第一部分　Linux网络管理员指南

Olaf Kirch　著

第1章 网络基础

自世上出现“电信”或“远程通信”技术那一天起，“连网”（Networking）的概念恐怕便已产生了。想想生活在石器时代的人，若相距较远，那么可通过鼓声，相互间传递消息。现在，假定穴居人A想邀请穴居人B参加投掷石块的游戏。但是，他们离得实在太远了，以至于B听不到A的鼓声。那么，穴居人A该采取何种对策呢？他面临着三种选择：1)步行到B处；2)做一只更大的鼓；3)请求住在A和B之间的穴居人C，帮助转发消息。最后一种选择便叫做“连网”！

当然，人类进化到今天，我们再也不会使用老祖宗那些简陋的工具。今天，我们有了计算机，它们通过电缆、光纤、微波以及其他媒介，相互间可以“交谈”。相距遥远的人，可使用计算机，约定在星期六玩一场足球比赛。后面，我们将探讨达成这一目标的途径及方式。但是，不打算涉及通信线缆的问题。当然，足球比赛也请暂时抛在一边吧！

在此，有两种类型的网络是我们感兴趣的：以UUCP为基础的网络，以及以TCP/IP为基础的网络。UUCP和TCP/IP均属于“协议套件”或者“软件包”的类别，它们提供了在两台计算机之间传输数据的途径。在本章，我们打算来看看这两种类型的网络，并讨论它们的基本原理。

我们将“网络”（Network）定义成一系列“主机”（Host）的集合，不同主机相互间能够通信。通常，这种通信要依赖某些“专用主机”或“服务器”（Server），在参与通信的两部主机之间，对数据进行转发或“中继”（Relay）。当然，最常见的主机便是计算机，但也并不全是。例如，X终端或智能打印机也属于主机的一种类型。小范围内的主机集合亦称作“站点”（Site）。

无论人还是主机，相互间想实现通信，不采用某种形式的语言或代码，那是根本不可能的。在计算机网络中，这些语言统称为“协议”（Protocol）。但是，不要把它们想象成书写的那种“协议”。相反，它们实际是一些高度规范化的代码，严格约束着通信双方的行为。用最简单的话来说，计算机网络中采用的协议就是一系列非常严格的规则，规定着两个或更多主机相互间如何交换消息。

1.1 UUCP网络

UUCP是“Unix到Unix拷贝”（Unix-to-Unix Copy）的简称。开始时，它以一个软件包的形式出现，通过串行线路传输文件，对文件传输进行安排和调度，并发起程序在远程站点的执行。自70年代末首次问世以来，尽管它在许多方面发生了重大变化，但就其提供的服务来说，却变化不大。它仍然主要应用于广域网（WAN）环境。在这种环境中，主机的连接要通过拨号电话线路来进行。

UUCP是由美国贝尔实验室于1977年开发出来的，用于在他们的Unix开发站点之间实现通信。到1978年年中，这个网络总共已连接了超过80个站点。除了运行非常原始的电子邮件（E-mail）服务之外，还支持远程打印功能。然而，该系统的用途还是发布新软件和错误修正

文件（补丁文件）。今天，UUCP已经不再局限在那样的环境内。在这个网络内，目前运行着各式各样的平台，比如AmigaOS，DOS，Atari的TOS等等。它们有的提供免费服务，有的则提供商业服务。

UUCP网络最大一个缺点便是带宽较低。一方面，电话设备紧紧限死了最大的传输速度；另一方面，UUCP中极少存在永久性的连接。相反，各主机以固定的周期，通过拨号方式建立与对方的连接。所以大多数时候，邮件消息往往不能一下子送至目的地；而是先躺在某个主机的硬盘里，等待下一次连接建立的时候再发送出去。

尽管存在这些限制，目前世界上仍运行着为数众多的UUCP网络。它们主要由一些业余人员负责维护，有时收取低廉的价格，让私人用户接入网络。UUCP之所以仍然受到某些人的青睐，主要原因便是和通过专线永久性接入Internet相比，它需要的费用便宜得多。要想使您的计算机成为UUCP的一个节点，需要的全部家当只有一部Modem、一个适合UUCP运行的软硬件环境以及另外一个UUCP节点（愿意为你提供邮件和新闻转发服务，亦即你的“上游节点”）。

如何使用UUCP

UUCP的工作方式非常简单。从它的名字（Unix到Unix拷贝）便可知道，它主要负责将文件从一个主机拷贝（复制）到另一个主机。但除此以外，它还允许在远程主机上采取一些特定的行动。

假定你的机器有权访问一个名为Swim的主机，现在想让它为自己执行lpr这条打印命令。如何做到呢？可在自己的命令行键入下述命令，让这本书在Swim机器上打印出来：

```
$ uux -r swim!lpr !netguide.dvi
```

这样一来，便可指示uux（来自UUCP套件的一个命令）为Swim安排一项作业。在这个作业中，包括一个输入文件，名为netguide.dvi；另外，还包括将该文件送给lpr的请求。其中，-r参数告诉uux不要马上呼叫远程系统，而是将这个作业暂存下来，等稍后建立了一次连接再说。这个过程称作“缓冲”（Spooling）。

UUCP的另一个特点便是允许经由几个主机，对作业及文件进行转发，只要各主机相互间能够协作。现在，假定上例的主机Swim建立了与Groucho的一条UUCP链路，后者维护着一个大型的应用程序档案库。那么，为了将文件tripwire-1.0.tar.gz下载到自己的站点，需执行下述命令：

```
$ uucp -mr swim!groucho!~/security/tripwire-1.0.tar.gztrip.tgz
```

创建的这个作业会请求Swim帮自己从Groucho处取得文件，并将其发至自己的站点。在自己的站点，UUCP会将文件保存为trip.tgz，并发一封电子邮件，通知文件已经到达。整个过程分三步走：第一步，从自己的站点将作业发给Swim。第二步，在Swim处，下一步请求建立同Groucho主机的连接，并从它那里下载回指定的文件。最后一步是将文件从Swim实际传回自己的主机（站点）。

对UUCP网络来说，目前它提供的最重要的一种服务便是电子邮件和新闻。以后，我们还会讲述这方面的问题。在这里，仅对它们进行一番简要的介绍。

电子邮件，简称E-mail、email，使我们能直接与远程主机交换消息或信件，而不用知道如何访问这些主机。将一条消息（信件）从自己的站点引导至目标站点的任务完全是由邮件控制系统执行的。在UUCP环境中，邮件传输通常需要在一个相邻的主机上执行rmail命令，

将邮件正文和接收方的地址传递给它。随后， rmail会将邮件转发至另一个主机……以此类推，直到最后抵达目标主机。在本书第12章，还会对此详述。

至于“新闻”（News），可想像成一种分布式的电子公告板系统。通常说到“新闻”的时候，是指通过Usenet新闻组发布的新闻，它目前是应用得最广的新闻交换网络，总共有大约12万个成员站点。Usenet的起源可追溯到1979年。当时，与新出的Unix-V7一道发布了UUCP之后，三位应届毕业生产生了一个想法，打算在Unix社区中实现常规的信息交换。他们设计了一些脚本，形成了世界上第一个网络新闻系统的雏形。1980年，这个网络在北卡州的两所大学里，成功地联通了Duke，UNC和PHS等大学，同时连通了北加州的两家大学。以此为基础， Usenet逐渐发展壮大。尽管它起源于UUCP网络，但现在已跨越了多种形式的网络，并不单单UUCP一种。

在Usenet中，最基本的信息单元便是“文章”或者“帖子”。所有文章都要投递到与主题对应的新闻组内。按主题分类，各个新闻组构成了一个层次分明的结构。由于每天发布的帖子数量众多，大多数站点（亦称“新闻组服务器”）只接收自己选择的一部分新闻组。即便这样，每天平均也会增加近60MB的新帖子。

在UUCP世界中，通常先从要求的新闻组中收集好所有文章，然后通过一条 UUCP链路发送出去。如果数量较大，一次传不完，便打包后，分成数次传输。打包（压缩）的新闻送至接收站点，由它们执行rnews命令，进行解包和其他处理。

最后，UUCP也是许多通过拨号连接的文件下载站点的一种传输媒介。这些站点通常允许公共访问，让人们免费下载需要的文件或软件。通常要通过 UUCP拨入这些站点，以一名“访客”（Guest）的身份登录，然后从公开的档案区下载需要的文件。这些 Guest账号通常以guest或anonymous（匿名）作为登录用户名，然后用 uucp/nuucp或类似的信息作为登录密码使用。

1.2 TCP/IP网络

尽管UUCP特别适合希望以低廉费用建立拨号网络连接的用户，但在其他许多情况下，这种“存储转发”技术也显得非常不灵活。比如在局域网中，数量不多的机器位于同一幢建筑物中，甚至位于同一层楼、同一个房间里。这些机器相互连接在一起，营造一个能够协同工作的环境。在这种情况下，我们需要在不同的主机间共享文件，或者在不同的机器上运行分布式的应用程序。

所有这些任务都要求以一种全然不同的方式来建立网络连接。此时，不再是将整个文件都“存储”起来，同时为其赋予一个作业说明，而是将所有数据都分割为较小的数据包（Packet），立即转发至目标主机。抵达目标主机后，再在那里重新“组装”回原来的样子。这种类型的网络叫作“包交换”网络。它的最基本的一个应用便是通过网络运行分布式的应用（程序）。当然，这样做的代价会比 UUCP高出许多 —— 这主要反映在软件和协议的复杂程度上。

目前，许多系统（包括非Linux站点）选择的协议都是TCP/IP。在这一节里，我们打算就其基本概念作一番论述。

1.2.1 TCP/IP网络入门

TCP/IP的历史可追溯到1969年由美国国防部高级研究计划局（DARPA）投资进行的一个研究项目。项目的宗旨是建立一个实验性网络，名叫 ARPANET（阿帕网）。1975年，项目完

成，网络正式投入运行。

1983年，新协议套件TCP/IP被接纳为正式标准，网络上的所有主机都必须使用它。当这个阿帕网最终进化成Internet后（阿帕网于1990年正式退出历史舞台），TCP/IP的应用范围已不仅仅在Internet之内。它最引人瞩目的应用是在局域网中；但随着快速数字电话设备比如ISDN的问世，TCP/IP也昭示着拨号网络的未来。

为便于下文对TCP/IP的讨论，我们在此打算以位于芬兰某地的Groucho Marx大学（GMU）为例。在这所大学中，大多数系都运行有自己的局域网（ LAN）。一些系和别的系共享一个网络，另一些系则同时运行着几个网络。所有网络相互间都是连接起来的（互联）。整个校园网通过一条高速链路，接入Internet。

假定我们的机器挂接的是数学系的 LAN，名字叫做Erdos。要想访问物理系中名为 Quark的一个主机，需要执行下述命令：

```
$ rlogin quark.physics
Welcome to the Physics Department at GMU
(ttyq2) login:
```

在提示行，需要输入自己的登录名： Andres，以及正确的密码。随后，便可获得在 Quark机器上的一个外壳，可在其中键入任何命令，就像自己亲身坐在那个系统的控制台前一样。退出这个外壳后，便可返回自己机器（本机）的提示行。

刚才，我们已试验了由TCP/IP提供的一种实时的交互式应用：远程登录！

登录进Quark的时候，有时也想运行一个以 X11为基础的应用，比如一个函数演算程序，或者一个PostScript格式预览程序。要想告诉应用程序你希望将它的窗口显示在自己主机的屏幕上，必须设置DISPLAY环境变量：

```
$ export DISPLAY=erdos.maths:0.0
```

如果现在启动程序，它便会同你的 X服务器联系，而不是同 Quark的服务器联系，并将所有窗口都显示在自己的屏幕上。当然，这要求你在 Erdos机器上运行X11。这里的关键在于，TCP/IP协议允许Quark和Erdos两部主机来回传送 X11数据包，从而营造在单个系统上运行的“ 假象 ”。此时的网络永远是“ 透明 ”的。

在TCP/IP网络中，另一个非常重要的应用是 NFS，亦即“ 网络文件系统 ”（Network File System）。它是让网络“ 透明 ”的另一种形式。 NFS主要用来让我们“ 复制 ”其他主机的目录结构，令其看起来就像本机的文件系统。举个例子来说，所有用户的“ 主目录 ”（Home Directory）都可存放在一台中心服务器中。从这台机器，局域网内的所有主机都复制它的目录。这样一来，用户实际可登录进入网内的任何一台机器，并发现自己处在一模一样的主目录中。类似地，可将那些要求占用大量磁盘空间的应用程序（如 TeX）安装到一台机器，然后将它的目录导出至其他机器。到本书第 10章，我们还会对NFS进行详细论述。

当然，这些仅仅是通过 TCP/IP网络能够做到的一些事情的例子。事实上，用它能做到的事情几乎是无限的。

接下来，我们打算就 TCP/IP的工作原理做一番详细探讨。只有掌握了这方面的知识，才能对自己机器的配置方法做到心中有数。首先从硬件开始，再从它慢慢地延伸开去。

1.2.2 以太网

目前在LAN中广泛采用的硬件形式叫做Ethernet，即“ 以太网 ”。其中包含了一条电缆，主

机通过接头、分路器或者收发器加以连接。简单以太网的安装费用十分低廉，最常见的传输速度是10Mbps（每秒10兆位，而非10兆字节），但也有逐步向100Mbps（百兆网）过渡的趋势。

以太网可划分为三种实施形式。根据采用的电缆规格，可分别称为粗缆、细缆以及双绞线网络。其中，细缆和粗缆以太网使用的都是同轴电缆，只是线缆粗细以及与主机的连接方式有所区别。细缆采用的是BNC接头，俗称T头，需要“拧”入计算机背后的一个圆柱形接头（做在网卡上）。粗缆要求在线缆上打一个小孔，然后用一个“Vampire Tap”连接一个收发器。随后，一个或多个主机建立与这个收发器的连接。细缆和粗缆以太网电缆分别最多可以长达200和500米，所以也叫做10Base-2和10Base-5网络。双绞线电缆则由两个铜芯线对绕合而成，类似于普通的电话线。但是，它通常要求新增额外的硬件。人们也把它叫做 10Base-T网络，其中的“T”代表Twisted pair（双绞线）。

尽管在粗缆以太网中增加一个主机显得比较麻烦，但在增加主机的同时，不会造成网络的中断。而要想在细缆网络中新增一个主机，则必须让网络服务暂停几分钟，因为必须对线缆进行处理（通常需要割断），插入新的接头。

大多数人都愿意选择细缆连接，因其造价相当低廉：一张网卡最多 100元人民币，线缆每米1~3元，其他便不再需要任何费用。但是，假如网络的规模较大，粗缆以太网就显得更为恰当。例如，GMU大学数学系的以太网采用的便是粗缆连接，所以每次有一个主机加入网络时，不会造成整个网络的中断。

以太网技术的一个缺陷是电缆长度（布线长度）有限，所以只适用于 LAN的建设。但是，利用转发器、网桥或路由器，几个独立的以太网网段也可以相互连接到一起。其中，转发器（Repeater）的作用最简单，只是在两个或更多的网段之间复制信号，使各个网段表面上似乎合并成了一个统一的以太网。

以太网工作起来就像一个总线系统，其中的一个主机可将单位长度多达 1500字节的数据包（或“帧”）发给同一以太网内的另一个主机。每个主机都分配有一个 6字节的地址，这种地址需要固化到网卡内部，但也可以通过专门的软件更改。通常，我们用两位十六进制数字的一个序列来表达这种地址，每对数字之间用冒号分隔，例如 aa:bb:cc:dd:ee:ff。

由一个主机发出的数据包可被网内其他所有主机“看到”。但是，只有目标主机才能实际地接收并处理它。假如两个主机同时试图发送数据，便会发生“冲突”。解决这种冲突的办法便是两个主机都取消这一次发送，各自等待随机的一小段时间，再进行发送数据包的尝试。

1.2.3 其他类型的硬件

在规格较大的网络中，比如像前例提到的 GMU大学校园网，以太网通常并非唯一的一种网络硬件安装方式。在 GMU大学，各个系的 LAN都同校园干线连接，后者是一条光纤线缆，运行的是FDDI（光纤分布数据接口）。FDDI采用一种全然不同的形式来实现数据的传送。其基本原理是发送大量“令牌”，令其在整个网络内循环。只有某个主机拿到了一个令牌，才有权将数据送入网络（随那个令牌一道）。FDDI的主要优点在于，它可达到很高的数据传输速度，通常100Mbps是没有问题的，而且布线长度可以高达 200公里。

对于远距离网络链接，通常还有另一种设备类型可供选择，它建立在一种名为 X.25标准的基础上。在美国，有许多所谓的“公共数据网络”（Public Data Network，PDN），比如Tymnet，它们提供的便是这种服务；而在德国，Datex-P网络提供的也是这种形式的服务。X.25要求安装

一种特殊的硬件，名为PAD，亦即“包汇编／反汇编器”（Packet Assembler／Disassembler）。X.25定义了一系列网络协议，由自己专用，但通常也用于连接运行 TCP/IP和其他协议的网络。由于IP包不能直接映射（转换）为X.25格式（反之亦然），所以其他协议的数据包只是简单地封装在X.25包里面，再通过网络传输。

通常，无线电爱好者利用自己的专门设备将计算机连成网络；这种技术称为“包无线电”，或称“火腿无线电”。这些爱好者理所当然也叫做“火腿一族”。火腿无线电使用的协议叫做AX.25，自X.25衍生而来。

另一种技术需要使用速度较慢、但价格便宜的串行线路，用于拨号访问。这要求另一种协议来进行数据包的传送，比如SLIP或PPP，后文还会详细讲述。

1.2.4 网际协议

当然，我们的网络连接并不仅仅限于一个以太网。在最理想的情况下，我们希望能任意使用一个网络，无论它运行于什么硬件形式之上，也无论它由多少个子网组成。比如，在像GMU校园网这样规模较大的网络中，通常包含了数量众多的独立以太网，它们需要以某种形式，相互连接到一起。在GMU大学，数学系运行着两个以太网。其中一个网络包含的都是快速机器，由教授和研究生使用；另一个网络的机器速度较慢，由学生平时上机使用。两个网络都同时接入FDDI校园干线。

这种连接是由一个专用主机控制的，名为“网关”。网关的作用是在两个以太网和光纤线缆之间，通过复制数据包的方式，对数据包的接收和发送进行处理。举个例子来说，如果你在数学系，想访问物理系的局域网上 Quark，然而连网软件不能直接向Quark发送数据包，因为它们不在同一个以太网上。因此必须依靠网关来“转发”数据包。然后，网关（命名为Sophus）再把数据包转发到物理系的、它的同级网关 Niels，由Niels把数据包转发到目标主机。

直接将数据导向远程主机的方法称作“路由”，数据包常常被称作“数据报”。为使一切简单化，数据报交换由一个独立的、与所用硬件无关的协议进行管理。这个协议就是 IP，或者说“网际协议”。我们将在第2章，就IP和路由进行全面论述。

IP的主要好处是：从物理上把不同的网络变成了同一个网络。这就是“联网”技术，而这种“变形网”也就是internet（网间网）。注意，internet和Internet之间，有明显的区别。后者是一个特定的、官方命名的全球化“网间网”。

当然，IP同时还需要一个与硬件无关的定址方案。也就是每台主机对应一个独一无二的32位数，这个数便是该主机的IP地址。IP地址通常用“点分四段”的格式来表示，每一段以十进制表示其四字节中的一个字节，段间用句点分隔开。比如，Quark可能有“0x954C0C04”这样的一个IP地址，它对应的是149.76.12.4，前三个字节由InterNIC注册服务中心分配，标识主机接入的网络，剩下的字节用来标识该主机本身。

大家还会注意到，我们现在有三种地址类型：其一是主机名，比如说 Quark；其二是IP地址；最后是硬件地址，比如那个 6字节的以太网地址。所有这些都必须彼此相符，只有这样，在输入rlogin quark时，联网软件才能给出Quark的IP地址；而且，在“网际协议”向物理系以太网投递数据时，必须找出其 IP地址对应的那个以太网地址 —— 这个地址是最容易搞混淆的。

我们不打算在此深入下去，因为第 2章为大家准备了丰富的IP大餐。目前，大家只需记住的一点是：寻找地址的过程叫做“ 主机名解析 ”（即把主机名映射成 IP地址）和“ 地址解析 ”（把IP地址映射成硬件地址）。

1.2.5 串行线路网际协议

串行线路网际协议 (SLIP)，又叫做“ 串行线路接口协议 ”。是一种允许通过拨号连接进行IP数据包传输的数据链路协议，使得计算机或局域网可以连接到因特网或其他网络中。在串行线路上，常用的标准一般是 SLIP或Serial Line IP。SLIP的修正版称为CSLIP，或者压缩过的SLIP，执行IP头的压缩，以便更好地利用串行链路提供的、相对较低的带宽。另一个不同的串行链路协议是PPP（参见第7章），或者说“ 点到点传输协议 ”（Point-to-Point）。和SLIP相比，PPP的特性更多，其中还包括一个链路协商段。但 PPP领先于SLIP主要表现在没有对 IP数据报的传输进行限制，而且可以发送任何一种类型的数据报。

1.2.6 传输控制协议

当然，从目前看来，从一个主机向另一个主机发送数据报只是任务之一。如果你登录到Quark，还打算在Erdos上的登录协议命令（ rlogin）进程和Quark上的外壳进程之间建立一条可靠的连接，那么，发送端必须把准备收发的信息分成若干个数据包，再由接收端把多个数据包重新组合成一个字符流。这样不仅琐碎，还会额外增加许多工作量。

关于IP，不容忽视的另一点是：不可靠。假设你的以太网上有 10名用户，他们都通过GMU的FTP服务器开始下载XFree86的最新版本。那么所产生的通信量对网关而言，是难以应付的，因为速度太慢了，而且内存也非常的紧张。现在，如果你碰巧向 Quark发送一个数据包，Sophus可能会出现缓冲区空间暂时短缺，导致不能转发数据包的情况。 IP解决此类问题的办法是“ 丢弃 ”它。因此，这个包必定会被丢失。所以，通信主机的责任就是在出现错误的情况下，检查数据的完整性、数据是否完全，以及另行转发。

然而，这是由另一个协议 —— TCP协议（也叫做传输控制协议） —— 来完成的。TCP在IP之上建立了一个可靠的服务。其本质特征是利用 IP，向大家勾划了主机和远程机器上的两个进程之间的一条简单连接，这样一来，就大可不必担心你的数据实际上是沿着哪条路由，以及怎样路由的了。从本质上来说， TCP连接的原理其实就像一条“ 双向 ”管道，两个进程可以边读取，边写入。也可把它想像成“ 打电话 ”。

TCP利用连接涉及的两台主机之IP地址和各主机上的所谓“ 端口 ”号，来标识此类连接的端点。端口可被视为网络连接的“ 附着点 ”（attatchment point）。如果把TCP连接想像成“ 打电话 ”，那么IP地址就是区号（对应一个地区或城市），而端口号就是本地代号（对应各家各户的电话机）。

在“ rlogin ”这一例子中，客户机应用程序（ rlogin）打开Erdos上的一个端口，并将它连接到Quark上的513端口， rlogind服务器将被告知对这个端口进行监听。这样便建立了一条TCP连接。利用这条连接， rlogind执行了认证进程，然后衍生出外壳，该外壳的输入和输出被重新导向TCP连接，这样一来，你在自己机器上输入的任何“ 登录协议命令 ”都会通过 TCP数据流得以传递，并被当作标准输入传给外壳。

1.2.7 用户数据报协议

当然，TCP/IP联网中，TCP并不是唯一的用户协议。虽然它非常适合于 rlogin之类的应用程序，但万万不能用于NFS这样的应用程序。相反地，应该用它的兄弟协议 —— UDP（也叫做“用户数据报协议”）。和TCP协议一样，UDP也允许应用程序和远程机器某个特定端口上的服务取得联系，但它不能为此而建立连接。取而代之的是，可通过它向目标服务发送独立的数据包。

假设你已经通过本系的中心 NFS服务器Galois装入了TeX目录结构，并且打算查看一个关于如何使用 LaTeX的文档。启动编辑器，先在整个文件中读取。但是，要和 Galois建立一条TCP连接、发送并再次发布这个文件，会花相当长的时间。相反地，如果向 Galois发出请求，编辑器则会把该文件分成两个 UDP数据包发送，这种方式要快得多。但美中不足的是， UDP协议不会处理数据包的丢失或中断。若出现这种情况，就由 NFS对此进行处理。

1.2.8 端口问题

端口可以看作是网络连接的附着点。如果一个应用程序想提供一项特定的服务，它就会把自己附着在一个端口上，等待客户机（即所谓的“在该端口上监听”）。想利用该项服务的客户机便在其本地主机上分配一个端口，并连接到远程主机的服务器端口。

端口的一个重要属性是：客户机和服务器之间的连接一旦建立，服务器的另一个副本可能就会附着在服务器端口，监听更多的服务器。也就是说，允许若干个并发登录利用同一个端口513登录到同一个主机。TCP能够把这些连接区分出来，因为它们来自不同的端口或主机。举个例子来说，如果你两次均从 Erdos主机登录进入Quark，第一个rlogin客户机就会采用1023这个本地端口，第二个则采用1022端口。然而，两者均是连接到Quark上同一个513端口上的。

上面的例子说明了端口可用做集合点，客户机通过它连接到另一个特定的端口，以获得特定的服务。为了让客户机知道正确的端口号，两个系统的主管必须在端口号的分配问题上达成一致。对那些用得较广的服务来说（比如 rlogin），其端口号就必须进行集中管理。这是由IETF（因特网工程任务组）来完成的， IETF定期发布一个标题为“已分配号”的 RFC。该RFC对分配给“众所周知”的服务的端口号进行了说明。 Linux采用的是一个文件，名为/etc/services，该文件把服务名映射为端口号。我们将在第 8章对此进行详述。

值得注意的是，尽管 TCP和UDP连接和端口有很大关系，但其端口号之间不会有冲突。在我们前面的例子中，这便意味着 TCP端口513有别于UDP端口513。事实上，这些端口被用作两类不同服务的访问点，也就是 rlogin（TCP）和rwho（UDP）。

1.2.9 套接字库

在操作系统中，执行所有任务的软件和前面所讲的协议通常是内核的一部分。全世界最流行的编程接口是 Berkeley Socket Library（伯克利套接字库）。其名字源于一个流行的比方，即把端口视作套接字，与端口之间的连接视作插拔。它提供了 (bind(2))调用，借此指定远程主机、传输协议和一个程序可以连接或监听的服务（利用连接 (2)、监听(2)和接受(2)）。但是，这个套接字库过于普通，不仅提供了一个基于 TCP/IP套接字的类（AF_INET套接字），还提供了一个可处理本地与远程机连接的类（ AF_UNIX类）。有的实施方案中还可以对其他类进行处理，比如XNS（Xerox连网系统）协议和X.25。

Linux操作系统中，套接字库是标准 libc C-library中的一部分。当前，它只提供了对AF_INET和AF_UNIX套接字的支持，但人们正在努力，提供对 Novell连网协议的支持，以便最后能增添更多的套接字类。

1.3 连网

由于世界各地程序员的共同协作， Linux才成为可能。没有如此广泛的协作， Linux不会有如此夺人的魅力。所以早期开发阶段，寥寥数人着手提供连网功能的工程，其艰难程度简直难以想象。UUCP实施的运作几乎从零开始，基于 TCP/IP连网的工作大约开始于 1992年秋，当时Ross Biro和其他人创建的即是后来人们熟知的“ Net-1 ”。

1993年5月， Ross中止了这项活动的实施， Fred van Kempen开始着手一项新的实现，重新编写主要的代码，这就是 Net-2。1992年夏，首次公开发行 Net-2d（作为0.99.10内核的一部分），而且从此以后， Net-2d得到了几个人的维护和延伸，最值得一提的是 Alan Cox。在对代码进行繁重而庞杂的调试修改之后， Alan Cox在1.0发布之后，把Net-2改成Net-3。

Net-3随同SLIP（用于在串行线路上发送网络数据流）和 PLIP（针对并行线）一起，为各式各样的以太网插板提供了设备驱动程序。 Net-3中有一个TCP/IP实现，其性能像在本地网络环境中一样出色，甚至可以和商业 PC软件人员开发的软件一比高下。

不同的开发倾向

与此同时， Fred继续开发他的Net-2e，主要对网络层的设计进行了大量的修改。在编写代码的同时， Net-2e仍然只是一个测试版软件。关于 Net-2e，最值得一提的是它结合了 DDI，即设备驱动程序接口。DDI针对所有的连网设备和协议，提供了一个统一的访问和配置方法。

然而， TCP/IP连网的另一种实现却源于 Matthias Urlichs，他曾经为Linux和FreeBSD编写了一个ISDN驱动程序。这次，他在内核中集成了一部分 BSD连网代码。

但在可以预见的将来， Net-3似乎停步不前了。 Alan目前在从事 AX.25协议实施的开发，该协议面向“ 火腿无线电发烧友 ”。但已准备开发的内核“ 模块 ”代码仍然会为连网代码带来新的活力，却是一个不争的事实。模块允许爱好者在运行时在内核中添加驱动程序。

尽管这些不同实施都可利用同一个设备，但在内核和设备级还有明显的差异。因此，你不能通过Net-2d或Net-3，利用公用程序对正在运行 Net-2e内核的系统进行配置，反之亦然。这条规则只适用于对内核内部进行处理的命令、诸如 rlogin或telnet之类的应用程序和连网命令。

虽然如此，所有这些不同的网络版本都不会让你无所适从。除非你加入了活动开发行列，否则，就不必担心自己运行的 TCP/IP是什么版本。正式发布的内核总附带有一套连网工具，它们适用于内核中介绍的连网代码。

1.4 系统维护

本书从头到尾，都将为大家讲解安装和配置方面的问题。但提得更多的却是管理。设立一项服务之后，还必须使之能够运行。大多数服务的维护并不繁杂，但有的服务，比如邮件和新闻，则要求你执行常规任务，以保持系统的更新。我们将在后面几章中讨论这些任务。

维护系统过程中，最基本的要求是定期检查系统和应用程序前的日志文件，以了解错误情况和异常事件。一般说来，大家可能想通过编写两个管理外壳脚本来执行该任务，并从

cron周期性地运行这两个脚本。有些主要应用程序（比如 smail和C-News）的源代码中包含有这样的脚本。只须按照自己的要求和喜好，适当地增减。

任何cron作业的输出结果都会寄到一个管理性质的帐号上。默认状态下，多数应用程序都会向根帐号发出错误报告、用法统计和登录文件总结。如果你经常以根的身份登录的话，这是很有用的；另一个较好的方案是把根的邮件转发到你自己的私人帐号，设立一个邮件别名，详情参见第13章。

不管你配置自己的站点时有多细心，始终会产生问题。因此，系统维护的另一层含义是接受用户的抱怨。通常情况下，人们希望通过电子邮件把自己的意思转达给作为“ BOOT ”的系统管理员，但同时还需要其他的地址，以便负责某一特定维护的人员也能够看到这些邮件。举个例子来说，报怨邮件配置故障的电子邮件通常被发送给邮局管理员；而和新闻系统有关的问题则向新闻管理员或新闻组报告。寄给主机管理员的邮件应该重新定向到主机基础网络服务的负责人，如果你运行的是命名服务器，则重定向到 DNS命名服务。

系统安全

网络环境中，对系统进行维护的另一个不容忽视的重要原因是：避免自己的系统和用户受到攻击。管理疏忽的系统往往是那些心怀不轨的人攻击的目标。攻击手段相当多，从猜测密码到以太网偷窥。导致的恶果轻重程度不一，从恶作剧式的邮件信息到数据丢失，或者侵犯用户的隐私权。在讨论出现这些特殊情况的背景时，我们将进一步深入这一话题，同时探讨常见的防御手段。

本小节将讨论几个实例和处理系统安全的一些基本技巧。当然，我们所涉及的主题也许不够全面；它们只对一些常见问题进行了说明。因此，找本安全方面的好书来读是绝对必要的，尤其是在连网系统中。强烈推荐大家参考 Simon Garfinkel所著的《Practical UNIX Security》，该书由O ’ Reilly出版。还可以在www.oreilly.com/catalog/puis/找到此书。

提起系统安全，首先得从优良的系统管理开始。这包括了检查所有重要文件和目录的属主和许可，监视特权帐号的使用情况等等。比如， COPS程序（www.cert.org/ftp/tools/cops）将检查你的文件系统和常用配置文件是否有异常许可或其他异常情况。它还聪明地采用了一个密码套件，强制性地对用户密码实施某一特定的规定，这样，密码就很难被人破译。比如影子密码套件，它要求密码至少要有5个字母，而且还包括大、小写形式的数字和数位。

在编写供网络访问的服务时，务必确定赋予它“ 最低特权 ”，意思是除了为它设计的任务外，不允许它执行别的任务。比如，在根用户和其他特权账号的确需要 setuid程序时，就应该为他们赋予相应的特权。同时，如果希望服务只限于特定的应用时，不要迟疑，在特定应用允许的范围内，尽可能地对它进行严格配置。例如，如果想允许通过你自己的机器启动无盘主机，你必须为之提供 TFTP（日常文件传输服务），以便它们能够从 /boot目录下下载基本配置文件。但是，如果在非限制条件下， TFTP允许全世界的用户都可以从你的系统中下载这些文件。如果不希望如此暴露，就必须把 TFTP服务限定在/boot目录下。

同样的道理，大家肯定还想过如何在自己的局域网内，限制某些主机的用户使用网络服务。这方面的详情，可参考第8章介绍的TCPD。

另一个不容忽视的要点是避免使用“ 危险 ”的软件。当然，我们平常所用的软件都可能是危险的，因为软件不可能总是十全十美的，难免存在错误，有些聪明人士肯定能够找出其

中的漏洞，进而造访你的系统。这是常有的事儿，而且根本无法杜绝。但是，和其他程序比较起来，要求特殊权限的程序似乎更缺乏安全保障，因为任何一个漏洞都可能导致难以想象的后果。所以，在网络中安装setiud程序时，一定要加倍小心，不能漏掉安装文档中的任何东西，以免无意中为“黑客”造成可乘之机。

正如天有不测风云，谁也料不到未来会发生什么事情，因为无论我们多么细心，总不能排除意外的发生。最好的办法是防患于未然。首先从检查自己的系统日志文件着手，但有些“闯入者”可能很聪明，会将自己的造访痕迹清除掉。但是，像 Tripwire（www.cert.org/ftp/tools/tripwire/）这类的工具，它们允许我们对重要的系统文件进行检查，看其中的内容或许可权是否已被篡改。Tripwire对这些文件进行各种校验和计算，并把它们保存在一个数据库内。在之后的运行过程中，将对行些校验和重新进行计算，并和数据库内保存的校验和进行比较，从而有效地防止旁人对系统数据的肆意篡改。

1.5 后续章节提要

下面几个章节将讨论如何配置 TCP/IP连网，如何运行一些主要的应用程序。在着手编辑文件之前，我们将在第2章详细讨论一下IP。已经掌握IP路由原理、知道如何执行地址解析的读者，可跳过此章。

第3章将讨论一些基本的配置问题，比如如何建立内核，如何安装自己的以太网卡等等。至于如何配置自己的串行端口，则在另外一章中讨论，因为它不仅适用于 TCP/IP连网，而且还适用于UUCP。

第5章将帮助大家针对TCP/IP连网，安装自己的机器。其中包括只启用loopback的单机主机安装题示，连接到以太网的主机安装提示。另外还要介绍几个颇有用的工具，大家可以用它们来测试并调试自己的安装结果。第6章将讨论如何配置主机名解析，如何安装名字服务器。

接下来的两章将分别介绍SLIP和PPP的配置和使用。第6章将对如何建立SLIP连接进行解释，并展示一个详细的Dip参考，该工具允许大家对某些必要的步骤进行自动化处理。第 7章将全面介绍PPP和pppd。

第8章将简要介绍一些重要网络应用程序的安装，比如 rlogin和rcp等等。第8章还将全面介绍inetd super是如何对服务进行管理的，以及如何将关系到系统安全的服务限定在信任主机范围内。

接下来的两章将讨论NIS（网络信息系统）和NFS（网络文件系统）。NIS是一个非常有用的工具，用于分发管理信息（比如局域网内的用户口令）。NFS则允许你与网络内的若干个主机共享文件系统。

第11章将广泛介绍Taylor UUCP管理，它是一个免费的UUCP套件。

本书的其余部分将介绍电子邮件和 Usenet新闻组。第12章介绍电子邮件的核心概念，比如邮件地址是什么样的，邮件处理系统如何将电子邮件发到收信人手中的等等。

第13章和第14章分别介绍smail和sendmail的安装。本章对两者都进行了介绍，因为它们各有千秋：对初学者来说， smail更易于安装， sendmail则更为灵活。

第15章和第16章介绍Usenet新闻组内的新闻管理方式，以及如何安装和使用 C-news（一个常用的软件包，用于管理Usenet新闻组）。第17章简要讨论如何安装NNTP daemon，为局域网提供新闻阅读访问。最后，第18章向大家展示如何配置和维护不同的新闻用户。

第2章 TCP/IP网络

大家在把自己的机器连到一个 TCP/IP网络上时，肯定会碰到许多问题，比如如何处理 IP地址和主机名，有时甚至还会碰到路由问题。本章将为大家提供相关背景，帮助大家尽快了解安装时需要什么，下一章将为大家介绍几个处理这些问题的工具。

2.1 网络接口

先撇开联网环境中的设备可能不同这一情况。 TCP/IP定义了一个抽象接口，所有对设备的访问都将通过这一接口。该接口为所有类型的硬件设备提供了同一个操作集，基本上用于包的收发。

针对准备用于联网的各个外设，内核中都必须有相应的接口。比如，以太网接口即称为eth0和eth1，SLIP接口称为sl0、sll，以此类推。在打算针对内核对特定的物理设备命名时，出于配置上的需要，可使用这些接口名。除此以外，它们没有别的用途。

如果要在TCP/IP联网中使用接口，必须为它（它们）分配 IP地址，藉此向网络中的其他机器标识自己。这个地址不同于上面所提的接口名；打个比喻说，如果把接口比作门，那么，其IP地址就是钉在这扇门上的门牌号。

当然，还需要设立别的设备参数；其中之一便是特定的硬件能够处理的数据报的最大字节数，也叫作“最大传输单元”或MTU。其他属性，我们将在后续章节中讨论。

2.2 IP地址

正如前面提到的那样， IP联网协议能够识别的地址是一个 32位的编号。在网络环境中，为每台机器分配的编号必须是独一无二的。如果你运行的是局域网，而且没有与其他网络进行TCP/IP通信传输，就可以按自己的喜好，为各台机器分配编号。但是，如果运行的是面向因特网的网站，其编号必须由ICANN（www.icann.org）分配。

为便于理解，IP地址由四个8位编号（叫做8位元）组成。比如，quark.physics.groucho.edu有一个IP地址是0x954C0C04，该地址表示为149.76.12.4。这一格式就是人们常说的点分四段式。

采用点分四段式的另一个原因是IP地址由两大部分组成：网络部分（即排在前面的8位元）和主机部分（其余的）。根据网络的大小，主机编号可大可小。

为适应不同的需求，这里为大家列出了几类IP编址，它们定义IP地址的不同划分方式：

■ 范围在1.0.0.0到127.0.0.0之间的A类地址。这类地址的网络部分在第一个8位元内。A类编址提供了一个24位的主机编号，也就是说，网络中的主机可以多达160万台。

■ 范围在128.0.0.0和191.255.0.0之间的B类地址。网络部分在前两个 8位元中。可接纳16 320个网络，而每个网络可接纳 65 024台主机

■ 范围在192.0.0.0和223.255.255.0之间的C类地址，其网络部分在前三个8位元内。可接纳近200万个网络， 254台主机。

■ D、E和F类地址的范围在224.0.0.0和254.0.0.0之间。这几类地址尚处于实验阶段，或者是为将来的应用而保留的，不指定任何网络。

回头看看前一章的示例，我们会发现 149.76.12.4（Quark的IP地址）指的是 B类网络149.76.0.0中的12.4主机。

需要注意的是，有两个地址是保留地址，它们是：0.0.0.0和127.0.0.0。前者称为默认路由，后者称为回送（loopback）地址。默认路由和IP协议对数据报的路由方式有关，它主要用于简化IP路由选择信息，详情参见下一小节。回送地址 127.0.0.0是为本地主机的IP通信保留的。通常，127.0.0.1这个地址是分配给本地主机上的特殊接口的，也就是所谓的“回送接口”（loopback interface）。回送接口就像一个封闭式的回路。任何自 TCP或UDP传给该接口的IP包都将被返回始发地，就像它们刚从某个网络返回一样。有了回送接口，就可以在单机主机上开发和测试联网效果，利用网络软件了。这类情况并不罕见，比如，许多 UUCP站点根本就没有IP连接功能，但人们仍然想在这些站点上运行 INN新闻系统。

2.3 地址解析

现在，大家已经了解了 IP地址的构成情况，可能还想进一步了解它们是如何用于以太网，为不同主机定址的。毕竟，以太网协议按 6位8位元编号（和IP地址完全没有关系）来识别主机。

是的，我们需要一种机制，能把 IP地址映射为以太网地址。这就是所谓的“地址解析协议”（简称ARP）。事实上，ARP的使用并不局限于以太网，它还用于其他类型的网络，比如“火腿无线电网”。ARP的基本思路源于此：必须在150个人当中寻找一个名为某某的人，大多数人都会采用类似的办法——四处转，高声呼叫他的名字，因为对方听到叫声，肯定会答应。

ARP想找出与具体IP地址对应的以太网地址时，将利用“广播”这一以太网特性，这样就能将一个数据报同步传递到网络中的各个角落。 ARP发送的广播数据报中，包含一个 IP地址查询。每个收到广播数据报的主机都会将其中的 IP地址和它自己的进行比较，如果相同，就向发出查询的主机返回一个 ARP应答。这样一来，发出查询的主机就可以从收到的应答中，获悉对方的以太网地址。

当然，有些人可能觉得奇怪：全世界的以太网举不胜举，主机是怎样在如此庞大的体系中找到自己的目标的呢？要解决这一困惑，还需要了解路由选择，有了路由选择，才能够在网络中找到一台主机的物理位置。这是我们下一小节的主题。

关于ARP，我们还有话说。一旦主机知道了自己想要的以太网地址，它就会把该地址保存在它自己的ARP缓冲区内，这样在下一次查询该地址时，它会直接向目标主机发送数据报。但是，要想永久性地保留该地址信息是很不明智的；比如说，远程主机的以太网卡可能因为技术上的原因已发生替换，所以你保留的 ARP条目将毫无意义。因此，要执行另一次 IP地址查询，就应该时不时地丢弃ARP缓冲区内的条目。

有时，还有必要找出与具体以太网地址相关联的 IP地址。比如说，想从网络上的一台服务器启动无盘客户机这样的事，在局域网内是家常便饭。但是，无盘客户机事实上根本就没有任何关于它自己的信息——除开它自己的以太网地址！所以，它能做的只能是广播一条消息，请求根服务器将它的 IP地址告诉它。这牵涉到另一个协议，名为“逆向地址解析协议”（简称RARP）。它和BOOTP协议一起，为网络上的无盘客户机的启动运行定义一个进程。

2.4 IP路由

2.4.1 IP网络

大家可能有这种经历；在写信给某人时，通常会在信封上写明收信人的确切地址，指定国家、地区、省份，城市以及详细到几栋几号。然后，再把信投入邮箱，邮政服务将把信送达目的地：先送到指定国家，再由那个国家的邮政服务将其分发到相应的省或地区。邮政服务的层次结构非常清楚：不管在哪里寄信，本地的邮政主管都知道信件投递路径，并将进行信件转发，并不注重信件在目标国内的投递方式（注意，邮件投递系统只注重完成任务，而不管如何去完成任务）。

IP网络其实与邮件投递系统类似。整个因特网由无数个称为独立系统的网络构成。每个这样的系统都会在各自的成员主机之间执行路由，所以投递数据报的任务实际上就是找出通往目标主机所在网络的路径。也就是说，只要数据报被传到特定网络上的任何一台主机，怎样到达最终的目标主机，则由这个特定网络自行负责。

2.4.2 子网

我们还可以从被分为网络部分和主机部分的 IP地址中，看出IP网络的结构。默认情况下，目标网络（即目标主机所处的网络）是从 IP地址的网络部分衍生的。所以， IP网络编号相同的主机应该处于同一个网络内，反之亦然。但一个独立系统中，往往包含不止一个 IP网络。

由于一个网络可能由上百个小型网络集合而成，各小型网中还有诸如以太网的更小的物理网络单元，所以，在这种网络内部提供类似于前面的方案是非常有意义的。这样，我们就可以把IP网络分成若干个子网。

子网主要负责从自己所处的 IP网络，把数据报投到特定范围内的 IP地址。A、B或C类地址的识别，是通过 IP地址的网络部分来完成的。但是现在的网络编号被扩展到包含主机部分的位数。这些被视作子网编号的位数就是所谓的子网掩码（或网络掩码）所赋予的。它同样是一个32位的编号，用于为IP地址的网络编号指定位掩码。

Groucho Marx大学(GMU)校园网就是一个很好的范例。它采用的是 B类地址，其网络编号是149.76.00，所以其网络掩码是255.255.0.0。

从其内部结构来说，GMU大学校园网由若干个小型网组成，比如各个系的局域网。所以，这个IP网络被分为254个子网，IP地址在149.76.1.0到149.76.254.0之间。举个例子来说，理论物理系分配的IP地址是149.76.12.0。校园主干网的地址是 149.76.1.0。这些子网共享同一个 IP编号，其中的第三个8位元是用来区分它们的。所以，它们的子网掩码将是 255.255.255.0。

值得注意的是子网只是一个网络内部分区。子网是由网络拥有者（或管理员）一手炮制的。通常情况下，子网主要用于反映现有的地址边界，用于各子网间的物理上（两个以太网之间）、管理上（两个系之间）或地理上的边界。但是，这类结构不仅会影响整个网络的内部行为，而且子网只能本地识别，其地址仍然被看成是标准的 IP地址。

2.4.3 网关

子网不仅能带来结构上的好处，还时常用来反映硬件边界。具体物理网络上的主机，比

如以太网，是非常受限的：它能够直接与之交谈的主机只能是本网络内的。要对其他的主机进行访问，只有通过所谓的“网关”来进行。网关是同时连接两个或两个以上物理网络的主机，被配置为执行网络间的包交换。

对IP网络来说，要想轻松识别主机是否在本地网络，不同的物理网络只能属于不同的 IP网络。比如，网络编号149.76.4.0是为数学系局域网上的主机保留的。在向 Quark发送一个数据报时，Erdos主机上的网络软件立即就能知道该数据报来自 149.76.12.4这个IP地址，而且其目标主机处于另一个物理网络上。因此，这个数据报只能通过一个网关（默认设置是 Sophus）抵达目的地。

Sophus本身连接了两个子网：数学系的局域网和校园主干网。它分别通过两个不同的接口（eth0和fddi0）访问这两个子网。现在，我们为这个网关分配什么样的 IP地址呢？应该根据149.76.1.0子网进行分配，还是根据149.76.4.0子网进行分配？

答案是：两者都要。在提到数学系局域网上的主机时，就应该用 149.76.4.1这个IP地址；在提及校园主干网上的主机时，就应该用149.76.1.4这个地址。

所以，这个网关就有两个IP地址。这两个地址——还有其相应的网络掩码——都绑在接口上（通过这个接口访问子网）。因此，接口和Sophus地址之间的对应关系就会像表 2-1列出的那样。

表2-1 接口和Sophus地址之间的对应关系

接 口	地 址	网络掩码
eth0	149.76.4.1	255.255.255.0
fddi0	149.76.1.4	255.255.255.0
l0	127.0.0.1	255.0.0.0

最后一条说明的是回送接口l0，我们在前面已经介绍过它。注意，随两个子网上的主机同时出现的还有两个地址。

一般说来，把地址附于主机和把地址附于其接口之间有些细微差别，但我们可以忽略它们之间的差别。对处于同一个网络的主机来说，比如 Erdos，一般能够很确切地指出主机的 IP地址是多少。比如说，这是一个以太网接口，它的 IP地址是这样的。但是在提及网关时，其间的区别则是不容忽视的，必须指明地址是附于主机上，还是附于其接口上。

2.5 路由表

接下来的重头戏是：把数据报投到远程网络时， IP协议是如何选择网关的。

在此之前，我们曾见过 Erdos网关的工作流程：它收到发向 Quark的数据报后，便对其目标地址进行检查，并发现其目标主机不在本地网络内。所以，它把该数据报发给默认网关 Sophus，现在就看网关怎样运作了。 Sophus识别出Quark不在它直接连接的任何一个网络内，所以，它就开始寻找另一个网关来接替它的工作。于是它选中了 Niels，这是通往物理系局域网的网关。如此一来， Sophus就需要更多的信息，把目标网络和一个适当的网关关联起用。

它采用的路由选择信息 IP实际上就是一个表，把网络和准备抵达的网关链接起来。一般说来，我们必须提供一个catch-all条目(默认路由)；它是一个与0.0.0.0网络关联在一起的网关。发向未知网络的所有包都会通过这个默认路由得以发送。针对 Sophus网关，它的路由表就像表2-2一样：

表2-2 Sophus的路由信息表

网 络	网 关	接 口
149.76.1.0	…	fddi0
149.76.2.0	149.76.1.2	fddi0
149.76.3.0	149.76.1.3	fddi0
149.76.4.0	…	eth0
149.76.5.0	149.76.1.5	fddi0
…	…	…
0.0.0.0	149.76.1.2	fddi0

对直接与Sophus连接的网络来说，通向它的路由不需要网关，所以显示的网关条目是“ - ”。

路由表的建立方式较多。对小型局域网来说，在启动时间（参见第 3章），利用route命令，手工构建路由表并把它们投入 IP网络，通常是最有效的。对于大型的网络，需要根据路由daemon，在运行时构建路由表并适时进行调整；路由信息运行于网络内的中心主机上，并在其成员网络间实行路由信息交换，从而计算最佳路由。

根据网络的不同大小，可能会用上不同的路由协议。对独立系统（比如 Groucho Marx校园网）内的路由选择来说，将采用内部路由协议。最优秀的是 RIP协议，即路由信息协议。它是由BSD路由daemon实施的。针对独立系统间的路由，则必须使用诸如 EGP（外部网关协议）或BGP（边界网关协议）之类的外部路由协议；这类协议（包括 RIP）已经被用于康奈尔大学的通道daemon中。

度量值

基于RIP的动态路由信息将根据网关“ 跳 ”（hop）数来选定抵达目标主机或网络的最佳路由。也就是说，在抵达目标主机或网络之前，数据报必须经过的网关越少，其 RIP级别就越高。一个网关即为一个跳。超长路由将被视为不可用路由而被丢弃。

要想利用RIP来管理本地网络中的路由信息，必须在所有主机上运行 gated（通道）。在启动时，通道将对所有激活的接口进行检查。如果通道发现激活接口不止一个（不包括回送接口在内），就会假设主机正在进行不同网络间的包交换，从而主动地进行路由信息的交换和广播。如若不然，它就会被动地接收 RIP更新信息，更新本地路由表中的信息。

在广播本地路由表中的信息时，通道将对路由长度进行计算，也就是说用与路由表中的条目关联在一起的度量值来衡量路由长度。这个度量值是系统管理员在配置路由时设置的，它应该能反映出利用该路由产生的花费。因为，对主机直接连接的子网来说，通向它的路由之度量值应该始终为 0，而对通过两个网关的路由，其度量值必须是 2。但需要注意的是，在没有使用RIP或通道时，大可不必理睬度量值。

2.6 Internet控制消息协议

IP还有一个“ 伴侣 ”协议。它就是Internet控制消息协议（ICMP），它是内核联网程序用以与错误消息和其他主机进行通信的协议。比方说，假设我们又回到 Erdos，并打算登录到Quark的12345端口，但这个端口上没有监听进程。所以，发向这个端口的第一个 TCP包就会抵达Quark，网络层将认出这个包并立即向Erdos返回一条ICMP消息，指出“ 不能抵达指定端口 ”。

ICMP能够识别的消息相当多，而且大多数都能对错误情况进行处理。然而，其中有一

条非常有意思的消息，叫作“重定向”消息。它是在有更短路由的情况下，发现另一个主机正把它用作一个网关时，由路由选择模块生成的。例如，在启动之后， Sophus的路由表可能会不完整，其中包含通向数学系局域网和 FDDI主干网的路由，以及通向 Groucho计算中心的网关（gcc1）的默认路由。因此，任何一个发向 Quark的包都会被发送到gcc1，而不是物理系局域网的Niels网关。在收到这类包后， gcc1将注意到这一路由非常糟糕，所以在把包转发到Niels时，向Sophus返回一条ICMP重定向消息，并将最佳路由告诉它。

现在看来，手动配置路由似乎比必须设立路由简单的多。但要注意，单纯依赖于动态路由方案以及RIP和ICMP重定向消息，始终不是上策。在验证某些路由信息是否真正可靠时，ICMP重定向消息和RIP能够提供的选择很少，甚至没有。这样某些恶意的、一无是处的包将扰乱你的整个网络通信，甚至可能导致网络瘫痪。鉴于此，联网程序有几个版本，对影响网络路由的重定向消息进行了处理，令其只能对主机路由进行重定向。

2.7 域名系统

2.7.1 主机名解析

如上所述，TCP/IP连网协议中的地址利用的是 32位的编号。但是，有几个人能够一口就答上来，某某主机的 IP地址是什么呢？因此，主机一般都有一个“普通”的（比如“高斯”或“异人”）主机名。然后，由特定的应用程序负责找出和这个主机名对应的 IP地址。这个过程就叫作“主机名解析”。

对想找出与具体主机名对应的 IP地址的应用程序来说，不必为主机和 IP地址的查找提供它自己的例程。相反地，它依靠大量的库函数进行透明操作，其中有： gethostbyname(3)和gethostbyaddr(3)。按照惯例，这两个函数和大量的相关进程都被归在一个独立的库内，这个库叫作解析器库；Linux中，它们是属于标准 libc的。所以说白了，这个函数集合就是“解析器”。

现在，在像以太网之类的小型网络上，甚至在一个以太网聚簇上，要维护主机名和 IP地址的对应表不是件难事。维护信息通常保存在一个名为 /etc/hosts的文件中。在增添或删除主机名或重新分配IP地址时，只须根据所有的主机，更新主机名即可。显而易见，这对由若干台机器组成的网络来说，是颇为头疼的。

要解决这一问题，方法之一是NIS（网络信息系统），这是Sun公司开发的。简单地说，叫作YP或黄页。NIS把主机文件（和其他信息）保存在一个主要主机的数据库内，客户机如有需要，就可从该数据库内提取。但是，这个方法仍然只适用于中等大小的网络，比如局域网，因为它只对整个主机数据库进行集中维护，并把它分发到各个服务器。

在因特网上，地址信息最初也是保存在一个独立的HOSTS.TXT数据库内。这个文件的维护在“网络信息中心”（NIC）进行，而且必须由所有参与站点进行下载和安装。随着网络的扩大，这一方案的问题就越来越突出。除了定期安装HOSTS.TXT涉及的管理开销外，各个服务器的下载量也越来越大。更为严重的是，所有主机名都必须利用NIC进行注册，以保证主机名不重复。

这就是1984年新的主机名解析方案出台的原因。它就是域名系统（ DNS）。DNS是保罗·莫克皮特里斯设计的，完满解决了上面提到的两大问题。

2.7.2 输入DNS

DNS采用域分层结构来管理主机名。一个域是若干个站点的集合，因为这些站点可以组成一个网络（例如，校园网内的所有机器，或 BITNET上的所有主机），而且因为它们都属于某个特定的组织（比如美国政府），或仅仅因为它们的地理位置相当接近。例如，所有的大学可以归为一个.edu域，而各个大学或学院又分为一个单独的子域。如果为 Groucho Marx大学指定的是groucho.edu域，那么，为数学系局域网分配的域就是 maths.groucho.edu。数学系局域网内的各台主机分到的域就是再在前面的域名前加上主机名，比如 Erdos的就是erdos.maths.groucho.edu。这就是所谓的“完整资格域名”（FQDN），它能够唯一地标识全球各地的主机。

这个域分层结构的根是一个单独的点（.），名副其实地称作根域（rootdomain），所有的域都包含在这个根域内。为指明一个主机名的确是一个完整资格域，而不是相对于某一（隐式）本地域的域名，有时会把它表示成一个追尾点。它表示该名的最后一个组件是根域。

根据其在域名分层结构中的位置，域可以称作顶级域、二级域或三级域。多级子分区不是没有，但极为罕见。下面是一些大家经常能见到的顶级域：

- .edu （主要在美国）教育部门，比如大学
- .com 商业组织，公司
- .org 非商业组织（这个域中通常是些UUCP网络）
- .net 网络上的网关和其他管理性主机
- .mil 美国军方
- .gov 美国政府部门
- .uucp 正式场合中，对以前用做UUCP名的所有站点名来说，如果没有为它们指定域，就会被统统归入这个域内。

从技术上来讲，前4个域属于Internet的美国部分。但这个域中同样包含有非美国站点。特别是.net域中，此类非美国站点比比皆是。但是，.mil和.gov是美国专用域。

除美国外的其他国家都可以使用得名于 ISO-3166中定义的两个字母，来作为其顶级域。比如芬兰，它使用的是 .f域；法国使用的是 .fr域；德国使用的是 .de域；澳大利亚使用的则是.au域。在这个顶级域下，各个国家可按照自己的方式组织主机名。比如澳大利亚，它有类似于国际顶级域的二级域，名为com.au和edu.au。其他的国家，比如德国，则不采用这种结构，而是采用一个稍长的名字直接指向正在运行特殊域的组织（或企业）。例如，ftp.informatik和uni-erlangen.de此类的主机名是很不常见的，但在德国，类似的主机名却非常普遍。

当然，这些国家域并不能代表这个域下面的主机真正就位于该国；它只是表示这台主机在这个国家的NIC（国家Internet中心）已经注了册。比如，一个瑞士厂商可能在澳大利亚有一个分部，仍然可以用.se顶级域来注册其下属所有主机。

现在，对域名分层结构内的域名空间进行组织，就能很好地解决域名唯一性的问题。利用DNS，主机名在其自己所处的域内，必须是独一无二的，这样才能保证它的名字有别于全球各地的主机。此外，完整资格域名也非常容易记，正因为有了它们，便可将一个大型的域分成若干个子域。

但DNS还有别的好处：它允许你选择子域内的授权者作为它的管理员。比如， Groucho计

算中心的维护人员可能为各系创建一个子域；我们已见识过数学系和物理系子域。在发现物理系局域网太大，太混乱，以至于很难自外部进行管理时（毕竟，学物理的人是出了名的“放荡不羁”），计算中心的维护人员会把physics.groucho.edu域的控制权移交给该网络的管理员。然后，该网络的管理员就可能自由地使用自己喜欢的主机名，并在不与网络外部发生冲突的前提下，按照自己的方式为主机分配IP地址。

最后，域名空间分为若干个区，每个区都包含在一个域内。注意，区和域之间的细微差别：groucho.edu域内涵括Groucho Marx大学的所有主机，而groucho.edu区内只包括计算中心直接管理的主机，比如数学系局域网内的主机。而物理系的主机则属于另一个不同的区，名为physics.groucho.edu。

2.7.3 利用DNS进行名字查找

乍一看，域和区几乎令域名解析复杂不堪。毕竟，如果没有集中式授权机构对主机名分配的控制，小小应用程序怎能识别出这些纷繁复杂的域名？

现在该来谈谈DNS的妙用了。如果想找到Erdos主机的IP地址，DNS就会说，去问管理这台主机的人，他们会告诉你的。

事实上，DNS是一个巨大的分布式数据库。它是通过一个所谓的“域名服务器”来实现的，这些域名服务器将提供具体域或域集合相关的信息。对每个区而言，至少有两个域名服务器中保存有那个区中的所有主机之验证信息。要想获得Erdos的IP地址，只须和groucho.edu区的域名服务器进行沟通，然后，它就会返回你所需要的数据。

大家可能会这样想：“说时容易，做时难。我怎样才能抵达Groucho Marx大学的域名服务器呢？”如果你的计算机没有装地址解析先知，DNS有。在你的应用程序想查找关于Erdos的相关信息时，它会与一个本地域名服务器取得联系，该服务器将为此实施所谓的“交互式”查询。向根域的一个域名服务器发出查询之后，该服务器便要求得到erdos.maths.groucho.edu的IP地址。根域名服务器认出这个名字不属于自己的辖区，而是属于.edu域下面的一个辖区。所以，它就会要求你与.edu区域名服务器取得联系，并将列有所有.edu域名服务器及其地址的清单封装起来。然后，你的本地域名服务器将开始对清单中的域名服务器一一进行查找，比如a.isi.edu域名服务器。它采用的方式类似于根域名服务器的方式，它知道groucho.edu处的人们在运行它们自己的区，并把你引向他们的服务器。最后，本地域名服务器再向其中一个服务器提出查找Erdos，Erdos所在的域名服务器认出它是属于自己这一区之后，便返回其对应的IP地址。

现在看来，查找一个小小的IP地址似乎会产生很大的网络开销，但事实上，和目前尚未解决的HOSTS.TXT传输比较起来，简直不足一提。但仍有必要对这一查找进行改进。

为了缩短查找时间，域名服务器将把获得的信息保存在自己的本地缓存内。所以，下一次你的网络中，有人想查找groucho.edu域内的主机之IP地址时，你的域名服务器就不必费心了，直接联系groucho.edu域名服务器即可。如果域名服务器不保存获得的信息，DNS和其他的方法一样糟糕，因为每次查找都涉及到根域名服务器。

当然，域名服务器不会永久性地保存这一信息，隔段时间后，它就会丢弃这一信息。何时丢弃由生存时间（也就是TTL）决定。DNS数据库中的每项资料都有一个由区管理员分配

的一个TTL。

2.7.4 域名服务器

容纳某一区内主机信息的域名服务器叫做这个区的验证服务器，有时也被称为主域名服务器。任何对区内主机的查询最终都会涉及到主域名服务器。

为了保证整个区的连贯性，其主域名服务器必须能够得到同步更新。这是通过令其中一个主域名服务器成为首要服务器来实现的。这个首要服务器定期令传输区数据的其他域名服务器作为其从属服务器，从而从数据文件中载入该区信息。

要有若干个域名服务器的原因之一是：分担工作负荷。另一个原因是备用。如果一个域名服务器良性“失效”，比如系统崩溃或丢失网络链接时，所有的查询都会转向其他的服务器。当然，这一方案并不能有效地避免服务器故障（导致所有的 DNS请求做出错误的应答）和服务器程序本身出现软件错误。

当然，我们也可以设想一些域内不存在作为验证域名服务器的情况（但一个域名服务器至少能为本地主机和127.0.0.1逆向查找提供域名服务）。不管怎么说，这类服务器都是相当有用的，因为它仍然能够针对运行于本地网络的应用程序实施 DNS查询。因此，它们通常也被称为“caching-only”（只用于缓存）服务器。

2.7.5 DNS数据库

由上可知，DNS不仅能够处理主机的IP地址，还能够交换关于域名服务器的信息。事实上，DNS数据库内可能有整整一打不同类型的条目。

DNS数据库内，一条单一的信息叫作一条资源记录，或简称 RR。每条记录都有一个与之关联的类型（描述了该记录代表的数据类别）和一个类（指定该记录适用的网络类型）。后者可根据不同的编址方案（比如 IP地址，是IN类；Hesiod网络则是MIT）需求进行调节。标准的资源记录类型是A记录，它把一个完整资格域名和一个IP地址关联在一起。

当然，一个主机可以有若干个主机名。但是，必须将其中一个主机名标识为正式名或规范名，而其他的则是可以代表前者的别名。其间的区别是：规范主机名是 A记录所关联的主机名，而其他的则只有一条CANME类型的记录，该记录指向规范主机名。

