第8章 文件系统

Linux是建立在文件系统的基础上的。文件系统是对磁盘进行组织，在扇区和磁道组成的物理基础上提供抽象操作层面的机制。本章我们将讨论 Linux的缺省文件系统ext2所支持的全部抽象操作层面的构成和管理操作。

注意 在开始学习本章之前，需要对 Linux环境中的文件、子目录、访问权限以及所有者等概念有比较清晰的理解。如果没有学习过第 6章的内容，在继续学习之前最好还是先去看看。

本章我们将讨论磁盘管理方面的许多问题，包括建立硬盘分区、建立文件系统、引导系统时自动挂装文件系统、以及发生系统崩溃之后如何对它们进行恢复等等。除了这些基本的东西之外，我们还将涉及Linux一些复杂的功能，比如挂装网络文件系统、硬盘空间配额管理、以及自动挂装（automounter）子系统等等。

8.1 文件系统的构成

我们从介绍Linux操作系统的文件系统结构开始。这样有助于加深对这些概念的理解，并让读者更容易地看到如何利用好这种结构。

8.1.1 i-结点

许多UNIX文件系统（包括Linux的ext2）最基础的建筑材料就是i-结点（i-node）。i-结点是一个包含着指针的控制结构，其中的指针要么指向其他 i-结点，要么指向数据块。

i-结点中的控制信息包括文件的所有者、访问权限、长度、最后一次存取时间、建立时间、用户分组 GID号等等（如果真的很好奇，在 /usr/src/linux/include/linux/ext2-fs.h文件中可以查到完整的内核数据结构 —— 当然需要假设你已经把/usr/src子目录树的内容全部都安装好了）。i-结点中没有保存的东西就是文件名。

我们在第 6章中已经介绍过，子目录本身就是文件的一种特殊形式。这就意味着每一个子目录都有一个 i-结点，这个i-结点指向的数据块中包含着关于这个子目录所有文件的信息资料。图 8-1给出了ext2文件系统中 i-结点和数据块的组织结构。

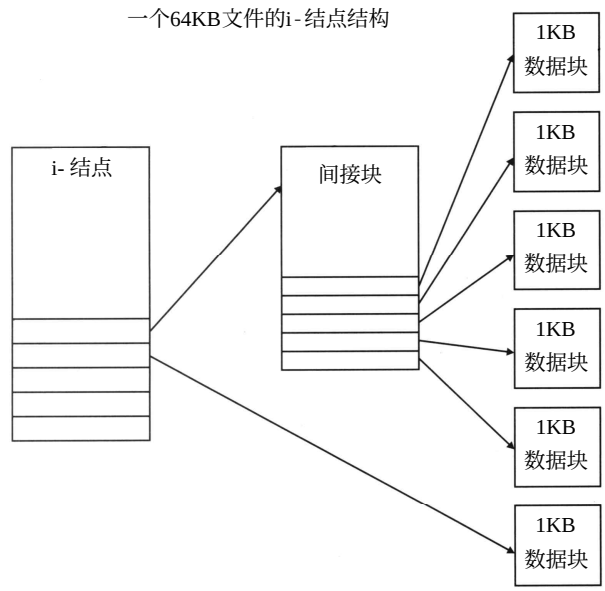

图8-1 ext2文件系统中i-结点和数据块的组织结构

从图8-1可以看出，i-结点的作用是“间接”的，这就可以指向更多的数据块——这也是为什么i-结点中不保存文件名的原因（只有一个 i-结点作为整个文件的代表；因此在 i-结点上保存文件名信息就是一种空间浪费）。举例来说，我的6GB硬盘上一共有1,079,304个i-结点。如果每个i-结点都花费256个字节来保存文件名的话，为了保存文件名预留的空间就要浪费掉大约33 MB的空间！

8.1.2 超级块

从磁盘上读出来的第一块信息就是它的超级块（ superblock）。这个小数据结构中保存着好几个关键的数据，包括磁盘的几何尺寸、可用空间容量、以及最重要的——第一个i-结点的位置。如果没有超级块，文件系统就没有使用意义。

超级块数据结构被拷贝复制了许多份，散布保存在整个磁盘上，以此对付第一个超级块被损坏事件的发生。在Linux的ext2文件系统中，在每一组数据块的后面就安排有一个超级块。每个数据块组包含着i-结点和数据。每个组有8192个块，这样第一个备份超级块就在8193，第二个在16385，依次类推。

8.2 管理文件系统

有人说，管理文件系统没有什么了不起——如果真的能记住网络化服务器、磁盘、备份、以及空间要求等所有这些方方面面，还能够保证它们不再发生变化，那么管理文件系统也确实没有什么了不起。换句话说，管理文件系统其实是相当复杂的工作。

文件系统不会出现许多技术方面的问题。系统一旦建立好、开始使用、并且进入备份循环之后，它们基本上就能够照顾自己了。使管理工作复杂化的东西是系统管理方面的问题——比如有的用户拒绝整理他们的磁盘空间、而繁杂的管理政策指定何人何时可以共享哪个磁盘的依据竟然是那个磁盘是由谁购买的、如此种种...（这听起来有点像是卡通片里的套路，但是现实生活中确实有很多情况与此相类似）。

更糟糕的是，没有可以用来对付办公室现象的教科书，因此在这一章中我们将围绕着文件系统管理技术方面的问题展开——主要内容有：挂装和卸载硬盘分区、使用 /etc/fstab文件、以及使用fsck工具程序进行文件系统恢复等。

8.2.1 挂装和卸载本地磁盘

Linux操作系统的长处包括它在文件系统管理方面的灵活性以及对文件定位的无缝管理。硬盘分区经过挂装之后看起来就像是一个下级子目录。从用户的角度看，即使是文件系统的某个基本部分也像是一棵巨大的目录树。这个特性特别有利于系统管理员，他可以把硬盘分区重新定位到各种服务器，但是仍然可以把那个硬盘分区挂装在目录树原来的位置上；而文件系统的用户根本就不必知道那些移动。

文件系统的管理工作是从根目录开始的（如图 8-2所示）。包含着操作系统内核及核心目录结构的分区是在系统引导时挂装的，这个分区上必须存放有使系统进入单用户模式所必须的全部工具程序和配置文件，其上许多子目录都是空的。

随着引导脚本程序的执行，其他分区也挂装到文件系统的结构上了。挂装命令用它正在挂装的分区上的目录树覆盖掉最初的某个子目录。举例来说，假设 /dev/hda1是根分区，它有

一个/usr子目录，但是其中没有任何文件；分区/dev/hda3中包含着我们希望出现在/usr子目录中的文件；因此我们把分区/dev/hda3挂装到/usr子目录的位置上。现在，用户只需简单地把路径切换到/usr子目录就可以看到分区/dev/hda3上的文件。用户不需要知道/usr其实是另外一个分区。

