第15章 文 件 系 统

本章介绍有关Linux文件系统的有关内容。

15.1 Linux文件系统概述

Linux系统的一个重要特征就是支持多种不同的文件系统。这样，Linux系统就十分的灵活，并且可以十分容易地和其他操作系统共存。目前， Linux系统支持大约15个文件系统：EXT、EXT2、 XIA、MINIX、UMSDOS、MSDOS、VFAT、 PROC、SMB、NCP、ISO9660、SYSV、HPFS、AFFS 和 UFS。并且，毫无疑问，Linux系统支持的文件系统还会增加。

在Linux系统中，每一个单独的文件系统都是代表整个系统的树状结构的一部分。当挂接一个新的文件系统时， Linux把它添加到这个树状的文件系统中。所有系统中的文件系统，不管是什么类型，都挂接到一个目录下，并隐藏掉目录中原有的内容。这个目录叫做挂接目录或者挂接点。当文件系统卸载掉时，目录中的原有内容将再一次的显示出来。

初始化时磁盘将被划分成几个逻辑分区。每一个逻辑分区可以使用一种文件系统，例如EXT2文件系统。文件系统把存储在物理驱动器中的文件组织成一个树状的目录结构。可以存储文件的设备称为块设备。 Linux文件系统把这些块设备当作简单的线形块的集合，而不管物理磁盘的结构如何，而将读写某一个设备块的请求转换成特定的磁道、扇区和柱面是通过设备的驱动程序实现的。因此，不同的设备控制器控制的不同设备中的不同文件系统在 Linux中都可以同样地使用。文件系统甚至可以不在当地的系统中，也就是说，文件系统可以通过网络远程连接到本地磁盘上。请看下面的例子，这是一个Linux在SCSI 磁盘上的根文件系统：

```
A       E       boot    etc     lib     opt         tmp     usr
C       F       cdrom   fd      proc    root        var     sbin
D       bin     dev     home    mnt     lost+found
```

在这里，用户和使用文件的程序都不必知道 /C实际上是挂接的VFAT文件系统，而VFAT文件系统本身却存储在系统中的一个 IDE硬盘上。同样， /E是系统中第二个IDE控制器控制的主硬盘系统。也可以使用调制解调器将远程的文件系统挂接到 /mnt/remote目录下。

一个文件系统中不仅包括含有数据的文件，而且还存储着文件系统的结构。文件系统中的信息必须是安全和保密的。

Linux系统中的第一个文件系统是 Minux，但它的文件名只能有14个字符，最大的文件长度是64M字节。所以， 1992年4月引进了第一个专门为Linux设计的文件系统 —— ext(extended file system)。但ext的功能还是有限。最后在1993年又推出了一个新的文件系统 —— ext2。

当Linux引进ext文件系统时有了一个重大的改进：真正的文件系统从操作系统和系统服务中分离出来，在它们之间使用了一个接口层 —— 虚拟文件系统VFS（Virtual File system）。

VFS允许Linux支持多种不同的文件系统，每个文件系统都要提供给 VFS一个相同的接口。这样所有的文件系统对系统内核和系统中的程序来说看起来都是相同的。 Linux系统中的VFS层使得你可以同时在系统中透明地挂接很多不同的文件系统。

VFS能高速度及高效率地存取系统中的文件，同时它还得确保文件和数据的正确性。这两个目标有时可能相互矛盾。 VFS在每个文件系统挂接和使用时把文件系统的有关信息暂时保存

在内存中，所以当内存中的信息改变时，例如创建、写入和删除目录和文件时，系统要保证正确地升级文件系统中相关的内容。如果你了解在系统内核中运行的文件系统的数据结构，那么你也就了解了文件系统读取的数据块的情况。系统缓存中最重要的就是缓冲区缓存，它被集成到每个单独的文件系统存取它们的块设备所使用的方法中。每当系统存取数据块时，数据块都被放入缓冲区缓存中，并且依照它们的状态保存到各种各样的队列中。缓冲区缓存中不仅保存了数据缓冲区，而且还可以帮助管理和块设备驱动程序之间的异步接口。

15.2 ext2文件系统

ext2的物理结构图参见图15-1。

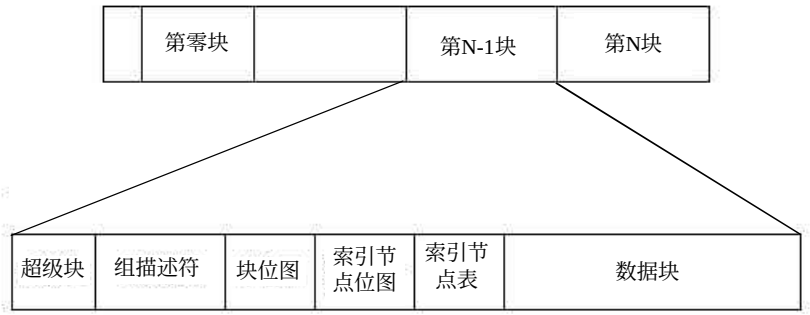

图15-1 文件使用ext2示意图

ext2文件系统是Linux系统中最为成功的文件系统，各种 Linux的系统发布都将ext2文件系统作为操作系统的基础。

ext2文件系统中的数据是以数据块的方式存储在文件中的。这些数据块具有同样的大小，并且其大小可以在ext2创建时设定。每一个文件的长度都要补足到块的整数倍。例如，如果一个块的大小是1024字节，那么一个1025字节大小的文件则占用两个数据块。所以，平均来说一个文件将浪费半个数据块的空间。但这样可以减轻系统中 CPU的负担。文件系统中不是所有的数据块都存储数据，一些数据块用来存储一些描述文件系统结构的信息。 ext2通过使用索引节点（inode）数据结构来描述系统中的每一个文件。索引节点描述了文件中的数据占用了哪一个数据块以及文件的存取权限、文件的修改时间和文件类型等信息。 ext2文件系统中的每一个文件都只有一个索引节点，而每一个索引节点都有一个唯一的标识符。文件系统中的所有的索引节点都保存在索引节点表中。 ext2中的目录只是一些简单的特殊文件，这些文件中包含指向目录入口的索引节点的指针。

