北京希望电脑公司UNIX技术丛书

UNIX环境中的高级数据处理技术

李春葆
马玉枫 编写

海洋出版社

北京希望电脑公司UNIX技术丛书

UNIX环境中的高级数据处理技术

李春葆
马玉枫 编写

- UNIX System V和伯克利文件的内部结构。
- 开发和处理普通文件系统的UNIX工具。
- 数据库概念、术语和开发关系数据库管理系统的基本原理。
- INFORMIX-SQL和嵌套SQL
- 在C程序中使用C-ISAM函数直接访问UNIX文件。
- INFORMIX-TURBO如何满足UNIX环境下连接事务处理的可靠性和高性能的需求。

海洋出版社

1993年·北京

内 容 提 要

本文详细展示了UNIX的数据处理功能，讨论了如何以很少的开发时间用简单的命令开发功能强大的系统。本书分为两部分，第一部分介绍开发所需要的一些UNIX 工具；第二部分讨论了各种INFORMIX产品及这些产品之间的交互作用。本书可供UNIX环境下的高级数据库设计师使用，对初学者和数据管理员也有所帮助。

需要本书的用户，请直接与北京8721信箱联系，电话2562329，邮码100080

（京）新登字087号

责任编辑：阎世尊

UNIX环境中的高级数据处理技术

李春葆　马玉枫　编写

希望　审校

•

海洋出版社出版（北京市复兴门外大街1号）

海洋出版社发行　　北京昌平建华印刷厂印刷

开本：787×1092 1/16　印张：16.185　字数：363千字

1993年2月第一版　　1993年2月第一次印刷

印数：1—5000册

•

ISBN 7-5027-2894-5/TP·132　　定价：11.00元

前　言

在最近几年里，UNIX操作系统已经从大学和研究机构的计算机实验室走向商业市场，UNIX不再仅仅作为教授学生学习程序设计技术的环境，它广泛应用于数据处理领域，UNIX已经成为数据处理应用的标准操作环境，同时许多厂商在UNIX下开发了成功的数据库管理系统。

INFORMIX是UNIX世界中第一个认识到数据库管理系统潜力的厂商，他在UNIX DBMS中开发大量的高效率的产品。

UNIX本身具有把一个普通文件和管道与换向使用以及shell相结合的能力，为应用开发提供了激动人心的环境。

本书展示了UNIX的数据处理功能，讨论了如何以很少的开发时间用简单的命令开发功能强大的系统。本书第一部分介绍开发所需要的一些UNIX工具，第二部分讨论各种INFORMIX产品以及这些产品之间的交互作用，从UNIX的观点看，INFORMIX产品既可以分开使用也可综合在一起使用。

阅读本书需要对UNIX系统有些基本了解，有些章节需要C语言程序设计知识。

虽然本书是为UNIX环境下的高级数据库设计师写的，但对初学者和数据管理员也有所帮助。

编者

1992

目　录

第一章 UNIX文件系统

数据处理，无论是传统的一般文件系统还是支持DBMS的系统，都涉及到文件信息的管理。这些文件被存放在诸如磁带、磁盘的二级设备上。本章讨论的是磁盘数据。文件中的字节不是物理相邻的，用户所见的文件是逻辑上连续的字符流，而提供给用户的数据结构和存取算法则是完全依赖于UNIX文件系统。而且，文件系统负责管理和维护数据存储的物理形式，这包括给新文件分配空间和存储信息。

UNIX文件系统自问世以来已经历了多次改进，除了基本原理外，在性能和灵活性方面有了很大发展。最广泛流行的文件系统是第七版的System V，本章将讨论System V和伯克利（Berkely）文件系统，并对分布式文件系统作一点简要说明。

1.1 文件的类型

UNIX文件系统有几种类型的文件，大部分版本中把文件分为普通文件，管道文件，目录文件和特殊文件（如设备文件）。

一个普通文件用来存放一列或一组字节的信息，对用户来说，这些信息好像是连续存放的，尽管实际上并非如此。

一个管道文件是用于进程间的通讯，与读取普通文件不同，要读取管道文件有一定限制，同一个数据只能被读取一次。

UNIX支持分级文件系统。在一个目录中的所有文件名字以一种特定格式存于一个文件，这个文件叫目录文件。当一个用户建立一个新文件时，这个目录文件就作相应改变。用户不能直接改变目录文件的内容，否则，目录的结构和完整性就无法保证。