请记住：当挂装上一个新子目录的时候，mount命令会把原来挂装在这个位置的全部内容隐藏起来。这样在我们的/usr例子里，如果在挂装/dev/hda3分区之前根分区的/usr子目录中确实保存有文件，挂装后那些文件就看不见了（那些文件并没有被删除，一旦/dev/hda3分区被卸载，原来/usr中的文件就又出现了）。

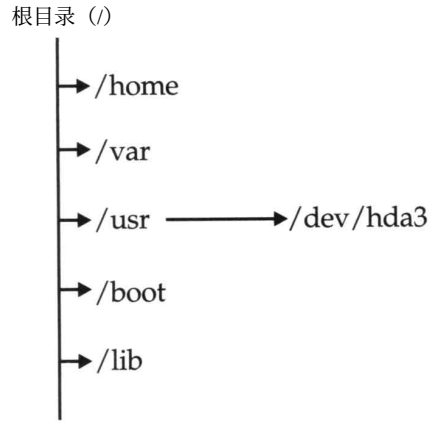

图8-2 子目录结构中的硬盘分区

1. 使用mount命令

mount命令和许多命令行工具程序一样，有非常之多的参数，其中大多数都不会在日常工作中用到。mount命令的使用手册页中有对这些参数全面详细的说明。在本小节里我们只介绍这个命令最常见的用法。

这个命令的格式如下所示：

```
mount [ options ] device directory
```

其中的[options]可以是表8-1中的任意一个。

用于mount -o命令的参数如表8-2所示。

下面的mount命令把/dev/hda3分区以只读属性挂装到/usr子目录上：

```
[ root@ford /root ] # mount -o ro /dev/hda3 /usr
```

表8-1 mount命令的部分参数

mount命令的参数	说 明
-a	把/etc/fstab文件（本小节后面介绍）中列出的文件系统都挂装上
-t *fstype*	定义挂装的文件系统类型。Linux可以挂装其他非ext2标准的文件系统，比如FAT、VFAT以及FAT32等等。mount命令通常可以自己检测出这类信息
-o *options*	定义作用于挂装过程的选项。它们通常是一些与文件系统类型有关的选项（挂装网络文件系统的选项不能够用来挂装本地文件系统）

表8-2 与mount命令的-o参数联用的部分参数

mount -o命令的参数（本机分区用）	说 明
ro	以只读属性挂装该分区
rw	以读-写属性挂装该分区（缺省值）
exec	允许二进制代码的执行（缺省值）
noatime	禁止刷新i-结点上的存取时间。用于存取时间不重要的分区（比如新闻队列），可以提高性能
noauto	如果使用了-a参数，禁止这个分区的自动挂装（只作用于/etc/fstab文件）
nosuid	禁止setuid程序的应用程序对此挂装分区置位
sb = n	告诉mount命令对一个ext2文件系统使用第n个数据块作为超级块

2. 卸载文件系统

如果想卸载一个文件系统，请使用umount命令。下面是这个命令的格式：

```
umount [ -f ] directory
```

其中的directory是准备卸载的子目录名。如下所示：

```
[ root@ford /root ] # umount /usr
```

把挂装在/usr子目录位置上的分区卸载下来。

当文件系统在使用中 umount命令有一个不足之处：如果文件系统正在使用中（也就是说有人在那个分区上打开了文件），就无法把这个文件系统卸载下来。有三种方法可以用来解决这个问题：

- 使用lsof程序（可以从站点ftp：//vic.cc.purdue.edu/pub/tools/unix/lsof下载）或者fuser程序检查有哪些进程打开了文件，终止那些进程的运行或者让进程的所有者停止操作。如果选择终止那些进程的运行，一定要明白自己在做些什么（我的意思是别为这个给自己招来麻烦）。
- 使用umount命令和-f参数强制执行卸载操作。任何在这个分区上打开的进程都将被挂起来，可能会造成数据丢失。
- 最安全和适当的办法是把系统调整为单用户模式，然后再卸载这个文件系统。在现实生活里，你可能不会总有这个奢侈之举。

3. /etc/fstab文件

我们在前面的内容里已经讲过，/etc/fstab是一个mount命令可以利用的配置文件。这个文件包含着一个系统中全部已知硬盘分区的清单。在引导过程中，这个清单被读出，其中包含的各个分区都被自动挂装到系统上。

下面是/etc/fstab文件中数据项的格式，表8-3定义了/etc/fstab数据项的各组成元素。

```
/dev/device /dir/to/mount      fstype   parameters   fs_freq    fs_passno
```

表8-3 /etc/fstab文件数据项中的各个数据元素

/etc/fstab文件的数据项	说 明
/dev/device	将被挂装的分区（比如/dev/hda3）
/dir/to/mount	分区挂装到其上的子目录（比如/usr）
fstype	文件系统的类型（比如ext2）
parameters	mount命令-o参数的附加参数
fs_freq	告诉dump命令备份这个文件系统的频率
fs_passno	告诉fsck程序在引导时确定文件系统的检查顺序（请注意所有文件系统在挂装之前都要被检查）

下面是一个示范性的/etc/fstab文件：

```
/dev/hda2       /          ext2    defaults        1 1
/dev/hda8       /home      ext2    defaults        1 2
/dev/hda7       /tmp       ext2    defaults        1 2
/dev/hda5       /usr       ext2    defaults        1 2
/dev/hda6       /var       ext2    defaults        1 2
/dev/hda1       /usr/local ext2    defaults        1 2
/dev/hda3       swap       swap    defaults        0 0
```

```
/dev/fd0        /mnt/floppy     ext2    noauto          0 0
/dev/cdrom      /mnt/cdrom      iso9660 noauto,ro       0 0
/dev/hdc        /mnt/cdrom2     iso9660 noauto,ro       0 0
none            /proc           proc    defaults        0 0
none            /dev/pts        devpts  mode=0622       0 0
```

我们来看看/etc/fstab文件中还没有讲到的那些细节，它们是 /dev/hda3的swap和/proc以及/dev/pts的none。一般说来，系统安装完成以后，你可能永远也不会与这些文件系统打交道，所以也不用担心它们。