对于一个文件系统来说，某一个块设备只是一系列可以读写的数据块。文件系统无须关心数据块在设备中的具体位置，这是设备驱动程序的工作。每当文件系统需要从块设备中读取数据时，它就要求设备驱动程序读取整数数目的数据块。 ext2文件系统将它所占用的设备的逻辑分区分成了数据块组。每一个数据块组都包含一些有关整个文件系统的信息以及真正的文件和目录的数据块。

15.2.1 ext2的索引节点

ext2的索引节点图参见图15-2。

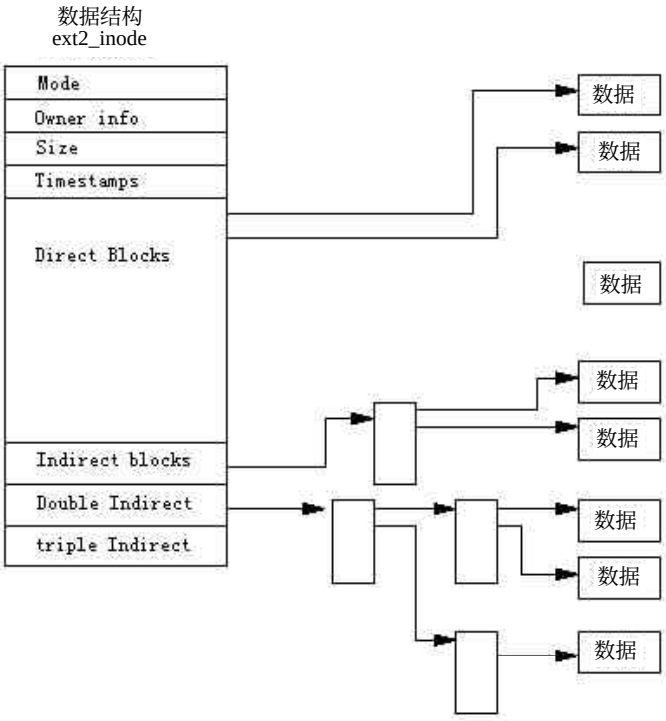

图15-2 EXT2 系统索引节点示意图

在ext2文件系统中索引节点是一切的基础，文件系统中的每一个文件和目录都使用一个唯一的索引节点。每一个数据块组中的索引节点都保存在索引节点表中。数据块组中还有一个索引节点位图，它用来记录系统中已分配和未分配的索引节点。下面是 ext2的索引节点的一些主要的字段：

1. mode

这里保存两个信息：一个是此索引节点描述的是什么，另一个是用户拥有的权限。例如，对于ext2，一个索引节点可以描述文件、目录、符号连接、块设备、字符设备以及 FIFO结构。

2. Owner Information

这是文件或目录所有者的用户和组标识符。这使得文件系统可以正确地授权某种存取操作。

3. Size

文件的字节大小。

4. Timestamps

索引节点建立的时间和索引节点最后修改的时间。

5. Datablocks

指向存储此索引节点描述文件的数据块的指针。前十二个指针是指向存储数据的物理数据块的指针，而后三个指针则包括不同级别的间接指针。例如，两级指针指向一个指向其他指针块的指针块。这意味着小于或者等于 12个数据块的文件的存取速度要高于多于12个数据块的文件。

你应该注意到ext2的索引节点可以描述一些特殊的设备文件。这些设备文件不是真正的文

件，但系统中的程序可以使用这些设备文件来存取它们相关的设备。所有的这些设备文件都在/dev目录下面。例如，挂接程序可以把它希望挂接的设备文件作为它的一个参数。

15.2.2 ext2超级块

超级块（Superblock）存储着描述文件系统的大小和形状的基本信息。文件系统的管理员可以使用其中的信息来使用和维护文件系统。一般情况下，当文件系统挂接时，系统只读取数据块组0中的超级块，但每一个数据块组中都包含一个超级块的副本，以防系统崩溃时使用。超级块包括如下的主要信息：

1. Magic Number（幻数）

使挂接程序确认这是ext2文件系统的超级块。目前其值为0xEF53。

2. Revision Level（修订级别）

这是文件系统的主版本号和从版本号。挂接程序可以根据此信息决定此文件系统是否支持一些特定文件系统的函数。

3. Mount Count（挂接数）和 Maximum Mount Count（最大挂接数）

系统用来决定文件系统是否应该全面地检查。文件系统每挂接一次， mount count的值就会加1。当mount count的值和maximum mount count的值相等时，系统将显示maximal mount count reached, running e2fsck is recommended信息，提示用户进行文件系统的检查。

4. Block Group Number（块组号）

包含此超级块的数据块组号。

5. Block Size（块大小）

文件系统中数据块的大小，例如1024字节。

6. Blocks per Group（每组块数）

数据块组中的数据块数目和Block Size一样，它在文件系统创建以后就是固定的了。

7. Free Blocks（空闲块）

文件系统中空闲的数据块的数目。

8. Free Inodes（空闲索引节点）

文件系统中空闲的索引节点的数目。

9. First Inode（第一个索引节点）

文件系统中的第一个索引节点号。在一个ext2根文件系统中，第一个索引节点是/目录的入口。

15.2.3 ext2 数据块组描述符

每一个数据块组都有一个描述它的数据结构和超级块一样，在每一个数据块组中都要复制一份数据块组描述符。

数据块组描述符包含以下的信息：

1. Blocks Bitmap（块位图）

数据块组中数据块分配位图所占的数据块数。在数据块分配和数据块撤消时使用。

2. Inode Bitmap（索引节点位图）

数据块组中索引节点分配位图所占的数据块数。在索引节点的分配和撤消中使用。

3. Inode Table（索引节点表）

数据块组中索引节点表所占的数据块数。系统中的每一个索引节点都由一个数据结构来描述。

4. Free blocks count（空闲块数）、Free Inodes count（空闲索引节点数）, Used directory count（已用目录数）

这是空闲的数据块数，空闲的索引节点数和已用的目录数。

数据块组描述符组成一个数据块组描述符表。每一个数据块组在其超级块之后都包含一个数据块组描述符表的副本。 EXT2文件系统事实上只是使用数据块组 0中的数据块描述符表。