关于所有的记录类型，将留在下一章深入讨论。这里只是简要介绍一下。

除了A和CNAME类记录外，我们还看到文件顶部有一条特殊的记录。这是 SOA资源记录，表示Start of Authority，其中容纳服务器所属的区之普通信息。比如，它包含所有记录的默认TTL值。

注意，不是以点“.”结尾的示范文件中，所有主机名都应该被解释为与 groucho.edu域有关。SOA记录中的特殊名“@”代表这个域本身的域名。

由上可知，groucho.edu域的域名服务器多少应该知道物理区，使之能够将主机查询引向这些主机各自的域名服务器。这通常是通过两条记录来完成的：指定该服务器之 FQDN的NS记录和把地址与主机名关联在一起的 A记录。由于这两条记录把域名和空间集中保存在一起，所有它们通常也被称作“glue records”。对真正保存子区内主机相关信息的父区来说，它们是唯一的记录实例。指向physics.groucho.edu域的域名服务器的glue记录如清单2-1所示。

清单2-1 物理系采用的程序，摘自named.hosts文件

```
;
; Authoritative Information on physics.groucho.edu
@                       IN      SOA             (
                        niels.physics.groucho.edu.
                        hostmaster.niels.physics.groucho.edu.
                        1034                    ; serial no
                        360000                  ; refresh
                        3600                    ; retry
                        3600000                 ; expire
                        3600                    ; default ttl
                       )
;
; Name servers
        IN      NS      niels
        IN      NS      gauss.maths.groucho.edu.
gauss.maths.groucho.edu. IN A           149.76.4.23
;
; Theoretical Physics (subnet 12)
niels                   IN      A       149.76.12.1
                        IN      A       149.76.1.12
nameserver              IN      CNAME   niels
otto                    IN      A       149.76.12.2
quark                   IN      A       149.76.12.4
down                    IN      A       149.76.12.5
strange                 IN      A       149.76.12.6
...
; Collider Lab. (subnet 14)
boson                   IN      A       149.76.14.1
muon                    IN      A       149.76.14.7
bogon                   IN      A       149.76.14.12
...
```

2.7.6 逆向查找

除了查找属于一个主机的 IP地址外，我们有时还希望查找和 IP地址对应的规范主机名。这就是所谓的“逆向映射”，有几个网络服务用它来验证客户机的身份。在使用一个独立的主机文件时，逆向查找只在该文件中查找问题中 IP地址所对应的主机名。如果用 DNS，当然还要对问题外的名字空间进行彻底查找。 DNS创建了一个in-addr.arpa特殊域，其中包含所有主机的IP地址，这些地址采用的格式是点分四段式。例如， 149.76.12.4这个IP地址对应的主机名是4.12.76.149.in-addr.arpa。把这些域名和其规范主机名链接起来的资源记录类型是 PTR。具体程序参见清单2-2。

清单2-2 GMU采用的程序，摘自named.hosts文件

```
;
; Zone data for the groucho.edu zone.
@                       IN      SOA             (
                         vax12.gcc.groucho.edu.
                         hostmaster.vax12.gcc.groucho.edu.
```

```
                        233             ; serial no
                        360000          ; refresh
                        3600            ; retry
                        3600000         ; expire
                        3600            ; default ttl
                        )
....
;
; Glue records for the physics.groucho.edu zone
physics                 IN      NS      niels.physics.groucho.edu.
                        IN      NS      gauss.maths.groucho.edu.
niels.physics           IN      A       149.76.12.1
gauss.maths             IN      A       149.76.4.23
...
```

创建一个特区通常意味着其管理员能够拥有分配 IP地址的绝对支配权。由于他们手中有一个或若干个IP网络或子网，所以DNS区和IP网络之间可能会存在一对多的映射关系。比如，物理条由149.76.8.0、149.76.12.0和149.76.14.0三个子网组成。

结果是，in-addr.arpa域内的新区必须随物理区一起创建，而且供该系的网络管理员专用，它们是：8.76.149.in-addr.arpa、12.76.149.in-addr.arpa和14.76.149.in-addr.arpa。否则的话，在Collider实验室安装一台新主机时，就会要求新区与其父域取得联系，以便将新地址输入它们的in-addr.arpa区文件。

子网12所用的区数据库参见清单2-3。

清单2-3 子网12所用的数据库，摘自in-addr.arpa文件

```
;
; the 12.76.149.in-addr.arpa domain.
@                   IN      SOA     {
                        niels.physics.groucho.edu.
                        hostmaster.niels.physics.groucho.edu.
                        233 360000 3600 3600000 3600
                        }
2                   IN      PTR         otto.physics.groucho.edu.
4                   IN      PTR         quark.physics.groucho.edu.
5                   IN      PTR         down.physics.groucho.edu.
6                   IN      PTR         strange.physics.groucho.edu.
```

这些新区之父区所用的记录，则参见清单2-4。

清单2-4 网络149.76所用的记录，摘自named.rev文件

```
;
; the 76.149.in-addr.arpa domain.
@                     IN        SOA           {
                       vax12.gcc.groucho.edu.
                       hostmaster.vax12.gcc.groucho.edu.
                       233 360000 3600 3600000 3600
                      }
...
; subnet 4: Mathematics Dept.
```

```
1.4                IN      PTR       sophus.maths.groucho.edu.
17.4               IN      PTR       erdos.maths.groucho.edu.
23.4               IN      PTR       gauss.maths.groucho.edu.
...
; subnet 12: Physics Dept, separate zone
12                 IN      NS        niels.physics.groucho.edu.
                   IN      NS        gauss.maths.groucho.edu.
niels.physics.groucho.edu. IN  A 149.76.12.1
gauss.maths.groucho.edu. IN  A   149.76.4.23
```

上面的记录产生的重要结果是验证区只能被视作 IP网络的超级子集，而且更严格地说，这类网络的网络掩码必须根据字节边界来定。 Groucho Marx大学的所有子网的网络掩码都是255.255.255.0，因此，应该为每个字网创建一个 in-addr.arpa区。但是，如果各子网的网络掩码是255.255.255.128，我们就可以为子网 149.76.12.128创建验证区，因为无法告知 DNS“ 12.76.149.in-addr.arpa域已经被分为两个特区，所以主机名分别是 1到127，和128到255 ”。

第3章　网络硬件的配置

迄今为止，关于网络接口和常见的 TCP/IP问题，我们已谈了不少，但尚未真正接触到内核中的“ 联网程序 ”访问硬件时所发生的事。鉴于此，还必须为大家讲讲接口和驱动程序这两个概念。

首当其冲的是硬件本身。比如以太网卡：它是一片环氧基树脂卡，上面布满了微晶片，这些微晶片上还有些编号，这块卡插在计算机内的一个插槽内。我们通常称之为设备（device）。

如果希望能够使用以太网卡，你必须在自己的内核中准备一些特殊的功能，使之能识别这种设备特有的访问方式。这就是所谓的设备驱动程序。例如， Linux中就有几个以太网卡驱动程序，这几个程序的功能都差不多。其中，最有名的是 Becker串行驱动程序（得名于其作者Donald Becker）。另一个是D-Link驱动程序，该程序对附着在一个并行端口上的 D-Link封装适配器进行控制。

在提到驱动程序“ 控制 ”设备时，其含义究竟是什么？首先回头看看上面提到的以太网卡。驱动程序必须能够与卡上的外设进行通信：它必须向卡发送命令和数据，而卡也应该将驱动程序发来的所有数据统统投递出去。

PC中，这种通信常常发生在一个 I/O内存区内，该内存区对应板载寄存器。内核发送给卡的所有命令和数据都必须通过这些寄存器。 I/O内存区一般被描述为起点或基础地址。以太网卡的典型基础地址是0x300或0x360。

通常情况下，不要去在意基础地址之类的硬件问题，因为内核会在启动时，对设备位置进行侦测。这就是所谓的“ autoprobing ”（自动侦测），意思是如果已安装特定的以太网卡，内核就会对若干个内存位置进行读取，并把它所读取的数据和它看到的数据进行比较。但是，也有内核不能自动侦测的以太网卡；比如，一些便宜仿造标准网卡的以太网卡。另外，内核在启动时，只能试着侦测一个以太网设备的位置。如果你使用的以太网卡不止一个，就必须清楚地将这些网卡的情况告诉内核。

另一个必须告诉内核的参数是中断请求通道(interrupt request channel)。有的硬件组件在特别需要重视时，通常可能中断内核。比如，数据抵达或出现特殊的情况。在 PC中，15个中断通道（编号0、1、3一直到15）中，其中之一可能会发生中断。分配给硬件组件的中断编号叫作“ 中断请求编号 ”或 IRQ（IRQ2和9是一样的，因为PC有两个层叠式中断处理器，每个处理器都有8个IRQ；辅助处理器连接的是主处理器的 IRQ 2）。

正如我们在第1章中所讲的那样，内核通过一个所谓的接口访问设备。接口提供了适用于所有硬件的一个抽象功能集，比如收发数据报。

接口的识别是通过接口名进行的。接口名是在内核内部定义的，而不是 /dev目录下的设备文件。常见的接口名用于以太网接口的 eth0、eth1等等。为设备分配接口常常和设备的配置顺序有关；比如，第一块以太网卡是eth0，下一个将是eth1，以此类推。唯一例外的是SLIP接口，它是动态分配的；也就是说，只要一建立 SLIP链接，就会为串行端口分配一个接口。内核将

在启动时，显示它所侦测的设备和它所安装的接口。

3.1 内核配置

在运行一个系统时，应该对内核的构建非常熟练。这方面的基础知识可参见马特·维尔希所著的《安装和入门指南》（这本指南也包括在 Coriolis Group的《Linux系统编程白皮书》内）。本小节，我们只为大家讨论一些连网所涉及的配置选项。

在运行make config时，首先会要求你回答几个常见的配置问题，比如，是否希望内核数学模拟等等。其中之一是问你是否需要 TCP/IP支持。必须回答“ Y ”（是），才能获得内核连网能力。

3.1.1 内核选项1.0及以上版本

注意 本小节无示例。要查找更新内容，请参考在线版。

结束常见配置询问之后，配置会继续问一些不同特性方面的问题，比如 SCSI驱动程序等。接下来的问题仍然和连网支持有关。由于 Linux的开放性，要想完整地罗列出所有的配置选项几乎是不可能的。不过， 1.0到1.1之间的大多数内核选项版本都提供了一份常见选项清单（加引号部分是批注）：

如果你想使用“ 任何 ”类型的连网设备（不管它是以太网，是 SLIP还是PPP）时，不管扩弧内显示什么样的宏名，都必须回答“ Y ”（是）。如果回答“ Y ”（是），就可以自动启用对以太网设备的支持。对其他类型网络驱动程序的支持则必须单独启用。

这些问题和Linux支持的不同链路层协议有关。 SLIP允许你通过串行线路传输 IP数据报。压缩报头（compressed header）选项提供了对CSLIP的支持，这种压缩技术将TCP/IP报头压缩为三个字节。注意，这个内核选项没有自动打开 CSLIP；它只是为CSLIP提供了必要的内核功能。

PPP是通过串行线路发送网络通信的另一种协议。它比 SLIP更为灵活，对IP也没有什么限制，同时还支持IPX。PLIP为通过并行连接发送IP数据报提供了一种解决办法。主要用于与运行DOS的计算机进行通信。

接下来的问题则和不同厂家的以太网卡有关。由于新的驱动程序层出不穷，这方面的问题肯定也是有增无减。如果想建立一个适用于不同类型机器的内核，可以采用多个驱动程序。

最后，是文件系统，配置脚本将问你是否想用 NFS（连网文件系统）。NFS会令你将文件系统导向若干台主机，使其类似于附在主机上的普通硬盘文件。

3.1.2 内核选项1.1.14及以上版本

注意 本小节无示例。

从1.1.14开始，由于增加了对IPX的Alpha支持（处于测试阶段），配置过程有了少许变化。常见选项这部分将问你是否需要常规连网支持。随后，是涉及到各种连网选项的两个问题。

要想采用TCP/IP连网，必须回答“ Y ”（是）。但回答“ N ”（否）的话，也能编译具有IPX支持的内核。

如果你的系统是两个以太网或一个以太网和一个 SLIP之间的网关，就必须启用这个选项。

尽管通过默认设置启用它没什么坏处，但你肯定想取消这一选项，把一台主机配置为一个所谓的防火墙。防火墙即是连接两个或两个以上网络，但不会在这些网络间路由通信的主机。它们常用于对来自公司网络的用户提供因特网访问，保护公司内部网络不受来自因特网的攻击和破坏。用户将得到许可登录到防火墙，使用因特网服务，但公司的机器不会因此而受到外界的攻击和破坏。因为任何接入的连接是不能通过防火墙的。

这个选项和PC/TCP的某些版本不兼容，后者是针对基于 DOS的计算机的一种商业 TCP/IP 实施方案。如果启用了这个选项，虽然仍然可以和普通计算机进行通信，但性能肯定会大受影响。

这个选项的作用是启用了 RARP（逆向地址解析）。无盘客户机和 X终端在启动时，利用 RARP来查询自己的IP地址。只有计划充当这类客户机时，才有启用 RARP的必要。网络公用程序的最新封装（net-0.32d）中，包含了一个小型的公用程序，其名为 rarp，它允许在RARP缓存内增添系统。

在通过TCP链路发送数据时，内核必须在把数据交给 IP协议之前，将它分为若干个包。对通过本地网络（比如以太网）就能抵达的主机来说，可采用较大的数据包（相对于必须通过长距离链路才能抵达的主机而言）。这样可避免小型数据包通过链路之后产生的碎片。如果不启用SNARL，内核将事实上只有一个接口的网络假定为本地网络。看看 Groucho Marx大学的网络B，整个网络都是本地的，但多数主机只连接了一个或两个子网。如果启用 SNARL，内核就会假定“所有”的子网都是本地的，在与校园内的其他所有主机通信时，都会发送大型数据包。

如果想对发往某些特殊主机（比如这种情况：数据将通过 SLIP链路）的数据采用小型数据包的格式，利用路由的mtu选项即可。关于mtu（最大传输单元）的详情，参见上一章。

对避免发送特别小的 IP包（也称作 tinygrams）来说， Negle算法是颇有启发性的。tinygrams（微型豆）通常由一些交互性的联网工具创建，这些工具只传输一个单独的键击，比如telnet和rsh。在诸如 SLIP之类的低带宽链路上，微型豆特别浪费带宽。某些情况下，Negle算法通过简单限制TCP数据传输的方式，试着避开这些微型豆。如果你碰上包丢失这一严重问题时，取消这一算法即可。

从1.1.16版本的内核开始， Linux对另一种驱动程序类型提供了支持，它就是伪驱动程序（dummy driver）。下面的问题将在设备驱动程序部分的开始处出现。

伪驱动程序的作用不大，但对单机或SLIP主机来说，它的用处就多了。它基本上是一个经过改头换面的回送接口（loopback）。出现这类接口的原因之一是在采用 SLIP，但没有以太网的主机时，人们希望有一个接口能一直保存自己的IP地址。有关详情，我们将在第5章讨论。

3.2 网络设备指南

内核针对不同类型的配置提供了大量的硬件设备驱动程序。本小节将简要介绍一些常见的驱动程序，以及用于这些驱动程序的接口名。

Linux中，有许多标准接口名（见下文）。大多数驱动程序支持的接口不止一个，所以这种情况下，就应该采用接口编号，比如 eth0、eth1等等。

■ lo 本地回送接口。和两个网络应用程序一起，供测试之用。它的运行类似于一个封闭式回路，任何写入这个接口的数据报都将立即返回本地主机的网络层。内核中始终都有一个

回送接口。

■ ethn n-th以太网接口。这是大多数以太网卡采用的一般性接口名。

■ dln 此类接口对D-Link DE-600包适配器（另一种以太网设备）进行访问。在衍生于并行端口的DE-600中，它是比较特殊的一种。

■ sln n-th SLIP接口。SLIP接口按照分配顺序，依次和串行线路对应起来。比如说，为SLIP配置的第一条串行线路就是sl0，第二条就是sl1，以此类推。内核能支持的SLIP接口多达四个。

■ pppn n-th PPP接口。和SLIP接口一样，PPP接口在串行线路转换为PPP模式之后，立即就和它对应起来。其时，内核能支持的接口多达四个。

■ plipn n-th PLIP接口。PLIP在并行线路上传递IP数据报。最多能支持三个PLIP接口。这些接口是在系统启动时，由PLIP驱动程序分配的，它们和并行端口一一对应。

对于将来可能增加的其他接口，比如 ISDN和AX.25，还可能引入其他接口名。用于 IPX（Novell连网协议）和AX.25 （供火腿无线电爱好者使用）的驱动程序尚处于开发阶段，但开始进入Alpha测试阶段了。

随后的几个小节中，我们将就上面提到的驱动程序展开讨论。

3.3 以太网安装

目前的网络程序对不同类型的以太网卡提供了广泛的支持。大多数驱动程序都是 Donald Becker（becker@cesdis.gsfc.nasa.gov）编写的，他为基于国家半导体 8390芯片的网卡编写了一系列驱动程序；即颇有名气的 Becker驱动程序系列。另外，还有两个产品是面向 D-Link的，其中的D-Link包适配器允许通过一个并行端口，访问以太网设备。针对这一用法的驱动程序是Bjørn Ekwall（bjOrn@blox.se）编写的。 DEPCA驱动程序则是 David C.Davies˘˘（davies@wanton.lkg.dec.com）编写的。

3.3.1 以太网接缆

如果你是生平第一次安装以太网卡，就有必要先了解一下布线方面的知识。以太网对布线是相当挑剔的。线缆的两段必须各有一个 50欧姆的电阻器，而且不能有任何分支（比方说，三条线缆组成一个星型连接）。如果利用一条带有 T型BNC连接头的细同轴线缆，就应该把这些连接头直接拧在网卡的连接器上，而不是插入一段线缆。

如果连接的是粗缆，就必须通过一个收发器（有时也称作以太网附单元）附上你的主机。可直接将收发器插入网卡上的 AUI端口，但也可采用一条屏蔽双绞线。

3.3.2 已获支持的网卡

下面将为大家列举一些广为人知的、已获 Linux支持的网卡。其完整列表在 HOWTO里面，其数目大约是这里列出的三倍之多。但是，即使你在这里的列表内看到了自己使用的网卡型号，但还是建议大家查看 HOWTO；HOWTO中列举的内容更为翔实、更为重要。需要注意的是，有些基于DMA的以太网卡使用的DMA通道和Adapter 1542 SCSI控制器默认状态下使用的通道一样。除非你亲自将其中之一移入另一个 DMA通道，否则的话，以太网卡就会把包数据写入你的硬盘专区内。

■ 3Com公司的EtherLink —— 获得支持的有3c503和3c503/16。3c507和3c509也如此。虽然Linux也支持3c501，但其速率太慢，不建议购买。

■ Novell公司的Eagle —— 获得支持的有NE1000和NE2000及其大量的克隆产品。NE1500和NE2100已获得支持。

■ Western Digital/SMC —— 获得支持的有WD8003和WD8013（SMC Elite和SMC Elite Plus）。另外，Linux还新增了对SMC Elite 16 Ultra的支持。

■ Hewlett Packard —— 获得支持的有HP 27252和HPJ27247B和HPJ2405A。另外还有D-Link DE-600包适配器DE-100、DE-200和DE-220-T。除此以外，还有一个用于 DE-650-T（一种PCMCIA卡）的补丁工具。

■ DEC —— 获得支持的有DE200(32k/64k) DE2O2、DE100和DEPCA rev E。

■ Allied Teliesis —— 已获Linux支持的有AT1500和AT1700。

要在Linux下使用上面列举的网卡，必须采用以上产品的主要分销商提供的一个预编译内核。这些产品一般含有内置驱动程序。但是，从长远的角度来看，最好用你自己的内核，编译真正能满足自己需要的驱动程序。

3.3.3 以太网自动侦测

系统启动时，以太网程序将试着找到网卡的位置，并判断它的类型。自动侦测代码存在两大局限。其一是，不能对所有的网卡进行准备识别。这不仅表现在一些便宜的仿造品上，还表现在WD80x3网卡上。其二是内核不能同时侦测多块网卡。因为它会假定你打算控制网卡和接口的分配问题。

如果你正在使用多块网卡，或自动侦测不能侦测你的网卡时，必须显式告诉内核该网卡的基础地址和设备名。

Net-3中，可通过两个不同的方案来完成上述任务。其一是改变或增加 drivers/net/Space.c文件的信息，该文件位于包含所有驱动程序信息的内核源代码内。不过，这一方案的前提是你对连网代码相当熟悉。第二种方案好的多，即在系统启动时，为内核提供驱动程序信息。如果用lilo来启动系统，通过lilo.conf内的append（添加）选项，指定一些参数之后，便可将这些参数传给内核。要将一个以太网设备的信息通知给内核，传递下面的参数即可：

```
ether=irq.base addr.param1.param2.name
```

前四个参数是数字化的，最后一个参数则是设备名。所有数字化的值都是可选的；如果都被省略或都设为零，内核就会试着通过侦测参数值的方式，找到这个值，或使用默认值。

第一个参数设置的是分配给这个设备的 IRQ。默认情况下，内核将试着自动侦测这个设备的IRQ通道。3c503驱动程序有一个非常特殊的特点，它从列出的 5、9、3、4中选出一个IRQ，并把网卡配置成使用这一行参数。

base_addr参数给出了网卡使用的I/O基础地址；零值表示内核将对上面列出的地址进行侦测。

至于其余两个参数，不同的驱动程序采用的方式是不一样的。对共享内存的网卡来说，比如WD80X3，它们指定了共享内存的起始点和终止点。其他网卡常用 param1来设置即将显示的调试信息级别。其值如果在 1到7之间，则表示冗余级不断上升，而如果是 8，就会把所有冗余都关掉；0表示默认设置。3c503驱动程序利用param2选定内部收发器（默认设置）或外部收发器（如果该值为1的话）。前者采用网卡的BNC连接器，后者采用它自己的AUI端口。

如果有两张以太网卡，可以自动侦测一张，将第二张卡的参数随 lilo一起传递出去。但是，必须保证驱动程序不会意外地先找到第二张卡，不然另一张卡根本没有注册的机会（也就是说根本不认你有两张网卡）。这是通过传递 lilo，一个保留选项来完成的，它显式告诉内核避免侦测第二张卡占用的I/O空间。

例如，要在作为eth1的0x300安装第二张以太网卡，需要向内核传递下列参数：

```
reserve=0x300,32 ether=0,0x300,eth1
```

reserve（保留）选项可保证内核在侦测某一网卡时，驱动程序不会去访问该网卡占用的I/O空间。另外，还可以用内核参数来改写 eth0的自动侦测：

```
reserve=0x340,32 ether=0,0x340,eth0
```

要关掉自动侦测，可将base_addr参数指定为-1：

```
ether=0,-1，eth0
```

3.4 PLIP驱动程序

PLIP代表并行线路IP，如果只需连接两台机器的话，这倒是最经济的连网方式。只须一个并行端口和一条特殊的线缆，便可获得 10Kbps到20Kbps的传输速率。

PLIP起初是Crynwr公司开发的。其设计思路相当高明（如果你愿意，也可把它称为hackish）：长期以来，个人电脑上采用的并行端口始终是单向的打印机端口，也就是说，要把数据从个人电脑传到某个外设，只能用 8条数据线路，除此以外，没有别的办法。 PLIP有效地解决了这一困扰，它将并行端口的 5条状态线路用作输入，并限定它们只能将所有数据当作半位元组来传输。这一操作模式就是所谓的零 PLIP模式。如今，这些单向并行端口很少有人用。因此，就有了PLIP扩展的产生，其名为模式1，它使用的是完整的8位接口。

目前， Linux只提供了对模式 0的支持。和早期的 PLIP代码不一样，现在它正试着与Crynwr的PLIP实施和NCSA Telnet中的PLIP驱动程序兼容（NCSA Telnet是一个DOS版本的常见程序，它在以太网或 PLIP线路上运行TCP/IP，而且还支持 Telnet和FTP）。要想利用PLIP连接两台机器，需要一种特殊的线缆，比如 Null Printer和Turbo Laplink线缆。但是，自己动手做一条这样的线缆并不费劲。详情参见第 19章。

参与PLIP驱动程序设计的人数不胜数。目前，它的维护主要由 Niibe Yutaka负责。如果把它编入内核，它就会为每个可能的打印机端口设置一个网络接口， plip0对应并行端口 lp0，plip1对应lp1，以此类推。

如果采用别的方式配置自己的打印机端口，必须对内核源代码的 drivers/net/Space.c内的这些值进行修改，并建立一个新的内核。

但是，这并不意味着你不能像往常那样使用这些并行端口。只有为其配置相应的接口，PLIP驱动程序才能对这些端口进行访问。

3.5 SLIP和PPP驱动程序

SLIP和PPP广泛用于在串行链路上发送 IP包。大量的机构为接入因特网的机器提供了拨号SLIP和PPP访问。因此，也为私人之间的 IP连接提供了可能（虽然有时可能得不偿失）。

要想运行SLIP或PPP，不必更换硬件设备；任何串行端口都可以用。由于串行端口的配置不只是针对TCP/IP连网的，所以我们单独为它设立了一章（即第4章）。有关详情可参考第4章。

第4章　串行硬件的设置

现在还有那么一些人，他们自己拥有一台计算机，不把钱花在 T1因特网链路上。日常新闻和电子邮件收发只依赖于采用公用电话网络的 SLIP链路、UUCP网络和电子公告板系统（BBS）。

本章旨在帮助那些依靠 modem接入因特网的人们维护他们的链路。但是，本章不准备就某些问题进行详细讨论（比如，配置 modem以供拨号之用）。所有这些主题的每一个细节都可在HOWTO中找到，它位于 www.redhat.com/mirrors/LDP/HOWTO/Serial-HOWTO.html（由 David S.Lawyer（bf347@lanf.org）编写。最初是由Greg Hankins（gregh@cc.gatech.edu）编写)。

4.1　Modem通信软件

用于Linux的通信软件不胜枚举。大部份都是终端软件，允许用户接驳另一台计算机。过去，用于Linux的终端软件叫作kermit。现在，有种称呼是 Spartan。随着IT业的不断进步，各种终端程序层出不穷，而且越来越方便好用，它们可支持脚本语言呼叫和登录到远程计算机系统等等。其中之一是 Minicom，该软件与早期DOS用户惯用的有些终端软件很相似。另外，还有基于X的通信软件，比如Seyon。与此同时，还有丰富的基于 Linux的BBS通信软件供那些想运行电子公告板的用户选择。大家可在 sunsite.unc.edu的/pub/Linux/system/Network内找到其中一些通信软件。

除了终端软件之外，还有一些软件利用一条专用的串行线路实现数据的非交互式传输。这种技术的好处是自动将邮件下载回本机，而不是在线浏览，从而节省时间。另一方面，它要求有较多的磁盘空间。

概括地说，这类通信软件就叫作 UUCP。它是一个通信套件，把一台主机上的文件复制到另一台主机，执行远程主机上的程序等等。它常用于私用网络内的邮件及新闻传输。 Ian Taylor编写的UUCP套件也可在Linux下运行，详情参见下一章。其他非交互式通信软件则用于Fidonet。诸如ifmail之类的Ficonet应用程序端口也是可以用的。

SLIP（串行线路Internet协议）介于交互式和非交互式传输之间。许多人用 SLIP拨号接入校园网或其他类型的公用 SLIP服务器，运行FTP会话等。但是， SLIP也可用于永久性地或半永久性的连接上，供局域网到局域网的联结所用，但只限于 ISDN。

4.2　串行设备概述

内核为访问串行设备所提供的设备一般称为 tty。其完整称呼是 teletype，它一度是早期Unix时代的主要终端产品。今天，这个术语代表的是任何一种基于字符的数据终端。本章中，我们用它来代表内核设备。

Linux将tty分为三类：（虚拟）控制台、伪终端（类似于一个双向管道，供 X11之类的应用程序使用）和串行设备。后者也被算做 tty，因为它们允许终端或远程计算机通过串行链路实

现交互式会话。

tty的配置参数较多，可利用 ioctl(2)系统方法调用来设定这些参数。多数参数都只适用于串行设备，因为它们需要更大的灵活性来处理不同类型的连接。

最突出的线路参数是 line speed（线路速率）和 parity（奇偶校验）。另外还有一些标记用于大小写转换、回车到链接速率的转换。tty驱动程序还支持不同的 line disciplines（线路规则，即线路协议），线路规则会令设备驱动程序的行为截然不同。例如， SLIP驱动程序就是通过一条特殊的线路规则来实施的。

应该怎样来描述线路速率呢？最恰当的术语是“ 位速率 ”（bit rate）。数据通信中，通过通信线路传输二进制位的速度。通常用每秒位数或比特／秒表示（ bps）。有时，也会听到有人称之为“ 波特率 ”（baud rate），其实这种称呼是很不恰当的。两个术语不能互换。波特率指的是某个串行设备的物理特性，即时钟频率，是每秒钟传送的信息位数量。而位速率，代表的是两个通信点之间的现有串行连接当前所处的状态，即每秒钟内传送的平均信息位数量。记住这两个值不同是很重要的，因为多数设备在每个电子脉冲处理的位数不止一位。

4.3 访问串行设备

像系统内所有的设备一样，对串行端口的访问是通过特定的设备文件（位于 /dev目录下）来完成的。有两类与串行驱动程序相关的设备文件，每个端口都有自己的设备文件。采用的文件不同，设备的行为也会有所不同。

第一类用于端口拨号时；它有一个主要的编号4，其文件名分别为ttyS0、ttyS1等等。第二类用于通过端口拨出时；其文件名为cua0、cua1等等，其主要编号是5（Linux设备有一个主编号和副编号。在做一个很长的目录清单（ls-1）时，也需列出设备编号。我们将在清单 4-1中向大家展示一个示例。主编号是5，副编号在64到67之间）。

两种类型的副编号完全相同。如果你的 modem在端口COM1到COM4之间的其中一个端口上，其副编号就会是 COM端口号再加上63。如果你的设置与此不符，比如使用的是一张支持多串行线路的网卡，就有必要查看 Serial HOWTO来了解详情。

现在，我们假设你的 modem位于COM2端口上。因此，其副编号是 65，主编号是5，用于拨出。另外还应该有一个拥有这些编号的设备 cua1。然后，列出/dev目录中所有的tty。第5和第6列将分别展示主编号和副编号，如清单 4-1所示。

清单4-1 设备cua1中的主编号和副编号

```
$ ls -l /dev/cua*
crw-rw-rw-   1 root     root       5,  64 Nov 30 19:31 /dev/cua0
crw-rw-rw-   1 root     root       5,  65 Nov 30 22:08 /dev/cua1
crw-rw-rw-   1 root     root       5,  66 Oct 28 11:56 /dev/cua2
crw-rw-rw-   1 root     root       5,  67 Mar 19  1992 /dev/cua3
```

如果没有此类的设备，必须建立一个： become super-user and type

```
# mknod -m 666 /dev/cua1 c 5 65
# chown root.root /dev/cua1
```

有人建议令/dev/modem作为一个象征性的链接，链接到自己的 modem设备，以便临时性的用户无须去记住颇费脑筋的 cua1。但是，没有人愿意在这个程序使用 modem，又在另一个程序内去取真正的设备文件名。正因为此，这类程序通常采用一个所谓的“ 锁文件 ”（lock

files）来表示设备正在使用中。按照惯例， cua1的锁文件名一般是LCK..cua1。针对同一个端口，使用不同的设备文件将意味着程序不能识别彼此的锁文件，因而两者都会同时采用这个设备。其结果是两个应用程序根本不能运行。

4.4 串行硬件

目前， Linux对采用RS-232标准的串行卡提供了广泛的支持。 RS-232是当前PC领域内最常用的串行通信标准。它利用大量回路来同步传送单一的信息位。新增的线路将用于标示载波（供modem所用）和握手的存在。

尽管硬件握手是可选的，但它非常有用。它允许通信的任何一方标示是否已准备接收数据，或另一方是否应该在接收方处理完接收到的数据之后，再继续发送数据。用于这些用途的线路分别称作“清除发送”（Clear to Send，CTS）和“准备发送”（Ready to Send，RTS），它们代表硬件握手名，即RTS/CTS。

在PC领域内， RS-232接口通常是一颗UART芯片（源于国家半导体16450芯片）或新版本的NSC 16550A（以前，有过NSC 16550芯片，但其FIFO一直未能起过任何作用）。其他牌子（全球最引人注意的是装备Rockwell芯片组的modem）也采用了完全不同的另类芯片，可以模拟16550芯片。

16450和16550芯片之间的主要区别是后者有 16个字节的FIFO缓冲，而前者只有1字节的缓冲。这样一来， 16450适用于9600波特以下的数据传送速率，而高速率则要求 16550芯片。除了前面提到的芯片外， Linux还支持8250芯片，该芯片是最初的UART（通用异步收发机），用于PC-AT。

默认配置方案中，内核会对COM1到COM4之间的四个标准串行端口进行查看。就像前面描述的那样，这些端口将分配到64到67之间的设备副编号。

如果打算正确配置自己的串行端口，你应该随rc.serial脚本一起，安装Ted Tso的setserial命令。应该在系统启动时间，从/etc/rc调用这个脚本。典型的rc.serial脚本像下面这样：

```
# /etc/rc.serial - serial line configuration script.
#
# Do wild interrupt detection
/sbin/setserial -W /dev/cua*
# Configure serial devices
/sbin/setserial /dev/cua0 auto irq skip test autoconfig
/sbin/setserial /dev/cua1 auto irq skip test autoconfig
/sbin/setserial /dev/cua2 auto irq skip test autoconfig
/sbin/setserial /dev/cua3 auto irq skip test autoconfig

# Display serial device configuration
/sbin/setserial -bg /dev/cua*
```

关于setserial命令的参数说明，请参考相应文档。

如果内核没有侦测到你的串行卡，或setserial-bg命令显示出一个错误的设置，你必须清楚提供正确的参数值，实施整个配置。对使用装备 Rockwell芯片组的内置modem的用户，经常会碰到这个问题。比如，将 UART芯片认作是NSC 16450芯片，而事实上，它是兼容于 NSC 16550的芯片，面对这种情况，你只有将配置命令改为

```
/sbin/setserial  /dev/cua1  auto irq skip test autoconfig
uart 16550
```

类似的情况也发生在COM端口、基础地址和IRQ设置上。详情参阅setserial(8)手册。

如果你的modem支持硬件握手，就应该确定启用这一特性。奇怪的是，默认情况下，多数通信软件都不会启用它；所以你必须自行手动设定。这一过程最好利用 stty命令，在rc.serial脚本内进行：

```
$ stty crtscts < /dev/cua1
```

要想查看硬件握手是否开始起作用，则采用

```
$ stty -a < dev/cua1
```

这样，就能看到该设备的所谓状态标记了；前面带有负号－的标记（比如－ crtscts），表示该标记已关闭。

第5章 TCP/IP网络配置

在本章中，我们将引导大家在自己的机器上实施设置 TCP/IP联网的全部必要过程。首先从IP地址的分配入手，再谈谈 TCP/IP网络接口的配置，随后针对网络安装过程中容易出现的几个问题，为大家介绍几个方便易用的工具。

本章涵括的任务大多只须执行一次。然后，只有在将新系统添加到自己的网络中时，或重新全盘配置你的整个系统时，才有必要接触到大量的配置文件。但是用于配置 TCP/IP的命令中，有些是每次启动系统时，都必须执行的。通常从 system/etc/rc脚本中调用这些命令。

注意 系统初始化脚本有两大主要派别：BSD风格和SysV init风格。Res Hat Linux采用的是修正过的SysV风格的初始化进程。这里的rc脚本和命名约定引用的是BSD风格的初始化进程。

通常，初始化进程的网络专有部分包含在一个名为 rc.net或rc.inet的脚本中。有时，还可能看见两个脚本，其名分别为 rc.inet1和rc.inet2，前者初始化联网的内核部分，后者开始基本的联网服务和应用。下面的示例中，我们采用的是后一个脚本。

rc.inet1脚本执行的动作和应用将留在后续章节中讨论。本章结束之时，大家应该建立起一序列的命令，这些命令完全能够完成 TCP/IP联网的配置工作。然后，再用 rc.inet1中的任何一个示例命令来替换它们，以保证 rc.inet1是在系统启动时执行的，并重新启动你的机器。联网用的rc脚本会给你一个较好的示例。

5.1 proc文件系统的设置

Net-2发布的版本中，有些配置工具依赖于 proc文件系统和内核进行通信（在内核 2.2.x版本中，联网代码被说成是 Net4）。这是一个接口，它允许通过一个类似文件系统的机制，访问内核运行时信息。装入时，可像对待其他所有文件系统一样，列出其所有文件，或显示其中的所有内容。典型的项目有 loadavg文件（其中包含系统载入平均数）或 meminfo（显示当前的核心内存和交换区的用法）。

所以，联网代码增加网络目录。其中包含大量的文件，分别显示内核 ARP表、TCP连接状态和路由信息表等等。许多网络管理工具都是从这类文件中获取相关信息的。

proc文件系统（或procfs）通常在系统启动时被装入 /proc。最好的办法是在 /etc/fstab中增加下面的代码：

```
# procfs mont point:
none            /proc       proc        defaults
```

然后，再从/etc/rc脚本中执行mount/proc。

如今，procfs通常采用默认设置进入内核。如果 procfs不在你的内核中，你就会得到这样一条消息：“ mount:fs type procfs not supported by kernel ”（内核不支持fs类型的procfs）。之后，

必须重新编译内核，并在要求回答是否需要 procfs支持时，回答“ Yes ”（是）。

5.2 二进制文件的安装

如果你采用的是预封装的工具，其中多半包含有主要的连网应用程序和实用程序以及一个混然一体的示例文件集。必须获得并安装这些新实用程序的唯一的条件是你安装了一个最新版本的内核。由于内核连网层时时可能发生变化，所以有必要经常性地更新自己的基本配置工具。至少须重新编译内核，但有时，还会要求你必须有最新的二进制文件集。这些文件一般随内核一起发放，封装在一个名为 net-XXX.tar.gz的归档文件中， XXX代表的是版本号。和-1.0对应的发布是0.32b，本书完稿之时，最新版本的内核（ 1.1.12）需要的是0.32d。

如果你想自行编译和安装标准的 TCP/IP网络，可以从大多数 FTP服务器那里获得源代码。这些源代码都是在 Net-BSD或其他源代码基础上，经过许多人的修订而成的。其他的应用程序，比如Xmosaic、Xarchie、 Gopher和IRC客户机，则必须单独从特定的地点获得。如按照指导去做，大多数程序都是行得通的。

5.3 另一个例子

从现在开始，我们将讨论一个更为实际的例子，它比 Groucho Marx大学校园网稍微简单一些。它就是Virtual Brewery。这是一家酿造啤酒的小公司。为了更有效地对自己的业务进行管理，老板想把各台机器连接起来，所有的机器都是新的。

同一层楼上，穿过大厅，就是Virtual Winery，这家公司的性质和Brewery差不多。他们各自运行一个以太网。很自然，只要可行的话，两家公司都想链接到对方的网络上。作为第一步，先在两个子网中间设置一个转发数据报的网关主机。然后，设法通过 UUCP链接到公司外面，以便收发新闻和电子邮件。以后，肯定是设置 SLIP链接接入因特网。

5.4 设置主机名

多数情况下（非绝对的），网络应用程序依赖于本地主机名（已被设为一个恰当的值）。这是在系统启动过程期间，利用 hostname命令来完成的。要设置主机名，就应该像下面这样调用它

```
# hostname name
```

通常用没有域名的不合格主机名来进行试验。比如， Virtual Brewery公司的主机可以是vale.vbrew.com、vlager.vbrew.com等。它们是这些主机的正式主机名，即完整资格域名。其本地主机名只能是公司名的前几个字母，比如 vale。但是，由于本地主机名常用于查找主机的IP地址，所以必须确保解析器库能够查找主机的 IP地址。这通常意味着必须在 /etc/hosts（参见下文）内输入主机名。

有人建议利用 domain name（域名）命令，将内核的域名概念设置为 FQDN的其他部分。通过这一方式，可以把主机名和域名的输出组合起来，获得一个 FQDN。但是，这也不能保证百分之百的正确。域名一般用于设置主机的 NIS（网络信息系统）域，这个域完全不同于你的主机所属的DNS。关于NIS的详情，请参见第9章。

5.5 分配IP地址

在自己的主机上配置联网软件，进行单机操作时（例如，能够运行 INN netnews软件），

完全可忽略本小节，因为你只需要回送接口的 IP地址，而这个接口的地址始终都是 127.0.0.1。

但面对真正的网络时（比如以太网），事情就没那么简单了。如果想将你的主机连接到一个现有网络，必须要求该网络的管理员为你分配一个 IP地址。在自行设置网络时，你必须像下面这样，给自己分配一个地址。

一个本地网络内的主机通常共享同一个逻辑 IP网络的地址。因此，你必须分配一个 IP网络地址。如果你手中有若干个物理网络，要么为它们分配不同的网络编号，要么利用子网技术，把自己的IP地址范围分成若干个子网。

如果你的网络没有接入因特网，便可以根据自己的喜好，自由选择网络地址，只要它是合法的。同时，必须保证从 A、B或C类地址中选择。但是，如果你打算不久将接入因特网，现在就应该想办法获得正式的 IP地址。然后，最好要求你的网络服务提供商为你提供相关帮助。如果想获得有朝一日用于因特网的网络编号，要向 hostmaster@internic.net请求一个“ 网络地址申请表 ”。

要想同时操作若干个以太网（或其他网络，只要驱动程序允许），就必须将自己的网络分成若干个子网。注意，只有你手中的广播网络不止一个时，才需要子网。点到点链接不能算作广播网络。例如，如果你有一个以太网和一个以上的 SLIP链接（接到外部世界），就不必将自己的网络分成若干个子网。

现在回到我们的示例上来。 brewery的网管采用的是NIC的B类网络编号，分配到的地址是191.72.0.0。为了能容纳两个以太网，她决定采用8位主机部分作为其增加的子网位。另8位用于主机部分，每个子网上允许接纳 254台主机。然后，她将子网编号 1分配给brewery，2分配给winery。因此，它们各自的网络地址分别是191.72.1.0和191.72.2.0。子网掩码是255.255.255.0。

作为两个网络间网关的 vlager，在两个网络上分到的主机编号都是 1，但它分到的IP地址分别是191.72.1.1和191.72.2.1。下面的示例展示了两个子网和网关。

```
#
# Hosts file for Virtual Brewery/Virtual Winery
#
# IP            local       fully qualified domain name
#
127.0.0.1       localhost
#
191.72.1.1      vlager      vlager.vbrew.com
191.72.1.1      vlager-if1
191.72.1.2      vstout      vstout.vbrew.com
191.72.1.3      vale        vale.vbrew.com
#
191.72.2.1      vlager-if2
191.72.2.2      vbeaujolais  vbeaujolais.vbrew.com
191.72.2.3      vbardolino   vbardolino.vbrew.com
191.72.2.4      vchianti     vchianti.vbrew.com
```

注意，这个示例中，采用的是B类网络，目的是为了简明扼要；其实， C类网络更为常见。有了新联网代码，子网的划分就不再受到字节边界的限制，所以C类网络也可分成若干个子网。例如，你可将两位的主机编号用于网络掩码，从而网络可分为四个子网，每个子网上可以有 64台主机(每个子网上的最后一个编号是为广播地址保留的，所以事实上每个子网只有63台主机)。

5.6 编写主机和网络文件

在将自己的网络分成若干个子网后，应该利用 /etc/hosts文件，进行简单的主机名解析了。如果不用DNS和NIS来进行地址解析，就必须将所有的主机信息放在一个 hosts文件(主机)内。

即使你想在普通操作期间运行 DNS或NIS，也应该先有/etc/hosts文件内的某些子网的所有主机名。比如在无网络接口运行的情况下 (引导时)，你想有某类主机名解析，那么，这样不仅很方便，还允许你采用 rc.inet脚本内的象征性的主机名。因此，在更改 IP地址时，只需将一份更新过的主机文件复制到所有的机器中，再重新启动即可，而不是重新逐个编辑数量庞大的rc文件。通常情况下，可把所有的本地主机名和地址放入主机文件内，如果需要，还可将网关和NIS服务器添加在内。(只有采用Peter Eriksson编写的NYS时，才需要NIS服务器的地址。其他NIS实施位于它们自己的服务器上，只能在运行时用 ypbind获得。)

与此同时，初始测试期间，你应该保证自己的解析器只采用主机文件内的信息。在使用自己的DNS和NIS软件时，可能会利用一些会产生怪异结果的示例文件。查询主机 IP地址时，如果要所有应用程序专用 /etc/hosts文件，必须对/etc/host.conf文件进行编辑。在以关键字 order开头的所有代码行前加上一个“ # ”号，就可以批注出所有的行，并插入下面这行

```
order hosts
```

注意 如何配置解析器库，请参见第6章。

主机文件中每行都有一个条目，由 IP地址、主机名和供选择的主机名别名清单组成。这些字段用空格或标号隔开，而且地址字段必须在第一列内。跟在“ # ”号后面的被视作批注，可忽略。

主机名要么已完全通过验证，要么关联在本地域内。以 vale为例，通常输入的是其完整资格名， vale.vbrew.com和主机文件内的 vale本身，所以它就有两个名字，一个是正式主机名，另一个是较短的本地名。

在我们上面的示例中， Virtual Brewery的主机文件像下面这样。其中包括两个特殊的主机名：vlager-if1和vlager-if2，它们为vlager网关上用的两个接口提供了地址。

正如主机的IP地址一样，有时人们肯定想用一个特殊好记的名字来代表某个网络。所以，主机文件还有一个如影随行的“ 搭档 ”—— /etc/networks，该文件指明网络名和网络编号之间的对应关系。

```
#
# Hosts file for Virtual Brewery/Virtual Winery
#
# IP            local       fully qualified domain name
#
127.0.0.1       localhost
#
191.72.1.1      vlager      vlager.vbrew.com
191.72.1.1      vlager-if1
191.72.1.2      vstout      vstout.vbrew.com
191.72.1.3      vale        vale.vbrew.com
#
191.72.2.1      vlager-if2
191.72.2.2      vbeaujolais  vbeaujolais.vbrew.com
```

```
191.72.2.3      vbardolino    vbardolino.vbrew.com
191.72.2.4      vchianti      vchianti.vbrew.com
```

注意 Res Hat Linux中没有/etc/networks文件。

我们在Virtual Brewery安装了一个networks（网络）文件，如下所示：

```
# /etc/networks for the Virtual Brewery
brew-net 191.72.1.0
wine-net 191.72.2.0
```

注意 网络文件中的网络名不得与主机文件内的主机名冲突，否则，会令某些程序产生怪异的结果。

5.7 IP接口配置

像前一章描述的那样设置好硬件之后，接下来的任务是让内核连网软件知道它们的存在。配置网络接口，并初始化路由表的命令有两条。配置任务通常是在系统启动时，从 rc.inet1脚本开始进行的。采用的基本工具叫作 ifconfig（if代表接口）和route，即接口配置和路由。

ifconfig用于令接口能够被内核联网层访问。这涉及到 IP地址和其他参数的分配，接口的激活（有时也称作启用）。这里的激活意思是内核将通过该接口收发 IP数据报。要想启用接口，最简单的方式是利用下面的代码调用它

```
ifconfig interface ip-address
```

上面的代码将ip-address分给接口，就激活了它。 其他所有参数都采用默认值。例如，默认的子网掩码衍生于 IP地址的网络类，比如代表 B类地址的255.255.0.0（ifconfig工具的详情，将在本章最后进行解释）。

第二个工具是route，它用于在内核路由表内增加或删除路由。可以这样调用它：

```
route [add|del] target
```

add和del这两个参数用于判断是增加还是删除路由。

5.7.1 回送接口

首先激活的接口是回送接口：

```
# ifconfig l0 127.0.0.1
```

有时，还可以看到本地主机用伪主机名代替了 IP地址的情况。 ifconfig将查找主机文件内的主机名，其中一个条目将该主机声明为 127.0.0.1的主机名：

```
# Sample /etc/hosts entry for localhost
localhost 127.0.0.1
```

如果要查看接口的配置情况，将接口名作为 ifconfig的参数，调用它即可：

```
$ ifconfig lo
lo        Link encap Local Loopback
inet addr 127.0.0.1  Bcast [NONE SET]  Mask 255.0.0.0
UP BROADCAST LOOPBACK RUNNING  MTU 2000  Metric 1
RX packets 0 errors 0 dropped 0 overrun 0
TX packets 0 errors 0 dropped 0 overrun 0
```

由此可知，回送接口已经分到一个网络掩码 —— 255.0.0.0，这是因为127.0.0.1是A类地址。这个接口也没有设置广播地址，因为对回送接口而言，一般没有多大用处。但是，如果你在

自己的主机上运行rwhod程序，就必须为回送设备设置广播地址，只有这样，才能保证 rwho正常运行。关于广播地址的设置，如下所示。

现在，你几乎可以运行你的网络了，但还缺一项，即路由表中的一条，它告诉 IP可以使用这个接口作为去往目标127.0.0.1的路由，可通过键入下行命令完成：

```
# route add 127.0.0.1
```

再次提醒大家注意，可以用本地主机名来代替 IP地址。

接下来，应该利用ping对所有的配置进行校验。ping相当于一种连网用的声纳设备，用于验证所给地址是否能够抵达，向它发送一个数据报后，稍隔一会，就会返回。一个 ping所花的时间一般称为一个周期（round-trip）。

```
# ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp seq=0 ttl=32 time=1 ms
64 bytes from 127.0.0.1: icmp seq=1 ttl=32 time=0 ms
64 bytes from 127.0.0.1: icmp seq=2 ttl=32 time=0 ms
^C

— localhost ping statistics —
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0/0/1 ms
```

像上面这样调用 ping时，它会永远不停地发送数据包，除非用户将其中断。上面的 ^C标出了我们按Ctrl+C中断校验的地方。

上面的示例展示了发到127.0.0.1的数据包传递无误，返回给ping的应答几乎是同步的。这表明你已成功设置了第一个网络接口。

如果ping的输出和上面的示例有出入，你就麻烦了。查看有些文件的安装是否有误。再看看你使用的ifconfig和route二进制文件是否与自己运行的内核兼容，最重要的是看内核与已启用的连网代码是否兼容（可从/proc/net目录得知）。如果得到的错误消息是这样的：“网络不可抵达”，说明你可能把route命令搞错了。一定要保证该命令中采用的地址和 ifconfig中的地址是一样的。

通过上面的步骤，完全足以在一台独立主机上使用连网应用程序了。在 rc.inet1增加上面的代码行，并保证从/etc.rc执行这两个rc.inet脚本之后，可能需要重新启动机器，试着运行各种应用程序。例如，“telnet localhost” 将建立一条通向你的主机的telnet链接，并给出一个登录提示。

然而，回送接口不仅用作网络丛书的示例，还用作开发过程中的测试床，但最常见的仍然是供有些应用程序用于普通操作。例如，基于 RPC的所有应用程序都利用回送接口，随portmapper程序一起在启动时注册自己。因此，这个接口总应该配置好，不管你的机器是否接入网络。

5.7.2 以太网接口

以太网接口的配置和回送接口的配置大致相同，唯一例外的是使用子网技术时，它要求的参数比后者要多一些。

在Virtual Brewery示例中，我们已经把起初是 B类网络的IP网络分成了若干个 C类子网。

要使接口能够对此进行识别，需像下面这样调用 ifconfig：

```
# ifconfig eth0 vstout netmask 255.255.255.255.0
```

这样，便为eth0接口分配了一个 vstout IP地址（191.72.1.2）。如果我们省略了网络掩码，ifconfig就会根据IP网络的类别推断出网络掩码是什么，而且这个网络掩码将是 255.255.0.0。现在，快速查看展示了下面的内容：

```
# ifconfig eth0
eth0      Link encap 10Mps Ethernet HWaddr  00:00:C0:90:B3:42
inet addr 191.72.1.2 Bcast 191.72.1.255 Mask 255.255.255.0
UP BROADCAST RUNNING  MTU 1500  Metric 1
RX packets 0 errors 0 dropped 0 overrun 0
TX packets 0 errors 0 dropped 0 overrun 0
```

由上可知，ifconfig自动将广播地址（上面的Bcast字段）设为普通值，即设置了所有主机位的主机网络编号。同时，消息传输单元（内核必须为该接口生成的以太帧最大字节数）必须设为最大，即1 500个字节。所有这些值都优先于后面所讲的特殊选项。

接下来的情形和回送接口类似，同样必须安装一个路由条目，向内核报告通过 eth0接口能够抵达的网络有哪些。就Virtual Brewery而言，应将route当作下面的代码行进行调用：

```
#route add -net 191.72.1.0
```

最初，这显得有点奇怪，因为真的不清楚 route如何检测出路由通过哪个接口。然而，这其实很简单：内核检查至今为止已配置好的所有接口，把目的地址 (此处为191.72.1.0)和接口地址的网络部分比较。唯一匹配的接口是 etho.

现在看看-net选项是干什么的。这被选用，因为 route即可以处理通向网络又可处理通向单个主机的路由。当给出点十进制地址后，它通过观察主机部分得出它是一个网络还是一个主机。如果地址的主机部分是 0，route假定它代表网络，否则它被视为主机地址。因此 route认为191.72.1.0是主机地址而非网络地址，因为它不知道我们使用了子网。所以我们必须使用 -net标记显式告诉它，此地址代表网络。

当然，上面的route命令敲起来很乏味，而且易于拼错。更方便的方法是使用我们在letc网络中定义的网络名。这使该命令更易读，此时-net标记可省略，因为route知道191.72.1.0代表网络。

```
# route add brew-net
```

现在，基本的配置步骤已告一段落，我们希望你的以太网接口能够正常运行。从你自己的以太网中选出一台主机，比如 vlager，并键入

```
# ping vlager
PING vlager: 64 byte packets
64 bytes from 191.72.1.1: icmp seq=0. time=11. ms
64 bytes from 191.72.1.1: icmp seq=1. time=7. ms
64 bytes from 191.72.1.1: icmp seq=2. time=12. ms
64 bytes from 191.72.1.1: icmp seq=3. time=3. ms
^C

—vstout.vbrew.com PING Statistics—
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 3/8/12
```

如果没有看到类似的输出，很显然，有什么被中断了。如果包丢失率异乎寻常的高，通常意味着硬件方面出问题了，比如终端坏了，或根本没有终端等等。如果根本不能接收数据包，就应该利用 netstat检查一些接口配置。 ifconfig显示出来的包特性将告诉你是否在某个接

口上发送了数据包。如果还访问了远程主机，也应该对这台远程主机及其接口特性进行检查。通过这种方式，便可准确地判断数据包是从哪里开始丢失的。另外，还应该利用 route显示路由信息，以了解两台主机的路由条目是否正确。如果在调用 route时，不带任何参数，就会在屏幕上输出路由信息（-n选项只能令其以点分四段式打印地址，而不是主机名）：

```
# route -n
Kernel routing table
Destination     Gateway         Genmask         Flags   Metric  Ref    Use
127.0.0.1       *               255.255.255.255 UH      1       0
127.72.1.0      *               255.255.255.0   U       1       0
```

这些字段的详细解释如下。 Flag列中包含一列针对每个接口的标记。 U始终是为活动接口设置的，H的意思是目标地址表示一台主机。如果 H标记是为准备用于网络路由的路由设置的，就必须采用带上-net选项的route命令。要验证你键入的路由真正能用，须检查两次调用 ping期间，倒数第二列中的Use字段是否有所增加。

5.7.3 通过网关的路由

前一小节，我们只介绍了在一个单独的以太网上设置主机的情况。但更为常见的是，通过若干个网关连接起来的网络群。这些网关虽然只链接了两个或两个以上的以太网，但也可以链接到外部网络，比如说因特网。要想使用网关提供的服务，必须为网络层提供给额外的路由信息。

例如， Virtual Brewery和Virtual Winery各自的以太网通过 vlager之类的网关链接在一起。假设vlager已经配置就绪，我们只须在 vstout的路由表内增加另一个条目即可，这个路由表将告诉内核它能够通过 vlager，抵达Winery网络上的所有主机。相应的 route调用如下所示； gw关键字告诉内核，下一个参数表示一个网关：

```
# route add wine-net gw vlager
```

当然，对于你希望与之通信的 Winery网络上的任何一台主机，都必须有相应的用于Brewery网络的路由条目，否则，就只能将数据从 vstout发送到vbardolino，但后者返回的所有应答都将进入一个很大的位桶（bit bucket）。

这个示例只说明了两个独立以太网之间，用于交换数据包的一个网关。现在，我们假设vlager还接入了因特网（比方说通过一条新增的 SLIP链接接入）。如果我们希望数据报能发送到非Brewery的任何一个网络，就应该把数据报交给 vlager。这是通过令其作为vstout的默认网关来完成的：

```
# route add default gw vlager
```

网络名的默认设置是0.0.0.0的速写，表示默认路由。不必将这个网络名加入 /etc/networks文件中，因为它是内置入默认路由中的。

如果在一个或多个网络之后，对一台主机进行 ping测试时，发现包丢失率很高，这通常意味着网络阻塞得相当厉害。包的丢失归咎于技术上的落后，或临时性的网络超载，因此主机产生延迟，甚至丢弃进入的数据报。

5.7.4 网关的配置

要配置一台机器，使其负责两个以太网间的包交换，是非常容易的一件事。假设以 vlager为例，它配有两张以太网卡，每张卡都连接到其中一个网络上。你只须为它们分配 IP地址，并

对这两个接口进行配置即可。

注意 Linux-2.0.x系列中，默认情况下的包转发是取消了的。所以可在运行时启用包的转发：# echo 1 > /proc/sys/net/ipv4/ip_forward。

像下面这样，在主机文件内增加关于这两个接口的信息是相当有用的，关于这两个接口，我们还有更容易记的名字：

```
191.72.1.1    vlager    vlager.vbrew.com
191.72.1.1    vlager-if1
191.72.2.1    vlager-if2
```

下面的命令用于设置这两个接口：

```
# ifconfig eth0 vlager-if1
# ifconfig eth1 vlager-if2
# route add brew-net
# route add wine-net
```

5.7.5 PLIP接口

与利用以太网连接相比，利用 PLIP链接连接两台计算机的情况稍有不同。后者是所谓的点到点链接，因为相对广播网络而言，它只涉及到两台主机（即节点）。

我们以Virtual Brewery公司的一名员工手中的膝上型计算机为例，该公司的网络通过 PLIP与vlager连接。这台膝上型计算机本身名为 vlite，而且只有一个并行端口。在启动时，这个端口被注册为plip1。要激活这条链接，必须利用下面的命令，配置 plip1接口：

```
# ifconfig plip1 vlite pointtopoint vlager
# route add default gw vlager
```

注意 pointtopoint不是排版错误，它真的就是这样拼写的。

第一个命令行告诉内核这是一条点到点链接，其远程端的地址是 vlager，配置了plip1接口。第二个命令行将 vlager作为网关，安装了默认路由。在 vlager上，还有必要利用一个类似的ifconfig命令激活这条链接（不必调用 route）：

```
# ifconfig plip1 vlager pointtopoint vlite
```

有趣的是：vlager上的plip1接口居然不必有其独立的IP地址，而是给定的191.72.1.1这个地址。但是，应该注意，只有在完全配置好以太网路由表之后，才能配置 PLIP或SLIP链接。不然的话，在使用有些老版本的内核时，会在配置点到点链接时中断。

膝上型计算机的路由配置之后，转向 Brewery网络的路由配置；还没有为 Brewery网络上的任何一台主机配置到vlite的路由呢！特别麻烦的一种方法是在每台主机的路由表上增加特定的路由，这些路由表将vlager指定为通向vlite的网关：

```
# route add vlite gw vlager
```

因此，在面对临时性的路由时，最好采用动态路由信息。其一是利用 gated（一种路由程序），首先在网络中的各台主机上安装这个程序，以便动态分发路由信息。但是，最简单的方法是利用代理ARP。有了代理ARP，只要一有查询vlite的ARP，vlager就会发送自己的以太网

地址，作出相应的应答。这样的后果是所有发向 vlite的包都会流入vlager，再由vlager将它们转发到膝上型计算机。我们将在随后的PLIP小节，为大家谈谈代理ARP。

将来的Net-3中，将包含一个名为plipconfig的工具，允许你设置打印机端口的IRQ。稍后，可能还有更为常用的ifconfig命令取而代之。

5.7.6 SLIP和PPP接口

尽管SLIP和PPP链接只是诸如PLIP之类的简单点到点链接，但其内容还是相当纷繁的。通常，SLIP连接涉及的步骤有：通过modem，拨打远程站点，将串行线路设置为SLIP模式等等。PPP的用法与此类似。设置SLIP或PPP链接需要哪些工具呢，详情参见第6和第7章。

5.7.7 伪接口

伪接口其实是个舶来品，但它的确有用。它给单机和其 IP网络连接是一条拨号链接的计算机带来诸多的好处。事实上，后者大多数时间也是单机。

单机的局限在于：它们只有一个单一的网络设备是活动的，也就是说仅有一个回送设备，这个设备的地址始终是 127.0.0.1。但是，我们偶尔需要向本地主机的正式 IP地址发送数据。以膝上型计算机vlite为例。这时，它已经取消了所有的网络连接。 vlite上的一个应用程序可能打算想向一台机器上的另一个应用程序发送某些数据。怎么办？在 vlite的/etc/hosts1文件内查找IP地址191.72.1.65，如此一来，应用程序便开始试着向这个地址发送数据。由于回送接口是当前本机上唯一的活动接口，所以内核无法判断这个地址引用的其实就是它本身！其结果是，内核将数据报丢弃，并向应用程序返回一个错误。

因此，伪设备应运而生。只须充当回送接口的替身，便解决可上面的问题。在 vlite这个示例中，只须给它一个 191.72.1.65这样的地址，并增加一条指向它的主机路由即可。每个目标为191.72.1.65的数据报都会得以在本地传送。其正确的调用方法是：

```
# ifconfig dummy vlite
# route add vlite
```

5.8 ifconfig详解

ifconfig的参数非常之多，远远超过了以前所讲过的那些。其普通调用结构是：

```
inconfig interface [[-net|-host] address [parameters]]
```

“ interface ”指接口名，“ address ”指准备分配给这个接口的IP地址。它即可以是点分四段式的IP地址，又可以是ifconfig在/etc/hosts和/etc/networks文件内找出的接口名。 -net和-host选项强制ifconfig将这个地址视作网络编号或主机地址。

如果ifconfig在调用时只有接口名，它就会显示出该接口的配置信息。如果调用 ifconfig时不带任何参数，它就会显示迄今为止已配置的所有接口； -a选项也将强制它显示所有非活动

```
# ifconfig eth0
eth0      Link encap 10Mbps Ethernet  HWaddr 00:00:C0:90:B3:42
          inet addr 191.72.1.2 Bcast 191.72.1.255 Mask 255.255.255.0
UP BROADCAST RUNNING  MTU 1500  Metric 0
RX packets 3136 errors 217 dropped 7 overrun 26
TX packets 1752 errors 25 dropped 0 overrun 0
```

的接口。以太网接口eth0的调用示例如下：

MTU（最大传输单元）和Metric（度量值）字段显示的是该接口当前的 MTU和度量值的值。按照惯例，度量值供某些操作系统所用，用于计算一条路由的成本。

RX和TX这两行展示了已经准备无误地收发了多少数据包、发生了多少错误、丢失了多少数据包（因内存少的原因），以及多少包是因为传输过载而遗失的。 ifconfig打印出的标本值或多或少与其命令行中的选项名相符；详情如下：

