更好地反映实际上多层次的复杂的文件结构关系,可以把二级目录自然推广到多级目录,即树形目录。图 5-20 所示为多级目录结构。

(2) 路径。路径是指从树形目录中的某个目录层次到某个文件的一条道路。此路径的主要构成是目录名称,中间用“/”或“\”分开。任意文件在文件系统中的位置都是由相应的

路径决定的。用户对文件进行访问时，要给出文件所在的路径。路径一般分为相对路径和绝对路径。

注意：绝对路径名以“/”开头。

相对路径名常和工作目录(也称当前目录)一起使用。用户可以指定一个目录作为当前的工作目录。这时，所有的路径名如果不是从根目录开始，则都是相对于工作目录的。在图 5-20 中，若当前目录为/A，则文件 L 的相对路径为 AC/L。

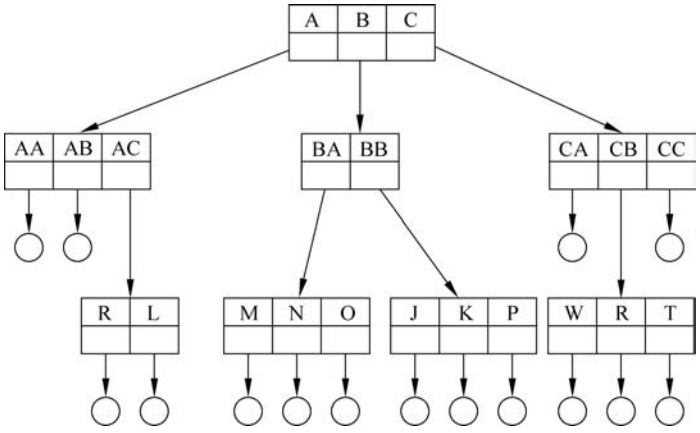

图 5-20 多级目录结构

注意：相对路径名是从当前目录的下级开始书写。

说明：大多数支持树形结构的操作系统，在每个目录中有两个特殊的目录项“.”和“..”，通常读作“点”和“点点”。“点”指当前目录，“点点”指其父目录。在图 5-20 中，若某进程的工作目录为/A/AC/R，它可以使用“..”沿树向上到达其父目录/A。

多级目录结构具有以下几个优点：能有效地提高对目录的检索速度；允许文件重名，允许用户在自己的分目录中使用与其他用户相同的文件名；便于实现文件共享，允许不同的用户按自己的命名习惯为共享的文件赋予不同的名字，若某一用户欲共享另一用户的文件，只需在权限许可的前提下，在自己的目录文件中增设一表目，其中的文件名项使用自己赋予该文件的符号名字，并填上该共享文件的唯一标识符即可。

4. 图形目录结构

所谓图形目录结构，就是在树形目录结构的基础上增加了一些指向同一节点的有向边，从而使整个目录的结构成为一个有向无循环图。

图形目录结构的引入是为了文件共享，因为树形目录结构不便于文件和目录的共享。而如果对于两个不同的目录都保存了同一个文件(此文件在磁盘中只有一份，没有备份)，那么这两个目录下都有这个文件，也就是说指向了这个文件，那么这个目录结构就不再是树形的，而是构成了一个有向非循环图，所以称为图形目录结构。图 5-21 所示为一个 Linux 系统的有向非循环图结构。

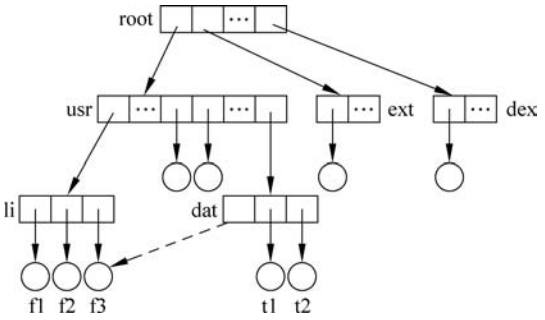

图 5-21 Linux 系统的有向非循环图结构

5.4.3 按名存取

用户访问文件时,系统首先根据文件名查找文件目录,找到它的文件控制块或索引节点号;然后经过合法性检查从控制块或索引节点中找到该文件所在的物理地址,换算为物理位置;然后再启动磁盘驱动程序,将所需的文件读入内存,进行相应的操作。目前,对文件目录进行查找的方式有顺序检索法和 Hash 方法。

1. 顺序检索法

顺序检索法又称线性检索法,在一级目录结构中,利用用户提供的文件名,用顺序查找的方法直接从文件目录表中找到指定文件的目录项。在树形目录结构中,用户提供的文件名是由多个文件分量名组成的路径名,此时需对多级目录进行查找,即系统先读入第一个文件分量名,用它和根目录文件或当前目录文件中各个目录项进行比较,若找到匹配者,便可找到匹配项的文件控制块或索引节点,然后再读入路径名中的第二个分量名,用它和相应的第二级文件目录中各个文件目录项的文件名顺序比较,若找到匹配项,再取第三个、第四个文件分量名进行比较,直至全部查完,最后可得到数据文件的文件控制块或索引节点。若在查找过程中发现一个分量名也没有查找到,则应停止查找并返回“文件未找到”的信息。

2. Hash 方法

建立一张 Hash 索引的文件目录,利用 Hash 方法进行查找,即系统利用用户提供的文件名,将它变为文件目录的索引值,再利用该索引值到目录中去查找,提高平均检索速度。顺便指出,现代操作系统通常提供模式匹配功能,即在文件名中使用通配符“*”“?”等。对于使用通配符的文件名,系统无法利用 Hash 方法进行检索目录。因此,还是需要利用顺序检索法来查找目录。

5.5 文件存储空间的管理

文件存储空间的管理是对磁盘空闲空间的管理,建立关于空闲盘块的数据结构,给文件分配存储空间提供依据。不同的外存空间分配方式(文件物理结构)可能采用不同的空闲盘块的数据结构。