特殊文件，如设备文件，是进程设备之间的接口。在UNIX系统中，设备被看成是特殊文件，设备驱动软件则是这些特殊文件的接口。

在数据处理环境中，用户将他们的数据存于普通文件中，用户或DBMS可直接控制数据的结构和内容。这些普通文件存放在如磁盘一类的块设备中。

1.2 磁盘组的物理结构

磁盘组由盘片、磁道、柱面和扇区组成，如图1.1。从用户角度来看，一个文件是字节的

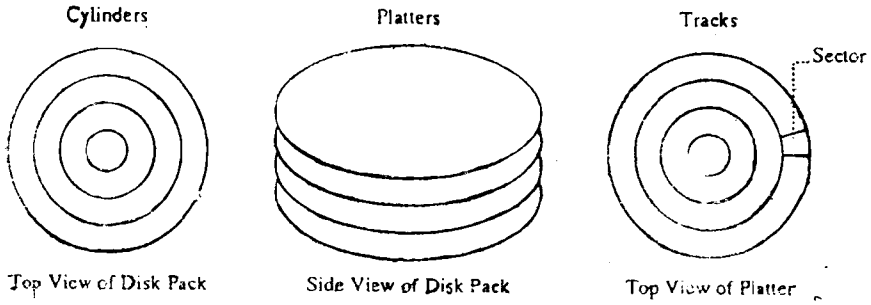

图 1.1 磁盘组的结构

逻辑组，这些字节组成块，因此，文件是由块组成，块与图1.1中的扇区相对应，但是，这不是一一对应的，一个块可以包括多个扇区。

1.3 System V 文件系统结构

每个盘组可以划分成一个或多个分区，每个分区容纳一个文件系统。文件系统由若干块组成，其中，一些块存放数据而另一些存放控制信息。了解这些控制块的结构和内容，对于全面了解文件系统是很重要的。

前面已经说过，一个文件的多个块可以是不连续，一个文件的各个块存放于什么地方由操作系统决定。为实现文件空间的管理，文件系统保留了占用表来存放占用信息。当一个文件产生时，将分配一些新块给它；当一个文件移出时，其相应的块必须被收回，以便分给以后的文件。下面几节将详叙文件块是如何维护的。

1.3.1 目录与文件

UNIX系统中，普通文件必须要有文件名，System V 规定文件名不多于14个字符。此外，一个文件必须属于某个目录。一个目录可以包括一个或多个文件。每个文件在目录中有一个16字节的登记项，前2个字节是整数，称为这个文件的索引节点号，可以用ls命令带参数i来显示某个文件的索引节点号。后面14字节是文件名。此外，还有两个登记项：一个是目录本身，另一个是当前目录。目录结构可以用od命令来显示。下面这个例子显示了目录文件/tmp的结构。

```
od-c /tmp
0000000 \0   7   .  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000020 \0 002   .   .  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000040 006 235   .   t   m   p  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000060 007   p   e   r   r  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000100 007   X   t   .   t   r  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000120 002   X   .   w   i   n   l   o   a   d  \0  \0  \0  \0  \0
0000140 007 206   n   e   w   t  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000160 007   d   l  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000200 007 217   t   .   s   h  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000220 007 221   t  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000240 003 031   R   x   .   c  \0  \0  \0  /0  \0  \0  \0  \0  \0
```

这里显示了前两个字节是索引节点号而后14字节是文件名。目录仅仅是文件/数据的结构，操作系统通过索引节点号来标识文件，索引节点号在索引节点表中。索引节点表的大小决定一个文件系统所能容纳的最大文件数目。表中的每个登记项称为索引节点。当数据块信息存于一个文件时，索引节点要存入文件的有关信息，索引节点的各部分信息列举如下：