15.2.4 ext2 中的目录

在ext2文件系统中，目录是一些特殊的文件，它们用来创建和保存系统中文件的存取路径。图15-3是一个目录入口在内存中的结构图。

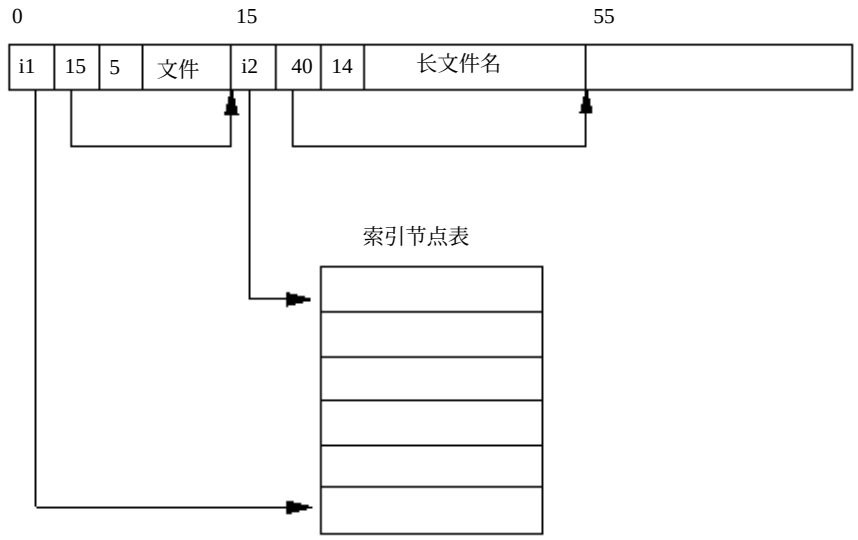

图15-3 EXT2文件系统目录结构示意图

一个目录文件包括很多的目录入口，一个目录入口包括以下的内容：

1. inode（索引节点）

目录入口的索引节点。这是保存在数据块组中的索引节点表数组的索引值。在上图中，file的目录入口的索引节点号的引用是i1。

2. name length（名称长度）

目录入口的字节长度。

3. name（名称）

目录入口的名字。

每一个目录的头两个入口都是标准的 . 和 .., 分别代表本目录和父目录。

15.2.5 在ext2 文件系统中查找文件

Linux系统的文件名格式和 Unix系统的文件名格式一样，其中的目录名用斜杠（ /）分隔。

例如，文件名/home/zws/.cshrc，其中/home 和 /zws是目录名，.cshrc则是文件名。Linux系统中的文件名可以由任何可打印的字符组成，也可以是任何的长度。系统通过分析目录中的文件，来查找文件对应的索引节点。

系统需要的第一个索引节点是文件系统根目录的索引节点，它的值保存在文件系统的超级块中。要读取一个ext2文件系统的索引节点，我们必须在相应的数据块组中的索引节点表中查找。例如，如果一个根目录的索引节点值是 42，那么我们需要读取在数据块组 0中的索引节点表中的第42个索引节点。根目录索引节点是一个 ext2的目录节点，也就是说，根目录索引节点的模式（mode）是目录，而它的数据块中包含的是ext2目录的入口。

home目录只是众多目录入口中的一个，我们可以从中查找出描述 /home/zws目录的索引节点值。接下来我们读取这个目录（首先读取它的索引节点，然后读取此索引节点描述的数据块中的zws目录），从中查找描述/home/zws目录的索引节点值。最后，在描述/home/zws目录的索引节点指向的目录入口中找到 .cshrc文件的索引节点值，通过它的索引节点值找到存储在文件中的数据的数据块。

15.2.6 改变ext2 文件系统中文件的大小

文件系统的一个常见的问题就是文件碎片问题。存储文件数据的数据块可能散布在整个文件系统中，这样顺序存取一个文件的数据块可能变得效率越来越低。 ext2文件系统通过为一个文件的新数据块分配靠近当前数据块的物理块或同一个数据块组中的数据块来解决文件碎片问题。只有当这样的分配策略不能实现时， ext2才为新的文件数据块分配其他数据块组中的数据块。

当进程往文件中写入数据时， ext2文件系统都要检查数据是否已经超出了文件最后分配的数据块的范围。如果写入的数据超出了该范围，那么文件系统必须为此文件分配一个新的数据块。在这之前，进程不能继续运行。 ext2文件系统的数据块分配过程所做的第一件事就是锁定此文件系统的 ext2超级块。分配和撤消数据块需要改变超级块中字段的内容，并且Linux文件系统不允许多个进程同时修改超级块。如果另外的进程在此时也需要分配新的数据块，那么它必须等待直到正在运行的进程处理完对超级块的修改并释放对超级块的控制才能恢复运行。对超级块的操作本着先来先服务的策略。在锁定了超级块以后，进程将检查文件系统中有无足够的空闲数据块。如果没有，那么进程的分配请求将失败，进程将放弃对文件系统超级块的控制权。如果系统中有足够的空闲数据块，则进程将为文件分配所需要的数据块。