下面介绍几种常见的存储管理方式。

5.5.1 空闲空间表法

系统为外存上的所有空闲区建立一张空闲表，每个空闲区对应一个空闲表项，包括序号、第一个空闲磁盘盘块号、空闲磁盘块个数以及对应的空闲物理块号等，如表 5-1 所示。

表 5-1 空闲空间表

序 号	第一个空闲磁盘块号	空闲磁盘块个数	物理块号
1	2	4	(2,3,4,5)
2	19	3	(19,20,21)
3	17	2	(17,18)
…	…	…	…

这种管理方法适用于采用顺序结构的文件。空闲盘区的分配与内存的动态分配类似，同样是采用首次适应算法、循环首次适应算法等。例如，系统为某新创建的文件分配空闲盘块时，先顺序地检索空闲表的各表项，直至找到第一个大小能满足要求的空闲区，再将该盘区分配给用户(进程)，同时修改空闲表。系统在对用户所释放的存储空间进行回收时，也采取类似于内存回收的方法，即要考虑回收区是否与空闲表中插入点的前区和后区相邻接，若相邻接，则把它们合并成一个大的空闲区，记在一个表项中。

5.5.2 位示图法

位示图是反映整个文件存储空间分配情况的一种数据结构。

1. 位示图

这种方法是利用二进制的一位来表示磁盘中每一个盘块的使用情况，磁盘上的所有盘块都有一个二进制位与之对应，从而由所有盘块所对应的位构成一个集合，即位示图。当其值为“0”时，表示对应的盘块空闲；为“1”时，表示已分配。例如，设下列盘块是空闲的：2，3，4，5，8，9，10，11，12，13，17，18，25，26，27，…，则对应的位示图为：

1000011000000111001111111000…

为所要管理的磁盘设置一张位示图。位示图的大小由磁盘的总块数决定，每一个盘块与位示图的一个二进制位对应。因为位示图所占空间较小，所以可以复制到内存中，使得盘区的分配和释放都可以高速进行。当关机或文件信息转存时，位示图信息要完整地在盘上保留下来。现代磁盘的容量都很大，一般划分为多个分区，可针对每个分区设立一个位示图。

2. 空闲盘块的分配

当分配存储空间时，查找位示图中对应位为 0 的位，然后将其转换为对应的物理块号，将其分配给申请者，并将该位置 1。根据位示图为某文件分配盘块的具体过程如下。

(1) 顺序扫描位示图,找出其值为空闲即 0 的二进制位。

(2) 将二进制位的行/列号转换为与之对应的盘块号,假设找到的值为 0 的二进制位位于位示图的第 i 行第 j 列,则相应的盘块号为 $b=n(i-1)+j$,其中 n 代表每行的位数。

(3) 把盘块号给该文件,同时修改位示图中的二进制位 Map[i,j]=1。

3. 盘块的回收

删除文件时,需要回收存储空间,即把原来所占用的物理块归还给系统,并将物理块对应位置 0。根据位示图回收盘块的具体过程如下。

(1) 将回收盘块的盘块号 b 转换为位于位示图中的行号 i 和列号 j,$i=(b-1)/n+1$,$j=(b-1)\%n+1$。

(2) 修改位示图中的二进制位 Map[i,j]=0。

位示图的优点是占用的存储空间少,因此可以将位示图全部装入内存,使得盘区的分配和回收都可以高速地进行。但分配时,需要顺序扫描位示图,且物理块号在图中并未直接反映出来,需要计算所在的物理块号。

5.5.3 空闲块链法

这种方法是将磁盘上的所有空闲存储空间以盘块为单位拉成一条链,用一个指针指向第一个空闲块,而各个空闲块中都含有下一个空闲区的块号,最后一块的指针项记为 NULL,如图 5-22 所示。

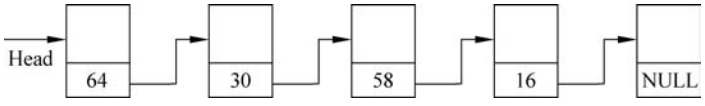

图 5-22 空闲块链

当用户因创建文件而请求分配存储空间时,就从链表头依次取下适当数目的空闲块,分配给用户。当删除文件时,就把新释放的块从链表插入,并使头指针指向最后释放的那一块。

这种管理方法适用于非连续分配。由于各个空闲块的链接指针隐含在空闲磁盘块中,因此管理时所需的额外开销很少,但工作效率较低,因为在空闲块链上增加或删除空闲块时需要做许多 I/O 操作。

5.5.4 空闲块成组链接法

空闲空间表法和空闲块链法都不适合用在大型的文件系统中,因为会使空闲表或空闲链太长。在 Linux 系统中,采用了一种改进的方法,称为成组链接法。这种方法兼备了上述两种方法的优点而克服了两种方法均有的表太长的缺点。

1. 空闲盘块的组织

(1) 将文件区中的所有空闲盘块依次进行分组,如将 100 个空闲盘块划分为一组,将组

中的第一块称为“组长块”，第一组为 99 块。从第二组开始，每组都为 100 块，剩下的块归并为后一组。

(2) 将每一组的盘块总数和该组所有盘块号放入后一组的“组长块”中，这样，由每一组的第一个盘块构成一条链。

(3) 第一组中只有 99 个空闲盘块，为了管理的需要，把它的总块数仍记为 100，而第二组的“组长块”中标记的第一组的首块号为 0，作为空闲盘块的结束标志(因为此标志占用了一个盘块号项，所以第一组中只有 99 个盘块)。