- /dev/hda3分区是虚拟内存驻留的地方。 Linux与微软的Windows或者其他类似的操作系统不一样，它的虚拟内存可以保存在根分区以外的另一个分区上。因为 swap分区遵守的规则不同于普通的文件系统，所以这样做可以提高性能。因为这个分区不需要备份或者在机器引导时由 fsck程序检查，所以最后的两个参数都设置为 0（请注意swap还可以被保存为一个普通的磁盘文件。详细资料请查阅 mkswap命令的使用手册页）。
- 和/proc联系在一起的none数据项用来定义 /proc文件系统（详细资料请查阅第 26章）。这是一个特殊的文件系统，它提供了一个到内核参数的接口，通过这个接口可以模拟任何其他的文件系统。虽然它看起来好像是保存在磁盘上的文件，其实它不是的——所有文件分别代表内核中的某些东西。最引人注目的是 /dev/kcore，它实际上就是抽取成文件的系统内存映像。不熟悉/proc文件系统的人经常把它误认为是一个没用的大文件，错误地删除了它，从而引起系统发生许多明显的故障。除非你确实知道自己在干什么，否则最好还是把/proc中的文件留在那里别动。
- 最后一个数据项/dev/pts，是一个新的机制，用来完善网络终端支持（ ptys）的实现。如果你打算让你的主机支持通过 rsh、telnet、rlogin或者ssh的远程登录，这个数据项就是必不可少的。

如果使用了Linuxconf配置工具程序，就不必直接编辑 /dev/fstab文件。但是与系统管理的其他方面一样，你必须熟悉这个文件的组织结构，以便在需要时手动编辑它。

窍门 有了配置好的/etc/fstab文件，执行mount命令挂装分区的时候就可以只使用一个参数：准备挂装的子目录名称。mount命令会在/etc/fstab文件中查找这个子目录名称；如果找到了，mount命令就使用文件里设置好的参数进行挂装操作。比如说，下面就是一个挂装CD-ROM光盘的命令，/etc/fstab文件就是前面内容里的那个示范文件：

```
[ root@ford /root ] # mount /mnt/cdrom
```

8.2.2 使用fsck程序

fsck工具程序的名字是File System CheckK（文件系统检查）的缩写，它被用来诊断和修复在日常操作中可能已经损坏的文件系统。系统发生崩溃的时候，一般都来不及把内部缓冲区中的全部数据转存到磁盘上，所以类似的修复通常是十分必要的（虽然这个工具程序的名字与系统发生崩溃后经常听到的一个说法惊人地相似，但是这个工具程序成为系统恢复过程的一个部分纯属巧合）。

通常，系统在引导过程中如果发现有某个分区没有正常地卸载，就会自动运行 fsck工具程序（和Windows运行scandisk程序的意思几乎一样）。Linux操作系统尽了极大的努力来自动修

复它所遇到的问题，而且在大多数情况下都能把自己照顾得很好。而 ext2文件系统强壮的本质对这些情况也有帮助。不管怎么说，你可能会看到下面这样的提示：

```
*** An error occurred during the file system check .
*** Dropping you to a shell ; the system will reboot
*** when you leave the shell .
```

在这个时候，就需要你手动运行 fsck并自己回答程序的提问。

如果确实发现某个文件系统的操作行为与其正常情况不一样（ log日志是这类情况的最佳提示），你可能会在一个运转着的系统上运行 fsck程序。唯一的不足之处是：为了执行这个程序，需要诊断的文件系统必须先卸载下来。如果你选择了这个方法，完成操作后别忘记把文件系统再挂装上去。

注意 fsck并不是ext2文件系统修复工具合适的名字；它实际上只是一个打包器。fsck打包器尝试确定哪一个文件系统需要修复，然后再调用适当的修复工具程序，把我们传递给 fsck的参数都传递过去。对 ext2文件系统来说，真正的工具叫做 e2fsck。当发生系统崩溃的时候，与其依靠其他应用程序替你调用 e2fsck，还不如自己直接来调用它。

1. e2fsck的可用参数

表8-4中列出的参数都可以供 e2fsck使用。比如说，如果想在 /dev/hda3文件系统上运行 e2fsck，需要输入下面的命令：

```
[ root@ford /root ] # e2fsck  /dev/hda3
```

强制进行文件系统检查并对出现的全部提示都回答“ Yes ”，需要输入下面的命令：

```
[ root@ford ] # e2fsck -f -y /dev/hda3
```

表8-4 e2fsck的参数

e2fsck的参数	说 明
-b *superblock*	让e2fsck读取分区信息的超级块编号。大多数情况下， e2fsck可以在第一个数据块中找到它，但是如果那个块损坏了，就需要指定另外一个号码。超级块每隔8192个出现一次，因此第二个超级块在 8193、然后是16385等等
-c	在运行e2fsck之前先执行badblocks程序。它对整个硬盘按块查找并校验该块的完整性。这是检查硬盘最彻底的方法，但是花的时间比较多
-f	强制进行检查，即使认为文件系统已经没有问题了
-y	告诉e2fsck对e2fsck提示的问题全部自动回答为“ Yes ”

2. 出现错误信息怎么办？

首先，要放松。fsck检查出来的问题很少是它自己不能纠正的。即使它要求人的干预，告诉fsck按照它缺省的操作建议通常也就足够了。很少出现使用 e2fsck检查一遍之后还有问题没有解决的情况。

在很少出现需要第二遍检查的情况下，绝对不应该再出现更多的问题了。如果还是有许多问题，那就有可能遇到了硬件故障。记住先从明显的地方开始：检查电源是否正常、线缆是否接好。运转 SCSI系统的人们需要检查自己是否使用了正确的终端头，接线是否过长、SCSI设备的ID编号有没有冲突、电线的质量是否合格等等（ SCSI特别挑剔接线的质量）。

3. lost+found子目录

另外一个比较少见的情况是e2fsck找到了一些文件碎片，但是没有办法把它们恢复到原始文件中去。这种情况下，它会把这些碎片放到该分区的 lost+found子目录里。这个子目录就在该分区挂装的位置，因此如果分区 /dev/hda3被挂装在子目录 /usr上，那么子目录/usr/lost+found就对应于分区/dev/hda3。

任何东西都可以放到lost+found子目录里 —— 文件碎片、子目录、甚至一些特殊文件。如果在这里找到了普通文件，可以看出它的所有者，你可以与该所有者联系看他们是否还需要这些文件（他们通常都不需要）。如果在lost+found中遇到了子目录，与其试图从 lost+found中重新建立它的结构，还不如从最近的备份中来恢复。

lost+found子目录至少可以告诉你有一些东西失去了定位。但是，这种错误实在太少见了。

8.3 对硬盘进行分区

为了澄清一些概念，也为了你了解硬盘分区及其工作原理，我们简单地讨论一下这个主题。硬盘都必须进行分区。分区把硬盘分成几个段落，每个段落本身的动作都像是一个完整的硬盘。一旦某个分区被添满了，就不能够（如果没有特殊的软件的话）自动溢出到其他的分区上。通常，对硬盘进行分区操作出于两个目的：要么用户需要安装两种不同的操作系统，每一种都需要有自己的分区；要么出于谨慎的目的，对一个分区的使用不会影响到为其他分区上的任务所分配的空间。