如果ext2文件系统内建有预分配的数据块，那么我们可以使用预分配数据块。预分配数据块实际上并不存在，它们保留在已分配的数据块位图中。申请分配新数据块的文件的 VFS索引节点有两个特别的字段：prealloc_block 和 prealloc_count，分别是第一个预分配数据块的块号和可以使用的预分配块数。如果没有预分配数据块或者系统中没有预分配数据块功能， ext2文件系统必须分配一个新的数据块。它首先查看文件中最后一个数据块的后一个数据块是否空闲。逻辑上，这个数据块是最为有效的数据块，因为这样可以加速对文件顺序的存取。如果此数据块没有空闲，那么文件系统将加大搜索的范围，在 64个数据块的范围内查找理想的数据块。

如果甚至连这样的空闲数据块找不到，进程则开始查看其他数据块组直到找到一个空闲的数据块为止。数据块分配程序查找某一个数据块组中的一簇，也就是 8个空闲的数据块。如果无法找到8个连续的空闲数据块，数据块分配程序将查找较少的连续的空闲数据块。

无论在那一个数据块组中查找到空闲的数据块，数据块分配程序都将更新数据块组中的数据块位图，并且在缓冲区缓存中分配一个数据缓冲区。这个分配的数据缓冲区由设备标识符和已分配的数据块的块号来标识。最后将超级块标记为 dirty来表明超级块已经改动，同时文件系统将超级块解锁。此时，如果有任何进程等待使用超级块，那么系统将允许等待队列中的第一个进程控制超级块继续运行。进程的数据被写入到新分配的数据块中。如果新分配的数据块无法容纳全部的数据，则整个分配过程将重复进行。

15.3 VFS

图15-4显示了Linux系统内核中的VFS和实际文件系统之间的关系。VFS必须管理同时挂接在系统上的不同的文件系统。它通过使用描述整个 VFS的数据结构和描述实际挂接的文件系统的数据结构来管理这些不同的文件系统。

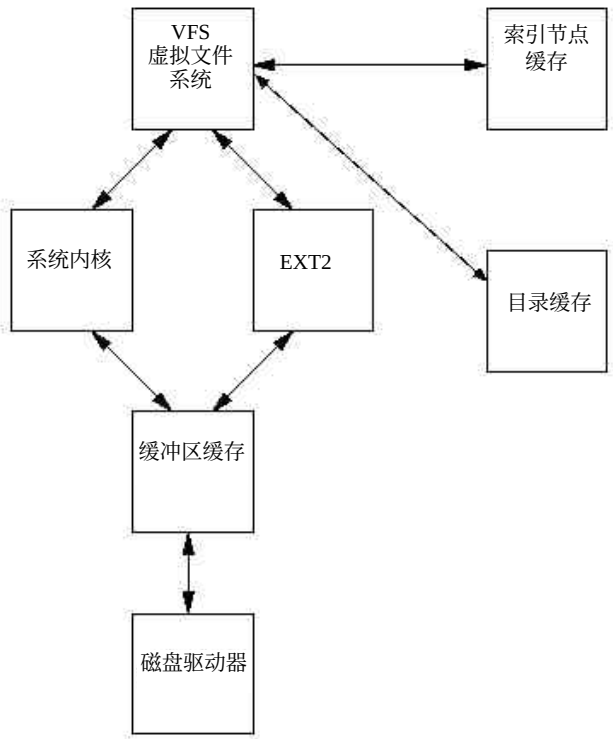

图15-4 VFS系统结构示意图

令人易于混淆的是，VFS和ext2文件系统一样使用超级块和索引节点来描述系统中的文件。和ext2中的索引节点一样，VFS的索引节点用来描述系统中的文件和目录。

当一个文件系统初始化时，它将在 VFS中登记。这些过程在系统启动操作系统初始化时完成。实际的文件系统或者内建到系统内核中，或者作为可装入模块在需要时装入。当一个基于块设备的文件系统挂接时，当然也包括根文件系统， VFS首先要读取它的超级块。每一个文件系统的超级块读取程序都必须先清楚文件系统的结构，然后把有关的信息添加到 VFS的超级块数据结构中。VFS中保存了系统中挂接的文件系统的链表以及这些文件系统对应的 VFS超级块。每一个VFS超级块都包含一些信息和指向一些执行特别功能的子程序的指针。例如，代表挂接的ext2文件系统的VFS超级块包含了一个指向ext2索引节点读取程序的指针。这个ext2索引节点

读取程序，和其他文件系统索引节点的读取程序一样，把信息添加到VFS索引节点中的字段中。每一个 VFS超级块都包含了一个指向文件系统中第一个VFS索引节点的指针。对于根文件系统来说，第一个VFS索引节点是代表/目录的索引节点。这种对应关系对于ext2文件系统来说十分有效，但对于其他的文件系统则效果一般。

当系统中的进程存取目录和文件时，需要调用系统中的子程序来遍历搜索系统中的 VFS索引节点。

例如，键入ls或者cat命令将导致VFS搜索整个文件系统中的VFS索引节点。因为系统中的每一个文件和目录都由一个索引节点来表示，那么有些索引节点将经常会被重复地搜索。这些索引节点将保存在索引节点缓存中，这样将增快以后的存取速度。如果要找的索引节点不在索引节点缓存中，那么进程将调用一个特殊的系统程序来读取相应的索引节点。读取了索引节点以后，此索引节点将放在索引节点缓存中。最少用到的索引节点将被交换出索引节点缓存。