(4) 将最后一组的盘块总数和空闲盘块号存入空闲盘块专用栈中，这样，空闲盘块的分配和回收就在这个专用栈中进行，如图 5-23 所示。

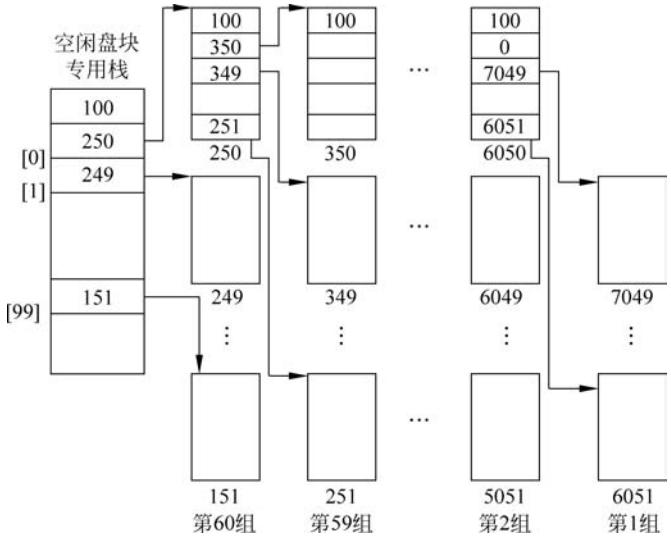

图 5-23 空闲块成组链接法

2. 空闲块的分配

当需要分配空闲盘块时，先把专用栈中表示栈深(即栈中有效元素的个数)的数值减 1，这里是 100－1＝99，以 99 作为检索专用块中空闲盘块号栈的索引。如图 5-23 所示，盘块号为 151，它就是当前分配的第一个空闲盘块。如果需要分配多块，重复上述操作。

需要注意的是，当栈深值减 1 后，其值等于零，此时系统需要做特殊处理，因为这一组的第 1 个盘块(总是在一组中的最后被分配出去的盘块)包含它前一组空闲块的信息。因此，在把这一块分配出去之前，应先把它记录的信息复制到专用栈中，然后再分配出去。分配时还要注意一个问题，若要把第二组的第 1 块分配出去，那么先把它所含信息复制到专用栈中，这就意味着只有 99 个盘块可供分配了。继续分配，直到把 99 个盘块全部分配出去。若再申请空闲盘块，栈深值减 1 变为 0，此时发现对应的空闲盘块号为 0，表示所有的空闲盘块号都已经分配出去了，对应进程只能等待。

3. 空闲块的回收

当删除文件，需要回收盘块时，将回收盘块的盘块号记入空闲盘块号栈的顶部，并将空

闲盘块数加 1。当栈中的空闲盘块号数目已达到 100 时,表示栈已满,若还要回收盘块,则需进行特殊处理:将栈深值和各空闲块的块号写到要释放的新盘块中;将栈深值及栈中盘块号清 0;将新回收的盘块写入相应单元中,栈深值加 1。新回收的块将作为新组的组长块。

5.6 文件共享与安全性

5.6.1 文件的共享

文件共享是指多个用户或进程可以共同使用一个或多个文件。利用文件共享可以减少大量的外存空间和内存空间,同时为用户完成各自的任务带来很大方便。随着计算机技术的发展,文件共享已不限于单机系统,而扩展到了全球的计算机网络系统。

实现文件共享的方法有两种:绕弯路法和链接法。

1. 绕弯路法

这是早期的 MULTICS 等操作系统中所采用的一种共享方法。在该方法中,允许每个用户获得一个当前目录,一般用“.”表示当前目录。当所访问的文件不在当前目录时,可以通过“向上走”的方式去访问其上级目录,一般用“..”表示目录的父目录。

2. 链接法

在树形结构目录中,当有两个(或多个)用户要共享一个子目录或文件时,必须将共享文件或子目录链接到两个(或多个)用户的目录中,以便能方便地找到该文件,此时该文件系统的目录结构是非循环树形目录结构(Directed Acyclic Graph),如图 5-24 所示。

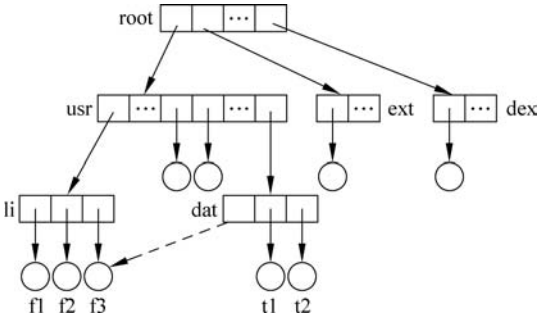

图 5-24 文件的链接

建立链接的基本思想是用一个文件目录项直接指向要共享文件的目录项,从而在两个文件之间建立起一种等价的关系。具体做法是在文件主允许的情况下,一个用户在他的文件目录中开辟一个目录项,该目录项直接指向所要共享的那个文件的目录项。

为了建立目录与共享文件之间的链接,并顺利实现共享,可以引用索引节点,即诸如文件的物理地址及其他的文件属性等信息,不再放在目录项中,而放在索引节点中。在文件目录中只设置文件名及指向相应索引节点的指针。

例如，用户 dat 要共享文件 li 中名为 f3 的文件。用户 dat 在自己的目录中开辟一个目录项，其指针直接指向用户 li 的文件 f3。这样，在用户 li 和用户 dat 之间就建立了一个链接关系，用户 dat 可以直接访问 f3 这个文件，如图 5-25 所示。