后者的一个例子就是用户的 /home子目录。如果系统上的用户不是系统管理员，那么系统管理员就必须保证不会有用户用他们的个人文件消耗掉整个硬盘空间。否则会占用登录上机和临时文件的空间，进而影响系统运行。为了防止这个情况的发生，特别建立了一个分区来保存用户的文件，这样它们就不会溢出到受保护的系统空间了。

注意 把硬盘整个划分为一个拥有全部硬盘空间的大分区是可以接受的。

8.3.1 硬盘的表示方法

在Linux中，每个硬盘都分配一个自己的设备名。 IDE硬盘的名称以/dev/hdX开始，其中的X是从a到z的小写字母，每个字母代表一个物理设备。举例来说，一个只有 IDE接口的系统中有一个硬盘和一个 CD-ROM光驱，它们都在同一个 IDE链上，那么这个硬盘就是 /dev/had，而CD-ROM光驱就是/dev/hdb。磁盘设备是在安装的时候自动建立的。

到建立分区的时候，就需要用到新的设备了。它们的形式是 /dev/hdXY，其中的X是设备字母（刚讲解的），而Y就是分区编号。这样，在 /dev/hda硬盘上的第一个分区就是 /dev/hda1，第二个分区就是/dev/hda2，依次类推。

SCSI硬盘遵守与IDE一致的基本机制，但是设备名不使用 /dev/hd的形式而是以/dev/sd打头。因此第一个SCSI硬盘上的第一个分区就是 /dev/sda1，第三个SCSI硬盘上的第二个分区就是/dev/sdc2，以此类推。

8.3.2 建立硬盘分区

警告 建立分区的操作会对硬盘上原来的数据造成无法恢复的损失。在对任何硬盘进

行分区的建立、修改或者删除操作的时候，必须对所做的改动做到心中有数，如果数据还有用的话，别忘了先制作备份。

在安装Linux操作系统的时候，你可能是使用了一个“漂亮”的工具程序建立的硬盘分区。但是Linux操作系统本身是没有统一的分区建立与管理工具程序的。各种 Linux发行版本上都有的一个基本程序是 fdisk。虽然这个工具规模不大，操作也麻烦，但它是一个可靠的分区工具。更进一步来说，当真的需要检查一个故障系统的原因时，你就必须熟悉那些基本的工具，如fdisk。fdisk最大的缺陷是缺少用户界面。

注意 确实还有一个比较好用的打包器叫做 cfdisk。但是这个工具在出现紧急事件的情况下无法使用，因此最好还是学习如何使用 fdisk，不要计较它的缺陷。

为了讲解的方便，我们假设准备对/dev/hdb设备、一个340 MB的IDE硬盘进行分区（是的，这些硬盘还没有全淘汰呢）。我们从执行带/dev/hdb参数的fdisk开始。输入下面的命令：

```
[ root@ford /root ] # fdisk /dev/hdb
```

屏幕上出现一个简单的提示符：

```
Command ( m for help ) :
```

输入字母“m”看看有哪些参数。菜单中的自我介绍还算不错：

```
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
Command (m for help):
```

我们从查看现有硬盘分区开始，使用 p命令（显示分区表），如下所示：

```
Command (m for help): p
Disk /dev/hdb: 16 heads, 63 sectors, 665 cylinders
Units = cylinders of 1008 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1   *         1       664    334624+   6  FAT16
Command (m for help):
```

这硬盘上的系统够老的吧，你怎么看？该给它升升级了——先使用d命令删除现有的分区：

```
Command (m for help): d
Partition number (1-4): 1
Command (m for help):
```

再使用p命令（显示分区表）检验操作结果：

```
Command (m for help): p
Disk /dev/hdb: 16 heads, 63 sectors, 665 cylinders
Units = cylinders of 1008 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
Command (m for help):
```

原来的分区没有了。开始建立新分区。为了讨论的方便，我们假定这个硬盘大到足以容纳一个完整的工作站配置。为了这个配置，我们需要建立表 8-5中列出来的分区。

表8-5 硬盘分区

分 区	说 明
/	根分区，存放用来把系统引导到单用户模式所必须的核心系统文件。一旦建立好以后，这个分区中的内容就不应该变化，也绝不需要增长。目的是把这个文件系统与其他文件系统隔离开来，防止影响核心操作
/usr	这个分区用来保存系统软件，比如用户工具程序、编译器、 X-Windows等等。因为今后可能需要我们为这些系统软件找一个更大的“家”，所以现在就把它放在一个单独的分区里
/var	/var分区用来保存变化很大的文件 —— 通常包括队列子目录（电子邮件、打印等等）和日志文件。这个分区令人担心的情况是：外部活动可能会使其内容增长到分配给它的空间以外去。举例来说， Web服务器上的日志文件就可能增长得很快，让你不好控制。为了防止这些文件散布到系统的其他部分，把这个分区独立出去是明智的（网络上有一种攻击类型其原理是这样的：在被攻击方的服务器上人为地制造出数量巨大的活动，使得其硬盘被系统日志占满，影响系统工作的可靠性）
/tmp	类似于/var，/tmp中的文件也有可能消耗大量的空间。当用户不照管自己的程序或者应用程序生成了巨大的临时文件的时候，就会出现这种情况。不管是哪一种情况，给它分配一个分区是一个安全的好办法
/home	如果需要在硬盘上保存登录子目录，特别是有一些需要限制其硬盘空间使用量的用户时，肯定就需要把它单独安排在一个分区
swap	swap分区是用来保存虚拟内存的。虽然它并不是必须的，但保留 swap以备物理RAM全部消耗殆尽的情况发生是个不错的主意。通常，可以把这个分区设置为与RAM同样的大小

我们现在已经知道将要建立哪些分区，那就开始吧！先建立根分区。因为我们只有 340 MB的工作空间，所以需要把根分区设置得小一点 —— 就25 MB吧。

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-665, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-665, default 665): +25M
Command (m for help):
```

请注意第一个提示，问我们是建立主分区还是扩展分区。这是一个由来已久（比 Linux还早）的老大难了，当时的硬盘实在太小了，因此没有人相信需要超过四个分区。硬盘是越来越大，可是对过去的兼容性就成了一个问题，需要采取一定的技巧才能容纳更多的分区。最后一个分区必须是一个“扩展”分区，对用户来说是透明的，但是可以容纳额外的分区。

下一个问题：该使用哪一个分区号？（可以看出，四个主分区的限制也是问题的一部分。）我们从“1”开始，接受默认的起始磁道，然后说明我们打算分配给它 25 MB。

继续输入下面的内容，为swap建立第二个分区：

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (52-665, default 52): 52
Last cylinder or +size or +sizeM or +sizeK (52-665, default 665): +16M
Command (m for help):
```