- 文件类型：它标识文件的类型；普通型、管道型、目录型或特殊类型。
- 用户标识（Id）：标识文件的拥有者，拥有者关系用整型数存储，其实际用户名可以在/etc/passwd文件中查到。
- 组标识，标识文件的组属关系，以整数形式存储，其真正属于哪个组的组名由文件/etc/group可查到。
- 连接号：标识文件的连接号，一个文件可能会出现在不同的目录下，而实际的物理文

件只有一个，一个文件的连接号是它在各个目录中的目录号。

• 文件大小：文件的字节大小。

• 文件存取时间：文件的最近一次被存取时间。

• 文件修改时间：文件的最近一次被修改时间。

• 索引节点修改时间：索引节点的最近一次被修改时间。注：读一个文件将改变其存取时间但并不改变索引节点的修改时间。

• 数据块号数组：13个数据块号放在该数组中，这些数据块号被用来标识与文件相关的数据块。

应用UNIX命令ls可从索引节点中查询到文件的信息。

1.3.2 文件创建

当一个新文件被创建，它将在被创建的那个目录中增加一个新的登记项，UNIX文件系统也要建一个新的索引节点号，并分配一个索引节点，其中的文件大小字段置为零。

如果没有空的索引节点可用，则创建进程失败。当然，用户在一个目录中创建一个文件必须得到合适的条件许可。

1.3.3 文件的增长

如果索引节点分配成功并且文件已创建，则用户可向该文件中插入数据。当数据块第一次被分配时，其数据块号将存入索引节点的数据块号数组中的第一入口。如果文件长超过一个块，则分配给它下一个块且该块的块号将存入数组的第二入口，如此分配，直到第10个数据块。如果文件继续增长，第11个入口将指示一个间址块地址，这个间址块中存放的才是数据块的块号。

如果一个块的块号是占4字节的整数，则一个大小为1k字节的间址块可以容纳256个块号，如果这些由间址块指示的数据块全部用上，则共有：$1K\times10+256\times1K=266K$（字节），这对于大部分开发环境来说已经相当充足了，但UNIX文件系统还提供了更多的间址块来分配更多的数据块。索引节点中数据块数组第十二入口指示二次间址块，它的内容则是一次间址块的块号，至此，文件的最大长度可占据：$1K\times10+256\times1K+256\times256\times1K=65802K$（字节）。同样，第十三入口指示三次间址块，它的内容是指示二次间址块的块号，如此一来，文件的最大长度可增长到：$1K\times10+256\times1K+256\times256\times1K+256\times256\times256\times1K=16843018K$（字节），这便是UNIX system V的最大文件长度。图1.2展示了索引结点的结构和数据块号数组的13个入口。

1.3.4 空闲块链表

当一个新的文件系统被创建时，各个空闲块将连接起来组成一个表。每个空闲块是该表的一个结点，每个结点包含一列可用块的块号，该列的第一个块号指示的是下一个结点，如图1.3所示，连接表的第一个结点的第一项105指示的是下一个结点的块号，同样第105块中的第一项指示的是第312块，以此类推。

1.3.5 空闲索引结点表

如前所述，每个文件有唯一的索引结点相对应。索引结点表存放在磁盘上。如果一个索引结点的状态字段为空或零值，则该结点为空闲结点。给空闲结点赋空值的最简单机制是，逐个搜索盘上的索引结点表，直到找到一个空结点。但这种机制速度太慢，为减少分配一个空结点时的读盘时间，文件系统建立了一个空闲结点子集表放在超级块中。