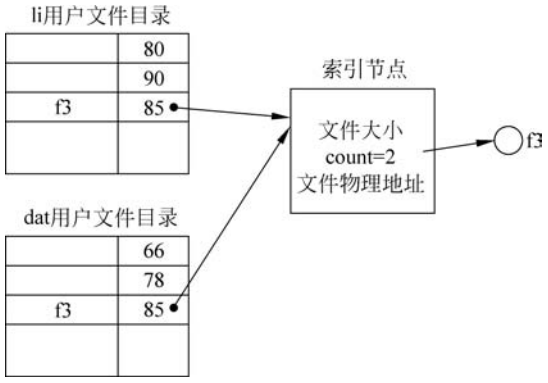

图 5-25　基于索引节点的共享方式

在 Linux 系统中，对文件的链接命令是 ln，命令格式为：

```
ln  <带路径指引的文件名称>  <带路径指引的目录名称>
```

例如，要将当前目录(/usr/li)下的文件 f3 链接到目录/usr/dat 中，可使用命令：

```
ln  f3  /usr/dat
```

执行后结果如图 5-25 所示。

此时，无论执行：

```
$ lst /usr/li/
```

还是执行：

```
$ ls /usr/dat
```

均会显示文件 f3 (ls 为显示目录内容命令)，而且无论在哪一个目录中修改文件 f3，在另一个目录中都能发现已做相应修改的文件 f3。若要在文件系统中彻底删除一个做了链接的文件，必须在其所属的所有目录下都进行删除。例如在上例中，如果要删除文件 f3，必须执行两次删除：

```
$ rm /usr/li/f3
$ rm /usr/dat/f3
```

如果只删除一次，则在另一目录中仍可看到文件 f3。

链接的具体实现过程是：

(1) 根据源文件名检索目录树，找到对应的文件控制块，并复制到内存中。

(2) 根据新的文件名检索目录树，若找到该文件，则判断出错。

(3) 若未找到，则在新目录中登记该文件的目录项，并增加源文件的链接计数。

值得注意的是，链接文件并不是创建文件，只是增加一条共享文件的路径。

5.6.2 文件的安全性

文件的共享给人们带来很多好处和方便的同时,也潜藏着诸多不安全的因素,如人们有意或无意的行为使文件系统中的信息遭到破坏或丢失、自然灾害造成的机器损坏、系统故障造成的数据破坏或丢失等,这些给系统的安全性带来了很大的影响。因此,在现代计算机系统中,不仅要为用户提供共享的便利,而且要充分注意系统和数据的安全性和保密性。

文件保护是指防止未经授权的用户使用文件或文件主的错误操作对文件造成破坏。通常可以采用口令、密码、存取控制矩阵和存取控制表等保护机制来达到保护文件不受侵犯的目的。

1. 口令方式

用户在文件创建时,可以为每一个文件设置口令(Password),并将其记录在文件说明信息中。文件系统在用户试图访问该文件时,首先要求用户提供口令并与文件说明信息中记录的口令进行比较,若匹配用户才能够存取该文件。

通过这样的方法可以简单地实现文件的共享与保密。口令验证的过程比较简单,占用空间少;保密性能相对较差,易被窃取。

2. 密码方式

为了防止破坏或泄密,对一些重要信息可采用密码方式来存储。密码(Encryption)方式是在文件创建时,文件主在文件写入存储设备之前,通过特定的算法和加密密钥对文件内容进行编码加密,读取文件时,必须提供相应的解密密钥进行译码解密。只有确切知道解密密钥的用户才能够读出被加密的文件。

密码方式保密性强,节省存储空间,但是加密和解密要花费很多时间,增加系统开销。

口令方式中有时也采用一定的加密技术。加密口令方式只是对口令本身加密,加密后的口令仍然存放在文件说明信息中,而密码方式是对整个文件进行加密,加密时采用的密钥掌握在用户自己手中,因此,密码方式的保密性强,但是要耗费大量的处理时间。以上两种文件存取权限控制方法是针对特定文件的,在多用户系统中,还可以针对特定的用户进行文件存取权限控制。大多数系统,包括 Linux,用户身份验证都采用用户口令方式,建立用户时,设置其口令,口令以加密的形式存放在系统中,每次用户登录都要提供口令,并与系统中存放的数据进行比较,如果相同,则用户登录成功,否则,拒绝用户登录。在这样的系统中,多个用户可以同时使用计算机,每个用户都拥有自己的文件,这些文件的存取使用控制权由用户拥有。

3. 存取控制矩阵

存取控制矩阵是指在整个系统中建立一个二维表,一维列出系统中的所有用户名,一维列出系统中的所有文件名,矩阵的每一个元素都表示该元素对应的用户对文件的存取控制权限,如表 5-2 所示。当用户对某个文件进行操作时,系统查找矩阵中该用户和该文件所对应的元素的值,只有当该元素表明有访问许可时,才能进行相应的访问。

表 5-2 存取控制矩阵

用 户	AA	AC	LI	FLAG	POIN
用户 1	RW	R	RE	RWE	
用户 2	E		R		RW
用户 3	RWE	E		E	
…			…		

存取控制矩阵可以对系统中所有文件的存取权限进行完整、有效的控制。但是当系统中用户和文件数很大时，存储和管理这样的矩阵需要比较大的系统开销，并且用户每访问一个文件，都要对矩阵的一行进行搜索，访问效率比较低。

4. 存取控制表

存取控制表是把文件主和相关的用户对某个文件的访问按照某种关系分类，不同的用户类赋予不同的文件存取权限。Linux 系统中，文件的存取权限分别赋予文件主、文件主所在用户组和其他用户组这三类用户，每一类访问的权限设置三位，分别为读、写和执行，这样就相当于每个文件只需要记录九位数据就可以进行权限控制了，如表 5-3 所示。

表 5-3 存取控制表

用 户	文件名
文件主	RWE
同组用户	RE
其他用户	E

实际上，每个文件的存取控制表都相当于整个存取控制矩阵中的一列元素。存取控制矩阵中的每一列对应于一个文件，去掉空元素，然后把所有用户按照文件所有者、所有者所在组和其他用户组简化后，就可以得到该文件的存取控制表，如表 5-3 所示。

