第3章　磁盘和其他存储媒体的使用

在安装或升级自己的系统时，必须在磁盘上做的事情太多了。必须在磁盘上制定文件系统，以便把文件保存在上面，并为系统中的各个部件预留空间。

本章将详细讨论所有的这些初始化活动。通常情况下，系统一旦设置好，你不必从头开始一切初始化活动，使用软盘的时候除外。如果你要加一个新磁盘或打算优化自己的磁盘使用率的话，一定不要错过本章。

管理磁盘过程中的基本任务是：

■ 格式化磁盘。它将做各种各样的“杂活”，为磁盘的使用做好准备，比如检查坏磁道（目前，许多硬盘都不需要格式化了）。

■ 对硬盘进行分区（如果你想把硬盘用于不同活动的话，因为不希望看到这些活动相互干扰）。之所以要分区，其原因之一是在同一个磁盘上保存不同的操作系统。另一个原因是把用户文件和系统文件分开，从而简化备份，并有利于防止系统文件受损。

■ 在每个磁盘或分区上制作恰当的文件系统。如果没有文件系统，对 Linux来说，磁盘是毫无意义的；有了文件系统，才能在它的基础上创建并访问文件。

■ 装入不同的文件系统，形成单一的树形结构，根据需要，自动或手动装入（手动装入的文件系统通常也需要手动卸载）。

第4章包含了虚拟内存和磁盘缓冲的有关信息，在使用磁盘时，必须知道这两个概念。

3.1　两类设备

Unix和Linux能够识别两类不同的设备：随机访问块设备（比如磁盘）和字符设备（比如磁带和串行线路），还有一些既是串行线路又是随机访问的设备。已获支持的每个设备在文件系统中，都以一个设备文件的形式表达出来。在你读或写一个设备文件时，数据就来自或发送到它所代表的设备中。通过这种方式，不再需要特定的程序（也不需要特定的应用程序编程方法，比如捕获中断或修剪串行端口）来访问设备；举个例子来说，如果打算向打印机发送一个文件，只须这样：

```
$ cat filename > /dev/lp1
$
```

文件内容就打印出来了（当然，这个文件必须采用打印机能够识别的形式）。但是，由于数名用户同时把自己的文件交给打印机并不是件好事，因此，一般都要用一个特殊的程序来发送准备打印的文件（这个程序一般是 lpr）。该程序保证一次只打印一个文件，只要当前文件打印一结束，就自动把下一个文件送入打印机。这一点和多数设备类似。事实上，人们根本没必要去操心设备文件。

由于设备以文件的形式保存在文件系统中（在 /dev目录下），所以，利用 ls或另一个适当命令，马上就能知识有哪些设备文件。在 ls -1的输出中，第一列中包含文件类型及其访问权限。例如，要检查我本人的系统所提供的串行设备

```
$ ls -1 /dev/cua0
crw-rw-rw- 1 root uucp 5,  64 Nov 30  1993 /dev/cua0
$
```

第一列中的第一个字符，也就是crw-rw-rw-中的“ c ”，告诉用户该文件的类型，这个例子中，表示该文件是一个字符设备。对普通文件来说，第一个字符是“ - ”；对目录来说，则是“ d ”；对块设备来说则是“ b ”；更多详情，请参考ls手册页。

注意，通常情况下，所有设备文件都是存在的，即使设备本身可能还没有安装。所以，如果你有一个文件/dev/sda，并不意味着你真的有一块SCSI硬盘。拥有所有设备文件是为了使安装程序更为简单，便于增加新的硬件设备（不必为此查找正确的参数，就可为新增设备创建设备文件）。

3.2 硬盘

这部分将介绍和硬盘有关的术语。如果你完全了解并掌握了相关的术语和概念，可跳过这部分。

图3-1展示了硬盘的各个重要部分。硬盘由一个或多个圆形的“ 盘片 ”组成（盘片是由硬的物质（比如铝）制成的，这就是硬盘一词的由来），盘片的单或双面涂有磁，用于记录数据。每面都有一个“ 读写头 ”（又称磁头），用于检查或修改记录下来的数据。盘片围绕着一个普通的轴旋转；一般转速为每分钟 3 600转，高性能的硬盘的转速更高，甚至可达 15 000转。读写头沿盘片半径移动，这种随着盘片旋转的移动允许读写头对表层上每个部分进行访问。

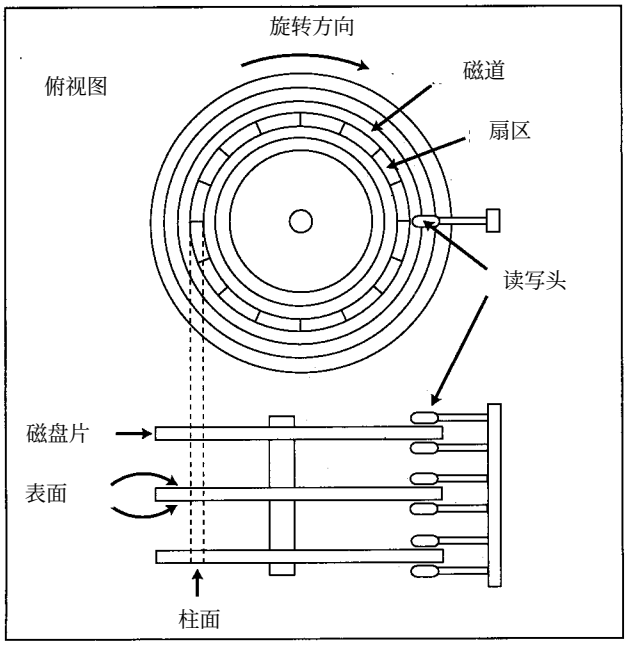

图3-1 硬盘图解

处理器（cpu）和实际的磁盘是通过一个“ 磁盘控制器 ”来进行沟通的。这样一来，计算机的其余部件无须了解如何使用驱动器，因为可以指示不同类型磁盘的控制器，令它们使用

和计算机其余部件相同的接口。因此，计算机就可以只是简单地发出一条指令，说：“嗨，磁盘！把我需要的东西给我！”，用不着产生一系列冗长和复杂的电子信号，将磁头移至正确的位置，并等候自己需要的数据“转”到磁头下，以及做其他一些非常麻烦和令人不快的事情。实际上，到控制器的接口仍然相当复杂，但和控制器本身相比，却要简单得多。此外，控制器也可以做其他的事情，比如高速缓存以及替换坏掉的扇区等。

上面就是通常需要了解的硬件知识。此外还有许多东西，比如推动盘片旋转和移动磁头的电机和控制机械部分操作的电子元件等，但大多与硬盘的工作原理关系不大。

表层通常分为若干个同心圆，就是通常所说的磁道，而这些磁道又依次分为若干个扇区。这样的划分用于指定硬盘上的位置，以及为文件分配磁盘空间。要想找出文件在磁盘上的具体位置，只须这样指出“表层3，第5磁道第7扇区”即可。通常情况下，所有磁道上的扇区号都是一样的，但有些硬盘在外部磁道上的扇区要多一些（所有扇区的物理大小都是一样的，所以，外部磁道可容纳更多的扇区）。一般说来，一个扇区可容纳512个字节的数据。磁盘本身不能处理数额少于一个扇区数的数据。

每个表层分为若干个磁道（和扇区）的方式都是一样的。这意味着一个表层的磁头位于一个磁道上时，另一个表层的磁头也会在相应的磁道上。聚集在一起的所有对应磁道就称为一个柱面。磁头从一个磁道（柱面）移到另一个磁道上是要花时间的，所以要把经常访问的数据放在一起（比如文件），使它们在同一个柱面内，这样，磁头就不必移来移去了（既花时间，又有损磁头），从而改进性能。文件被分存在磁盘上的多个地方称之为“碎片”，这样放置文件是绝不允许的。