这次出现的提示符和设置上一个分区时的一样，只有几个数字变了。但在缺省的情况下，fdisk建立的是ext2类型的分区。我们需要把这个分区设置为 swap类型。使用t命令（改变分区类型）来完成这项工作：

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): L
 0  Empty           16  Hidden FAT16    61  SpeedStor       a6  OpenBSD
 1  FAT12           17  Hidden HPFS/NTF 63  GNU HURD or Sys a7  NeXTSTEP
 2  XENIX root      18  AST Windows swa 64  Novell Netware  b7  BSDI fs
 3  XENIX usr       24  NEC DOS         65  Novell Netware  b8  BSDI swap
 4  FAT16 <32M      3c  PartitionMagic  70  DiskSecure Mult c1 DRDOS/secFAT-
 5  Extended        40  Venix 80286     75  PC/IX           c4 DRDOS/secFAT-
 6  FAT16           41  PPC PReP Boot   80  Old Minix       c6 DRDOS/secFAT-
 7  HPFS/NTFS       42  SFS             81  Minix / old Lin c7  Syrinx
 8  AIX             4d  QNX4.x          82  Linux swap      db CP/M / CTOS
 9  AIX bootable    4e  QNX4.x 2nd part 83  Linux           e1  DOS access
 a  OS/2 Boot Manag 4f  QNX4.x 3rd part 84  OS/2 hidden C:  e3  DOS R/O
 b  Win95 FAT32     50  OnTrack DM      85  Linux extended  e4  SpeedStor
 c  Win95 FAT32 (LB 51  OnTrack DM6 Aux 86  NTFS volume set eb  BeOS fs
 e  Win95 FAT16 (LB 52  CP/M            87  NTFS volume set f1  SpeedStor
 f  Win95 Ext'd (LB 53  OnTrack DM6 Aux 93  Amoeba          f4  SpeedStor
10  OPUS            54  OnTrackDM6      94  Amoeba BBT      f2 DOS secondary
11  Hidden FAT12    55  EZ-Drive        a0  IBM Thinkpad hi fe  LANstep
12  Compaq diagnost 56  Golden Bow      a5  BSD/386         ff  BBT
14  Hidden FAT16 <3 5c  Priam Edisk
Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap)
Command (m for help):
```

第一个问题是：我们想对编号第几的分区进行修改？因为我们打算把第二个分区设置为swap类型，所以我们输入2。接下来的提示有些古怪：分区类型的 16进制代码。有时候妻子还得提醒我几月几号呢，所以我是肯定记不住全部这些 16进制代码的。L命令列出了全部可供选用的分区类型。我们查到82对应于Linux Swap，所以把这个数字输入进去，完成这个步骤。

现在建立/usr。我们把它的大小设置为100 MB。如下所示：

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (85-665, default 85): 85
Last cylinder or +size or +sizeM or +sizeK (85-665, default 665): +100M
Command (m for help):
```

到这里我们已经有三个分区了：root、swap和/usr。输入p命令来看看它们：

```
Command (m for help): p
Disk /dev/hdb: 16 heads, 63 sectors, 665 cylinders
Units = cylinders of 1008 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1             1        51     25672+  83  Linux
/dev/hdb2            52        84     16632   82  Linux swap
/dev/hdb3            85       288    102816   83  Linux
Command (m for help):
```

现在来建立容纳/tmp、/var和/home的扩展分区。还是使用n命令，就像建立其他新分区一样，如下所示：

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
Partition number (1-4): 4
First cylinder (289-665, default 289): 289
Last cylinder or +size or +sizeM or +sizeK (289-665, default 665): 665
Command (m for help):
```

这里我们不再以兆为单位输入数字分配分区空间了，直接把最后一个磁道号码输入进去，这样就把硬盘的剩余空间全部都分配给这个分区。再来看看现在的分区情况：

```
Command (m for help): p
Disk /dev/hdb: 16 heads, 63 sectors, 665 cylinders
Units = cylinders of 1008 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1             1        51     25672+  83  Linux
/dev/hdb2            52        84     16632   82  Linux swap
/dev/hdb3            85       288    102816   83  Linux
/dev/hdb4           289       665    190008    5  Extended
Command (m for help):
```

现在来建立最后的三个分区：

```
Command (m for help): n
First cylinder (289-665, default 289): 289
Last cylinder or +size or +sizeM or +sizeK (289-665, default 665): +100M
Command (m for help): n
```

```
First cylinder (493-665, default 493): 493
Last cylinder or +size or +sizeM or +sizeK (493-665, default 665): +45M
Command (m for help): n
First cylinder (585-665, default 585): 585
Last cylinder or +size or +sizeM or +sizeK (585-665, default 665): 665
Command (m for help):
```

请注意对最后的一个分区我们再次输入了最后一个磁道的号码而不是一个以兆为单位的数字，确保把整个硬盘上的空间都分配完了。再使用 p命令看看现在的分区情况：

```
Command (m for help): p
Disk /dev/hdb: 16 heads, 63 sectors, 665 cylinders
Units = cylinders of 1008 * 512 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hdb1             1        51     25672+  83  Linux
/dev/hdb2            52        84     16632   82  Linux swap
/dev/hdb3            85       288    102816   83  Linux
/dev/hdb4           289       665    190008    5  Extended
/dev/hdb5           289       492    102784+  83  Linux
/dev/hdb6           493       584     46336+  83  Linux
/dev/hdb7           585       665     40792+  83  Linux
Command (m for help):
```

棒极了。现在使用w命令（把分区表写到硬盘并退出）使修改生效并退出 fdisk程序，如下所示：

```
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
[root@ford /root]#
```

如果需要我们根据什么配置情况自行编写一个 /etc/fstab文件，它的内容看起来应该和下面的差不多：

```
/dev/hdb1       /           ext2     defaults    1 1
/dev/hdb2       swap        swap     defaults    0 0
/dev/hdb3       /usr        ext2     defaults    1 2
/dev/hdb5       /home       ext2     defaults    1 2
/dev/hdb6       /var        ext2     defaults    1 2
/dev/hdb7       /tmp        ext2     defaults    1 2
none            /proc       proc     defaults    0 0
none            /dev/pts    devpts   mode=0622   0 0
```

8.3.3 建立文件系统

建立好分区以后，需要在其上建立文件系统（如果你对微软的 Windows很熟悉，就会看出建立文件系统的过程类似于对磁盘进行格式化的操作）。

Linux操作系统中，我们使用两个工具程序来完成这个过程： mke2fs建立ext2文件系统；mkswap建立swap文件系统。mke2fs工具有许多命令行参数，但一般只有在特殊情况下才用得着。而如果你真的遇到了特殊情况，我敢说你也用不着按照本小节的指导建立文件系统了！

经常使用的命令行参数只有一个，就是在其上建立文件系统的分区名称。如果想在分区/dev/hdb3上建立一个文件系统，需要输入下面的命令：