所有的Linux系统中的文件系统都使用一个相同的缓冲区缓存来保存相应的块设备中的数据缓冲区，这样可以加速所有的文件系统对其物理设备的存取。

这个缓冲区缓存和 Linux系统中的各个不同的文件系统无关，并且缓冲区缓存集成到了Linux内核用来分配和读写数据缓冲区的机制中。把 Linux中的各个文件系统和其相应的物理设备分开是有很大好处的。所有的块设备都在 Linux系统内核中注册，并且提供一个相同的，以数据块为基础的，一般情况下是异步的接口。当实际文件系统从相应的物理设备中读取数据时，将导致文件系统控制的物理设备读取它的物理块。当文件系统读取数据块时，它们把数据块保存在所有文件系统和系统内核共同使用的公共的缓冲区缓存中。缓冲区缓存中的缓冲区以它们的数据块号和读取设备的标识符名来区分。所以，如果需要一些相同的数据，那么这些数据将从系统的缓冲区缓存而不是磁盘中读出，这就加快了读取的速度。 VFS中也保存了一个目录查找缓存，一些经常使用的目录的索引节点将会很快地找到。目录缓存中并不保存目录的索引节点本身，索引节点保存在索引节点缓存中，目录缓存只是简单地保存目录的名字和目录的索引节点号的对应关系。

15.3.1 VFS 超级块

每一个挂接的文件系统都有一个VFS超级块， VFS超级块包括以下主要信息：

1. 设备

这是存储文件系统的物理块设备的设备标识符。例如，系统中第一个 IDE磁盘/dev/hda1的标识符为0x301。

2. 索引节点指针

挂接的（mounted）索引节点指针指向文件系统的第一个索引节点。覆盖的（ covered）索引节点指针指向文件系统挂接的目录的索引节点。根目录文件系统的 VFS超级块中没有覆盖的索引节点指针。

3. 数据块大小

文件系统数据块的字节数，例如， 1024字节。

4. 超级块操作

指向文件系统的一系列超级块子过程的指针。 VFS可以使用这些子过程读写索引节点和超级块。

5. 文件系统类型

指向挂接的文件系统的file_system_type数据结构的指针。

6. 文件系统的特殊的信息

指向文件系统所需要的信息的指针。

15.3.2 VFS 索引节点

像EXT2文件系统一样，VFS 中每一个文件和目录都有一个且仅有一个VFS索引节点。

每一个VFS索引节点中的信息都是文件系统的一些特殊的子过程根据相应的文件系统的信息产生的。VFS索引节点只在系统需要时才保存在系统内核的内存及 VFS索引节点缓存中。它包括以下主要内容：

1. 设备

这是索引节点代表的文件或目录所在的设备的设备标识符。

2. 索引节点号

索引节点号在文件系统中是唯一的。索引节点号加上设备号在 VFS中是唯一的。

3. 模式

用来描述VFS索引节点代表的是文件、目录或其他内容，以及对它的存取权限。

4. 用户的标识符

表示用户的标识符。

5. 时间

创建、修改和写入的时间。

6. 数据块大小

文件数据块的大小，例如，1024字节。

7. 索引节点操作

一个指向一系列子程序地址的指针。这些子程序和相应的文件系统有关，它们执行有关此索引节点的各种操作，例如，截断此索引节点代表的文件。

8. 计数器

正在使用VFS索引节点的系统进程。

9. 锁定

用于锁定VFS索引节点，例如，当文件系统读取索引节点的时候。

10. 已修改（dirty）

指示索引节点是否已经被修改了，如果已修改，则相应的文件系统也需要修改。

11. 文件系统的一些特殊的信息

指向文件系统所需要的信息的指针。

15.3.3 登记文件系统

构建Linux系统内核时，需要选择文件系统。当系统内核建好以后，文件系统的起始程序将包括调用系统中所有文件系统的初始化程序的调用。

Linux文件系统也可以作为系统的模块在需要时装入，或者使用 insmod命令手工地装入。每当一个文件系统装入时，它都要在系统内核中登记。同样，当文件系统卸载时，也要从系统内核中取消登记。每一个文件系统的初始化程序都在 VFS中登记，并且创建一个file_system_type数据结构，其中包括文件系统的名字和指向VFS超级块读取程序地址的指针。

图15-5显示所有file_system_type 结构组成的一个由file_systems 指针指向的链表。每一个

file_system_type 结构都包括以下的信息：

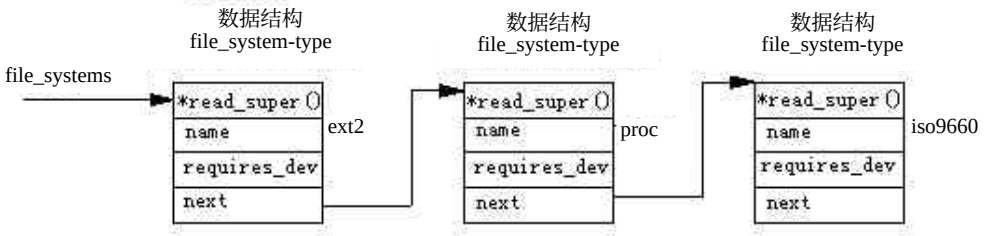

图15-5 文件系统结构示意图

1. 超级块读取程序

当一个文件系统挂接时，VFS使用此程序读取文件系统的超级块。

2. 文件系统名

文件系统的名称，例如ext2。

3. 需要的设备

指出文件系统是否需要设备的支持。并不是所有的文件系统都需要设备的支持，例如，/proc就不需要块设备的支持。

你可以通过查看/proc/filesystems了解系统中已注册的文件系统，例如：

```
ext2
nodev          proc
iso9660
```

15.3.4 挂接文件系统

当一个超级用户试图挂接一个文件系统时，Linux系统内核必须首先检查系统调用需要使用的参数的有效性。虽然挂接时要做一些基本的检查，但它并不知道此系统内核中已经支持的文件系统，也不知道文件系统的挂接点是否存在。请看下面这个命令：

```
$ mount -t iso9660 -o ro /dev/cdrom /mnt/cdrom
```