表层（或磁头）、柱面和扇区的编号区别很大；各个编号的指定称为硬盘的“几何信息”。几何信息通常保存在一个特殊的、由电池驱动的内存区域（称为CMOS RAM）内，操作系统在启动时或驱动程序初始化时，从这个区域取出这些几何信息。

但遗憾的是，BIOS（BIOS是保存在ROM芯片上的内置软件，它和其他部分一起，负责启动过程中的初始化阶段）有一个设计上的缺陷，令其不可能指定超过1024的磁盘编号，这个值是保存在CMOS RAM中的，对一个大容量的硬盘来说，这个数太小了。为了克服这一局限，硬盘控制器就几何信息撒了谎，把计算机指定的地址解释为更适用的数值。例如，硬盘就可以有8个磁头、2048个磁道，以及每个磁道有35个扇区（这些编号完全是我假想的）。

硬盘控制器可对计算机撒谎，声称自己有16个磁头，1025个磁道以及每个磁道有35个扇区，由此无法突破限定值，所以就把计算机为它指定的地址进行了解释，具体做法是减半磁头数，加倍磁道数。现实中更为复杂，因为数目不会像这里的示例一样好用（但再次提醒大家注意，其细节问题于硬盘的基本原理是不相干的）。这种解释歪曲了操作系统对磁盘组织形式的认识，因此，要想用所有数据都在一个柱面的磁道来提升性能是不切实际的。

这种解释在IDE磁盘上出现了问题。SCSI磁盘为CPU和控制器之间的沟通，采用了一个连续性的扇区号（也就是说，控制器把一个连续性的扇区号解释为磁头、柱面和扇区这种三个一组的序号）和一个完全不同的方法，用来解决这一问题。但是要注意，计算机可能也不了解SCSI磁盘的实际几何信息。

由于Linux系统常常不了解磁盘的几何信息，即使其文件系统也无法尝试把文件保存在一个单独的柱面内。相反地，它会努力把后来的扇区分配给文件，这样也可获得性能的改进。这个问题由于控制器上缓冲，而由控制器预先自动取得文件而复杂化。

每个硬盘都是以一个独立的设备文件来表示的。一台计算机上可以（通常）有 2到4个IDE硬盘。它们分别命名为/dev/hda、/dev/hdb、/dev/hdc和/dev/hdd。SCSI硬盘则命名为/dev/sda、/dev/sdb等等。类似的命令约定适用于其他类型的硬盘。注意，可通过硬盘的设备文件来访问整个磁盘，与分区无关（随后将详细讨论），稍不注意，分区或分区内的数据就会一团糟。磁带的设备文件通常只用于访问主引导记录（请参见随后的讨论）。

3.3 软盘

软盘是一个圆形而柔软的塑料薄片。它的一面或两面覆盖着铁氧化物颗粒。和硬盘类似，这些颗粒也具有磁性。软盘本身并没有读写头，那是由软盘驱动器来提供的。可将软盘想象成硬盘中的一个盘片，但前者却是可以抽取的。换言之，用同一个软盘驱动器，可访问许多不同的软盘。用完一张，换上另一张即可。硬盘却不同，它和硬盘驱动器是一个牢可不分的整体。

与硬盘类似，软盘表面也划分为不同的磁道和扇区（软盘两侧两个对应的磁道构成了一个“柱面”）。但是，这些磁盘和扇区的数量要比硬盘少得多。

软盘驱动器通常可读写几种不同类型的磁盘。举个例子来说，一部 3.5英寸的驱动器即可使用720K的软盘，亦可使用 1.44MB的软盘，有的甚至能读写 2.88MB的软盘。由于驱动器必须以不同的方式工作，而且操作系统必须知道磁盘的容量有多大，所以对软盘来说，需要同时运行几个设备文件。每个设备文件都对应于驱动器及特定磁盘类型的一个组合。换言之，/dev/fd0H1440表明是第一个软盘驱动器（fd0），它肯定是一个3.5英寸的驱动器，使用3.5英寸的高密盘（H），磁盘容量为1440KB（1440）。我们常用的3.5英寸有壳小软盘便属于这种类型，它的格式化容量为1440KB或者1.44MB。

但是，软盘本身的种类却非常多，所以 Linux设计了一种特殊的软盘设备类型，能自动侦测驱动器内插入的磁盘属于何种类型。它会试着读取一张新插入的软盘的第一个扇区，并与不同的软盘类型对照，直到找到相符的那一种为止。自然地，这要求软盘首先必须完成格式化。这些自动设备叫作/dev/fd0，/dev/fd1，等等。

自动设备用于访问一张盘的参数亦可通过 \cmd{setfdprm}程序加以设置。假如需要使用一张不属于任何标准软盘容量规格的磁盘，该程序便显得相当有用。换句话说，假如一张盘包含了非标准的扇区数量，或者自动侦测过程由于某种原因而失败，找不到相符的设备，那么请考虑使用该程序。

除全部标准类型之外， Linux也有能力支持许多非标准的软盘格式。其中有一些要求使用特殊的格式化程序。尽管从现在开始，我们会忽略这些磁盘类型的存在，但大家如果有兴趣的话，可以参考一下/etc/fdprm文件。该文件指定了setfdprm程序能够识别的设置。

注意一旦在软盘驱动器中插入一张新盘，必须让操作系统知道。这样做有许多好处，其中最重要的一点便是避免操作系统继续使用前一张盘的缓存数据。但不幸的是，用于此目的的信号线经常出故障。而且更糟糕的是，在 MS-DOS操作系统中，换盘的操作往往被“安静”地忽略了。假如你在使用软盘时，经常出现顽固记忆前一张软盘内容的情况，请考虑更换驱动器。

3.4 CD-ROM

CD-ROM（只读光盘）驱动器利用光学原理，读取塑料做成的、表面涂有光学反射层的

光盘（CD）。二进制信息记录在肉眼看不见的一系列小凹坑上。这些凹坑呈螺旋形状，自光盘中心向外圈扩散。光盘驱动器最重要的部件就是光学读取机构，或称“光头”。它利用一束激光，沿着盘面读取以一系列凹坑形式保存的数据。激光射中一个坑时，会造成激光反射方向的变化；若射中的是正常的平滑盘面，则又向另一个方向反射。这样一来，二进制数据可以很容易地表现出来。知道了原理，剩下的事情就非常简单了，纯粹是一些机构方面的问题。

和硬盘驱动器相比，CD-ROM驱动器的速度要慢得多，但又要比软盘驱动器快得多。对典型的硬盘驱动器来说，其平均寻道时间通常都在 10ms以下。但对CD-ROM驱动器来说，它的寻道时间一般都是 100多毫秒。目前市面上见到的光驱一般都能达到每秒数 MB的数据传输速度（24倍速以上，一倍速度为每秒 150KB）。但再快的光驱，都无法取代硬盘。除了速度比不上，不能写入，是其最致命的弱点。注意在目前发行的一些 Linux版本中，在光盘上刻录了“可直接使用”的文件系统。换言之，不必将文件复制到硬盘，便可在光盘上直接使用 Linux。这样一来，安装变得更加简单，并能节省大量磁盘空间。安装新软件的时候，CD-ROM是一个很好的载体，因为安装文件的长度一般都很大，而且速度在安装时似乎并不是最重要的。

有几种方式在CD-ROM盘片上记录数据。最常见的一种是由国际标准 ISO 9660规定的。该标准定义了一个非常小、非常粗略的文件系统，甚至比 MS-DOS采用的文件系统还要简单。但在另一方面，每种操作系统都能够将 CD-ROM上的文件系统转换成自己的格式，以便顺利地读出光盘上保存的数据。

对普通的Unix应用来说，ISO 9660文件系统似乎并不大管用。为此，后来又在该标准的基础上开发了一套扩展，名为 Rock Ridge。它允许使用长文件名、符号链接以及其他大量有用的东西。这样一来，CD-ROM才能真正在Unix文件系统中安家落户。甚至还有更让人高兴的，Rock Ridge仍然属于一种合法的ISO 9660文件系统，所以那些非 Unix系统一样可以使用它。Linux同时支持ISO 9660和Rock Ridge扩展。注意假如光盘采用了这种扩展，那么 Linux能自动侦测出来，并自动以恰当的格式使用。