```
[ root@ford /root ] # mke2fs /dev/hdb3
```

命令执行的结果和下面的内容差不多：

```
[root@ford /root]# mke2fs /dev/hdb3
mke2fs 1.14, 9-Jan-1999 for EXT2 FS 0.5b, 95/08/09
Linux ext2 file system format
File system label=
25792 inodes, 102816 blocks
5140 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
13 block groups
8192 blocks per group, 8192 fragments per group
1984 inodes per group
Superblock backups stored on blocks:
        8193, 16385, 24577, 32769, 40961, 49153, 57345, 65537, 73729, 81921,
        90113, 98305
Writing inode tables: done
Writing superblocks and file system accounting information: done
[root@ford /root]#
```

使用mkswap命令设置swap空间也同样简明。唯一的参数是在其上建立 swap空间的分区名称。如果想在/dev/hdb2上建立swap空间，需要输入下面的命令：

```
[ root@ford /root ] # mkswap /dev/hdb2
```

命令执行的结果和下面的内容差不多：

```
[root@ford /root]# mkswap /dev/hdb2
Setting up swapspace version 0, size = 17027072 bytes
[root@ford /root]#
```

8.4 网络文件系统

网络文件系统可以让你在使用客户端处理用户大计算量任务的同时对外提供磁盘存储服务。集中的硬盘意味着简单化的备份解决方案和现实的安全性。在 Linux（和作为一个整体的UNIX）中，硬盘的集中是通过网络文件系统（ Network File System，NFS）来实现的。在本小节中，我们将讨论NFS客户端的问题，服务器端的问题留到本书的后面（第 18章）讨论。

8.4.1 挂装NFS分区

挂装NFS分区与挂装本地分区的过程几乎是一样的。唯一的区别在于分区是如何确定的。本地硬盘上的分区由它们的设备名确定，比如 /dev/hda1。对NFS挂装来说，分区由它们的主机名和开放子目录名称来确定。如果名字是 technics的服务器允许你的主机挂装它的/export/SL1200/MK2子目录，你又想把它挂装到 /projects/topsecret1，那么就需要输入下面的

命令：

```
[ root@ford /root ] # mount technics:/export/SL1200/MK2 /projects/topsecret1
```

正如你所看到的，给本地子目录取名时也不必与服务器上的子目录名字相同。

NFS又引入了一些可以与mount命令的-o参数联合使用的参数，如表8-6所示。

下面是/etc/fstab文件中的一个NFS挂装数据项示例：

```
denon:/export/DN2000F /proj/DN2k nfs bg, intr, hard, wsize=8192, rsize=8192 0 0
```

表8-6 与mount命令的-o参数联合使用的参数

mount -o命令的参数（NFS分区）	说 明
soft	“软挂装”该分区。如果服务器没有响应，客户端会倒计时一个预定的期间，并撤消请求的操作
hard	“硬挂装”该分区。如果服务器没有响应，客户端会一直等下去直到服务器恢复为止。如果服务器恢复了，不会有任何数据丢失
timeo=n	把倒计时时间设置为n秒
wsize=n	把写缓冲区大小设置为n个字节。缺省值是1024；推荐值是8192
rsize=n	把读缓冲区大小设置为n个字节。缺省值是1024；推荐值是8192
bg	让挂装操作在后台运行。如果在开始的时候挂装不上，就可以把挂装进程放到后台去继续尝试。如果在 /etc/fstab文件中有NFS挂装点，就必须包括这个参数

8.4.2 使用Automounter自动挂装子系统

随着站点的成长，你会发现维护管理挂装数据表变得越来越复杂。也许你会觉得为所有系统都固定地建立一套标准的 NFS挂装比较合适，但是这也有副作用，会同时浪费服务器与客户机的系统资源。

为了解决这个问题，可以使用 Automounter自动挂装子系统。正如它的名字所揭示的，它会自动地在需要的时候把分区挂装上。当与 NIS（请参考第19章）联合使用时，你可以制定一个集中的适用于你整个的站点映射图，让 Automounter自动挂装子系统只在用户需要挂装某个分区的时候才自动地工作。

如今，所有流行的Linux发行版本在发行时都已经把 Automounter自动挂装子系统预先设置好了，就等用户自己制定的配置文件了。

1. 启动Automounter

因为Automounter依赖于NFS，因此在使用它之前一定要肯定已经能够进行普通的 NFS挂装操作了。只要能够正常进行 NFS挂装，在引导启动命令脚本程序中做一个简单的改动来调用Automounter：确认autofs脚本程序（可以在Red Hat Linux的/etc/rc.d/init.d中找到）在系统进入运行级别3的时候被启动。你可以使用诸如 tksysv或者ksysv之类的图形化运行级别编辑器完成这个工作，也可以自行在 rc3.d子目录中添加到init命令脚本程序的符号链接。记住要把符号链接的号码设置得足够大，使之在网络系统都启动了之后才启动（在 Red Hat Linux中，你可以建立一个从/etc/rc.d/rc3.d/S72autofs到/etc/rc.d/init.d/ autofs的符号链接）。

如果想手动启动Automounter，需要输入下面的命令：

```
[ root@ford /root ] # /etc/rc.d/init.d/autofs start
```

使用同一个命令还可以停止这个守护进程的运行，只要把 start参数换成stop就可以了。

2. 配置/etc/auto.master文件

Automounter的主配置文件是/etc/auto.master文件。这个文件的格式如下所示：

```
#
# Sample /etc/auto.master file
# (lines which begin with a '#' are comments)
#
/mount/point          map-file        global-options
/home             auto.home
/usr/local        auto.local
/misc             auto.misc
```

文件中的第一列是挂装点；第二列给出的是那个挂装点映射细节的文件；最后一列是作用于全部挂装点的NFS参数。

映射文件与auto.master网络略有不同，比如说，下面给出了一个 auto.misc文件：