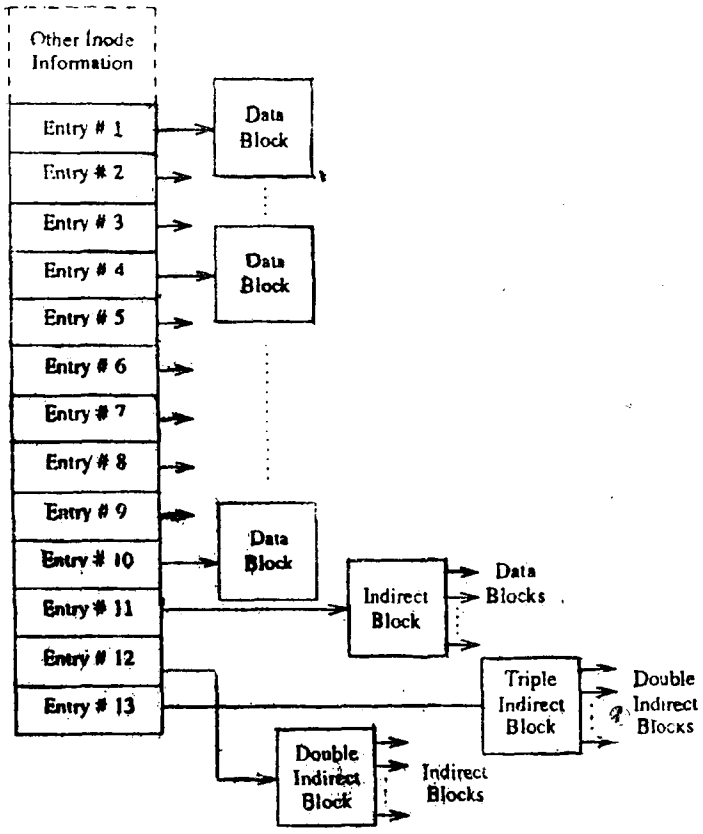

图 1.2 索引结点中空闲数据块数组

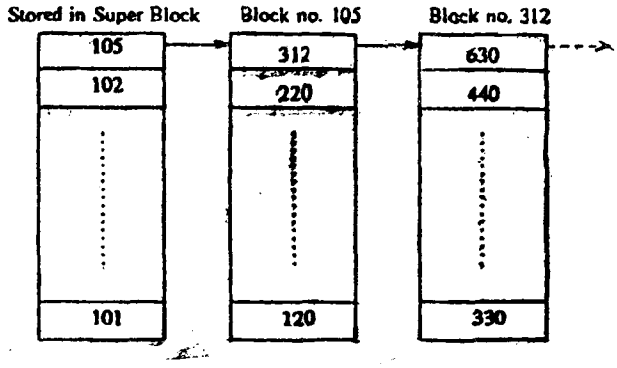

图 1.3 空闲块链表

1.3.6 超级块

超级块是文件系统中最重要的一个块，它存储文件系统的有关信息。一般超级块被安装在文件系统的磁盘分区首端的一定偏移位置上。超级块包括如下信息。

- 文件系统的大小
- 空闲块的数量（整个文件系统的块数）
- 空闲索引结点数
- 索引结点表的大小
- 空闲块表：该字段存放空闲块链表中的第一个结点
- 空闲索引结点表：是空闲结点的一个子集，存放一列索引结点号，其作用是避免在分配一个新结点时对磁盘的搜索。
- 空闲块表索引：空闲数据块表的第一个空位中放该索引
- 空闲索引结点表索引：空闲索引结点表的第一空位中放该索引
- 超级块修改标志：当使用mount命令时文件系统被联机，此时，超级块被读入内存修改文件系统将导致超级块本身的修改。这个修改标志可以被程序来检验，如sync（）系统调用可检查标志是否被修改。
- 只读标志：如果该标志被置位，则文件系统就只可读不可写入，这个参数可以在文件系统联机时置入。
- 锁字段：该字段用于封锁空闲块表和空闲索引结点表。

1.3.7 索引结点的分配和释放

创建一个新文件的第一步就是为新文件分配一个索引结点。超级块中保留了空闲索引结点数组和进入该数组的索引，从该数组可获得一个可用的结点号。当需要一个新结点时，索引被减去而该索引所指示的索引结点被分配。如果超级块中的索引结点表为空，则读出索引结点表的磁盘拷贝，并且给超级块分一个新的空闲结点号子集。重要的是超级块中只保留空闲结点数组，而空闲结点本身仍在磁盘上。

一旦一个索引结点号被分配，则超级块中的空闲结点数量则相应减少，该结点的磁盘拷贝被读入内存，并且其占用字段被初始化，类型字段也从零或空变为相应的类型，从而指示该结点不再为空。