然而，文件系统只是顺利读写光盘数据的先决条件之一。除此以外，对大多数光盘包含的数据来说，它们都要求一个特殊的程序来访问。而且大多数这样的程序都不能在 Linux中运行（Linux的MS-DOS仿真程序除外）。

CD-ROM驱动器是通过对应的设备文件来访问的。有几个办法可将一部 CD-ROM驱动器同计算机连接起来 —— 通过SCSI、通过声卡上带的 IDE接口或者通过EIDE接口等等。至于更详情的情况，已超出了本书的范围。大家在此唯一需要注意的是，你选择的连接类型决定了最终使用的设备文件！

3.5 磁带

表面涂有磁性材料（允许记录数据）的聚脂膜薄带。磁带必须顺序读写，不能像软盘和硬盘一样随机读取。所以它的读取速度比后者慢得多。用于在磁带上读取数据的设备是磁带机或磁带驱动器。

虽然如此，但磁带的价格相对便宜，而且保存的时间更久，其中可保存大量的数据，所以磁带非常适合于归档和备份这些不过分追求速度但需要低廉而大存储能力的任务。

3.6 格式化

格式化是指为了使磁盘能够存储数据而对其进行初始化的过程，用于标记磁道和扇区。在磁盘格式化之前，其磁表面完全只是一些磁信号。格式化之后，一切就开始井然有序。需要记住的是磁盘未经格式化，是不能用的。不过，目前，大多数磁盘都不需要你自己格式化。

有一点是大家容易混淆的： MS-DOS中，“ 格式化 ”一词还包括了创建文件系统的进程（随后将详细讨论）。两个进程通常组合在一起，尤其是软盘。需要区分的是，实际上的格式化称为“ 低级格式化 ”，而制定文件系统则称为“ 高级格式化 ”。Unix中，两者分别称为格式化和制定文件系统，本书将采用这一说法。

对IDE和一些SCSI磁盘来说，格式化事实上是在工厂进行的，不需要你自行重新格式化；因此，许多人用不着操心它。事实上，格式化硬盘可令其运行得不如格式化以前好，可能是因为磁盘需要以某种特殊的方式格式化，以便允许自然损坏的扇区恢复工作吧。

需要或可以格式化的磁盘通常需要一个特殊的程序，因为与驱动器内部的格式化逻辑沟通不同于驱动器与驱动器之间的沟通。格式化程序通常不是在控制器 BIOS上，就是以MS-DOS程序的方式提供；这两者都不能用于Linux。

格式化过程中，可能会碰到磁盘上介质被损坏或欠缺，这些就称为“ 坏块 ”或“ 坏扇区 ”。它们有时由驱动器自己处理，但如果其数量越来越多，就需要采取一定的行动避免使用这些坏掉的扇区。用于这个目的的逻辑信息内置于文件系统中；随后将讨论如何在文件系统内增加这一信息。另一种办法是：可创建一个小的分区，只覆盖坏掉的磁盘；如果坏掉的磁盘介质较大，这种方法可能较好，因为磁盘大面积损坏时，文件系统有时会出现问题。

软盘的格式化是用fdformat来进行的。准备使用的软盘设备文件是作为参数给出的。例如，下面的命令将对第一个软驱内的一张高精度的 3.5英寸的软盘进行格式化：

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
$
```

注意，如果你打算使用一个自动侦测设备（比如 /dev/fd0），就必须先用setfdprm设置该设备的参数。为了获得上面的效果，必须像下面这样：

```
$ setfdprm /dev/fd0 1440/1440
$ fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
$
```

通常情况下，选择一个恰当的、与软盘类型匹配的设备文件更为方便。注意，格式化软盘，使其容纳超过其容量的更多数据是极不明智的。

除此以外，fdformat还将校验软盘，也就是说，检查它是否有坏扇区。它将多次尝试修复坏扇区（你肯定不会想不起驱动器噪音的变化吧）。如果软盘只有少许损伤（由于读写头上的灰尘，有些错误并不代表真正的损坏），fdformat是可以应付的，但如果是真的损坏，校验进程就会被终止。内核将打印出它找出的各个I/O错误的日志消息；这些消息将发送到控制台，如果正在使用syslog（系统日志）的话，这些消息就会被发送到 /usr/log/message文件。fdformat本身不会说

出错在何处（人们通常也不在意，反正软盘便宜，买张新的，也不过2、3元）。

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... read: Unknown error
$
```

badblocks命令可用来搜索任何磁盘或分区上的坏块（包括软盘在内）。它不格式化磁盘，所以还可用于检查现成的文件系统。下面的示例对一张5英寸的软盘进行检查，这张盘有两个坏块。

```
$ badblocks /dev/fd0H1440 1440
718
719
$
```

badblocks输出了自己找出的坏块编号。许多文件系统都可消除这样的坏块。它们维护了一个已知坏块列表，该列表是在制定文件系统时被初始化的，而且后期还可修改。最初的坏块搜索可以用mkfs命令来进行（这个命令对文件系统进行了初始化），但后期的检查则应该由badblocks来进行，对新增块的检查由fsck来进行。我们稍后将详细讨论\cmd{mkfs}和fsck。

现在的许多磁盘都可侦测坏块，并尽力用一个特殊的、预留的块来修复它们。这对操作系统来说，是不可视的。这个特性应该编入磁盘的使用手册内，如果你有强烈的求知欲，不妨看看手册。

3.7 分区

分区是指内存或存储设备中一个逻辑上不同的部分，其功能类似于物理上的独立的单元。一个硬盘可分为若干个分区。每个分区可用做独立的用途，也就是说，如果你只有一个硬盘，但又想装两个或更多的操作系统的话，就可以把硬盘分为两个或更多的分区。每个操作系统可在同一个硬盘上和平共处。如果没有分区，就必须为每个操作系统买一个硬盘。

软盘不能进行分区。这不是因为技术上的原因，而是它们本身容量太小，没有必要。光盘通常也不能分区，因为它们太容易获得了，而且完全可以用作一个大磁盘，很少需要在上面装几个操作系统。

3.7.1 主引导记录、引导扇区和分区表

与硬盘分区方式有关的信息保存在其第一个扇区内（也就是第一个磁带表层上的第一个磁道上的第一个扇区）。第一个扇区就是磁盘的“主引导记录”（MBR）；这是计算机初次启动时，BIOS读取并启动的那个扇区。主引导记录中包含一个小程序，这个小程序读取分区表，查看哪个分区是活动的（也就是说标记为可以启动的），并读取那个活动分区的第一个扇区，也就是该活动分区的引导扇区（MBR也是启动扇区，但它有一个特殊状态，所以它有一个专门的名称）。这个引导扇区中包含另一个小程序，该程序读取存储在这个分区上的操作系统中的第一部分并开始启动操作系统。

分区方案不是内置于硬件中的，也不是内置于 BIOS的。它只是许多操作系统都遵循的一种约定。并非所有的操作系统都遵循这一约定。有些操作系统支持分区，但它们会在硬盘上占用一个分区，并在这个分区内采用自己的内部分区方案。这种操作系统能和其他操作系统和平共处（包括Linux在内），而且不需要任何特殊的标准，但对一个不支持分区的操作系统