这个mount命令将向内核传递三个参数：文件名、包含文件系统的物理设备和文件系统的挂接点。

VFS所要做的第一件事就是找到要挂接的文件系统。

VFS首先通过查找file_systems指针指向的链表中的每一个file_system_type数据结构来搜索已知的文件系统。

如果VFS找到了一个匹配的名字，那么说明系统内核支持此种文件系统。这样可以得到读取文件系统的超级块的程序的地址。如果没有找到相应的名字，但内核可以装入文件系统模块，那么系统内核将要求内核守护进程装入相应的文件系统模块。

接下来如果mount命令中的物理设备没有挂接到系统中，那么 VFS必须查找作为新文件系统挂接点的目录的VFS索引节点。一旦找到索引节点，VFS将检查此目录下有无其他挂接的文件系统。同一个目录下不能挂接多个文件系统。

VFS挂接程序必须分配一个VFS超级块，并且传递给它一些有关文件系统挂接的信息。系统中所有的VFS超级块都在由super_block数据结构组成的super_blocks数组中。超级块读取程序使用它从物理设备中读取的信息来填充 VFS超级块的有关字段。对于ext2文件系统，这个映射和翻译信息的过程十分的容易，它只是简单地把 ext2的超级块转换成VFS的超级块。对于其

他的文件系统，例如msdos文件系统，这个过程将较为复杂。无论是那一种的文件系统，创建VFS超级块都意味着文件系统必须从存储该文件系统的块设备中读取有关的信息。如果该块设备无法读取，或者该块设备不包括此文件系统， mount 命令都将失败。

一个挂接的文件系统参见图15-6。

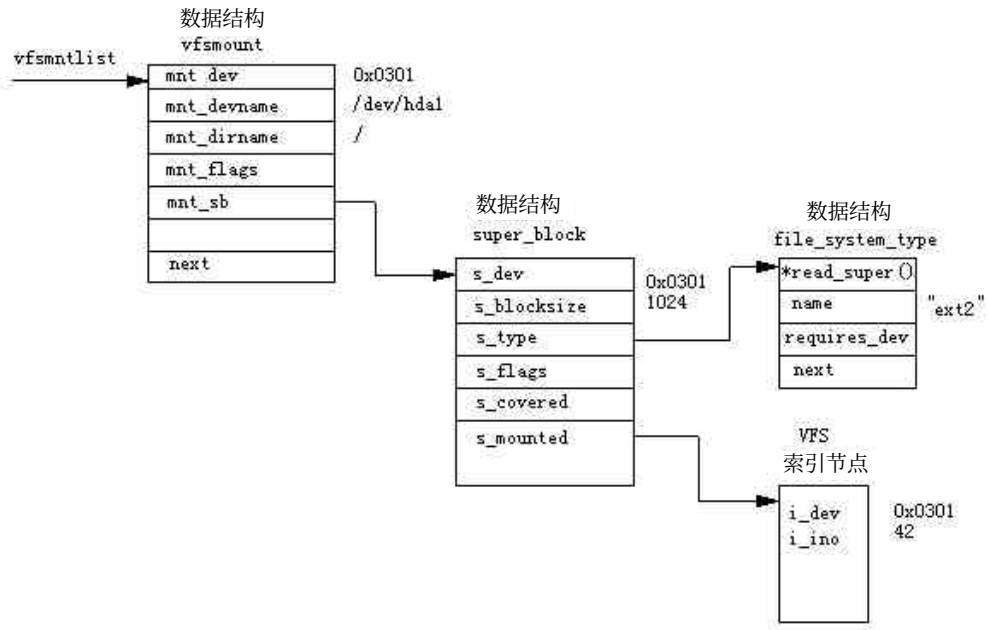

图15-6 虚拟文件系统挂接结构示意图

每一个挂接的文件系统都使用一个vfsmount数据结构，这些vfsmount结构组成了一个由指针vfsmntlist指向的链表。另一个指针 vfsmnttail指向链表的最后一个入口，并且还有一个mru_vfsmnt指针指向链表中最近被使用的文件系统。每一个 vfsmount结构都包括存储文件系统的块设备的设备号，文件系统挂接的目录和一个指向文件系统的 VFS超级块的指针。而VFS超级块中则包括指向此文件系统的file_system_type数据结构的指针和指向此文件系统根索引节点的指针。当文件系统装入以后，此根索引节点就一直保存在 VFS索引节点缓存中。

15.3.5 在VFS中查找文件

如果希望在VFS中查找一个文件的VFS索引节点， VFS必须一个个解释文件路径名中的中间目录名，查找中间目录名的索引节点。每一个目录名的查找都要调用文件系统的查找子程序，这个子程序的地址保存在该目录的父目录的 VFS索引节点中。每当文件系统查找一个索引节点时，它都要在目录缓存中查找该目录。如果目录缓存中没有该目录，它将从文件系统中或索引节点缓存中调入此目录的索引节点。

15.3.6 撤消文件系统

撤消文件系统基本上与挂接文件系统的过程相反。

如果文件系统的某一个文件正在被使用，那么该文件系统就不能被撤消。例如，如果系统中的一个进程正在使用/mnt/cdrom目录或者它的一个子目录，那么你就不能撤消此目录。如果

有进程正在使用将要被撤消的文件系统，那么此文件系统的某一个 VFS索引节点可能正在VFS索引节点缓存中。检查程序可以通过查找索引节点链表来确定有无该文件系统的索引节点。如果此挂接文件系统的VFS超级块已经被改写，那么VFS超级块就必须写回到磁盘的文件系统中。写回完成以后， VFS超级块占用的内存就可以释放。最后这个文件系统的 vfsmount结构从vfsmntlist链表中断开。

15.3.7 VFS 索引节点缓存

当查找挂接的文件系统时，文件系统中的索引节点将被不断的读取。 VFS提供的索引节点缓存可以加快对系统中所有挂接的文件系统的存取。

VFS索引节点缓存使用散列表实现，表的入口是指向具有相同 hash值的VFS索引节点链表的指针。索引节点的hash值是通过索引节点号和存储文件系统的物理设备号计算出来的。每当VFS需要存取一个索引节点时，它将首先查找索引节点缓存。为了在索引节点缓存中找到一个特定的索引节点，系统首先需计算索引节点的 hash值，然后使用hash值作为索引节点的散列表的索引。这样就可以找到一个指向具有相同 hash值的索引节点链表的指针。系统接着逐个读取链表中的每一个索引节点直到找到一个索引节点号和设备标识符都相同的索引节点为止。