当一个文件被取消时，其对应的索引结点将被释放。这个过程有几步：首先，类型字段被改为空值，然后，相应的超级块中的结点总数增加，该结点即成为可用的空闲结点。但从效率上考虑，此新结点被放入超级块中的索引结点数组中，并建立相应的索引指示它。

如果索引结点的超级块表中没有空位，则要将新释放的结点号与超级块中结点数组第一入口的结点号相比较，如果新结点号小，则它将进入第一入口，而原第一入口中的结点被删除。注意，这种删除并不影响该结点的可用性，它只是被放到所有空结点的磁盘拷贝上去，其标志仍为空闲结点，当现有的数组中的全部结点均被使用时，新的结点表将从磁盘中取到数组中，换言之，每次使用的结点总是从数组中的结点选择，这样可以减少系统的查找时间。

1.3.8 数据块的分配与回收

给一个新创建的文件分配一个数据块，要通过空闲数据块表来分配。与索引结点的分配一样，空闲数据块的块号保留在超级块中，在数组中有索引指示表的第一个空位。超级块中

的数组实际上是空闲数据块链表的第一个结点。这在图1.3中已有说明。如果需要分配一个新块而超级块中又只剩下一个入口，则该入口指示的块被读入到超级块数组中，该块被分配。

图1.4显示了105号块分配前表的状态，图1.5则是105号块分配后的情形。

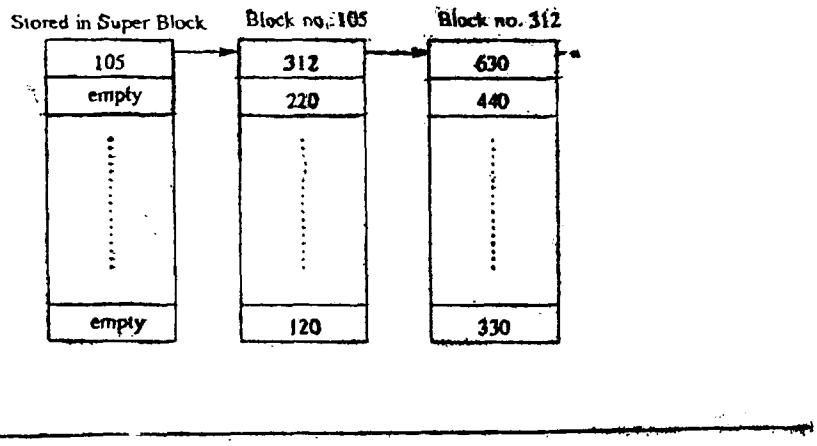

图 1.4 105号块分配前

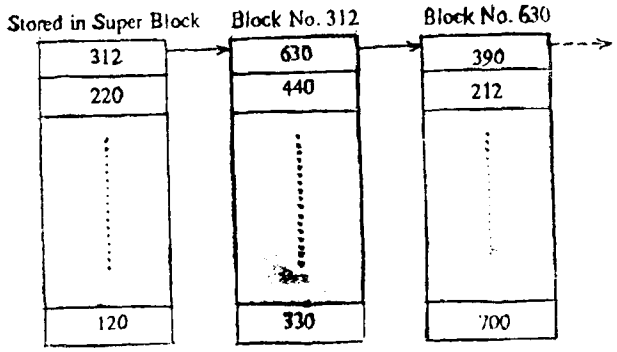

图 1.5 105号块分配后

释放一个块需要把该块的块号连接到空闲链表中。把新数据块号置入索引指示的空位上，索引本身增加1即可。

当超级块中的空闲块表已满时，释放一个块的过程则稍复杂一些。这时，超级块中的数组内容被拷贝到新释放的块中，而该块的块号则填入超级块数组中的第一个位子上。图1.5显示了621号块释放以前表的状态，图1.6显示了该块释放后的情形。

1.3.9 文件中的缺口

当用户试图搜索和写入一个文件到当前不存在的位置上时，我们称文件存在着缺口。如果用户试图搜索一个位置，而该位置将导致不可用的块上去，则这些块将不被实际分配，且块号在索引结点的入口中将被置为零。