来说，它是不能和任何操作系统同处于一块硬盘上的。

所以，事先给大家提个醒儿，最好在一张纸上写下分区表，即使它被讹用，你也不必丢弃自己的所有文件（坏的分区表可用fdisk来修复）。利用fdisk-1命令，就能得到分区表的相关信息：

```
$ fdisk -l /dev/hda

Disk /dev/hda: 15 heads, 57 sectors, 790 cylinders
Units = cylinders of 855 * 512 bytes

   Device Boot  Begin    Start      End  Blocks   Id  System
/dev/hda1           1        1       24   10231+  82  Linux swap
/dev/hda2          25       25       48   10260   83  Linux native
/dev/hda3          49       49      408  153900   83  Linux native
/dev/hda4         409      409      790  163305    5  Extended
/dev/hda5         409      409      744  143611+  83  Linux native
/dev/hda6         745      745      790   19636+  83  Linux native
$
```

3.7.2 扩展和逻辑分区

拿PC上的硬盘来说，起初的分区方案只允许有四个分区。在实际应用中，这个分区数显然有点少，部分原因是有些人想同时采用四个以上的操作系统（ Linux、MS-DOS、OS/2、Minix、FreeBSD、NetBSD、Windows NT等），但主要原因是人们希望一个操作系统可拥有多个分区。举个例子来说，由于速度上的原因，你肯定希望交换空间有其自己的 Linux分区，而不是只有一个主要的Linux分区（随后将详细讨论）。

为了解决原有分区方案中存在的问题，扩展分区就应运而生。这就是允许把一个主分区分为若干个子分区。下分的主分区就被称为扩展分区；子分区就是逻辑分区。子分区的行为和主（非逻辑的）分区的行为类似，只是创建方式不同而已。

硬盘的分区结构如图 3-2所示。图中，磁盘被分为三个主分区，其中的第二个主分区被分为两个逻辑分区。磁盘上的某些部分根本就没有被分区。这个磁盘和每个主分区各自都有一个引导扇区。

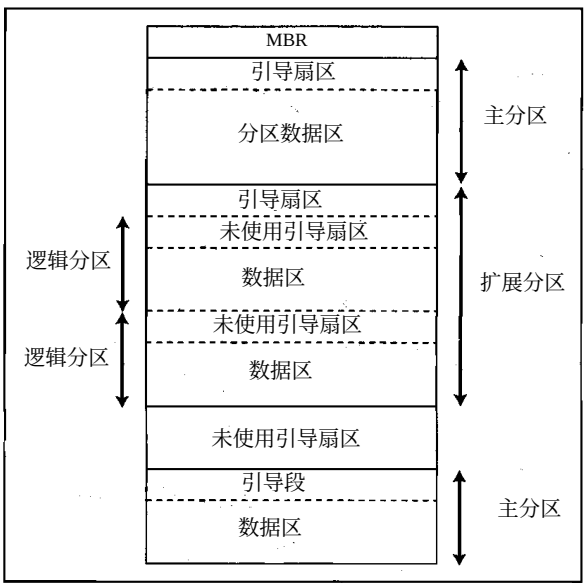

图3-2 硬盘分区示例

3.7.3 分区类型

分区表（主引导记录中的分区表和扩展分区所用的分区表）内有一个标识分区类型的位元组，每个分区一个，借此标识使用该分区的操作系统。其目的是尽可能避免两个不同的操作系统争用同一个分区。但是，实际应用中，操作系统实际上并没有注意到表示分区类型的位元组；举个例子来说，至少 MS-DOS的某些版本就会忽视这个位元组中最重要的一位，其他的则不然。

虽然没有一个固定的标准来指定位元组每一位的含义，但有些定义已经得到大家的认可，参见表3-1。它取自Linux fdisk程序。

表3-1 分区类型（取自Linux fdisk程序）

0	Empty	40	Venix 80286	94	Amoeba BBT
1	DOS 12-bit FAT	51	Novell?	a5	BSD/386
2	XENIX root	52	Microport	b7	BSDI fs
3	XENIX usr	63	GNU HURD	b8	BSDI swap
4	DOS 16-bitf <32M	64	Novell	c7	Syrinx
5	Extended	75	PC/IX	Db	CP/M
6	DOS 16-bit >=32M	80	Old MINIX	e1	DOS access
7	OS/2 HPFS	81	Linux/MINIX	e3	DOS R/O
8	AIX	82	Linux swap	f2	DOS secondary
9	AIX bootable	83	Linux native	Ff	BBT
a	OS/2 Boot Manag	93	Amoeba		

3.7.4 对硬盘进行分区

用于建立和删除分区的程序有许多。大部分操作系统都有其自己的、用于建立和删除分区的程序，建议采用各操作系统自己的程序。这类程序大多被称为 fdisk，Linux及其变体也不例外。关于Linux fdisk的用法，请参考其手册页。 cfdisk命令类似于fdisk，但前者有一个更好的（全屏）用户界面。

在使用IDE磁盘时，引导分区（具有可引导内核镜像文件的分区）必须完全包含在前 1024个柱面内。这是因为磁盘是在启动期间（在系统进入保护模式之前）通过 BIOS来使用的，而且BIOS处理的柱面不能超过1024这个限制。有时，也可能采用有一部分在前 1024个柱面内的引导分区。但前提是BIOS读取的所有文件都必须在1024个柱面内。但这是很难做到的，所以不是上上之策；因为你从不知道内核更新的时间或磁盘碎片何时会导致系统不可启动。因此，你必须保证自己的引导分区完全在前1024个柱面内。

事实上，BIOS和IDE磁盘的有些新版本能够对多于 1024的柱面进行处理。如果你有这样的系统，可不理睬前面提到的限制；如果不敢保证，最好把它存放在前 1024个柱面内。

每个分区都应该有偶数个扇区，因为Linux文件系统使用的是1K大小的块，一块就相当于两个扇区。奇数个扇区将导致最后一个扇区的浪费。这虽然不能带来任何问题，但令人不快，有些版本的fdisk会就此发出警告。

通常情况下，要改变分区的大小，需要先备份自己希望保存的数据（最好是全盘备份），然后再删除指定分区，建立新分区，最后再把数据恢复到新分区内。如果新分区比原来大，就需要调整邻近分区的大小（并进行备份和恢复）。

由于改变分区的大小是件令人头疼的事，因此最好第一次就把分区的大小设定好，要不

就应该有一个高效易用的备份系统。如果你通过不需要太多人工操作的媒体（比如光盘，与之对应的是软盘）进行安装的话，先运行不同的配置就要容易得多。由于你不需要备份数据，所以改变分区大小就要相对简单得多。

目前，有一个MS-DOS版本的程序，称为fips，该程序可以调整MS-DOS分区的大小，无须备份和恢复，但对其他的文件系统仍然需要备份和恢复。

3.7.5 设备文件和分区

每个分区和扩展分区都有其自己的设备文件。这些文件的命名约定就是在整个硬盘名称后面，再加一个分区号。命名约定中， 1至4指的是主分区（不管实际上的主分区有多少），5至8指的是逻辑分区（不管驻留在哪个主分区内）。例如，/dev/hda1指的是第一块IDE硬盘上的第一个主分区，而/dev/sdb7指的是第二块SCSI硬盘上的第三个扩展分区。

3.8 文件系统

3.8.1 何谓文件系统

在操作系统中，文件命名、存储和组织的总体结构就称为文件系统。一个文件系统内包括文件、目录以及定位和访问这些文件和目录所需的信息。它也可以表示操作系统的一部分，把应用程序对文件操作的要求翻译成低级的、面向扇区的、并能被控制磁盘的驱动程序所理解的任务。它还可以用来指代用于保存文件或文件系统类型的分区或磁盘。因此，如果有人对你说“ 我有两个文件系统 ”时，千万不要惊羡，他可能是指自己有两个分区，一个分区用于保存文件，另一个在用“ 扩展文件系统 ”(它表示文件系统类型)。

磁盘或分区与文件系统间的区别就是：文件系统中包含一些重要的数据。少数程序（其中包括创建文件系统的程序）直接在磁盘或分区的原始扇区上操作；如果这些扇区上有现成的文件系统，这个文件系统就会被遭到破坏或严重损伤。大多数程序都是在文件系统上操作的，因此，不能在不包含文件系统的分区上运行（或者说不能在其中包含的文件系统类型有误的分区上运行)。