如果系统可以在索引节点缓存中找到索引节点，那么此索引节点的计数器将加 1，表明又有另一个进程在使用该索引节点。否则，系统必须找到一个空闲的 VFS索引节点，这样系统才可以从内存中读取索引节点。 VFS有几种方法可以获得一个空闲的索引节点。如果系统可以分配更多的索引节点，那么系统将分配新的索引节点。系统把新分配的内存页面分隔成一些新的、空闲的索引节点，然后把这些新的索引节点放到索引节点链表中。系统中的所有的 VFS索引节点都放在一个由指针 first_inode指向的链表中，当然也存放在索引节点散列表中。如果系统已经不能分配新的索引节点了，那么它必须查找一个已用的索引节点以便重新分配。可以重新分配的索引节点是那些用户计数器为 0的索引节点，这说明系统中没有任何进程正在使用这些索引节点。一些非常重要的索引节点，例如文件系统的根目录索引节点，它们的索引节点计数器总是大于0，这样它们永远不能被重新分配。一旦找到一个可以重新分配的索引节点，它将被清除。如果此索引节点被修改过，则必须写回到文件系统中。如果它被锁定，则系统需要等到此索引节点解锁以后再继续进行。

不论用什么方法找到一个新的索引节点，系统都必须调用一个特殊的子程序来把实际文件系统的信息添加到此索引节点中。当向新的索引节点中写入信息时，此索引节点计数器的值为1，并且系统将锁定此索引节点直到索引节点中的信息有效为止。

为了获得可以使用的VFS索引节点，文件系统可能需要读取很多其他的索引节点。例如，当你读取一个目录时，只有最后一个目录的索引节点才是你所需要的，但你还必须要读取此目录路径名中其他中间目录的索引节点。当 VFS索引节点缓存用完以后，最少用到的索引节点将会被扔掉，而较为常用的索引节点将继续保存在 VFS索引节点缓存中。

15.3.8 VFS目录缓存

为了加速对常用目录的存取， VFS还提供一个目录缓存。

当实际文件系统读取一个目录的时候，目录的详细信息将添加到目录缓存中。下一次查找同一个目录时，系统就可以在目录缓存中找到此目录的有关信息。只有短于 15个字符的目录才能保存在目录缓存中，但正是这些短目录才是最经常用到的目录。

目录缓存包括一个散列表，表中的每一个入口都是指向具有相同 hash值的目录缓存链表的

指针。散列表使用文件系统所在设备的设备号和目录名来计算位移。

为了保证缓存的有效和及时的更新，VFS使用一个最新用到（LRU）目录缓存入口的链表。当一个目录入口第一次被放入到目录缓存时，也就是说当第一次查找此目录时，此目录将添加到第一层LRU链表的末尾。如果此缓存已满，它将替换LRU链表中最前面的一个目录入口。当此目录再一次被存取的时候，此目录将提升到第二层LRU链表的末尾。它也可能替换掉第二层LRU链表中最前面的目录入口。这种替换第一层和第二层LRU链表最前面的目录入口的策略可以把最近最不常用到的目录入口替换掉。第二层LRU链表的目录入口的级别要比第一层LRU链表的入口高。这也是我们希望得到的效果。

15.4 缓冲区缓存

当系统中的进程使用挂接的文件系统时，挂接的文件系统将会产生很多和块设备之间的读写请求。所有的数据块读写请求都通过标准的内核调用以buffer_head数据结构的形式传递给设备驱动程序。数据结构buffer_head中包含了设备驱动程序所需要的所有信息，其中设备标识符唯一地表示了所使用的设备，数据块号表明了驱动程序需要读取设备中的那一块。所有的块设备都可以认为是同样大小的数据块的一个线性的集合。为了加速对物理块设备的存取， Linux系统中使用了一个数据块缓冲区缓存。系统中所有的数据块缓冲区都存储在这个缓冲区缓存的某一个地方，即使是一些新的，没有用到的数据块缓冲区。所有的物理块设备都可以共享这个缓冲区缓存。在某一时刻，缓存中可以保存着属于不同块设备的，不同状态的多个缓冲区。如果缓冲区缓存中的数据是有效的，那么就可以节省系统存取物理块设备的时间。任何一个从块设备中读取的数据块缓冲区或者往块设备中写入的数据块缓冲区都要通过缓冲区缓存。

缓存中的数据块缓冲区是以拥有此数据块的设备的设备标识符和缓冲区的块号来唯一标识的。缓冲区缓存由两部分组成。第一部分是空闲的数据块缓冲区的链表。每一个有效大小的缓冲区都有一个链表。当系统中的空闲缓冲区新建或者缓冲区被扔掉以后就在这里排队。当前系统支持的缓冲区的大小是512、1024、2048、4096 和 8192 字节。第二部分是缓存本身。这是一个由指针数组构成的散列表，其中的指针指向具有相同 hash值的缓冲区。散列表的索引由设备的标识符和数据块的块号产生。图 15-7显示了散列表和几个散列表的入口。数据块缓冲区要么在一个空闲数据块缓冲区链表中，要么在缓冲区缓存中。当处于缓冲区缓存中时，它们也会在LRU链表中。每一种缓冲区类型都会有一个LRU链表。系统使用这些链表对某一种类型的缓冲区进行某种操作，例如，把保存了新数据的缓冲区写入到磁盘中。数据库缓冲区的类型反映了缓冲区的状态， Linux系统现在支持以下几种类型：