在 Linux 系统中，每一个文件都在文件说明信息中保存着自己的文件存取控制表，只有符合存取控制表中规定的条件，才允许进行具体的访问。

需要说明的是，设置文件访问口令和文件加密属于文件级的访问权限控制。在多用户系统中，必须进行用户验证。文件系统针对特定的用户，可以通过存取控制矩阵或存取控制表来设置对文件的存取权限。这类方法可以认为是基于用户身份的存取权限控制，属于用户级的访问权限控制。

5.7 Linux 文件系统

在 UNIX、Linux 等操作系统中，把包括硬件设备在内的能够进行字符流式操作的内容都定义为文件。Linux 系统中文件的类型包括普通文件、目录文件、链接文件、管道(FIFO)文件、设备文件(块设备、字符设备)和套接字。操作系统根据文件的类型来处理文件。

5.7.1 文件类型

下面介绍常用的几种文件类型。

1. 普通文件

系统核心对这些数据没有进行结构化,只是作为有序的字节序列把它提交给应用程序,应用程序自己组织和解释这些数据,通常把它们归并为下述类型之一。

(1) 文本文件。由 ASCII 码字符构成。例如,信件、报告等。

(2) 数据文件。由来自应用程序的数据型和文本型数据构成。例如,Word 文档、电子表格、数据库等。

(3) 可执行的二进制文件。由机器指令和数据构成。例如,系统提供的一些命令。

2. 目录文件

目录文件是一类特殊的文件,利用它可以构成文件系统的分层树形结构。如同普通文件一样,目录文件也包含数据,不同的是,内核对这些数据加以结构化,它是由成对的目录"文件名/I 结点号/"构成的列表。

3. 链接文件

在 Linux 中,一个文件可以同时归属于多个不同的目录,相应的操作称为链接。被链接的文件可以存放在相同或不同的目录下。如果在同一目录下,两者必须有不同的文件名,如果在不同的目录下,那么被链接的文件可以与原文件同名。只要对一个目录下的该文件进行修改,就可以完成对所有目录下同名链接文件的修改。对于某文件的各个链接文件,可以给它们指定不同的存取权限,以增强安全性。

4. 设备文件

在 Linux 中,计算机上的硬件设备也与一种特殊类型的文件相对应,即所谓的设备文件。

Linux 下的设备文件均放在/dev 目录下,这些设备文件都是在安装过程中自动产生的。/dev 目录下的设备文件分为两大类:块设备文件和字符类型的设备文件。

5.7.2 Linux 文件目录

Linux 文件系统采用带链接的树形目录结构,整个文件系统有一个"根"(Root),然后在根上分"权"(Directory),任何一个分权上都可以再分权,权上也可以长出"叶子"。"根"和"权"在 Linux 中被称为是"目录"或"文件夹",而"叶子"则是一个个的文件。

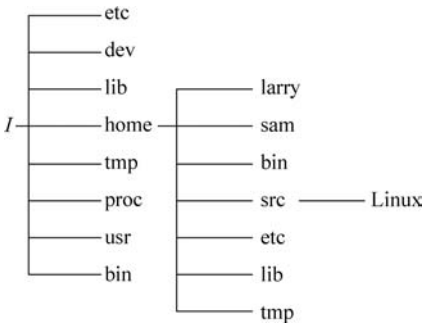

图 5-26 Linux 系统的树形目录结构

Linux 系统通过目录将系统中所有的文件分级、分层组织在一起,形成了 Linux 文件系统的树形目录结构。以根目录为起点,其他目录都由根目录派生而来。一个典型的 Linux 系统树形目录结构如图 5-26 所示,用户可以浏览整个系统,进入任何一个已授权进入的目录,并访问那里的文件。

Linux 目录提供了管理文件的方便途径。每个目录里面都包含文件。用户可以为自己的文件创建自己的目录，也可以把一个目录下的文件移动或复制到另一目录下，而且能移动整个目录，并和系统中的其他用户共享目录和文件。也就是说，能够方便地从一个目录切换到另一个目录，而且可以设置目录和文件的管理权限，以便允许或拒绝其他人对其进行访问。文件目录结构的相互关联性使共享数据变得十分容易，几个用户可以访问同一个文件，因此允许用户设置文件的共享程度。

需要说明的是，根目录是 Linux 系统中的特殊目录。Linux 是一个多用户系统，操作系统本身的驻留程序存放在以根目录开始的专用目录中，有时被指定为系统目录，图 5-26 中那些根目录下的目录就是系统目录。

如前所述，目录是 Linux 系统组织文件的一种特殊文件。为使用户更好地使用目录，我们介绍有关目录的一些基本概念。

1. 工作目录与用户主目录

从逻辑上讲，登录到 Linux 系统后，每时每刻都处在某个目录之中，此目录称作工作目录(Working Directory)或当前目录。工作目录可以随时改变。用户初始登录到系统中时，其主目录(Home Directory)称为工作目录。工作目录用“.”表示，其父目录用“..”表示。

用户主目录是系统管理员增加用户时建立的(以后也可以改变)，每个用户都有自己的主目录，不同用户的主目录一般互不相同。用户刚登录到系统时，其工作目录便是该用户主目录，通常与用户的登录名相同。用户可以通过一个“～”字符来引用自己的主目录。例如，当前用户主目录为/home/zhang，则 $ cat ～/class/software_1 和 $ cat /home/WANG/class/software_1 意义相同，将用户主目录名来替换“～”字符。目录层次建立好后，就可以把有关的文件放到相应的目录中，从而实现对文件的组织。

2. 根目录(/)

目录结构上的最高点称为根目录，它使用与超级用户相同的名称。单个字符斜杠(“/”)表示根目录。

Linux 系统中的其他目录都包含在根目录之下的层次结构中，这一点不同于 Windows 系统，Windows 系统中的每个驱动器被赋予了自己的字母及其自己的目录结构。在 Linux 中，系统上所有的存储设备都被装载到根目录之下的每个目录中，或者直接在根目录下，或者更下一层。