表5-1随括号中的标记名一起，列出了ifconfig能够识别的参数。选项对应的特性可以打开也可以取消，只在选项名前加一个破折号即可（ -）。

表5-1 ifconfig可以识别的参数

up	标志接口处于“ up ”状态，也就是说， IP层可以对其进行访问。这个选项用于命令行上给出一个地址之时。如果这个接口已被“ down ”选项临时性取消的话（与该选项对应的标记是UP RUNNING），还可以用于重新启用一个接口
down	标志接口处于“ down ”状态，也就是说， IP层不能对其进行访问。这个选项有效地禁止了IP通信流通这个接口。注意，它并没有自动删除利用该接口的所有路由信息。如果永久性地取消了一个接口，就应该删除这些路由条目，并在可能的情况下，提供备用路由
netmask	分配子网掩码，供接口所用。要么给一个前面是 0x的32位十六进制号码，要么采用只适用于两台主机所用的点分四段式号码。对 SLIP和PLIP接口来说，这个选项是必须配置的
pointtopoint address	该选项用于只涉及两台主机的点到点链接。对 SLIP和PLIP接口来说，这个选项是必须配置的（如果已经设置了一个点到点地址， ifconfig就会显示出POINTTOPOINT标记）
broadcast address	广播地址通常源于网络编号，通过设置主机部分的所有位得来。有的 IP采用的方案有所不同：这个选项可适用于某些奇怪的环境（如果已经设置了广播地址， inconfig就会显示出一个BROADCAST标记）
metric number	该选项可用于为接口创建的路由表分配度量值。路由信息协议（ RIP）利用度量值来构建网络路由表。ifconfig所用的默认度量值是0。如果不运行RIP程序，就没必要采用这个选项。如果要运行RIP程序，就尽量不要改变这个默认的度量值
mtu bytes	该选项用于设置最大传输单元，也就是接口一次能处理的最大字节数。对以太网接口来说， MTU的默认设置是1 500个字节；对SLIP接口来说，则是296个字节
arp	这个选项专用于以太网或包广播之类的广播网络。它启用 ARP（地址解析协议）来保护网络上各台主机的物理地址。对广播网来说，默认设置是“ on ”（开）
-arp	取消在接口上使用ARP
promisc	将接口置入promiscuous（混乱）模式。广播网中，这样将导致该接口接收所有的数据包，不管其目标是不是另一台主机。该选项允许利用包过滤器和所谓的以太网窥视技术，对网络通信进行分析。通常情况下，这对揪出网络故障的元凶来说，是相当有用的。但另一方面，如果有人蓄意攻击你的网络，也可浏览到通信数据，进而获得密码，破坏你的网络。一项重要的保证措施是杜绝任何人将他们的计算机接入你的以太网。另一个选项用于保护某些身份验证协议的安全，比如 Kerberos或SRA登录套件（该选项对应的标记是PROMISC）
-promisc	取消混乱模式
allmulti	多播地址即是向不在同一个子网上的一组主机广播数据。多播地址尚未获得内核支持（该选项对应的标记是ALLMULTI）
-allmulti	取消多播地址

5.9 netstat详解

下面，将为大家介绍一个特别有用的工具，用来检查网络配置和活动情况。它就是 netstat。

事实上，它是若干个工具的汇总

5.9.1 显示路由表

在随-r标记一起调用netstat时，将显示内核路由表，就像我们利用 route命令一样。vstout上产生的输出如下：

```
# netstat -nr
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use
127.0.0.1       *               255.255.255.255 UH    1      0
191.72.1.0      *               255.255.255.0   U     1      0
191.72.2.0      191.72.1.1      255.255.255.0   UGN   1      0
```

-n选项令netstat以点分四段式的形式输出IP地址，而不是象征性的主机名和网络名。如果想避免通过网络查找地址（比如避开DNS或NIS服务器），这一点是特别有用的。

netstat输出结果中，第二列展示的是路由条目所指的网关。如果没有使用网关，就会出现一个星号（*）。第三列展示路由的概述。在为具体的 IP地址找出最恰当的路由时，内核将查看路由表内的所有条目，在对找到的路由与目标路由比较之前，将对 IP地址和genmask进行按位“与”计算。

第四列显示了不同的标记，这些标记的说明如下：

- G 路由将采用网关。
- U 准备使用的接口处于“活动”状态。
- H 通过该路由，只能抵达一台主机。例如，回送接口的路由器条目 127.0.0.1就如此。
- D 如果路由表的条目是由ICMP重定向消息生成的，就会设置这个标记（详情参见第 3章）。
- M 如果路由表条目已被ICMP重定向消息修改，就会设置这个标记。

netstat输出结果的Ref列展示了对这条路由的引用数，也就是说，有多少其他的路由（例如通过网关的路由）是建立在这条路由基础上的。最后两列展示了路由条目的使用次数和用于投递数据报的接口。

5.9.2 显示接口特性

在随-i标记一起调用时， netstat将显示网络接口的当前配置特性。除此以外，如果调用时还带上-a选项，它还将输出内核中所有接口，并不只是当前配置的接口。 vstout上，netstat的输出结果是这样的：

```
$ netstat -i
Kernel Interface table
Iface       MTU     Met     RX-OK  RX-ERR  RX-DRP  RX-OVR  TX-OK   TX-ERR  TX-DRP  TX-
10          0       0       3185   0       0       0       3185    0       0
eth0         1500   0       972633 17      20      120     628711  217     0
```

MTU和Met字段表示的是接口的 MTU和度量值值。RX和TX这两列表示的是已经准确无误地收发了多少数据包（RX-OK/TX-OK）、产生了多少错误（RX-ERR/TX-ERR）、丢弃了多少包（RX-DRP/TX-DRP），由于误差而遗失了多少包（RX-OVR/TX-OVR）。

最后一列展示的是为这个接口设置的标记。在利用 ifconfig显示接口配置时，这些标记都

采用一个字母。它们的说明如下：

- B 已经设置了一个广播地址。
- L 该接口是一个回送设备。
- M 接收所有数据包（混乱模式）。
- N 避免跟踪。
- O 在该接口上，禁用ARP。
- P 这是一个点到点链接。
- R 接口正在运行。
- U 接口处于“活动”状态。

5.9.3 显示链接

netstat支持用于显示活动或被动套接字的选项集。选项 -t、-u、-w和-x分别表示TCP、UDP、RAW和UNIX套接字连接。如果你另外还提供了一个-a标记，还会显示出等待连接（也就是说处于监听模式）的套接字。这样就可以得到一份服务器清单，当前所有运行于系统中的所有服务器都会列入其中。

在vlager上调用netstat -ta时，输出结果如下：

```
$ netstat -ta
Active Internet connections
Proto Recv-Q Send-Q Local Address    Foreign Address    (State)
tcp        0      0 *:domain         *:*                LISTEN
tcp        0      0 *:time           *:*                LISTEN
tcp        0      0 *:smtp           *:*                LISTEN
tcp        0      0 vlager:smtp      vstout:1040        ESTABLISHED
tcp        0      0 *:telnet         *:*                LISTEN
tcp        0      0 localhost:1046   vbardolino:telnet  ESTABLISHED
tcp        0      0 *:chargen        *:*                LISTEN
tcp        0      0 *:daytime        *:*                LISTEN
tcp        0      0 *:discard        *:*                LISTEN
tcp        0      0 *:echo           *:*                LISTEN
tcp        0      0 *:shell          *:*                LISTEN
tcp        0      0 *:login          *:*                LISTEN
```

上面的输出表明大多数服务器都处于等待接入连接状态。但是，第四行表明其中一条进入的连接来自 vstout，第六行表示有一条通往 vbardolino的telnet连接。利用 -a选项的话，netstat还会显示出所有的套接字。

注意 根据端口号，可以判断出一条连接是否是外出连接。对呼叫方主机来说，列出的端口号应该一直是一个整数，而对众所周知服务（well known service）端口正在使用中的被呼叫方来说，netstat采用的则是取自/etc/services文件的象征性服务名。

5.10 检查ARP表格

某些时候，查看甚至改变内核 ARP表格是非常有用的，比如说，在你怀疑某个备份的Internet地址可能会引发网络问题时更是如此。arp工具就是你的得力助手。其命令行选项如下：

```
arp [-v] [-t hwtype] -a [hostname]
arp [-v] [-t hwtype] -s hostname hwaddr
arp [-v] -d hostname [hostname...]
```

所有主机名参数都可能是象征性的主机名或采用点分四段式的 IP地址。

第一行调用显示出 IP地址或指定主机的 ARP条目，如果没有指定主机，则显示出所有已知的主机名。例如，在vlager上调用arp可能会产生下面的结果：

```
# arp -a
IP address        HW type          HW address
191.72.1.3        10Mbps 以太网    00:00:C0:5A:42:C1
191.72.1.2        10Mbps 以太网    00:00:C0:90:B3:42
191.72.2.4        10Mbps 以太网    00:00:C0:04:69:AA
```

上面的输出显示出vlager、vstout和vale的以太网地址。

利用-t选项，可只显示指定的硬件类型。显示结果可以是： ether、ax25或pronet，它们分别代表10Mbps 以太网、AMPR-AX.25和IEEE-802.5令牌环设备。

-s选项用于将主机名的以太网地址永久性地加入 ARP表格中。hwaddr参数指定的是硬件地址，默认情况下，将是以太网地址，被指定为用冒号隔开的 6个十六进制字节。当然，也可以用-t选项，为其他类型的硬件设置硬件地址。

出于某种原因，在 ARP表格内查找远程主机失败时（比如该远程主机的驱动程序漏洞百出，或网络中的另一台主机采用了它的 IP地址，使其能够冒充这台远程主机），要求大家手工把IP地址加入ARP表格。ARP表格内的硬性绑定的 IP地址也是一种能够避免以太网中别的主机冒充你的（非常有效）手段。

利用-d调用arp时，将删除与给定主机相关的所有 ARP条目。它可以用于强制接口再次尝试获取未知的IP地址。配置有误的系统中出现错误的广播 ARP信息时，这是非常有用的（不过，首先得重新配置被中断的主机）。

还可用-s选项来实施代理ARP。这是一种比较特殊的技术；通过假称两个地址指的是同一台主机（即gate）这一方式，主机（即gate）同时还扮演网关通向另一台名为 fnord的主机。这是通过出版一条fnord ARP条目来完成的，这个条目指代的是该主机自己的以太网接口。现在，主机发出ARP查找fnord时，gate将返回一条应答，其中包含它自己的以太网地址。发出查找的主机再将所有数据报发给gate，最后再由gate负责将它们转发给fnord。

透彻理解上面的描述是非常必要的，因为这种情况屡有发生：比如你想从一台 DOS版本的机器（已中断了 TCP实施，不能很好地识别路由）访问 fnord。此时使用代理 ARP，它就可能出现在DOS机器上，就像fnord位于本地子网上一样，对它而言，没有必要知道通过网关进行路由。

关于代理ARP，另一个非常重要的应用表现在你的主机之一临时性充当通向其他主机的网关时，比如通过一条拨号链接时。在前一个示例中，我已经见识过膝上型计算机 vlite，它只是偶尔通过一条PLIP链接和vlager建立连接。当然，能使这一切行之有效的前提是你准备为其提供代理ARP的主机必须和你的网关位于同一个 IP子网上。例如， vstout可以为Brewery子网（191.72.1.0）上所有主机提供代理 ARP，但对Winery子网（191.72.2.0）上的任何一台主机来说，则是不可能的。

为fnord提供代理ARP的调用结构如下；当然，所给的以太网地址必须是 gate的地址。

```
# arp -s fnord 00:00:c0:al:42 e0 pub
```

再通过调用的方式，删除代理ARP条目：

```
# arp -d fnord
```

5.11 未来展望

连网技术不断在进步。内核层的主要变动给我们带来了更为灵活的配置方案，使我们能够在运行时配置网络设备。例如， ifconfig命令可以采用设置IRQ和DMA通道的参数。

另一个即将面世的主要变动是在route命令中增加了mtu标记，这样就可以为特定路由设置“最大传输单元”了。最大传输单元简称 MTU。路由专用的MTU优先于为接口指定的 MTU。人们一般都会将这个选项用于通过网关的路由，这种情况下，网关和目标主机之间的链接需要非常低的MTU值。打个比方，假设主机wanderer通过一条SLIP链接与vlager相连。在把数据从vstout发给wanderer时，wanderer上的网络层将采用1 500字节（最大值）的标准来打包数据，因为是通过以太网来发送数据。而另一方面， SLIP链接只能传送MTU值为296的数据包，所以vlager上的网络层不得不将IP包分成296个字节大小的数据段。如果不这样做，就要对 vstout上的路由进行重新配置，使其一开始就采用 296的MTU值，这样可避免花销太大的数据分段：

```
# route add wanderer gw vlager mtu 296
```

注意， mtu选项还允许你选择性地取消“子网属本地网络”策略（ SNARL）的影响。这个策略是一个内核配置选项，我们曾在第 3章讨论过。

5.12 名字服务和解析器配置

正如第2章描述的那样， TCP/IP连网可能依赖于某些不同的方案，将主机名转换为相应的地址。一种最简单的方法便是主机表，它没有采用类似于把域名空间分成若干个区的方式，这个主机表保存在 /etc/hosts文件内。但它只适用于小型的局域网，这些局域网由一个管理员负责。主机文件的格式参见第 3章。

另外，也可以用BIND即Berkeley Internet域名服务，将主机名解析为 IP地址。BIND的配置相当琐碎，但一旦配置好了，要在网络拓扑中进行更改就变得非常简单。在许多 Linux系统上，域名服务是通过一个名为 named的程序来提供的。该程序在启动时，将 master（主要）文件载入自己的缓存内，等待远程或本地用户进程的查询。设置 BIND的方法很多，而且都不要求你在各台主机上运行域名服务器。

本章将简要介绍一下如何操控域名服务器。如果你手里的网络不仅仅是一个小型局域网，而且还有可能接入因特网，那么，要想在这种环境下使用 BIND，最好找一本关于 BIND的书来作参考。推荐大家使用 Cricket Liu所著的《DNS和BIND》，该书由O’Reilly出版，现在已有了第三版，可访问 www.oreilly.com/catalog/dns3/获得你需要的资料。关于这方面的最新信息，还可查看一下 BIND源代码中包含的注意事项。另外，还有两大新闻组供大家参考：comp.protocols.dns和comp.protocols.dns.bind。

5.13 解析器库

“解析器”并不是一种特殊的应用程序，而是解析器库的代名词，它是一个可在标准 C语言库内能够找到的函数集合。核心例程分别是： gethostname(2)和gethostbyaddr(2)，这两个例

程用于查找主机对应的所有IP地址，反之亦然。它们还可以配置为简单地查询主机内的信息，查询域名服务器或使用NIS（网络信息服务）的hosts（主机）数据库。其他应用程序，比如smail，可能包含不同的驱动程序，需要特别注意。

5.13.1 host.conf文件

控制解析器设置的核心文件是host.conf。它驻留在/etc中，并告诉解析器使用哪些服务，使用顺序又如何。

host.conf文件内的选项必须以单行的形式出现。各个字段必须用空格隔开。“#”号引入的是对下一行进行扩展的批注。

下列选项是可用的：

■ order —— 该选项决定解析服务的使用顺序。有效选项有：

- bind —— 用于查询域名服务器。
- hosts —— 用于/etc/hosts内的查找。
- nis —— 用于NIS查找。

可以同时指定一个选项或所有的选项。各选项在命令行出现的顺序将决定各自的服务顺序。

■ multi —— 其选项有on和off。它用于判断/etc/hosts内的主机是否获得允许使用多个IP地址，如果可以，这个主机就被称为“多宿主主机”。这个标记对DNS和NIS查询是没有用的。

■ nospoof —— 就像前一章解释的那样，DNS允许你利用inaddr.arpa域，查找属于某一台主机的主机名。域名服务器提供假主机名的作法称为“电子欺骗”。为防止这种情况的发生，可将解析器配置为查看原始IP地址是否真的与获得的主机名相符。如果不相符，这个主机名就会被丢弃并返回一个错误。将“电子欺骗”设为“on”，就可以启用这个行为。

■ alert —— 该选项将on和off作为其参数。如果设置为“on”，任何电子欺骗尝试（如上所述）将令解析器将消息记录到系统日志中。

■ trim —— 该选项将域名作为其参数，在实施主机名查找之前，这里的域名会从主机名文件中被删除。这个特性对主机条目来说，是非常有用的，因为人们只想指定不带域名的主机名。对绑定本地域名的主机所进行的查找将删除这个特性，因此，允许/etc/hosts文件内的查找继续下去。trim选项累积起来，使其很容易将你的主机视为处于本地域内。

vlager所用的文件范例如下所示：

```
# /etc/host.conf
# We have named running, but no NIS (yet)
order    bind hosts
# Allow multiple addrs
multi    on
# Guard against spoof attempts
nospoof on
# Trim local domain (not really necessary).
trim     vbrew.com.
```

5.13.2 解析器环境变量

利用大量的环境变量，可以改写host.conf文件中的设置。

它们是：

■ RESOLV HOST CONF —— 该变量指定准备读取的文件，并用该文件来代替/etc/host.conf。

■ RESOLV SERV ORDER —— 改写host.conf文件内指定的order（顺序）选项。指定的服务分别为hosts、bind和nis，并分别用空格、逗号、冒号和分号隔开。

■ RESOLV SPOOF CHECK —— 判断针对电子欺骗的尺度。如果设置为 off，则表示彻底取消。warn和warn off这两个值都会启用电子欺骗校验，但将分别打开和关闭记录。“ * ”值将打开电子欺骗校验，但会像host.conf文件内定义的那样，保留记录设备。

■ RESOLV MULTI —— 可选项有on和off，用于改写tt host.conf文件内的multi选项。

■ RESOLV OVERRIDE TRIM DOMAINS —— 该环境变量指定一份trim域清单，它们将用于改写host.conf内的那些域。

■ RESOLV ADD TRIM DOMAINS —— 该环境变量指定一份trim域清单，这些域将增加到host.config中指定的域内。

5.13.3 域名服务器查找 —— resolv.con的配置

在配置解析器库，使之利用 BIND域名服务进行主机查找时，还必须告诉它准备使用哪些域名服务器。对此，有一个专门的文件，名为 resolv.conf。如果该文件不存在，或者为空，解析器就会假定域名服务器在你的本地主机上。

如果在本地主机上运行一个域名服务器，就必须单独安装它，具体做法参见下一小节。如果你处于一个本地网络上，而且有机会采用一个现成的域名服务器，用这个现成的最好。

resolv.conf文件中，最重要的选项是nameserver（域名服务器），它将给出准备使用的域名服务器的IP地址。如果你通过nameserver选项，多次指定了若干个域名服务器，查找就会按照指定时的顺序来进行。因此，应该将最稳定可靠的服务器放在最前面。当前，最多可指定 3个服务器。

如果不采用nameserver选项，解析器就会试着链接到本地主机上的域名服务器。

另外两个选项：domain和search，用于处理绑定在主机名上的默认域（前提是 BIND在第一条查询中未能解析出这个主机名。 search选项指定了一份准备查找的域名清单。清单中的项目用空格隔开。

如果不采用search选项，解析器就会利用域名本身，再加上上推至根的所有父域名，根据本地域名构建一份默认查找清单。本地域名的赋予是通过 domain（域）语句来进行的；如果一个选项也不指定，解析器就会通过 getdomainname(2)系统调用，将域名包括在内。

如果你对此仍然不很清楚，可仔细想想 Virtual Brewery所用的resolv.conf文件，如下所示：

```
# /etc/resolv.conf
# Our domain
domain          vbrew.com
#
# We use vlager as central nameserver:
nameserver      191.72.1.1
```

在解析主机名 vale时，解析器将查找 vale，如果失败，就会查找 vale.vbrew.com和vale.com。

5.13.4 解析器的健壮性

如果在一个大型网内运行一个局域网，在可能的情况下，人们肯定会采用中央域名服务器。其好处在于能够开发出这些服务器的巨大潜力，因为所有的查询都将转发到这些中央服务器来。但是，这一方案有个明显的不足：校园网内的主干线发生突发性的灾难事件时，各系的局域网也将无法正常运行，因为解析器不能抵达任何一台中央域名服务器。也无法从任何一台X终端登录，不能打印，什么也不能做。

尽管此类事件并不是频频发生，但最好能做到防患于未然。

一种办法是设置一个本机域名服务器，解析你所处的本地域内的主机名，并将所有对其他主机名的查询转发给主服务器。当然，前提是你在自己的域内运行。

另一种办法是：把自己的域或局域网备份主机表保存在 /etc/hosts文件内。然后，再将“ order bind hosts ”主机包括在/etc/hosts.conf内，在中央域名服务器关闭时，令解析器退回主机文件，继续查找。

5.14 named的运行

在大多数计算机上提供域名服务器的程序通常被命名为 named。这个服务器程序起初是为BSD开发的，用于为客户机提供域名服务，还可用于其他的域名服务器。当前，大多数人安装的版本是BIND 8.2（8.X是源于4.X系列的下一个主版本；目前4.x的版本号是4.9.7）。

注意 BIND的维护工作由“ 因特网软件协会 ”（Internet Software Consortium，其网址是www.isc.org）负责。大家可在www.isc.org/bind.html上找到需要的BIND资料。

本小节要求大家掌握域名系统（ DNS）的工作原理。如果觉得下面的讨论如同天书，就应该回头看看第2章，了解DNS的基本知识。

named的启动通常是在系统启动时开始的，而且会一直处于运行状态直到关机。它从一个名为/etc/named.boot的配置文件中，取出自己的有关信息，而另外一些文件中包含将域名映射为地址的所有数据。后者叫作“ zone ”（区）文件。这类文件的格式和含义将在下一小节讨论。

要运行named，只须在提示行键入

```
# /usr/sbin/named
```

然后， named出现，并读取named.boot文件和所有指定的 zone文件。再以ASCII的形式，将自己的进程ID写入/var/run/named.pid，如有必要，将从主服务器下载所有的 zone文件，并在端口53上，开始监听DNS查询。

注意 目前，许多FTP站点上都有named二进制文件，各自的配置稍有不同。有的将自己的pid文件放在/etc内，有的则保存在/tmp或/var/tmp内。

5.14.1 named.boot文件

named.boot文件一般都非常小，其中包含一些指向 master文件和其他域名服务器的指针，master文件中包含的是区信息。启动文件中的批注以分号开头，并扩展到下一行。在详细讨论named.boot文件的格式之前，我们将看看 vlager主机的示范文件：

```
;
; /etc/named.boot file for vlager.vbrew.com
;
directory      /var/named
;
;              domain                    file
;-------------------------------------
cache          .                         named.ca
primary        vbrew.com                 named.hosts
primary        0.0.127.in-addr.arpa      named.local
primary        72.191.in-addr.arpa       named.rev
```

上面的示例中，cache和primary命令将信息载入named文件。该信息取自第二个参数中指定的master文件。其中包含DNS源记录的文本描述，详情如下。

这个示例中，我们将named配置为三个域的primary（主要）域名服务器，如同文件最后primary语句指定的那样。比如说，该示例中的第一行从名为 named.hosts的文件中，取出区数据，指令named充当vbrew.com的主要域名服务器。 directory（目录）关键字告诉它所有的区文件都在/var/named目录下。

cache条目非常特殊，其作用包括两方面：指导 named激活自己的缓冲区，并载入 cache文件指定的根域名服务器提示（在我们的示例中是 named.ca）。稍后将为大家详细讨论域名服务器提示。

下面列出了named.boot文件中最重要的几个选项：

■ diretory —— 该选项指定区文件驻留的目录。另外，还可能给出与该目录相关的文件名。重复利用 directory选项，可指定若干个目录。按照 Linux文件系统标准，这个目录应该是/var/named。

■ primary —— 该选项将一个域名和一个文件名作为自己的参数，声明本地域名服务器属于named域。作为一个主要域名服务器， named从给定的master文件加载区信息。一般说来，每个root文件中，至少总都应该有一个主要条目，用于逆向映射网络 127.0.0.0，该网络就是本地回送网络。

■ secondary —— 该选项将域名、地址清单和文件名作为自己的参数。它声明本地域名服务器是指定域的辅助主要服务器。辅助服务器中也保存有该域名的管理数据，但它不会从不同的文件收集这些数据，而是试图从主要服务器上下载。因此，地址清单中，至少应该为named指定一个主要服务器的IP地址。然后，本地服务器将依次和各域名服务器取得联系，直到成功传送区数据库，然后，这个区数据库便保存在第三个参数中指定的备份文件中。如果无主要域名服务器作出应答，本地服务器只好从备份文件中获取区数据了。最后， named将定期对区数据进行更新。详情可参见 SOA源记录类型。

■ cache —— 该选项将域名和文件名作为自己的参数。这个文件中包含根服务器提示，是一份指向根域名服务器的记录清单。目前，只能识别 NS和A类型的记录。域名参数一般是根域名“ .”。这一点对named而言是绝对重要的：如果cache选项不能在根文件中执行， named就根本不能开发出一个本地缓冲区。如果下一个服务器查询不是在本地网络上进行，必然会降低性能，增加网络负担。更有甚者， named将不能抵达任何一台根域名服务器，因而也不可能解析自己管理范围之外的其他地址。但在使用转发服务器时，情况有所不同（比较下面的forwarders选项）。

■ forwarders —— 该选项将地址清单作为自己的参数。该清单中的 IP地址指定的是一系列的域名服务器，如果named不能从自己的本地缓冲区内解析地址查询，它就会要求查询清单中的这些域名服务器。named依次对这些服务器进行查询，直到其中之一能够作出应答。

■ slave —— 该选项令域名服务器作为从属服务器。也就是说，这类服务器自身永远不会执行递归查询，只是将查询转发到 forwarders中指定的服务器。

sortlist和domain两个选项不在这里的讨论之列。另外，区数据库文件内，还有两条指令，它们是$INCLUDE和$ORIGIN。由于平常较少用到它们，所以，我们也不打算深入讨论。

5.14.2 DNS数据库文件

master文件包含在named.hosts之类的named文件内， master文件一直都有一个与之关联的域，叫作origin。这个域是用 cache和primary命令指定的。 master文件内部，允许你指定和该域相关的域和主机名。如果配置文件中给定的域名最后是一个句点“ .”，就会被视为绝对名，否则就被视为与origin相关。用“ @ ”，可借代这个origin。

包含在master文件内的所有数据被分为若干条 resource records（源记录，简称RR）。它们组成能通过DNS的最小信息单元。每条源记录都有类型。比如说 A类型的记录，将主机名映射为IP地址，而CNAME类型的记录，则将主机的别名和其正式主机名关联在一起。示例可参见清单5-1，该示例将向大家展示virtual brewery网络的named.hosts master文件。

master文件内的源记录表达式的格式如下：

```
[domain] [ttl] [class] type rdata
```

每个字段间用空格隔开。如果新行以大括号 {开头，最后一个字段以大括号 }结尾，这个条目就会在以后的几行内重复地出现。冒号和新行之间的所有记录都会被忽略。

■ domain —— 指定条目所用的域名。如果不指定域名， RR就沿用上一条RR采用的域。

■ ttl —— 为了强制解析器定期丢弃信息，每个 RR都有一个“ 生存时间 ”（ttl）。ttl字段指定自服务器获取的信息之有效时间，以秒计数。它是一个十进制数，最多有 8个数位。如果不指定ttl值，就会沿用前一条SOA记录的最小值。

■ class —— 这是一个地址类，类似于 IP地址的IN或Hesiod类中的对象所用的 HS。对TCP/IP连网来说，这个字段必须是IN。如果不指定类字段，将沿用前一条 RR的类。

■ type —— 该字段描述RR的类型。最常用的类型是 A、OA、PTR和NS。后续的小节将对RR的不同类型进行深入讨论。

■ rdata —— 该字段中容纳的是与RR关联的数据。它的格式与RR的类型有关。下面将针对各种RR类型，解释这一字段的格式。

下面列举了DNS master文件内所用的RR，但不完整。另外两条不常用的 RR将不作讨论。

■ SOA —— 描述权限区（SOA的意思是“ start of authority ”）。它标志SOA RR之后的记录中包含这个域的管理信息。 primary选项中包含的master文件内必须包含该权限区的 SOA记录。源数据中包含下面的字段：

■ origin —— 这是该域之主要域名服务器的规范主机名。通常被指定为绝对主机名。

■ contact —— 负责维护一个域的人的电子邮件地址，不过句点（ .）代替了“ @ ”。例如，Virtual Brewery的负责人是janet，那么这个字段就应该是janet.vbrew.com。

■ serial —— 区文件的版本号，用一个单独的十进制数来表示。只要区文件内的数据发生

了变化，这个数也应该相应增加。序列号是辅助域名服务器用以识别区信息何时发生变化的根据。为了保持更新，辅助服务器定期请求主要服务器的 SOA记录，并把它的序列号和缓存起来的SOA记录的序列号进行比较。如果不同，辅助服务器就会从主要服务器内将整个区数据库传回来。

■ refresh —— 指定时间间隔，以秒计。表示辅助服务器在两次复查主要服务器的 SOA记录期间，应该等待多久。再次提醒大家注意，这也是一个十进制数，最大有 8个数位。通常情况下，网络结构是不会经常变化的，所以对大型网来说，这个间隔可指定为一天，对小型网来说，甚至可以更长。

■ retry —— 指定时间间隔，表示在请求或区更新失败的情况下，辅助服务器应该隔多久才能重新尝试与主要服务器联系。这个值不能太低，因为服务器的临时性故障或网络问题可能导致辅助服务器浪费网络资源。最好设为 1.5h。

■ expire —— 指定时间，以秒计。这段时间一过，如果服务器仍然不能与主要服务器沟通，它就会将全部区数据最终丢弃。这个值一般较大。 Craig Hunt建议值是42天。

■ minimum —— 默认的ttl值，用于没有显示指定生存时间的源记录。它要求其他域名服务器在间隔一段时间后，丢弃 RR。但是，这个值和辅助服务器试着更新区信息的时间间隔是不相干的。这个minimum应该是一个非常大的值，对网络结构几乎一成不变的局域网来说，更是如此。最好设为一周或一个月。如果个别 RR可能频繁更动，仍然可以单独为它们设置 ttl。

■ A —— 将一个IP地址和主机名关联在一起。源数据字段中包含的地址是点分四段式。每台主机必须只有一条 A记录。这条A记录中的主机名被视为正式主机名或规范主机名。其他主机名均为别名，而且必须利用一条 CNAME记录，将它们与规范主机名对应。

■ NS —— 指向下属区的主（master）域名服务器。含有NS记录的原因，参见第3章。源数据字段中包含该域名服务器的主机名。要对这个主机名进行解析，就需要 A记录，它就是指定域名服务器之IP地址的所谓的glue record。

■ CNAME —— 将主机的别名和其规范主机名关联在一起。规范主机名是 master文件为一条A记录提供的；别名只是通过一条 CNAME记录，和规范主机名链接起来，而不是真正地拥有记录。

■ PTR —— 这类记录用于将 inaddr.arpa域内的名字和主机名关联起来。用于将 IP地址逆向映射为主机名，但所给的主机名必须是规范主机名。

■ MX —— 该RR宣布域的邮件交换器。宣布邮件交换器的原因，可参见第 14章。MX记录的格式如下：

```
[domain] [ttl] [class] MX preference host
```

host对域邮件交换器进行了命名。每个邮件交换器都有一个相关的整数型首选项值。对负责将邮件投递到域的邮件传输代理来说，它们将试着将邮件投递到拥有该域 MX记录的所有主机，直到成功为止。先从首选项值低的着手，从低到高尝试所有的主机。

■ HINFO —— 该记录提供系统硬件和软件的相关信息。其格式如下：

```
[domain] [ttl] [class] HINFO hardware software
```

hardware字段标识该主机所用的硬件。关于这个字段的指定，有相关的约定。其有效值可参考“ 已分配编号 ”（RFC 1340）。如果该字段中有空格，则必须用双引号将其封闭起来。software字段指的是该主机所用的操作系统软件。其有效值同样可从“ 已分配编号 ”（RFC

1340）中获得。

5.14.3 编写Master文件

清单5-1、5-2、5-3和5-4给出了关于域名服务器的几个示范文件，这些文件用于 brewery网络中的vlager主机。由于已经介绍过这个网络的特性（是一个单一的局域网），所以这里的示例相当简单。如果你面对的网络较为复杂，就不能只运行 named，还应参考Cricket Liu和Paul Albitz合著的《DNS和BIND》一书。

清单5-1中的named.ca cache文件展示了根域名服务器的提示记录示例。一个典型的 cache文件内，一般都描述了一打以上的域名服务器。利用本章最后要讲的 nslookup，便可获得根域的所有当前域名服务器清单。

注意 在没有安装根服务器提示：catch-22的情况下，是不能在自己的域名服务器内查询根服务器的！要避免此类情况的发生，可令nslookup另行采用域名服务器，或从清单5-1中的示范文件着手，获得一份列有所有有效服务器的清单。

清单5-1 named.ca文件

```
;
; /var/named/named.ca          Cache file for the brewery.
;                 We're not on the Internet, so we don't need
;                 any root servers. To activate these
;                 records, remove the semicolons.
;
; .                 99999999    IN    NS   NS.NIC.DDN.MIL
; NS.NIC.DDN.MIL    99999999    IN    A    26.3.0.103
; .                 99999999    IN    NS   NS.NASA.GOV
; NS.NASA.GOV       99999999    IN    A    128.102.16.10
```

清单5-2 named.hosts文件

```
;
; /var/named/named.hosts       Local hosts at the brewery
;                               Origin is vbrew.com
;
@                   IN  SOA    vlager.vbrew.com. (
                               janet.vbrew.com.
                               16          ; serial
                               86400       ; refresh: once per day
                               3600        ; retry:   one hour
                               3600000     ; expire:  42 days
                               604800      ; minimum: 1 week
                               )
                    IN  NS     vlager.vbrew.com.
;
; local mail is distributed on vlager
                    IN  MX     10 vlager
;
; loopback address
localhost.          IN  A      127.0.0.1
; brewery Ethernet
```

```
vlager              IN  A      191.72.1.1
vlager-if1          IN  CNAME  vlager
; vlager is also news server
news                IN  CNAME  vlager
vstout              IN  A      191.72.1.2
vale                IN  A      191.72.1.3
; winery Ethernet
vlager-if2          IN  A      191.72.2.1
vbardolino          IN  A      191.72.2.2
vchianti            IN  A      191.72.2.3
vbeaujolais         IN  A      191.72.2.4
```

清单5-3 named.local文件

```
;
; /var/named/named.local        Reverse mapping of 127.0.0
;                                Origin is 0.0.127.in-addr.arpa.
;
@                   IN  SOA    vlager.vbrew.com. (
                               joe.vbrew.com.
                               1           ; serial
                               360000      ; refresh: 100 hrs
                               3600        ; retry:   one hour
                               3600000     ; expire:  42 days
                               360000      ; minimum: 100 hrs
                               )
                    IN  NS     vlager.vbrew.com.
1                   IN  PTR    localhost.
```

清单5-4 named.rev文件

```
;
; /var/named/named.rev          Reverse mapping of our IP addresses
;                                Origin is 72.191.in-addr.arpa.
;
@                   IN  SOA    vlager.vbrew.com. (
                               joe.vbrew.com.
                               16          ; serial
                               86400       ; refresh: once per day
                               3600        ; retry:   one hour
                               3600000     ; expire:  42 days
                               604800      ; minimum: 1 week
                               )
                    IN  NS     vlager.vbrew.com.
; brewery
1.1                 IN  PTR    vlager.vbrew.com.
2.1                 IN  PTR    vstout.vbrew.com.
3.1                 IN  PTR    vale.vbrew.com.
; winery
1.2                 IN  PTR    vlager-if1.vbrew.com.
2.2                 IN  PTR    vbardolino.vbrew.com.
3.2                 IN  PTR    vchianti.vbrew.com.
4.2                 IN  PTR    vbeaujolais.vbrew.com.
```

5.14.4 验证域名服务器的设置

这里有一个很优秀的工具，可用于验证域名服务器的设置。其名为 nslookup，既可以交互式操作，又可以通过命令行操作。如果采用后一种方式，只须这样调用它：

```
nslookup hostname
```

然后，它就开始在resolv.conf指定的域名服务器查询主机名（如果改文件命名的服务器不止一个，nslookup将依次进行查找）。

但在交互模式中，情形更为有趣。除了查找个别的主机外，还要查找各种类型的 DNS记录并传送域的整个区信息。

如果调用nslookup时不带参数，nslookup就会显示出它所用的域名服务器，并进入交互式模式。出现“ > ”提示时，可键入任何一个你认为应该查询的域名。默认情况下，它会要求 A类型的记录，即其中包含与该域名相关的IP地址。

当然，也可执行“ set type=type ”，改变这个类型，后一个 type应该是上面所讲的源记录名之一或ANY。

比如，和nslookup的对话可能如下所示：

```
$ nslookup
Default Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

 sunsite.unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

Non-authoritative answer:
Name:    sunsite.unc.edu
Address:  152.2.22.81
```

如果是查询这样的域名 —— 无相关 IP地址、但可在 DNS数据库内找到其别的记录时，nslookup将无功而返，并带回这样的错误消息：“ No type A records found ”（未找到A类记录）。但是，可以利用set type命令，查询其他类型的记录。比如，要想获得 unc.edu的SOA，可以执行下列程序：

```
 unc.edu
*** No address (A) records available for unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

 set type=SOA
 unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

Non-authoritative answer:
unc.edu
        origin = ns.unc.edu
        mail addr = shava.ns.unc.edu
        serial = 930408
```

```
        refresh = 28800 (8 hours)
        retry   = 3600 (1 hour)
        expire  = 1209600 (14 days)
        minimum ttl = 86400 (1 day)

Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU       internet address = 128.109.157.30
```

类似的方式还可用于查询MX记录等。使用ANY类型时，将返回与指定名相关的所有源记录。

```
 set type=MX
 unc.edu
Non-authoritative answer:
unc.edu preference = 10, mail exchanger = lambada.oit.unc.edu
lambada.oit.unc.edu     internet address = 152.2.22.80

Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU       internet address = 128.109.157.30
```

除了用于调试设置之外，nslookup还可实际用于获得named.ca文件之当前根域名服务器列表。这是查询与根域关联的所有NS类型的记录而得的。

```
 set typ=NS

Name Server:  fb0430.mathematik.th-darmstadt.de
Address:  130.83.2.30

Non-authoritative answer:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL

Authoritative answers can be found from:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL
NS.INTERNIC.NET internet address = 198.41.0.4
AOS.ARL.ARMY.MIL        internet address = 128.63.4.82
AOS.ARL.ARMY.MIL        internet address = 192.5.25.82
AOS.ARL.ARMY.MIL        internet address = 26.3.0.29
C.NYSER.NET     internet address = 192.33.4.12
TERP.UMD.EDU    internet address = 128.8.10.90
NS.NASA.GOV     internet address = 128.102.16.10
```

```
NS.NASA.GOV      internet address = 192.52.195.10
NS.NASA.GOV      internet address = 45.13.10.121
NIC.NORDU.NET    internet address = 192.36.148.17
NS.NIC.DDN.MIL   internet address = 192.112.36.4
```

利用nslookup内部的help命令，便可获得nslookup能够使用的所有命令。

5.14.5 其他工具

还有几个工具，有助于你成为更好的 BIND管理员。下面简要为大家介绍两个。具体用法参见它们的相关文档。

hostcvt这个工具对最初的 BIND配置来说，是非常有帮助的，具体说来，是将你的/etc/hosts文件转换为用于 named的master文件。它同时生成了 forward（A类）和逆向映射（PTR类）条目，要注意别名等等。虽然它不可能全盘接管你的任务，比如你打算自行调节SOA记录中的超时值，增加MX条目等等，至少说来，能为你省几片阿斯匹林（意为为你省心）。hostcvt是BIND源代码中的一部分，少数FTP站点上也有其独立封装版本。

面对已设置好的域名服务器，你肯定想迫不及待地测试一番。据我所知，目前唯一的理想工具是dnswalk，它引导着你步步深入DNS数据库，查找常见错误，验证其间的信息是否一致。dnswalk新近已在comp.sources.misc上发布。归入DNS组的FTP站点都能使用它（如果你对这类站点一无所知，到ftp.uu.net去，准没错）。

第6章　串行链路网际协议

串行链路网际协议SLIP和PPP为“穷人”提供了接入因特网的可能。只需要一个 Modem和一个配备FIFO缓存的串行卡，除此以外，不再需要别的硬件。它的使用非常简单，而且费用低廉，越来越多的私人企业，以普通用户能够承受的价格，开始提供拨号 IP服务。

本章和第9章，我们将为大家介绍 SLIP和PPP驱动程序。SLIP驱动程序的存在有相当长一段时间了，而且其运行相当可靠。 PPP驱动程序是迈克·克拉翰和艾尔·依伊尔近来开发出来的。我们将在下一章详细讨论。

6.1　常规需求

要想使用SLIP或PPP，必须像前面几章描述的那样，配置一些基本的网络特性。比如，起码应该安装LoopBack接口，以及提供名字解析。在连上因特网时，人们肯定想使用 DNS。最简单的方法是把某个域名服务器的地址放入自己的 resolv.conf文件中：只要SLIP链接一激活，就会对这个服务器进行查询。该名字服务器离你上网的地方越近，越好。

但是，上面的方法不是最佳解决之道，因为所有服务器名查找仍然通过你自己的SLIP/PPP链接来完成。如果担心因此而造成的带宽开销，最好安装一个 caching-only域名服务器。它不能真正地充当一个域，对发自你的主机的所有 DNS查询来说，它只是一个中间站。不过，它的好处在于建立了一个缓冲区，所以所有的查询都只能在这个串行链接上发送一次。caching-only服务器所用的named.boot文件像下面这样：

```
: named.boot file for caching-only server
directory                       /var/named
primary     0.0.127.in-addr.arpa    db.127.0.0      : loopback net
cache       .                   db.cache        : root servers
```

除了named.boot文件外，还必须用一个有效的根域名服务器清单来安装 db.cache文件。

6.2　SLIP的工作原理

拨号IP服务器通过特定的用户账号，不间断地提供 SLIP服务。用户登录后，执行的不是一个常见的外壳，而是一个程序或外壳脚本，之后，再启用串行线路 SLIP驱动程序，并配置相应的网络接口。

有些操作系统上， SLIP驱动程序是一个用户空间程序；在Linux操作系统上，这个驱动程序则集成在内核中，因此，其运行速度快得多。但是，它要求串行线路必须被显式转换为SLIP模式。这是通过一个特殊的 tty线路法，即SLIPDISC来完成的。在tty处于普通线路法（DISC0）时，只采用普通的读(2)和写(2)调用与用户进程交换数据， SLIP驱动程序不能对tty进行读写操作。SLIPDISC中，角色发生了变化：所有用户空间进程都被封锁，不得对 tty进行读写操作，而来自串行端口的所有数据都直接传送到 SLIP驱动程序。

SLIP驱动程序本身能够识别 SLIP协议的各种变形。除了常规的 SLIP外，它还能识别

CSLIP，显著提升交互式对话的流通量。CSLIP对输出IP包进行所谓的Van Jacobson报头压缩。至于Van Jacobson报头压缩的详情，可参见RFC-1441。另外，每个SLIP协议的变形都有6位版本。

要把串行线路转换为SLIP模式，最简单的方式是利用slattach工具。假设Modem已经存在于/dev/cua3目录中，而且已经成功登录到SLIP服务器。那么，执行下面的语句：

```
# slattach /dev/cua3 &
```

就会把cua3的线路切换为SLIPDISC，并把它附着在其中一个SLIP网络接口上。如果是初次激活SLIP链接，该链接就会附在sl0接口上；如果是第二次激活SLIP链接，就会附在sl1接口上，以此类推。目前，内核能支持的并发SLIP链接多达8条。

slattach选择的默认封装是CSLIP。大家可利用-p交换，选用其他模式。要想采用普通SLIP（无压缩的），就应该用

```
# slattach -p slip /dev/cua3 &
```

其他模式是：cslip、slip6、cslip6（6位版本的SLIP）和适用于SLIP的自适应（adaptive）模式。后者留在内核中，用于查找远程终端采用的SLIP封装类型。

注意，你采用的封装模式必须和你的对等体一样。比如，如果 cowslip采用的是CSLIP，你也必须如此。如果不一样，发给远程主机的 ping就不能收到自远程主机返回的包。如果其他主机ping你，你的控制台上可能会出现这样的信息：“不能建ICMP报头”。为避免此类情况的发生，必须采用自适应SLIP模式。

事实上，slattach不仅允许你启用SLIP，还允许其他协议利用串行线路，比如PPP和KISS（“火腿”无线电发烧友使用的另一个协议）。有关详情，请参阅slattach手册。

从串行线路上转入SLIP驱动程序之后，还需要配置网络接口。再次提醒大家注意，我们是利用ifconfig和route命令来完成这一配置的。假设现在已经从 vlager，拨号连接到一个名为cowslip的服务器，然后执行

```
# ifconfig sl0 vlager pointopoint cowslip
# route add cowslip
  # route add default gw cowslip
```

第一个命令把网络接口配置为到 cowslip服务器的点到点链接，第二和第三个命令将cowslip作为一个网关，增加了一条到cowslip的路由和一条默认路由。

取消SLIP链接时，必须先用带有del选项的route命令，删除所有通过cowslip服务器的路由，然后再取消网络接口，发送slattach挂起信号。最后，再次利用终端程序，将Modem挂起。

```
# route del default
# route del cowslip
# ifconfig s10 down
# kill -HUP 516
```

6.3 dip的使用

接下来的事情非常简单。但是，人们常常希望上述步骤能够自动进行，比如说，只须调用一个简单的命令，便可执行上面的所有步骤。这种想法可行吗？当然，这正是 dip的绝活（dip的意思是拨号IP，由弗雷德·范·肯蓬编写）。当前发布的dip版本是3.3.7。

dip为一种处理Modem的简单脚本编写语言提供了一个翻译器，能把串行线路转换为 SLIP

模式，并配置网络接口。这是非常原始而非常受限的，但对大多数情况来说，却是行之有效的。dip的下一个版本可能会扩展到其他语言。

为了能配置SLIP接口，dip提出“ root ”权限要求。现在，它开始试着建立到“ root ”的dip setuid，所以所有的用户都可以在没有root访问权的情况下，拨号连到特定的SLIP服务器。这是非常危险的，因为利用dip安装的bogus接口和路由可能导致网络上分发的路由混乱不堪。更为糟糕的是，可能会造成有些用户随意连接到任何一台 SLIP服务器，并向你的网络发动恶意攻击。所以，如果你想允许自己的用户向各个 SLIP服务器发起SLIP连接，并编写交换程序的话，就应该利用建立连接的特定脚本，让这些交换器来调用 dip。然后，这些程序就可以成为setuid root了（diplogin也可以，同时也必须运行setuid，参见本章的最后一节）。程序6-1中包含了一个dip脚本示例。

程序6-1 dip脚本示例

```
# Sample dip script for dialing up cowslip
# Set local and remote name and address
get $local vlager
get $remote cowslip

port cua3                # choose a serial port
speed 38400              # set speed to max
modem HAYES              # set modem type
reset                    # reset modem and tty
flush                    # flush out modem response

# Prepare for dialing.
send ATQ0V1E1X1\r
wait OK 2
if $errlvl != 0 goto error
dial 0123456789
if $errlvl != 0 goto error
wait CONNECT 60
if $errlvl != 0 goto error

# Okay, we're connected now
sleep 3
send \r\n\r\n
wait ogin: 10
if $errlvl != 0 goto error
send Cvlager\r
wait ssword: 5
if $errlvl != 0 goto error
#better not leave your password in ascii (thanx noud)
password
wait running 30
if $errlvl != 0 goto error
#to set up your remote and local IP
get $remote remote
print remote = $remote
if $errlvl != 0 goto error
```

```
wait to 3
get $local remote
print local = $local
if $errlvl != 0 goto error


# We have logged in, and the remote side is firing up CSLIP.
print Connected to $remote with address $rmtip
default                    # Make this link our default route
mode CSLIP                  # We go to CSLIP mode, too
# fall through in case of error
error:
print CSLIP to $remote failed.
```

在上面的示例中可产生一个样本脚本。该脚本可被用来连接到 cowslip，方法是用脚本名作参数引发dip。

```
# dip cowslip.dip
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.

connected to cowslip.moo.com with addr 193.174.7.129
#
```

连接到cowslip服务器，启用CSLIP链接之后，dip便离开终端，转入后台运行。之后，你就可以开始利用CSLIP链接上的普通网络服务。如果想中止这条链接，只须调用带有 -k选项的dip。该调用利用/etc/dip.pid:中的进程ID dip记录，向dip进程发出一个“挂起”信号（关于三字缩写词的详情，参阅newsgroup alt.tla）。

```
# kill -k
```

dip脚本编写语言中，前缀是 $符号的关键字是变量名。 dip有一个预先定义好的变量集（参见下面）。比如，$remote和$local这两个变量中，包含的是 SLIP链接所涉及的本地主机和远程主机名。

前面的脚本示例中，前两个语句是 get命令。dip中，get命令是用来设立变量的。我们的示例中，本地和远程主机名分别被设为 vlager和cowslip。

随后的五条语句设立了终端线路和 Modem。reset命令向Modem发出一个reset字串；采用贺氏标准的Modem使用的则是ATZ命令。下一个语句直接作出 Modem应答，所以下面几行的登录对话将正常进行。这里的登录对话相当直接：简单地拨叫 41988（cowslip的电话号码），再利用“ hey-jude ”这个密码，登录到 Svlager账号。wait命令使dip等待作为其第一个变量的字串；如果没有收到此类的字串，作为第二个参数的指定编号将等待数秒，直到超时为止。 if命令分布在登录过程中，以复查执行命令期间是否有错误发生。

登录后执行的命令都是 default和mode，前者令SLIP链接到通向所有主机的默认路由；后者则在线路上启用SLIP模式，并开始配置接口和路由表。

dip参考

尽管dip的应用非常广泛，但目前尚未出台标准文档。所以我们将在本小节向大家介绍一些常用的dip命令。如果在测试模式下调用 dip，并输入help命令，就能看到关于dip所有命令的

总述。如果想了解各个命令的相关语法，输入该命令时，不带任何参数即可；当然，不带参数的命令是没有用的。

```
DIP help
DIP knows about the following commands:

databits default  dial     echo     flush
get      goto     help     if       init
mode     modem    parity   print    port
reset    send     sleep    speed    stopbits
term     wait

DIP echo
Usage: echo on|off
DIP
```

随后的描述中，显示DIP>提示行的示例将向大家展示如何在测试模式下输入命令，以及该命令产生的输出又是什么。没有提示行的示例则当作脚本引用使用。

1. Modem命令

许多dip命令都提供了对串行线路和 Modem的配置。有的是显式的，比如用于选定串行端口的port命令，以及用于设立命令行参数的 speed、databits、stopbits和parity命令。

modem命令用于选定 Modem类型。目前为止，只提供了对贺氏 (HAYES，要求大写）标准的支持。执行dip时，必须为它提供一个 Modem类型，不然，它会拒绝执行 dial和reset命令。reset命令向Modem发出一个reset字串；这个字串的用法和选定的 Modem类型有关。对采用贺氏标准的Modem来说，则是ATZ。

刷新（Flush）代码可用来刷新 Modem发出的所有应答信号。不要把它和 reset后面的对话脚本搞混淆了，因为后者读取的是从前一条命令返回的 OK应答。

拨号之前，须通过 init命令选定一个准备传递给 Modem的初始化字串。贺氏 Modem的默认字串是“ ATE0 Q0V1 X1 ”，它将打开命令和长结果代码应答，并选定屏蔽拨号（即不复查拨号声音）。

最后，dial命令把选定的初始化字串发给 Modem，并拨叫远程系统。贺氏 Modem的默认dial命令是ATD。

2. echo和term命令

echo命令实际上是一个调试助理，利用 Echo On能使dip把发给串行设备的全部内容都反映到控制台。如果想关闭反映功能，调用 Echo Off即可。

dip还允许暂时性离开脚本模式，进入终端模式。在终端模式下，可以像使用其他普通终端程序一样，使用 dip，在串行线路上实施读写操作。要退出终端模式，输入Ctrl+]即可。

3. get命令

dip方法中，get命令用于设立变量。最简单的方式是把变量设为常量，就像前一个示例那样。但是，也可以指定关键字询问来代替值，提示用户输入变量值：

```
DIP get $local ask
Enter the value for $local:
```

第三种方式是试着从远程主机中取得一个变量值。虽然这种方式有些离谱，但在某些情

况下，却是非常有用的：比如，有的 SLIP服务器禁止在SLIP链接上使用你自己的IP地址，只要你一拨号，它就会为你分配一个取自地址缓冲池的地址，屏幕上还会提示你已经分配地址。如果屏幕上的提示消息像这样：“ 你的地址： 193.174.7.202 ”，下面的dip代码段就会让你选定这个地址；

```
wait address: 10
get $locip remote
```

4. print命令

这个命令用于向控制台反映 “ 已经开始使用dip ”。任何dip变量都可用于print命令中，比如

```
DIP print Using port $port at speed $speed
Using port cua3 at speed 38400
```

5. 变量名

dip只能理解预先定义好的变量集。所以变量名总是以美元符号（ $）为前缀，而且必须采用小写形式。

$local和$locip变量中包含本地主机名和IP地址。主机名的设立会使dip将合法的主机名保存在$local内，与此同时，为$locip分配相应的IP地址。类似的情形常发生于设立$locip变量时。

$remote和$rmtip变量和上面的两个变量差不多，只不过对象换为远程主机名及其地址。$mtu变量中有针对连接的MTU值。

上面这5个变量是唯一能用get命令直接进行分配的变量。主机的其他变量只能通过相应的命令来设立，但这些变量也可当作打印语句来使用，它们是： $modem、$port和$speed。

至于$errlvl，我们可通过该变量访问上一条命令的执行结果。如果错误级为 0，则表示命令成功，若是非零值，则表示有错。

6. if和goto命令

至于if命令，与其说它是通常调用 if条件的方法，倒不如说它是一个条件方法。它的语法如下：

```
if var op number goto label
```

上面的表达式中，必须对 $errlvl、$locip或$rmtip变量进行一个简单的比较。第二个运算对象必须是一个整数；运算符可以是： ==、!=、<、>、<=和>=。

goto命令的作用是令脚本在出现标签的那一行之后，继续执行下去。标签必须作为首要的令牌出现在行上，而且后面必须紧跟一个冒号（ :)。

7. send、wait和sleep命令

这三个命令有助于实施 dip中的简单对话脚本。 send向串行线输出自己的参数。该命令不支持变量，但能够识别全部 C样式的反斜杠字符序列，比如 n和b。平铺字符（~）用作回车／新行的缩写。

wait采用一个词来作为自己的参数，并对串行线路上的所有输入进行查看，直到认出自己需要的那个词为止。这个词本身不含任何空格。还可以为 wait指定一个超时值，作为它的第二个参数；如果规定时间内没有收到 wait需要的词，该命令就会随值为 1的$errlvl变量返回。

8. mode和default命令

这两个命令用于将串行线路置入 SLIP模式，并配置接口。

mode命令是dip在进入daemon模式之前所用的最后一个命令。如果不出错的话，这个命令

就不会返回。

mode采用一个协议名作为自己的参数。目前的 dip能够把SLIP和CSLIP识别为合法的协议名。但dip的当前版本还不能识别自适应 SLIP。

在串行线路上启用SLIP模式之后，dip便执行ifconfig，把接口配置为点到点链接，并调用route，设立到远程主机的路由。

另外，如果在对话脚本执行 mode命令之前，就执行default命令，dip同样会令默认路由指向SLIP链接。

6.4 运行于服务器模式

SLIP客户机的安装是最困难的一步。相对而言，把自己的主机配置为 SLIP服务器，要简单的多。

安装SLIP的方法之一：在服务器模式下使用 dip，这是通过把它作为 diplogin，再对它进行调用的方式来完成的。其主要配置文件是 /etc/diphosts，该文件把登录名和为该主机分配的地址关联在一起。还有一种方法是：利用 sliplogin，它是一个衍生于BSD的工具，其特色在于它是一种更为灵活的配置方案，允许你在主机连接或取消连接时，执行外壳脚本。不过，这一方案正处于测试阶段。

两种方法都要求你为每个 SLIP客户机设立一个登录账号。比方说，如果你为dent.beta.com的Authur Dent提供了SLIP服务，就要在你的passwd文件中添加下面一行，创建一个名字为dent的账号：

```
dent:* 501:60:Authur Dent ' s SLIP account:/tmp:/usr/sbin/diplogin
```

然后，再用passwd工具设立dent的密码。

现在，dent登录时，dip就会作为它的服务器，开始启动。为查看他是否拥有 SLIP使用许可，它会在/etc/diphosts文件中查找这个用户名。该文件详细记载了每个 SLIP用户的访问权限和连接参数。比如，dent的登录条目是这样的：

```
dent::dent.beta.com:Authur Dent:SLIP,296
```

用冒号隔开的字段中，第一个是用户必须采用的登录名。第二个字段中有一个额外的密码（见下面）。第三个字段是呼叫方主机的主机名或 IP地址。下一个字段是一个没有（目前还没有）特别含义的信息字段。最后一个字段描述的是连接参数。它用逗号隔开，逗号前指定的是协议名（目前，只能是SLIP或CSLIP），逗号后则是MTU（最大传输单元）。

dent登录时，diplogin便从diphosts文件中抽出关于他的相关信息，如果第二个字段非空，就会提示用户输入外部安全密码。然后，它会将用户输入的字串与 diphosts文件中的（未加密）密码进行比较。如果不相同，就会拒绝该用户登录。

另一种方法是：将串行线路置入 CSLIP或SLIP模式，diplogin继续进行，并开始配置接口和路由。这条链接会一直存在，直到用户取消链接或 Modem掉线为止。然后，diplogin将串行线返回普通线约束，并退出。

diplogin要求用户拥有超级用户特权。如果没有运行 diplogin setuid根的dip，就应该令diplogin做一份独立的dip备份，而不是作一个简单的链接。然后，diplogin就可以在不影响dip本身状态的前提下，安全地进行 setuid。

第7章　点到点协议

和SLIP一样，PPP也是通过串行链接收发数据报时采用的协议，但解决了前者存在的两大不足，它让通信双方自行对诸如启动时的IP地址、最大传输单元等选项进行协商，并提供客户机验证。针对每个功能，PPP都有一个单独的协议。下面，我们将简要介绍一下PPP协议的基本构成。如果想进一步了解PPP，强烈推荐大家参阅RFC-1548中的PPP规格，以及相关的参考丛书。

PPP的底层是“高级数据链路控制协议”（High-Level Data Link Control Protocol, HDLC）。HDLC定义了单帧PPP的边界，并提供了一个16位的校验和。与更为原始的SLIP封装模式相反，PPP帧能够保存自其他协议（IP协议除外）包，比如Novell的IPX协议和Apple Talk协议。PPP协议是怎样做到这一点的呢？答案是：在基本 HDLC帧内增添一个协议字段。所谓基本 HDLC帧，就是验证包类型是否由某个帧负责传送的帧。

链接控制协议(Link Control Protocol,LCP)，位于HDLC的顶部，用于协商数据链接选项，比如“最大接收单元”（MRU）等。最大传输单元表示链接方同意接收的数据报的最大字节数。

在配置PPP链接时，一个重要步骤是客户机验证。尽管它不是强制实施的，但对拨号线路来说，却是必不可少的。通常，被呼叫方（即服务器）要求客户机证明它知道密钥，并藉此对客户机的身份进行验证。如果呼叫方不能提供正确的密钥，连接就会中断。利用 PPP时，验证是双向的：也就是说，呼叫方也可以要求服务器验明它自己的身份。这两个验证过程彼此并不相干。不同类型的身份验证，采用的协议是不同的，这一点，我们将在后续小节进行讨论。一个叫做“密码验证协议”（Password Authentication,PAP），另一个叫做“盘问握手验证协议”（Challenge Handshake Authentication Protocol,CHAP）。

对通过数据链路路由的每个网络协议（比如IP和Apple Talk）来说，我们都可以利用相应的网络控制协议来对它们进行动态配置。例如，在打算通过链路发送IP数据报时，两端的PPP必须先对彼此使用的IP地址进行协商。这时所用的控制协议就是IPCP，即“Internet协议控制协议”。

除了通过链路发送标准的IP数据报外，PPP还支持IP数据报的Van Jacobson报头压缩。这种压缩技术将TCP包的报头缩小到3个字节。另外，它还用于CSLIP，即人们常说的VJ报头压缩。是否使用这种压缩，同样要在启动时，通过 IPCP进行协商。

7.1　PPP打开

在Linux中，PPP的作用分为两大类：其一，位于内核的低级 HDLC驱动程序；其二，处理不同控制协议的用户空间 pppd daemon。编写本书时，PPP的当前版本是Linux-ppp-1.0.0，其中包含内核PPP模块pppd和一个程序，该程序名为chat，用于拨号远程系统。

注意　当前PPP协议支持属于内核专用，诸如 pppd daemon之类的支持程序位于metalab.unc.edu/Pub/Linux/system/network/serial/ppp/（当前版本是ppp-2.3.4）。

PPP内核驱动程序是迈克·克拉翰编写的。 pppd衍生于一个免费的PPP执行程序，适用于Sun和386BSD机器，是德鲁·帕金斯和其他人一起编写的。由艾尔·依伊尔移植到 Linux中。

像SLIP一样，PPP是通过一种特殊的线路规则来实现的。要把串行线路当作 PPP链接使用，应该像以往一样，先通过 Modem建立一条链接，再把该线路转换为 PPP模式。这种模式下，收到的所有数据都被传递到 PPP驱动程序，驱动程序再对收到的这些 HDLC帧进行有效性检查（每个HDLC帧都带有一个16位的校验和），然后再解开封装，并开始分发数据。目前，可以选择性地采用Van Jacobson报头压缩，对IP数据报进行处理。如果有 IPX支持，PPP驱动程序还能够对IPX包进行处理。

内核驱动程序在 pppd的帮助下，得到了增强。在链接发送真正的网络通信之前，PPP daemon会执行必要的初始化和身份验证阶段。可通过许多选项，对 pppd的行为进行调整。由于PPP非常复杂，要想在一章里，全面介绍它是不可能的。因此，本书不能涵括 pppd的方方面面，只能为大家做一个简要的介绍。关于 PPP的更多详情，可参考 PPP手册和pppd源文件中的 README，它们将有助于你深入了解本书中没有介绍到的内容。如果在参阅所有文档之后，还难解心中的困惑，建议访问 comp.protocols.ppp和linux.dev.ppp，这个地方群英汇萃，多是开发pppd的专家。

7.2 运行pppd

在打算通过PPP链接接入因特网时，必须设立一些基本的网络特性，比如 loopback设备和解析器。关于这两者的设立，我们已在前一章讨论过。要注意的是：串行链接上使用的是 DNS，参见前一章。

如何利用 pppd建立一条PPP连接呢？好了，假设我们又从 vlager着手。现在，已经拨号呼叫PPP服务器c3po，登录到ppp账号。c3po服务器已经启动了自己的 PPP驱动程序。退出用于拨号的通信程序之后，执行下面的命令

```
## ppppd /dev/cua3 38400 crtstcs defaultroute
```

串行线路cua3便转换为PPP模式，并建立了一条通向 c3po服务器的IP链接。串行端口上所用的传输速率将是38400bps。crtscts选项打开了端口上的硬件设备握手，其绝对传输速率必须在9600bps以上。

启动之后，pppd所做的第一件事是利用 LCP与远程用户端协商一些链接特性。通常情况下，pppd试图协商的默认选项集都有用，所以我们不打算在这里深入讨论。关于 LCP的详情，我们将在后续小节讨论。

然后，pppd利用IPCP（IP控制协议），与其对等体一起，对 IP参数进行协商。由于我们没有指定任何特定的 IP地址，所以pppd将试图利用它获得的地址，这个地址是令解析器查找主机名所获得的。之后，通信双方向彼此通告各自的地址。