```
#
# auto.misc
#
# This is an automounter map and it has the following format
# key [ -mount-options-separated-by-comma ] location
# Details may be found in the autofs(5) manpage
kernel          -ro,soft,intr           ftp.kernel.org:/pub/linux
cd              -fstype=iso9660,ro      :/dev/cdrom
# the following entries are samples to pique your imagination
#floppy         -fstype=auto            :/dev/fd0
#floppy         -fstype=ext2            :/dev/fd0
#e2floppy       -fstype=ext2            :/dev/fd0
#jaz            -fstype=ext2            :/dev/sdc1
```

这个文件的第一列是关键字；第二列保存着挂装参数；最后一列是挂装开始的设备名。如果设备名数据项是以冒号（:）打头的，就给系统一个信号，表明这个挂装是本机上的。因此，/misc/cd将按照下面的挂装参数挂装/dev/cdrom设备：

```
-fstype = iso9660, ro
```

注意 NFS要求在挂装操作执行之前，相关的子目录就应该已经建立好了；但是Automounter与它不一样。就拿 /misc挂装点来说，唯一需要事先建立好的只有 /misc子目录，其他下级子目录会由 Automounter守护进程根据需要自动建立。

如果在 Automounter运行的时候对任何一个映射文件都作了改动，就必须重新启动Automounter，让它重新读自己的配置文件。这个操作是使用 autofs命令完成的。如下所示：

```
[ root@ford /root ] # /etc/rc.d/init.d/autofs reload
```

8.5 硬盘空间配额的管理

无论在哪一种多用户环境里，你都有可能遇到这样的用户：他们 —— 或者是因为拒绝公

平的游戏规则，或者是缺乏公德心——浪费硬盘资源，占用超出合理数量的硬盘空间。这个问题可以有好几种办法来解决。第一个也是最容易想到的办法是讲道理，这个方法很少能够奏效。第二招是公众目光压力，定期公布这些用户侵占的硬盘空间容量，如果他们对其他人的看法还敏感的话，这一招可能会有效。最后也是最成功的一个办法是：实行硬盘空间配额管理，每个用户只允许消耗一定数量的硬盘空间，不允许超标。强制性管理由操作系统负责。

硬盘配额通常是系统管理员最好的选择，因为配额管理是自动进行的，不需要面对愤愤不平的用户。但是，技术或政治方面的障碍可能会阻碍实行这个办法（比如说，首席执行官可能就不喜欢别人告诉她只能使用5MB的服务器硬盘空间）。当然，配额也是可选用的——如果没有它们，你的系统也可能跑得挺得意。它们是否能够在你的系统上产生效果完全取决于你本人和你的管理技巧。

8.5.1 实现配额管理的准备工作

实行硬盘配额的过程分为两个步骤。第一步是配置系统使用硬盘配额的一次性设置工作，通过这个设置把实行配额管理所必须的软件安排就位，以后每次开机时就进入预备状态，第二步是在/etc/fstab文件中进行必要的设置，并在每个准备实行硬盘配额管理的分区里加上必须的文件。第二步的工作需要对每一个准备实行硬盘配额管理的分区分别进行设置操作。

1. 设置引导过程

大多数Linux发行版本（包括Red Hat）都已经把对配额管理的支持作为标准安装的一个部分预先设置好了。因此不必再对引导过程进行什么改动了。但是，如果确实需要修改引导脚本程序的话，一定要先学习LILO和rc脚本程序那一章（第7章）。在这里，我们假设读者都已经对这两个问题相当熟悉了。

如果想启动配额管理程序，在用户的引导脚本程序末尾（ rc.local是个不错的选择）加上下面这几行语句：

```
# Check quota and then turn quota on.
  if [ -x /usr/sbin/quotacheck ]
          then
                  echo "Checking quotas. This may take some time."
                  /usr/sbin/quotacheck -avug
                  echo " Done."
          fi
          if [ -x /usr/sbin/quotaon ]
          then
                   echo "Turning on quota."
                   /usr/sbin/quotaon -avug
          fi
```

重新启动系统，测试上述修改已经起效，系统可以在没有人工干预的情况下回到正常的工作状态。

2. 配置各个分区

硬盘配额是对每个用户或者每个用户分组起作用的，不同的硬盘分区对他们可以实行不同的配额。举例来说，某用户可以在甲分区上有一个与其登录子目录相关联的配额，还可以在乙分区有另外一个与其所属用户分组相关联的配额。

对每个准备实行硬盘配额管理的分区，需要进行三项设置： usrquota参数、grpquota参数和配额数据库。

- **usrquota参数** 编辑/etc/fstab文件，在每一个需要配额的硬盘分区的挂装参数里加上usrquota参数。举例来说，假设/dev/hda5挂装到/home子目录，现在准备对这个子目录设置用户配额，/etc/fstab文件中的对应数据项应该修改为如下所示的样子：

```
/dev/hda5    /home    ext2    defaults, usrquota  1    1
```

- **grpquota参数** 需要用户分组配额支持的硬盘分区对应的 /etc/fstab文件中的数据项应该有grpquota参数，与刚才介绍的 usrquota参数一样（请注意同时设置这两个参数是可以的）。举例来说，假设/dev/hda5挂装到/home子目录，现在准备对这个子目录设置用户分组配额，/etc/fstab文件中的对应数据项应该修改为如下所示的样子：

```
/dev/hda5    /home    ext2    defaults, grpquota  1    1
```

- **配额数据库** 建立相应的数据库文件保存用户和用户分组配额信息。这些文件是空的，它们被放置在每一个实行配额的硬盘分区的根目录下；只有根用户才能读取它们的内容；而且必须给它们起名为quota.user和quota.group。

如果想按照适当的访问权限建立这些文件，需要使用 su命令转换为根用户身份，再使用touch命令建立这些保存配额数据库资料的文件。举例来说，为了对挂装在 /home子目录上的/dev/hda5分区设置配额控制，需要使用 cd命令把路径切换到子目录 /home，再输入下面的内容：