注意：/目录与 root 用户的主目录不是一回事，其主目录为/root，因此/root 目录是/的子目录。

3. Linux 子目录

(1) /bin。这个目录包含超级用户和一般用户使用的命令，这些命令提供一些操作，如复制、移动和删除文件，登录、创建和打开文件，识别系统名称，查看文本文件等。用户通常不会去改变/bin 目录的内容。

(2) /dev。/dev 目录包含设备文件和其他特殊文件。

(3) /etc。这个目录包含启动和正常运行 Linux 系统所需的配置文件,这些文件大多能够被编辑(通过配置工具或文本编辑器来完成)。大多数 Linux 集成套件提供了许多辅助软件用于配置/etc 目录中的文件,以便使用户更容易地使用 Linux。在安装过程中用户所回答的一些问题将自动填充到相关的/etc 目录文件中。

(4) /home。在典型情况下,这个目录拥有系统中每个用户的子目录。例如,如果 Mom、Dad、Erin 和 Matt 是系统中的所有用户,那么/home 目录可以包含四个用户目录:

/dad

/erin

/matt

/mom

如果系统中有大量用户,可以将它们分组放入部门子目录。有的 Linux 系统根本不使用/home 目录,并且将主目录放置在其他地方,但这种系统比较少见。

(5) /boot。这个目录包含系统启动所需的大多数文件,计算机启动时需要的其他文件存储在/etc 和/shin 目录中。

(6) /lib。这个目录包含了位于/bin 目录和/shin 目录中程序需要的库文件。一个库文件是一个程序文件,它包含能够被多个不同程序使用的代码。将这些共用代码以库的形式存放起来,可以减轻程序设计者的工作量。

(7) /proc。这个目录用于同 Linux 内核交换数据。该目录中有一些能够查看的文本文件,它们包含一些系统信息,如内核版本、系统正常工作时间和有关系统中处理器及内存的信息。

(8) /tmp。系统利用该目录存储暂存文件。不必计划在这里存储自己的暂存文件,程序将自动完成这一工作。

(9) /usr。/usr 目录包含系统中每个用户都使用的文件和程序。这里存放了随同 Linux 集成套件一起安装的大多数程序和实用工具,并且能够供普通账户(不仅仅是超级用户)使用。文件系统的层次结构规定了这个目录具有只读访问许可权,换句话说,用户不能改变/usr 目录中的内容。

(10) /mnt。/mnt 存放临时的映射文件系统,我们常把软驱和光驱挂装在这里的 floppy 和 cdrom 子目录下。

(11) /sbin。/sbin 存放系统管理程序。

(12) /var。/var 包含系统产生的经常变化的文件,如打印机、邮件、新闻等假脱机目录、日志文件、格式化后的手册页以及一些应用程序的数据文件等。

(13) /root。/root 是超级用户的主目录,归系统管理员所有。

5.7.3 虚拟文件系统

虚拟文件系统(Virtual File System,或称为 Virtual Filesystem Switch,VFS)是 Linux 内核中的一个软件层,用于给用户空间的程序提供文件系统接口,它也提供了内核中的一个抽象功能,允许不同的文件系统共存。

实现 Linux 虚拟文件系统要使得它对文件的访问尽可能快、尽可能高效,而且一定要确

保文件和数据的正确性，这两个要求彼此是不对称的。Linux VFS 在安装和使用每一个文件系统时都在内存中高速缓存信息。在文件和目录创建、写和删除时，这些高速缓存的数据被改动，必须非常小心才能正确地更新文件系统。如果能看到运行的核心中的文件系统的数据结构，就能看到文件系统读写数据块，描述正在访问的文件和目录的数据结构会被创建和破坏，同时设备驱动程序会不停地运转，获取和保存数据。这些高速缓存中最重要的是 Buffer Cache，它在文件系统访问底层的块设备时结合进来。当访问块时，它们被放到 Buffer Cache，根据状态的不同放在不同的队列中。Buffer Cache 不仅缓存数据缓冲区，也帮助管理块设备驱动程序的异步接口。

5.7.4 EXT2

Linux 支持许多不同的文件系统，第二扩展文件系统 EXT2 是 Linux 系统固有的，它运行稳定并且效率高，EXT2 和它的下一代操作系统 EXT3 已成为广泛使用的 Linux 文件系统。各种 Linux 的系统发布都将 EXT2 作为操作系统的基础。

1. EXT2 磁盘布局

EXT2 和其他逻辑块文件一样，由逻辑块序列组成，根据用途划分，这些逻辑块通常有引导块、超级块、inode 区及数据区等。

EXT2 将其所占的逻辑分区划分为块组，由一个引导块和其他块组组成，每个块组又由超级块、组描述符表、块位图、索引节点位图、索引节点表和数据区构成，如图 5-27 所示。

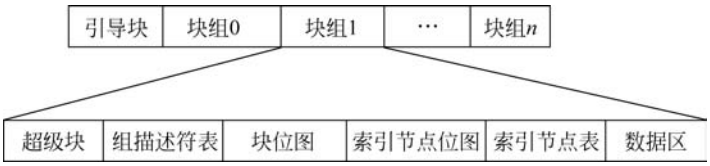

图 5-27 EXT2 磁盘布局在逻辑空间的映像

每个块中保存的这些信息是有关 EXT2 文件系统的备份信息。当某个块组的超级块或 inode 受损时，这些信息可以用来恢复文件系统。

2. EXT2 信息块

EXT2 文件系统中的数据是以数据块的方式存储在文件中的。这些数据块的大小相同，其大小在 EXT2 创建时设定。EXT2 用一个 inode 数据结构描述系统中的每一个文件，定义了系统的拓扑结构。一个 inode 描述了一个文件中的数据占用了哪些块以及文件的访问权限、文件的修改时间和文件的类型。EXT2 文件系统中的每一个文件都用一个 inode 描述，而每一个 inode 都用一个独一无二的数字标识。文件系统的 inode 都放在一起，在 inode 索引表中。