一般说来，这些默认设置都不会有错。即使你的机器运行于以太网，也可以在以太网和 PPP接口上采用同样的IP地址。不管怎么说，pppd都允许采用不同的地址，甚至要求你的对等体也采用某个特定的地址。我们将在下一小节讨论这些选项。

走完IPCP设立过程后，pppd将进行主机网络层的准备工作，以使用这条 PPP链接。首先，对已激活的第一条PPP链接，采用ppp0，第二条链接采用 ppp1，以此类推，把PPP网络接口配置为“点到点链接”。接下来，设立一条路由表条目，使之指向链接另一端的主机。在上面的示例中，pppd令默认路由指向c3po，因为我们把“默认路由”选项给了它（如果目前还没有路由的话，只能安装默认网络路由）。这样，将导致所有发向非本地网络上主机的数据报被发

送到c3po。pppd支持大量不同的路由选择方案，我们将在本章稍后详情讨论。

7.3 使用选项文件

pppd解析自己的命令行参数之前，会对一些文件进行查看，找出默认选项。这些文件中可能存在任何一个合法的命令行参数，它们分散在不同的命令行中。 hash符号引入了译注。

第一个选项文件是/etc/ppp/options，该文件通常是pppd启动时的第一个查看对象。利用它来设置全局默认选项倒是个好主意，因为它允许你防止你的用户进行那些破坏网络安全性的操作。例如，如果想要 pppd要求某种形式的身份验证（非 PAP，即CHAP），就要在该文件中增添auth选项。网络用户是不能修改这个选项的，所以要和不在身份验证数据库内的任何一个系统建立PPP链接是完全不可能的。

继/etc/ppp/options之后，pppd查看的第二个选项文件是用户根目录中的 .pprc文件。它允许每个用户指定他／她自己的默认选项集。

/etc/ppp/options选项文件示例：

```
# Global options for pppd running on vlager.vbrew.com
auth                # require authentication
userhostname        # use local hostname for CHAP
lock                # use UUCP -style device locking
domain vbrew.com    # our domain name
```

前两类选项文件适用于身份验证，我们将在稍后详细讨论。 lock关键字令pppd采用标准的UUCP设备锁定方法。有了这一约定，访问串行设备（比如 /dev/cua3）的每个进程都会在UUCP假脱机目录中，创建一个名为 LCK...cua3的锁定文件，表示该设备正在使用中。对 PPP正在使用的串行设备来说，这样可以有效地防止其他程序（比如 minicom和uucico）也来打开这个串行设备。

全局配置文件提供这些选项的原因在于：禁止网络用户对上面所说的选项进行修改，为整个网络的安全提供一定的保障。但请注意，有些选项是可以修改的，比如连接字串。

7.4 用chat拨出

上一个示例中，在启动 pppd之前，必须手工建立连接，这样带来的不便，可能是很多人都不愿意经历的。和 dip不一样， pppd没有可以用于拨叫远程系统和登录的脚本编写语言，但它也不纯粹依赖于某些外部程序或外壳脚本。利用连接命令行选项，便可把执行脚本编写的命令交给pppd。然后， pppd将该命令的标准输入和输出重新定向到串行线路。有用的编写语言之一是Expect，它是唐·利贝斯编写的。它是一种基于 Tcl的功能非常强劲的语言，简直就是为这类应用量身定做的。

pppd封装内有一个类似的程序，名为 chat（对话），它允许你指定一个 UUCP样式的对话脚本。对话脚本基本上由一个交替式的字串序列组成，这些字串是我们希望从远程系统中收到的（即expect字串），以及我们准备发送的应答。下面，我们将分别调用希望收到的和准备发送的字串。这是从一个对话脚本中摘录出来的典型字串：

```
ogin: b1ff ssword: s3krst
```

它要求对话等待远程系统发送登录提示并返回登录名 b1ff。我们只有等待登录：登录提示

是大写L，还是小写l，都无关紧要。下一个字串是 expect字串，它也会令对话等待密码提示，并发送应答中的密码。

这基本上就是一个对话脚本所包含的内容。当然，拨号到 PPP服务器的一个完整脚本中，还必须包含恰当的Modem命令。假如你的Modem能够识别贺氏标准的指令集，而且服务器的电话号码是318714，那么，与c3po建立连接的完整对话脚本调用如下所示：

```
$ chat -v ' ATZ OK ATDT318714 CONNECT ' ogin: ppp word: GaGariN
```

根据定义，第一个字串必须是一个 expect字串，但由于modem在我们剔出这个字串之前，是没有任何反应的，所以在指定一个空字串之后，我们就将对话跳到第一个 expect字串。然后，发送ATZ命令，这是采用贺氏标准的 modem所用的“ 重新设定 ”命令，并等待 modem应答（OK）。下一个字串向对话发送 dial命令的同时，还发送了电话号码，并希望在应答中收到CONNECT消息。随后又是一个空字串，因为我们不打算发送任何命令，只是等待登录提示。对话脚本的其余部分的工作原理和上面描述的完全一样。

-v选项会令对话把所有的活动记录到系统日志 daemon的loacal2设备中。（如果对syslog.conf进行编辑，将这些日志消息重定向到一个文件中，一定要确保该文件是非全球可读型，因为默认情况下，该对话也会对整个对话脚本进行记录 —— 包括密码和所有字串）。

在命令行指定对话脚本是很不安全的，因为网络用户用 ps命令，就能够看到一个进程的命令行。怎样避免呢？把对话脚本放在一个诸如 dial-c3po的文件内。令对话从该文件中读取脚本，而不是为它指定后面跟该文件名的 -f选项，以从命令行读取。完整的 pppd结果如下所示：

```
# pppd connect “ chat -f dial-c3po ” /dev/cua3 38400 -detach \crtscts modem
defaultroute
```

除了指定拨号脚本的连接选项外，我们还在命令行增加了两个选项： nodetach（该选项吩咐pppd不要脱离控制台，转为后台进程）和 modem关键字。modem关键字令pppd在串行设备上执行一些modem专有的动作，比如在拨号前后，挂断线路。如果不用这个关键字， pppd就不会对端口的DCD线路进行监控，从而导致远程终端意外挂断时， pppd不会对该线路进行保护。

上面的示例相当简单；对话可以接受更为复杂的对话脚本。其中一个最有用的特性是能够指定终止对话的错误字串。典型的终止字串是诸如 BUSY、NO CARRIER之类的消息，被呼叫的号码正忙或不能接受电话接入时， modem常常会产生此类消息。为了使对话能立即辨认出这些消息，而不是等待超时，可以在脚本开始处，利用 ABORT关键字指定这些消息：

```
$ chat -v ABORT BUSY ABORT ' NO CARRIER' " ATZ OK ...
```

也可采用类似的方式，插入 TIMEOUT选项，修改对话脚本各部分的超时值。有关详情，请查看对话手册。

有时，大家可能还想看到对话脚本各部分的违例情况。比如，在没有收到远程端的登录提示时，大家可能想发送一个 BREAK，或一个回车。这是通过在 expect字串后添加一个子脚本来完成的。子脚本由一序列 send-和expect-字串组成，与一个完整的脚本一样，用连字号 -隔开。只要规定时间内没有收到希望收到的字串，就会开始执行子脚本。在上面的示例中，我们将对对话脚本作出以下的修改：

```
ogin:-BREAK-ogin: ppp ssword: GaGariN
```

现在，当对话看不到远程系统发送的登录提示时，就会通过先发送一个 BREAK的方式，执行子脚本，然后再次等待登录提示。如果出现提示，脚本就会一如既往地运行，否则，就会因出现错误而中断。

7.5 PPP设置的调试

默认情况下，pppd会将所有的警告和错误消息记录在系统日志的 daemon设备中。所以，你必须在syslog.conf中添加一个条目，使其重定向到一个文件中，甚至可以定向到控制台，不然，系统日志就会将这些消息丢弃。下面的条目将所有警告和错误消息统统发到 /var/log/ppp-log：

```
daemon.*     /var/log/ppp-log
```

如果你的PPP设置不能立即运行，那么查看这个日志文件，将提示你错在哪里。如果仍然没有解决问题，还可利用 debug选项，打开特别调试输出。它将使 pppd把所有收发控制包的内容统统记录在系统日志内。这样一来，所有消息都能进入 daemon设备。

最后，PPP最突出的特性便是启用内核级的调试。这是通过调用带有 kdebug选项的pppd来完成的。它后面跟着一个数字化参数，该参数是下列值的按位“ 或 ”：1)代表一般调试消息；2)代表打印所有接入 HDLC帧的内容；3)是令驱动程序打印所有流出 HDLC帧的内容。为捕捉内核调试消息，必须运行对 /proc/kmsg文件进行读取的 syslogd daemon，或者klogd daemon。两者都将内核调试定向到系统日志的内核设备。

7.6 IP配置选项

IPCP用于协商链路配置时的两个 IP参数。通常情况下，每个对等体都可发送一个“ IPCP配置请求 ”包，表示它想修改哪些默认值，并且将它们改为什么值。远程端收到之后，便依次检查各个选项，然后，要么接受它，要么否决它。

pppd将对大量的IPCP选项进行协商。要对这一协商进行调整，可通过不同的命令行选项来完成，如下所示。

7.6.1 IP地址的选择

上面的示例中，我们让 pppd拨号c3po，并建立了一条IP链接，没有预计在链接的任何一端选择特定的IP地址。相反地，我们选择 vlager的地址来作为本地IP地址,而让c3po仍然用它自己的地址。但是，有时，能够对链接两端的地址进行控制是非常有用的。 pppd支持IP地址的变更。

要想得到特定的IP地址，一般需要提供带有下一选项的 pppd：

```
local addr:remote addr
```

local_addr和remote_addr必须指定为点分四段式或主机名（这个选项中采用主机名会对CHAP身份验证产生影响。详情参考下面的 CHAP小节）。这样，pppd就会试图将第一个地址作为自己的IP地址，第二个地址作为其对等体的 IP地址。如果IPCP协商过程中，其对等体否决了这两个地址，就不能建立 IP链接了（可通过为 pppd提供ipcp-accept-local和ipcp-accept-remote这两个远程选项，令其对等体 PPP忽略前面提供的IP地址。有关详情，可参考手稿）。

如果只想设立本地 IP地址，接受对等体使用的任何一个地址，省略 remote_addr即可。例

如，要想vlager不采用它自己的IP地址，而使用地址130.83.4.27，就应在命令行上为它提供130.83.4.27:。类似地，如果只想设立远程地址，可令local_addr为空。默认情况下，pppd将采用与你的主机名相关的那个地址。

对那些控制多个客户站点的PPP服务器来说，它们通常动态分配地址：只对来访的系统分配地址，也就是说，登录主机所分配到的地址是动态的。在拨叫这类服务器时，必须保证pppd不从服务器发出任何特殊IP地址请求，但可以接受服务器要求你使用的那个地址。这意味着禁止你指定local_addr参数。另外，对你来说，必须采用noipdefault选项，该选项将令pppd等待对等体提供IP地址，而不是采用本地主机的地址。

7.6.2 通过PPP链路的路由

设置网络接口之后，pppd通常只设置一条通向其对等体的主机路由。如果远程主机在局域网上，你应该确信还能够连接到对等体以后的那些主机：也就是说，还必须设置一条网络路由。

由上可知，利用defaultroute（默认路由）选项，可以要求pppd设置默认路由。如果你拨叫的PPP服务器担当你的Internet网关这一角色，这个选项就会变得相当有用。

另一方面，如果你的系统只是一个单一主机的网关，情况也会非常容易处理。比如，以Virtual Brewery的一名员工为例，他的主机名为Loner。在通过PPP链路连接到vlager时，他使用的是Brewery子网上的一个地址。在vlager这一端，我们在pppd后添加了proxyarp选项，它将为Loner主机安装一个代理ARP条目。如此一来，我们就可以从Brewery和Winery的任何一台主机，访问Loner主机。

但是，事情并不如想像中的那样简单。以链接两个局域网为例，通常要求增加一条特殊的网络路由，因为这类网络都可能有自己的默认路由。另外，让链接的对等体都将PPP链接用作默认路由，将产生一个循环，也就是说不知其目的地的数据包将在对等体间不停地ping、pong，直到这些包的生存期满为止。

在将子以太网连接到vlager时，它将一如既往地把默认路由指向vlager。但在vlager这一端，我们必须为通过子以太网的子网3安装一条网络路由。鉴于此，我们采用了一个以前尚未谈到的pppd特性——ip-up命令。这是一个外壳脚本或程序，位于/etc/ppp（在配置完PPP接口之后执行的）。ip-up命令是随下面的参数一起调用的：

```
ip-up iface device speed local addr remote addr
```

iface命名网络接口使用的名字，device是串行设备文件所用的路径名（如果是stdin/stdout，路径名则是/dev/tty），speed是设备的传输速率。local_addr和remote_addr以点分四段式给出链路两端的IP地址。我们的示例中，ip-up脚本中可能包含下面的代码片段：

```
#!/bin/sh
case $5 in
191.72.3.1)             # this is sub-etha
        route add -net 191.72.3.0 gw 191.72.3.1;;
esac
exit 0
```

类似地，在PPP链路再次取消之后，利用/etc/ppp/ip-down来撤消ip-up的所有动作。

但是，路由并没有因此而结束。我们在两个PPP主机上都设置了路由表，但迄今为止，这

两个网络上的其他主机对PPP链路还一无所知。如果处于次要地位的所有主机都有各自指向子以太网的默认路由，而且所有Brewery主机都可通过默认路由到达vlager，其他主机知不知道PPP链路都无关紧要。但如果不是上面提到的情况，通常只有选用 gated之类的路由daemon。在vlager主机上创建网络路由之后，路由 daemon就会向邻近子网上的所有主机广播这条新的网络路由。

7.7 链路控制选项

由上可知，我们已见识过LCP，它用于协商链接特性和测试链路。

LCP协商的最重要的两个选项是“最大接收单元”和“异步控制字符映射”。此外还有许多LCP配置选项，但和我们这里讨论的主题不太相干。有关详情参见 RFC-1548。

异步控制字符映射，一般称为async map，用于电话线之类的异步链路上，用来标识必须避开的控制字符（用一个特殊的两字符序列来替换它）。例如，有些配置不当的modem可能在收到XOFF后，就会阻塞，这种情况下，人们可能会想避开用于软件握手的 XON和XOFF字符。其他的还有Ctr+]（Telnet必须避开的字符）。在asyn map中指定从0到32的带有ASCII代码的所有字符后，PPP就能避开它们了。

asyn map是一个32位宽的位图，带有对应ASCII NUL字符的最无意义的位，和对应ASCII 31的最有意义的位。如果设置了一位，就表示在通过链路发送该位之前，必须避开它的对应字符。最初，asyn map被设置为0xffffffff，也就是说，将避开所有的控制字符。

要想告诉你的对等体，不必避开全部控制字符，只避开少数几个时，可利用 asyncmap选项，为pppd指定一个新的asyncmap。例如，如果必须避开的字符只有^S和^Q（ASCII 17和19，常用于XON和XOFF），采用下面的选项即可：

```
asyncmap 0x000A0000
```

“最大接收单元”（简称MRU），通知对等体我们希望收到的HDLC帧的最大字节数。尽管它令人想起MTU值（最大传输单元），但两者几乎没有共同之处。 MTU是内核连网设备参数，说明的是接口能够处理的帧的最大字节数。而 MRU更偏向于建议远程端不要生成大于 MRU值的帧；但接口必须能够接收的帧不得超过 1 500字节。

因此，对链路能够传输的帧选择一个 MRU是非常简单的，重要的是选择最适合自己的流通量。如果试图在链路上运行交互式应用程序，最好把 MRU设置为低于296字节的值，如此一来，在偶尔碰到一两个稍大一点的包（这里指的是 FTP会话）时，你的光标不会因此而“跳动不停”。要告诉pppd请求一个296字节的MRU值，就应该为它指定mru 296这个选项。但是，只有在没有取消VJ报头压缩时，小型MRU才有意义。

pppd也可识别两个LCP选项，这两个选项配置了协商进程的总体行为，比如链路中断之前，可以交换的最大配置请求数。除非你知道自己正在做什么，否则不应该改动它们。

最后，还有两个适用于LCP响应消息的选项。PPP定义了两条消息：响应请求和响应应答。pppd利用这一特性，检查一条链接是否仍处于运作过程中。可以利用 lcp-echo-interval选项和一个以秒计时的时间值来启用这一特性。如果在指定时间内，没有收到发自远程主机的帧，pppd会生成一个“响应请求”，并希望其对等体返回一个“响应应答”。如果其对等体没有作出应答，在发送特定数量的请求之后，这条链路就会中断。可利用 lcp-echo-failure选项，设置发送请求量。默认情况下，这个特性也是取消了的。

7.8 常规安全问题

一个配置不当的PPP daemon可能严重破坏你的网络。比如任何人都可把他们的机器插入你的以太网(这是最糟糕的)。本小节将讨论保证 PPP配置安全可靠的几个标准。

pppd存在的一个问题是配置网络设备和路由信息表时，要求拥有根特权。一般说来，这个问题的解决办法通常是运行 setuid root。

pppd允许用户设置大量不同安全方面的选项。为避免用户在计算这些选项之后，再启用它们，对你的网络进行恶意攻击，在此建议设置 global/etc/ppp/options文件（类似于示例文件）中的两个默认值。用户不能修改其中某些选项，比如身份验证选项。所以，从某种程度上来讲，也能提供一些安全保障。

当然，还必须对和你进行 PPP链接的系统进行保护。比如拥有对等体的某些身份验等等。另外，还不应该允许国外主机利用他们自己选择的 IP地址，如果允许的话，也应该为数不多。下一小节将就此详细讨论。

7.9 PPP身份验证

7.9.1 CHAP和PAP

利用PPP链路时，每个系统可能都要求其对等体采用下面两个协议之一来验明自己的身份。它们是：密码验证协议（ PAP）和盘问握手验证协议（ CHAP）。建立链接之时，链路两端都要求另一方验明各自的身份，无论是呼叫方，还是被呼叫方，都如此。在区别身份验证系统和身份验证者之前，先为大家谈谈“ 客户机 ”和“ 服务器 ”。对PPP daemon来说，通过发送标识需要哪个身份验证协议的另一个 LCP配置请求，便可要求其对等体进行身份验证。

PAP的运作基本上类似于普通登录过程。客户机向服务器发送用户名和一个（可以加密）密码，向服务器验明自己的身份，服务器将收到的用户名和密码和自己的机密数据库中保存的内容进行比较。这一技术有很大的漏洞，如果有人想获得密码的话，在该串行线路实行监听，就可以重复尝试对网络发起恶意攻击。

CHAP则不存在这些缺陷。使用 CHAP，身份验证者（也就是服务器）把自己的主机名和一个随机生成的“ 盘问 ”字串发送给客户机。客户机利用这个主机名查找相应的密钥，并把找到的密钥和盘问组合起来，用一个单向的散列函数对该盘问字串进行加密。其结果再随客户机的主机名一起，返回服务器。服务器再执行同样的计算，如果结果相同，就认可这个客户机。

pppd将CHAP和PAP所用的密钥保存在两个独立的文件内，分别是 /etc/ppp/chap-secrets和pap-secrets。在任何一个文件内输入一个远程主机名，均可很好地对用于自己与对等体之间的身份验证中的CHAP或PAP进行控制，反之亦然。

默认情况下， pppd不要求远程主机验明身份，但在远程主机发出验证请求时，它会同意验明自己的身份。由于 CHAP比PAP更为强壮，只要可能， pppd就会先尝试使用前一个协议。如果对等体不支持，或 pppd不能在自己的 chap-secrets文件内找到远程系统的 CHAP密钥，pppd就会转而使用PAP。如果同样没有其对等体的 PAP密钥，它就会拒绝进行身份验证。其结果是连接中断。

有几种方式可对上面的行为进行修改。例如，在收到 auth关键字时， pppd将要求其对等

体验明自身的身份。只要pppd的CHAP或PAP数据库内分别包含这个对等体的密钥，它就会同意使用CHAP或PAP。另外，还有别的选项，用于打开或关闭特殊身份验证协议，但在此不作讨论。详情参阅pppd手稿。

如果和你进行PPP链接的所有系统都同意验明自己的身份，就应该把auth选项放入global/etc/ppp/options文件内，并在chap-secrets文件中为每个系统定义密码。如果某个系统不支持CHAP，则在pap-secrets文件内为它增加一个密钥条目。通过这种方式，就可以保证没有通过身份验证的系统不能链接你的主机。

下面两个小节讨论两个PPP密钥文件：pap-secrets和chap-secrets。它们位于/etc/ppp下，其中包含许多三个（客户机、服务器和密码）一组的条目，有的后面还跟有IP地址清单。CHAP和PAP对客户机与服务器的解释是有区别的，与我们是向对等体验明身份，还是要求服务器向我们验明它们的身份有关。

7.9.2 CHAP密钥文件

在必须利用CHAP向某一服务器验明自身时，pppd将在pap-secrets文件内搜索带有客户机字段和服务器字段的条目。这里的客户机字段和服务器字段分别等同于CHAP盘问中的本地主机名和远程主机名。在要求对等体验明自身时，只有角色发生了变化：pppd查找带有客户机字段和服务器字段的条目，它们分别等同于客户机发出的CHAP响应中的远程主机名和本地主机名（注意，前后顺序发生了变化）。

下面是vlager的chap-secrets文件（双引号不属于密码，只是保护密码内的空格）：

```
# CHAP secrets for vlager.vbrew.com
#
# client          server            secret                addrs
#-----------------------------------
vlager.vbrew.com  c3po.lucas.com    "Use The Source Luke" vlager.vbr
c3po.lucas.com    vlager.vbrew.com  "riverrun, pasteve"   c3po.lucas
*                 vlager.vbrew.com  "VeryStupidPassword"  pub.vbrew.
```

在与c3po建立PPP链接时，c3po要求vlager利用CHAP，通过发送一个CHAP盘问的方式，验明自己的身份。然后，pppd在chap-secrets文件中查找客户机字段是vlager.vbrew.com，服务器字段是c3po.lucas.com（这个主机名选自CHAP盘问）的条目，并找到了上面列出的第一行。然后，pppd生成取自盘问字串和密钥（UseThe Source Luke）的CHAP应答，并把它发回c3po。

与此同时，pppd为c3po写一个CHAP盘问，其中包含一个独一无二的盘问字串及其完整资格主机名vlager.vbrew.com。c3po以刚才我们介绍的方式，构建了一个CHAP应答，并将该应答返回vlager。现在，pppd从应答中取出客户机主机名（c3po.vbrew.com），并在chap-secrets文件内搜索客户机为c3po，服务器为vlager的条目行。结果找到了第二行，所以pppd便把CHAP盘问和密钥riverrun与pasteve组合起来，并对它们进行加密，与c3po的CHAP应答中的结果进行比较。

仅供选择的第四个字段列出了一系列IP地址，这些地址都是第一个字段内的客户机能够接受的。它们可采用点分四段式，也可以是利用解析程序查找到的主机名。例如，IPCP协商过程中，如果c3po请求使用这个地址清单中没有的IP地址，请求就会遭到拒绝，而且IPCP也

会被关闭。在上面的示例文件中，限定 c3po只能使用自己的IP地址。如果地址字段为空，意味着所有的地址都不会遭到否决；“ - ”值可避免随客户机一起使用IP地址。

上面的chap-secrets示例文件中，第三行允许任何一台主机与 vlager建立PPP链接，因为客户机和服务器字段均为“ * ”（通配符）。唯一的要求是它们已知密钥并采用 pub.vbrew.com这一地址。带有通配主机名的条目可能出现在密钥文件中的任何地方，因为 pppd总是采用应用了服务器／客户机对的最具体的条目。

由上可知，远程主机名一直是由对等体在其发出的 CHAP盘问或应答包中提供的。默认情况下，本地主机名是调用 gethostname(2)函数产生的。如果已经把系统名设为你自己的未合格主机名，就必须另行利用domain（域）选项，为pppd提供域名。domain选项如下所示：

```
# pppd ...domain vbrew.com
```

这样， Brewery的域名便会增加到vlager，进行全部身份验证活动。修改 progpppd本地主机名的其他选项是 usehostname和name。在命令行上利用 local:varremote，指定本地IP地址，而且本地指的是本地主机名而不是点分四段表达式时， pppd就会把它用作本地主机名。有关详情，请参考pppd手册。

7.9.3 PAP密钥文件

PAP密钥文件的用法类似于CHAP文件。前两个字段总是包含用户名和服务器名；第三个字段保存的是PAP密钥。远程主机发出一个身份验证请求时， pppd便采用其服务器字段和用户名字段分别等同于请求中的本地主机名和用户名的条目。向其对等体验明自身后， pppd就从准备发送的行（其中的用户名字段和服务器字段分别等同于本地用户名和远程主机名）内选出密钥。

PAP密钥文件示例：

```
# /etc/ppp/pap-secrets
#
# user          server          secret          addrs
vlager-pap      c3po            cresspah1       vlager.vbrew.com
c3po            vlager          DonaldGNUth     c3po.lucas.com
```

第一行用于在与 c3po交流时，验明自己的身份。第二行描述了名为 c3po的用户是如何向我们验明它自己的身份的。

第一列中的vlager-pap名是我们发给c3po的用户名。默认情况下， pppd将把本地主机名用作其用户名，但也可通过user（用户）选项，后面跟上另一个主机名，另行指定。

从pap-secrets文件内选择用于向对等体验明身份的条目时， pppd必须知道远程主机名。由于它无法自行找出远程主机名，所以你必须在命令行上，利用 remotename（远程主机名）关键字指定它。例如，如果想采用上面的条目，向 c3po验明身份，必须在pppd的命令行增加下面的选项：

```
# pppd ...domain vbrew.com
```

第四个字段（以及随后的所有字段）中，可以像 CHAP密钥文件中那样，指定特定的主机能接受哪些IP地址。然后，对等体只需请求列出的地址。在示例文件中，我们要求 c3po使用它自己的真实IP地址。

注意，PAP是相当容易“受伤”的身份验证协议，所以我们建议大家尽可能使用 CHAP。这也是我们不打算深入讨论PAP的原因。如果想了解PAP的更多特性，可参考pppd手册。

7.10 PPP服务器的配置

把pppd当作服务器运行只是在命令行增加了恰当的选项而已。理想情况下，可创建一个特殊的账户，说一声ppp，并给它一个脚本或程序作为其登录外壳，外壳将利用这些选项调用pppd。例如，我们可在/etc/passwd内增加这一行：

ppp:*:500:200:Public PPP Account:/tmp:/etc/ppp/ppplogin

当然，大家可能还想采用不同于上面的uid和gid。这时别忘了用passwd命令，为上面的账户设置密码。

然后，ppplogin脚本就可能像下面这样：

```
#!/bin/sh
# ppplogin - script to fire up pppd on login
mesg n
stty -echo
exec pppd -detach silent modem crtscts
```

mesg命令禁止其他用户使用 write命令写入tty。stty命令关闭了字符响应。这是必要的，因为如果不这样，对等体发出的所有一切都会返回。上面给出的 pppd选项中，最重要的是-detach，因为它可防止pppd从控制tty中分离出去。如果我们不指定这个选项， pppd就会转入后台，令外壳脚本退出。这样将依次导致串行线路挂断，链接丢弃。 silent选项令pppd在开始发送包之前处于等待状态，直到它收到呼叫系统发来的一个包为止。这样可防止呼叫系统缓慢启动它的PPP客户机所产生的传输超时。 modem使pppd监控DTR线路，查看对等体是否已丢弃连接，crtscts则用于打开硬件握手。

除了上面提到的选项外，大家可能还想实施其他身份验证，比如在 pppd的命令行或全局选项文件内指定auth等等。如果想进一步了解打开和关闭个别的身份验证协议时所用的特定选项，可参考pppd手册。

第8章　各种网络应用

成功配置IP和解析器之后，必须转向准备提供的网络服务。本章将全面讨论如何配置简单的网络应用，其中包括inetd服务器和源于rlogin家族的几个程序。另外，还将为大家简要介绍用作“网络文件系统”（NFS）和“网络信息系统”（NIS）的“远程进程调用”。但是，由于NFS和NIS配置涉及到的内容非常广泛，我们将分别在独立的章节中就此进行讨论。它们也适用于电子邮件和netnews新闻组。

8.1　inetd超级服务器

通常情况下，服务都是由daemon来执行的。所谓daemon，是一种程序，用于打开特定的端口，并等待进入的连接。如果有连接接入，它就会创建一个用于接受该连接的子进程，父进程则继续监听别的连接请求。这种方式的缺点是：对提供的每项服务而言， daemon都必须运行并在端口上监听连接请求，这通常意味着某些系统资源（比如交换空间）的浪费。

因此，几乎所有的安装过程都会运行一个“超级服务器”，这个超级服务器针对大量的服务创建套接字，并利用select(2)系统调用，同步监听所有的服务。在远程主机请求其中一项服务时，这个超级服务器就会注意到这种情况，并为该端口生成特定服务器。

常用的超级服务器是inetd，即Internet Daemon。它是在系统启动时，开始运行的，它从一个名为/etc/inetd.conf的启动文件中，选出自己准备管理的服务列表。除了调用相关的服务器外， inetd本身还要调用内部服务所涉及的大量日常服务。其中包括“ chargen”（只生成一个字符串）和“ daytime”（返回系统日期和时间）。

这个文件中的条目只有一行，这一行由下列字段组成：

```
service type protocol wait user server cmdline
```

各字段的含义参见表8-1。

表8-1　字段定义

service	提供服务名。必须在/etc/service文件内查找这个服务名，并把它译为端口号。有关 /etc/services文件的详情，参见第10章。
type	指定套接字类型，要么是stream（用于面向连接的协议），要么是dgram（用于数据报传输协议）。因此，基于TCP的服务应该一直采用stream，而基于UDP的服务则应该一直采用dgram。
protocol	对服务所用的传输协议进行命名。这个字段值必须是一个能够在 protocols文件内找到的有效协议名，稍后将进一步讨论。
wait	该字段只用于dgram套接字。即可以是wait，也可以是nowait。如果指定的是wait，inetd在任何时候，对指定的端口，都只执行一个服务器。如若不然，它就会在执行服务器之后，立即返回端口，监听接入连接。这对单线程服务器来说是非常有用的。所谓单线程服务器，是单纯地读取所有进入的数据报，直到再已没有数据报进入时，才退出。多数 RPC（远程进程调用）服务器都是这种类型的服务器，因此多数情况下都应该指定为 wait。与单线程服务器相反的是多线程服务器，它允许同时运行多个实例进程，数额不限。这类服务器较少使用，如果有，应该指定为 nowait。stream套接字应该始终采用nowait。
user	这是用户的登录ID，进程就在这个ID下执行。其值通常应该是根用户，但有些服务可采用不同的账号。

(续)

server	给出准备执行的服务器之完美路径名。内部服务通常用 internal 关键字标注。
cmdline	准备传给服务器的命令行。其中包括参数0，也就是命令名。通常情况下，这是一个服务器程序名(除非该程序的行为有别于用另一个名字调用时)。如果是内部服务，这个字段就为空。

下面是一个/etc/inetd.conf文件示例。

清单8-1 /etc/inetd.conf文件示例

```
#
# inetd services
ftp         stream tcp nowait root   /usr/sbin/ftpd     in.ftpd -l
telnet      stream tcp nowait root   /usr/sbin/telnetd  in.telnetd -b/etc/issue
#finger      stream tcp nowait bin    /usr/sbin/fingerd  in.fingerd
#tftp        dgram  udp wait   nobody /usr/sbin/tftpd    in.tftpd
#tftp        dgram  udp wait   nobody /usr/sbin/tftpd    in.tftpd /boot/diskless
login       stream tcp nowait root   /usr/sbin/rlogind  in.rlogind
shell       stream tcp nowait root   /usr/sbin/rshd     in.rshd
exec        stream tcp nowait root   /usr/sbin/rexecd   in.rexecd
#
#       inetd internal services
#
daytime     stream tcp nowait root internal
daytime     dgram  udp nowait root internal
time        stream tcp nowait root internal
time        dgram  udp nowait root internal
echo        stream tcp nowait root internal
echo        dgram  udp nowait root internal
discard     stream tcp nowait root internal
discard     dgram  udp nowait root internal
chargen     stream tcp nowait root internal
chargen     dgram  udp nowait root internal
```

finger服务被标示出来，因此它是不可用的。这样做通常是出于安全方面的考虑，因为黑客常通过这一方式获得各个网络用户名。

tftp也被标示出来。tftp实行的是“原始文件传输协议”，该协议允许在不经过密码验证的情况下，把你的系统文件传播出去。这一点对/etc/passwd文件来说，危害性不言而喻。

无盘客户机和X终端常用tftp从启动服务器下载自己的代码。如果你由于这个原因，需要运行tftpd，务必保证将其限定在客户机将从中获取文件的目录内，这是通过将这些目录名加入tftpd命令行来完成的。显示在清单8-1中的第二个tftp行。

8.2 tcpd访问控制工具

由于通过计算机访问网络涉及到安全方面的问题，所以在设计应用程序时应该全盘考虑。但是，有些应用程序仍然存在安全方面的不足（以RTM Internet蠕虫的表现尤为突出），或不能区分安全主机（该类主机请求的特定服务将被接受）和非安全主机（其请求将被拒绝）。我们前面已谈过tftp和finger服务，因此，应该把对这些服务的访问限定于“信任主机”，常规设置是不可能做到这一点的，这种情况下，inetd要么将该服务提供给所有客户机，要么一个也

不提供。

另一个非常有用的访问控制工具是 tcpd，该工具是 Wietse Venema（邮件地址：wietse@wzv.win.tue.nl）编写的，是一个所谓的 daemon封装器。如果你打算监控或保护的是TCP服务，就可以弃服务器程序而调用它。 tcpd将请求记入系统日志 daemon，查看是否允许远程主机使用该项服务，而且只有得到肯定的答复之后，才执行真正的服务器程序。注意，这一点不适用于基于UDP的服务。

例如，要封装finger daemon，必须将inetd.conf中的相应行改为

```
# wrap finger daemon
finger  stream  tcp  nowait  root  /usr/sbin/tcpdin.fingerd
```

没有增加任何的访问控制，除了所有请求都被记入系统日志的授权设备外，客户机上进行了一个常规的finger设置。

访问控制的实施是通过两个文件来实现的，它们是 /etc/hosts.allow和/etc/hosts.deny。这两个文件中分别包含接受和拒绝访问特定服务和主机的条目。 tcpd从一个名为biff.foobar.com的客户机主机，处理finger之类的服务请求时，它将在 hosts.deny和hosts.allow这两个文件中查看与该服务和客户机主机相符的条目。如果在 hosts.allow中找到了与请求的客户机主机相符的条目，就认可访问权，不管 hosts.deny内的情况如何。如果在 hosts.deny中找到与请求服务相符的条目，这个请求就会被拒绝，其方式是取消连接。反之，则接受服务请求。

访问文件中的条目是这样的：

```
servicelist: hostlist [:shellcmd]
```

servicelist和hostlist的定义如表8-2所示。

表8-2 servicelist和hostlist的定义

servicelist	这是取自/etc/services或关键字ALL的服务列表。要使之与除了finger和tftp之外的所有服务对应，利用“ ALL EXCEPT finger tftp ”即可
hostlist	这是主机名或IP地址、或ALL、LOCAL或UNKNOWN这几个关键字的列表。 ALL对应任何一台主机， LOCAL对应不包含句点的主机名（一般说来，只有通过查找 /etc/hosts得来的本地主机名中才不包含句点）。UNKNOWN对应任何一台无法找到其地址的主机。以句点开头的主机名字串对应所有其域等同于这个名字串的主机。例如， .foobar.com对应的是biff.foobar.com。另外，还为IP网络地址和子网编号准备了相应的关键字，详情参见 hosts_access手册

要拒绝除了本地主机以外的机器访问 finger和tftp服务，将下面一行放入/etc/hosts.deny内，让/etc/hosts.allow文件为空。

```
in.tftpd.  in.fingerd: ALL EXCEPT LOCAL, .your.domain
```

仅供选择的shellcmd字段中可以包含一个shell命令，这个命令将在已找到匹配请求条件的条目时调用。最好布置一些小把戏，引诱黑客上钩：

```
in.ftpd: ALL EXCEPT LOCAL, .vbrew.com :
echo "request from %d@%h"  /var/log/finger.log;
if [ %h != "vlager.vbrew.com" ]; then
    finger -l @%h  /var/log/finger.log
fi
```

%h和%d参数是tcpd派生出来的，分别代表客户机主机名和服务名。详情参阅 hosts_access手册。

8.3 服务和协议文件

对提供特定“标准”服务的端口来说，其端口号是在“已分配编号” RFC中定义的。要将服务器和客户机程序用于服务名到这些端号之间的转换，每台主机上至少得保留一份服务编号对应列表，该列表保存在一个名为 /etc/services的文件内。其中的条目结构如下：

```
service port/protocol   [aliases]
```

这里，service指的是服务名，port定义的是上面准备提供服务的端口，而 protocol定义的是准备采用的传输协议。通常，不是 udp，就是tcp。为一项服务提供多个协议是行得通的，同样，在同一个端口上提供不同的服务也是可行的，只要各项服务采用的协议不同。 aliases字段允许为同一项服务指定备用名。

一般说来，没必要更改随网络软件一起安装在系统中的 services文件。虽然如此，我们还是从中摘录了一小段代码，如下所示：

```
# The services file:
#
# well-known services
echo            7/tcp                   # Echo
echo            7/udp                   #
discard         9/tcp   sink null       # Discard
discard         9/udp   sink null       #
daytime        13/tcp                   # Daytime
daytime        13/udp                   #
chargen        19/tcp   ttytst source   # Character Generator
chargen        19/udp   ttytst source   #
ftp-data       20/tcp                   # File Transfer Protocol (Data)
ftp            21/tcp                   # File Transfer Protocol (Contr
telnet         23/tcp                   # Virtual Terminal Protocol
smtp           25/tcp                   # Simple Mail Transfer Protocol
nntp          119/tcp   readnews        # Network News Transfer Protoco
#
# UNIX services
exec          512/tcp                   # BSD rexecd
biff          512/udp   comsat          # mail notification
login         513/tcp                   # remote login
who           513/udp   whod            # remote who and uptime
shell         514/tcp   cmd             # remote command, no passwd use
syslog        514/udp                   # remote system logging
printer       515/tcp   spooler         # remote print spooling
route         520/udp   router routed   # routing information protocol
```

注意，echo服务是端口7提供的，采用的协议是 TCP和UDP，而端口512用于两项不同服务，即UDP上的COMSAT daemon（通知最近收到邮件的用户，参见 xbiff(1x)）和利用TCP的远程执行（rexec(1)）。

与services文件类似，networking库也需要一种方法将协议名（比如 services文件中所用的那些）翻译为协议编号，以便其他主机上的 IP层能够识别。这是通过在 /etc/protocols文件中查找协议名的方式得来的。该文件中每行包含一个条目，每个条目中包含一个协议名以及关联的协议编号。其示例文件如下：

```
ip    0   IP      # internet protocol, pseudo protocol number
icmp  1   ICMP    # internet control message protocol
igmp  2   IGMP    # internet group multicast protocol
tcp   6   TCP     # transmission control protocol
pup   12  PUP     # PARC universal packet protocol
udp   17  UDP     # user datagram protocol
idp   22  IDP     # WhatThis?
raw   255 RAW     # RAW IP interface
```

8.4 远程过程调用

对客户机-服务器应用程序来说，它们所用的常见机制是“远程进程调用”（Remote Procedure，Call RPC）包提供的。RPC是有Sun微系统公司开发的，是一个工具和库函数集。内置于RPC包之上的重要应用程序分别是网络文件系统 (NFS)和网络信息系统(NIS)，两者将在后续章节中讨论。

RPC服务器由一系列进程和进程参数组成，客户机可通过向服务器发送 RPC请求的方式，调用这些进程。服务器将调用客户机这一端指定的进程，如果有返回值的话，再将它们传回客户机。为了做到数据的传送与机器无关，客户机和服务器之间交换的所有数据都将由发送端将它们转换为一种所谓的“外部数据表示法”格式，再由接收端把数据转换成本地表示法格式。

有时，对RPC应用程序的改进往往与进程调用接口内的变动不相容。当然，只更改服务器将导致仍然在原地踏步的所有应用程序崩溃。因此， RPC程序为接口分配了不同版本号，一般从1开始，版本越新，这个数就越大。通常，服务器可以同时提供好几个不同版本；客户机根据服务请求中的版本号，作出应答。

RPC服务器和客户机之间的网络通信有些特殊。 RPC服务器提供一个或多个进程集；每个进程集就叫做一个程序，而且有一个唯一的程序编号。将服务名映射为相应程序编号的列表通常保存在/etc/rpc文件内，下面是摘自这个文件的片段：

```
#ident    "@(#) rpc          1.11          95/07/14 SMI"          /* SVr4.0 1.2
*/
#
#          rpc
#
rpcbind             100000   portmap sunrpc rpcbind
rstatd              100001   rstat rup perfmeter
rusersd             100002   rusers
nfs                 100003   nfsprog
ypserv              100004   ypprog
mountd              100005   mount showmount
ypbind              100007
walld               100008   rwall shutdown
yppasswdd           100009   yppasswd
etherstatd          100010   etherstat
rquotad             100011   rquotaprog quota rquota
sprayd              100012   spray
3270_mapper         100013
rje_mapper          100014
selection_svc       100015   selnsvc
```

```
database_svc        100016
rexd                100017    rex
alis                100018
sched               100019
llockmgr            100020
nlockmgr            100021
x25.inr             100022
statmon             100023
status              100024
ypupdated           100028    ypupdate
keyserv             100029    keyserver
bootparam           100026
sunlink_mapper      100033
tfsd                100037
nsed                100038
nsemntd             100039
showfhd             100043    showfh
ioadmd              100055    rpc.ioadmd
NETlicense          100062
sunisamd            100065
debug_svc           100066    dbsrv
ypxfrd              100069    rpc.ypxfrd
bugtraqd            100071
kerbd               100078
event               100101    na.event      # SunNet Manager
logger              100102    na.logger     # SunNet Manager
sync                100104    na.sync
hostperf            100107    na.hostperf
activity            100109    na.activity   # SunNet Manager
hostmem             100112    na.hostmem
sample              100113    na.sample
x25                 100114    na.x25
ping                100115    na.ping
rpcnfs              100116    na.rpcnfs
hostif              100117    na.hostif
etherif             100118    na.etherif
iproutes            100120    na.iproutes
layers              100121    na.layers
snmp                100122    na.snmp snmp-cmc snmp-synoptics snmp-unisys snmp-
utk
traffic             100123    na.traffic
nfs_acl             100227
sadmind             100232
nisd                100300    rpc.nisd
nispasswd           100303    rpc.nispasswdd
ufsd                100233    ufsd
pcnfsd              150001
amd                 300019    amq
bwnfsd              545580417
fypxfrd             600100069 freebsd-ypxfrd
```

TCP/IP网络中，RPC的作者面临着如何将程序编号映射为常见的网络服务这一难题。他们的解决办法是：让每台服务器为每个数据报和每个版本同时提供 TCP端口和UDP端口。一

般说来，RPC应用程序将在发送数据时，采用 UDP，而在传输的数据不能装入一个单一的UDP数据报时，转而采用TCP。

当然，客户机程序必须有一种方法，能够找出程序编号对应的是哪个端口。采用配置文件未免太死板，由于RPC应用程序不采用保留端口，因此无法保证原本打算供我们的数据库应用程序所用的端口没有被别的进程占用。鉴于此，RPC应用程序选出了自己能够得到的端口，并将其注册为所谓的portmapper daemon（端口映射器程序）。后者充当的是所有运行于这台机器上的RPC服务器的“经纪人”：对希望将服务和指定程序编号联系起来的客户机来说，先在服务器的主机上查询端口映射器，后者将返回请求服务能够抵达的 TCP和UDP端口。

上面这种方法有个明显的不足，比标准 Berkeley服务所用的inetd daemon更甚。因为在端口映射器失效时，所有的 RPC端口信息也会丢失；这通常意味着你不得不手工启动所有 RPC服务器或重启整台机器。

端口映射器名为rpc.portmap，驻留于/usr/sbin中。

注意 对Red Hat 6来说，端口映射器位于/sbin/portmap。

8.5 r 命令的配置

可在远程主机上执行的命令有许多。它们是： rlogin、rsh、rcp和rcmd。它们在远程主机上创建一个外壳，允许用户执行这些命令。当然，客户机需要拥有相应的主机账号，以便能在这些远程主机上执行命令。所以，所有这些命令都要执行一个身份验证进程。一般情况下，客户机将用户的登录名告诉服务器，服务器再依次请求以常规方式验证的密码。

```
#
# /etc/rpc - miscellaneous RPC-based services
#
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rstat svc rup perfmeter
rusersd         100002  rusers
nfs             100003  nfsprog
ypserv          100004  ypprog
mountd          100005  mount showmount
ypbind          100007
walld           100008  rwall shutdown
yppasswdd       100009  yppasswd
bootparam       100026
ypupdated       100028  ypupdate
```

但有些时候，需要对特定的用户发放免检通行证。比如，如果你需要频频登录到局域网内的其他机器，你肯定希望免去每次的键入密码之苦。

我们建议取消身份验证只限于其密码数据库同步更新的少数主机，或少数特权用户（由于管理方面的原因，他们需要访问多台计算机）。只要你想允许人们未指定任何一个登录 ID或密码，就进入你的主机，一定要先保证不会偶然将访问权授予任何人！

对r命令来说，取消身份验证的方法有两种。其一是针对超级用户而言，是允许特定或所有主机上的特定或所有用户（毫无疑问，“所有”将是非常糟糕的选择）在不要求提供密码的情况下，登录进来。这种访问是由一个名为 /etc/hosts.equiv的文件控制的。该文件中包含一系列主机和用户名，这里的主机名和用户名被视为与本地主机上的用户等同。其二是针对普通

用户而言，授权特定主机上的其他用户可以访问自己的账号。这种访问控制包含在用户主目录的file.rhosts文件中。出于安全方面的考虑，该文件必须为用户或超级用户专有，而且不能是象征性的链接，否则就是无效的（在 NFS环境中，可能还需要为它提供 444的保护级，因为超级用户只能通过NFS访问磁盘上的文件）。

客户机请求r服务时，将先后在/etc/hosts.equiv和.rhosts文件内查找准备用来登录的主机和用户名。比如，Janet在Gauss这台机器上工作，想登录到 Joe在Euler主机上的账号。下面的示例将以Janet作为客户机用户，Joe作为本地用户为例。现在，Janet键入

```
$ rlogin -1 joe euler
```

在Gauss上，如果Janet被授予自由访问的话，服务器将先搜索 hosts.equiv（注意，在有人试图以根的身份登录时，hosts.equiv文件是不能搜索的），如果搜索失败，它就会试着在 Joe的根目录内的.rhosts文件中查找Janet。

Euler上的hosts.equiv文件如下所示：

```
gauss
euler
-public
quark.physics.groucho.edu      andres
```

该文件中的一个条目由一个主机名构成，用户名是可选的。如果主机名自行显示出所有的用户，该主机的所有用户都得以允许在无须验证的情况下，进入他们的本地账号。在上面的示例中，Janet得以允许从Gauss登录到她自己的账号，除了根之外的其他用户也如此。但如果Janet想以Joe的身份登录，就会像往常一样，提示她输入密码。

如果主机名后面跟有用户名，就像示例文件中的最后一行那样，该用户就可以无须任何验证，自由地访问除了根账号外的所有账号。

主机名前面还可以加上一个负号－，例如“ － public ”。它要求公开对所有的账号进行身份验证，不管用户在自己的 .rhosts文件内的授权如何。

.rhosts文件的结构和 hosts.equiv完全相同，但其含义有些差别。比如 Euler机器上，Joe的.rhosts文件是这样的；

```
chomp.cs.groucho.edu
gauss      janet
```

第一个条目授权Joe可以从chomp.cs.groucho.edu自由访问所有的账号，但该条目并没有影响Euler或Chomp上的其他账号。第二个条目是前者的变体，它授权 Janet可以从Gauss自由访问Joe的账号。

注意，客户机的主机名是通过将呼叫方的地址逆向映射为主机名这一方式获得的。所以如果解析器不知道主机的话，就无法使用这一特性。在下面这两种情况下，客户机的主机名被认为对应于hosts文件内的主机名：

- 从字面上来讲，客户机的规范主机名（而不是别名）对应于文件内的主机名。
- 如果客户机的主机名是一个完整资格域名（比如是在运行 DNS时，解析器返回的），从字面上来讲，它不对应hosts文件内的主机名，它利用本地域名，对主机名进行了扩展。

注意 勿庸置疑，启用r命令实在不是个好主意。相比而言，为了更好地保护自己的数据，应该采用非常严格的SSH加密和用户身份验证软件。令人遗憾的是，由于美国政府的保密策略，Linux分销商不能将SSH之类的优秀软件包含在他们的光盘内。

第9章　网络信息系统

在运行一个局域网时，你的主要任务通常是为你的用户提供一个友好、透明的网络环境。最重要的一步是实时保存各台主机的用户账号信息之类的大量数据。在接触主机名解析之前，我们还介绍过一个强大而复杂的服务，那就是 DNS。对其他任务来说，还没有这类特殊服务。而且，如果管理的只是一个没有接入因特网的小型局域网，对许多管理员来说，都觉得没必要费太多脑筋去设置DNS。

这就是Sun微系统开发NIS（网络信息系统）的原因。 NIS提供了常见的数据库访问设备，可用于将信息分发到网络上的各台主机，这些信息原本包含在 passwd和groups文件中。这样一来，整个网络就像一个独立的系统一样，所有主机的账号都是一样的。以类似的方式，还可使用NIS将主机名信息从 /etc/hosts分发给网络上的各台主机。

NIS是建立在RPC基础上的，由一个服务器、一个客户端库和若干个管理工具构成。起初，NIS叫做“ Yellow Page ”（黄页）或YP，至今，这个称呼仍然非常普遍。另一方面，“ Yellow Page ”是英国电信的注册商标，他们要求Sun放弃这个名字。但人们仍然记得最初的这个名字，YP仍然作为多数NIS命令名的前缀广为流传，比如ypserv、ypbind等等。

如今，有一些免费的NIS实施方案。其中之一来自BSD Net-2，衍生于Sun免费发放的公用域参考实施方案。长期以来，它发布的库客户机程序一直包含在 GNUlibc内，而管理程序近来才水落石出，是Swen Th ü mmler移植过来的（邮件地址 swen@uni-paderborn.de。这些NIS客户机程序可用作yp-linux.tar.gz，后者源于system/Network内的sunsite.unc.edu）。NIS服务器没有包括在参考实施方案内。 Tobias Rebert编写了另一个NIS包，其中包含所有的工具和一个服务器；该NIS包名为yps（当前版本是yps-0.21，可从/pub/NYS目录下的ftp.lysator.liu.se获得）。

目前，完全重写NIS代码的NYS代码正由Peter Eriksson（邮件地址pen@lysator.liu.se）负责编写，它将同时提供对普通 NIS和Sun修订过若干次的NIS+的支持。 NYS不止提供一个NIS工具集和一个服务器，还将增加一个全新的库函数集，后者最终将 NIS变成一个标准的libc。NYS中包括了一个用于主机名解析新配置方案，将利用 host.conf替换当前采用的方案。其中各个函数的特性将在随后的小节中讨论。

本章的重点在于NYS，而不是另外传统意义上的NIS程序包。如果想运行这些包，本章内容并不充分。要想获得更多的详情，可参考一本关于 NIS的书，比如Hal Stern 所著的《NFS和NIS》，或者查看 howto文件，该文件位于 www.suse.de/~kukuk/linux/HOWTO/NIS-HOWTO.html。

现在，NYS仍处于开发阶段，所以诸如网络程序或登录程序之类的标准实用程序还不能识别NYS配置方案。直到有一天， NYS揉合到主流libc中，你希望自己所有的二进制程序使用NYS配置方案时，才有必要对这些二进制程序进行重新编译。这类程序的 Makefiles会将libc前的最后一个选项-lnsl指定为连接程序。这样便链接到 libnsl的相关函数和NYS库，而不是标准的C语言库。

9.1 NIS概述

NIS将数据库信息保存在自己的所谓映射表内，该映射表内包含的是成对出现的关键字 - 值。映射信息保存在运行NIS服务器的中央主机内。客户机可以通过各种RPC调用，获取中央主机内的信息。一般说来，映射是保存在 DBM文件内的（DBM是一个简单的数据库管理库，利用散列技术加快搜索操作。 GNU工程组曾发布一个免费的 DBM实施，名为gdbm，许多Linux厂商都将其包含在自己的产品内）。

映射本身一般源于主管文本文件，比如/etc/hosts和/etc/passwd。对于有的文件，可创建好几个映射，一个映射代表一类搜索关键字。例如，可以在搜索 IP地址的同时，在hosts文件内搜索主机名。相应地，就会从这个 hosts文件内创建两个 NIS映射，分别是hosts.byname和hosts.byaddr。

在某些NIS包或其他地方，还可找到相应的其他文件和映射。其中包含的应用程序信息本书没有讨论，比如某些BOOTP服务器所用的bootparams映射，或当前还没有排上用场的映射，如ethers.byname和ethers.byaddr映射。

对于有些映射，人们常采用 nickname（绰号）来表示它们，这是因为绰号更短，更方便键入。要想得到一份你的NIS工具能识别的绰号列表（以Red Hat 6为例），运行下面的命令即可：

```
$ cat /var/yp/nickname
```

过去， NIS服务器被称为ypserv。对一个中等大小的网络来说，一台服务器就能够应付了；大型网络可能会在不同的机器上，不同的网络分段上运行若干个服务器，减轻服务器上路由器的负荷。这类服务器的同步更新是通过令其中一台服务器为主管（ master）服务器，其他为从属（slave）服务器来完成的。只有主管服务器主机才创建映射。建好之后，再分发到所有的从属服务器。

大家将注意到，我们一直在谈的“ 网络 ”一词非常含糊；诚然， NIS中，这一词指的是一个通过NIS，共享系统配置数据的所有主机的集合：即 NIS域。但令人遗憾的是， NIS域和我们在DNS域内见过的东西没什么两样。为尽可能使本章明晰可辨，我将着重指出我所指的域究竟是什么。

NIS域只有一个纯管理功能。除了该域内的所有主机共享密码之外，多数时候，用户都不能看到该功能的体现。因此，为NIS域指定的域名只和管理员有关。一般情况下，任何名字都是允许的，只要不和本地网络内的其他 NIS域同名就行。例如， Virtual Brewery网络管理员创建了两个NIS域，一个供Brewery自己用，一个供Winery用，她将它们分别命名为 brewery和winery。另一个相当常见的方案是简单地利用 DNS域名来代表NIS域名。要想设定和显示主机的NIS域名，可利用domainname命令。在不带参数的情况下，它的调用结果是当前的 NIS域名；要想设置域名，必须先成为超级用户，然后再键入：

```
# domainname <YOURNISDOMAIN>
```

NIS域判断应用程序将查询哪个 NIS服务器。比如，以Winery上的一台主机为例，它的登录程序只能查询Winery的NIS服务器（如果该网络有若干台服务器的话，则是其中之一），要求得到用户的密码信息；Brewery上的主机同样如此，只能查询自己的服务器。

还有一点疑问有待解决，那就是客户机如何查找准备与之连接的服务器。最简单的方法

是能够有一个配置文件，其中将客户机指定与某台服务器连接。但这种方法非常不灵活，因为它不允许客户机使用别的服务器（当然指的是源于同一个域的那些）。因此，传统的NIS实施依赖于一个名为ypbind的特殊daemon（程序），通过它，在NIS域内侦测合适的NIS服务器。在能够执行任何 NIS查询之前，所有应用程序都应该先从 ypbind中找出准备使用的服务器。ypbind通过向本地IP网络广播查询，查出自己需要的服务器，第一个应答的服务器将被定为速度最快的，并将用于后续的所有 NIS查询中。特定时间间隔之后，或服务器不能使用之时，ypbind将再次查找活动的服务器。

现在，关于动态绑定，其有待考证的是其用途少，而且还可能引发安全问题： ypbind盲目地相信任何人的回答，无论对方是低级的 NIS服务器，还是心怀不轨的入侵者。需要提醒大家注意的是，如果你通过 NIS管理自己的密码数据库，这种方法会为你带来许多麻烦。因此，默认情况下，NYS是不会采用ypbind的，而是从配置文件内选出服务器的主机名。

9.2 NIS与NIS+之比较

除了名字和常见用法相同外，很难再找出 NIS和NIS+的相同点了。NIS+采用了另一种截然不同的结构。原先带有分散NIS域的单调的域名空间，被一个类似于 DNS的分层式域名空间代替。原先的映射，被所谓的表格代替，这类表由列和行组成，一行代表 NIS+数据库内的一个对象，而列代表NIS+已知的并关心的对象之属性。针对具体NIS+域的表由其父域的表组成。另外，表中的条目还可能包含一条指向另一个表的链接。这些特性使得我们可以采用不同的方式构建信息。

传统NIS的RPC版本号是2，NIS+则是3。

目前，NIS+的使用似乎不太广泛，我本人对它也知之甚少（坦白说是一无所知）。鉴于此，我们不打算对它进行讨论，如果你对它有兴趣，不妨去看看它的 howto文档，可在www.suse.de/~kukuk/linux/HOWTO/NIS/NIS-HOWTO.html找到它。

9.3 NIS的客户端

如果你熟谙网络应用程序的编写和移植，肯定会注意到上面列出的许多 NIS映射都对应于C语言库内的库函数。例如，要想获得 passwd信息，一般会采用getpwnam(3)和getpwuid(3)这两个函数，它们将分别返回与具体用户名或数字化的用户 ID关联的账号信息。正常情况下，这些函数将对标准文件，比如/etc/passwd执行所要求的查找。

但NIS实施将修改这些函数的行为，并任命 RPC调用让NIS服务器来查找用户名或ID。这个过程对应用程序来说，是完全透明的。这个函数有两种可能：要么绑定在 NIS映射上，要么用原始文件来“替换”这个映射。当然，这里的修改，并不是指真正修改了文件，只是针对应用程序而言，就像这个文件真的已被替换或绑定。

就传统NIS实施来说，哪些映射应该替换，哪些又应该绑定到原始信息，过去曾有相关的约定。有的映射，比如 passwd，要求对passwd文件稍作修改，以免该文件一旦出错，就会导致安全漏洞。为避免这类情况， NYS采用了一个常规配置方案，用于判断特定的客户机函数集是否使用的是原始文件、 NIS还是NIS+，使用顺序又是什么样的。有关详情，参见本章最后一节。

9.4 NIS服务器的运行

吹了那么多理论上的技术泡泡，现在该动手实际配置了。本小节将全面讨论 NIS服务器的配置。如果你所在的网络上已经有一台 NIS服务器，不必再自行设置；最好跳过本小节。

注意 如果只是想体验一下配置服务器，也不能对网络中正在运行的NIS域名动手动脚。因为这样可能瓦解整个网络服务，令许多人感到不快，甚至愤怒。

目前，有两个NIS服务器可用于Linux，一个包含在Tobias Reber的yps包内，另一个包含在Peter Eriksson的ypserv包内。不管你使用的是 NYS还是当前libc内的标准NIS客户机代码，其运行结果都大同小异。在本书完稿时， yps中的用于处理NIS从属服务器的程序也更为完整。所以，在不得不处理从属服务器时， yps便是你的首选。

下一小节将解释如何配置 NIS客户机代码。如果你的设置无效，应该试着查找请求是否已抵达你的服务器。如果将 -D命令行标记指定为 NYS服务器，它就会在控制台打印出关于所有进入NIS查询的测试信息并返回结果。你也可从中了解问题出在哪里。

9.5 用NYS设置一个NIS客户机

从现在开始，我们将全面介绍 NIS客户机的配置。

第一步，应该在 /etc/yp.conf配置文件内，设置服务器，从而告诉 NYS，你准备将这台服务器用于NIS服务。以Winerey网络上的一台主机为例，它的示范配置文件如下所示：

```
# /etc/yp.conf - ypbind configuration file
# Valid entries are
#
#domain NISDOMAIN server HOSTNAME
#           Use server HOSTNAME for the domain NISDOMAIN.
#
#domain NISDOMAIN broadcast
#           Use broadcast on the local net for domain NISDOMAIN
#
#ypserver HOSTNAME
#           Use server HOSTNAME for the local domain.
#           The IP-address of server must be listed in /etc/hosts.
```

第一个语句告示所有的 NIS客户机，它们属于 winery NIS域。如果省略这一行， NYS就会采用你通过domainname命令为自己的系统所分配的域名。 server语句命名准备使用的 NIS服务器。当然，还必须在 hosts文件内，设置与 vbardolino对应的hosts的IP地址；或者采用 server语句自带的IP地址。

上面的示例中， server命令要求 NYS采用named服务器，不管当前的 NIS域是什么。但是，如果你的机器经常性地在不同的 NIS域间游移，肯定想将各个域的相关信息统统保存在 yp.conf文件内。将你需要的 NIS域名增加到 serer语句内， yp.conf文件中便能拥有若干个 NIS域的相关信息。

创建好基本的配置文件，并保证全世界的人都能读懂它之后，就应该开始第一轮的测试，看你是否能连接到自己的服务器。务必保证选择你的服务器分发的映射，比如 hosts.byname映射，并试着通过 ypcat实用程序获取这些映射。与其他管理性的 NIS工具一样， ypcat应该保存

在/usr/sbin内。

此时，你得到的输出应该和上面的示例类似。如果出现类似的错误消息：“不能绑定充当域的服务器”（Can’t bind to server which serves domain），意味着不是你设置的NIS域名之匹配服务器尚未在yp.conf文件内定义，就是由于某种原因，该服务器不能抵达。后一种情况中，要保证发给主机的 ping会产生一个肯定结果，而该主机的确正在运行一个 NIS服务器。利用rpcinfo，可对后一种情况进行验证，它将产生表 9-1那样的输出。

表9-1 NIS配置的验证输出

程序	vers	协议	端口	
100000	4	tcp	111	rpcbind
100000	3	tcp	111	rpcbind
100000	2	tcp	111	rpcbind
100000	4	udp	111	rpcbind
100000	3	udp	111	rpcbind
100000	2	udp	111	rpcbind
100024	1	udp	32772	status
100024	1	tcp	32771	status
100133	1	udp	32772	
100133	1	tcp	32771	
100021	1	udp	4045	nlockmgr
100021	2	udp	4045	nlockmgr
100021	3	udp	4045	nlockmgr
100021	4	udp	4045	nlockmgr
100012	1	udp	32773	sprayd
100001	2	udp	32774	rstatd
100001	3	udp	32774	rstatd
100001	4	udp	32774	rstatd
100221	1	tcp	32772	
100235	1	tcp	32773	
100068	2	udp	32775	
100068	3	udp	32775	
100068	4	udp	32775	
100068	5	udp	32775	
100083	1	tcp	32774	
100021	1	tcp	4045	nlockmgr
100021	2	tcp	4045	nlockmgr
100021	3	tcp	4045	nlockmgr
100021	4	tcp	4045	nlockmgr
805502976	2	tcp	911	
805502976	1	tcp	912	
300598	1	udp	32796	
300598	1	tcp	32782	
805306368	1	udp	32796	
805306368	1	tcp	32782	
100249	1	udp	32799	
100249	1	tcp	32783	

9.6 挑选合适的映射

确定自己能够抵达特定的NIS服务器后，必须决定用NIS映射来替换或增加配置文件。通常，大家会想到主机和密码查找函数所用的 NIS映射。如果不运行BIND的话，前者显得身手不凡。后者允许所有的用户从这个NIS域的任何系统，登录到他们的账号；这通常要求所有的主机通过NFS，共享一个certral/home目录。有关详情将留在稍后讨论。其他映射，比如services.byname，就没有这么大的本事了，但可以为你省些编辑工夫，前提是你安装的网络应用采用的服务名是标准services文件内没有的。