分区或磁盘可用作文件系统之前，需要进行初始化，而且管理操作数据结构也需要写入磁盘。这个过程就叫作“ 制定文件系统 ”。

许多Unix文件系统类型都有一个类似的通用结构，虽然其细节千差万别。主要的几个概念是：超级块、 inode、数据块、目录块和目录块。“ 超级块 ”中包含文件系统的整体信息，比如文件系统的大小（具体信息和文件系统决定)。“ inode ”中包含一个文件的所有信息（文件名除外)。文件名随inode号一起，保存在目录内。目录条目由文件名和代表该文件的 inode编号组成。inode内包含几个数据块的编号，这些数据块是用来保存文件中的数据的。但是，inode内只能容纳少数几个数据块编号，如果需要的数据块越多，就要为指向数据块的指针动态分配更多的空间。这种动态分配的块就叫作“ 间接块 ”；其名表示，为找出数据块，必须先在间接块内找出其数据块编号。

Unix文件系统通常允许人们在文件内建立一个漏洞 (hole)（即不连续存储，这是利用 lseek来完成的，详情参见手册页)，这意味着文件系统只是伪称文件中的某个特定位置全部是零字节。但是，针对文件中的那个位置，却没有为其保留实际的磁盘扇区（换言之，文件实际占

用的磁盘空间要比显示的文件长度少一些)。对一些小程序、Linux共享库、某些数据库以及另外一些特例来说，这种情况非常普遍。“漏洞”是怎样实现的呢？具体的做法是在间接块或inode中保存一个特殊的值，将其作为数据块的地址。这个特殊的地址意味着不为文件的那一部分分配实际的数据块。这样一来，文件中便好像出现了一个“漏洞”。

漏洞的好处不多。在我本人的系统中，一个简单的例子显示了通过磁盘上所用的 200MB漏洞，大约为我节省了近4MB的空间。但这个系统中仍然有较少的程序，而且没有数据库文件。

3.8.2 文件系统综述

Linux支持几类文件系统。下面将为大家介绍几个重要的：

1. minix

最早的、大概也是最可靠的文件系统，但特性相当少（有些时间戳没有，文件名限定在30个字符内），而且容量小（每个文件系统最多只能有64MB大）。

2. xia

minix文件系统的修订版，对文件名和文件系统长度的限制放宽，但没有引入新的特性。它应用不广，但其表现相当不错。

3. ext2

原来的Linux文件系统中最富有特性的一个，也是当前最流行的文件系统之一。它的向上兼容性很棒，所以有些文件系统代码的新版本不需要重新制定现有的文件系统。

4. ext

ext2的早期版本，它不能向上兼容。几乎不能用于新版本安装，许多人都转为采用 ext2。

除此以外，由于它提供了对几个外来文件系统的支持，所以和其他操作系统交换文件就方便得多。这些外来系统的运行就向原生文件系统一样，只是可能缺少某些常用的 Unix特性，或有些奇怪的限制或其他稀奇古怪的事。

5. msdos

兼容于MS-DOS（和OS/2以及Windows NT）FAT文件系统。

6. umsdos

对Linux下面的msdos文件系统驱动程序进行扩展，从而获得长文件名、拥有者、访问许可、链接和设备文件。这样一来，就允许把一个普通的 msdos文件系统用作Linux文件系统，从而无须再为Linux分配一个独立的分区。

7. iso9660

标准的光盘文件系统；常见的光盘标准文件系统扩展是 Rock Ridge，它允许自动支持长文件名。

8. nfs

网络文件系统，允许多台计算机共享一个文件系统，以便能通过其中一台连网计算机轻松地访问文件。

9. hpfs

OS/2文件系统

10. sysv

SystemV/386、Coherent和Xenix文件系统。

选择什么样的文件系统，要根据具体情况而定。如果由于兼容性或别的原因，需要某个非原生（即外来的）的文件系统，就必须使用它。如果想自由选择文件系统，最明智的作法是利用ext2，因为它具有丰富的特性，而且不会对性能产生影响。

另外，还有proc文件系统，通常作为 /proc目录供人们访问，该文件系统实际上根本不是文件系统。 proc文件系统能够使访问特定的内核数据结构（比如进程列表， proc就得名于process）更为方便。它使这些数据结构看起来像一个文件系统，而这个文件系统可以利用所有的常用文件工具来进行处理。举个例子来说，要想获得列出所有进程的列表，就可能需要命令

```
$ ls -l /proc
total 0
dr-xr-xr-x   4 root     root            0 Jan 31 20:37 1
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 63
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 94
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 95
dr-xr-xr-x   4 root     users           0 Jan 31 20:37 98
dr-xr-xr-x   4 liw      users           0 Jan 31 20:37 99
-r--r--r--   1 root     root            0 Jan 31 20:37 devices
-r--r--r--   1 root     root            0 Jan 31 20:37 dma
-r--r--r--   1 root     root            0 Jan 31 20:37 filesystems
-r--r--r--   1 root     root            0 Jan 31 20:37 interrupts
-r--------   1 root     root      8654848 Jan 31 20:37 kcore
-r--r--r--   1 root     root            0 Jan 31 11:50 kmsg
-r--r--r--   1 root     root            0 Jan 31 20:37 ksyms
-r--r--r--   1 root     root            0 Jan 31 11:51 loadavg
-r--r--r--   1 root     root            0 Jan 31 20:37 meminfo
-r--r--r--   1 root     root            0 Jan 31 20:37 modules
dr-xr-xr-x   2 root     root            0 Jan 31 20:37 net
dr-xr-xr-x   4 root     root            0 Jan 31 20:37 self
-r--r--r--   1 root     root            0 Jan 31 20:37 stat
-r--r--r--   1 root     root            0 Jan 31 20:37 uptime
-r--r--r--   1 root     root            0 Jan 31 20:37 version
$
```

可能有少数几个特别的文件不能对 process（进程）作出响应。上面的示例已被缩短。

注意，虽然被称为文件系统，但 proc文件系统压根儿就不存在于磁盘上。它只存在于内核的映像中。只要有人想查看 proc文件系统的某个部分，内核就会令这部分看起来就像保存在磁盘上的某个地方一样，事实上根本没那回事。所以，即使有一个长达几 GB的/proc/kcore文件，它丝毫不占用你的任何磁盘空间。

3.8.3 如何选用文件系统

在不同文件系统间选择自己需要的文件系统时，通常不用花太多心思。目前， ext2fs是最常用的文件系统，而且它也可能是你最明智的选择。选择依据是管理操作结构的开销、速度、可靠性、兼容性和其他各种因素。文件系统的选择需要具体情况具体决定。

3.8.4 如何建立文件系统

文件系统的建立（也就是初始化）是利用 mkfs命令来完成的。实际上，每个文件类型都有一个独立的程序来建立系统。 mkfs正好是一个开端，它根据期待的文件系统类型，运行相

应的程序。文件系统类型的选择是利用 -t fstype选项来完成的。

1. -t fstype

选择文件系统类型。

2. -c

搜索坏块并相应初始化坏块列表。

3. -l filename

从同一个文件内读取最初的坏块列表。

为了在一张软盘上建立一个ext2文件系统，应该给出下面这些命令：

```
$ fdformat -n /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
$ badblocks /dev/fd0H1440 1440 $>$ bad-blocks
$ mkfs -t ext2 -l bad-blocks /dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

首先，格式化软盘（-n选项用来避免校验，也就是避开坏块检查）。然后，badblocks命令对坏块进行搜索，其输出重新定向到一个 badblocks文件内。最后，再利用由badblocks初始化的坏块列表建立文件系统。

-c选项也可以随mkfs（而不是badblocks）和一个独立的文件一起使用。例如下面的示例。

```
$ mkfs -t ext2 -c /dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group

Checking for bad blocks (read-only test): done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