EXT2 块组中的组描述符中的项称为组描述符，用于描述某个块组的整体信息。每个块组都有一个相应的组描述符来描述它，所有的组描述符形成一个组描述符表并在使用时被调入块高速缓存。

EXT2 中每个块组有两个位示图块：一个用于表示数据块的使用情况，叫数据块位图；另一个用于表示索引节点的使用情况，叫索引节点位图。位图中的每一位表示该组中一个数据块或一个索引块的使用情况，用 0 表示空闲，用 1 表示已分配。

5.7.5 Linux 常用系统调用

1. 建立文件 create

当用户想把一批信息作为一个文件存放在磁盘上供以后使用时，可用此操作向系统提出建立文件的要求。

功能：创建一个新文件或重写一个已存在的文件。如果系统中不存在指明文件，核心便以给定的文件名和许可权限方式来创建一个新文件；如果系统中已有同名文件，核心将该文件的长度截短为 0。创建后的文件随之被打开，并返回其文件描述符 fd；若创建失败，则返回－1。

格式：

```
int create (char * pathname, int mode)
```

其中，pathname 是用户给予新文件的路径名，mode 为允许对文件进行访问的权限。mode 以二进制形式表示，取其低 9 位，每 3 位为一组，分别表示文件主、同组用户和其他用户对文件的读写、执行的权限。

2. 打开文件 open

当用户要使用一个已存在的文件时，首先要打开文件，建立用户与文件的联系，把涉及文件的有关目录信息复制到内存中，以便加速系统对文件的检索，为执行文件的读写等操作做好准备。

格式：

```
int open( char * pathname, int flags);
int open(char * pathname, int flags, mode_ t mode);
```

open 函数有两个形式。其中，pathname 是要打开的文件名(包含路径名称，默认时认为在当前路径下面)。flags 是读写标志，可以是下面的一个值或几个值的组合。

O_RDONLY：以只读方式打开文件。

O_WRONLY：以只写方式打开文件。

O_RDWR：以读写方式打开文件。

O_APPEND：以追加方式打开文件。

O_CREAT：创建一个文件。

O_EXEC：如果使用了 O_CREAT，而且文件已经存在，就会发生一个错误。

O_NOBLOCK：以非阻塞方式打开一个文件。

O_TRUNC：如果文件已经存在，则删除文件的内容。

前面三个标志只能使用其中任意一个。如果使用了 O_CREAT 标志，就要使用 open 的第二种形式。还要指定 mode 标志，用来表示文件的访问权限。mode 也可以用数字来代

表各位的标志。Linux 总共用五个数字来表示文件的各种权限。00000 中第一位表示设置用户 ID,第二位表示设置组 ID,第三位表示用户自己的权限位,第四位表示组的权限,最后一位表示其他人的权限。每个数字可以取 1(执行权限)、2(写权限)、4(读权限)、0(什么也没有)或是这几个值的和。

例如,要创建一个用户读写执行,组没有权限,其他人读执行的文件。设置用户 ID 位可以使用的模式是 1(设置用户 ID)0(组没有设置)7(1+2+4)0(没有权限,使用默认)5(1+4),即 10705。

```
open("temp",O_CREAT,10705);
```

如果打开文件成功,open 会返回一个文件描述符,以后对文件的所有操作就可以通过对这个文件描述符进行。

3. 关闭文件 close

当不再使用一个已打开的文件时应将文件关闭,以切断用户与该文件的联系。由于文件可被多个进程共享,故只有再无其他进程需要此文件时,对索引节点中的 i.count 进行减 1 后,若值为 0,才能关闭该文件。

格式:

```
int close(fd);
```

close 处理程序根据文件描述符 fd 找到进程打开文件表表项,从相应的用户文件描述符表项中获得它指向文件表项的指针 fp,再对该文件表项中的 f.count 和 i.count 做减 1 操作,若 f.count=0,则置该表项为空闲,若 i.count=0,则置该活动索引节点为空闲,把 fd 所指的进程打开文件表表项清 0。

4. 读文件 read

打开文件后,就要对文件进行读取了,可以调用函数 read()进行文件的读操作。

格式:

```
int read(int fd, void * buffer,nbytes)
```

其中,fd 是读操作的文件描述符,buffer 是所读的文件信息存放地址,nbytes 是要读的字节数。

对于普通的文件,read 从指定的文件中读取 nbytes 字节到 buffer 缓冲区中(必须提供一个足够大的缓冲区),同时返回 nbytes。

如果 read 读到了文件的结尾或被一个信号所中断,返回值会小于 nbytes,如果由信号中断引起返回,而且没有返回数据,read 会返回-1。当程序读到了文件结尾时,read 会返回 0。

5. 写文件 write

当用户要向文件增加记录时,可调用 write()函数执行写操作。

格式:

```
int write(int fd, void * buffer,nbytes)
```

其中,fd 是要进行写操作的文件描述符,buffer 是要写入文件内容的内存地址,nbytes 是要写的字节数。

write 从 buffer 中写 nbytes 字节到文件 fd 中,成功时返回实际所写的字节数,若出错则为-1。

其返回值通常与参数 nbytes 的值不同,否则表示出错。write 出错的一个常见原因是磁盘已写满,或者超过了对一个给定进程的文件长度限制。对于普通文件,写操作从文件的当前位移量处开始。如果在打开该文件时,指定了 O_APPEND 选择项,则在每次写操作之前,将文件位移量设置在文件的当前结尾处。在一次成功写之后,该文件位移量增加实际写的字节数。

6. lseek()函数

每个打开文件都有一个与其相关联的"当前文件位移量",它是一个非负整数,用以度量从文件开始处计算的字节数。通常,读写操作都从当前文件位移量处开始,并使位移量增加所读写的字节数。按系统默认,当打开一个文件时,除非指定 O_APPEND 选择项,否则该位移量被设置为 0。