一般说来，在查找（lookup）函数采用local文件，并查询NIS服务器时，人们都想能有丰富的选项！NYS允许大家对函数访问服务的顺序进行配置。这是通过 /etc/nsswitch.conf文件来控制的，该文件代表的是“域名服务开关”，当然，它并不局限于域名服务。针对 NYS支持的所有数据查找函数，它含有一行，对准备采用服务进行命名。

服务访问顺序和数据的类型有关。 services.byname映射不可能包含能够把本地 services文件内的条目区分开来的条目，它只能包含尽可能多的条目。所以，最好先查询本地文件，如果未找到服务名，再查看 NIS。另一方面，主机名信息可能会频繁变动，所以 DNS或NIS服务器应该始终保持最新、最准确的账号，而本地 hosts文件只能充当DNS或NIS不能使用时的替补对员。这种情况下，只好查看本地文件了。

目前， NYS支持的nsswitch.conf条目有：hosts、networks、passwd、group、shadow、gshadow services、protocols、rpc和ethers。还可能添加更多的条目。

清单9-1向大家展示了一个比较复杂的示例，引入了 nsswitch.conf的另一个特性：hosts条目中的[NOTFOUND=return]关键字要求NYS在没有在NIS或DNS数据库内找到所需要的项目时返回。也就是说，只有对 NYS和DNS服务器的调用由于某种原因而失败时， NYS才会继续在本地文件内查找所需项目。 NIS服务器关闭后再启动时，将本地文件当作备份使用。

清单9-1 域名服务开关配置文件示例

```
#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.
#
#The entry '[NOTFOUND=return]' means that the search for an
#entry should stop if the search in the previous entry turned
#up nothing. Note that if the search failed due to some other reason
#(like no NIS server responding) then the search continues with the
#next entry.
#
#Legal entries are:
#
#       nisplus or nis+ Use NIS+ (NIS version 3)
#       nis or yp Use NIS (NIS version 2), also called YP
#       dns Use DNS (Domain Name Service)
#       files Use the local files
#       [NOTFOUND=return] Stop searching if not found so far
#
```

```
passwd:     files nisplus nis
shadow:     files nisplus nis
group:      files nisplus nis

hosts:      files nisplus nis dns

services:   nisplus [NOTFOUND=return] files
networks:   nisplus [NOTFOUND=return] files
protocols:  nisplus [NOTFOUND=return] files
rpc:        nisplus [NOTFOUND=return] files
ethers:     nisplus [NOTFOUND=return] files
netmasks:   nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files

netgroup:   nisplus

publickey:  nisplus

automount:  files nisplus
aliases:    files nisplus
```

9.7 使用passwd和group映射

NIS的主要应用之一是同步更新NIS域内的所有主机的相关用户和账号信息。鉴于此，一般都要保留一个小型的本地/etc/passwd文件，NIS映射的站点级信息将绑定在一个文件内。但是，只将NIS查找用于nsswitch.conf内的这项服务是远远不够的。

在依赖NIS分发的密码信息时，首先必须保证：你自己的本地passwd文件内任何一个用户的数字化ID与NIS服务器中的用户ID匹配。当然，你肯定还想过把它用于别的用途，比如从网络内的其他主机装入NIS卷。

如果/etc/passwd或/etc/group内的数字化ID不包含在映射中，就必须对属于特定用户的所有文件之所属关系进行调整。首先，将passwd和group内的所有uid和gid改成新值，然后，再找出属于新用户的所有文件，最后，再更改这些文件的所属关系。假设news过去的用户ID是9，而okir的用户ID是103，它们的用户ID将发生变动；可执行下面的命令：

```
# find /-uid 9 -exec chown news {} \;
followed by a
# find / -uid 103 -exec chown okir {} \;
```

先说说第一条命令：从根目录中，找出用户ID是9的用户拥有的所有目录及文件，在找到的目录或文件中，将拥有者改为“news”用户（其UID刚才在密码文件中进行了更新）。需要查找(find)的原因是用户名和组名只代表对应的用户，所有文件和目录都归这个数字化的用户ID（uid）和组ID（gid）所有。passwd或group文件内，特定用户的uid和gid值发生变动时，磁盘上的文件仍然归以前的uid/gid对的uid/gid拥有。要纠正这些文件的所属关系，必须通过文件系统进行递推，并将文件“chown”向新uid/gid。为什么说chown呢？其原因是chown将在密码文件内查找和news对应的uid。最后一个\;终止了find命令；外壳程序中，斜杠后面的冒号是省略了的，将被解释为命令终止符。

利用才安装的新passwd文件来执行这些命令，并且在改变文件所属关系之前收集所有文

件名是非常重要的。要更新文件的组所属关系，可采用和前面类似的命令。

至此，系统上的uid和gid值都将和NIS域内其他主机上的那些值一致。下一步是在启用NIS查找用户和组信息的nsswitch.conf文件内，增加相应的配置行（参见上面的清单9-1。特别是“passwrd”、“shadow”和“group”这几行）。

如此一来，用户在试图登录时，login命令和它的所有对等体都会率先查询NIS映射，如果查找失败，就回到本地文件中查找。通常，你会从自己的本地文件中删除所有的用户信息，只保留根和常用账号所用的条目，比如mail。这是因为有些重要的系统任务可能要求将uid映射为用户名，或把用户名映射为uid。例如，管理cron的任务是执行su命令，临时性地充当news系统，或UUCP子系统可能将邮寄一份状态报告。如果本地passwd文件内没有针对news和uucp的条目，这些任务就会以失败告终。

这里为大家提出两大警告：一方面，迄今为止介绍的设置只适用于没有采用“影子”密码的登录套件，比如包括在util-linux包内的那些。关于随NIS一起使用的影子密码之复杂性将在后面讨论。另一方面，登录命令不是唯一的可访问passwd文件的命令——看看ls命令，它是多数人一直都在使用的。只要需要大型的列举，ls就会显示出一个文件之用户和组拥有者的象征性用户名；也就是说，对自己碰到的每一个uid和gid，它都必须一一查询NIS服务器。这样显然效率不高，更糟糕的是，NIS服务器不在同一个物理网络上时，数据报只好通过路由器进行传递。

还有一点需要说明：看看用户打算更改自己的密码时，会发生什么样的事情。通常，她会调用passwd，该命令读取新密码并更新本地passwd文件。这对NIS来说，是不可能的。因为本地passwd文件已不能在本地使用，而是在用户打算更改自己的密码时，只有登录到NIS服务器才能更改自己的密码，别无选择。鉴于此，NIS用名为yppasswd的一个daemon来替换了passwd，后者能在NIS中实现本地更改密码。所以，为更改服务器主机的密码，它通过RPC，与该服务器主机上的yppasswd daemon取得联系，为它提供更新过的密码信息。通常，执行下面的代码，便可在普通程序上安装ypasswd：

```
# cd /usr/bin
# mv passwd passwd.local
# ln yppasswd passwd
```

如此这般，便用yppasswd命令替换了passswd。

与此同时，你还必须在服务器上安装rpc.yppasswdd，并从rc.inet2开始启用。这样，便可有效地偷梁换柱，NIS一如既往地运行。

9.8 NIS与影子支持

John F.Haugh（shadow套件的作者）近来在comp.source.misc发布了影子库函数的一个版本，该版本符合GNU Library GPL标准。它已经能够支持NIS，但不很全面，所以尚未包括在标准C语言库内。另一方面，通过NIS，从/etc/shadow出版信息也可能对影子套件的性能产生影响。

尽管NYS密码查找函数没有采用shadow.byname映射或其他类似的东西，但NYS支持透明使用本地/etc/shadow文件。调用getpwnam的NYS实施方案查找与指定登录用户名相关的信息时，就会对nsswitch.conf文件内passwd指定的设备进行查找。NIS服务只查找NIS服务器上

passwd.byname映射内的用户名。但是， files服务将检查是否有/etc/shadow，如果有，就会试着打开它。如果没有，或用户没有root特权，它就会恢复其原有的行为，只在/etc/passwd内查找用户信息。但是如果影子文件存在，并且可以将其打开， NYS将会从影子文件内抽取用户密码。getpwuid函数的实施也如前所说。这种方式中，用 NYS编译的二进制文件将透明地处理影子套件的本地安装。

9.9 使用传统的NIS代码

如果你采用的客户机代码属于当前标准 libc的一部分， NIS客户机的配置则稍有不同。一方面，在要求得到自己需要的信息时，它用一个 ypbind daemon向所有活动服务器发出广播，而不是从配置文件内收集。因此，一定要在启动时，开始启用 ypbind。而且必须在NIS域已经被设定和RPC portmapper（端口映射器）已经启动之后才调用它。然后，像前面所说的那样，调用ypcat测试服务器。

近来，有大量的错误报告出现，说NIS失败时将出现这样一条错误消息：“clntudp_create:RPC portmapper failure -RPC:unable to receive”，意思是RPC端口映射器无效，RPC不能接收信息。之所以出现这种情况，是因为 ypbind将绑定信息传达给库函数时，采取的方式与以前的不兼容。最新 NIS实用程序源代码的获得和编译可以解决这个问题（ yp-linux的源代码可从ftp.uni-paderborn.de1获得，后者位于/pub/Linux/LOCAL目录中）。

与此同时，对传统NIS来说，判断是否和如何合并 NIS信息和本地文件内的信息的方式和NYS所用的方式有所不同。例如，为了使用 NIS密码，必须将下面这行包含在 etc/passwd映射内的某个地方：

```
+::::::
```

这一行标志密码查找函数从这里“插入” NIS映射。在/etc/group内插入类似的行（减去最后两个冒号），对group.*映射来说，同样如此。如果要用 NIS分发的hosts.*映射，改变host.conf文件内的“顺序”行即可。比如，如果想用 NIS、DNS和/etc/hosts文件（按下面的顺序），需要将顺序行改为：

```
order nis，bind，hosts
```

目前传统NIS实施方案还不支持其他的映射。

第10章　网络文件系统

网络文件系统(NFS)是目前最典型的采用了RPC的网络服务。它允许用户完全像访问自己的本地文件一样，访问远程主机上的文件。这是通过综合客户机端上的内核功能（使用远程文件系统）和服务器端上的NFS服务器（提供文件数据）来实现的。对客户机来说，这种文件访问完全是透明的，而且是通过不同的服务器和主机架构来实现的。

NFS的优点如下：

■ 供所有用户访问的数据可保存在一台中央主机上，并在启动时随客户装载这个目录。举个例子来说，你可将所有用户的账号信息保存在一台主机上，令网络上的所有主机都从它那里装入/home。依靠已安装了的NIS，用户就可以登录到任何系统，采用的仍是同一个文件集。

■ 占据大量磁盘空间的数据也可保存在一台单一的主机上。比如，所有与　LaTeX和METAFONT相关的文件和程序可集中在同一个地方保存和维护。

■ 管理性数据也可保存在一台单一的主机上。要在 20台机器上安装同一个文件时，不再需要rpc。

NFS中的大部分都是Rick Sladkey（邮件地址jrs@world.std.com）编写的，他还编写了NFS内核代码的Linux实施方案和大部分NFS服务器。NFS服务器源于unfsd用户空间NFS服务器和hnfs Harris NFS服务器，前者最初是Mark Shand编写的，后者则Donald Becker编写的。

先来看看NFS的工作原理：客户机可以像安装物理设备一样，要求在本地目录上安装远程主机内的目录。但是用于指定远程目录的句法有所不同。比如，为了将 vlager上的/home目录安装到vale主机上的/users，管理员将在vale上执行下面的命令（注意，可省略 -t nfs参数，因为mount能够从冒号得知它指定了一个NFS卷）：

```
# mount vlager:/home /users
```

然后，mount将试着通过RPC，链接到vlager主机上的mountd安装程序。服务器便查看vale是否被许可安装这个目录，如果可以，就为 vale返回一个文件句柄。后续发出的请求/users目录下所有文件时，都将采用这个句柄。

当有人通过NFS访问文件时，内核会任命一个指向服务器主机上的 nfsd（NFSdaemon）的RPC调用。这个调用将采用前面的文件句柄、准备访问的文件名和用户的用户 ID以及组ID作为自己的参数。采用这些参数的目地是判断用户是否被许可访问指定文件。为了防止未授权用户读取和修改文件，两台主机上的用户ID和组ID必须是一样的。

在多数实施中，客户机和服务器的 NFS机器都是作为内核级程序来实施的，这种内核级程序在系统启动时，便从用户空间开始运行。它们就是服务器主机上的 NFS daemon（nfsd）和客户机上运行的BLOCK I/O Daemon（biod）。为了改进网络流通量， boid执行了先读后写方式的异步I/O数据块；并且，若干个nfsd程序可同步运行。

NFS实施和客户机代码紧密集成在内核的虚拟文件系统（ VFS）层相比，稍有不同，它不需要通过boid进行额外的控制。另一方面，服务器代码完全在用户空间内运行，所以同时运

行多个服务器备份将由于涉及到同步问题的缘故，几乎是不可能的（请注意， NFS的内核版正在开发过程中）。目前，NFS还缺乏先读后写机制，但有幸的是， Rick Sladkey正在筹划此事（后写这个问题是指内核缓冲区按 device/inode对制作索引，因此，不能用于 NFS安装文件系统）。

关于NFS代码，最大的问题是内核版本 1分配的内存块不得大于4K；不这样的话，将导致连网代码不能对有些大型数据报进行处理，这些数据报在减掉报头等之后，字节数仍然超过3500。也就是说，对运行于这些系统（它们采用的是默认设置大型 UDP数据报，例如，SunOS系统采用的是8K）上的NFS daemon来说，要对它们进行投递，需要将它们人工地拆成小的数据包。某些情况下，这样做可能导致性能急剧降低（ Alan Cox是这样解释的：NFS规范要求在服务器返回应答之前，将每次写操作填入（ flush）磁盘。由于 BSD内核只能接受[4K]的写操作，所以将四个 1K大小包写入基于BSD的NFS服务器，将产生四个 4K大小的写操作）。这个问题在后来的内核 1.1版本中得以解决，而且还对客户机代码进行了修改，使之可以利用这一特性。

10.1 NFS的准备工作

在将NFS用作服务器或客户机之前，必须确保自己的内核能支持 NFS。新版本的内核中，proc文件系统（/proc/filesystems）上有一个简单的接口，就是为此专门设计的，利用 cat便可显示它：

```
$ cat /proc/filesystems
minix
ext2
msdos
nodev   proc
nodev   nfs
```

如果上面的显示中没有出现 nfs，就必须利用已启用的 NFS编译你自己的内核。关于配置内核网络选项的详情，参见第 3章。

对于1.1版本之前的内核，要找出其中是否已启用 NFS支持，最简单的方法是安装一个NFS文件系统。对此，可在 /tmp目录下创建一个目录，试着在它上面安装一个本地目录，如下所示：

```
# mkdir /tmp/test
# mount localhost:/etc /tmp/test
```

如果安装失败，并出现这样的错误消息：“ fs type nfs no supported by kernel ”（内核不支持文件系统类型的 NFS），你就必须利用已启用的 NFS，编译一个新内核。其他的错误消息则无伤大雅，因为你还没有开始动手在自己的主机上配置 NFS deamon呢！

10.2 NFS卷的安装

NFS卷（不是指文件系统，因为它们并不是真正的文件系统）的安装方式和普通文件系统的安装方式非常相似。利用下面的语法，调用 mount：

```
# mount -t nfs nfs volume local dir options
```

nfs_volume被指定为remote_host:remote_dir。由于这种表示法是NFS文件系统独有的，所以可省去-t nfs选项。

另外还有许多选项，供安装NFS之后的mount所用。它们要么跟在命令行的 -o开关后，要么位于NFS卷的/etc/fstab条目的options字段内。两种情况下，不同的选项间用句点隔开。命令行上指定的选项始终优先于fstab文件内所给的选项。

/etc/fstab内的示例条目如下：

```
# volume mount point type   options
news./usr/spool/news   /usr/spool/news nfs time-14，intr
```

然后，用下面的语句，安装这个NFS卷：

```
# mount news:/user/spool/news
```

在没有fstab条目的情况下， NFS安装调用看起来更为难看。比方说，假设你从一台名为Moonshot的主机安装用户的根目录，该主机采用的数据块是默认的 4K大小。为了适应Linux数据报的要求，可能需要执行下面的命令，将原先的数据块减少为 2K：

```
# mount moonshot:/home /home -o rsize=2048, wsize=2048
```

其所有有效的选项列表与NFS能够识别的mount工具（由Rick Sladkey编写）一起，在nfs手册中都对它们有着详细的说明。大家可在 Rik Faith的util-linux包内，找到上文的mount工具。表10-1列举了一部分常用的选项。

表10-1 部分安装选项

选　　项	说　　明
rsize=n，wsize=n	选项指定NFS客户机读取请求的数据报大小。由于上面提到的UDP数据报大小的限制，所以当前的默认设置是1024个字节
time0=n	设置NFS客户机等待请求完成所需的时间（以十分之一秒计）。默认设置是0.7秒
Hard	显式标记该NFS卷为硬安装。这是默认设置
Soft	软安装驱动程序（与硬安装相反）
Intr	允许发出信号，中断NFS调用。该选项用于服务器没有作出应答时

如果服务器暂时不可访问，除了 rsize和wsize之外，其他所有选项都可用于客户机行为。它们以下面的方式进行合作：只要客户机向 NFS服务器发出请求，便希望在 timeout（超时）选项中指定的时间间隔之后完成操作。如果在指定时间内，没有收到确认，就会产生副超时，将在下一个时间段内重试这一操作。如果超过了 60秒，就会产生一个主超时。

默认情况下，主超时将导致客户机在控制台打印一条消息并重新再试，这一次采用的超时值是前一次的两倍。这种情况可能永无休止地重复下去。固执地重试某次操作，直到有服务器作出应答的卷叫做硬安装卷。与之对应的是软安装卷，只要一发生主超时，软安装卷就会为调用进程生成一个I/O（输入／输出）错误。由于缓冲区引入了后写机制，在调用进程下一次调用write(2)函数之前，这个错误条件是不会传回调用进程本身的，所以，程序根本无法确定自己是否已成功将数据写入软安装卷。

不管你安装卷时采用的是硬安装还是软安装，必须考虑到你想从该卷中访问何种类型的信息。例如，如果你用NFS安装自己的程序，肯定不想自己的 X会话变得一团糟，因为有人同时启用了7个xv备份或以太网插件，使你的网络凝住了。如果对这些程序采用硬安装的话，必须确保你的计算机能够在与自己的 NFS服务器重新建立联系之前，一直处于等待状态。另一

方面，对诸如NFS安装新闻分区或FTP规档之类的非关键数据来说，它们可采用软安装，这样的话，在远程主机暂时不可抵达或已经关机时，不会将你的会话挂断。如果到服务器的网络连接比较脆弱，或通过一个已载入的路由器中转，只好要么利用 timeo选项增大初始超时值，要么硬安装NFS卷，但允许发出信号中断NFS调用，如此一来，仍然可以中断任何一个已经挂断的文件访问。

通常，mountd程序将以特定的方式监视哪些目录是由哪台主机安装的。这些信息可利用showmount程序显示出来，showmount程序也包含在NFS服务器包内。但mountd还没有这个功能。

10.3 NFS Daemon

如果想为其他主机提供NFS服务，必须在自己的机器上运行 nfsd和mountd程序。作为基于RPC的程序，它们不是由inetd管理，而是在系统启动时，就开始运行并注册为端口映射器的。因此，只须在运行 rpc.portmap之后，保证启用它们就行了。通常，应该把下面的内容包含到你的rc.inet2脚本中：

```
if [ -x /usr/sbin/rpc.mountd ]; then
        /usr/sbin/rpc.mountd; echo -n " mountd"
fi
if [ -x /usr/sbin/rpc.nfsd ]; then
        /usr/sbin/rpc.nfsd; echo -n " nfsd"
fi
```

对NFS daemon为其客户机提供的文件来说，它们的拥有者信息通常只包含在数字化的用户和组ID内。如果客户机和服务器两者关联的用户和组名和这些数字化 ID内的一致，就可以说它们共享同一个uid/gid空间。例如，利用NIS将passwd信息分发到局域网内的所有主机就属于这种情况。

但有时，也有彼此不符的情况。除了更新客户机的 uid和gid，使之与服务器的相符之外，还可用ugidd映射程序来处理这种情况。利用下面介绍的 map_daemon选项，便可借助于客户机上的ugidd，要求nfsd将服务器的uid/gid空间映射为客户机的uid/gid空间。

ugidd是一个基于RPC的服务器，与nfsd和mountd一样，是通过rc.inet2开始启用的。

```
if [ -x /usr/sbin/rpc.ugidd ]; then
/usr/sbin/rpc.ugidd; echo -n “ ugidd ”
fi
```

注意 当我发现包括这一信息很有用时，某些人可能把“参照rc.inet看作是较低级的启动方法。取而代之的是，可在etc/rc.d/*层次中找到你希望运行的各项服务的脚本。

10.4 导出文件

上面选项适用于客户机的 NFS配置，而服务器应该采用哪些选项呢？服务器的配置选项应该在/etc/exports文件内设置。

默认情况下，mountd不允许任何人从本地主机安装目录，这是非常明智的一种措施。为允许一台或多台主机安装目录，必须在 exports（导出）文件中指定它。导出文件示例如下：

```
# exports file for vlager
/home              vale(rw) vstout(rw) vlight(rw)
/usr/X386          vale(ro) vstout(ro) vlight(ro)
/usr/TeX           vale(ro) vstout(ro) vlight(ro)
/                  vale(rw,no root squash)
/home/ftp          (ro)
```

每一行都定义了一个目录和允许装入该目录的主机。主机名通常采用完整资格域名，也可以增加 *和?通配符，表示采用的是 Bourne外壳。举个例子来说， lab*.foo.com对应lab01.foo.com和laber.foo.com。如果不指定主机名，就像上面示例中的 /home/ftp目录一样，是不允许任何一台主机安装所定义的目录的。

在导出文件中查看客户机主机时， mountd将使用gethostbyaddr (2)调用，查找客户机的主机名。利用DNS之后，这个函数调用就会返回客户机的规范名，所以你必须确保不要在导出文件内采用主机别名。如果不采用 DNS，函数调用返回的就是它在主机文件内找到的第一个与客户机地址相符的主机名。

主机名后面是一个可选项，它是一个用括号括起来的标记清单，各标记用句点隔开。表10-2展示了一部分标记及值。

表10-2 部分标记及值

标 记	值
nsecure	允许从该机起，进行未经验证的访问
unix-rpc	需要从该机起，进行Unix域的RPC身份验证。这个标记只要求请求从一个保留 Internet端口（也就是端口号必须小于1024）发起。默认情况下，这个标记是打开的
secure-rpc	要求从该机起，进行安全 RPC身份验证。这个标记尚未实施。个中原由，参见 Sun的Secure RPC文档
kerberos	要求从该机起，对访问实行 Kerberos身份验证。这个标记也没有实施。参见 MIT的Kerberos身份验证系统文档
root squash	这是一个安全特性，它在客户机上的 uid 0到服务器上的uid65534(-2)之间，实行请求映射，从而否决超级用户对特定主机的特殊访问权。后一个 uid应该和用户nobody（无人）对应
no root squash	不从uid 0映射请求。默认设置是开
ro	安装文件结构，只读。默认设置是开
rw	安装文件结构，只写。默认设置是开
link relative	通过在../ ' s上加必要的号码，将绝对象征性链接（链接内容以斜杠开头）转换为相对链接，从而从包含指向服务器上根目录的目录中，取得自己需要的链接。这个标记只适用于一台主机的整个文件系统都已安装时，有些链接可能无处可指，或更糟的是，指向它们根本不打算拥有的链接。默认设置是开
link absolute	让所有的象征性链接保持原样（对 Sun提供的NFS服务器而言，则是普通行为）
map daemon	该标记要求NFS假定这样一个前提：客户机和服务器不共享同一个 uid/gid空间。然后，nfsd通过查询客户机的ugidd程序，建立一个映射客户机和服务器 ID关系的列表

只要开始运行 nfsd或mountd，导出文件错误分析将以“注意”形式，报告给 syslogd的daemon设备。

注意，主机名是根据客户机的 IP地址逆向映射而来的，所以必须保证你的解析器配置无误。如果采用的是 BIND，而且非常注重安全问题，则应该启用电子欺骗，检查自己的host.conf文件。

10.5 自动安装器

有时，安装所有的NFS卷相当费事，一来因为卷数较多，二来因为耗时。因此，一个所谓的“自动安装器”应运而生。它其实是一个daemon，会自动而透明地将安装需要的NFS卷，并在没有使用它们达一段时间后，取消安装。自动安装器的过人之处在于它能够从备用地方安装特定的卷。比如，你将自己的X程序和支持文件备份保存在两台或三台主机上，通过NFS，令其他主机安装它们。如果利用自动安装器，可指定这三台主机将被安装在 /usr/X386上，然后，自动安装器便试着进行安装，直到安装尝试成功为止。

常随Linux一起使用的自动安装器叫作amd。这个程序起初是Jan Simon Pendry编写的，已经被Rick Sladkey移植到Linux系统中。目前版本是amd-5.3。

关于amd的详情，不在本章讨论之列；要想找到一本好的参考手册，不防参阅其源代码；其中包含一个内容丰富翔实的texinfo文件。

第11章　泰勒式UUCP

UUCP是二十世纪70年代后期，由AT&T公司贝尔实验室的Mike Lesk设计出来，用于通过公用电话线路，提供一个简单的拨号网络。由于许多人都想在自己的家中，通过 modem收发电子邮件和Usenet新闻，所以UUCP至今仍然非常流行。尽管不同的硬件平台和操作系统上运行着大量不同的实施方案，但总的说来，都不会产生冲突。

但是，若干年来，有的软件越来越“ 标准 ”，已经没有能够称为 UUCP的UUCP了。自1976年发布第一个版本以来， UUCP一直处于一个非常稳定的发展过程。目前， UUCP主要分为两大类，其区别表现在硬件支持和它们的配置上。还有其他许多实施方案，大都与这两类大同小异。

其中一类就是所谓的“ 版本 2 UUCP ”，它起源于1977年，由Mike Lesk、David A. Novitz和Greg Chesson合作编写的第一版UUCP。尽管它相当老了，但仍然广为使用。新近发布的版本2，使最近的UUCP更为好用。

第二类是在1983开发的，常被称为BNU（基本连网程序）、HoneyDanBer UUCP或HDB。这个奇怪的名字源于其作者： P.Honeyman、D.A. Novitz和B.E.Redman。HDB被认为是消灭了版本2 UUCP之不足的功臣。例如，增加了新的传输协议，拆分了假脱机目录，能让你与之进行UUCP通信的每个站点上只有一个目录。

当前，软件厂商在其产品中捆绑的UUCP实施是Taylor UUCP 1.04（该程序是Ian Taylor于1993年编写并申请版本），它是本章所讲的版本的基础。以下将其统称为泰勒式 UUCP，其1.04版本是1993年2月发布的。除了识别传统的配置文件外，对泰勒式 UUCP稍作编译，也能够识别新式（比如a.k.a.“ Taylor ”）的配置文件。

本书编写过程中，最新的1.05版本也发布了，很快将进入分销渠道。这两个版本的不同之处对你而言，没什么影响，因此，你大可利用本书中的信息，配置泰勒式 UUCP 1.05。

本章的目的不是彻底论述UUCP命令的命令行选项有哪些，其作用是什么，而是向大家介绍如何设置一个正在运行的 UUCP节点。第一小节简要介绍 UUCP是怎样实施远程命令执行和文件传输的。

但是，我们将假设这个前题：大家对 UUCP套件的用户程序有所了解。它们是 uucp和uux。有关说明可参考在线手册。

除了公开供用户访问的程序、 uucp和uux程序外， UUCP套件中还包含有大量的只用于管理的命令。这些命令用于监控节点间的 UUCP通信，删除旧的日志文件和编译特性等。详细情况未作讨论，因为我们这里的重点是 UUCP。除此以外，这些命令已经编入文档，非常容易理解。但是， UUCP套件中还有第三种类型，这类由真正的 UUCP“ 工程兵 ”组成，它们叫作uucico（cico表示copy-in copy-out）和uuxqt，后者执行从远程系统发来的作业。

11.1　关于UUCP

对于那些在本章内找不到自己需要的资料的用户来说，应该仔细阅读软件包自带的文档。

它是一个texinfo文件集，对利用泰勒式配置方案进行的设置进行了详细说明。利用 tex和makeinfo，可将texinfo分别转换为DVI和GNU info文件集。

如果想采用BNU甚至版2配置文件，则可参考另一本好书《 Managing UUCP和Usenet》。我觉得这本书讲得非常透彻。关于 Linux系统上可用的UUCP，其详情可参考Guylhem Aznar（邮件地址guylhem@.oeil.qc.ca）编写的UUCP-HOWTO文档，该文档定期在comp.os.linux.announce发布。

另外，还有一个UUCP新闻组，名为comp.mail.uucp。如果有和泰勒式UUCP相关的问题，最好提交到该新闻组，让其中的专家为你解决疑难，而不是访问 comp.os.linux。

11.1.1 UUCP传输和远程作业的执行

理解UUCP的关键在于搞清“作业”（jobs）一词的概念。用户利用uucp或uux发起一次传输就叫作一个作业。一次作业由一个命令和一个文件集组成，这个命令将在远程系统上执行，而文件集则将在不同站点间进行传输。

UUCP一般不会立即调用远程系统来执行作业 (或你可利用kermit执行)。相反地，它会临时性地将作业说明保存起来。这就是所谓的“假脱机”（spooling）。下面保存作业说明的目录树就被称作假脱机目录（spool directory），一般位于/var/spool/uucp内。我们的示例中，作业说明中包含与即将执行的远程命令（ lpr）相关的信息、请求执行作业的用户和两个其他的项目。除了作业说明外， UUCP还必须保存输入文件netguide.ps。

spool文件的确切位置和命名可能会依据某些编译时选项而发生变化。兼容于 HDB的UUCP一般将spool文件保存在名为/var/spool/uucp/site的目录内，而“ site ”代表的是站点名。在针对泰勒式配置方案进行编译时， UUCP将在站点专有的spool目录下，为不同类型的spool文件创建子目录。

UUCP定期拨入远程系统。与远程主机建立连接之后， UUCP将说明作业的文件和所有的输入文件传递给它。进入远程主机的作业不会立即执行，而是在连接中断之后才进行。这是由uuxqt来完成的。uuxqt还会将目的地是另一个站点的作业进行转发。

为了将重要和次要作业区分开来， UUCP为各个作业制定了级别（grade）。从0到9的级别用字母来表示，即从A到Z。级别越高，越优先。邮件对应的级别一般是 B或C，而新闻对应的则是N。级别越高的作业，其传输次序也优先。级别的分配是在调用 uucp或uux时，利用-g标记来分配的。

某些时候，还可禁止传输低于指定级别的作业。这就是所谓的对话期间的最高 spool级别，其默认设置是-z。注意：只有其级别等于或高于最大spool级别的文件，才能得以传输。

11.1.2 uucico的内部运行

为了解uucico需要知道某些情况的原因，先简要谈谈它在事实上是如何与远程系统建立连接的。

当你从命令行执行uucico -s系统时，它首先必须有一条物理上的连接。根据连接类型，开始采取行动，比如，在使用电话线时，它必须先找到一个 modem。在TCP链接上，它必须调用gethostbyname(3)，将主机名转换为网络地址，再找出应该打开的端口，并将网络地址绑定到相应的套接字上。

在链接建立之后，必须传递一个身份验证进程。这个进程一般由要求登录名的远程系统和一个密码组成。这就是常说的“登录对话”（login chat）。身份验证进程要么由常用的getty/login套件执行，要么由uucico自己在TCP套接字上执行。如果验证成功，远程主机就会发起uucico。用于初始化链接的uucico之本地备份常被称为主管，远程备份常称为从属。

接下来是握手阶段：现在，主管主机发送其主机名和一些标记。从属便查看这个主机名是否能够登录，收发文件等等。标记说明了准备传输的 spool文件的最高级别。如果一切正常，就将产生对话计数或呼叫下个号码检查。有了这一特性，两个站点便可维持成功的链接。如果文件的级别低于指定的级别，握手就会失败。这个特性特别适合于对付网络骗子。

最后，两端的uucico试图与常用的传输协议达成一致。这个协议掌管数据传输、数据一致性检查和出现错误时的重新传送数据的方式。为什么需要不同的协议呢？因为需要支持不同的连接类型。比如，由于电话线路担心出错，因此要求一个“安全”协议，而 TCP传输又是天生的可靠，可以使用一个更为高效的协议，这类协议将摒弃特殊错误查找。

握手结束之后，真正的文件传输阶段开始了。连接双方都打开选定的协议驱动程序。这些驱动程序将可能执行协议特定的初始化序列。

其一，主管将排队等候远程系统的所有文件发送出去，这些文件的 spool级别非常之高。发完之后，它便通知从属发送完毕，而从属此刻可能已经挂断。现在，从属要么同意挂断，要么接管登录对话。这时，角色发生了变化：此刻，远程主机成了主管，本地主机则成为从属。现在，新任的主管开始发送自己的文件。发送完毕之后，两个 uucico的交换终止消息并关闭连接。

我们不打算继续深入讨论。详情参考 UUCP源代码或相关书籍。网上有一篇见解独到的好文章，是David A.Novitz写的，其中详细介绍了 UUCP协议。泰勒式 UUCP常见问题集也对UUCP的实施进行了细致的探讨。这个 FAQ集定期出版于comp.mail.uucp。

11.1.3 uucico命令行选项

本小节只说明几个最重要的命令行选项，要想获得一份关于所有选项的完整列表，请参考uncico手册。

11.2 UUCP配置文件

与较为简单的文件传输程序相比，设计UUCP的目的是为了能够自动处理所有的文件传输。一旦正确设置了 UUCP，管理员就不必在每日不变的一些基本操作上花太多的时间。设置UUCP所需的信息保存在两个配置文件内，这两个文件驻留在 /usr/lib/uucp目录下。它们只用于拨号。

注意 本章中，大家还将看到若干个实例，其中引入的示例目前还不能用。虽然留有相关的文本说明，但最好参考它们的LDP。

11.2.1 泰勒式UUCP简介

说UUCP配置很难，并非言过其实。它的确涵括了许许多多的细节问题，即便配置文件的简洁格式也无法使事情变得简单（尽管与老式的 HDB和版本2中的格式相比，泰勒式 UUCP的格式更易于理解）。

为了让大家感觉一下这些文件是如何相互影响的，我们将介绍几个最重要的文件，看看这些文件内的条目。但现在不打算一一细讲；将在随后的几个小节内，重点介绍其中的一个。如果将自己的机器设置为UUCP，最好从某些示范文件着手，并逐步地改编它们。示范文件可从下面的示例中选出，也可从你自己喜欢的软件产品中选出。

本小节提到的所有文件都包含在 /usr/lib/uucp或子目录thereof内。有的软件产品中还有UUCP二进制文件（这些文件为HDB和泰勒式配置方案的启用提供了支持），并分别为每个配置文件采用不同的子目录。/usr/lib/uucp内，一般都有一个README文件。

要想UUCP运行无误，这些文件必须只能归 uucp用户所用。其中一些文件中包含密码和电话号码，因此，它们应该有600模式许可。

注意 尽管多数UUCP命令都必须setuid为uucp，但你同样必须确定uuchk不是uucp。否则，用户将能显示密码，即使他们拥有模式600。

核心的UUCP配置文件是/usr/lib/uucp/config，用于设置常规参数。它们之中，最重要的是（迄今为止，也是唯一的）你的主机之 UUCP名。

下个重要配置文件是 sys文件。该文件中包含的信息和与你链接的站点之系统有关。其中包括站点名和关于链接本身的信息（比如使用 modem连接时采用的电话号码）。

port命名准备使用的端口。 time指定何时拨号。 chat说明登录对话脚本 —— 字串顺序必须进行交换，以便允许 uucico登录进入 Pablo。稍后再回头讨论对话脚本。 port命令不会命名/dev/cua1之类的设备专有文件名，而是命名端口文件内的条目。只要这些名字引用的是端口文件内的有效条目，你就可以按照自己的喜好，对它们进行分配。

端口文件中保存有与链接本身相关的信息，它说明了即将使用的设备专有文件名、支持的速率范围和连接到该端口的拨号设备的类型。随后的条目说明了 /dev/cua1（a.k.a COM2），连接到这个设备的NakWell modem的速率可高达38 400bps。选择这个条目的名称时，应该使它和sys文件指定的端口名相符。

关于拨号者本身的信息，则保存在另一个文件内，名为 dial。对每类拨号者，它一般都包含拨叫远程站点时执行命令的序列和具体的电话号码。再次提醒大家注意，它也可作为一个对话脚本。

对话中的第一行指定 modem对话，表示命令的执行序列（这些命令或发给 modem，或是从modem发回），用于初始化 modem并令其拨叫指定的电话号码。 T序列将由uucico用电话号码来替换。

uucico的第一步是在 sys文件内查找 Pablo。从Pablo的sys文件条目中，得知它应该采用serial1端口来建立连接。端口文件告诉它这是一个 modem端口，而且已经绑定了一个 NakWell modem。

接着，uucico便开始搜索描述NakWell modem的拨号条目，找到之后，便打开 /dev/cua1，执行拨号者对话。也就是说，它发出一个“ ATZ ”，并等待“ OK ”应答等。如果碰到T这个字串，它就将其替换为电话号码 123-4567，这个号码取自sys文件。

modem返回CONNECT之后，表示已经建立连接， modem对话已经完成。现在， uucico返回sys文件，并执行登录对话。在我们的示例中，它将等待 login:提示，然后，再发自己的用户名（Neruda），等到出现password:提示时，在发自己的密码 lorca。

身份验证完成之后，远程端主机便假定发起自己的 uucico。然后，连接双方开始进入握手

阶段（参见前一小节）。

11.2.2 UUCP需要知道些什么

在开始编写自己的UUCP配置文件之前，必须收集一些它需要知道的信息。

首先，必须搞清楚你的 modem所绑定的串行设备是什么。一般说来， DOS端口COM1到COM4对应的设备特有文件是/dev/cua1到/dev/cua3。许多软件（比如Slackware），都创建了一个/dev/modem链接，链接到其相应的cua*设备文件，配置kermit、seyon等，从而能够使用这些文件。这种情况下，也应该在你的 UUCP配置文件内采用/dev/modem链接。

这样做的原因是所有的拨号器都采用所谓的锁文件来标识串行端口正在使用中。这些锁文件名是一连串的LCK..字串和设备文件名。比如 LCK..cua1。如果拨号器对同一个设备采用了不同的设备名，它们就不能对彼此的锁文件进行识别。其结果是，在开始的那一刻，同时中断彼此的会话。这在你利用 crontab条目，安排自己的UUCP调用时，是不太可能的。关于如何设置自己的串行端口，请参考第 4章。

接下来，必须找出modem的传输速率。必须将其值设为自己希望得到的最大有效传输速率。有效传输速率可以稍高于 modem规定的原始物理传输速率。比如，许多 modem收发数据的速率都是2 400bps。而利用V.42bis之类的压缩协议时，真正的传输速率可高达 9 600bps。

当然，如果希望自己的UUCP十项全能，肯定还需准备要拨叫的那个系统之电话号码。而且，还需要一个有效的登录ID和密码。

> **提示** 如果你正准备拿UUCP做试验，最好能找到一个离你较近的归档站点。记下其登录名和密码 —— 是可随意下载的。多数情况下，它们是类似于 uucp/uucp和nuucp/uucp的。

另外，还必须知道如何才能真正地登录到远程系统。举个例子来说，在登录提示出现之前，有必要按Break键吗？登录提示显示的是“ login: ”，还是“ user: ”？这一点对编写对话脚本来说，是非常必要的，对话脚本是告诉 uucico如何登录的“ 剧本 ”。如果不知道这一点，或平常所用的对话脚本失效了，就试着用 kermit或minicom之类的终端程序来拨叫远程系统，并准确地记下你必须做的事。

11.2.3 站点的命名

由于有基于TCP/IP的连网，所以你的主机必须有一个用于 UUCP连网的主机名。如果只想将UUCP用于站点间或本地网络上的文件传输，这个名就不需要符合任何标准。唯一的限制是不得超过7个字符，而且不得与文件系统的主机名有冲突。

但是，如果打算把 UUCP用于邮件或新闻链接，就应该考虑使用 UUCP Mapping Project（UUCP映射工程）注册的名字。 UUCP映射工程的详情，参见第 12章。即使你参与了一个域，也有必要考虑为自己的站点选择一个正式的 UUCP名。

一般说来，人们爱在自己的完整资格域名中，选择第一个组件来作为自己的 UUCP名。比如，如果你的站点域名地址是 swim.twobirds.com，那么，你的UUCP主机名就是Swim。许多UUCP站点都采用这一命名原则。当然，也可采用一个和自己的完整资格域名完全不相干的UUCP名。

但是，务必保证不要在邮件地址中使用未取得资格的站点名，除非你已经将其注册为自

己的正式UUCP名（UUCP映射工程负责对全球的所有 UUCP主机名进行注册，并保证它们的唯一性。要注册自己的UUCP主机名，可询问负责处理你的邮件之站点维护人员，他们能够帮助你）。因为，如果向尚未注册的 UUCP主机发邮件，最终会犹如泥牛入海。如果你使用的名称与别的站点雷同，邮件也会被路由到那个站点，对其邮发主管来说，是件非常头疼的事。

默认情况下，UUCP套件采用的主机名是站点的 UUCP名。这个名字通常是在 /etc/rc.local脚本内设置的。如果你的 UUCP名和你自己设置的主机名不同，就必须利用 config文件内的主机名选项，将你的UUCP名告知uucico。这是下文讨论的主题。

11.3 泰勒式配置文件

现在回过头来看看配置文件。

泰勒式配置文件一般由若干行组成，这些行中包含一些关键字 -值对。#号代表一个批注，表示这一行是备注。如果程序本身要采用 #号，就可以在#前加一个斜杠。

利用这些配置文件，可对相当多的选项进行调整。我们不打算对所有的参数进行深入讲解，只讲一些最重要的。有了它们，才可能配置基于 modem的UUCP链接。如果你想在TCP/IP或直接串行线路上使用 UUCP，则可参见其他小节，它们对一些必要的修正进行了说明。其全面而完整的参考资料包含在泰勒式 UUCP源代码附带的Texinfo文档内。

当你认为自己已经完成 UUCP系统的配置时，可利用 uuchk工具（位于/usr/lib/uucp），对自己的配置进行校验。uuchk读取你的配置文件，再打印一份详细报告，报告中对每个系统所用的配置值进行了详细说明。

11.3.1 常规配置选项：config文件

一般情况下，除了 UUCP主机名外，是不用这个文件的。默认情况下， UUCP采用的是hostname命令设置的主机名，但最好显式设置 UUCP名。

当然，还有许多参数是可以设置的，比如假脱机目录名和匿名 UUCP的访问权限等。后者将在稍后进行讨论。

11.3.2 如何将其他的系统告知UUCP：sys文件

sys文件描述的是你的机器所了解的系统。条目是系统关键字引入的；后面几行则是系统为站点指定的参数。一般说来，系统条目会对电话号码和登录对话之类的参数进行定义。

第一个系统行之前的参数设置的是用于所有系统的默认值。通常情况下，应该在默认值这部分设置协议参数等等。

下面，便为大家详细介绍最典型的几个字段。

1. 系统名

system命令用于命名远程系统。必须指定规范的远程系统名，而不是你自行发明的别名，因为uucico将在你登录时，再次检查远程系统广播的系统名（版本 2 UUCP在调用时，不会广播系统名，但新版本的UUCP实施通常都会这样做，泰勒式 UUCP也不例外）。

每个系统名可能只出现一次。如果你想针对同一个系统，采用多个配置文件集（比如多个电话号码，让uucico依次拨打），就可以指定备用配置文件。下面将具体介绍备用配置。

2. 电话号码

如果远程系统能够通过电话线路抵达，phone字段指定的就是modem应该拨叫的电话号码。

该字段内可能包含若干个记号，这些记号的解释由 uucico过程负责。等号=表示等待第二次拨号音调，“ — ”号则产生1秒钟的暂停。例如，在拨打区号和电话号码之间，如果没有暂停的话，有些电话线路会中断（不知道应该用哪个术语来表达这种情况，说简单点，有点类似于公司的内部电话，必须先拨打0或9才能打出去）。

任何一个内含阿拉伯数字的字串都可用于隐藏站点比如区号之类的站点相关信息。利用dialcode文件，便可将此类字串翻译成dialcode。

有了翻译文件，就可以在 sys文件内采用 Bogoham 7732之类的电话号码，它们更容易理解。

3. 端口和速率

端口和速率选项用于选定拨打远程系统的设备和设置设备的最大速率（ tty的波特率至少和最大传输速率一样高）。系统条目即可以单独采用一个选项，又可以两者都用。在端口文件内查找合适的设备时，只能选定那些有匹配端口名和／或速率范围的端口。

一般情况下，用速率选项就够了。如果端口文件内，只定义了一个串行设备，无论如何，uucico始终都会选择右边的端口，你只须为它指定合适的速率。如果你的系统上备有多个modem，通常也不愿意命名一个特殊的端口，因为 uucico一旦发现有这么多个设备，就会依次地尝试每一个设备，直到找到没有使用的设备为止。

4. 登录对话

由上可知，登录对话脚本是告诉uucico如何登录到远程系统的脚本。它由一连串记号组成，用于指定本地uucico进程发出和希望收到的字串。其目的在于令 uucico等待远程主机向其发送登录提示，随后返回登录名，等待远程系统向其发送密码提示，最后，再发送密码。 expect（希望收到）和send（发送）字串是在备用文件中指定的。 uucico自动将回车字符（r）添加到send字串上。

uucico还考虑到了某些违例情况，比如，在发送提示之前，远程主机的 getty需要进行重新设置。所以，你可将一个子对话添加到一个 expect字串上，偏移一个“ — ”。只有主要的expect字串失败时（比如，发生超时），才执行这个子对话。利用这一特性的方法之一是：远程站点没有显示登录提示时，发送一个 BREAK。下面的例子给出了一个全面的对话脚本，如果你必须在登录提示出现之前按回车的话，这个脚本应该是有效的。 Linux系统中，要求UUCP不等待提示，即刻进行下一个send字串的发送。

5. 备用

有的时候，对于一个单独的系统，我们希望能够有多个配置。比如，可以通过不同的modem线路，接通这一系统等。利用泰勒式 UUCP，定义一个所谓的备用，就能立即实现这一想法。

备用条目中保持主要系统条目中的所有设置，并只指定了默认系统条目中那些应该改写或增添的值。备用从系统条目偏移含有关键字“ 备用 ”（alternate）的一行。

在拨打Pablo时，uucico首先拨打123-456，如果失败，再拨打备用条目中的号码。备用条目中保持有主要系统条目中的所有设置，并只改写了电话号码。

6. 限制拨打次数

泰勒式UUCP提供了许多方法，可以用来限制拨打远程系统的次数。为什么要限制拨打次数呢？其原因不外乎远程主机只在上班时间提供服务，或只是为了避免高频率的拨打次数。

注意，只要为uucico提供-S或-f选项，就能改写拨打次数限制。

默认情况下，泰勒式 UUCP不允许随时连接，所以你必须采用 sys文件内的时间规格。如果不在意拨打次数限制，可利用 sys文件内的Any值，指定time选项。

要限制拨打次数，最简单的方法是利用时间条目，这个条目后面跟一个字串，该字串由两个子字段组成：日期和时间。日期可以是 Mo、Tu、We、Th、Fr、Sa、Su，或Any、Never和代表工作日的Wk。时间则由两个24时的时钟值组成。这些时间记号之间没有空格符或制表符。任何一个日期和时间规格都必须用逗号列为一组。

特殊的时间字串Any和Never分别表示：随时拨打或从不拨打。

time命令采用了一个可选的第二个参数，该参数以分钟计，说明几分钟后重试。在连接尝试失败之后， uucico不允许在特定的时间间隔内，另行尝试拨打远程系统。默认情况下，uucico采用的是一个exponential backoff方案，也就是说，时间间隔随失败次数的增多而增大。例如，你指定的重试时间间隔是 5分钟， uucico将在上次尝试失败之后的 5分钟内，拒绝拨打远程系统。

timegrade命令允许你在作业安排中增加一个最大假脱机级别。这样一来，只要一建立拨号连接，假脱机级别为 C或更高的作业（邮件的队列级别是 B或C）就能得以传输，而新闻（其队列级别一般是N）只能在夜间或周末得以传输。

和time命令一样， timegrade命令也采用了一个重试时间间隔（以分钟计）作为其第三个可选参数。

然而，与假脱机级别有关的说明则适合这种情况：首先， timegrade命令只适用于你的系统发出的作业；远程系统仍然一如既往地传输自己的作业。因此，你可利用 call-timegrade选项，显式请求它只发送假脱机级别高于指定级别的作业；但这样不能保证它会接受一个请求（如果远程系统运行的是泰勒式 UUCP，它就会接受这一请求）。

类似地，远程系统拨入本地时，不会对 timegrade字段进行检查，所以所有排队等候拨叫系统的作业都能得以发送。但是，远程系统可以显式请求你的 uucico限定发送特定级别的假脱机作业。

11.3.3 设备：端口文件

端口文件用于告诉uucico哪些端口是可以用的。既可以是modem端口，又可以是其他类型的端口（比如直接串行线路）或已获支持的 TCP套接字。

与sys文件一样，端口文件由几个独立的条目组成。首先是关键字 port，然后是端口名。这个名可以供sys文件内的端口语句使用。端口名不必是唯一的；如果几个端口同名， uucico将依次尝试于各个端口建立连接，直到找出一个目前尚未使用的为止。

port命令后面应该紧跟type语句，该语句用以说明端口类型。有效的类型有 modem、用于直接连接的direct和用于TCP套接字的tcp。如果不采用 port命令，端口类型就会采用默认的modem类型。

本小节中，我们只介绍modem端口， TCP端口和直接串行线路将留在稍后进行讨论。

对modem和直接端口来说，都必须利用device directive（设备指令）指定用于拨出的设备。通常，这个设备是一个设备专有文件名，位于/dev目录下，比方说/dev/cua1这个例子就是如此。

注意 有人偏爱ttyS*设备，这类设备只能用于拨入。

如果是modem设备，端口条目还能用于判断连接到这个端口的 modem类型是什么。不同类型的modem必须采用不同的配置。即使它声明自己采用的是贺氏标准也是如此。你必须告诉uucico如何初始化modem、如何令其拨打指定的电话号码。泰勒式 UUCP将所有拨号器的相关描述保存在一个名为 dail的文件内。要想采用这些拨号器，必须利用 dialer命令，指定拨号器名称。

根据自己拨打的系统，你可能想采用别的方式来利用 modem。例如，高速modem试图以14 400bps的速率连接老式的modem时，后者不能识别它们，只是将线路断掉，而不是协商以9 600bps的速率连接。如果你得知自己准备连接的站点采用的正是这类该进博物馆的 modem时，就必须另行设置自己的 modem。所以，就需要在端口文件内增加一条端口条目，指定另外的拨号器。现在，可以为新端口指定一个新端口名，比如 seial1-slow，并采用 sys文件中drop系统条目内的端口指令。

要想区分各个端口，最好看它们支持的速率。站点drop的系统条目将serial1作为其端口名，但只请求以9 600bps的速率进行连接。然后， uucico自动采用第二个端口条目。最后，利用第一个端口条目，拨打系统条目中有 38 400bps速率的其他所有站点。

11.3.4 如何拨号：拨号文件

拨号（dial）文件说明的是不同的拨号器采用的方法。过去， UUCP提到的是拨号器，而不是modem，因为早期，普遍采用一个（非常昂贵）自动拨号设备来充当 modem。今天，许多modem都有内置的拨号支持，所以拨号器和 modem之间的界线才日益淡化。有的甚至干脆把拨号器叫作modem。

不管怎么说，不同的拨号器和 modem都可能要求不同的配置。你可将各类拨号器和modem的相关描述写入dial文件内。dial文件内的条目以dialer命令开头，该命令会给出拨号器的名称。

除dialer命令之外，另一个最重要的条目就是 chat命令指定的modem chat（modem对话）。它和登录对话类似，由一连串字符组成，这些字符是由 uucico发给拨号器的字串和依次返回的应答字串组成。它常用于将modem重新设置为已知状态并进行拨号。

uucico modem对话以空的expect字串开始。因此， uucico立即发送第一条命令（ ATZ）。ATZ是Hayes兼容modem采用的命令，用于重新设置 modem。然后， uucico等待modem发回OK之后，再发送下一条命令，这条命令将关闭本地应答，如此等等。 modem再次发回OK之后，uucico便发出拨号命令（ATDT）。这个字串中的避开序号 T被系统条目文件内的电话号码所代替。然后， uucico就等待modem返回CONNECT字串，后者标志着与远程 modem的连接已经成功建立。

modem不能连接到远程系统，这是常有的事。比如，远程系统正忙于和别人对话，线路正忙等。这时， modem返回的错误消息，能够指出不能连接的原因。 modem对话不能侦测此类消息；所以uucico将继续等待自己希望收到的字串，直到超时为止。因此， UUCP日志文件中将只能有这样一行：“ timed out in chat script ”（对话脚本超时），而不能记录不能连接的真正原因。

但是，泰勒式UUCP允许你利用前面所讲的 chat-fail（对话失败）命令，将错误消息告知uucico。uucico在执行modem对话期间，一侦测到 chat-fail字串，就会中止拨号，将错误消息

记录在UUCP日志文件内。

前面的示例中，最后一个命令要求UUCP在开始modem对话之前，一直停留在DTR这一行。侦测到DTR行有变动时，许多modem都可被配置为挂起，并进入命令模式。

注意 也可将有些modem配置为侦测到DTR发生变化时，自行重新设置。但是有的则不能这样，有时还会自行挂断。

11.3.5 TCP上的UUCP

利用UUCP，在TCP链接上传输数据，乍一听，有些可笑。事实上，这并不是个坏主意，特别是在传输Usenet新闻组之类的大型数据时。在基于 TCP的链接上，新闻通常是用 NNTP协议来交换的，在NNTP协议中，文章的请求和发送是单独进行的，无须压缩和任何优化。尽管这对大型的站点来说，可以同时接收多个新闻传输，但对通过速度较慢的连接（比如 ISDN）接收新闻的小型站点来说，这种方法并不理想。这类站点通常想综合利用 TCP的传输质量和大批量发送新闻的好处，也就是说既能够对新闻进行压缩，又能够降低传输开销。传输批量新闻的标准作法是在TCP链接上使用UUCP。

11.3.6 直接连接的使用

假设你采用一条直接线路将自己的系统 vstout连接到tiny。与modem连接的情况极为相似，必须在sys文件内写入一条系统条目。 port命令标识串行端口tiny已处于连接状态。

端口文件中，必须对直接连接所用的串行端口进行说明。这里不需要拨号器条目，因为没必要进行拨号。

11.4 UUCP的注意事项：调节权限

11.4.1 命令执行

UUCP的任务是把文件从一个系统复制到另一个系统，并请求在远程主机上执行特定的命令。当然，作为管理员的你，肯定想对各系统的命令执行权进行控制 —— 允许它们在你的系统上执行所有命令并不是件好事。

默认情况下，泰勒式 UUCP允许其他系统在你的机器上执行的命令只有两条，那就是rmail和rnews。这两个命令常用于在UUCP上交换电子邮件和Usenet新闻。uuxqt采用的默认搜索路径是一个编译时选项，但其中通常应该包含 /bin、/usr/bin和/usr/local/bin。要针对特定的系统改变命令设置，可采用 sys文件内的命令关键字。类似地，也可以用 command-path（命令路径）语句，改变搜索路径。例如，可以让系统 Pablo在执行rmail和rnews命令之外，还执行rsmtp命令（rsmtp用于投递带有批量SMTP的邮件；有关详情，将在后续的邮件章节中讨论）。

11.4.2 文件传输

泰勒式UUCP还允许更为详细地优化文件传输。甚至还可以取消将文件传输到特定系统或禁止从特定的系统传输文件。只须把请求设置为 No，远程系统就不能向你的系统上传或下载文件。类似地，你也可将传输设置为 No，禁止你的用户向特定的系统上传或下载文件。默认

情况下，本地和远程系统上的用户都可上传和下载文件。

另外，可以配置准备复制文件的目录。通常情况下，作为系统管理员的你，多半打算限制远程系统用户访问你的目录结构，但仍允许你自己的用户发送其根目录中的文件。此时，远程系统用户将只能接收公用UUCP目录（/var/spool/uucppublic）中的文件。这个目录中常放置一些公用文件，与因特网上的FTP服务器非常类似。这个目录常用“ ~ ”符号来表示。

为此，泰勒式UUCP为收发文件提供了四个用于配置目录的不同命令。它们是：

local-send（指定一份目录清单，用户要求 UUCP发送这一系列目录中的文件）、local-receive（给出一份目录清单，用户要求 UUCP将收到的文件放入这些目录）、remote-send和remote-receive，对远程系统的收发请求进行类似的处理。

local-send命令允许你的用户将 /home目录下和公用UUCP目录中的所有文件发给 Pablo。local-receive命令允许他们接收发向人人可写的目录或 /home目录下的人人可写文件，这些人人可写的目录在 uucppublic的receeive目录中。 remote-send指令允许 Pablo请求/var/spool/uucppublic内的文件，但incoming和receive目录下的文件除外。uucico是怎样得知这一点的呢？原来目录名前面有一个感叹号！最后一行允许 Pablo将任何文件上传到incoming目录。

利用UUCP传输文件时，最大的问题是它只接收那些发到人人可写的目录中的文件。这将导致有的用户为其他用户设计一些圈套，或捉弄他们。但是，这个问题是无法避免的，除非你不用UUCP文件传输。

11.4.3 文件转发

UUCP提供了一种机制，可令其他系统在你这一端执行文件传输。例如，允许你使 seci为自己取得uchile上的文件，并将它发送到自己的系统中。

这种通过多个系统的传输机制就叫作“ 转发 ”。上面的例子中，使用转发机制的原因可能是seci能够通过UUCP访问uchile，你的主机却不能。但是，如果你的主机也运行 UUCP系统，可能想将转发服务限定在自己信得过的少数几台主机上，以免自己为了下载最新的 X11R6源代码不得不支付巨额的电话账单。

默认情况下，泰勒式UUCP完全不允许转发。要针对特定的系统启用转发机制，可以采用forward命令。这个命令指定一系列站点，系统要求你将转发作业交给这些站点去完成。

uchile的forward-to条目是非常必要的，因为它返回的任何一个文件事实上都会传递到Pablo。如果没有这个条目， UUCP就会将这些文件丢弃。这个条目采用的是 forward命令的变体，允许uchile通过seci，只把文件发给Pablo；不得发向别的系统。

如果想把文件转发到所有的系统，可采用一个特殊的关键字 ANY（必须采用大写）。

11.5 如何设置拨入

如果你想将自己的站点设置为拨入，必须允许用户登录到你的串行端口，并自定义一些系统文件，提供UUCP账号。这就是我们这一小节的主题。

11.5.1 设置getty

如果打算将串行线路用作拨入端口，就必须在这个端口上启用一个 getty进程。但是，有

些getty实施的确不适合拨入，因为人们通常想采用同一个串行端口进行拨入和拨出。因此，必须保证你使用的getty进程能够和其他程序（比如uucico和minicom）共享同一条线路。具有此功能的一个程序是getty_ps包内的uugetty。许多软件中都有这个程序，在/sbin目录下，可找到它。我所知道的另一个程序是mgetty，它是Gert Doering编写的，同样具有这一功能。可从sunsite.unc.edu获得其最新版本。

至于uugetty和mgetty在控制登录方面的不同，不在本书讨论之列。有兴趣的读者，可参考Greg Hankins编写得《Serial HOWTO》，以及getty_ps和mgetty自带的文档。

11.5.2 提供UUCP账号

接下来，必须设置的是用户账号，有了账号，远程站点才能登录到你的系统，建立 UUCP连接。一般说来，要为每个拨入系统提供一个独立的登录名。在为 Pablo系统设置账号时，可将Upablo作为它的用户名。

对于通过串行端口拨入的系统，通常情况下，你都必须将其账号添加到系统密码文件/etc/passwd中。最好把所有的UUCP登录信息统统归为一个特殊的组，比如 uuguest。系统账号的根目录应该设为公用假脱机目录 /var/spool/uucppublic；其登录外壳脚本必须是uucico。

如果系统中已经安装了影子密码套件，就可以用 useradd命令来设置账号：

```
# useraddr -d /var/spool/uucppublic -G uuguest -s /usr/lib/uucp/uucic
```

如果没有安装影子密码套件，则需要手工编辑 /etc/passwd。即增加下面这一行，其中5000和150是数字化的uid和gid，分别分配给用户Upablo和组uuguest。

```
Upablo;x:5000:150:UUCP Account:/var/spool/uucppublic:/usr/lib/uucp/u
```

安装账号之后，必须激活它。也就是说利用 passwd命令，将其设置为密码。

针对通过TCP连接到你的站点上的UUCP系统，如果要为它们提供服务，必须设置 inetd，令其处理进入uucp端口的连接。在/etc/inetd.conf内增加下面这一行即可：

```
uucp  stream  tcp  nowait  root  /usr/sbin/tcpd  /usr/lib/uucp/uuc
```

注意 通常情况下，tcpd有700模式，所以你必须以root用户的身份来调用它，这一点和uucp是不同的。

-1选项令uucico执行其本身的登录身份验证。 uucico将像标准的登录程序一样，提示登录名和密码，但依据是自己的私用密码数据库，而不是 /etc/passwd。这个私用密码数据库文件名为/usr/lib/uucp/passwd，其中包含成对的登录名和密码，比如：

```
Upablo IslaNegra
Ulorca co ' rdoba
```

当然，这个文件的拥有者必须是uucp，其保护模式为600。

如果你认为这个数据库不错的话，还可将其用于普通的串行登录。

首先，你需要泰勒式UUCP 1.05，因为它允许getty利用-u选项，将拨号用户的登录名传递给uucico（1.04版本中也有-u选项，但它只是一个no-op）。然后，必须让你正在使用的 getty，调用uucico，而不是常用的/bin/login。有了getty_ps，就可以通过在配置文件内设置 LOGIN选项来达到这一目的。但是，与此同时，也取消了交互式登录对话。另一方面， mgetty有一个讨人喜欢的特性，允许你根据用户提供的登录名，调用不同的登录命令。比如，你可以要求

mgetty针对特定的所有用户（他们提供的登录名统统以大写的 U开头）使用uucico，而个别用户则采用标准的登录命令。

要保护自己的UUCP用户，防止恶意呼叫者指定一个假系统名，中途拦截那些用户的所有邮件，你应该在sys文件内的每一条系统条目中，增加 called-login命令。

11.5.3 预防措施

关于UUCP，最大的问题之一是呼叫系统可以采用假名；它在登录之后，向被呼叫系统声明自己的名字。因此，攻击者可能登录到他／她自己的 UUCP账号，假装某某人，中途截取其他站点的邮件。如果你提供通过匿名 UUCP的登录，这个问题就严重了，因为，这样的话，登录密码几乎是公开的。

除非你知道自己信得过呼叫自己的那些站点，不然，你必须采取适当的措施，来防范那些欺骗者。具体的做法是在 sys中指定一个called-login，要求每个系统都使用一个特定的登录名。

这样做的结果就是，一旦某个系统登录进入，并伪称自己是 Pablo，那么uncico就会查实它是否曾以Upablo的身份登录过。假如没有，那么呼叫的系统就会被拒绝，并取消此次连接。大家应养成为增加到 sys文件的每个系统条目，增加一个 called-login命令的习惯 —— 无论它们是否曾经呼叫过你的站点。对那些从未呼叫过你的站点，或许也应将 called-login设为某些明显是伪造的用户名，比如neverlogsin等等。