1. clean（干净的）

未被使用的，新的缓冲区

2. locked（锁定的）

已经被锁定的缓冲区，等待数据的写入。

3. dirty（已用的）

已经改变的缓冲区。也就是缓冲区中包括有效的新数据。这些数据将要被写入到磁盘中，但目前还未写入。

4. shared（共享的）

可以共享的缓冲区。

5. unshared（非共享的）

这些缓冲区以前可以被共享，但现在不能被共享。

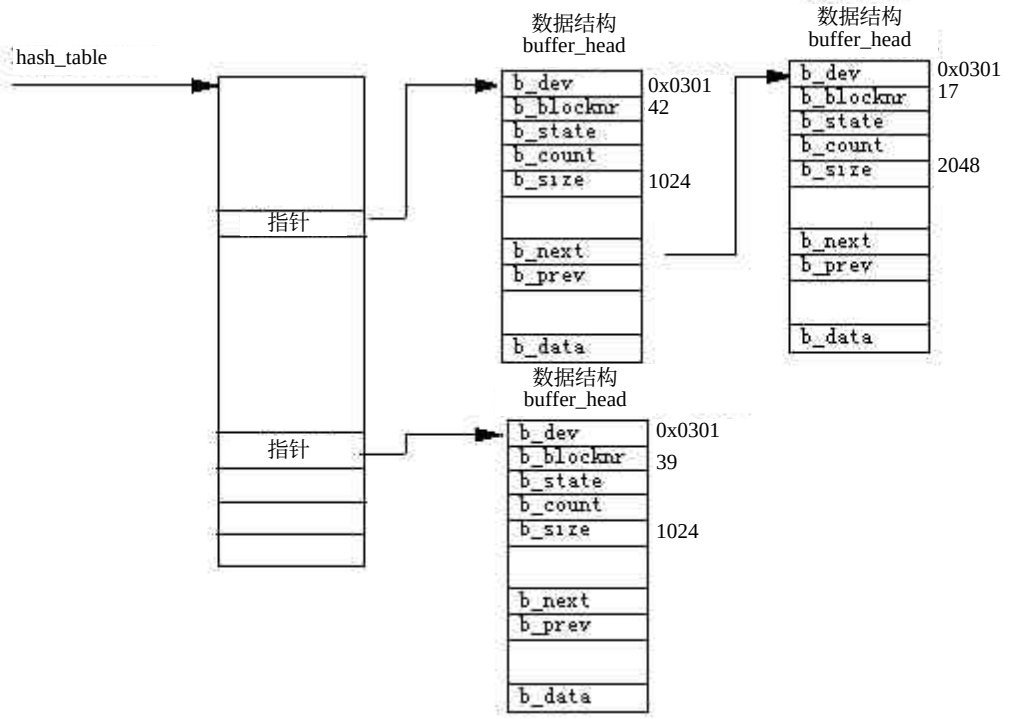

图15-7 缓冲区缓存示意图

当一个文件系统想要从物理块设备中读取一个数据块时，它都首先试图从缓冲区缓存中查找此数据块。如果缓冲区缓存中没有此数据块，那么文件系统将会从相应大小的空闲数据块缓冲区链表中分配一个空闲的数据块缓冲区，然后把它放入到缓冲区缓存中。如果文件系统希望查找的数据块缓冲区已经在缓冲区缓存中了，那么也许此数据块缓存包含的不是最新的数据。如果是这样，或者这是一个新分配的数据块缓冲区，那么文件系统则必须请求设备驱动程序从磁盘中读取相应的数据块。

像所有的其他缓存一样，系统必须经常地维护缓冲区缓存以使得它可以十分有效地运行，并且可以在各个使用缓冲区缓存的块设备之间公平地分配缓存入口。 Linux系统使用bdflush内核守护进程来完成管理缓存的功能。

bdflush 内核守护进程

内核守护进程bdflush的任务是当缓冲区缓存中被改动的缓冲区数量太多时，提供一个管理功能。它在系统启动时作为一个系统的线程运行，此线程的名字叫做 kflushd。你可以使用ps命令查看系统中的此线程。在大部分时间中，此守护进程都处于睡眠状态，等待系统中被改动的数据块缓冲区的数量增大到一定的值。每当缓冲区缓存中的数据块缓冲区分配和放弃时，文件系统都要检查缓存中被改动的数据块缓冲区的数量。当此数量达到一定的百分比数时， bdflush守护进程将被唤醒。缺省的百分比是60%，但如果系统中急需数据块缓冲区，那么守护进程可以随时被唤醒。你可以使用update命令查看和改变这个百分比：

```
# update -d

bdflush version 1.4
0:    60 Max fraction of LRU list to examine for dirty blocks
```

```
1:  500 Max number of dirty blocks to write each time bdflush activated
2:   64 Num of clean buffers to be loaded onto free list by refill_freelist
3:  256 Dirty block threshold for activating bdflush in refill_freelist
4:   15 Percentage of cache to scan for free clusters
5: 3000 Time for data buffers to age before flushing
6:  500 Time for non-data (dir, bitmap, etc) buffers to age before flushing
7: 1884 Time buffer cache load average constant
8:    2 LAV ratio (used to determine threshold for buffer fratricide).
```

所有被改动的数据块缓冲区都连接到BUF_DIRTY LRU链表中。

15.5 /proc 文件系统

/proc文件系统是最能显示Linux的VFS作用的文件系统。它实际上并不存在，并且 /proc目录和它下面的子目录以及其中的文件也不存在。但和其他实际存在的文件系统一样， /proc文件系统也在VFS中注册。当VFS调用/proc文件系统中打开的目录或者文件的索引节点时， /proc文件系统才利用系统内核中的有关信息创建这些文件和目录。例如， /proc/devices文件是从内核中描述系统设备的数据结构中产生的。/proc文件系统提供给用户一个可以了解内核内部工作过程的可读窗口。