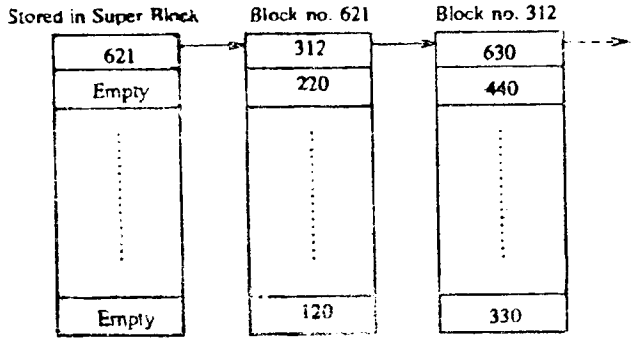

图 1.6 第621号块释放之后

1.3.10 系统输入/输出缓冲区

当进程要存取一段文件时，在主存中用于传送文件的一片中间区域称为高速缓冲区。应用数据缓冲区有几个好处：

首先，磁盘上的数据的修改只能以块为界，在没有系统缓冲区时，要想存取不等于一个块的数据，则必须写追加码来处理输入/输出操作。

如果一个文件是顺序存取的，则在下一个块被存取前先读入缓冲区，这种预读节省了相当的时间，因为进程不需要等待输入/输出操作。

缓冲区使用户不必每次操作都写盘，这种延迟写机制节省了输入/输出处理时间，因为用户可能会再次存取这个块。当用户要读这个块时，情形也是一样的，如果该块在缓冲区中，则再读这个块就无需任何输入，输出操作。

如果没有缓冲区，切换进程也是很困难的。当进程起动一个I／O时，必须等I／O操作完成后才能切换出内存。如果用缓冲区，该进程可以在I／O的同时切换出来，空出主存给别的进程，当I／O操作完成后，该进程再进入，继续执行。

对于数据的缓冲，内存中的某个固定区域被分配作缓冲区数组，每个缓冲区有两个部分：一是数据部分，用于存储磁盘块上的实际数据，另一部分是缓冲区头部，存放关于缓冲区的信息。这些信息包括缓冲区对文件系统的映象以及当前状态，映象指示着缓冲区描述的文件系统中的块。缓冲区当前状态指示缓冲区是否为空，进程是否在等待另一进程来释放它，以及内核是否在读或在写缓冲区内容，等等。

当一个进程要从文件中读数据时，可以顺序读或直接搜索并读出。不论哪种方式，进程必须首先读一些特定字节。这些字节被转换为块号，然后在缓冲区中开始查找。如果在缓冲区中找到该文件和块号，且当前也没有其它进程占用它，则缓冲区搜索操作完成。如果当前有用户在用，那么进程被挂起直到缓冲区空出时被唤醒。如果在一个缓冲区中没有找到，则从缓冲区表中分配下一个继续查找，以此类推。还有一些更复杂的情形，我们不在此讨论。

1.3.11 封锁机制

新发行的UNIX System V提供了封锁机制，这种封锁机制是通过同步机制来保持数据读取和修改时的完整性。用户进程可以很方便地对记录或文件加锁，以防止其它进程存取操作。C语言程序设计者可以使用系统调用fcntl（）或库函数lockf来实现上述功能。

记录封锁属于对任意文件段加锁，因为文件封锁可以看作是记录封锁的特殊情况，即整个文件看作一个记录。锁分读锁和写锁。如果一个文件是“读封锁”的，则其它进程只能读它而不能写入；如果是“写封锁”，则其它进程即不可读也不可写该文件。锁可以通过某个进程从读锁升为写锁。当一个读锁被设置在一个记录上时，该记录亦可被其它进程设为读锁，但如果其它进程要设置该记录为写锁时，必须先释放读锁，一旦写锁设置后，其它进程就不能在该记录上加锁了。

一个记录被加的锁也可以由某个进程来部分释放，这种锁机制称为非强制锁，它不是由I／O系统实施。fcntl系统调用增加了一些命令提供这种非强制锁的能力。由I／O系统执行的另一类记录锁称为强制封锁，除了用户进程提供的封锁之外，这种强制封锁还提供了同步检验，但这将增加系统的开销。应用chmod命令带+l选择项即可对一个文件加上强制封锁。例如，下面这个命令

```
chmod +l file1
```