11.5.4 呼叫序列号检查

要想防范网络骗子，有效地探测那些虚假的登录，另一个办法是利用“ 呼叫序列号检查 ”。利用呼叫序列号检查，可防范非法闯入者找出我们登录进入 UUCP系统时使用的密码。

在使用呼叫序列号检查的情况下，双方的机器都会追踪并维护着迄今为止建立过的连接数量。每建立一次连接，这个序列号都会递增 1。登录进入后，呼叫者需发出它的呼叫序列号。同时，被呼叫者需要将其与自己的号码核对。假如两者不同，连接请求便会被拒绝。因此，假如初始的编号是随机选定的，那么攻击者很难猜出正确的呼叫序列号是多少。

除此以外，呼叫序列号还能做更多的事情：即使某些“ 异常聪明 ”的人能侦测出你的呼叫序列号，同时找到你使用的密码，我们也能及时地发现这一情况！一旦攻击者调用了你的 UUCP feed，并偷走你的邮件，那么 feeds调用序列号就会增 1。我们下一次调用自己的 feed，并试图登录时，远程的uucico就会拒绝登录，因为数字（编号）已不再相符了！

假如启用了呼叫序列检查功能，便应定期检查自己的日志文件，找出那些可能是由于外来攻击而造成的错误消息。假如你的系统拒绝由呼叫系统提供的一个序列号， uucico就会在日志文件中记录一条错误消息，类似于“ Out of sequence call rejected ”（序列调用被拒绝）。假如由于序列号不同步，而造成 feed拒绝你的系统登录，它就会在日志文件中写入一条错误消息，注明“ Handshake failed(RBADSEQ) ”（握手失败）。

接下来，我们必须实际地创建文件，文件中包括序列号本身。 Taylor UUCP将序列号保存在一个名为.Sequence的文件中。该文件放在远程站点的 spool目录下。注意该文件必须由 uucp所有，而且必须采用模式 600（亦即只能由 uucp读写）。最好用一个任意的值对这个文件进行初始化。否则的话，攻击者就有可能轻松地猜出序列号，并用小于它的所有值进行尝试（比

如小于60)。

当然，远程站点也必须允许呼叫序列检查。而且在最开始的时候，采用和我们指定的完全一样的序列号。

11.5.5 匿名UUCP

如果要为系统提供匿名UUCP访问权限，首先必须设置一个特殊的账号（原因如前所述）。一种常见的做法是将它的登录名和密码都设为 uucp。

除此以外，必须针对那些未知的系统，设定几个安全防护选项。举个例子来说，或许应禁止它们在我们的系统上执行任何命令。但是，我们不能在 sys文件条目中设置这些参数。这是由于system命令要求用到系统的名字，但这个名字是我们所没有的。 Taylor UUCP解决这个问题的办法是使用 unknown命令。可在config文件中使用该命令，指定那些通常可在一个系统条目中出现的任何命令。

这样一来，我们就可以限制未知的系统从 pub目录中下载任何文件，并能禁止它们将文件上载到/var/spool/uucppublic目录下的incoming目录中。下一行可令uucico忽略来自远程系统的、从而打开本地调试功能的任何请求。最后两行则允许未知的系统执行 rmail；但是，指定的命令路径将令uucico只在一个名为anon-bin的私用目录内查找rmail命令。这样一来，你就可以提供某一特殊的rmail命令，将所有邮件转发到超级用户那里进行检查。这样便允许匿名用户抵达系统维护人员那里，与此同时，还防止他们将邮件投入其他站点。

要启用匿名UUCP，必须在config文件内至少指定一条未知语句。否则， uucico就会拒绝任何一个未知系统。

11.6 UUCP低级协议

为了与远程终端协商会话控制和文件传输， uucico采用了一个标准化的消息集。这就是通常所说的“ 高级协议 ”。在初始化阶段和挂断阶段，这些消息被作为一系列字串得以传输。但是，在真正的传输阶段，采用的就是低级协议，这些低级协议对高级协议来说，大部分是透明的。这样很方便进行错误检查，特别适用于使用不太可靠的传输线路的情况。

11.6.1 协议概述

由于UUCP使用在不同类型的连接上，比如串行线路和 TCP，甚至于X.25，所以需要采用某些特殊的低级协议。另外，几个UUCP实施方案已介绍了不同的协议。

协议大致可分为两类：流式传输协议和面向包的传输协议。近来的协议通过计算校验和的方式，将一个文件作为一个整体进行传输。这几乎没有任何开销，但要求有一条非常可靠的连接，因为任何一个错误都可能导致整个文件被重新传输。这些协议常用于 TCP连接，但不适于电话线连接。尽管如今的 modem的确能够及时纠错，但仍然不够理想，不能对你的计算机和modem之间存在的错误进行侦测。

另一方面，面向包的传输协议则是将文件分为若干个相同大小的数据包。每个包单独进行收发，单独计算校验和，并向发送方返回收到确认。为了使这类协议更为有效，人们还发明了滑动窗口协议，极大地减少了 uucico在文件传输过程中，必须等候的时间。与流式传输协议相比，包传输协议的开销较大，所以它不太适合于 TCP连接。

数据路径的宽度也有所不同。有时，根本不可能通过串行连接发送 8位字符。比如，如果连接通过一个反应迟钝的终端服务器，就会出现这种情况。此时，传输过程中，带有 8位字符集的字符必须用引号表示出来。在通过 7位连接传输 8位字符时，这种情况被视为是最糟糕的；因为它增加了一倍的数据传输量，尽管硬件可以对这些数据进行压缩。可传输任意 8位字符的线路叫作8-bit clean。所有TCP连接和大多数modem连接都属于这种情况。

11.6.2 传输协议的调节

所有协议都允许对数据包的大小、超时等进行调整。通常情况下，默认设置能够保证数据包得以正常传输，但对具体情况而言，可能不是最好的选择。比如， g协议采用的窗口大小是1到7之间，包的大小则是从 64到4 096之间的2的乘幂个字节（包含在开发软件中的多数二进制都默认采用窗口大小在 7到128字节之间的包）。如果你使用的电话线路噪音很大以至于丢包率高达5%，那你就应该减少包的大小，缩小窗口的大小。另一方面，如果电话线路较好，每个128字节的包都发送一个 ACK（收到确认），又太浪费开销了，所以可以将包的大小增大至512，甚至1 024个字节。

泰勒式UUCP提供了一种机制，可满足你的不同需求，即利用 sys文件内的 protocol-parameter命令，对传输参数进行调节。

不同协议的可调节参数和参数名是不相同的。要想得到一份相关的完整列表，请参考泰勒式UUCP源代码中包含的文档信息。

11.6.3 如何选定特殊协议

并不是每个uucico实施方案都能分辨和识别出每个协议，所以在起初的握手阶段，两方的进程都必须同意采用一个常见的协议。主管 uucico向其从属uucico发送Protlist，为其提供一份已获支持的协议列表，从属再从其中挑选一个协议。

根据使用端口的类型（ modem、TCP或直接连接），uucico将组建一个默认的协议列表。对modem和直接连接来说，这份列表通常包含 i、a、g、G和j。对TCP连接来说，则包含t、e、i、a、g、G、j和f。你也可利用protocols命令，改写默认的协议列表，该协议命令可在系统条目和端口条目中指定。

协议列表要求所以输进或输出的连接都要通过这个端口，采用 i、g或G协议。如果远程终端不支持这些协议，连接对话就会失败。

11.7 故障排除

本小节将对UUCP连接可能出现的故障进行描述，并建议从哪里着手查错。但是，这方面的问题太多了，无法一一列举，本文将着眼于一些比较常见的重要问题。

任何情况下，利用-xall启用调试，都可在假脱机目录中的 Debug内查看调试输出。它将有助于你快速识别问题所在。同时，我还发现在 modem没有连接时，它还可帮助我打开 modem。对采用贺氏标准的 modem来说，具体作法是在拨号文件内的 modem对话内，增加ATL1M1OK。

第一步通常是检查所有的文件访问许可是否设置正确。 uucico应该被setuid为uucp，而/usr/lib/uucp、/var/spool/uucp和var/spool/uucppublic内的所有文件都应该属 uucp拥有。另外，

假脱机目录内还有一些隐藏文件（也就是说，其文件名以“ .”开头的文件；此类文件一般不能通过ls命令显示出来），同样也必须归uucp所有。

uucico总是说“Wrong time to call”（呼叫时间有误）：通常意味着sys文件内的系统条目中，没有指定time命令，该命令详细规定了呼叫远程系统的时间，或目前禁止呼叫你指定的系统。如果不指定呼叫时间表，uucico就会假定从不呼叫该系统。

uucico总是抱怨站点被锁：这通常意味着uucico侦测到远程系统的/var/spool/uucp内有一个锁文件。该锁文件可能源于上一次呼叫的已崩溃或已被杀死的系统。但是，更可能存在另一个uucico进程，该进程准备呼叫远程系统，但又凝在对话脚本中。如果这个 uucico进程未能成功连接到远程系统，就应该用挂断信号将其杀死，并删除所有的锁文件。

我能够连接到远程站点，但对话脚本却不能：查看你从远程站点收到的文本。如果它含混不清，可能就是速度方面的问题。否则，就确定它真的与你的对话脚本所希望的行为保持一致。记住，对话脚本的开头是一个expect字串。在你收到登录提示时，就应该发送自己的用户名，但绝不可能在发送密码之前得到密码提示，插入延时。可能是因为你的 modem太慢了的缘故。

我的modem不能拨号：uucico在拨出时，如果你的modem没有指明DTR线路已被激活，可能是因为你没有为uucico指定一个恰当的设备。如果你的modem识别出了DTR，就会利用你可以写入的终端程序进行验证。如果这样做行得通，就在 modem对话开始之即，利用E打开应答。如果在modem对话期间，它没有对你的命令作出应答，就查看自己的 modem，线路速率是否设置过高或过低。如果有了应答，就看你是已经取消 modem响应，还是将它设置为数字代码。最后再验证对话脚本本身是否正确。记住，你必须写入两个后斜杠，并将其中之一发给modem。

我的modem在拨号时，总不能成功：在电话号码中插入延时。这样做是非常有用的，特别是通过公司内部电话网拨出时。

日志文件说我的包丢失率非常之高：这看来是速度问题。计算机和 modem之间的连接是不是太慢（记住将其速率调整为最高）？或你的硬件速度太慢，以至于不能及时提供服务？如果你的串行端口采用的是NSC-16550A芯片，38Kbps这个速率是比较理想的；但是，如果没有FIFO（比如16450芯片），9 600bps就是极限了。另外，你还应该保证串行线路上已经启用硬件握手。

另一个可能存在的问题是端口上未启用硬件握手。泰勒式 UUCP 1.04没有打开硬件握手这一规定。

我能够登录，但不能握手：产生此类故障的原因很多。日志文件中的输出应该能够给出一大堆原因。看看远程站点提供的协议是什么（在握手期间，它发送了一个 Pprotlist字串）。但也许日志文件内什么东西也没有（如果你没有在 sys或端口文件内选定协议的话）。

如果远程系统发回RLCK，说明远程系统上有一个很早以前的锁文件。如果因为你已经通过另外的线路连接到远程系统，它没有与你握手的话，就可要求将它删除。

如果它发回RBADSEQ，另一个站点就会为你启用对话计数检查，但数字不符。如果它发回RLOGIN，就说明你不能通过这个ID进行登录。

11.8 日志文件

在编辑UUCP套件，使其能够采用泰勒式登录时，只有三个日志文件，这三个日志文件都

驻留在假脱机目录内。主要的日志文件名为 Log，与已建立连接和已传输文件相关的所有信息统统包含在这个文件内。

下一个比较重要的日志文件是 Stats，它列出了文件传输特性。

第三个日志文件便是 Debug。该文件内主要保存一些调试信息。使用调试时，应该保证这个文件的保护模式是 600。根据你所选定的调试模式，该文件中可能包含用于连接远程系统的登录和密码信息。

已发布的Linux产品中，有的UUCP二进制文件已被编辑为采用 HDB式记录。HDB UUCP采用了大量的日志文件，这些文件保存在 /var/spool/uucp/.Log目录下。这个目录中包含另外三个子目录，名为uucico、uuxqt和uux。这三个子目录中包含每条相应命令生成的记录输出，而且这些输出被保存在各个站点的不同文件内。因此，在拨叫 pablo站点时，uucico的输出将记入.Log/uucico/pablo文件，而后来uuxqt的运行结果将被写入 .Log.uuxqt/pablo。但是，写入不同日志文件的各行输出和泰勒式记录的一样。

在你启用HDB式记录的调试输出时，它将被写入 /var/spool/uucp目录下的.Admin目录。在拨出期间，调试信息也会被发送到 .Admin/audit.local中，在有人拨入时，uucico的输出结果将被记入.Admin/audit。

第12章 电子邮件

自第一个网络诞生以来，网络的主要用途便是收发电子邮件。最初，电子邮件只是一项简单的服务，把一台机器上的文件复制到另一台机器，并把它添加到接收端的邮箱文件内。尽管网络随着其复杂多样的路由要求和信息量的日益增长，要求更精心的设计，但电子邮件的基本原理始终是差不多的。

目前，邮件交换标准种类繁多。因特网上的站点采用的标准包含在 RFC-822内，后来又增加了一些RFC，说明如何采用与机器无关的方式，传输特殊字符等等。另外一些标准则针对目前的“多媒体邮件”传输，用于处理包含在多媒体邮件中的图像和声音。标准 X.400是由CCITT（国际电报电话咨询委员会）制定的。

目前，针对Linux系统，已有相当多的邮件传输程序得到了实施。其中最有名的是伯克利分校的sendmail，它适合于大量的平台，作者是 Eric Allman，目前，Eric又回到sendmail小组进行下一阶段的开发。现在，sendmail 5.56c有两个移植版本，其中一个将在第 14章介绍。目前正处于开发阶段的 sendmail版本是 8.9.3，有关详情，可参考 http://send.mail/org和http://sendmail.com。

常和Linux一起使用的邮件代理是smail 3.1.28，Curt Landon Noll和Ronald S.Karr编写并申请版权。下面，我们将其简称为 smail。虽然还有许多截然不同的邮件代理，但我们不打算对它们进行讨论。

与历史悠久的sendmail相比，smail显得尚不成熟。在处理小型站点的邮件信息时，由于此类站点没有复杂的路由要求，所以其过人之处表现不出来。但对于大型的站点，sendmail是绝对的赢家，因为它的配置方案要灵活得多。

smail和sendmail都支持一个必须自定义的配置文件集。除了能使邮件子系统正常运行的基本信息外（比如主机名），还有大量需要调整的参数。最初，sendmail的主要配置文件非常难以理解。它看起来就像是你的猫一直按住 Shift键，在你的键盘上小睡一样。和 sendmail的相比，smail配置文件的结构要好得多，而且易于理解，但用户不能调整邮件发送机的行为。然而，对小型的UUCP和因特网站点来说，两个配置文件的设置过程几乎是一样的。

本章，我们将为大家讲解何为邮件消息，以及作为管理员必须执行哪些操作。第 13章和第14章将指导大家初次设置 smail和sendmail。这里提供的信息足以应付小型站点的邮件系统运作，但还有更多、更新鲜的选项，你可以试试它们。

本章最后将简要介绍如何设置 elm，它是许多Linuxish系统上采用的常见邮件用户代理。

关于Linux上特有的电子邮件，其详情可参见 Linux Electronic Mail HOWTO文档，它是Guylhem Aznar(邮件地址guylhem@oeil.qc.ca),位于http://metalab.unc.edu/ LDP/HOWTO/Mail-HOWTO.html。elm、smail和sendmail的源代码也包含在许多文档内，在设置它们时遇到的问题，都可在这些文档内得以解决。如果想查找电子邮件的常见信息，可参考这方面的 RFC。它们列在本书最后的参考书目内。

12.1 何谓邮件消息

邮件消息一般由消息主体（发送者写的文本内容）和指定接收者、传输载体等的特殊数据组成，后者和我们平常看到的信封上的内容相似。

管理性数据分为两类：第一类数据是指与传输媒体相关的所有数据，比如发送人和收信人的地址。因此，这类数据也被称为信封。它可以随邮件消息的投递，通过传输软件进行转换。

第二类数据是处理邮件消息所需的所有数据，它不是某个传输机制特有的，比如说消息的主线，收信人清单和发送消息的数据等。许多网络中，都将其加入邮件消息内，形成所谓的邮件头。这正逐渐成为一个标准。邮件头从邮件主体偏移一个空行（一般习惯于在邮件消息内增加一个签名或.sig，其中通常包含作者信息和一段笑话或作者的座右铭。它从邮件消息偏移含有“ - ”的一行）。

现在，许多邮件传输软件都采用 RFC-822中勾划的消息头格式。其原意是为 ARPANET上的使用制定一个标准，但由于被设计成不依赖于任何一个环境，所以，稍作改动，它就可以适用于其他的网络，包括许多基于 UUCP的网络在内。

但是，RFC-822只是最常见的标准之一。随着日益增长的需求，比如数据加密、国际化字符集支持和多用途Internet邮件扩展（MIME），越来越多的标准也相应出台。

所有这些标准中，邮件头由若干行组成，中间用一个新行字符分开。一行又由一个字段名（从第一列开始）、字段本身，偏移一个冒号和空格字符或制表符组成。字段名不同，相应的字段格式和含义也会有所不同。如果下一行以标签开头，头字段就可能继续出现在下一行。字段的排列顺序是任意的。

一个典型的邮件头可能是这样的：

```
From brewhq.swb.de!ora.com!andyo Wed Apr 13 00:17:03 1994
Return-Path: <brewhq.swb.de!ora.com!andyo
Received: from brewhq.swb.de by monad.swb.de with uucp
        (Smail3.1.28.1 #6) id m0pqqlT-00023aB; Wed, 13 Apr 94 00:17
Received: from ora.com (ruby.ora.com) by brewhq.swb.de with smtp
        (Smail3.1.28.1 #28.6) id <m0pqoQr-0008qhC>; Tue, 12 Apr 94 2
Received: by ruby.ora.com (8.6.8/8.6.4) id RAA26438; Tue, 12 Apr 94
Date: Tue, 12 Apr 1994 15:56:49 -0400
Message-Id: <199404121956.PAA07787@ruby
From: andyo@ora.com (Andy Oram)
To: okir@monad.swb.de
Subject: Re: Your RPC section
```

通常，所有必要的头字段都是由你使用的接口生成的，比如 elm、pine、mush或mailx。但是，有些字段是可选的，也可以是用户添加的。比如， elm允许你对部分消息头进行编辑。其他的则由邮件传输软件添加。表 12-1列出了部分常见的头字段及其含义。

表12-1 常见头字段及其含义

From	其中包含发件人的邮件地址和真名
To	收件人的邮件地址
Subject	以简短的几句话，说明邮件内容。至少说明其主题
Date	发件日期
Reply-To	指定地址，发件人要求收件人直接回复到这个地址。如果你有若干个账户，但常用的只有一个时，这个字段便特别有用。该字段可选

(续)

Organization	发邮件的这台机器所属公司。如果是私有的，就可以令其为空，或插入“私用”。该字段可选
Message-ID	发件系统上邮件传输生成的一个字串，是该邮件消息独有的
Received	对你的邮件进行处理的每个站点（包括发件人和收件人的机器）都会将这一字段插入邮件头，给出其站点名、消息 ID、以及它收到该消息的时间和日期、该消息来自的站点，所使用的传输软件等等。这样一来，你就可以跟踪邮件消息经过的路径。假如发现问题，还可据此向出问题的那一部分的管理员投诉
X-anything	假如邮件头以X-字样开头，那么任何邮件相关程序都不能对它进行苛刻的检查。它用实现一些特殊的、附加的特性，这些特性尚未在 RFC里正式实施，或者根本就不打算制订到 RFC规范里。例如，Linux Activists邮件列表便采用了这种形式，频道（channel）由“X-Mn-Key:”头字段选择

该结构的一个例外是它的第一行。它以关键字“ From ”开头，后面跟一个空格或制表符，而非冒号。为将其与普通的 From:字段明确区分开，通常也把它写作 From_。其中包含了该邮件经过的路由——采用 UUCP的bang-path风格（下面还会详细解释）；包含了最后一台机器收到它的时间和日期；并包含了一个可有可无的部分，指出该邮件是从哪个主机收到的。由于该字段会由处理过该邮件的每一个系统重新生成一遍，所以有时会在信封数据的下面进行一番小计。

之所以要保留这个 From_字段，是为了保持与某些较老的邮件发送程序的“向后兼容”。但实事求是地说，它真正的用处并不大，只是邮件的用户接口必须依赖它，在用户的邮箱中，标注出一封信件的开头。要注意的是，邮件正文有时也会以“ From ”起头，所以为避免与之冲突，通常应该在其前面加一个换码字符“ > ”。

12.2 邮件如何发送

通常，邮件的编写是用 mail、mailx之类的邮件程序接口来完成的，也有人用更为复杂的邮件程序接口，比如elm、mush和pine。这类程序称为“邮件用户代理”，简称MUA。如果需要发送一条电子邮件消息，这个接口程序大多会将其交给另一个程序进行投递。这就是所谓的“邮件传输代理”，简称MTA。有些系统上，本地投递和远程投递采用的邮件传输代理是不一样的；而其他一些系统上，则可能采用同一个代理。远程投递所用的命令通常称为 rmail，本地投递所用的则称为lmail（如果有的话）。

当然，本地邮件的投递，不仅仅是把收到的消息添加到收件人的邮箱内。通常情况下，本地MTA可以识别别名（设置本地收件地址，使其指向另一个地址），和转发邮件（将用户的邮件重新定向到其他目的地）。另外，无法投递的邮件则必须返回，也就是说，随错误消息一起返回发件人。

对于远程邮件的投递，所用的传输程序和链接的属性有关。如果邮件必须在一个使用TCP/IP的网络上传递，则常用 SMTP。SMTP即简单邮件传输协议，其定义见 RFC-788和RFC-821。SMTP通常直接链接到发件人的计算机，和远程主机的 SMTP daemon（程序）协商消息的传输。

在UUCP网络中，邮件的投递一般不会直接进行，而是通过大量的中间系统，转发到目标主机。要想通过UUCP链接发送邮件，发送端的MTA在转发系统上利用uux，执行rmail，并将作为标准输入的邮件消息发给这个转发系统。

由于这一切是专门针对每条邮件消息来进行的，因此，可能使主邮件中间站负荷过多，

占据着大量磁盘空间的上百个文件将导致UUCP假脱机队列发生混乱（这是因为磁盘空间一般分为1024个字节的若干个数据块。即使一条消息只有 400个字节，也会占据1KB的空间）。因此，有些MTA允许你把准备发给远程系统的若干条消息统统装入一个单独的批处理文件内。这个批处理文件内包含的是SMTP命令，如果采用直接SMTP链接，本地主机一般会执行这些命令。这就是所谓的BSMTP或批处理SMTP。然后，批处理文件被发给远程系统上的 rsmtp或bsmtp程序，这样一来，远程系统就像对待一条普通的 SMTP链接一样，对这一批处理文件进行处理。

12.3 邮件地址

至于电子邮件，其地址由处理邮件的计算机名和能够为其系统识别的用户身份信息组成。后者可以是收件人的登录名，也可以是别的信息。其他邮件地址定址方案，比如 X.400，采用的是一个常用的“ 属性 ”集，用于在X.500目录服务器内查找收件人的主机。

计算机名的解释（也就是说，你的邮件消息最终将在哪个站点停留）和将这个计算机名和收件人的用户名组合在一起的方式和你所处的网络有极大的关系。

RFC-822标准中，规定因特网站点采用user@host.domain表示法，host.domain表示主机的完整资格域名。中间的@叫作“ at ”。由于这种表示法不涉及到目标主机的路由，而是给出主机名（唯一性的），所以这就叫作绝对地址。

在最初的UUCP环境中，比较流行path! host! user这种表示法，其中“ path ”表示消息在抵达目标主机之前，必须经历的一系列主机。这种结构叫作 bang路径表示法，因为感叹号（!）就叫作一个“ bang ”。如今，许多基于UUCP的网络都有经过修正的RFC-822，都能识别这类地址。

目前，这两类地址还不能混用。以 hostA!user@hostB这个地址为例，它不能清楚地说明“ @ ”符号优先于路径，还是路径优先于这个符号：是必须将邮件消息发给 hostB，再由其投递到hostA!user呢，还是将其发给hostA，再由它转发到user@hostB？

同时采用这两类地址符号的地址称为混合地址（hybrid address）。最典型的就是前面那个地址。一般把它解释为“ @ ”符号先于路径。因此，前面的地址意思是先向hostB发送邮件消息。

然而，RFC-822中，还有一种指定路由定址的方法：<@hostA,@hostB:user@hostC>表示主机hostC上的用户地址，hostC指的是通过hostA和hostB（按这个顺序），最后到达的目标主机。这类地址通常称为“ route-addr ”地址。

此外，还有“ % ”地址符：user%hostB@hostA，邮件消息首先被发送到hostA，再由后者将最右面的（这里，没有最右面的主机了）百分号沿伸到“ @ ”符号。现在的地址就变成了user@hostB，邮件程序将你的消息转发到hostB，后者再将其投递给相应的用户。这类地址有时被称为“ Ye Olde ARPANET Kludge ”，一般不鼓励大家采用这类地址。但事实上，许多邮件传输代理都会产生这类地址。

其他网络中仍然存在不同的定址方案。比如，基于 DECnet的网络，它采用两个冒号来作为分隔符，生成这样的地址：host::user（从RFC-822环境，试着抵达DECnet地址时，可能会用到host::user@relay，“ relay ”代表大家熟知的Internet-DECnet relay）。最后，X.400标准利用的是另一种截然不同的定址方案，它用若干个属性 -值对组成的集合来描述收件人，比如国家、公司等。

FidoNet上，每个用户是用诸如2:320/204.9此类的代码识别的，它由四个代表区域的号码(2代表欧洲)、网(320代表巴黎)，节点(本地hub)和端点(各个用户的计算机)组成。Fidonet地址可以映射为RFC-0822内的地址；上面的地址可写成Thomas.Quinot@p9.f204.n320.z2.fidonet.org。看来，还)是域名好记呀！

使用这些不同类型的定址方案时，有几点需要注意，详情情况将在随后的几个小节内进行讨论。但是对RFC-822环境来说，最好采用user@host.domain之类的绝对地址，其他的则不常用。

12.4 邮件路由的工作原理

将邮件消息导向到收件人主机的过程就叫作路由。除了找出发送站点到目标站点的路径外，路由还涉及到错误检查和传输速度和成本的优化。

UUCP站点和因特网站点处理路由的方式是有很大区别的。因特网上，将邮件数据导向收件人主机（只要能得知其IP地址）的主要任务是由IP层来完成的，而UUCP区内，路由必须由用户提供或由邮件传输代理生成。

12.4.1 因特网上的邮件路由

因特网上，完全依靠目标主机来执行邮件路由的选择。默认情况是通过查找目标主机之IP地址，将消息直接投递给它，把真正的路由数据留在IP传输层。

对许多站点来说，都想将所有封装好的邮件导向一个具有较强处理能力的邮件服务器，这类服务器能够对所有的邮件通信进行处理，并在本地就开始进行分发。为宣布这项服务器，站点须在DNS数据库内，为其本地域出版一条所谓的“ MX ”记录。MX代表“ 邮件交换者 ”，基本上可表示服务器主机愿意充当该域内所有主机的邮件转发者。 MX记录还可用于处理主机通信，这些主机本身没有接入因特网，比如UUCP网络或公司内网内用于传输机密信息的主机。

同时，MX记录还有一个与之关联的优先级。这个首选项是一个正整数值。如果一个主机对应若干个邮件交换者，邮件传输代理就会试着将邮件消息传给优先级最低的交换者，失败之后，再尝试优先级较高的交换者。如果本地主机本身就是目标地址的邮件交换者，它就会禁止向任何一台优先级高于它自己的MX主机转发邮件；其目的是为了避免邮件循环。

我们以Foobar公司为例。该公司打算让一台名为mailhub的机器来处理他们的邮件。所以，他们的DNS数据库内就有一条这样的MX记录：

```
foobar.com IN   MX 5 mailhub.foobar.com
```

该记录宣布mailhub.foobar.com作为foobar.com的邮件交换者，其优先级为5。希望将邮件消息投递给joe@greenhouse.foobar.com的主机，将在DNS数据库内查找foobar.com，并发现这条MX记录位于mailhub。如果没有找到优先级低于5的MX，这条邮件消息就会投递到mailhub，然后，再由后者将其分发到greenhouse。

上面的例子只是简单说明了MX的工作原理。如果想更多地了解因特网上的邮件路由，请参考RFC-974。

12.4.2 UUCP网络内的邮件路由

UUCP网络内的邮件路由比因特网上的复杂得多，因为传输软件自身是不会执行任何路由

的。早期，所有的邮件都必须利用 bang路径进行寻址。bang路径指定一系列邮件必须经由的主机，中间用感叹号隔开，最后才是用户名。如果你想为发给 Janet的邮件寻址，而她是Moria机器的用户，就应该采用这条路径： eek!swim!moria!janet。这样，就会把邮件从你的主机发送到Eek，再由Eek发给Swim，最后再投递到Moria。

这种路由的不足是它要求你必须记住网络拓扑、必须有快速链接等等。更糟糕的是，一旦网络结构发生了变化，比如链接已被删除或主机已发生增减，都可能导致邮件投递失败，究其原因，仅仅是你不知道这一变化。最后，如果你搬到另一个地方，很可能将不得不更新所有路由。

采用源路由的另一个必要原因是主机名指代不明。比如，我们以两个名为 moria的站点为例，一个位于美国，另一个位于法国。 moria!janet指的究竟是哪个站点呢？只有指定抵达moria的路径之后，才能得知。

解决主机名指代不明的第一步是：成立 UUCP映射工程。它位于 Rutgers大学，对所有正式的UUCP主机名及其UUCP邻居和地理位置进行注册，以保证主机名不会发生雷同。映射工程收集的这些信息作为 Usenet Maps出版，并定期通过 Usenet向外发布（用 UUCP映射工程注册的站点映射信息则通过新闻组 comp.mail.maps发布；其他一些组织可能也会发布自己的网络映射信息）。一个典型的映射系统条目（在删除批注之后）如下所示：

```
moria
    bert(DAILY/2),
    swim(WEEKLY)
```

这个条目是说Moria分别有一条指向Bert和Swim的链接，前者每天拨打前两次，后者则是一周一次。接下来详情讲解映射（map）文件的格式。

利用映射中提供的连接信息，便可自动生成从主机到任何一个目标站点的完整路径。这些信息通常都保存在 paths（路径）文件内，该文件有时也称为路径别名数据库（ pathalias database）。以通过Ernie抵达Bert的连接映射状态为例，从前面摘录的映射针对 Moria生成的路径别名如下：

```
moria ernie!bert!moria!%s
```

现在，如果你指定目标地址 janet@moria.uucp，你的MTA就会从上面选出一条路由，并采用bert!moria!janet这个地址向Ernie发送消息。

但是，根据 Usenet映射构建路径文件并不是上策。因为映射中提供的连接信息通常变动很大，而且有时已经过时。因此，只有主要主机才会采用完整的 UUCP全球映射来构建自己的路径文件。许多站点只是在其邻近维护自己的路径信息，将目标站点不在其数据库内的邮件消息发送给一个更“ 聪明 ”的主机，后者具有更完整的路径信息。这种方案称为“ 智能主机路由 ”（smart-host routing）。只有一条UUCP邮件链接的主机（也就是所谓的叶子站点）没有自己的路由信息；完全依靠它们的智能主机进行路由。

12.4.3 UUCP和RFC-822

迄今为止，要解决UUCP网络内邮件路由问题，最好的办法是修改 UUCP网络内的域名系统。当然，你是不能在 UUCP网络上查询域名服务器的。但不管怎样，许多 UUCP站点都有与

其内部路由对应的小型区域。在这些映射中，这些域声明了一台或两台主机作为自己的邮件网关，所以域内的每台主机都不必有自己的映射条目。网关对流入或流出该域的所有邮件进行处理。该域内部的路由方案在外界看来，是不可见的。

这对前面提到的智能主机路由方案来说，也非常有用。全局路径信息只由网关进行维护；该域内的次要主机只负责一个小型的手工写入的路径文件，该文件中列出了域内的路由和指向邮件中转器的路由。甚至于邮件网关，也不必再有针对全球范围内每台独立 UUCP主机的路由信息。对邮件网关来说，除了有完整的、用于自己提供服务的域的路由信息外，只需要在其数据库内有指向整个域的路由。比方说，下面的路径别名条目将把 sub.org域内站点的所有邮件路由到Smurf：

```
.sub.org swim!smurf!%s
```

地址是claire@jones.sub.org的所有邮件都将投到swim，后者的地址是smurf!jones!claire。

域名空间的分层式结构允许邮件服务器采用多种特定路径。例如，位于法国的一个系统，可能有用于“ fr ”子域的特定路由，但它会把目标为“ us ”域内的主机的所有邮件路由到美国的特定系统。通过这种方式，基于域的路由将大大减少路由数据库的规模并减少路由管理上的开支。

但是，在UUCP环境内采用域名的好处主要是顺应 RFC-822的需要，后者要求 UUCP网络和因特网之间必须容易沟通。目前，许多 UUCP域都有一条具有因特网网关的链接，由该网关充当其智能主机。通过因特网发送消息更为快捷，而且路由信息也更为可靠，因为因特网主机可用DNS来代替Usenet映射。

对基于UUCP的域来说，为了邮件能通过因特网抵达自己域内的主机，它们通常都有自己的因特网网关为其声明一条 MX记录。比如，假设 Moria属于orcnet.org域。Gcc2.groucho.edu充当其因特网网关。因此， Moria将采用gcc2作为自己的智能主机，发到域外的所有邮件都将通过因特网得以投递。另一方面， gcc2将为orcnet.org域声明一条MX记录，并将目标为orcnet站点的所有进入邮件都投递到 Moria。

现在，唯一的问题是 UUCP传输程序不能处理完整资格域名。大多数 UUCP站点的站点名都限制在8个字符之内，有的甚至更少，而且是不包括文字或数字的名字。多数情况下，点 . 是绝不可用的。

所以，需要一些映射进行RFC822名和UUCP主机名之间的转换。映射方法与实施方案有关。对FQDN（完整资格域名）和UUCP主机名之间的映射来说，常见方法是利用路径别名文件：

```
moria.orcnet.org  ernie!bert!moria!%s
```

该文件将根据指定完整资格域名的一个地址，产生一个纯粹的 UUCP样式的bang路径。有些邮件服务器会为此提供一个特殊文件；比如 sendmail采用的就是uucpxtable。

从UUCP网络向因特网发送邮件时，有时还需要逆向转换（俗称 domainizing，定域）。只要发件人在其目标地址中采用的是完整资格域名，这个问题便可迎刃而解。作法是：在把邮件消息转发给智能主机时，不要从信封地址中删除域名。但是，还有些 UUCP站点不属于任何一个域。它们通常会通过增加伪域 UUCP的方式，进行定域。

12.5 路径别名和映射文件格式

路径别名数据库提供基于 UUCP网络内的主要路由信息。典型的条目如下（站点名和路径

用制表符隔开）：

```
moria.orcnet.org      ernie!bert!moria!%s
moria                 ernie!bert!moria!%s
```

该条目使发给Moria的所有邮件消息都通过Ernie和Bert进行投递。对Moria的完整资格域名和其UUCP主机名来说，如果邮件服务器无法单独将它们映射为对应的域名空间，这两者都是必须指定的。

如果打算把目标为某域内主机的所有邮件消息都导向其邮件中转器，还需要在路径别名数据库内指定一条路径，给出作为目标域的域名，前面加上点 .。比如，如果sub.org域内的所有主机都可通过swim!smurf抵达，其路径别名条目可能会像这样：

```
\&.sub.org swim!smurf!%s
```

只有运行的站点不需要太多路由时，才可编写路径别名文件。如果必须为大量的主机进行路由，最好采用pathalias命令，根据映射文件创建一个别名文件。映射的维护要简单得多，因为只须通过编辑系统的映射条目，重建映射文件，便可增加或删除系统条目。尽管 Usenet映射工程出版的映射不再适用于路由，但小型的 UUCP网络则可能在自己的映射集内提供路由信息。

映射文件主要由站点列表组成，该列表列出了每个系统轮询或被其轮询的站点。系统名始于第一列，然后是用句点隔开的链接列表。如果新行以制表符开头的话，这个列表可能会持续若干行。每个链接则由站点名，和紧随其后的双括号中的成本组成。这个成本是一个算术表达式，由数字和象征性的成本值构成。以 #号开头的行则忽略不计。

以Moria例，它每天对swim.twobirds.com轮询两次，对bert.sesame.com每周轮询一次。而且，指向bert的链接只采用了一个非常慢的2 400bps的modem。Moria公布的映射条目如下：

```
moria.orcnet.org
        bert.sesame.com(DAILY/2),
        swim.twobirds.com(WEEKLY+LOW)

moria.orcnet.org = moria
```

最后一行将令其知晓其UUCP主机名。注意，它必须是DAILY/2（一天2次），因为一天拨打两次事实上将使该链接的成本减半。

对路径文件内列出的任何一个目标站点来说，利用源于此类映射文件的信息， pathalias命令能够计算出到这些目标站点的最优路由，并根据这些路由的信息，生成一个路径别名数据库。

pathalias还提供了另外两个特性：比如隐藏站点（也就是说，令站点只能通过网关进行访问）等等。关于pathalias命令以及链接成本的完整列表，可参见相关的手册页。

映射文件的批注中，一般还包含站点说明。指定批注的格式非常严格，所以一般从映射文件中获取。比如说，一个名为 uuwho的程序可以利用根据映射文件建立的数据库，以比较清晰的方式显示出这些信息。

在将自己的站点注册为一个组织（把映射文件分发给其成员）时，一般都必须找出这样的映射条目。

下面是一个映射条目示例（事实上，取自我自己站点）：

```
#N      monad, monad.swb.de, monad.swb.sub.org
#S      AT 486DX50; Linux 0.99
#O      private
#C      Olaf Kirch
#E      okir@monad.swb.de
#P      Kattreinstr. 38, D-64295 Darmstadt, FRG
#L      49 52 03 N / 08 38 40 E
#U      brewhq
#W      okir@monad.swb.de (Olaf Kirch); Sun Jul 25 16:59:32 MET DST
#
monad   brewhq(DAILY/2)
# Domains
monad = monad.swb.de
monad = monad.swb.sub.org
```

前两个字符之后的空格字符是制表符。条目中许多字段的含义都是相当清楚的；可从你注册的域收到一分详细说明。L字段最有趣；它给出了你所处位置的经度／纬度，并用于制定PostScript地图，显示各个国家的所有站点乃至全世界的每个站点（要知道，它们的数目是相当惊人的，定期在news.lists.ps-maps公布）。

12.6 elm的配置

elm代表“ electronic mail ”（电子邮件），是最理想的命名工具之一。它提供了一个非常有用的全屏接口。我们不打算讨论elm的用法，但将详细讨论其配置选项。

理论上讲，可以运行未经配置的elm，如果运气好的话，一切都会正常运行。但有几个选项是必须设置的，尽管它们用的地方较少。

启动时，elm读取elm.rc文件内的配置参数集，该文件位于 /usr/lib/elm。然后，再试着读取根目录下的.elm/elmrc文件。通常，这个文件不需要你自行编写。你从 elm的“ 选项 ”菜单中，选定“ Save ”选项，便可建立它了。

私用elmrc文件的选项集也可用于全局 elm.rc文件。你自己的私用 elmrc文件中的多数设置优先于全局文件内的设置。

12.6.1 全局elm选项

全局elm.rc文件中，必须设置和你的主机名相关的选项。比如，在 VirtualBrewery公司网络内，vlager主机所用的全局elm.rc文件就应该包含下面的内容：

```
#
# The local hostname
hostname = vlager
#
# Domain name
hostdomain = .vbrew.com
#
# Fully qualified domain name
hostfullname = vlager.vbrew.com
```

这些选项设置了elm的本地主机名。尽管该信息较少使用，但仍应该设置这些选项。注意，

只有在全局配置文件内设置它们时，这些选项才会生效；如果是在你自己的私用 elmrc文件内配置的，它们就会被忽略。

12.6.2 国家特有字符集

近来，人们一直期望对 RFC-822标准进行修改，使之能支持诸如纯文本、二进制数据、PostScript文件等等之类的不同类型的信息。涉及到这些不同类型支持的标准字符集和 RFC常被称为多用途因特网邮件扩展 (Multipurpose Internet Mail Extensions,MIME)。它们使收件人能够得知邮件消息中是否采用了除标准 ASCII之外的其他字符。比如邮件消息中采用了法语重音或德语元音等。这是通过扩展 elm来支持的。

Linux内部用以代表字符的字符集通常称为 ISO 8859-1，这是它的标准名。它也被称为 Latin-1。对采用该字符集内字符的任何一条邮件消息来说，其邮件头都应该有下面这一行：

```
Content-Type: text/plain: charset=iso-8859-1
```

收件系统应该识别出这个字段，并采取相应的措施显示邮件消息。针对文本／纯文本消息的默认设置是us-ascii字符值。

为了能够显示带有 ASCII和其他字符集的邮件消息， elm必须知道如何打印这些字符。默认情况下， elm在收到这样的邮件消息（除了 us-ASCII字符之外，其中还有一个 charset字段，或除了文本／纯文本以外，还有内容类型的消息）时，它将试着利用一个名为 metamail的命令，显示这一消息。需要 metamail命令显示的邮件消息将在屏幕上的带有 M的第一列中出现。

由于Linux原始字符集是ISO-8859-1，所以没必要调用 metamail命令来显示使用了该字符集中字符的邮件消息。如果 elm得知显示端能够识别ISO 8859-1，它就不会采用 metamail命令，而是直接在屏幕上显示邮件消息。这是通过在全局 elm.rc配置文件内设置下面的选项来完成的：

```
displaycharset = iso-8859-1
```

注意，即使在你根本不可能收发除了 ASCII之外其他字符的邮件时，仍然应该设置这个选项。这是因为发送此类邮件的人们通常会对他们的邮件服务器进行配置，通过默认设置，将恰当的Content-Type:field放入邮件头，不管它们是否发送其中只包含 ASCII字符的消息。

但是，仅仅在 elm.rc文件内设置这个选项是不够的。因为，在用其内置的 pager显示邮件消息时， elm将针对每个字符，调用一个库函数来判断该字符是否能够打印。默认情况下，这个函数只能将ASCII认作是可以打印的，而把其他所有字符显示为“ ？”。所以，要解决这个问题，就应该把环境变量LC_CTYPE设置为ISO8859-1，后者将要求库函数把 Latin-1也认作是可以打印的。自 libc-4.5.8以来，已经可以支持这一设置和其他特性了。

在发送的邮件中，如果其中包含取自 ISO 8859-1的字符，就应该保证在 elm.rc文件内设置另外两个变量：

```
charset =  iso-8859-1
textencoding = 8bit
```

这两个变量使 elm在邮件头，把这个字符集当作 ISO-8859-1报告出来，并将它当作一个 8位值进行发送（默认情况是将所有的字符拆为 7位值）。

当然，这些选项还可以在私用的 elmrc文件而不是全局 elmrc文件内进行设置。

第13章　smail的设置和运行

本章打算对 smail的设置和功能进行一番概述。尽管 smail的行为在很大程度上兼容于sendmail，但它们的配置文件却是完全不同的！

主配置文件是/usr/lib/smail/config。针对每个人建立的站点，必须对这个文件进行编辑，以反映出自己的实际情况。但是，假如建立的只是一个 UUCP叶站点，那么要做的事情就简单得多。另外，还有可能要用到对路由及传输选项进行配置的其他文件。本章也会简要地介绍它们。

默认情况下，smail会立即处理和投递输入的所有邮件。但是，假如站点的事务量非常高，也可考虑让smail在队列中先收集好所有消息，再定时进行处理，不一定非要“立即”处理。

在一个TCP/IP网络中对邮件（消息）进行处理时，smail会经常用一种daemon模式运行。换言之，系统启动时，会自 rc.inet2调用它。然后，它“默默”地进入后台，等候自 SMTP端口（通常是端口号 25）进入的TCP连接。假如站点非常繁忙，这样做便显得相当有用，因为用不着为进入的每个连接，都单独启动一个 smail。当然，你也可以采取另一种做法，那便是让inetd管理SMTP端口，并在该端口上建立了一个连接之后，立即调用 smail。

smail本身有大量标志可供设置，以便对其行为进行全方位控制。但假如要在这里全部列出来，恐怕也没什么好处。幸好，smail支持大量标准的工作模式。调用它的时候，只需利用一个特殊的命令名（别名），比如rmail或者smtpd等，就可以启用这些模式。

通常，这些别名是针对 smail二进制程序设置的一些“符号链接”。以后讨论smail的各项特性时，大家会逐渐接触到它们中的绝大多数。

任何情况下，你都应该有两条指向 smail的链接：即/usr/bin/rmail和/usr/sbin/sendmail（这是根据“文件系统”而定的新 smail标准位置，另一个常见的位置是 /usr/lib）。在利用elm之类的用户代理编写和发送邮件消息时，这一消息应该随命令行上指定的收件人清单一起，送入rmail进行投递。对通过 UUCP收到的邮件来说，同样如此。但是 elm的几个版本调用的都是/usr/sbin/sendmail，而不是rmail，所以对你而言，这两者都必须有。例如，如果你将 smail保存在/usr/local/bin内，在外壳脚本输入下面两行：

```
# ln -s /usr/local/bin/smail /usr/bin/rmail
# ln -s /usr/local/bin/smail/usr/sbin/sendmail
```

如果你想进一步了解 smail的配置情况，请参考手册 smail(1)和smail(5)。如果你喜爱的软件包内没有这两个手册页，则可参考 smail的源代码。

13.1　UUCP的设置

要想在一个纯 UUCP的环境中使用 smail，基本的安装过程很简单。首先，你必须确保建立与前面提到的rmail和sendmail的两个象征性链接。如果希望从其他站点接收 SMTP批处理数据，还必须为rsmtp建立与smail的一个链接。

在Vince Skahan发布的smail版本中，可找到一个示范性的配置文件。它的名字叫做

config.sample，保存在/usr/lib/smail目录下。必须将其复制到 config文件中，并对其进行恰当的编辑，以反映出与自身站点情况相符的值。

假定你的站点名为 swim.twobirds.com，并在UUCP的映射表中注册为 swim。同时，你的智能主机名为Ulysses。那么，配置文件的内容应该像下面这个样子：

```
#
# Our domain names
visible domain=two.birds:uucp
#
# Our name on outgoing mails
visible name=swim.twobirds.com
#
# Use this as uucp-name as well
uucp name=swim.twobirds.com
#
# Our smarthost
           smart host=ulysses
```

其中，第一条语句告诉 smail你的站台从属于哪个域。请在这里插入它们的名字，相互间用冒号分隔。假如你的站点已在 UUCP映射表中完成了注册，还应在此添加 uucp。传来一条邮件消息后， smail会利用hostname(2)系统调用，来判断你的主机的名字是什么。同时，根据这个主机名，依次跟踪该列表中的所有名字，从而检查接收人的地址。假如地址与这些名字中的任何一个相符，或者与未认证的主机名相符，则认为是一个本地接收人。随后， smail会试图将邮件投递给本地主机上的一名用户或别名。否则的话，就认为接收人在远程主机上，并试图将邮件投递至目标主机。

在visible_name中，应包括你的站台的一个完整形式的域名，我们想在发到外面的邮件中使用这个域名。在所有“外出”的邮件中生成发送者的地址时，便要用到这个名字。必须使用smail能将其与本地主机对应起来的一个名字（比如与 visible_domain属性中列出的域对应的某个主机名）。否则的话，一旦有人直接回复从你那里发出的邮件，邮件便会被发至不知所云的地方！

最后一条语句设置智能主机路由选择要用到的路径。在我们这个示范性的设置中， smail会将发至远程地址的任何邮件都转发给智能主机。 smart_path属性中指定的路径将作为到智能主机的一条路由使用。由于邮件会通过 UUCP投递，所以在属性中，必须指定你的 UUCP软件已经知道的一个系统。请参考第 11章，了解如何让UUCP“知道”一个站点。

在上述文件中，还用到了一个我们尚未解释的选项 —— uucp_name。之所以要使用这个选项，原因是：在默认情况下， smail会将hostname(2)返回的值用于UUCP特有的一些东西，如From_header行中给出的返回路径等等。假如你的主机名尚未经由 UUCP映射项目完成注册，那么应告诉smail，令其换用完整形式的域名（举个例子来说，假定你的主机名是 Monad，但尚未在映射表中完成注册。此时，如果映射表中已经有一个名为 Monad的站点，那么对发给monad!root的任何邮件而言，包括从你的某个直接 UUCP邻居发出来的邮件，它们都会投递到那个Monad。显然，这种情况不是我们所希望看到的）。要想解决这个问题，唯一的办法就是在配置文件中，设置一个uucp_name选项。

在/usr/lib/smail目录中，还有另一个值得注意的文件，名为 paths.sample。它同样是一个

示范文件，告诉大家一个路径文件应该如何配置。但是，除非同时拥有到多个站点的邮路链接，便用不着对它进行配置。另一方面，假如确实拥有多条邮路，便必须自行编制这样的一个文件，或者根据Usenet映射表来生成一个。路径文件的详情，本章稍后还会详加解释。

13.2 在局域网环境中的设置

在自己的站点中，假如同时包括几个主机，相互间通过 LAN连接，便必须指定其中的一个主机，令其负责管理与外部世界的 UUCP连接。在LAN内各主机之间，通常还想通过TCP/IP协议，实现邮件的交换。假定我们现在又回到 Virtual Brewery，并将vstout设为UUCP网关。

在一个连网环境中，最好将所有用户的邮箱都放在单独一个文件系统中。然后，通过NFS装载，令其适用于其他所有主机（被它们“看到”）。这样一来，用户便可自由地改变自己使用的机器，同时不必“搬”着自己的邮箱到处乱走（或者更糟，每天早上要检查三、四台机器，看看是否有自己的新邮件）。考虑到这种情况，我们还想让发件人的邮件地址独立于实际写入邮件的那台机器（机器可以不同，地址只有一个）。一种常见的做法是在发件人的地址中，使用域名本身，而不要使用主机名！例如，假定有一位名为 Janet的用户，它的邮箱地址就应该是janet@vbrew.com，而不应是janet@vale.vbrew.com。后面，我们会详细解释如何对服务器进行配置，令其将域名识别成一个在自己站点内有效的名字。

要想将所有邮箱都集中放在一个主机上，另一个办法是使用 POP或IMAP协议。POP的全称是Post Office Protocol，即“邮局协议”。用户通过一个简单的TCP/IP连接，便能实现对自己的邮箱的访问。而另一方面， IMAP的全称是Interactive Mail Access Protocol（交互式邮件访问协议）。IMAP与POP非常相似，具有 POP的所有功能。但是， IMAP最重要的特点便是“交互式”。用户可像管理本地文件夹一样，组织与管理自己在远程邮件服务器上的邮件文件夹。无论IMAP还是POP，在Linux平台上，都有移植版本。大家可自 sunsite.unc.edu下载，目录是/pub/Linux/system/Network。

13.2.1 编写配置文件

对Brewery来说，我们打算像这样进行配置：除邮件服务器 vstout本身之外的所有主机都将“外出”的邮件路由至服务器，使用智能主机路由方式。而就 vstout自己来说，它将所有外出的邮件发给真正的智能主机，令其对 Brewery的所有邮件进行路由选择。这个真正的智能主机名为Moria。

对所有主机（vstout除外）来说，它们的标准配置文件如下：

```
#
# Our domain:
visible domain=vbrew.com
#
# What we name ourselves
visible name=vbrew.com
#
# Smart-host routing: via SMTP to vstout
smart path=vstout
smart transport=smtp
```

这与我们为纯UUCP站点使用的配置非常相似。最主要的差别在于，用来将邮件发给智能主机的传送层协议是SMTP（这是理所当然的）。visible_domain属性指示smail在外出的所有邮件中使用域名，而不是使用本地主机名。

而在UUCP邮件网关vstout上，配置文件却像下面这样（有少许的不同）：

```
# Our hostnames:
hostnames=vbrew.com:vstout.vbrew.com:vstout
#
# What we name ourselves
visible name=vbrew.com
#
# in the uucp world, we're known as vbrew.com
uucp name=vbrew.com
#
# Smart transport: via uucp to moria
smart path=moria
smart transport=uux
#
# we're authoritative for our domain
auth domains=vbrew.com
```

显然，配置文件利用了一种不同的方案，来告诉smail本地主机叫什么名字。它没有给出一份域列表，令其找出采用了系统调用的主机名，而是显式指定了一个列表。上面的列表中包含了完全资格的和未经验证的主机名以及各自的域名。这样一来，smail就能够把janet@vbrew.com识别为本地地址，并将邮件消息投递给Janet。

auto_domain参数命名域名， vstout将被视为这个域内的主机。也就是说，如果 smail收到其目标地址是host.vbrew.com（其中的主机没有对现成的本地主机进行命名）的任何邮件，它都会将其驳回并将其返回发送端。如果没有这个条目，此类邮件消息就会被发送到智能主机，由它将邮件消息返回vstout，直到超过最大网关跳计数时才将其丢弃。

13.2.2 运行smail

首先，必须决定是将 smail作为一个独立的后台程序来运行，还是令 inetd来管理SMTP端口，并且只在某一客户机发出连接请求时，才调用 smail。通常情况下，都宁愿考虑在邮件服务器上操作后台程序，因为这样的话，远远比一次又一次地针对每次连接调用 smail简单得多。由于邮件服务器还会把多数进入的邮件直接投递给用户，所以你要在其他的多数主机上选定inetd操作模式。

不管你针对每台主机选择的操作模式是什么，都必须保证你的 /etc/services文件内有下列条目：

```
smtp 25/tcp # Simple Mail Transfer Protocol
```

这个条目定义了TCP端口号， smail将利用这个端口进行SMTP对话。已分配编号RFC对此定义的标准端口号是25。

在后台程序模式下运行时， smail将把自己置入后台运行并等待 SMTP端口上发生的连接请求。一有连接，它就转向并利用对等体进程，管理 SMTP对话。通常情况下，启动 smail后台程序的过程是这样的：利用下面的命令，从 rc.inet2脚本调用它：

```
/usr/local/bin/smail -bd -q15m
```

-bd标记用于打开后台程序模式，而 -q15m标记则是令其每隔15min，就对消息队列中累积的消息进行处理。

如果你想用inetd来代替它，你的/etc/inetd.conf文件内就应该有下面这一行：

```
smtp stream tcp   nowait   root   /usr/sbin/smtpd smtpd
```

smtpd应该是一个指向 smail二进制文件的象征性链接。记住，你必须令 inetd重新读取inetd.conf，具体作法是在作出改动之后，向它发送一个 HUP信号。

后台程序模式和inetd模式是彼此不相容的。如果你在后台程序模式下运行 smail，就应该确定在inetd.conf文件内批注出用于smtp服务的所有行。同样地，在令 inetd管理smail时，一定要保证rc.inet2没有启动smail后台程序。

13.3 故障排除

如果你的配置有错，这里有大量的特征可帮助你找出错误的根源。第一个需要查看的是smail的日志文件。它们保留在 /var/spool/smail/log中，其文件名分别是 logfile和paniclog。前者列出了所有的事务，后者则只列出了与配置错误及行为相关的错误消息。

logfile文件内的典型条目如下所示：

```
04/24/94 07:12:04: [mOpuwU8-00023UB] received
|            from: root
|         program: sendmail
|            size: 1468 bytes
04/24/94 07:12:04: [mOpuwU8-00023UB] delivered
|             via: vstout.vbrew.com
|              to: root@vstout.vbrew.com
|         orig-to: root@vstout.vbrew.com
|          router: smart host
|       transport: smtp
```

这个条目展示了从root到root@vstout.vbrew.com的邮件消息已经通过 SMTP被准确投递到vstout。

smail不能投递的消息将在日志文件内生成一个类似的条目，但错误消息取代了已投递部分：

message instead of the delivered part:

```
04/24/94 07:12:04: [mOpuwU8-00023UB] received
|            from: root
|         program: sendmail
|            size: 1468 bytes
[illegible]
```

上面的错误一般发生在这种情形下： smail准确识别出消息应该被投递到 vstout，但不能和vstout上的SMTP服务建立连接。如果出现这种情况，就表明如果不是存在配置问题，就是你的smail二进制程序不支持TCP。

这个问题不是很常见的。即使有的发布版本中，有已预先编辑好的 smail二进制程序，这

些二进制程序中没有提供TCP/IP连网支持。如果碰到这种情况，你必须自行编辑 smail。安装smail之后，就可检查它是否提供 TCP连网支持了，具体做法是 telnet到你计算机上的 SMTP端口。下面展示了一个到SMTP服务器的成功连接（你的输入应该像这里标记的一样）：

```
[stuff deleted]
[stuff deleted]
[stuff deleted]
[stuff deleted]
[stuff deleted]
[stuff deleted]
[stuff deleted]
[stuff deleted]
```

如果这个测试不能生成 SMTP旗标（以220代码开头的那一行），在你自行从头编辑 smail之前，首先要确定你的配置是否真的准确无误，详情稍后讨论。

如果你在使用smail时碰到了问题，不能通过smail产生的错误消息来找出问题所在，这时，就需打开调试消息。具体作法是利用 -d标记，后面可选择性地跟上一个比较长的参数（标记和参数之前，不能有空格）。这样，smail就会在屏幕上打印出其操作报告，你就可以从中了解出错原因了。

如果实在没办法，只有在 Rogue模式下，调用 smail，具体做法是在命令行指定 -bR选项。然后，屏幕上就出现这样的消息：“ 输入giant域的邮件消息和RFC标准名册。试着令其降到 26这个协议级别并回滚 ”。虽然这个选项没有解决你的实际问题，但至少给了你某些安慰。

smail的编译

如果你确定smail二进制程序中没有 TCP连网支持，只有想办法得到其源代码。它可能包含在你的系统内（如果你通过光盘获得它的话)。不然，就可从网络上通过 FTP，从ftp://ftp.planix.com/pub/Smail/下载。

编辑smail时，最好从配置文件集着手，这些文件源于 Vince Skahan编写的newspak系统。为了用TCP连网驱动程序进行编辑，必须把 conf/EDITME内的DRIVER_CONFIGURATION宏设置为bsd-network或arpa-network。前者适用于局域网安装，但因特网要求的则是 arpa-network。两者之间的区别在于后者有一个特殊的用于 BIND服务的驱动程序，这个服务能够识别MX记录，前者却没有。

13.4 邮件投递模式

正如我们在前面提到的那样， smail能立即投递消息或把它们排成队列，等待处理。如果选择对消息排队， smail就会把所有邮件保存在 /var/spool/smail下面的消息目录内。在得到明确提示之前，是不会对队列中的消息进行处理的（这就是所谓的 “ 运行队列 ”）。

投递模式有三种；前台、后台和排队等候。前台模式下，将立即处理进入的邮件消息；后台模式下，将消息交给接收进程的子进程投递，父进程在分叉之后立即退出。排对等候模式则不言而喻。通过把 config文件内的delivery_mode属性设置为其中之一，就可选择一种投递模式了。如果在config文件内设置布尔变量queue_only的话，对进入邮件消息来说，不管有没有这个选项，始终都会排入队列等候处理。

如果你打开队列，就必须保证定期检查队列；时间间隔可能是 10或15min。如果以后台模

式运行smail，就必须在命令行上增加选项-q10m，表示每隔10min，就处理一次队列。另一种办法是在这些时间间隔内，从cron调用runq。runq应该是指向smail的一个链接。

随-bp选项调用smail，就可显示当前的邮件队列了。同样地，还可以令 mailq成为指向

```
$ mailq -v
m0pvB1r-00023UB From: root  (in /var/spool/smail/input)
                Date: Sun, 24 Apr 94 07:12 MET DST
                Args: -oem -oMP sendmail root@vstout.vbrew.com
Log of transactions:
 Xdefer: <root@vstout.vbrew.com reason: (ERR 148) transport smtp:
 connect: Connection refused
```

smail的一个链接，并调用mailq：

以上展示了消息队列中的一条单独的消息。事务日志（如果为 mailq指定-v选项的话，这是唯一能显示出来的内容）可能会给出为什么该消息还在等候投递的原因。如果以前不曾尝试过投递这封邮件消息，就不会出现其事务日志。

即使你没有采用队列，smail在发现暂时不能立刻投递消息时，有时候会把消息投入队列。对SMTP连接来说，其原因可能是因为收件人的主机是不可抵达的；但在文件系统被发现满了时，消息可能会延迟投递。因此，你应该大约每隔一小时（利用 runq）就运行一次队列，不然，延迟的消息将一直“赖”在队列中。

13.5 其他配置选项

config文件内，有相当多的选项供你设置，虽然有些选项有用，但与运行 smail没有本质上的联系，所以不是我们讨论的主题。相反地，我们只提少数没有理由不用的选项：

■ error copy postmaster —— 如果设置这个布尔变量，任何错误都会产生一条发送给邮局负责人的消息。通常情况下，这只针对由于配置不当而引起的错误。这个变量可以被打开，具体做法是把它放入config文件，并在其前面加一个“+”号。

■ max hop count —— 如果消息的网关跳计数（也就是说邮件传输经过的主机数）等于或大于这个数，远程投递尝试就会以出错告终，向发件人返回一条错误消息。这是用来防止消息永久循环的。跳计数一般是从邮件头的Received:字段计算而来的，但也可在命令行上利用-h选项，手动进行设置。这个变量的默认设置是20。

■ postmaster —— 邮局负责人的地址。如果邮局负责人的地址不能解析成一个有效的本地地址，这就是最后的“法宝”。默认设置是root。

13.6 消息路由和投递

smail把邮件投递划分到三类不同的任务中，亦即三个功能模块：路由器、定向器 (director)以及传输模块。

其中，路由器模块负责解析所有远程地址，判断一条消息接下去应投给哪个主机，以及使用哪种传输协议。以具体的链接为基础，可选用像 UUCP或SMTP这样的传输协议。

至于本地地址，则发给定向器，由它决定是转发还是别名转换。举个例子来说，指定的地址可能是一个别名，或者是一个邮件列表。另外，用户可能想把自己的邮件转换到另一个地址。假如最终判断出来的地址位于远程，则将其传送给路由器模块，作进一步的路由选择

处理。否则的话，就为其分配一种传输协议，进行本地投递操作。迄今为止，最常见的一种操作是将消息投递到一个邮箱，但也可以将消息（邮件）输送给一个命令，或者为其追加“附件”（随正文一起发送的文件）。

最后，传输模块根据选定的投递方法，完成邮件的实际投送操作。它会试着发出邮件。假如操作失败，那么要么生成一封回函说明情况，要么等待下一次继续尝试。

使用smail，我们可以非常灵活地配置所有这些任务。针对每一种任务，都同时提供了大量驱动程序。可根据自己的实际情况，在这些驱动程序中，选出自己需要的一种。我们要通过不同的文件，向smail描述它们。这些文件的名字分别是路由器、定向器和传输模块，均位于/usr/lib/smail下面。

假如这些文件不存在，就会自动选择一些合理的默认值。这些默认值适用于采用 SMTP或UUCP进行邮件传输的大多数站点。假如你想修改 smail的路由策略，或打算对一种传输协议进行改动，便需从 smail的源码中，找到相应的示范文件（默认配置文件可在源码目录的samples/generic内找到）。随后，将示范文件复制到/usr/lib/smail。最后，根据自己的实际需要，对它们进行修改。本书的附录也给出了这些示范配置文件。