和单独使用badblocks相比，-c选项更为方便，但对于文件系统已经建立好之后的坏块检查来说，badblocks又是必不可少的。

在硬盘或分区上制订文件系统的过程和软盘是一样的，只不过不需要格式化。

3.8.5 装入和卸装

在使用文件系统之前，必须装入它。然后，操作系统才能执行管理操作，以保证一切正常运行。由于Unix内的所有文件都在一个单独的目录树内，因此，装入操作将令新文件系统中的内容类似于一个现有子目录的内容，这个子目录位于某个已装入了的文件系统中。

举个例子来说，图 3-3展示了三个独立的文件系统，每个文件系统都有自己的 root目录。后两个文件系统分别装在 /home和/usr下面，在第一个文件系统上，我们可得到一个单独的目录树，如图3-4所示。

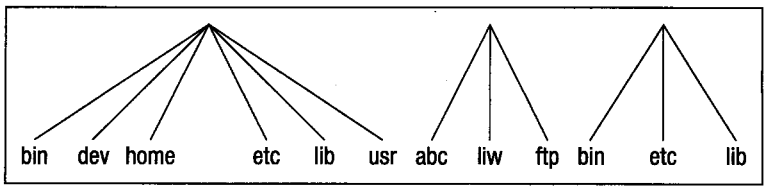

图3-3 三个独立的文件系统

可像下面的示例一样，装入文件系统：

```
$ mount /dev/hda2 /home
$ mount /dev/hda3 /usr
$
```

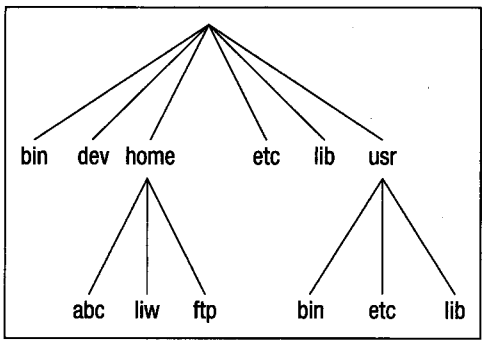

图3-4 已经装入的/home和/usr

mount命令采用了两个参数。第一个是设备文件，该文件对应于其中包含准备装入的文件系统的磁盘或分区。第二个参数是目录，文件系统将装在这个目录下。执行完装入命令之后，这两个文件系统中的内容正好分别和 /home和/usr目录下的内容一样。然后，我们就可以说“ /dev/hda2已装入/home ”，对/usr来说，同样如此。要想查看其中一个文件系统，就应该查看目录内容，文件系统已装入这个目录下。注意，设备文件/dev/hda2和装入目录/home之间的区别。设备文件提供的是对磁盘原始内容的访问，而装入目录提供的则是对磁盘上文件的访问。装入目录被称为“ 装入点 ”。

Linux支持许多文件系统类型。 mount命令可尽力猜测文件系统的类型。另外，你也可用 -t fstype选项，直接指定文件系统的类型；有时必须这样，因为启发式的 mount命令不是“ 万能的 ”。举个例子来说，要装入一个MS-DOS软盘，可采用下面的命令：

```
$ mount -t msdos /dev/fdo /floppy
$
```

装入目录不要求一定是空的，尽管它必须存在。但对其中的任何文件来说，在文件系统装入期间，将不能按其文件名访问它们（已经打开的任何文件仍然可以访问。从其他目录硬链接过来的文件也可通过其文件名得以访问）。这一要求没什么坏处，甚至还可说有好处呢！举个例子来说，有些人喜欢有 /tmp和/var/tmp这两个同义词，并令 /tmp/为var/tmp的符号链接。系统启动时，在 /var文件系统装入之前，采用的是驻留于 root文件系统上的/var/tmp目录。/var在装入时，将令root文件系统上的/var/tmp目录为不可访问。如果 /var/tmp没有存在于root文件系统上，就不能在装入/var之前，使用temporary（临时）文件。

如果你不打算在文件系统内写入任何东西，可采用用于 mount的-r开关，进行只读装入。这样一来，就会令内核防止在文件系统内进行写操作尝试，并防止内核更新 inode内的文件访问次数。只读装入对不能写的媒体（比如光盘）来说，是必要的。

细心的读者可能已经注意到一个逻辑上的问题了。很显然，第一个文件系统（称为 root文件系统，因为其中包含root目录）是不能装在另一个文件系统上的，那么它是怎样装入的呢？答案曰“ 它是自动装入的 ”（更多详情，请参考内核源码或《内核黑客指南》）。root文件系统是在启动时自动装入的，所以我们可以假定它是始终存在的。如果 root文件系统都不能装入的话，系统压根儿就不能启动。以 root身份自动装入的文件系统名要么已经编入内核，要么用LILO或rdev设置。

root文件系统通常先以只读方式装入。然后，启动脚本将运行 fsck，验证其有效性，如果没有问题，它们就会重装这个文件系统，以便允许对它进行写操作。 fsck禁止运行于一个已装入的文件系统上，因为fsck运行期间，任何对该文件系统的改动都将带来麻烦。由于 root文件系统是只读装入的，在它接受验证期间， fsck可解决任何难题，因为重装入操作将刷新文件系统保存在内存中的任何元数据。

许多操作系统上，还有其他的文件系统也应该在启动时自动装入。这些文件系统是/etc/fstab文件内指定的；关于格式方面的详情，请参考 fstab手册页。特定文件系统的装入细节和许多因素有关，必要时，每个管理员都可对其进行配置；详情参见第 5章。

当一个文件系统不再需要被装入时，就可以用 umount来卸装（当然，这个命令应该是unmount，但自从“ n ”于70年代自动消失之后，就再已没有采用了）。umount采用了一个参数：不是设备文件，就是装入点。举个例子来说，要卸装前一个示例中的目录，可像下面这样：

```
$ umount /dev/hda2
$ umount /usr
$
```

关于这个命令的用法，参考其手册页。无论如何，总是需要卸装已装入的软盘。不要以为把软盘从驱动器中取出就万事大吉！由于磁盘缓冲的原因，在你卸装软盘之前，没必要把数据写入软盘，所以过早把软盘从驱动器中取出可能导致系统内容混乱不堪。如果你只能通过软盘读取系统，这样做或许没什么大不了，但如果你是在系统上进行写操作的话，后果就难以设想。

装入和卸装要求有超级用户权限，也就是说只有 root才能执行。其原因是如果任何用户都可以在任何目录上装入软盘的话，利用乔装为 /bin/sh或别的常用程序的特洛伊木马来建立一个软盘系统就太容易了。但是，通常情况下，允许用户使用软盘是很有必要的，其方式如下：

■ 把root密码给用户。显然，这将带来安全隐患，但这是最简单的解决办法。如果没有安全方面的需要，它比较理想，适用于多数没有连网的个人系统。

■ 使用sudo之类的程序，允许用户执行装入。这仍然存在安全隐患，但它没有直接赋予每个用户超级用户的权限。

■ 令用户使用mtools，它是一个用于操控MS-DOS文件系统的包，是无需装入的。如果需要的只是MS-DOS软盘的话，它就比较理想，其他情况下的表现则相当笨拙。

■ 利用/etc/fstab内的适当选项，列出软盘设备及其允许装入点。

要想实施最后一种方法，具体做法是在 \fn{/etc/fstab}文件内增添下面这一行：

```
/ dev/fd0 / floppy msdos user,noauto 0 0
```

这些列分别是：准备装入的设备文件、装入点、文件系统类型、选项、备份频率（供dump使用）和fsck传递号（指定启动之后应该按什么顺序检查； 0表示不检查）。

noauto选项在系统启动时终止了自动装入（也就是说，它禁止了 mount -a执行装入）。user选项允许任何一个用户装入文件系统，而且由于安全方面的原因，禁止通过已装入的文件系统执行程序（普通的或 setuid）和解释设备文件。之后，任何一个用户都可利用下面的命令，装入带有msdos文件系统的软盘：

```
$ mount /floppy
$
```