将文件file1加强制性封锁。如果一个进程在已加了锁的一些记录上应用了非强制性锁，则这些记录被封锁，而其它记录上的强制性锁被解除，它们可以被存取，就如同没有强制性封锁一样。主要的开销与强制性锁相关，每个输入输出操作都必须被检验记录是否封锁。

1.3.12 有关的UNIX命令

为了与UNIX文件系统交互，用户需要一些UNIX命令，下面我们讨论几个UNIX命令。

mkfs命令用于创建UNIX文件系统。文件系统的建立需要磁盘格式化，文件系统不能在一个没有格式化的磁盘上创建。格式化或建立一个新的文件系统，会删掉已经存在磁盘上的数据。下面这个命令在设备为“/dev/fp002”上创建了一个文件系统，占2000个块和400个索引结点：

```
mkfs /dev/fp002 2000 : 400
```

如果索引结点号没有被说明，则缺省号为块号除以4。

用df命令可以打印出一个文件系统的空闲索引结点数和块数。此外，-t选择项可用于得到总的块数和索引结点数，见下例：

```
df
/          ( /dev/fp002):      85144 blocks      13073 i-nodes
df -t
/          ( /dev/fp002):      85411 blocks      13073 i-nodes
                 total:       120944 blocks      15104 i-nodes
```

ncheck命令可用于生成索引结点号和对应的文件名表，-a选择项则显示出缺省时隐含的“.”和“..”的入口。下面是一个例子，这里只有前20行输出被打印出来：

```
ncheck -a |head -20
/dev/fp002:
2       /./.
2       /../.
3       /bin.
15      /mnt/.
16      /etc/.
```

```
30      /dev/.
55      /tmp/.
56      /lib/.
272     /lost+found/.
273     /mnta/.
274     /mntb/.
275     /u/.
300     /.profile
301     /UNIX3.51
301     /unix
302     /usr/.
3       /bin/./.
2       /bin/../.
4       /bin/sh
```

一个文件系统可用mount命令安装启动，用unmount命令可将已安装的系统卸下。如果文件系统正在运行，用unmount命令时会显示提示信息。用pwd命令可看到用户的当前目录。

1.3.13 文件系统的坏与恢复

UNIX文件系统结构为解决不可预知的冲突留了足够的空间，这些矛盾可能是由不良的文件管理所致，如，不合时宜的关机引起的系统故障等。文件系统存在几种矛盾冲突是可能的，一个索引结点可能被指示为已分配，但它都在任何目录中都没有登录；超级块本身也可能被损坏；块号也可能不合法。此外还有一些可能出现的错误，如：目录被损坏，目录的大小不是16的整数倍，一个块被多次分配，超级块中登录的空闲结点数与检验的磁盘上空闲结点表中的数目不匹配，等等。

有些冲突可以被纠正，有些则不行，UNIX程序fsck可用于检验和修复UNIX文件系统。它可以交互式或非交互式模式在一个没有安装的系统上运行，文件系统应该卸出，因为fsck是多道程序且文件系统的状态不能被改变。在交互式模式中，用户或操作员可以指定fsck的操作，在非交互模式中，则执行缺省的操作。如果使用－P选择项，则fsck作如下一致性验证：

- 没有定义的索引结点
- 索引结点的链太长
- 空闲表中丢失了数据块
- 块指示丢失但仍旧分配给了文件
- 超级块中有非法的数量，如空闲结点号不合法等。

如果有其它错误出现，则fsck将中止执行并呈现非正常返回状态。这时，可调用交互模式来解决。如果有－y选择项调用，则是假设对所有的问题均以yes来作答。如果fsck检测到一个目录或文件没有被连入文件系统，则把它连入到lost+found目录中。

在启动时，fsck应该调用系统初始化程序.rc，如果一个损坏的文件没有修复就使用，则将引出多路冲突，并极可能导致文件系统的不可修复。

clri命令可用于清一个索引结点，其语法为：

clri *file-syssem i-number*