13.7 消息的路由

smail收到一条消息（邮件）之后，首先会检查目的地是本地主机，还是一个远程站点。假如目标主机地址是config中配置好的某个本地主机名，那么消息会被传送给定向器模块。否则的话，smail就会将目标地址传给大量路由器驱动程序，找出可向其转发消息的那个路由器。可在路由器文件中对它们进行描述；假如文件不存在，同样地会使用一系列默认路由器。

目标主机会依次传递给所有路由器，直到发现“最相符”的那个路由器为止。举个例子来说，假定有一条消息投递给joe@foo.bar.com。此时，假设有一个路由器知道bar.com域内所有主机的一条默认路由；但与此同时，另一个路由器却掌握了 foo.bar.com本身的信息。由于后者与那条消息的目标地址“更为相符”，所以最终选定的便是它，而不是前一个“广泛意义上”的路由器。假如同时有两个路由器都“最相符”，那么在路由器列表文件中排在最前面的会被选中。

现在，假定这个路由器指定的邮件传输协议是 UUCP，同时生成了一个新的目标地址。新地址会传递给传输协议，同时传递的还有下一步负责接收转发邮件的主机名。在前述例子中，smail可能发现要想通过UUCP抵达foo.bar.com，需要使用路径ernie!bert。随后，它会据此生成一个新的目标地址：bert!foo.bar.com@!user。同时，它会指示UUCP传输协议将它作为信封地址使用，再传送给ernie。

若采用的是默认安装，那么可使用下述路由器：

■ 假如目标主机地址可用gethostbyname(3)或gethostbyaddr(3)库调用解析出来，那么消息会通过SMTP传送。唯一的例外是，最终发现该地址引用的是一个本地主机！此时，它也会传递给定向器模块。

smail也能识别以“点分十进制”格式写成的 IP地址，并能将其看作一个合法的主机名，只要该IP能通过gethostbyaddr(3)调用解析出来。例如，scrooge@[149.76.12.4]便可能是一个有效的邮件地址，尽管普通人记住的Scrooge的地址在quark.physics.groucho.edu上。

■ 如果你的机器连在Internet上，这些路由器便没有丝毫用处，因为它们不支持 MX记录。请参考后文，了解在这种情况下，该如何操作。

■ 假如存在/usr/lib/smail/paths路径别名数据库，smail就会试着在该文件中搜索目标主机（消除尾随的任何 .uucp字样）。若邮件发给由这个路由器匹配的一个地址，那么它们会通过UUCP传送，并利用在该数据库中找到的路径。

■ 主机地址（消除任何尾随的.uucp后缀）会与uuname命令的输出结果对照，核实目标主机是否是一个事实上的UUCP邻居。假如答案是肯定的，邮件消息就会用UUCP传输协议进行投递。

■ 假如地址不与上述任何一种路由器相符，便会将其投递给智能主机。至于到智能主机的路径，以及打算采用的传输协议，则在 config文件中设置。

这些默认操作适用于大量简单的站点安装，但假如路由情况稍微有些复杂，也很容易出现错误，造成失败。假如你遇到了我们下面描述的任何一种问题，都必须安装自己的路由器文件，令其覆盖默认设定。在附录中，我们给出了一个可首先拿来“开刀”的示范路由器文件。在某些发行版本中，也配套提供了一系列配置文件，它们都经过了仔细的调校，可有效地解决我们碰到的问题。

或许最糟的情况在于，你的主机存在于两个网络区域的交汇处，既有拨号 IP，也有UUCP链路。此时，在自己的主机文件中，可能放置了一些只有极少数情况才会通过 SLIP链接通信的主机名。这样一来，smail就会试图通过SMTP，投递发给这些主机的任何文件。但是，这往往并非我们所希望的，因为尽管 SLIP链接并不是经常用到，但SMTP的速度要比通过UUCP发送邮件慢得多。在采用默认设置的前提下，我们没有办法改变 smail的行为。

要想解决这个问题，可让smail在查询解析器之前，检查路径文件，并将自己希望强行通过UUCP投递邮件的所有主机都放到路径文件中。如果不想通过 SMTP发送任何邮件，亦可将那些基于解析器的路由器彻底地“剔除”——具体做法是将它们变成“注释”，加上一个注释符号即可。

另一个问题是默认设置并不具备任何真正的 Internet邮件路由功能，这是由于基于解析器的路由器不会对MX记录进行任何处理。要想实现对 Internet邮件路由的完全支持，要标示出这个路由器并取消使用BIND的那个路由器。但是，有的版本中的smail二进制程序中没有编入BIND支持。如果你启用BIND，但在painclog文件内却得到这样的消息“ routerinet_hosts:driver bind not found ”时，你就必须获得源码，重新编辑smail。

最后，一般说来，使用 uuname驱动程序都不是上上之策。比如，在没有安装 UUCP时，该驱动程序会产生配置错误，因为无法找到 uuname命令。其二就是UUCP系统文件内列出的站点多于你事实上与之有邮件链接的站点。它们可能是仅与你交换新闻的站点，或是你偶尔通过匿名UUCP从中下载文件的站点，但没有其他通信往来。

为了解决第一个问题，你可用一个外壳脚本来代替执行简单退出的 uuname。但是最常见的解决办法是编辑路由器文件，并删除这个驱动程序。

路径数据库

smail希望在/usr/lib/smail下面的路径文件中，找出路径别名数据库。这个文件是可选的，所以如果你从不打算执行任何路径别名路由，只须删除现有的任何路径文件即可。

路径文件必须是一个分类的ASCII文件，其中包含一些条目，这些条目把目标站点名映射为UUCP bang路径。这个文件必须分类的原因是smail利用二元搜索来查找一个站点。该文件内是不允许出现批注的，而且必须用空格符把站点名和路径分开。有关路径别名数据库的详

情，请参见第12章。

如果这个文件是你手动建立的，就应该确保把一个站点的所有合法名称包含在这个文件内。比如，如果一个站点既有一个纯 UUCP名，又有一个完整形式的域名，你就必须分别为这两个名称增加条目。利用 sort(1)命令把该文件传输一遍，就可对它进行分类。

但是，如果你的站点只是一个叶子站点，就不需要任何路径文件：只须在你的 config文件内设置智能主机属性，把所有的路由交给你的邮件发送机处理。

13.8 将消息投递至本地地址

更为常见的情况是，本地地址只是一个用户的登录名，此时的邮件被投递到他／她的信箱 —— /var/spool/mail/user。其他情况还包括别名和邮件列表，以及用户转发的邮件。出现这些情况时，本地地址就扩展成一个新的地址列表，既可能有本地地址，又可能有远程地址。

除了“ 普通 ”地址， smail还可以处理其他类型的本地消息，比如文件名和管道命令等。这些不是地址，所以你不能向其发送邮件，比如 /etc/passwd@vbrew.com；只有当它们取自转发或别名文件时，才是有效的。

文件名以“ / ”或“ ~ ”开头。后者指代用户的根目录，而且可能是唯一的文件（如果这个文件名取自 .forward文件或信箱中的一个转发条目的话），详情参见后续的讨论。在向一个文件投递消息时， smail会在必要时，把消息添加到这个文件内。

任何以管道符 “ | ” 开头的命令都属于“ 管道命令 ”。smail碰到这种命令后，会将它传递给外壳程序，同时还会传递它的一些相关参数，只是会将前置的“ | ”删去。注意消息（邮件）本身会以标准输入的形式，传给命令。

举个例子来说，要想将一个邮件列表送入本地新闻组，那么可使用一个名为 gateit的外壳脚本。同时，设置一个本地别名，利用“ |gateit ”，将来自该邮件列表的所有消息都传给脚本。

假如调用中包含了空白（white space），那么必须将其封闭在一对双引号中。由于这里牵涉到安全性问题，所以千万注意假如是通过某种可疑的方式例如，假如从中取得地址的那个别名文件具有“ 人人可写 ”的属性！取得的地址，千万不要执行命令。

13.8.1 本地用户

对本地地址来说，最常见的一种情况便是直接引用某个用户的邮箱。该邮箱保存在 /var/spool/mail中，而且会以用户的名字来命名。注意“ 邮箱 ”是该用户拥有的，其中保存有一组邮件，而且采用660模式。假如发现该邮箱不存在， smail会自动创建一个。

要注意的是，尽管 /var/spool/mail目前是用来保存邮箱文件的标准场所，但某些邮件软件可能会采用不同的路径，比如像 /usr/spool/mail这样的形式。向本机用户发信时，假如常出现失败的情况，那么便应试着建立一个到 /var/spool/mail的符号链接，看看能否解决问题。

smail要想正常工作，有两个地址（帐号）必须存在： MAILER-DAEMON和postmaster。为一封无法投递的邮件生成反馈消息时，会向 postmaster（邮局管理员）帐号发送该邮件的一份拷贝，以检查是否由于配置不当而造成了问题。至于 MAILER-DAEMON，则用作反馈消息的“ 发件人 ”地址。

假如这些地址没有在你的系统上建立有效的帐号， smail会将MAILER-DAEMON自动映射到postmaster，而将postmaster映射到root。显然，无论如何都不应采用这种“ 默认 ”的映射

关系，而应为postmaster帐号选择一个恰当的别名，将其映射到负责管理和维护邮件软件的人（帐号）身上。

13.8.2 转发

用户可指示将自己的邮件转发到另一个地址。此时，可选择由 smail提供的两种方法之一。一个办法是在邮箱文件的第一行中，加入下述字句：

```
Forward to recepient...
```

这样一来，收到的所有邮件都会自动转发给指定的一系列收件人。另外，用户亦可在自己的home目录中，创建一个.forward文件。在该文件中，包含了用逗号分隔的一系列收件人。进行大批量转发时，文件的每一行都会读入，并加以正确的解析。

注意可使用任意类型的地址。也就是说，完全可以像下面这样设置一个 .forward文件：

```
janet, "|vacation"
```

第一个地址会将收到的邮件投递到 Janet的邮箱，而vacation命令会向收件人返还一条简短的通知。

13.8.3 别名文件

smail能正确控制大量别名文件，它们与 Berkeley的sendmail已知的那些文件兼容。

别名文件的条目有下列的形式

```
alias: recipients
```

recipients（收件人）是一个地址列表，各地址间用逗号隔开，这个列表将被别名代替。如果下一行以空格符开头的话，收件人列表可能会持续到若干行。

有一个特殊的特性允许smail处理取自别名文件的邮件列表：如果你指定“:include:filename”为收件人，smail就会读取指定的文件，并将其内容作为收件列表提交。

主要的别名文件是/usr/lib/aliases。如果你让这个文件成为全球可写文件， smail将不向该文件内指定的外壳命令投递任何消息。下面是一个示例文件：

```
# vbrew.com /usr/lib/aliases file
hostmaster: janet
postmaster: janet
usenet: phil
# The development mailing list.
development: joe, sue, mark, biff
        /var/mail/log/development
owner-development: joe
# Announcements of general interest are mailed to all
# of the staff
announce: :include: /usr/lib/smail/staff,
          /var/mail/log/announce
owner-announce: root
# gate the foobar mailing list to a local newsgroup
ppp-list: "|/usr/local/lib/gateit local.lists.ppp"
```

在向别名文件产生的地址投递消息期间，如果出现了错误， smail就会试图向“别名拥有者”（alias owner）发送该错误消息的副本。比如，如果在把消息投递到开发邮件列表时，投

递屡次失败，错误消息副本就会被邮寄到发件人以及邮局负责人和开发邮件列表拥有者手中。如果拥有者地址不存在，就不会再出现其他的错误消息了。

在把消息投递到文件或调用别名文件内指定的程序时， smail将成为nobody用户，从而避免出现任何安全隐患。特别是在投递到文件时，这种情况会显得非常讨厌。比如在前面给出的文件中，日志文件必须由 nobody用户拥有，而且“ 可写 ”。否则的话，假若投递给他们，就会失败。

13.8.4 邮件列表

此外，邮件列表也可放弃别名文件，而换用 /usr/lib/smail/lists目录中的文件来进行管理。文件lists/nag-bugs文件描述了一个名为nag-bugs的邮件列表。在这个列表中，包含了所有列表成员的地址，相互间用逗号分隔。该列表亦可以多行形式出现，用一个斜杠符号注明所有的注释性内容。

针对每个邮件列表，都应存在一个名为 owner-listame的用户（或别名）。对地址进行解析时，出现的任何错误都会报告给该用户。该地址也作为所有外发邮件的“ 发件人 ”地址使用，保存在Sender:这个邮件头字段中。

13.9 以UUCP为基础的传输

目前，有许多传输都被编入 smail，其目的是为了利用 UUCP套件。在一个UUCP环境内，消息的传递过程是：在下一台主机上调用 rmail，再在其命令行上提供作为标准输入的消息和信封地址。在你自己的主机上， rmail应该是指向smail命令的一个链接。

在把消息传递给 UUCP传输时， smail把目标地址转换为一个 UUCP bang路径。例如，user@host将被转换为 host!user。含有“ % ”地址操作符的地址则会被保留。这样一来，user%host@gateway就会变成%gateway!user%host。但是， smail本身永远都不会生成这样的地址。

此外， smail可通过UUCP收发BSMTP批量邮件。通过BSMTP，在同一“ 批 ”邮件中，会封装一条或多条消息（邮件）。而在这个批量邮包中，包含了本地邮件发送程序在建立了一个真正的SMTP连接后要执行的命令。 BSMTP常用在大量邮件的“ 存储与转发 ”（以UUCP为基础）服务中以节省磁盘空间。在附录中，我们提供的示范传输文件包含了一个示范的 bsmtp，它可在一个队列目录中生成部分 BSMTP批量邮包。注意稍后必须将它们编译成最终的邮包，这是用一个外壳脚本来完成的，它会自动添加恰当的 HELO和QUIT命令。

针对特定的UUCP链路，要想启用BSMTP传输方式，必须使用所谓的“ 方法文件 ”（可参考smail(5)手册页了解详情）。假如只有一条UUCP链路，而且正在使用智能主机路由器，那么要想通过SMTP发送批量邮包，可将smart_transport配置变量设为bsmtp，而不是uux。

要想通过UUCP来接收SMTP批量邮包，必须保证目标远程站点可执行一个拆除封装的命令。假如远程站点使用的也是 smail，那么必须让 rsmtp建立与smail的一个链接。假如远程站点运行的是sendmail，则还要安装一个外壳脚本，名为 /usr/bin/bsmtp，令其执行一次简单的“ exec rsmtp ”操作（注意一个符号链接在此是不管用的）。

13.10 以SMTP为基础的传输

目前， smail支持SMTP驱动程序通过 TCP连接投递邮件（编者把这个支持称为“ 简单 ”。

在将来的smail版本中，他们将提供完整的后端支持，使之能更有效地对这种传输进行处理）。它能够在一台单独的主机上，把邮件消息投递到任何地址，主机名要么被指定为完整形式的域名（能够被连网软件解析），要么被指定为用方括号封闭起来的点分四段式。一般说来，通过BIND、gethostbyname(3)或gethostbyaddr(3)路由器驱动程序解析出来的地址都将被投递给SMTP传输。

SMTP驱动程序将立即试着通过smtp端口（/etc/services中列出的）连接到远程主机。如果不能连接到远程主机或连接超时，邮件投递将在稍后重试。

对因特网上的邮件投递来说，它们要求到目标主机的路由须按照 route-addr格式进行指定（详见第12章），而不是被指定为一个 bang路径（但是，因特网上的路由使用也有限制。所以应该采用完整形式的域名）。由此， smail将把/usr%host@gateway（gateway是通过host1!host2!host3!抵达的）转换为source-addr地址<@host2,@host3:user%host@gateway>，后者将被当作消息的信封地址发送到 host1。为了启用这类变形（以及内置的 BIND驱动程序），你必须对transports文件内的smtp驱动程序条目进行编辑。附录将给出一个 transports文件示例。

13.11 主机名的限制

有时，人们总想在发件人或收件人的地址内采用不完整的主机名（换言之，不想用域名），比如，两个网络进行网关沟通时，其中一个需要完整形式的域名。在 Internet-UUCP中转机上，不完整的主机名应该默认对应于uucp域。除此以外的其他地址修改都会有问题。

/usr/lib/smail/qualify文件告诉smail哪些域名对应于哪些主机名。 qualify文件的组成情况是：第一列是主机名，然后是域名。如果行内包含作为其第一个非空字符的“ *”号，就表明这一行是批注。条目的搜索是按其出现的顺序进行的。

有“*”的特殊主机名可对应于任何一个主机名，因此，你可以把以前从未提及的所有主机映射于一个默认的域。此类主机名只能作为最后一个条目出现。

在Virtual Brewery，所有主机都已经被设置为“ 在收件人地址内，采用完整形式的域名 ”。不完整的收件人地址被看作在uucp域内，所以， qualify文件内，只需要有一个条目。

```
# /usr/lib/smail/qualify, last changed Feb 12, 1994 by janet
#
*           uucp
```

第14章　Sendmail+IDA指南

只有编辑过sendmail.cf文件的人，才能称之为合格的 Unix管理员。但也有一种说法是，只有疯子才会再次编辑这个文件。

看来，sendmail既是一个功能强劲的、优秀的程序。但与此同时，对许多人来说，也是一个难以学习和理解的程序。显然，对任何一个定义参考说明多达 792页的程序来说，它令人望而却步恐怕也是正常的吧？（Sendmail由O ’Reilly and Asssociates发行。）

但是，sendmail+IDA却与众不同。

注意　关于最新的Sendmail版本，可参见http://send.mail/com/或http://send.mail/org/。

它剔除了编辑其含义一直不详的sendmail.cf文件的必要，提供了一些相对简单的支持文件，名为tables，允许管理员定义站点专有的路由和寻址配置信息。转而采用 Sendmail+IDA，相信会为你节约时间和减轻工作负担。

看看其他主要的邮件传输代理，恐怕没有哪一个能够比 Sendmail+IDA更快，更简单。有了Sendmail+IDA，运行一个普通的 UUCP或因特网站点之类的常事变得越来越简单。以前普遍较难的配置也变得更易于建立和维护。

编写本书之即， Sendmail+IDA的最新版本是1.5，可通过匿名FTP，从vixen.cso.uiuc.edu下载。它在Linux系统下，不需要任何补丁就可编辑。

对于要获得sendmail+IDA源代码，以便进行编辑、安装和运行等操作来说，它们所需的所有配置文件都包含在 newspak.2.2.tar.gz内（可通过匿名FTP下载它。它位于 sunsite.unc.edu的/pub/Linux/system/Mail目录下）。

14.1　配置文件综述

传统 sendmail的设置是通过一个系统配置文件（一般是　/etc/sendmail.cf或/usr/lib/sendmail.cf）来完成的，这个系统配置文件和大家以前见过的文本语言没有任何类似的地方。编辑sendmail.cf文件，为系统提供自定义行为，不见得是种享受。

Sendmail+IDA令这一切成为过去，它将所有的配置选项表格化，并且采用的语法相当易于理解。这些选项是怎样配置的呢？具体作法是：通过源代码提供的 Makefiles，针对大量的数据文件，运行m4（一个宏处理程序）和dbm（一个数据库处理程序）。

senmail.cf文件只定义系统的默认行为。事实上，所有特殊的自定义行为都是通过大量的选项表格来定义的，而不是直接对sendmail.cf文件进行编辑。下面列出了最近的sendmail文件：

```
/etc/aliases            raw data for alias names
/etc/aliases.db         data base of alias names
/etc/sendmail.cf        configuration file
/etc/sendmail.hf        help file
/var/log/sendmail.st    collected statistics
/var/spool/mqueue/*     temp files
/var/run/sendmail.pid     The process id of the daemon
```

14.2 sendmail.cf文件

Sendmail+IDA的sendmail.cf文件是不能直接编辑的，但可通过一个 m4配置文件生成，这个配置文件是由本地系统管理员提供的。下面，我们将其称为 sendmail.m4。

这个文件中包含少数定义，另外指向真正的功臣 —— 表格。一般说来，这个文件只需指定：

- 本地系统上采用的路径名和文件名。
- 站点名，用于收发电子邮件之用。
- 需要什么样的默认邮件服务器（也可能是智能中转机）。

另外，还可定义大量的参数，用于建立本地站点的行为或改写已编辑的配置项目。这些配置选项被标识在源目录的ida/cf/OPTIONS文件内。

用于小型配置的sendmail.m4文件（指UUCP或SMTP，它们的所有非本地邮件被中转到一个直接连接的智能主机）可以只有 10或15行那么短，其中还包括批注。

sendmail.m4文件常用参数

sendmail.m4文件中，少数项目是始终都需要的；其他的则可以忽视（默认设置）。下面几个小节将对sendmail.m4文件内的每个项目进行详细讨论。

1. 定义路径的项目

LIBDIR定义目录，Sendmail+IDA希望从这个目录下找到自己需要的配置文件、 dbm表格和特殊的本地系统定义。在一个典型的二进制程序中，这个目录通常被编辑到 sendmail二进制程序内，不需要在sendmail.m4文件内显式设置。

2. 定义本地邮件服务器

许多操作系统都提供了一个能处理本地邮件投递的程序。大部分用于许多 Unix变体的典型程序已经归入sendmail二进制程序。

在Linux系统中，有必要显式定义恰当的本地邮件服务器，因为你手中的文档中，尚未安装本地邮件投递程序。具体作法是在 sendmail.m4文件内指定LOCAL_MAILER_DEF。

例如，要想一个常用的投递程序提供此项服务，可将 LOCAL_MAILER_DEF设为mailers.linux。

注意 deliver是Chip Salzenberg（其邮件地址是chip%tct@ateng.com）编写的。它已经包含在一些已发布的程序内，可在ftp.uu.net之类的匿名FTP归档内找到它。

3. 处理退回的邮件

许多站点都认为保障邮件收发的成功率越高越好，最好是百分之百。所以对 syslogd(8)进行检查是非常有助于了解邮件投递情况的，本地邮件管理员一般都需要查看退回邮件的邮件头，以便判断邮件未能准确投递的原因是用户错误，还是某个系统的配置错误。

定义POSTMASTERBOUNCE将复制退回的每份邮件消息，并将这些备份发给被定义为系统Postmaster的人。

不幸的是，设置上面的参数也会导致邮件消息被传给邮局管理员，这对该系统的用户来说，似乎有些侵犯个人隐私。

站点的邮局管理员一般应该严于律己（也可通过技术手段来实现，比如说通过外壳脚本将退回邮件中的正文消息删除），不看那些不是写给他们的邮件。

4. 域名服务相关项目

目前几个著名的网络，由于历史上的原因，经常出现在邮件地址中，但 DNS却不能识别它们。因此定义PSEUDODOMAINS（伪域）是为了避免出现徒劳无益的DNS查找。

5. 定义本地系统名

系统经常都希望将自己真正的身份隐藏起来，充当邮件网关，接收和处理采用其曾用地址的邮件。

PSEUDOYMS指定一份所有的主机名列表，本地系统将为这些主机接收邮件。

DEFAULT_HOST指定主机名，这些主机名将出现在本地主机发送的邮件消息中。这个参数必须设为一个有效值，这是非常重要的，如若不然，所有返回的邮件都将成为不可投递的“死信”。

6. 与UUCP相关的项目

通常情况下，出于DNS和UUCP的需要，系统只有名称。 UUCPNAME允许你另行定义一个主机名，出现在外出UUCP邮件的邮件头。

UUCPNODES定义若干个命令，这些命令为通过UUCP直接连接我们的系统返回主机名列表。

对采用“bang”语法定址的邮件来说， BANGIMPLIESUUCP和BANGONLYUUCP是为了确保它们的处理方式符合UUCP行为，而不是符合如今因特网上风行的域名解析服务行为。

7. 中转系统和邮件服务器

对许多系统管理员来说，他们的确不希望再为自己的系统能否抵达全球所有网络上的所有系统而操心。相反地，他们更愿意把所有外出邮件都交给另一个系统（即所谓的智能主机）来中转。

RELAY_HOST定义此类智能邻近系统的UUCP主机名。

RELAY_MAILER定义用于中转邮件消息的邮件服务器。

记住这一点是非常重要的：设置这两个参数将导致你的外出邮件被转发到这个远程系统，这样无疑会加大此类系统的工作量。在将自己的系统配置为“将另一个系统作为自己的常用中转主机”之前，务必确信已得到远程邮局管理员的同意。

8. 各种配置表格

有了这些宏之后，你就可以对Sendmail+IDA能够在其中找到各个dbm表的位置进行更改。这些表定义系统的真正行为。通常，最明智的做法是把它们留在 LIBDIR内。

9. 主控sendmail.mc文件

Sendmail+IDA的作者们提供了sendmail.mc文件，其中包含组成sendmail.cf文件精髓的东西。其新版本定期发布，修改错误或增添新功能，它不要求完成发布以及通过源代码重新编辑sendmail。

记住，重要的是不要编辑这个文件。

10. 哪些条目是真正需要的呢？

在不使用任何一个可选dbm表时， Sendmail+IDA通过DEFAULT_MAILER（也可能通过RELAY_HOST和RELAY_MAILER）投递邮件， DEFAULT_MAILER是在sendmail.m4文件中定义的，用于生成sendmail.cf文件。通过domaintable或uucpxtable中的条目，便可轻松改写这一行为。

就因特网上采用域名服务的普通站点，或只有 UUCP连接，通过 RELAY_HOST，经由UUCP转发所有邮件的站点而言，它们根本不需要任何特殊的表条目。

事实上，所有系统都应该设置 DEFAULT_HOST、PSEUDONYMS和DEFAULT_MAILER宏，前两者定义了正式站点名和别名。如果你只有一个中转主机和中转邮件服务器，就不必设置这些默认值，因为它会自动进行处理。

对UUCP主机来说，可能也需要将 UUCPNAME设置为其正式的 UUCP主机名。还可能需要设置RELAY_MAILER和RELAY_HOST，两者将启用通过邮件中转机的智能主机路由。准备采用的邮件传输定义在 RELAY_MAILER内，对UUCP站点来说，这个传输通常应该设置为UUCP-A。

如果你的站点只有 SMTP，而且采用域名解析服务，就要把 DEFAULT_MAILER改为TCP-A，或许还应该删除RELAY_MAILER和RELAY_HOST这两行。

14.3 Sendmail+IDA表格指南

Sendmail+IDA提供了大量的表格，允许你针对特殊情况、远程系统和网络，改写sendmail的默认行为（sendmail.m4文件中指定的），并定义特殊行为。这些表格利用开发商提供的Makefile，随dbm一起进行投递。

如果可能的话，许多站点都需要此类的表格。若你的站点不需要这些表格，最简单的作法便是令其成为零长度的文件（利用 touch命令），并采用LIBDIR内的默认Makefile文件，而不是直接编辑Makefile。

14.3.1 mailertable

mailertable根据远程主机或网络名，为特定的主机或域定义特殊处理。它常用于因特网站点上，用于选定中间邮件中转主机或网关，从而抵达或通过一个远程网络，指定准备采用的特定协议（UUCP或SMTP）。UUCP站点一般不需要采用这个文件。

Order是非常重要的。 sendmail从上到下读取该文件，并按照与之匹配的第一条准则对邮件消息进行处理。所以，最明智的作法一般是把最明显准则放在文件的顶部，较普通的准则放在底部。

邮件服务器的数目可能比较多。其间的区别一般是其处理地址的方式。典型的邮件服务器是TCP-A（具有Internet式地址的TCP/IP）、TCP-U（具有UUCP式地址的TCP/IP）和UUCP-A（具有Internet式地址的UUCP）。

位于一个mailertable行左边，将邮件服务器和主机部分分开的字符，定义了 mailertable修改地址的方式。重要的是大家要意识到它只能改写信封（使邮件能进入远程系统）。除了信封之外，任何改写一般无效，因为可能会破坏邮件配置。

14.3.2 uucpxtable

通常情况下，对发向具有完整资格域名主机的邮件来说，它们的投递是利用域名服务，通过Internet式的SMTP或中转主机来完成的。 uucpxtable强制投递通过 UUCP路由，具体作法是将已定域的主机名转换为 UUCP式的未定域的远程主机名。

uucpxtable常用于这些时候：当你作为站点或域的邮件转发者时，或你希望通过一条直接

而可靠的UUCP链接，而不是通过默认邮件服务器、任何中间系统和网络的多次网关跳投递邮件时。

对UUCP站点来说，如果与其对话的 UUCP邻居采用了定域的邮件头，它就会利用这个文件强制邮件的投递流经两个系统之间的直接 UUCP点到点链接，而不是通过 RELAY_MAILER和RELAY_HOST之间或通过DEFAULT_MAILER的路由。

没有与UUCP对话的因特网站点可能不会采用这个 uucpxtable文件。

假设你为DNS内的一个系统提供邮件转发服务，这个系统名为 sesame.com，在UUCP映射中是sesame。你就需要uucpxtable条目，强制发到这些主机的邮件通过你的直接 UUCP连接。

14.3.3 pathtable

pathtable用于定义指向远程主机或网络的显式路由。 pathtable文件应该采用路径别名式的语法。每行的两个字段间，必须用一个真正的制表符隔开。不然， dbm会拒绝识别。

许多系统都不需要任何pathtable条目。

14.3.4 domaintable

domaintable一般用于强制DNS查找后的特定行为。它还允许管理员利用速记名来代替常用的系统或域名，这是通过用恰当的系统或域名自动替换速记名的方式来实现的。它还可用于用“恰当”信息类替换错误的主机或域名。

许多站点都不需要任何domaintable条目。

14.3.5 别名

别名允许发生的事情很多：

■ 它们为准备定址的邮件提供速记或众所周知的系统或域名，使之能投递给一个或多个用户。

■ 它们利用作为程序输入的邮件消息对该程序进行调用。

■ 它们向一个文件发送邮件。

按照RFC的描述，所有系统都需要Poster和MAILER-DAEMON的别名。

在定义用于调用程序的别名或写入一个程序时，一定要注意安全，因为 sendmail一般都会运行setuid-root。

关于邮件别名的详细情况，可参考别名手册。

14.4 sendmail的安装

本小节将为大家讲解如何安装一个典型的 Sendmail+IDA二进制程序，并讨论怎样才能使其本地化和发挥作用。

目前， Sendmail+IDA的最新发布的二进制程序可从 sunsite.unc.edu获得，它包含在/pub/Linux/system/Mail下面。即使你有sendmail的早期版本，仍然强烈建议大家下载最新的sendmail5.67+IDA1.5版本，因为所有必须的Linux专有补丁都包含在vanilla源代码中， 1993年12月1日之前发布的版本中存在的几个重要的安全漏洞已经得到弥补。

如果你正在根据源代码构建 sendmail，就应该按照发布的源程序内的 README所说的去

做。新版的Sendmail+IDA源程序可从vixen.cso.uiuc.edu获得。要建立Sendmail+IDA，还需要Linux专有的配置文件，它包含在 newspak-2.2.tar.gz包内。后者位于 sunsite.unc.edu的/pub/Linux/system/Mail目录下。

如果以前已经安装了smail或另外的邮件投递代理，将smail的所有文件删除或重命名可能要安全得多。

14.4.1 sendmail.cf文件的建立

为了根据自己站点的情况建立一个 sendmail.cf文件，你必须编写一个 sendmail.m4文件，并用m4对它进行处理。在/usr/local/lib/mail/CF中，你可以找到一个示例文件，名为sample.m4。把它复制到yourhostname.m4，并对它进行编辑，以反映你的站点上的情况。

示例文件是为只具有UUCP服务的站点而设的，这些站点有定域的邮件头，并且和智能主机打交道。此类站点只需要编辑少数几条项目。

14.4.2 sendmail.cf文件的测试

为了把 sendmail置入“测试”模式，在调用它时须带上 -bt选项。默认配置文件是sendmail.cf，这个文件已经安装在你的系统中。利用 -Cfilename选项，就可以测试备用文件了。

14.4.3 对sendmail.cf和表格进行综合测试

此时，你已经查证了邮件具有预期的默认行为，并且也能收发地址无误的电子邮件。为了完成安装过程，还有必要创建相应的 dbm表格获得预期的最终效果。

在针对自己站点的需求创建好表格之后，必须通过 dbm对它们进行处理。具体作法是在包含这些表的目录内键入make。

如果你只有UUCP服务，就不必创建README.linux文件内提到的任何表格。只须单击这些文件，Makefile便开始运行。

如果你只有UUCP服务，并和除自己的智能主机之外的其他站点打交道，就需要为每个站点增加uucpxtable条目（或发到这些站点的邮件也将通过智能主机进行投递），并对修改过的uucpxtable运行dbm。

首先，需要确定流经自己的RELAY_HOST的邮件通过RELAY_MAILER得以发送。

如果除了RELAY_HOST之外，你还有UUCP邻近，就需要确定发送到这些邻近的邮件具有恰当的行为。对地址采用 UUCP式语法的邮件来说，如果它们将投递到正在和你进行 UUCP对话的主机，就应该直接抵达其目的地（除非你利用 domaintable条目显式阻止这一行为）。

如果你有uucpxtable条目，该条目用于强制 UUCP投递到特定的UUCP邻居，这些邻居发送具有Internet式定域邮件头的邮件，这也是需要测试的。

14.5 邮件的操作技巧

我们已从理论上讨论了 Sendmail+IDA系统的配置、安装和测试，现在来看看一个邮件管理员的实际操作。

有时，远程系统可能会宕机。 modem或电话线路失效、 DNS的设置有误都归结于人为错

误。网络意外断开。这些情况下，邮件管理员需要知道如何最快、最有效、最安全地解决此类状况，让邮件通过备用路由传输，直到远程系统或服务供应商恢复正常的服务为止。

本章的其余小节将为大家提供解决“常见电子邮件紧急状况”的良策。

14.5.1 向中转主机转发邮件

为了把特定的主机或域的邮件转发到指定的中转系统，一般都要采用 mailertable。

14.5.2 强制邮件进入配置不当的远程站点

因特网站点经常会碰到这样的事：要把邮件投递到一些配置不当的远程站点。这类问题还有多种表现形式，但常见的症状是邮件被远程系统退回或根本无法抵达目的地。

这些问题常常令系统管理员头疼不已，因为用户一般不注意管理员个人要管理若干个甚至全世界的系统或者不知道如何让远程系统的管理员修复这个问题。他们只知道自己的邮件未能抵达收件人的手中，所以所有的抱怨都会冲着你来。

远程站点的配置不当是他们的问题，与你无关。碰到此类情况，切记不要改动自己的站点，以适应与配置不当的站点进行通信。如果不能和远程站点上的邮局管理人取得联系，要求他们及时订正不恰当的配置，你只有下面这两种选择：

■ 一般说来，强制邮件进入远程站点是可能成功的，即使由于远程站点配置不当，远程端上的应答没有发挥作用…，但那仍然是远程系统管理员的问题。

你能做的只是利用这些收件主机或域的 domaintable条目，订正外出消息信封内有误的邮件头。对那些始发于你站点的邮件来说，这样就可纠正其中的错误信息。

■ 通常说来，配置有误的站点会将邮件退回发件系统，声称“该邮件不属于本站点”，因为这些站点的配置中，没有它们的 PSEUDONYMNS或等同设置。对从你的站点发送到此类站点的邮件消息来说，它们信封上的所有主机和域信息都是可能剔除掉的。

14.5.3 强制邮件通过UUCP进行传输

理想世界中（从因特网的角度来看），所有主机在域名服务内都有记录，并利用完整的域名形式，发送邮件。

如果你碰巧通过 UUCP和此类站点通信，就可以通过 uucpxtable，对这些邮件的主机名定域，从而强制邮件通过点到点 UUCP连接，而不是通过自己的默认邮件服务器。

14.5.4 防止邮件通过UUCP进行传输

但往往事与愿违。系统经常都可能有大量的直接 UUCP连接，这些连接不常用或始终不如默认邮件服务器或中转主机那么可靠和有用。

比如，Seattle区内有许多系统，在二进制程序发布时，它们通过匿名 UUCP交换各种二进制程序。这些系统只有在必要的时候，才采用 UUCP通信，所以和通过多个非常可靠的网关跳及普通（始终都是可用的）中转主机传输邮件相比，它的速度要快得多。

14.5.5 按需运行sendmail队列

要想立即处理排队等候的消息，只须键入“ /usr/lib/runq ”。这样就可利用恰当的选项调用

sendmail，令sendmail立即运行排队等候处理的作业，而不是等候下一次预定运行。

14.5.6 报告邮件特征

许多站点管理员（和其负责人）都非常关心始发、经由或抵达本地站点的邮件通信量。要想了解邮件通信量，方法很多。

■ sendmail附带一个实用程序，名为mailstats，该实用程序读取一个名为/usr/local/lib/mail/sendmail.st的文件，从而报告每个邮件服务器传输的邮件消息数和字节数，这些邮件服务器是sendmail.cf文件中采用的。该文件必须由本地管理员，针对将要发生的 sendmail记录手工创建。运行合计的清除通过删除并重建sendmail.st文件这一方式来实现。

■ 关于谁使用了邮件以及本地系统已发送、接收或中转了多少邮件这方面的报告，最好利用syslogd(8)，打开邮件调试。这通常意味着要从你的系统启动文件运行 /etc/syslogd后台程序，并在/etc/syslog.conf(5)内增加一行。

如果你用sendmail.debug，而且邮件量为中到大型， syslog输出量就会非常之大。 syslogd的输出文件需要通过crond(8)定期循环或清除。

可以总结syslogd邮件记录的常用实用程序有许多。其中最著名的便是 syslog-stat.pl，这是一个Perl脚本，是随同Sendmail+IDA源代码一起发布的。

14.6 二进制附件的合成和匹配

对电子邮件传输和投递代理来说，不存在标准配置的说法，也没有确切的目录结构。

但是，仍有必要保证系统各部分配置（ USENET新闻组、邮件和 TCP/IP）与本地邮件投递程序（lmail、deliver等等）、远程邮件投递程序（rmail）及邮件传输程序（sendmail和smail）的位置保持一致。此类假设一般不会编入文档，尽管利用字串命令有助于判断自己需要哪些文件和目录。下面是我们在使用已发布二进制程序和源代码时碰到的一些问题。

■ TCP/IP的Net-2二进制程序的某些版本中，有一些服务是为一个名为umail的程序定义的，而不是sendmail。

■ 各种elm端口和mailx查找的是/usr/bin/smail投递代理，而不是sendmail。

■ Sendmail+IDA有一个内置的本地邮件服务器，用作投递服务器，但它位于 /bin，而不是更为常见的/usr/bin下。

为避免通过源代码建立所有邮件客户机的麻烦，我们一般用相应的软链接来代替它们。

14.7 获取更多的信息

要想了解sendmail的更多信息，可供选择的资源非常之多。比如，可以参考 MAIL Howto文档，其定期在comp.answers发布。还可以通过匿名 FTP，从rtfm.mit.edu获得相关资料。但最恰当的地点是Sendmail+IDA源代码。查看源目录下的ida/cf目录，便可看到DBM_GUIDE、OPTIONS和sendmail.mc文件了。

第15章 网络新闻

网络新闻的起源要追溯到1979，当时，两个大学生，Tom Truscott和Jim Ellis想到利用UUCP把计算机连接起来，供各用户交换信息之用。于是，他们便于自己就读的北卡罗莱大学内，建立了一个由三台计算机组成的小型网络。

最初，信息的传输是用大量外壳脚本（后来采用C语言重写）来处理的，但这些脚本从来没有公开发布过。因为，它们很快就被“A”新闻替代了，后者是第一个公开发布的新闻软件。

设计“A”新闻软件的时候，并没有打算让它对每天每个新闻组大量文章进行处理。但随着信息量的不断增大，它已经不堪重负，所以Mark Horton和Matt Glickman对它进行了改写，他们把它叫作“B”版（也就是Bnews）。Bnews的第一个版本是1982年发布的2.1。后来人们又对它不断的扩展，并增加了许多新功能。其当前版本是Bnews 2.11。随着其最后一个正式维护人员转向INN，它也逐渐被废弃不用。

对“A”新闻软件的另一次改写完成并发布于1987年，是Geoff Collyer和Henry Spencer改写的，这被称为“C”版。后来还出现了许多C-News补丁，最为突出的是C-News Performance Release。对那些运行许多个新闻组的站点来说，频繁调用中转新闻所涉及的开支是非常之大的，因为要负责分发进入的所有新闻组文章。Performance Release为中转新闻增加了一个选项，允许管理员在后台程序模式下运行它，这种模式下，程序将自己置入后台运行。

Performance Release是当前最新的C-News版本。

C版本的所有新闻主要用于UUCP网络，但也可用于其他环境中。在TCP/IP、DECNet或相关网络上有效地传输新闻，都需要一个新方案。所以NNTP（网络新闻协议）于1986年应运而生。该协议以网络连接为基础，指定了大量的命令，用于交互传输和获取新闻。

网上有许多基于NNTP的应用。其中之一是nntpd包，它是Brian Barber和Phil Lapsley共同编写的，可用于为局域网内的大量主机提供新闻阅读服务。设计nntpd的目的是完善Bnews和C-News之类的新闻包，为它们赋予NNTP特性。

另一个不同的NNTP包是INN，也称作Internet News。它不只是一个前端，而是独立的新闻系统。它包含一个复杂的新闻中转后台程序，这个程序能够有效维护并发的若干个NNTP链接，因此，它也是许多因特网站点首选的新闻服务器。

15.1 何谓Usenet

关于Usenet，最意外的是它并不属于哪个组织，也没有任何核心网络管理人员。事实上，除了技术说明之外，它只是Usenet知识的一部分，所以你不能确切地定义它是……，只能说它不是……。如果你手中有Brendan Kehoe编写的《因特网中的禅机与艺术》（位于www.cs.indiana.edu/docproject/zen/zen-1.0_toc/html）；或看过他的主页（www.zen.org/~brendan），就能充分了解Usenet的特性。

虽然如此，有人仍然将Usenet定义为由各个独立站点组成的群体，这些站点间彼此交换

Usenet新闻。要想成为Usenet站点，你只须找到另一个Usenet站点，与其拥有者和维护人员达成一致，允许他们与你交换 Usenet新闻即可。为另一个站点提供新闻，被称为“为它发送、分配或交换新闻”，套用另一句大家熟悉的Usenet名言：“提供新闻，你就能上Usenet了”。

Usenet新闻的基本单元是文章。即用户写入或“投递”到网上的消息。为了使新闻系统能够处理它们，所以在这些文章前面加了一些管理信息，也就是所谓的“文章头”。它与因特网邮件标准RFC-822中规定的邮件头极为相似，由若干个文本行组成，每行的格式是这样的：字段名:字段值。

注意 Usenet新闻消息格式是在RFC-1036“USENET消息交换标准”中指定的。

新闻组文章被提交到一个或多个新闻组。我们可以把新闻组想像为一个聚集了与某一主题相关的文章的场所。所有的新闻组都是以分层式结构来组织的，各组的组名便代表它在这个结构中的位置。这样，一眼就能看出一个新闻组的主题是什么。比如，任何人都可从comp.os.linux.announce这个新闻组名得知，该新闻组用于发布名为 Linux的计算机操作系统之相关消息。

提交之后，这些新闻组文章便在愿为该组传播新闻的所有 Usenet站点间进行交换。两个站点同意交换新闻后，便可根据自己喜好，自由交换新闻组，甚至可加入自己的本地新闻结构。例如，groucho.edu有一条指向barnyard.edu（是一个主要的新闻发送站点）和若干条指向次要新闻发送站点的链接。现在， Barnyard学院可能收到了所有的 Usenet新闻组，而GMU只想传输少数主要的新闻结构比如（ sci、comp、rec等）。有的下游站点，比如说一个叫做brewhq的UUCP站点，打算传输比前者还要少的新闻组，因为它们缺乏网络和硬件资源。另一方面， brewhq可能想收到fj结构中的新闻组， GMU却没有将这些新闻组传输过来。因此，它维护了另一条到gargleblaster.com的链接，后者将传输所有的 fj新闻组结构，并把它们发送给brewhq。

15.2 Usenet如何对新闻加以控制

如今， Usenet增长比例日趋加大。以传输整个网络新闻的站点为例，它们每天传输的数据量一般都在3~5GB之间。这当然要求对新闻加以更多的控制。因此，为大家讲讲多数系统怎样处理Usenet新闻是非常必要的。

注意 如果想了解最新的Usenix主题会（会上涉及了一些非常有趣的主题），一定要到这里看看：www.infosys.tuwien.ac.at/staff/pooh/papers/NewsCacheHP/。

新闻在网络间的分发是由不同的传输点来完成的。过去常用的是 UUCP，现在则主要由因特网站点负责传输。采用的路由被称作“扩散式路由选择”（flooding）。每个站点维护许多指向其他站点的链接（新闻发送），本地新闻系统发出或收到的任何一篇文章都将被转发到链接站点，但如果链接站点上已有这些文章的话，它们就会被丢弃。通过查看路径 :文章头字段，站点就可以找到文章经历了哪些站点。文章头中包含一个所有系统列表，通过 bang路径表达式可看出文章经历了哪些系统。

为了区分文章和识别文章是否重复， Usenet文章必须传输一个消息 ID（是Message-ID:header字段中指定的），这个ID由发送站点名和一个序列号组合而成，即< serial@site>。针对每篇处理过的文章，新闻系统都会把这个 ID记录在历史文件内。要检查所有新到的文章，

可查看这个历史文件。

任何两个站点间的通信流都要受两个标准的限制：其一，一个文章对应一个分配（定义在Distribution:header字段内，用于把该文章的发送限定在特定的站点组内。其二，参与交换的新闻组要受到发送和接收系统双方的限制。允许传到某一个站点的新闻组集和分配通常保存在sys文件内。

文章的绝对编号通常要求对前面提到的方案进行改进。 UUCP网络本身用于定期收集文章并把收集到的文章归入一个单一的文件内。该文件再被压缩，发送到远程站点。这就是所谓的“ 批处理法 ”（batching）。

另一种方法是采用ihave/sendme协议，它可防止重复性的文章从本地传输到远程站点，从而节省网络带宽。它采取的作法不是把所有文章放入 batch文件，并将它们一并发送到远程站点，而是，只将文章的消息ID合成一条大型的“ ihave ”消息，再将它发送到远程站点。然后，读取这条消息，把它和自己的历史文件进行比较，并在返回的“ sendme ”消息中，列出自己需要的文章列表。最后，只有消息中的那些文章才被发送到远程站点。

当然，ihave/sendme只适用于这种情况：交换新闻的两个大型站点各自接收数个发送站点发来的文章，两者频繁交换文章。

因特网上的站点一般依靠基于 TCP/IP的软件来进行传输。它采用的是“ 新闻传输协议 ”（NNTP，具体说明参见RFC-977）。它在新闻发送双方进行新闻的传输，并提供对远程主机上的个体用户的访问。

NNTP可识别三种新闻传输方式。其一是 ihave/sendme的实时版本，也称为“ 推 ”（pushing）新闻。其二是“ 拉 ”（pulling）新闻，这种方式中，客户机请求一份列出指定新闻组或结构中文章的列表（这些文章是在指定日期后到达服务器站点的），并选出自己历史文件内没有的那些文章。第三种方式则用于交互式阅读，允许你或你的新闻阅读机获取指定新闻组的文章和投递文章头信息不完整的文章。

每个站点上，新闻都是保存在 /var/spool/news下面的一个目录结构内，每篇文章在一个单独的文件内，每个新闻组在一个单独的目录内。目录名由新闻组名和路径部分组成。因此，comp.os.linux.misc文章被保存在/var/spool/news/comp/os/linux/misc内。新闻组内的文章根据其到达顺序，被编了号。这个编号被当作文件名使用。目前，在线文章的编号范围被保存在一个名为active的文件内，该文件同时充当你的站点所知的新闻组列表。由于磁盘空间是有限的，所以隔段时间，就应该删除某些文章。这就是所谓的“ 期满 ”。通常情况下，来自特定新闻组和结构的文章都会在它们抵达之后的既定天内期满。这个期限可由投递者改写，具体做法是在文章头的Expires:field字段内，指定期满日期。

第16章 C-News

Netnews所用的软件包中，最常用的是 C-News。你可从ftp://ftp:cs.toronto.edu/pub/c-news/c-news.tar.Z下载它。它是为通过UUCP链接传输新闻的站点设计的。本章将详细讨论关于C-News的几个重要概念及其基本安装和维护。

C-News将自己的配置文件保存在 /usr/lib/news内，它的多数二进制程序都保存在/usr/lib/news/bin目录内，文章则保存在/var/spool/news目录下。实际上，你还应该保证这些目录内的所有文件都归用户新闻和组新闻拥有。许多问题都是因为 C-News不能对这些文件进行访问而引起的。对你而言，在行动之前，一定要利用 su，使自己成为用户新闻，这是非常有用的。唯一例外的是setnewsids，它用于设置某些新闻程序真正的用户 ID。它的拥有者必须是root，而且必须设置setuid位。

接下来，为大家详细讲解所有的 C-News配置文件，并向大家展示怎样才能使自己的站点正常运行。

16.1 新闻投递

文章被发送到C-News的方式有许多种。本地用户投递一篇文章时，新闻阅读机通常会把它交给inews命令，该命令将文章头信息补充完整。从远程站点发来的新闻，不管是一篇单独的文章，还是整个的batch，都被交给rnews命令处理，该命令将它保存在/var/spool/newsin.coming目录中，稍后再由newsrun从这个目录中将其剔出来。但是，这两种方法中，不管采用哪一种，文章最终都会被交给relaynews命令处理。

对每篇文章来说， relaynews命令都会首先检查它是否在本地站点出现过，也就是说在历史文件内查找其消息ID。重复性的文章将被丢弃。然后，relaynews命令再查看Newsgroup:header字段，找出本地站点是否请求得到新闻组内的文章。如果是，新闻组就会被列入 active文件，relaynews试着将本地站点请求的文章保存在新闻假脱机区内的相应目录下。如果这个目录不存在，它就会创建一个。然后，该文章的消息 ID被记入历史文件内。另一方面，如果本地站点没有请求文章， relaynews就会把它丢弃。

就“进入文章”(incoming article)已经被投往一个新闻组而言，如果 relaynews不能保存进入文章是因为这个组没有包含在 active文件内，那么这篇文章就会被移入垃圾组（你站点上存在的新闻组和你站点希望接收的新闻组是有区别的。举个例子来说，订阅列表可能指定comp.all，意思是comp结构下的所有新闻组，但对你的站点来说，则是只列出 active文件内comp新闻组的数目。投向这些新闻组的文章会被移入垃圾组）。relaynews还可以检查出陈旧的或过时的文章并将它们丢弃。如果由于其他原因，进入的 batch文件失败，它就会被移到/var/spool/news/in.coming/bad，错误消息也会被记录下来。

随后，利用为每个特定站点指定的传输方式，这篇文章被中转到其他的所有站点（它们都请求了这些组的新闻）。为了保证不将文章发到已经看过它的站点，要对每个目标站点和该文章的Path:header字段进行核对，该字段中包含一份站点列表，表明这篇文章已经经过了哪

些站点，它采用的是路径格式。只有目标站点名不在这份列表内时，这篇文章才会被转发给它。

虽然C-News可用于NNTP环境中，但它常用于UUCP站点间的新闻中转。为了把新闻投递到一个远程UUCP站点 —— 无论是单独的文章，还是 batch文件 —— 需要在远程站点上利用 uux来执行rnews命令，并按标准输入，将文章或 batch文件发送给它。

在具体站点上采用“成批处理法”（batching）时，C-News不会立即发送任何进入的文章，而是将其路径名添加到一个文件内，该文件通常是 out.going/site/togo。通过crontab条目，周期性地执行batcher程序（注意，这个条目应该是新闻的 crontab，目的是不干扰文件的访问许可权），把文章放入一个或多个文件内，对它们进行选择性地压缩，最后在远程站点把它们发送给rnews。

16.2 安装

要安装C-News，须通过untar操作，把文件放入恰当的位置，并对随后黑点列出的配置文件进行编辑（这些配置文件都位于 /usr/lib/news内）。其格式将在随后进行说明。

注意 看了下面的描述，大家便可了解新闻是如何通过relaynews的：

1) 你站点上存在的新闻组和你的站点希望接收的新闻组之间，是有区别的。举个例子来说，订阅列表可能指定comp.all，意思是comp结构下的所有新闻组，但对你的站点来说，则是只列出active文件内comp新闻组的数目。投向这些新闻组的文章会被移入垃圾组。

2) 注意，它应该是新闻的crontab，目的是不干扰文件的访问许可权。

■ sys —— 虽然使用all/all始终是万无一失的，但你仍然必须修改描述系统的 ME行。还必须为向其发送新闻的每个站点增加一行。如果你是叶子站点，只需要一行，将所有本地生成的文章发送到你的发送站点。假设你的发送站点是 moria，那么你的sys文件就应该像这样：

```
ME:all/all::
moria/moria.orcnet.org:all/all,!local:f:
```

■ organization —— 公司或组织名。比如，Virtual Brewery公司。在你的家用电脑上，输入“private site”（私人站点）或其他自己喜欢的名字。如果没有自定义这个文件，你的站点就叫作配置不当。

■ mailname —— 站点的邮件名。比如vbrew.com。

■ whoami —— 供收发新闻之用的站点名。通常采用 UUCP站点名，比如vbrew。

■ explist —— 编辑这个文件时，应该让它能反映某些特殊新闻组的期满时限。磁盘空间此时显得非常重要。为了创建一个新闻组的初始结构，须从你的新闻发送站点获得 active和newsgroups文件，并把它们安装在 /usr/lib/news内，确保它们归news拥有，并且其许可模式是644。从active文件内删除所有的 to.*新闻组，增加 to.mysite和to.feedsite和垃圾组以及控制。to.*新闻组常用于交换 ihave/sendme消息，但你应该创建它们，不管你是否会用到ihave/sendme。随后，在active文件的第二和第三个字段内，替换文章编号，这是通过下面的命令来完成的：

```
# cp active active.old
# sed 's/ [0-9]* [0-9]* / 0000000000 00001 /' active.old active
# rm active.old
```

第二个命令是一个sed(1)调用，也是我本人比较喜欢的一个命令。这次调用将分别用零字串和000001字串来替换两个数位字串。

最后，创建新闻假脱机目录和子目录，这些目录供进入和外出的新闻使用：

```
# cd /var/spool
# mkdir news news/in.coming news/out.going
# chown -R news.news news
# chmod -R 755 news
```

如果你使用的是C-News的最新版本，还必须在新闻假脱机目录内创建 out.master目录。

如果你使用的新闻阅读机源于一个不同于自己正在运行的 C-News程序，你可能会发现新闻假脱机是在/usr/spool/news上，而不是在/var/spool/news内。如果你的新闻阅读机找不到任何文章，就应该在/usr/spool/news和/var/spool/news之间，创建一个象征性的目录。

现在，准备接收新闻。注意，除了前面列出的目录外，你不必创建任何目录，因为 C-News每次从一个还没有假脱机目录的新闻组收到文章时，都会创建它。

对交叉投递文章的所有新闻组来说，尤其如此。稍隔一会儿，你就会发现自己的新闻假脱机目录中充满了从未订阅的新闻组目录，比如 alt.lang.teco。怎样防止这类情况的发生呢？答案是：从 active删除所有不想要的新闻组，或定期运行一个外壳脚本，该脚本将删除/var/spool/news下面的所有空目录（当然， out.going和in.coming除外）。

C-News需要用户向其发送错误消息和状态报告。默认情况下，是 usenet。如果你采用的是默认设置，就必须为用户设置一个别名，该用户把自己的全部邮件都转发到一个或多个负责人处（第13和14章详细讨论了 smail和sendmail的具体做法）。另外，还要通过把环境变量NEWSMASTER设置为相应的主管名，改写这一行为。在新闻的 crontab文件内，以及每次手工调用一个管理工具时，都必须如此。只有这样，安装别名才可能比较简单。

在探测/etc/passwd时，一定要保证每个用户在密码文件的 pw_gecos字段（即第四个字段）内有其真名。对Usenet netiquette来说，发送端的真名出现在文章的 From:内。当然在使用邮件时，总希望如此。

16.3 sys文件

sys文件位于/usr/lib/news内，用于控制你收到的结构目录，并将其转发到其他站点。虽然addfeed和delfeed之类的维护工具可以使用，但我觉得最好手工对该文件进行维护。

sys文件内包含一些条目（针对你把新闻转发到的那些站点）和针对你将收到的新闻组的说明。典型条目如下所示：

```
site[/exclusions]:grouplist[/dist list][:flags[:cmds]]
```

利用反斜杠 \ 的话，条目可能会持续到新行。“ # ”号表示它是批注。下面是sys条目的定义：

■ site —— 该条目适用的站点名。通常选用站点的 UUCP名。sys文件内，也必须有用于你自己站点的条目，不然，你自己是不能接收任何新闻组文章的。

特殊站点名ME代表你的站点。ME条目定义了你希望本地保存的所有新闻组。与 ME行不匹配的文章将被置入垃圾新闻组。

由于C-News会将站点和Path:header字段内的站点名进行核对，所以，必须保证两者真正一致。有的站点采用的是这个字段内的完整形式的域名，有的则采用 news.site.domain之类的

别名。为避免文章被返回这些站点，必须把这些站点添加到例外列表（ exclusion list）中，中间用逗号隔开。

以moria站点为例，它的site字段中将包含moria/moria.orcnet.org这个名字。

■ grouplist —— 这是一个用于特定站点的新闻组和结构订阅列表，各组或分层结构间用逗号隔开。怎样指定分层结构呢？答案是：给出该分层结构的前缀（比如 comp.os，表示名字中有这个前缀的所有新闻组），再加上关键字all，后者仅供选择（比如comp.os.all）。

在分层结构或新闻组前加上感叹号后，这些结构或组便被排除在转发对象之外。如果对新闻组和列表进行核对，将采用最接近的匹配。比如，如果 grouplist中包含

```
!comp.comp.os.linux.comp.folklore.computers
```

则comp分层结构中，除了 comp.folklore.computers和comp.os.linux下面的所有新闻组外，再已没有新闻组被投递到那个站点。

如果站点请求转发你收到的所有新闻，将 all作为grouplist输入即可。

■ distlist —— 向grouplist偏移一个斜杠，其中包含即将转发的程序列表。再次提醒大家注意，可在特定的程序前加一个感叹号，将其排除在外。要转发所有的程序，用 all来表示。省略distlist意味着采用all列表。

例如，你可以用一个程序列表 all,!local来防止只限于本地使用的新闻被投递到远程站点。

至少有两个程序是常用的，它们是 world和local。如果用户什么也没有指定，就会采用前者，它是默认设置。其他还有些程序应用于特定的区、州、国家等。最后，还有两个只适用于C-News的程序，那就是sendme和ihave，它们用于sendme/ihave协议。

这些程序的用法一直是业界争论的主题。比如，有的新闻阅读机只用顶级的分层结构，创建一个假的程序，例如向 comp.os.linux投递新闻的comp。应用于区的程序同样不可信，因为在通过因特网投递时，新闻可能根本不经过你所处的那个区。但是应用于某个公司的程序号却是相当有意义的，比如，它可防止公司的机密文件通过公司网络外泄。但要达到这样的目的，可采用更好的方式，那就是创建一个单独的新闻组或分层结构。

■ flags —— 描述发送站点的特定参数。该条目可以为空，也可以合用下面的标记：

- F —— 启用批处理法。
- f —— 完全等同于F标记，但允许C-News更精确地计算外出批处理文件的大小。
- I —— 令C-News建立一个适合于 ihave/sendme使用的文章列表。对 sys和batchparms文件所做的其他修改都需要启用 ihave/sendme。

■ n —— 为活动的NNTP传输客户机（比如 uutpxmit，参见第19章）创建批处理（ batch）文件。这个批处理文件中包含文章名及其消息 ID。

■ I —— 令C-News建立一个适合于 ihave/sendme使用的文章列表。对 sys和batchparms文件所做的其他修改都需要启用 ihave/sendme。

■ n —— 为活动的NNTP传输客户机（比如 uutpxmit，参见第19章）创建批处理（ batch）文件。这个批处理文件中包含文章名及其消息 ID。

■ L —— 要求C-News只传输投递你站点上的文章。该标记后面如果跟一个十进制数 n，则令C-News只传输 n次网关跳（从你的站点开始计数）范围内投递的文章。 C-News根据Path:field来判断跳次数。

■ u —— 要求C-News只对非中继新闻组的文章进行批处理。

■ m —— 要求C-News只对主持式新闻组的文章进行批处理。最多可用下列标记之一： F、f、I和n。

■ cmds —— 该字段中包含准备针对每篇文章执行的命令，启用批处理时除外。文章将作为标准输入发送给命令。它只适用于小型的文章发送；不然的话，发送和接收端系统的载入量都会大得惊人。

默认命令是

```
uux - -r -z system!rnews
```

将文章作为标准输入，把它发送到远程系统，便调用了 rnews。对该字段内给出的命令来说，它们所用的默认搜索路径是 /bin:/usr/bin:usr/lib/news/bin/batch。后一个目录中包含许多外壳脚本，这些脚本名均以via开头；具体情况参见本章稍后的说明。

如果利用F、f、I或n四个标记之一，启用了成批处理法， C-News就希望能够在该字段内找到一个文件名，而不是通过命令去找。如果文本名没有以斜杠 / 开头，就会被假定与/var/spool/news/out.going有关。如果该字段为空，它就会采用默认的 system/togo。

在设置C-News时，很可能你必须编写自己的sys文件。为了帮助大家进行编写，下面给出了一个示例文件，该文件是用于vbrew.com的，你可以从中复制自己需要的内容。

```
# We take whatever they give us.
ME:all/all::

# We send everything we receive to moria, except for local and
# brewery-related articles. We use batching.

moria/moria.orcnet.org:all,!to,to.moria/all,!local,!brewery:f:

# We mail comp.risks to jack@ponderosa.uucp
ponderosa:comp.risks/all::rmail jack@ponderosa.uucp

# swim gets a minor feed
swim/swim.twobirds.com:comp.os.linux,rec.humor.oracle/all,!local:f:

# Log mail map articles for later processing
usenet-maps:comp.mail.maps/all:F:/var/spool/uumaps/work/batch
```

16.4 active文件

active文件位于/usr/lib/news内，它列出了你的站点上已知的所有新闻组和目前在线的文章。一般很少碰这个文件，但为了完整起见，仍然为大家讲讲它。它其中的条目采用这样的形式：

```
newsgroup high low perm
```

newsgroup当然是指新闻组名。 low和high字段指目前能用的文章之最低和最高编号。如果目前无文章可用，低就等于高 +1。

这至少是low字段的本义。但是，出于提高效率的原因， C-News没有更新这个字段。如果没有依赖该字段的新闻阅读机，不更新这个字段根本就没什么大的损失。例如， trn会查看这个字段，进而判断是否可以将某些文章从自己的线程数据库内清除。为了更新 low字段，你

必须定期运行updatemin命令（早期的 C-News版本中，则是upact）。

perm是一个参数，详细说明了被授权访问该新闻组的用户。它可采用下面的值：

■ y —— 允许用户向该新闻组投递文章

■ n —— 不允许用户向该新闻组投递文章。但是该新闻组仍然是可以读取的。

■ x —— 该新闻组已经被本地取消。通常发生在新闻管理员（或其上级）对投递到特定新闻组的文章采取攻击性行为时。

对该新闻组收到的文章来说，虽然它们被转发到了请求获得它们的站点，但不能实现本地保存。

■ m —— 表示主持式新闻组。用户试图向该新闻组投递文章时，一个聪明的新闻读取器将向她发出信号，并将文章发送给主持人。主持人的地址是从 /usr/lib/news内的（moderator）主持人文件内选出来的。

■ = real-group —— 这标明newsgroup作为另一组，即 real-group的本地别名。所有发送到newsgroup的文章将重定向到它。

C-News中，一般都不必直接访问这个 active文件。利用addgroup或delgroup，可在本地增添或删除新闻组。为整个 Usenet增添或删除新闻组，通常分别发送一条 newsgroup和rmgroup控制消息即可。绝不可以自行发送此类的消息！关于创建一个新闻组所需的步骤，可参考news.announce.newusers内定期发布的文章。