软盘可以（当然，也需要）利用相应的 \cmd{umount}命令卸装。

如果你打算提供对若干种类型的软盘的访问，就需要指定若干个装入点。每个装入点的设置可以不同。例如，要想同时提供对 MS-DOS和ext2软盘的访问， /etc/fstab文件内就会有下面两行：

```
/dev/fd0 /dosfloppy msdos user,noauto   0   0
/dev/fd0 /ext2floppy ext2 user,noauto    0   0
```

对MS-DOS文件系统来说，如果你想限制对它的访问，可利用 uid、gid和umask文件系统选项来实现，详情参见 mount手册页。如果你不细心的话，装入一个 MS-DOS文件系统之后，每个用户都会默认得到该文件系统中的文件读取访问权，这不见得是个好办法。

3.8.6 利用fsck检查文件系统的完整性

文件系统是个非常复杂的东西，而且容易出错。一个文件系统的正确性和有效性是可以检查出来的，具体做法是利用 fsck命令。对于它找到的任何一个小错误，它都可以对其进行修复，并提醒用户文件系统是否存在不可修复的错误。所幸的是，对实施文件系统所用代码进行的调试是相当有效的，所有文件系统本身几乎不存在错误，如果有的话，通常都是由于电源故障、硬件故障或操作错误引起的；比方说没有正确关闭系统。

许多系统都被设置为启动时自动运行 fsck，所以在使用系统之前，需要侦测错误（希望一切正常）。使用损坏了的文件系统将引发恶果：如果数据结构一团糟，使用这个文件系统将使其结构更为恶化，导致更多数据的丢失。但是， fsck在一个大型的文件系统上运行时，花的时间要多一些。因为几乎没有错误产生（如果已正确关闭系统的话），可采用两个“ 技巧 ”来避免错误检查。其一是：如果 etc/fastboot存在的话，就不执行检查。其一： ext2文件系统在其超级块内有一个特殊的标记，用以说明这个文件系统在前一次装入之后，是否正确卸装。这样一来，如果这个标记指出卸装已完成的话（正确卸装表明没有问题），就允许 e2fsck（用于ext2文件系统的fsck版本）避免对文件系统的检查。 /etc/fastboot技巧对你的文件系统来说，是否有用和你的启动脚本有关，但每次你使用 e2fsck时，ext2技巧都会运行。因此，必须利用一个你准备避免的e2fsck选项，显式绕过ext2技巧（关于具体做法，参见 e2fsck手册页）。

自动检查只适合启动时自动装入的文件系统。利用fsck手动检查其他文件系统，比如软盘。

如果fsck发现了不能修复的错误，你就需要深入了解文件系统的工作原理，以及特殊情况下文件系统损伤的类型或如何备份。后者（有时有点费时）更易于安排，如果你自己没有什么实际知识的话，可从朋友、 Linux新闻组和邮件列表或其他资源处获得。我也乐意尽力帮助你。另外， Theodore Ts ’ o编写的debugfs程序也将对你有所帮助。

fsck只可运行于未装入的文件系统上，绝不能运行于已装入的文件系统上（启动期间只读装入的root除外）。这是因为它访问的是原磁盘，因而可以在操作系统不注意的情况下，修改文件系统。如果操作系统注意到了它，情况就难办了。

3.8.7 利用badblocks检查磁盘错误

周期性地检查磁盘中的坏块是个好习惯。这是通过 badblocks命令来执行的。该命令输出一份列表，上面列出它自己找到的所有坏块。该列表可以传给 fsck，记录到文件系统结构内，以便操作系统不在尝试用坏块保存数据。下面的示例向大家解释了整个过程：

```
$ badblocks /dev/fd0H1440 1440 > bad-blocks
$ fsck -t ext2 -l bad-blocks /dev/fd0H1440
Parallelizing fsck version 0.5a (5-Apr-94)
e2fsck 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Check reference counts.
Pass 5: Checking group summary information.
/dev/fd0H1440: ***** FILESYSTEM WAS MODIFIED *****
/dev/fd0H1440: 11/360 files, 63/1440 blocks
$
```

如果badblocks报告一个块已被采用， e2fsck就会试着把这个块移到另一个地方。如果这个块真的坏了，不仅仅是无关紧要的损伤，就说明文件的内容也可能被损坏了。

3.8.8 抵制碎片

碎片是指某一磁盘文件被分解成几个部分，散落存储在磁盘上的不同区域。在磁盘上删除或新建文件均会产生碎片。碎片现象会降低从磁盘存取数据的速度并降低磁盘操作的整体性能（虽然不很严重）。

ext2文件系统会试着把碎片控制在最低限度，具体作法是把一个文件内的所有存储块尽可能地放在一起，即使它们不能连续保存在彼此相邻的扇区内。 ext2总是有效地分配最靠近文件内其他块的没有用过的块。以 ext2来说，不必再担心碎片的问题。用于整理 ext2文件系统碎片的程序，请参考 ftp://tsx-1 1.mit.edu/pub/linux/packages/ext2fs/defrag-0.70.lsm和ftp://tsx-1 1.mit.edu/pub/linux/packages/ext2fs/defrag-0.70.tar.gz。

MS-DOS磁盘碎片整理工具也有许多。它们把块移到文件系统内，进而清理碎片。对其他文件系统来说，碎片整理必须通过备份、重建并从备份中恢复文件系统的形式来进行。对所有文件系统来说，在整理碎片之前备份文件系统都是上上之策，因为在进行碎片整理期间，可能会出错。

3.8.9 适用于所有文件系统的其他工具

其他一些工具也适用于管理文件系统。 df展示一个或多个文件系统上的剩余磁盘空间；du展示一个目录及其所有文件占用了多少磁盘空间。它们可用于找出谁是浪费磁盘空间的“ 真凶 ”。

sync把缓冲区内所有没有写入的块 (参见4.6节）写入磁盘。这个过程很少需要手动进行；

后台进程会自动进行此类的更新。它特别适合于灾难性场合，比如更新或其助理进程 dbflush 已死时，或你必须即刻关掉电源，不能等待运行更新时。

3.8.10 适用于ext2文件系统的其他工具

除了可直接或通过不依赖于前端的文件系统类型访问文件系统创建程序（ mke2fs）和检查程序（e2fsck）外，ext2文件系统还有一些有用的其他工具。

tune2fs用于调整文件系统参数。下面列出其中一些较为有趣的参数：

■ 最大装入数。e2fsck在文件系统已被装入多次时进行检查，即使在清除标记已设定的情况下。对用于开发或自行测试的系统来说，这个数目越小越好。

■ 两次检查之间的最大装入次数。 e2fsck也可在两次检查期间实施一个最大装入次数，即使在清除标记已设定而且文件系统未曾经常性地装入的情况下也是如此。但这个参数是可以禁用的。

■ 为root预留的块之数目。 ext2为root保留了一些块，以便在文件系统填满的情况下，仍有可能在无须删除任何东西的情况下，进行系统管理。默认情况下，保留的块数大约是总数的5%，对大多数磁盘来说，这样的设置不会太浪费。但是，对软盘来说，没有预留任何块。