```
[root@ford /root]# cd /home
[root@ford /home]# touch quota.user
[root@ford /home]# touch quota.group
[root@ford /home]# chmod 600 quota.user
[root@ford /home]# chmod 600 quota.group
```

接下来就可以配置个别配额设置值了。

注意 当第一次打开配额功能的时候，因为原始的数据库文件还是空的， quotaon命令会报告quota.user和quota.group文件中有非法数据。不必担心，在使用 edquota命令往数据库文件中添加至少一个用户之前，这条消息会反复出现。

8.5.2 设置配额

无论是对用户还是对用户分组，建立、修改和删除配额的操作都是由 edquota命令来完成的。在本小节，我们将学习这个命令的使用方法，并且给出一些具体的操作示例。首先学习几个术语：

- **软限制** 这个限制作用于用户或者用户分组。如果用户的账户超过了软限制，就开始进入限制期，也就是这个账户在超过软限制后还能存在多长的时间。在这个期间，用户会收到警告他们的账户超标的消息。
- **硬限制** 这个限制是由操作系统实行的，不允许超过。试图在硬限制以外进行数据写操作会被拒绝。
- **限制期**（时间限制） 当用户的账户超过软限制的时候，时钟开始计时。在限制期结束

之前，用户是无法再访问这个账户的。这个限制期的长度因系统而异，通常会是一个星期。如果不想让这个账户被禁用，用户需要删除或者压缩文件，使自己的硬盘空间消耗量降低到软限制以下。

1. edquota的命令行参数

当管理单个用户的配额时， edquota命令只有三个参数，如下所示：

参　数	说　明
-u *login*	为参数定义的用户设置配额数据
-t	为硬盘分区设置限制期。把它与 -u或者-g参数联合使用可以分别为用户或者用户分组全部设置好限制期。请注意：如果用户 /用户分组在同一个分区上，它们就不能有不同的限制期
-g *group*	为参数定义的用户分组设置配额数据
-p *login*	允许把一个用户的配额资料（用户名为 login）克隆给另外一个用户。这个参数必须与 -u参数合用

当调用edquota命令的时候，它会根据命令行参数调出相应的用户、用户分组或者限制期资料进行增减或修改。缺省的情况下， vi编辑器用来把这些资料显示为易于理解的格式。如果你喜欢另外的编辑器，比如 pico或者emacs，可以在执行 edquota命令之前改变环境变量EDITOR的值。比如说，如果你正在使用的 shell是BASH（根用户的缺省 shell），需要输入下面的命令：

```
[ root@ford ] # EDITOR = pico；export EDITOR
```

edquota命令把资料显示到屏幕上以后，编辑它们，然后保存文件，再退出编辑器程序。edquota进程会接手那些资料并把它保存到数据库里。请注意： edquota命令使用了/etc/passwd文件来确定用户登录子目录的位置。

2. edquota命令示范

为了给用户heidi（我网络上有名的硬盘黑洞）加上配额，使用下面的命令：

```
[ root@ford ] # edquota -u heidi
```

这个命令调用编辑器程序打开文件，显示如下所示的内容：

```
Quotas for user heidi:
  /dev/hda2: blocks in use: 0, limits (soft = 0, hard = 0)
             inodes in use: 0, limits (soft = 0, hard = 0)
```

各项限制的值都是0，表示此时在heidi的账户上还没有设置任何配额。请注意：对 i-结点和数据块都可以进行限制。别忘了 Linux中的数据块的长度是1K字节，而i-结点则是保存文件所必须的控制信息。每个文件一般只需要几个 i-结点，文件长度加大时再增加。

好了，我准备把heidi的数据块软限制设置为5000、数据块硬限制设置为6000、i-结点软限制设置为2500、i-结点硬限制设置为3000。下面就是输入完成后文件的样子：

```
Quotas for user heidi:
  /dev/hda2: blocks in use: 0, limits (soft = 5000, hard = 6000)
             inodes in use: 0, limits (soft = 2500, hard = 3000)
```

警告　edquota命令编辑这些信息时会给其临时文件起一个唯一的名字。保存这个临

时文件 —— 不要把这些信息写到quota.user或者quota.group文件上！edquota命令会把这些信息自动调整到这两个文件里去。

要是想修改heidi的配额数值，我可以再使用 edquota -u heidi命令修改那些设置值。如果Heidi和我结了婚，再给她的账户设置配额好像就不太合适，那么我可以把配额数值都改回 0。

接下来我们介绍怎样使用 edquota把某个用户的设置值克隆给另外一个用户。举例来说，我们打算克隆用户jyom，对用户ebosze也分配同样的配额数值。需要输入下面的命令：

```
[ root@ford ] # edquota -p jyom ebosze
```

再看下面的例子。如果想一次性完成对大批用户的配额的设置工作，可以使用一行脚本程序语句。假设这批用户的 UID值都大于（或者等于） 500，并且已经手动设置了至少一个用户（我们再假设这个用户的用户名是 artc）的配额数值，那么下面的语句就可以魔术般地达到我们的目的：

```
[ root@ford ] # edquota -p artc ` awk -F: '$3 > 499 ( print $1 ) ' /etc/passwd `
```

8.5.3 管理配额

激活配额功能并正常运行以后，另有三个可以帮助我们管理硬盘配额的工具程序，它们是quotacheck、repquota和quota。

quotacheck命令检查配额数据库的完整性。在我们前面给出的打开配额子系统的脚本程序里，在使用quotaon激活配额之前我们就使用这个技术进行了检查。

可以传递到quotacheck的参数如下所示：

-v	打开报告模式。再检查配额数据库的时候就会看到许多有用又有意思的信息
-u *uid*	检查UID是uid的用户的配额情况
-g *gid*	检查GID是gid的用户的配额情况
-a	检查所有设置了配额的文件系统（以 /etc/fstab文件中的设置为准）
-R	与-a参数合用。检查所有设置了配额的硬盘分区，但是不包括根分区

repquota命令用来生成系统上配额使用情况的统计报告。这个命令有下面几个参数：

-a	统计所有文件系统的配额使用情况
-v	统计所有配额的使用情况，没有用到的也要统计
-g	以用户分组为单位统计配额使用情况
-u	以用户为单位统计配额使用情况

最后，quota命令是供用户使用的。这个命令可以让用户查看分配给自己的配额。它有下面几个参数：

-g	给出用户所在分组的配额使用情况
-u	给出该用户的配额使用情况（缺省操作）
-v	给出支持配额的所有文件系统里与该用户有关的配额使用情况
-q	如果该用户已经超标，显示一个消息给他

举例来说，如果用户sshah输入了quota -v命令，会看到下面的内容：

```
[sshah@ford ~]% quota -v
Disk quotas for user sshah (uid 500):
     File system  blocks   quota   limit   grace   files  quota  limit  grace
      /dev/hda8        0       0       0               0      0      0
      /dev/hdb1        0    5000    5100               0   1000   1100
```

8.6 小结

在本章中，我们讨论了对文件系统进行管理的流程：从如何建立开始，到实行配额结束。掌握了这些知识，你就具备了在多变的环境中对商业应用水平的服务器进行管理所必须的能力。

与其他操作系统一样，Linux也在不断的变化发展当中。虽然文件系统的设计人员和维护人员都尽了最大的努力来维持界面不发生变化，你还是可以看到许多的改进不断地涌现出来。有些变化使界面更简单，有些变化则是文件系统本身的戏剧性进步。请注意观察这些变化。Linux提供了一流的文件系统，它是健壮的、反应敏捷的，使用起来也很顺手。拿起我们在本章介绍的工具，自己去寻宝吧。