可以调用 lseek()函数显式地定位一个打开文件。

格式:

```
off_t  lseek(int fd,off_t offset,int whence);
```

返回:若成功则返回新的文件位移,若出错则返回-1。

对参数 offset 的解释与参数 whence 的值有关。

若 whence 是 SEEK_SET,则将该文件的位移量设置为距文件开始处 offset 个字节。若 whence 是 SEEK_CUR,则将该文件的位移量设置为其当前值加 offset,offset 可为正或负。若 whence 是 SEEK_END,则将该文件的位移量设置为文件长度加 offset,offset 可为正或负。

若 lseek 成功执行,则返回新的文件位移量,为此可以用下列方式确定一个打开文件的当前位移量:

```
off_t currpos;
currpos = lseek(fd,0,SEEK_CUR);
```

这种方法也可用来确定所涉及的文件是否可以设置位移量。如果文件描述符引用的是一个管道或 FIFO,则 lseek 返回-1,并将 errno 设置为 EPIPE。

本章小结

本章介绍了文件系统功能及其实现。文件是存储在外部介质上的一组相关记录的集合。文件系统是操作系统中负责存取和管理文件信息的机构,同时对用户提供文件系统接口。

文件的逻辑结构分为两种:记录式文件和字符流式文件。

文件的物理结构是文件在外存上的存储组织形式,主要表达文件信息在磁盘上存储的

方式。文件的物理组织形式有连续文件结构、链接文件结构、索引文件结构、多重索引文件结构以及混合索引文件结构。连续文件结构是把逻辑文件中的记录顺序地存储到连续的物理盘块中，它的优点是管理简单，顺序存取速度快；缺点是不利于文件的动态增长，容易产生磁盘碎片。链接文件结构是以分配给文件的磁盘块为节点，形成链表结构，因此文件存储在磁盘上不需要占用连续的磁盘空间。它的优点是消除了磁盘碎片，文件的动态增长、删除、修改也非常方便。索引结构是另一种非连续分配的文件存储结构，通过索引表实现逻辑地址与物理地址的映射关系。多重索引文件结构可以缩短索引表的长度，但会增加访问磁盘的次数。在 Linux 系统中文件使用混合索引文件的方式，这种方式对于短文件实现直接索引，访问速度较快，同时也支持长文件。

文件目录是用来组织文件和检索文件的关键数据结构。一个文件目录项的构成可以是如下形式：①用文件名＋文件控制块 PCB，如 DOS 操作系统的文件目录项；②用文件名＋索引节点指针，如 Linux 的文件目录项。两种目录项的构成各有特点。第二种结构文件的检索速度更快，目录文件更小。文件目录有单级、二级及树形目录等形式。在 Linux 系统中采用的是有向非循环图结构，它利于实现对文件或目录的共享。

文件系统实现了对空闲磁盘块的管理。当创建文件或扩充文件时，需要申请磁盘空间；删除文件时需要回收磁盘空间。磁盘块的组织管理方式有空闲空间表法、空闲块链、位示图、空闲块成组链接法。在 Linux 系统中采用的就是空闲块成组链接法。

文件系统实现了文件的共享和安全性。实现共享常用的方法有绕弯路法和链接法。对文件存取控制是和文件共享、保护紧密相连的。存取控制方法有口令、密码、存取控制表、存取控制矩阵等。

习题 5

5-1 什么是文件和文件系统？文件系统有哪些功能？

5-2 根据对文件的管理方式，Linux 文件可以分为哪几类？

5-3 什么是文件的逻辑结构？什么是文件的物理结构？

5-4 操作系统对文件的存取有哪两种基本方式？各有什么特点？

5-5 文件系统对目录管理的主要需求是什么？

5-6 什么是文件目录？文件目录的内容是什么？

5-7 在 Linux 系统中，为什么要将文件的控制信息从目录中分离出来，单独构成一个 I 节点？

5-8 Linux 文件系统采用怎样的物理结构？有什么优点和缺点？

5-9 文件存储空间管理有哪几种？各自有什么特点？Linux 系统的存储空间采用什么方法？

5-10 说明在 Linux 系统中，如何由文件的逻辑块号 n 找到文件的物理块号。

5-11 在 Linux 系统中，一个盘块大小为 1KB，每个盘块号占 4B，则一个进程要访问一个相对于文件开始的偏移量为 2 631 68B 处的数据时，试计算应直接访问还是索引访问？几级索引？

5-12 基于索引节点的文件共享方式有什么优点？

5-13 说明 Linux 的 EXT2 文件系统磁盘的结构及各部分的功能。

5-14 什么是“打开文件”操作？什么是“关闭文件”操作？引入这两个操作的目的是什么？

5-15 文件 A 是连续文件，它由 4 个逻辑记录构成，假设每个逻辑记录的大小与磁盘块大小相等，均为 1KB。若第一个逻辑记录存放在第 100 号磁盘块上，试说明文件 A 在磁盘上存放的全部磁盘块。

5-16 文件 B 是串联文件，它由 4 个逻辑记录构成，假设每个逻辑记录的大小与磁盘块大小相等，均为 1KB。这 4 个逻辑记录分别存放在第 100、157、66、67 号磁盘块上，回答下列问题：

(1) 画出此串联文件的结构。

(2) 若要读文件 B 第 3120B 处的信息，需要访问的是哪个磁盘块？

(2) 若要读文件 B 第 3120B 处的信息，需要访问几次磁盘？

5-17 文件 C 是直接索引文件，它由 4 个逻辑记录构成(R0～R3)，假设每个逻辑记录的大小与磁盘块大小相等，均为 1KB。这 4 个逻辑记录分别存放在第 280、572、96、169 号磁盘块上，试画出此索引文件的结构。如果要访问第 R2 条记录时，需要访问几次磁盘？