更多详情，参见tune2fs手册页。

dump2fs展示了一个ext2文件系统的有关信息，这些信息大多来自超级块。下面列出的代码展示了一个示范输出。该输出中的有些信息是关于技术上的，要求掌握文件系统的工作原理，但对才入道的管理员来说，其中的大部分信息都是很容易理解的。

```
dumpe2fs 0.5b, 11-Mar-95 for EXT2 FS 0.5a, 94/10/23
Filesystem magic number:  0xEF53
Filesystem state:         clean
Errors behavior:          Continue
Inode count:              360
Block count:              1440
Reserved block count:     72
Free blocks:              1133
Free inodes:              326
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         360
Last mount time:          Tue Aug  8 01:52:52 1995
Last write time:          Tue Aug  8 01:53:28 1995
Mount count:              3
Maximum mount count:      20
Last checked:             Tue Aug  8 01:06:31 1995
Check interval:           0
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)

Group 0:
  Block bitmap at 3, Inode bitmap at 4, Inode table at 5
  1133 free blocks, 326 free inodes, 2 directories
  Free blocks: 307-1439
  Free inodes: 35-360
```

debugfs是一个文件系统调试程序。它允许用户直接访问保存在磁盘上的文件系统数据结

构，因此，对fsck不能自动修复的磁盘错误，就可以用它来修复。它还可用于恢复被删除了的文件。但是使用debugfs时，要求你一定要清楚自己正在干什么；如果冒然行事的话，将导致你的全部数据受损。

dump和restore可用于备份ext2文件系统。它们是过去Unix备份工具的ext2专用版本。关于备份的更多详情，请参考第9章。

3.9 无文件系统的磁盘

并非所有的磁盘或分区都可用作文件系统。以交换分区为例，它上面就没有文件系统。许多软盘都是以磁带驱动模拟方式来使用的，所以 tar或其他文件是直接写入原磁盘上的，也没有文件系统。Linux启动盘中没有文件系统，只有原内核。

不采用文件系统的好处在于可获得更多的磁盘空间，因为文件系统总存在一些管理操作数据上的开销。另外，不用文件系统还可以令磁盘更好地兼容其他的系统；举个例子来说，所有系统上的tar文件格式都是一样的，而在许多系统上，文件系统却是有区别的。如果你需要无文件系统的磁盘，很快就能掌握其用法。虽然 Linux启动盘内可以有文件系统，但是没多大必要。

使用原磁盘的理由是为其制作镜像备份。例如，如果磁盘内包含一个部分受损的文件系统，在尝试修复它之前，最好为其做个备份，因为在你的修复把事情弄得更糟之前，还可以重新再来。具体做法是使用dd：

```
$ dd if=/dev/fd0H1440 of=floppy-image
2880+0 records in
2880+0 records out
$ dd if=floppy-image of=/dev/fd0H1440
2880+0 records in
2880+0 records out
$
```

第一个dd为文件floppy-image制作一个完全镜像，第二个 dd把这个镜像写到软盘上（前提是用户在执行第二个命令之前，要切换软盘。不然的话，这对命令就会没有用）。

3.10 磁盘空间的分配

3.10.1 分区方案

要想一下子说出最佳分区方案是什么，恐怕不是件容易的事儿。更糟的是，也没有一个通用的标准；分区涉及到的因素太多了。

传统做法是：拥有一个相对较小的 root文件系统，该文件系统内包含 /bin、/etc、/dev、/lib、/tmp以及其他启动并运行系统所需要的东西。这种作法中， root文件系统（在其自己的分区或自己的磁盘上）就是启动系统所需要的一切。这种做法的依据是：如果 root文件系统较小，而且用得不是很多，那么在系统崩溃时，它受损的可能性就越少，你就能迅速而轻易地查出系统崩溃的原因。然后，再为 /usr（是用户的根目录，通常在 /home下面）下面的目录树和交换空间，建立独立的分区，或者采用单独的磁盘。把根目录（其中有用户文件）单独保存在其分区内有利于备份，因为通常情况下，没有必要备份程序（驻留在 /usr下）。在连网环境中，多台计算机共享/usr也是可能的（比如通过NFS），藉此可节约磁盘空间。

采用多个分区的问题在于：它把未使用的磁盘空间分成许多更小的部分。如今，磁盘和操作系统比以前更为可靠，人们宁愿只用一个分区来容纳他们的所有文件。另一方面，不采用多个分区的话，还可减少备份和恢复小型分区的麻烦。

对小硬盘来说（假设你不进行内核开发），最好的办法是只用一个分区。对大硬盘来说，最好有少数几个大分区（注意，“小”和“大”是相对而言的；根据你自己的工作需要而定）。

如果你有若干个磁盘，就可以把 root文件系统（包括/usr在内）装入一个磁盘，用户根目录装入另一个磁盘。

最好亲自体验各种分区方案。虽然会花相当多的时间（因为要多次从头安装系统），但这样可增加你的经验值。

3.10.2 空间要求

你安装的Linux系统将指出各种配置所需的磁盘空间是多少。单独安装的程序也是如此。这样有助于你合理安排自己的磁盘空间，但与此同时，你也应该为将来做好准备，保留一些空间。

用于用户文件的磁盘空间和你的用户的想法有关。许多人希望为自己的文件分配尽可能多的空间，但他们各自的需求是不一样的。有些人只执行简单的文字处理，可能只需要几 MB，有的人确要执行大型的图像处理，他们则需要几 GB。

顺便说说，在比较以KB或GB表示的文件和以GB表示的磁盘空间时，重要的是要知道这两个单位是有区别的。有的磁盘厂商喜欢称 1KB相当于1000个字节，而1MB相当于1000千字节，而计算机行业里，采用的换算比例则是 1024。因此， 345MB的硬盘实际上只是一个330MB硬盘。

交换空间的分配，参见4.5节。

3.10.3 硬盘分区示例

我过去有一个109MB的硬盘。现在用的是330MB的硬盘。下面，将介绍我是怎么对这两个硬盘分区的，以及分区根据又是什么。

在我的需要和使用的操作系统改变时，我采用了许多方式对 109MB硬盘进行分区。下面将介绍两个典型方案。起先，我同时运行的操作系统有 MS-DOS和Linux。对此，我需要20MB的磁盘空间，能装MS-DOS、一个C编译器、一个编辑器、少数其他的实用程序、应用程序，另外还有部分不至于让我缩手缩脚的剩余空间。对 Linux系统，我为其分配了 10MB的交换空间，其余的 79MB用于一个独立的分区，装有 Linux系统下的所有文件。我曾经尝试过把root、/usr和/home各自保存在独立的分区内，但剩余空间始终不够。

当我不再需要MS-DOS时，就把磁盘重新分了区，所以又有了一个 12MB的交换分区，而且再次让剩下的8MB空间供独立的文件系统使用。

330MB磁盘被分为若干个分区，如下所示：

5MB	root文件系统
10MB	交换分区
180MB	\fn{/usr}文件系统
120MB	\fn{/home}文件系统
15MB	起始分区

起始分区用于各种操作需要自己的分区时：比如你尝试不同的 Linux版本或对不同文件系统的速度进行比较时。不需要时，起始分区就被用做交换空间（我喜欢开许多个窗口）。

3.10.4 为Linux增添更多磁盘空间

为Linux增添更多磁盘空间很简单，前提是只要硬件已经正确安装（硬件的安装不在本书讨论之列），必要时，格式化磁盘，再向前面所讲的那样，建立分区和文件系统，并在/etc/fstab内增加恰当的数据行，使其能够自动装入。

3.10.5 关于节省磁盘空间的几个提示

关于节省磁盘空间的提示，上上之策是尽量不要安装不必要的程序。许多 Linux版本都有一个选项，允许你根据自己的需要，选择性地只安装程序包里的一部分程序。这样有助于为你节省大量的磁盘空间，因为许多程序都是相当大的。虽然你的确需要某个特定的包或程序，但可能没必要全部都要。举个例子来说，有些在线文档可能不是你必需的，用于 GNU Emacs的某些Elisp文件、X11的某些字体和用于编程的某些库也如此。

如果你实在不能卸载程序包，就可考虑对其进行压缩处理。诸如 gzip或zip此类的压缩程序将压缩（和解压）为单独的文件或文件集。 gzexe系统将采用用户不可视的方式，压缩和解压程序（压缩未用过的程序，在需要使用它们时，再进行解压）。现在处于实验阶段的DouBle系统将把所有的文件压缩到一个文件系统中，对使用它们的程序来说，都是不可视的（如果你熟悉类似于Stacker for MS-DOS之类的产品，其原理和它是一样的）。