与active密切相关的一个文件是active.times。只要创建了一个新闻组，C-News就会将相关消息记入这个文件，该文件中有创建的新闻组名、创建日期，是由新闻组控制消息创建的，还是本地创建的，以及创建者是谁。对通知用户新近建立了哪些新闻组的新闻阅读机来说，这是非常方便的。该文件还可以供 NNTP的NEWSGROUP命令使用。

16.5 新闻组文章的批处理

新闻大批处理采用的特殊格式和 Bnew/s、C-News及INN一样。每篇文章都以这一行开头：

```
#! rnews count
```

count指的是文章内的字节数。采用成批处理法压缩之后，文件就被压缩为一个整体，以另外一行开头，表示该消息将用于解压。标准的压缩工具是 compress，用下面这行标记：

```
#! Cunbatch
```

有时，必须通过 mail软件（从所有数据中删除第 8位）发送批处理文件时，压缩过的批处理文件可利用一种名为c7编码的技术得以保护；这些批处理文件统统用 c7unbatch标记出来。

批处理文件被发送给远程站点上的 rnews时，它会检查它们的标记，并利用相应的解压方案对它们进行处理。有的站点还利用了其他的压缩工具，比如 gzip，所以它们采用的是gzipped标记，而不是zunbatch标记。C-News不能识别这些非标准的文章头，所以你必须修改源代码，以支持它们。

C-News中，文章的批处理是由 /usr/lib/news/bin/batch/sendbatches来执行的，它采用了一份取自site/togo文件的文章列表，并把这些文章放入几个新闻批处理文件内。根据通信量的大小，批处理文件应该每小时执行一次，或更为频繁。

批处理操作由/usr/lib/news内的batchparms文件控制。该文件描述了每个站点允许的批处理文件的最大字节数、准备采用的批处理和可选的压缩程序以及用于把它投递到远程站点的

传输程序等。你可以一个站点一个站点地指定批处理参数，对没有明显提及的站点，则采用默认参数集。

为了针对特定的站点执行批处理，可这样调用它：

```
# su news-c"/usr/lib/news/bin/batch/sendbatches site"
```

当不含参数情况下被引发时， sendbatch处理所有批处理队列。 all的意义依batch参数中缺省条目是否出现而定。如果出现，在 /var/spool/news/out.going下的所有目录都被检查，否则，将循环通过所有 batch参数条目。注意当浏览 out.going目录时， sendbatch只取站点名中不含。或@的那些目录。

当安装C-News时，你很可能在发布盘中找到batch参数文件，这个文件包含一些缺省条目，这样你就不会接触batch参数文件。我们现给出它的格式，每行有 6个字段，由空格分隔开：

```
site size max batcher muncher transport
```

■ site —— 条目所应用的站点名。该站点的 togo文件必须驻留在新闻假脱机目录下面的out.going/togo内。/default/站点名代表的是默认条目。

■ size —— 已创建的批处理文件的最大字节数（指压缩之前）。对大于这个数目的文章，C-News将执行一个违例，并把它们放入另一个单独的批处理文件中。

■ max —— 批处理文件的最大数目。这些批处理文件是在准备将批处理文件发送到特定站点之前，创建和安排传输的。这个字段非常有用，特别是远程站点长时间没有运行时，因为它可防止C-News把数额庞大的新闻批处理文件统统塞到你的 UUCP假脱机目录内。

C-News利用/usr/lib/news/bin内的排队脚本，判断排队等候批处理文件的编号。 Vince Skahan编写的newspak中包含了一个兼容BNU UUCP的脚本。如果你打算采用另类的假脱机目录，比如说泰勒式UUCP，就必须自行编写。如果不在乎假脱机文件的多少（因为你的计算机属你个人专用，而且没有编写兆字节的文章），就可以用一个简单的exit 0语句替换这个脚本的内容。

对batcher字段来说，其中包含的命令用于生成一个批处理文件，其依据是 togo文件内的文章列表。对定期发送的文章来说，这个字段通常是 batcher。出于其他的目的，还可提供别的batcher。例如， ihave/sendme协议要求文章列表被转换为 ihave或sendme控制消息，这些消息被投递到新闻组to.site。文章列表的转换是由batchih和batchsm来执行的。

muncher字段指定压缩所用的命令。通常，这个命令是 compcun，它是一个生成压缩批处理文件的脚本（以C-News为例， compcun采用12位选项来压缩文件，这是多数站点都不采用的。如果你可以对其进行复制，比如 compcun16，采用的就是16位压缩选项。但和前面的压缩选项相比，改进并不明显）。另外，也可自行提供一个采用 gzip的muncher，比如说gzipcun（强调：你必须自行编写它）。除此以外，还必须保证远程站点上有解压命令，并且能够识别采用gzip压缩的文件。

如果远程站点上没有解压命令，可能需要指定 nocomp，表示不做任何压缩。

最后一个字段是transport，它描述了准备采用的传输命令。针对不同类型的传输，可采用不同的标准命令，这些命令以 via.开头。sendbatches把命令行上的目标站点名投递给它们。如果batchparms条目不是/default/，它就从site字段中截取站点名，具体做法是剔除该字段内的第一个句点或斜杠及其以后的内容。如果batchparms条目是/default/，就采用out.going中的目录名。

有两个命令利用 uux，在远程系统上执行 rnews。它们是：viauux和viauuxz。后者为旧版

本的uux设置了-z标记，阻止它为已投递的文章返回成功消息。另一个命令， viamail，通过邮件，向远程系统上的用户 rnews发送文章批处理文件。当然，这要求远程系统为其本地新闻系统发送rnews邮件。要想得到一份完整的传输命令列表，可参考 newsbatch(8)手册。

后三个字段的所有命令都必须位于 out.going/site或/usr/lib/news/bin/batch内。这些命令之中，大部分都是脚本，所以你可以根据自己的需求，轻松地定制新工具。它们被当作管道调用。文章列表被当作标准输入，发送给 batcher，后者再生成作为标准输出的 batch文件。然后，batch通过管道输送到muncher等等。

下面给出一个示例文件。

```
# batchparms file for the brewery
# site        | size   |max    |batcher  |muncher    |transport
#-------+-----+---+-----+------+------
/default/       100000  22      batcher   compcun     viauux
swim             10000  10      batcher   nocomp      viauux
```

16.6 对新闻进行过期处理

Bnews中，期满一直由一个名为 expire的程序执行，该程序采用新闻组列表作为参数，还有一个时间说明，表示文章的到期日是多少。为了让不同的分层结构在不同的时间期满，你不得不编写一个脚本，令其针对每个结构，单独调用 expire。C-News提供了一个更为方便的解决之道：在一个名为 explist的文件内，指定特定新闻组及其期满时间间隔。这个名为 doexpire的命令利用cron，通常每天运行一次，并根据 explist文件内的新闻组列表，对所有的新闻组进行处理。

有时，你可能想在特定新闻组已经到期之后，仍然持有这些新闻组内的文章；比方说，打算保存已投递到 comp.sources.unix的程序。这叫作“ 归档 ”。explist允许你将新闻组标记为“ 归档 ”。

explist文件内的条目如下所示：

```
grouplist perm times archive
```

grouplist（新闻组列表）是该条目所应用的一个新闻组列表，各组间用句点隔开。分层结构的指定是这样的：给出新闻组名的前缀，后面可以选择性地加上 all。比如，以应用于 comp.os下面所有新闻组的一个条目为例，在 grouplist这个地方，既可以输入 comp.os，又可以输入comp.os.all。

在判断一个新闻组的新闻到期时，应该按照指定的顺序，将该新闻组名和 explist内的所有条目进行核查。然后，采用第一条与之匹配的条目。例如，要在 4天之后，把大部分 comp内的新闻丢弃，你自己想保留一星期的 comp.os.linux.announce不计入内，你只须选用后者的条目（指定7天的到期时间），然后才是用于comp的条目（指定4天的到期时间）。

perm字段详细说明了该条目是应用于主持式新闻组、非主持式新闻组还是任何类型的新闻组。它可采用的值有：m、u和x，分别代表主持式组、非主持式组和任何类型的新闻组。

第三个字段times，通常只包含一个单独的数字。如果这些文章头的 Expires:字段内，没有为其人为地分配到期日期，这个数指出哪些文章将在多少天后到期。注意，这里的天数是从文章抵达你的站点那一天开始计算，而不是投递之日。

然而，times字段还可以更为复杂。其中可包含的数字多达 3个，中间用破折号“ - ”隔开。

第一个数指出，多少天后，对文章进行过期处理。这个数除了零值以外，较少采用别的值。第二个数是前面提过的默认到期值。第三个数指出，多少天后，无条件地对文章进行过期处理，不管它是否有Expires:字段。如果只指定中间那个数，其他两个数都将采用默认值。这些值是利用特殊的/bounds/条目来指定的，详情随后介绍。

第四个字段archive，指出是否将新闻组归档以及归入哪个目录。如果不打算采用归档，就应该采用破折号“ —— ”。不然，就采用一个完整路径名（指向一个目录）或“ @ ”符号。“ @ ”符号代表默认的归档目录，这个目录是必须为 doexpire指定的，通过在命令行上采用 -a 标记，就可以完成指定了。归档目录应该属 news拥有。例如， doexpire对comp.source.unix的文章进行归档处理时，它会把它保存在归档目录下的 comp/sources/unix目录内，如果没有这个目录的话，它就创建一个。但是，归档目录本身是不能创建的。

作为doexpire命令基础的explist文件，有两个特殊的条目。采用的不是新闻组列表，而是两个关键字：/bounds/和/expired/。/bounds/条目中包含的默认值是前面提过的times字段的三个值。

/expired/字段表示C-News将把文章对应的行保存在历史文件内的时间。这个字段是必要的，因为文章一旦到期， C-News不会从历史文件内删除与之对应的行，仍然将其保存在历史文件内，这样一来，就可能导致文章重复的情况。如果你只有一个发送站点，可把这个时间值设小一些。UUCP网络中，则根据你从这些站点获取文章的时限来定，一般建议设为两周。

下面是一个explist示范文件：

```
# keep history lines for two weeks. Nobody gets more than three mont
/expired/                       x       14      -
/bounds/                        x       0-1-90  -

# groups we want to keep longer than the rest
comp.os.linux.announce          m       10      -
comp.os.linux                   x       5       -
alt.folklore.computers          u       10      -
rec.humor.oracle                m       10      -
soc.feminism                    m       10      -

# Archive *.sources groups
comp.sources,alt.sources        x       5       @

# defaults for tech groups
comp,sci                        x       7       -

# enough for a long weekend
misc,talk                       x       4       -

# throw away junk quickly
junk                            x       1       -

# Archive *.sources groups
comp.sources,alt.sources        x       5       @
# defaults for tech groups
comp,sci
                                x       7       -

# enough for a long weekend
misc,talk
                                x       4       -
```

```
# throw away junk quickly
junk                            x       1       -

# control messages are of scant interest, too
control                         x       1       -

# catch-all entry for the rest of it
all                             x       2       -
```

C-News中，采用过期处理时，可能还存在许多问题。其一，你的新闻阅读机可能会依赖于active文件内的第三个字段，该字段内包含在线文章的最低限量。在对文章进行过期处理时，C-News没有对这个字段进行更新。如果你需要（或希望）让这个字段代表实际情形，就需要在每次运行doexpire之后，运行一个名为updatemiin的程序（在旧版本的C-News中，是由一个名为upact的脚本来完成的）。

其二，C-News在进行过期处理时，不会查看新闻组的目录，只查看历史文件，从而得知文章是否已经到期（自1970年1月以来，文章的期满日期一直保留在历史列中间的字段内，以秒计）。如果你的历史文件莫名其妙地没有同步，新闻组文章就可能一直待在你的磁盘上，因为C-News已经当它们不存在了（为什么会这样，我也不得而知，但对我本人来说，这样的事的确常有发生）。补救办法是利用/usr/lib/news/bin/maint内的addmissing脚本，这个脚本将把丢失的文章添加到历史文件或mkhistory内，从头重新建立整个历史文件。别忘了在调用它之前，先成为news用户，不然你就以C-News不能阅读历史文件而告终。

16.7 其他文件

可控制C-News行为的文件有许多，但都不能从根本上控制它。所有文件都驻留在/usr/lib/news内。下面简要谈谈它们。

■ newsgroups（新闻组）—— 它是active文件的附加文件，其中包含新闻组名列表和一行主题说明。C-News收到新闻检查消息时，便自动更新这个文件（参见第18章）

■ localgroups（本地新闻组）—— 如果你有许多本地新闻组，不希望每次收到新闻检查时C-News的抱怨，就可把这些新闻组的名字及其说明放入这个文件内，它们就会在新闻组文件内出现。

■ Mailpaths —— 该文件内包含各主持式新闻组主持人的地址。每一行内包含新闻组名和主持人的邮件地址（偏移一个制表符）。

有两个特殊条目是默认提供的。它们是backbone（骨干网）和Internet（因特网）。两者均以bang路径表达式提供了到最近的骨干站点和可识别RFC 822式地址（user@host）的站点的路径。默认的条目如下所示：

```
internet      backbone
```

如果你已安装了smail和sendmail，没必要更改Internet条目，因为它们能够识别RFC 822式的地址。

只要有用户向其主持人没有显示列出的主持式新闻组投递文章，就可采用backbone条目。如果该新闻组名是alt.sewer，而backbone条目中包含path!%，C-News就会将文章邮寄到path!alt-sewer，希望骨干主机能够对该文章进行转发。采用什么路径，可向你的新闻组发送站点询问。最后才考虑uunet.uu.net!%s这条路径。

■ distributions —— 这个文件不是一个真正的 C-News文件，但它可供有些新闻阅读机和nntpd使用。它包含了你的站点能够识别的程序列表及其结果（或目的）说明。例如， Virtual Brewery就有下面这些文件：

- world
- everywhere in the world
- local
- Only local to this site
- nl
- Netherlands only
- mugnet
- MUGNET only
- fr
- France only
- de
- Germany only
- brewery
- Virtual Brewery only

■ log —— 该文件中包含对所有C-News活动的记录。通过运行newsdaily，便可定期获得这个文件；旧的日志文件副本保存在log.o、log.oo等文件内。

■ errlog —— 这是一个记录所有错误消息的文件，这些错误是 C-News引起的。该文件内没有包含投错新闻组之类的垃圾文章。如果该文件是非空的，就会定期被 newsdaily自动邮寄给newsmaster（新闻主管，是usenet的默认设置）。errlog的清除也由newsdaily负责。旧的错误消息日志文件副本被保存在errlog.o、errlog.oo等文件内。

■ batchlog —— 记录sendmatches的所有运行情况。这个文件较少使用。它也是由 newsdaily来负责的。

■ watchtime —— 每次运行newswatch时创建的空文件。

16.8 控制消息

Usenet新闻协议可识别另类的特殊文章，这些文章引起新闻系统的特定反应或动作。它们叫作控制消息。根据文章头的 Control:字段（其中包含即将执行的控制操作），就能够识别出它们。控制消息有许多类，统统由外壳脚本来操作，这些脚本位于 /usr/lib/news/ctl下。

许多控制消息在C-News处理文章时，没有通知新闻主管，自动执行自己的动作。默认情况下，只有checkgroups（新闻组检查）消息才被交给新闻主管（ RFC-1036上，有一个有趣的原型：实施者和管理员既可允许控制消息得以自动实现，也可安排它们等候一年一次的处理），但你可以通过编辑脚本的方式，修改这个默认设置。

16.8.1 cancel消息

最广为人知的控制消息就是 cancel（取消），用户可用这类消息来取消以前发送的文章。如果文章还在的话，这类消息便有效地将它从假脱机目录中删除。对被该消息影响的新闻组

内的文章来说，只要站点收到过它们， cancel消息就会被转发到这些站点，不管它们是否见过这些文章。这是因为考虑到这种可能：原始文章已经被作废消息延迟。有的新闻系统允许用户取消其他人的消息；不过，这当然是禁忌之事。

16.8.2 newgroup和rmgroup消息

涉及到新闻组创建和删除的消息有两条，它们是 newgroup和rmgroup。对usual结构下的新闻组来说，只有在 Usenet读者举行讨论和选举之后，才能创建。适用于中等结构的创建原则允许一些看似混乱的新闻组。关于这方面的详情，可参考 news.announce.newusers和news.announce.newgroups定期发布的贴子。千万不要自行发送 newgroup和rmgroup消息，除非你能确定你已经得到了许可。

16.8.3 checkgroups消息

checkgroups消息是由新闻管理员发送的，用于令网络内的所有站点将自己的活动文件和Usenet同步。例如，商业性的因特网服务供应商可能会向其客户的站点发送此类的消息，每月一次，针对主要结构的“ 正式 ” checkgroups消息通过comp.announce.newgroups的主持人，被投递到这个站点。但是，该消息是作为一篇普通文章来投递的，而不是控制消息。为了执行checkgroups操作，要把这篇文章保存到一个文件比如 /tmp/check内，然后删除所有内容，只留下控制消息本身，再利用下面的命令，把控制消息投递给 checkgroups脚本；

```
# su news -c "/usr/lib/news/bin/ctl/checkgroups" < /tmp/check
```

这样就将localgroups（本地新闻组）内列出的新闻组添加到你的 newsgroups文件，对其进行了更新。旧的newsgroups文件就会被移到newsgroups.bac中。注意，在本地投递该控制消息是不会有效的，因为inews拒绝接受这样大的文章。

如果C-News发现checkgroups列表和active文件不匹配，它就会出示一个命令列表，这些命令将刷新你的站点并把它投递给新闻管理员。其典型输出如下所示：

```
From news Sun Jan 30 16:18:11 1994
Date: Sun, 30 Jan 94 16:18 MET
From: news (News Subsystem)
To: usenet
Subject: Problems with your active file
The following newsgroups are not valid and should be removed.
alt.ascii-art
bionet.molbio.gene-org
comp.windows.x.intrisics
de.answers

You can do this by executing the commands:
/usr/lib/news/bin/maint/delgroup alt.ascii-art
/usr/lib/news/bin/maint/delgroup bionet.molbio.gene-org
/usr/lib/news/bin/maint/delgroup comp.windows.x.intrisics
/usr/lib/news/bin/maint/delgroup de.answers

The following newsgroups were missing.
comp.binaries.cbm
comp.databases.rdb
comp.os.geos
comp.os.qnx
```

```
comp.unix.user-friendly
misc.legal.moderated
news.newsites
soc.culture.scientists
talk.politics.crypto
talk.politics.tibet
```

从你的新闻系统收到此类控制消息时，不要盲目地相信它。取决于投递 checkgroups消息的人，该消息可能需要少数新闻组或整个结构内的新闻组，所以在删除新闻组的时候，你一定要小心行事。如果发现列出的新闻组是你站点上没有的，就必须利用 addgroup脚本，将它们添加到自己的站点上。把所缺新闻组的列表保存在一个文件内，再将该文件投递给下面的小型脚本：

```
#!/bin/sh
cd /usr/lib/news

while read group; do
    if grep -si "^$group[[:space:]].*moderated" newsgroup; then
        mod=m
    else
        mod=y
    fi
    /usr/lib/news/bin/maint/addgroup $group $mod
done
```

16.8.4 sendsys、version和senduuname

最后是用于查找网络结构的三条消息：sendsys、version和senduuname。它们令C-News分别向发送端返回sys文件、一个软件版本字串和 uuname(1)的输出。C-News的版本消息是相当简明扼要的；它返回一个简单的、没有任何说明的“ C ”。

再次提醒大家注意，你绝对不要执行此类消息，除非你充分肯定它使你（地区性）的网络宕掉。对sendsys消息的响应能够使你的UUCP迅速瘫痪（我还没有在因特网上试过）。

16.9 NFS环境中的C-News

在一个局域网内分发新闻，最简单的方法是把所有新闻保存在一个中央主机内，并通过NFS导出所有的相关目录，以便新闻读者可以直接浏览文章。在 NNTP上采用这种方法的好处是：获取和传输新闻耗费的开支明显降低。另一方面，在一个由不同成分组成的网络内使用NNTP，显然好处多多。这类网络由不同种类的主机组成，或者说其用户在服务器主机上没有对应账号。

在使用NFS时，对于本地主机上投递的文章来说，它们必须被转发到中央主机，因为如果不这样的话，访问管理文件可能会令系统处于不安全的状态，使得文件不连贯。此外，你可能还想通过只读方式导出新闻假脱机，对它进行保护，它也需要被转发到中央主机上。 C-News对此类情况进行的处理是透明的。在你投递一篇文章时，你的新闻阅读机通常会调用inews，把该文章注入新闻系统。这个命令对该文章运行了大量的检查，完成文章头的处理，并在/usr/lib/news内核实文件服务器。如果发现该文件服务器存在，而且其中包含一个不同于本地主机名的主机名，就通过 rsh，在那台服务器主机上调用 inews。由于inews脚本要采用C-News的大量二进制命令和支持文件，所以你必须在本地安装C-News，或通过服务器装入新闻软件。

为了让rsh调用正常运行，每个用户在服务器系统上，都必须有对应的账号，也就是说，他／她能够在不要求密码的情况下登录系统。

另外，要保证服务器内指定的主机名语义上与服务器主机上 hostname(1)命令的输出相匹配，如若不然，C-News在试着投递文章时，就会无休止地循环下去。

16.10 维护工具及任务

尽管C-News非常之复杂，但新闻管理员的生活仍然可能非常之轻松，因为 C-News提供了相当多的维护工具。其中一些通过 cron定期运行，比如newsdaily。使用这些脚本，可大大减轻日常维护和新闻发送工作。

除非特别说明，这些命令通常都位于 /usr/lib/news/bin/maint内。注意，在调用这些命令之前，务必保证自己成为news user。以超级用户的身份运行它们，以便能够对 C-News不能访问的文件进行访问。

■ newsdaily —— 正如其名，这个命令是一天运行一次。它是一个非常重要的脚本，有助于你保证日志文件不至于太大，保持前三次运行记录的副本。另外，它还试着探测异常情况，比如进入和外出目录中的失效批处理文件，试着探测被投递到未知或主持式新闻组的目录等。运行产生的错误消息都将被发送给新闻主管。

■ newswatch —— 这是一个应该定期运行的脚本，用于查找新闻系统中的异常情况，一般一小时运行一次。其目的是探测故障，并把故障报告发送给新闻主管。这些故障将立即对新闻系统的操作性产生影响。它检查的内容包括：没有及时删除的失效锁文件、空的输入批处理文件和磁盘空间是否缺乏。

■ addgroup —— 在你的站点上，本地添加一个新闻组。正确的调用是：

addgroup groupname y|n|m|=realgroup

第二个参数的含义和 active文件内的标记一样，意思是任何人都可向指定组（ y），无人主持式新闻组（n）和主持式新闻组（m），或采用别名的另一个新闻组（realgroup）投递文章。

此外，在新近建立的新闻组内第一篇文章先于 newgroup控制消息（打算建立改新闻组）抵达时，还可打算采用addgroup。

■ delgroup —— 允许你本地删除新闻组。其调用形式如下：

delgroup groupname

仍然必须删除新闻组假脱机目录中保留的文章。另一种办法是把它留给自然事件处理(比如过期处理)。

■ addmissing —— 把遗漏的文章添加到历史文件内。有些文章“ 阴魂不散 ”时，可运行这个脚本。

■ newsboot —— 该脚本应该在系统启动时运行。它把关机时被杀死的新进程遗留下来的锁文件删除，并关闭和执行 NNTP连接遗留下来的所有批处理文件。这里的 NNTP连接已在系统关机时被中断。

■ newsrunning —— 这个脚本驻留在/usr/lib/news/bin/input内，可用于允许对进入的新闻进行批处理。也可通过下面的调用关掉“ 取消批处理 ”：

/usr/lib/news/bin/input/newsrunning off

第17章　NNTP简介

由于采用不同的网络传输，NNTP提供了多种C-News新闻交换方法。NNTP代表的是“网络新闻传输协议”，它不是一个特殊的软件包，而是一种因特网标准，是在RFC-977在正式定义的。它以面向流的连接（通常为TCP连接）为基础，这种连接的一端是网络中处于任何一个地方的客户机，另一端是其磁盘上保存netnews（网络新闻）的主机上的服务器。流式连接允许客户机和服务器几乎没有任何延迟地交互式协商文章传输，这样可尽量避免文章的雷同。有了因特网的高传输速率，新闻传输较之以前的UUCP网络更为快捷。虽然若干年前，一篇文章抵达Usenet的最后一个站点需要两天甚至更长的时间是常有的事，但现在通常只须花不到两天的时间；而在因特网上，甚至可在数分钟之内就能完成。

许多命令允许客户机恢复、发送或投递文章。发送和投递之间的区别在于：后者还涉及到头信息不完整的文章（在NNTP上投递文章时，服务器始终会加上至少一个头字段，那就是Nntp-Posting-Host:。其中包含客户机的主机名）。文章的恢复可供新闻传输客户机和新闻读者使用。这些都令NNTP成为一个出色的工具，在使用NFS时，无须通过必要的中转，就可为本地网络上的许多客户机提供新闻访问。

NNTP还提供了主动和被动新闻传输程序，通称“推”和“拉”。“推”程序和C-News ihave/sendme协议基本一样。客户机通过ihave ＜varmsgid＞命令，为服务器提供文章，而服务器返回相应的响应代码，指出自己是否已有这篇文章，或者说自己是否想收到这篇文章。如果是，客户机就将该文章发送出去，在一个单独的行上，以一个句点结束。

“推”新闻的不足之处在于：它会为服务器系统带来较大的工作量，因为它必须在自己的历史数据库内搜索每一篇独立的文章。

相反的技术是“拉”新闻。在这种情况下，客户机从一个组请求获得指定日期之后抵达的所有文章的一个列表。这一查询是由NEWNEWS命令来执行的。从消息ID的返回列表中，客户机选择那些自己还没有的文章（依次为每篇新文章使用ARTICLE命令）。

另外还有大量便利的命令可供新闻阅读程序利用，使其能分别接收文章标题和文章主体，甚至从指定范围的文章中，取得每一个标题行。这样一来，我们可就将全部新闻（文章）都保存在一台中心主机上。同时，整个网络中的所有用户都利用基于NNTP的客户机程序，读取及发表文章。另外，也可以通过NFS导出新闻目录，具体做法已在第16章详细介绍过了。

对NNTP来说，它存在的一个重要问题在于：某些头脑发达的人士，可以使用虚拟的发送人信息，将文章插入新闻流中。我们将这称为“新闻欺骗”（SMTP —— 简单邮件传输协议 —— 也存在同样的问题）。然而，通过对NNTP进行一定程度的扩展，便可针对特定的命令，要求证明用户的真实身份。

目前，大家可以找到大量NNTP的变种。其中最著名的恐怕要算NNTP Daemon，亦称作“参考NNTP”。它最初是由Stan Barber和Phil Lapsley编写的，用于阐示RFC-977的细节。其最新的版本是nntpd 1.5.11，后面会以它作为范例详细讲述。此外，大家能拿到它的源码，并可自行编译，或使用由Fred van Kempen的net-std二进制封装提供的nntpd。注意nntpd并没有现

成的二进制文件可供直接使用，这是由于存在大量与具体站点对应的值，必须自行编译。

注意 自本书完稿以后，许多情况都发生了变化。后文提到的 INN封装目前正由ISC（Internet软件协会）负责维护，并被公布为最新的“参考”版本。INN的最新版本是2.2，请访问下述网址了解进一步的情况： http://www.isc.org/view.cgi?/products/INN/index.phtml。

在nntpd封装中，包含了一个服务器模块，以及两个客户机模块，分别用于新闻的“拉”与“推”。此外，还有一个用于取代inews的模块。它们要求在一个Bnews环境中使用，但只需作少许调节，就能很好地兼容于 C-News环境。但是，假如你计划使用 NNTP，让新闻阅读程序访问自己的新闻服务器，那么参考版本其实并不是一个很好的选择。因此，我们将只讨论包含在nntpd封装中的NNTP Daemon；至于具体的客户机程序，则暂时不作讨论。

注意 另外还有一个名为“InterNet News”的封装，简称为INN，由Rich Salz编写。它同时具有NNTP和UUCP新闻传输功能，而且对那些大型的新闻交换中心来说，显得更加稳定。通过NNTP传输新闻消息时，这个版本要明显优于nntpd。目前，INN的版本号是inn-1.4sec。Arjan de Vet为我们提供了一个工具包，专门用来在一台机器上构建INN；你可利用FTP从sunsite.unc.edu网站下载，目录是system/Mail。假如想设置INN，请务必参阅与源码配套提供的说明文档。同时，还应仔细阅读网上公布的 INN-FAQ，网址是news.software.b和news.software.inn。

17.1 NNTP 服务器的安装

NNTP服务器的名字叫作 nntpd，可通过两种方式进行编译。至于具体采用哪种方式，取决于新闻系统预计的负载有多大！注意目前找不到一个现成的、编译好的版本，因为在执行程序中，以“硬编码”的方式，强行设置了许多各个站点都不尽相同的默认选项。所有配置都是通过common/conf.h内的宏定义来完成的。

可考虑将nntpd配置成一个独立运行的服务器，令其在系统引导的时候，自 rc.inet2中启动。另外，亦可考虑将其配置成由 inetd负责管理。如果是后一种情况，必须在 /etc/inetd.conf中设置下述条目：

```
nntp stream tcp nowait news / usr / etc / in.nntpd nntpd
```

假如选择前一种情况，将 nntpd配置成独立运行的，那么务必将 inetd.conf中的每一行都标注为注释。但无论在哪种情况下，都必须在 /etc/services中设置下面这一行：

```
nntp 119 / tcp readnews untp # 网络新闻传输协议
```

为临时保存进入的文章等信息， nntpd还需要在你的新闻 spool中，设置一个 .tmp目录。创建方法如下：

```
# mkdir /var/spool/news/.tmp
# chown news.news /var/spool/news/.tmp
```

17.2 限制NNTP访问权限

对NNTP资源的访问要受 nntp_access文件的约束，该文件保存在 /usr/lib/news目录下。文件中的一系列行描述了授予外部主机的访问权限。每一行都必须采用下述格式：

site read | xfer | both | no post | no [!exceptgroups]

假如一个客户机连接到 NNTP端口，nntpd就会根据其IP地址，试着“逆向搜索”出它的完整形式的域名。根据每个条目的设置，按照它们在文件中出现的顺序，会依次检查客户机的主机名及IP地址是否相符。这种“相符”既可以是部分的，也可以是完全的。假如一个条目完全相符，便立即接纳它；假如仅部分相符，那么只有在以后再也找不出其他相符的条目时，才会接纳它。可用下述几种形式来指定一个具体的站点：

■ 主机名 —— 这是一个完整形式的主机域名。如果从字义上讲，它与客户机的规范主机名相符时，就应用这个条目，随后的所有条目都被忽略。

■ IP地址 —— 这是一个采用点分四段式表达式的 IP地址。如果客户机的 IP地址与此相符，就采用这个条目，随后的条目全部忽略。

■ 域名 —— 这是一个域名，被指定为 *.domain。如果客户机的主机名与该域名相符，就说明这个条目是匹配的。

■ 网络名 —— 这是/etc/network中指定的网络名。如果客户机 IP地址中的网络编号和与这个网络名关联的网络编号相符，就说明这个条目是匹配的。

■ 默认 —— 默认与任何客户机匹配。

带有更多常见站点规格的条目应该在此之前指定，因为任何与之相符的条目都被后来的更与之相符的条目改写。

第二和第三个字段描述了被授予客户机的访问权限。第二个字段详细指出，允许利用“推”（read）恢复新闻，利用“拉”（xfer）传输新闻。采用 both值，将启用“推”和“拉”，同时无访问拒绝。第三个字段授予客户机投递文章权，也就是说，投递其文章头信息不完整的文章，由新闻软件将其补充完整。如果第二个字段内有 no，第三个字段就会被忽略。

第四个字段是可选的，其中包含一个新闻组列表，中间用逗号隔开，客户机被拒绝对这些新闻组进行访问。

nntp_access文件示例如下：

```
#
# by default, anyone may transfer news, but not read or post
default                 xfer            no
#
# public.vbrew.com offers public access via modem, we allow
# them to read and post to any but the local.* groups
public.vbrew.com        read            post    !local
#
# all other hosts at the brewery may read and post
*.vbrew.com             read            post
```

17.3 NNTP身份验证

在用大写字母表示 nntp_access中的诸如xfer和read之类的访问特征时，针对不同的操作，nntpd会要求客户机验明自己的身份。例如，在指定 Xfer或XFER访问许可时，nntpd不会让客户机向你的站点传输文章，除非它通过了身份验证。

身份验证进程是通过一个新的 NNTP命令来实施的，该命令名为 AUTHINFO。使用该命

令时，客户机把用户名和密码传递给 NNTP服务器。nntpd把它们和/etc/passwd数据库内保存的进行核对，从而证实它们的身份并证实该用户属于 nntp组。

目前的NNTP身份验证实施尚处于实验阶段，所以其通用性不强。其结果只能适用于纯文本格式的密码数据库；还不能识别影子密码。

17.4 nntpd与C-News的沟通

在接收新闻组文章时， nntpd必须将其投递到新闻子系统。再根据该文章是否作为 IHAVE或POST命令结果被接收，分别将它上交给 rnews或inews命令。你可以不调用 rnews，而是在编辑时，把它配置为“对进入文章进行批处理”，并将所有批处理文件移到/var/spool/news/in.comi ng，这些批处理文件便等待relaynews在下一次运行时对它们进行处理。

为了能正确执行 ihave /sendme协议， nntpd必须能够访问历史文件。因此，你在编辑时，必须知道确定路径的设置是完全正确的。此外，还应该确定 C-News和nntpd商定了历史文件的格式。C-News利用dbm散列函数来访问它；但是， dbm库也有许多不同的，而且有些互不兼容的实施方案。如果C-News已经链接的dbm库和你的标准libc内的不同，你只好也用这个库来链接nntpd。

nntpd和C-News在数据库格式上的不一致，主要表现在 nntpd打不开系统日志内的错误消息，或通过 NNTP收到重复文件。最好的测验是从你的假脱机区内选出一篇文章， telnet到nntp端口，并将它提供给nntpd，如下所示。当然，你必须用自己准备发送给 nntpd的文章之ID来替换＜msg@id＞。

```
$ telnet localhost nntp
Trying 127.0.0.1...
Connected to localhost
Escape characters is '^]'.
201 vstout NNTP[auth] server version 1.5.11t (16 November
1991) ready at Sun Feb 6 16:02:32 1194 (no posting)
IHAVE <msg@id
435 Got it.
QUIT
```

这个对话说明了如何正确重建nntpd；消息“ Got it ”说明它已经有这篇文章了。如果你得到的是“ 335 OK ”这一消息，则说明出于某种原因，历史文件内的查找失败了。键入 Ctrl+D，就可中断这个对话。通过查看系统日志，可了解到出错的原因； nntpd把所有类型的消息统统记入系统日志后台程序。兼容性不好的 dbm库通常会出现在抱怨dbminit失败的消息内。

第18章　新闻阅读机的配置

新闻阅读机的目的是提供用户功能，允许他／她更轻松地访问新闻系统功能，比如以一种更为舒适的方式投递文章，或浏览新闻组内的内容等。这个界面的才能是无止境的。

目前，已经有两个新闻阅读机被移植到了 Linux。下面将描述三个常见新闻阅读机的基本设置，它们是：tin、trn和nn。

最有效的新闻阅读机之一是

```
$ find /var/spool/news -name '[0-9]*' -exec
cat {} \; | more
```

这是顽固分子阅读新闻的方式。

但是，大多数新闻阅读机都是相当复杂的。它们通常提供全屏界面，这个界面分为若干个单独的层。一层用于显示用户已经订阅的所有新闻组，一层用于显示一个新闻组内的所有文章概述，一层用于显示每篇单独的文章。

在新闻组层，许多新闻阅读机都显示出一个文章列表，展示它们的主题行和作者。在大型的新闻组内，对用户来说，要了解所有相关文章是不可能的，虽然对早期文章的答复进行鉴定是可能做到的。

答复通常重复原文章的主题，做法是在原文章前加一个“ Re:”。另外，对一个直接重复出现的文章来说，其消息 ID可能要在Reference:header这一行指定。按照这两点对文章进行分类，将生成较小的文章簇（事实上，是树），这些簇称为“主线”或“连载”（thread）。编写新闻阅读机的任务之一是设计一种更有效的连载方案。

这里，我们不打算深入讨论如何建立用户界面。目前， Linux系统上可用的所有新闻阅读机都有相关的帮助文档，供大家参考。

下面，我们只讨论一些管理任务。大部分与建立连载数据库和创建账户有关。

18.1　tin配置

与连载有关的、功能最齐全的新闻阅读机是 tin。它是Iain Lea编写的，利用一个名为 tass的老式新闻阅读机作为原型。后者是 Rich Skrenta编写的。用户进入新闻组之后， tin才开始执行连载，它的速度相当快，但通过 NNTP连接时，速度不太理想。

在一台486DX50机器上，直接从磁盘上读取时，它只花 30秒，就可连载1 000篇文章。但通过与一个加载新闻服务器的 NNTP连接时，则要花5分钟。

注意　如果由NNTP服务器自己执行连载处理，让客户机获取连载数据库，能够显著改进速度；比如INN-1.4版本，就是这样的。

也可以利用 -u选项，或随 -U选项一起，调用 tin，定期更新自己的索引文件，从而对此进行改进。

通常， tin把自己的连载数据库放在 .tin/index下面的用户根目录中。但是，这样可能会占

用系统资源，所以应该考虑把每个数据库的副本集中放在某个地方。具体作法是：令 tin setuid为news，或某个完全没有特权的账户（但是，千万不要将其设为 anybody。通常，无论什么命令和文件都不应该和这个用户有任何关联）。

然后，tin将所有的连载数据库保存在 /var/spool/news/.index下。对任何文件访问和脚本的换码，它都会把自己的有效uid重新设为调用它的用户之实际 uid（这便是你作为超级用户调用它时会出现不详错误消息的原因。之后，无论如何，你都不能以 root的身份执行任何操作了）。

最好的解决之道是安装 tind索引后台程序，它作为后台程序运行，并定期更新索引文件。但是，这个后台程序没有包含在 tin内，所以你必须自行编辑它。如果你运行的局域网带有一个中心新闻服务器，就可以在该服务器上运行 tind，让所有的客户机通过 NNTP，获取索引文件。当然，这需要对NNTP进行扩展。实施这一扩展的nntpd补丁程序包含在tin源代码内。

有些已发布的程序内包含的tin版本没有编入对NNTP的支持，但现在，大多数程序内都有了。在作为 rtin或随-r选项调用时， tin试着与 NNTP服务器建立连接，这个服务器是 /etc/nntpserver或NNTPSERVER环境参数中指定的。 nntpserver文件中用单独的一行来包含该服务器的名称。

18.2 trn配置

与早期的新闻阅读机（即rn，意为阅读新闻）相比， trn是个当然的赢家。其名字中的“ t ”代表“ 连载 ”（threaded）。它是Wayne Davidson编写的。

与tin不一样， trn不能在运行时生成其连载式数据库。相反地，它采用的是由一个名为 mthreads的程序提供的文件，该程序必须通过 cron定期调用，以便更新索引文件。

但是，不运行mthreads，并不意味着你不能访问新闻组文章；它只说明你的文章选择菜单中，将散布着所有“ Novell buys out Linux!! ”文章，而不是你可轻易跳过的单一主题。

要为特定新闻组打开连载播出，就要在命令行上随一个新闻组列表调用 mthreads。这个列表的结构和sys文件内的完全一样：

```
mthreads comp.rec.!rec.games.go
```

这样，将针对comp和rec内的所有文章进行连载处理， rec.games.go除外（玩Go的人通常都不需要新奇的主题）。之后，根本不用任何选项，调用它，就会令其将所有新近抵达的文章按照主题罗列在一起。 active文件内所有新闻组的连载处理也是可以打开的，具体做法是随一个all新闻组列表调用mthreads。

如果你是在夜间接收新闻，可自行定义在每天早上运行 mthreads，但如果需要，还可以更频繁地运行它。对通信量大的站点来说，它们可能希望在后台程序模式下运行 mthreads。利用-d选项，在系统启动时启用它时，它会把自己置入后台运行，每隔 10分钟，就出来检查是否有新文章到达，如果有，就对它们进行连载处理。要在后台程序模式下运行 mthreads，须把下面这一行放入你的rc.news脚本内：

```
/usr/local/bin/rn/mthreads -deav
```

-a选项令 mthreads在新闻组建立时，自动打开对这些新闻组的连载处理； -v选项为 mthreads文件启用冗长的记录消息，该文件名为 mt.log，位于你安装的trn目录中。

不再有用的旧文章必须定期从索引文件内删除。默认情况下，只有其编号在最低编号以

下的文章才会被删除。

注意 C-News不会自动更新这个最低编号；所以只有运行updatemin。详情参见第16章。

对已经被过期处理的、最低编号以上的文章（因为最旧的文章已经被 Expires:header字段分配了一个很长的过期日期）来说，也是可以删除的，具体做法是为 mthreads加上-e选项，实施一个“增强型”的过期运行。 mthreads在后台模式运行时， -e选项会令其在午夜之后，进入增强型过期运行，一天一次。

18.3 nn配置

nn是Kim F.Storm编写的，它被认为是一个新闻阅读机，其最终目的是不阅读新闻。其名字代表的是“没有新闻”（即no news），其座右铭是“没有新闻，便是好事。 nn更好。”

为了实现这一报负， nn为我们带来了名目繁多、功能齐全的维护工具，它们不仅允许你生成新闻主题，还扩展到检查主题数据库的一致性、清算和收集特性以及访问限制等。另外，还有一个管理程序，名为nnadmin，它允许你交互执行这些任务。这个程序非常直观易懂，所以我们不打算对它进行详细讨论，只说一下如何生成索引文件。

nn主题数据库管理程序称为 nnmaster。它通常作为后台程序运行，从 rc.news或rc.inet2脚本开始。它的调用形式如下：

```
/usr/local/lib/nn/nnmaster -1 -r -c
```

如此这般，就为active文件内的所有新闻组启用了主题归类处理。

同样地，也可周期性地从cron调用nnmaster，具体作法是为其指定一个新闻组列表，作为其调用依据。这个列表和 sys文件内的订阅列表极为类似，不同的是它采用的是空格，后者采用的则是逗号。它没有采用假的新闻组名 all，而是用一个空白参数来代表所有的新闻组。其示范调用如下：

```
# /usr/local/lib/nn/nnmaster !rec.games.go rec comp
```

注意，这里的顺序有特定的含义：最左面的组定义总是有效的。所以，如果我们把!rec.games.go放在rec之后，这个组内的所有文章都会被忽略。

nn提供了若干种方法，从其数据库内删除过期文章。其一是：查看新闻组目录，丢弃失效文章对应的条目，从而更新数据库。这是一个默认操作，是通过调用带有 -E选项的nnmaster来执行的。它的速度相当快，通过 NNTP进行此操作时除外。

第2种方法和默认的mthreads过期处理运行极为相似，它只删除其编号低于最低编号的文章之对应条目，最低编号在 active文件内。利用-e选项，可启用它。

最后是第3种方法，丢弃整个数据库，重新收集所有文章。具体做法是为 nnmaster指定-E3选项。

对即将过期的新闻组列表来说，它们是由 -F选项采用上面的方式指定的。但是，如果你把nnmaster作为后台程序运行，必须在它过期之前，利用 -k杀死它，随后再利用原来的选项重新启动它。所以在所有新闻组上运行过期处理的正确命令是：

```
## nnmaster -kf ""
# nnmaster -lrc
```

用于优化nn行为的标记还有许多。如果还想知道如何删除不好的文章或摘取文章提要，

可参考nnmaster手册。

nnmaster依赖于一个名为GROUPS的文件，这个文件位于/usr/local/lib/nn内。如果最初没有这个文件，就会创建一个。对于每个新闻组，它都包含一行，以新闻组名开头，后面选择性地跟一个时间戳和若干个标记。这些标记是可以编辑的，用于针对拿不准的新闻组启用特定行为，但你不能更改新闻组出现的顺序。(这是因为它们的顺序必须和二进制MASTER文件内的条目顺序一致。)允许设置的标记及其作用的有关详情也包含在nnmaster手册内。

第19章 其他问题

19.1 PLIP的空打印机电缆

为了将空打印机电缆用于 PLIP连接，需要两个25针的连接器（也就是 DB-25）和一根11导体电缆。这根电缆至少要有15米长。

查看连接器时，应该能看清楚每个针脚上的编号，从位于最左上部的针对应的 1到最右下部的针对应的25。对空打印机电缆来说，必须将它和连接器上的针一一对应，具体对应如下：

```
D0          2  -  15    ERROR
D1          3  -  13    SLCT
D2          4  -  12    PAPOUT
D3          5  -  10    ACK
D4          6  -  11    BUSY
GROUND     25  -  25    GROUND
ERROR      15  -   2    D0
SLCT       13  -   3    D1
PAPOUT     12  -   4    D2
ACK        10  -   5    D3
BUSY       11  -   6    D4
```

其余的针脚保持未连接状态。如果该电缆被塑料外壳包起来了，这个塑料外壳就应该只和DB-25规格的金属外壳的一端连接起来。

19.2 示范smail配置文件

本节向大家展示一个示例配置文件，它用于局域网上的一个 UUCP叶子站点。这些配置文件以smail-3.1.28源代码中包含的示例文件为基础。虽然我打算为大家讲讲这些文件的工作原理，但仍然建议大家参考 smail手册，其中详细讨论了这些配置文件的每一个细节。虽然你已经知道了smail配置文件的基本思路，但smail手册还是值得一读的，它相当简单，非常出色！

下面展示的第一个文件是routers文件，其中描述了smail的路由器集。当smail必须把邮件消息投递到指定的地址时，它会依次把这个地址交给所有的路由器，直到其中之一的地址与这个指定的地址相符。这里的相符指的是路由器在其数据库内找到了目标主机，其依据是paths文件、/etc/hosts或该路由器与之打交道的任何路由机制。

smail配置文件内的条目始终以一个唯一性的名字开头，该名字用以标识路由器、传输或执行者。条目之后为定义其行为的属性列表。该列表由一个全局属性集组成，比如采用的驱动程序和只有特定驱动程序才能识别的私用属性等。各属性之间用逗号隔开，而全局属性集和私用属性集之间分别用分号隔开。

为了更好地进行区别，我们以此为例：假设你打算维护两个各自独立的 pathalias文件；其中之一包含你所在域的路由信息，另一个文件内包含全局路由信息，可能是UUCP映射产生的。有了smail时，你可在routers文件内指定两个路由器，两者都采用 pathalias驱动程序。该驱动程序在pathalias数据库内查找主机名。它希望你能在一个私用属性内为它指定文件名。

前面的两个路由条目中，各自的第二个全局属性都定义了一旦找到与地址相符的路由器，应该采用的传输方式。我们的例子中，邮件消息将采用 uux传输形式来投递。传输形式是在transports文件内定义的，稍后我们将就此进行讨论。

要想具体规定一条消息由谁投递，可指定一个方法文件，而不是指定传输属性。方法文件可在目标主机名与传输之间建立一个对应关系。但这方面的详情，这里不打算深入讨论。

在混合了UUCP和TCP/IP的一个环境中，我们可能会碰到这样一个问题：对 /etc/hosts文件中指定的主机而言，可能只有极少数的情况，才会有人要求建立与它的 SLIP或PPP连接。通常，我们仍然想通过 UUCP发送给它们的任何邮件。为防止 inet_hosts在与这些主机相符的情况下投递，我们必须将它们置于 paths/force文件中。事实上，这又是一种“路径别名”（Pathalias）样式的数据库，它会在smail查询解析者之前，进行咨询。

对本地地址相关的邮件来说，对它们的控制是在 directors文件中进行的。它只由路由器文件构成，同时有一个条目列表，定义了每一个定向器。注意定向器本身并不负责一条消息的投递，它们只是执行所有可能的邮件重定向工作，比如通过别名重定向，通过邮件转发重定向……等等。

将邮件投递给一个本地地址比如 janet时，smail会将用户名依次传递给所有定向器。假如找到一个相符的定向器，它要么指定一个具体负责邮件投递的传送人（比如用户的邮箱文件），要么马上生成一个新地址（比如在匹配出一个别名之后）。

由于这里牵涉到的安全性问题，定向器通常要执行大量检查，判断它们使用的文件是否有害。凡是用可疑方式得到的地址（比如通过一个全局可写的别名文件），它们都会被作上标记，指出“不安全”。有些传送者干脆将此类地址屏蔽，比如将一条消息发给一个文件的传送者。

除此以外，smail也会将用户与每个地址对应起来。所有读写操作都会以用户的身份执行。例如，假定将一条消息投递到Janet的邮箱，那么地址理所当然需要与janet关联到一起。而对其他地址来说，比如自别名文件获得的那些地址，则让其他用户同它们关联，比如一个nobody用户。

欲了解这些特性的详情，可参考 smail手册。

注意 smail手册放在Linux文档项目联机手册的第8部分。

成功路由或定向了一条消息后， smail会将消息传递给由与地址相符的路由器或定向器指定的传送者。这些传送者定义在 transports文件中。同样地，一名传送者是由一系列全局和私用选项来定义的。

对每个条目定义的选项来说，其中最重要的便是具体负责传输的一个驱动者（ driver），比如管道驱动（pipe driver），它的任务是调用 cmd属性中指定的命令。除此以外，传送者还可使用大量全局属性，它们可对消息头进行方方面面的转换，同时还可对消息的主体进行处理。例如， return_path属性可令传送者在消息头（邮件头）中插入一个返回路径字段。unix_from_hack属性则令其位于以一个>符号开头的、每一个可能的“ From ”字样之前。在清单19-1中，我们列出了一个示范性的 /usr/lib/smail/transports文件。

清单19-1 一个示范性的/usr/lib/smail/transports文件

```
# A sample /usr/lib/smail/transports file

# local - deliver mail to local users
local:  driver=appendfile,        # append message to a file
```

```
        return_path,            # include a Return-Path: field
        from,                   # supply a From_ envelope line
        unix_from_hack,         # insert > before From in body
        local;                  # use local forms for delivery

        file=/var/spool/mail/${lc:user}, # location of mailbox files
        group=mail,             # group to own file for System V
        mode=0660,              # group mail can access
        suffix="\n",            # append an extra newline

# pipe - deliver mail to shell commands
pipe:   driver=pipe,            # pipe message to another program
        return_path,            # include a Return-Path: field
        from,                   # supply a From_ envelope line
        unix_from_hack,         # insert > before From in body
        local;                  # use local forms for delivery

        cmd="/bin/sh -c $user", # send address to the Bourne Shell
        parent_env,             # environment info from parent addr
        pipe_as_user,           # use user-id associated with address
        ignore_status,          # ignore a non-zero exit status
        ignore_write_errors,    # ignore write errors, i.e., broken pipe
        umask=0022,             # umask for child process
        -log_output,            # do not log stdout/stderr

# file - deliver mail to files
file:   driver=appendfile,
        return_path,            # include a Return-Path: field
        from,                   # supply a From_ envelope line
        unix_from_hack,         # insert > before From in body
        local;                  # use local forms for delivery
        file=$user,             # file is taken from address
        append_as_user,         # use user-id associated with address
        expand_user,            # expand ~ and $ within address
        suffix="\n",            # append an extra newline
        mode=0600,              # set permissions to 600

# uux - deliver to the rmail program on a remote UUCP site
uux:    driver=pipe,
        uucp,                   # use UUCP-style addressing forms
        from,                   # supply a From_ envelope line
        max_addrs=5,            # at most 5 addresses per invocation
        max_chars=200;          # at most 200 chars of addresses

        cmd="/usr/bin/uux - -r -a$sender -g$grade $host!rmail $(($user)$)",
        pipe_as_sender,         # have uucp logs contain caller
        log_output,             # save error output for bounce messages
#       defer_child_errors,     # retry if uux returns an error

# demand - deliver to a remote rmail program, polling immediately
demand: driver=pipe,
        uucp,                   # use UUCP-style addressing forms
        from,                   # supply a From_ envelope line
        max_addrs=5,            # at most 5 addresses per invocation
        max_chars=200;          # at most 200 chars of addresses

        cmd="/usr/bin/uux - -a$sender -g$grade $host!rmail $(($user)$)",
```

```
        pipe_as_sender,          # have uucp logs contain caller
        log_output,              # save error output for bounce messages
#       defer_child_errors,      # retry if uux returns an error

# hbsmtp - half-baked BSMTP. The output files must
#       be processed regularly and sent out via UUCP.
hbsmtp: driver=appendfile,
        inet,                    # use RFC 822-addressing
        hbsmtp,                  # batched SMTP w/o HELO and QUIT
        -max_addrs, -max_chars;  # no limit on number of addresses

        file="/var/spool/smail/hbsmtp/$host",
        user=root,               # file is owned by root
        mode=0600,               # only read-/writable by root.

# smtp - deliver using SMTP over TCP/IP
smtp:   driver=tcpsmtp,
        inet,
        -max_addrs, -max_chars;  # no limit on number of addresses
        short_timeout=5m,              # timeout for short operations
        long_timeout=2h,               # timeout for longer SMTP operations
        service=smtp,                  # connect to this service port
# For internet use: uncomment the below 4 lines
#       use_bind,                      # resolve MX and multiple A records
#       defnames,                      # use standard domain searching
#       defer_no_connect,              # try again if the nameserver is down
#       -local_mx_okay,                # fail an MX to the local host
```

第二部分　Linux系统管理员指南

Lars Wirzenius　著

作者简介

几乎从Linux诞生之日起，Lars Wirzenius就和它结下了不解之缘，可以说他是第二个在自己计算机上运行Linux系统的人。在觉得内核编程索然无趣之后，他把精力集中在高层次的问题上，比如主持comp.os.linux.announce新闻组。1992年，他和Michael K.Johnson以及Matt Welsh一起，合作成立了“ Linux文档项目 ”（简称LDP）。LDP着眼于为Linux用户、程序员和系统管理员制定完整而免费的文档集。这个项目已成功接近尾声。 Lars为LDP编写了《Linux系统管理员指南》。

Lars仍然执迷于LDP，同时也是Debian Project（Linux版本之一）开发人员。他的大部分时间都花在实现自己喜爱的项目上。如想进一步了解他的工作情况，可访问他的主页，其地址是http://www.iki.fi/liw。也可通过电子邮件联系他，他的邮件地址是 http://liw@iki.fi（遗憾的是，为保证自己的休息时间， Lars通常不回答常见的Linux问题。但他热忱欢迎大家提出本书以及和他的工作项目相关的问题和建议）。

本书简介

从哪里获得源代码和预先格式化好的版本

本书的源代码和其他机器可读的格式可通过匿名 FTP在因特网上找到， Linux文档项目的主页（地址是http://sunsite.unc.edu/LDP）和本书的主页（http://www.iki.fi/liw.linux/sag/）上都可找到本书。可以用的格式有 PostScript和TeX.DVI格式。

前　言

“开始写本书的时候，脑海中一片空白。当作者的手指在键盘上游移时，口中念着：该写点内容了，于是就有了内容”。

《Linux系统管理员指南》主要讲述利用Linux进行系统管理方面的知识。它是为对系统管理一无所知的读者编写的，不过读者应该具备起码的基本操作知识。这本指南没有介绍如何安装、设置Linux；因为这些知识可在相关的“安装和启动”文档内找到。看了下面的内容，你会对各种Linux手册有更多的了解。

系统管理就是一个用户要将计算机系统维持在一个正常工作状态所必需做的事情。它包含了复制文件（如需要的话可以恢复），安装新的程序，为用户建立帐户（不再需要的时候再取消），确保文件系统正常等等。如果把一台计算机比作一间房子，系统管理就好似对这间房子的保养，比如打扫清洁，修理坏了的窗户等等类似的事情。不过，系统管理不同于保养，因为那样就太简单了（有人一定要称谓它为保养，但那不过是因为他们还没看过这本手册）。

在本书的结构安排上，很多章节相互之间都是独立的，所以如你需要某方面的相关信息时，比如说备份方面的，你就可以有针对性地只看那一章节。这就很容易地将这本书当成工具手册来翻阅，需要哪部分就查看哪部分。尽管这样，这本书主要还是一本指导性手册，偶尔也可兼作工作参考手册。

本手册不打算对Linux系统的方方面面逐一讨论。对系统管理员来说，其他的Linux文档也同样不可忽视。毕竟，系统管理员是一个拥有特殊权限和职责的用户。手册页也是一个非常重要的资源，大家在用到一个不熟悉的指令时，就可以在上面查到相关信息。

本手册以Linux为讨论主题时，还遵循了这样的普遍准则，即本书要能同时适用于其他与Unix相关的操作系统。但不幸的是，一般说来，Unix不同版本之间的差异太多，尤其是系统管理方面。所以要想全盘考虑所有的版本根本不可能。而且，由于Linux自身的发展，全面讲述Linux的方方面面也不是件容易的事。

现在，由于Linux极富个性，还没有一个“正式”的Linux版本，所以不同的用户有不同的设置，并且很多用户都只采用自己的设置。本书不是以某一个版本为讨论对象的，虽然我本人一直专用Debian GNU/Linux系统。只要可能，我仍然会尽力指出另外几个版本间的区别，并对它们进行详细说明。

我打算详细说明Linux系统的工作原理，而不是罗列相关的操作任务。也就是说，本书包含的信息可能不是每个人都需要了解的，但大部分如此。.如果你采用的是一个预先配置好的系统，大可跳过某些章节。通过本书的讲解，你将自然而然地增进对Linux系统的了解，更轻松愉快地使用和管理它。

像其他所有与Linux相关的开发一样，这本书的写作也是出于自愿。我写这本书只是因为我觉得这件事比较有趣，也非常必需。尽管如此，就像所有的义务工作一样，我所付出的努力，运用的知识和经验也是有限的。这就意味着，这本手册并没有当初心目中那么好，假设我能够得到一笔丰厚的报酬，并花一定的时间去完善它，我想这本书会出色得多。

在有一点上，我取了巧；对于记载在其他适用免费手册上的内容，我在这本手册里就没再多写。特别是对与程序有关的文档，例如选用mks的所有细节，我仅讲述了这个程序的目的，也尽可能地讲它的用法。如要了解更深的知识，我建议读者看一下其他的资料。通常，所有和这些文档有关的东西都是整个Linux文档集的一部分。

尽管我曾尽力使本书完善，但我仍然想听听你们有什么关于完善这本书的宝贵意见。语言不得体，内容出错，有重复的内容或部分，还是要补充关于不同的 Unix变体版本的信息，我对这些建议都非常关心。通过万维网的 http://www.iki.fi/liw/mail to lasu.html和我联系。

我写本书的过程中，很多人曾帮过我，不论直接地还是间接地。我特别要感谢 Matt Welsh，他给我的鼓励和领导着 LDP的完善工作；还有 Andy Oram，为我提供有价值的反馈信息，令我不断更新作品； Olaf Kirch为我指出哪些内容是必须包括的，以及 Adam Richter和其他人，他们为我指出大家都比较感兴趣的热点问题。

感谢Stephen.Tweedie、H.Peter Anvin、R ’emy Card和Theodore Ts ’o，允许我借用他们的作品（因此使这本书看起来比较厚，也更容易让人留下深刻印象），它们是：xva和ext2文件系统之间的对比、 ext2文件系统的描述和设备列表。我对这点非常感激，非常遗憾早期的版本中欠缺这些东西。

除此之外，我还要感谢 Mark Komarinski，他在1993年，将自己的资料以及许多系统管理栏目放入Linux Journal中。这些内容令人大开眼界，也很富有启发性。

另外，还有许多人为我提供了许多有用信息，已知的人员有（按字母表顺序）：Paul Caprioli、Ales Cepek、Marie-France Declerfayt、Dave Dobson、Olaf Flebbe、Helmut Geyer、Larry Greenfield及其父亲、 Stephen Harris、Jyrki Havia、Jim Haynes、York Lam、Timothy Andrew Lister、Jim Lynch、Michael J.Micek、Jacob Navia、Dan Poirier、Daniel Quinlan、Jouni K Seppanen、Philippe Steindl、G.B.Stotte。如有遗漏，我很感报歉。

Linux文档项目

Linux文档项目，即LDP，是一个由作者、校对人员和编辑组成的松散组织。我们共同为Linux操作系统提供完整的文档。该项目的领头人是 Greg Hankins。

本手册只是LDP出版的系列丛书之一，这一系列中包括《 Linux用户指南》、《系统管理员指南》、《网络管理员指南》以及《内核黑客指南》。这些手册以源码格式、 .dvi格式和PostScript输出的格式提供，可通过匿名 FTP，从sunsite.unc.edu的/pub/Linux/docs/LDP目录中下载。

我们鼓励任何有写作和编辑爱好的人加入我们的行列，改进 Linux文档。如果你能上网的话，你可通过gregh@sunsite.unc.edu，和Greg Hankis取得联系。

第1章　Linux系统综述

本章是对Linux系统的概述。首先为大家讲述该操作系统提供的主要服务。然后粗略谈谈实现这些服务的程序。本章的目的是使大家对该系统有一个全面了解，具体的各个部分将在以后章节里讲述。

1.1　操作系统的各个组件

Unix操作系统由一个内核和一些系统程序组成。其中也有执行特定工作的应用程序。内核是操作系统的核心（实际上，它通常被误认为是操作系统本身，但事实并非如此。操作系统提供的服务比内核提供的服务要多得多）。它能维护磁盘磁道中的文件、启动并同时运行多个程序、将存储空间和其他资源分配给不同程序，在网上收发数据包等。内核自身所做的工作少之又少，但它能提供建立所有服务程序的工具。它还能阻止任何用户直接访问硬盘，迫使每个用户都使用它提供的工具。通过这种方法，内核为用户相互间提供了一种保障。内核提供的工具是通过系统调用来使用的；关于这方面的详情，可参考手册的第二部分。

系统程序利用内核所提供的工具执行操作系统要求的各种服务程序。系统程序和其他所有的程序一起，以“用户模式”运行于内核顶部。系统程序和应用程序之间的区别在于其目的不同：应用程序用来做一些有用的、实际的事（或是娱乐，假如它正好是游戏的话），而另一方面，系统程序则是用来维护系统工作的。例如，字处理程序是一个应用程序； Telnet是一个系统程序。通常，系统程序和应用程序之间的界限有些模糊，虽然如此，这种区别对那些热衷于归类的人来说，仍然是非常重要的。

操作系统中，还包括编译程序和与它们对应的库（尤其是 Linux下的GCC和C语言库），虽然并非所有的编程语言都必需成为操作系统中的一部分。文档，有时甚至于游戏都可成为操作系统的一部分。过去，操作系统一直由安装盘或安装磁带来定义，但 Linux则不一样，它相当个性化，任何人只要有兴趣，都可在全球各 FTP站点下载并制定自己的操作系统。

1.2　内核的重要组件

Linux内核由几个重要部件组成：进程管理、内存管理、硬件设备驱动程序、文件系统驱动程序、网络管理和其他零碎的东西。图 1-1展示了部分组件。

内核部件中，最重要（没有它，什么也干不了的）的可能是内存管理和进程管理。内存管理照管已分配给进程、内核部件和缓冲区的内存区和交换空间。进程管理则创建进程，并通过在处理器上交换活动进程的方式，实施多任务操作。

在最低级上，针对每个自己支持的硬件设备，内核中都包含相应的驱动程序。由于各种硬件设备名目繁多，所以对应的驱动程序也多如牛毛。有些硬件设备的行为会因为驱动程序的不同而不同。不过，按其类似之处可以对支持类似操作的设备进行归类；同类的设备采用同样的方式与内核中的其他部件沟通，但实施方式不尽相同。例如，所有的磁盘驱动程序看起来和内核中的其他部件差不多，也就是说，它们都有类似于“初始化驱动器”、“读取扇区N”

和“写入扇区N”之类的操作。

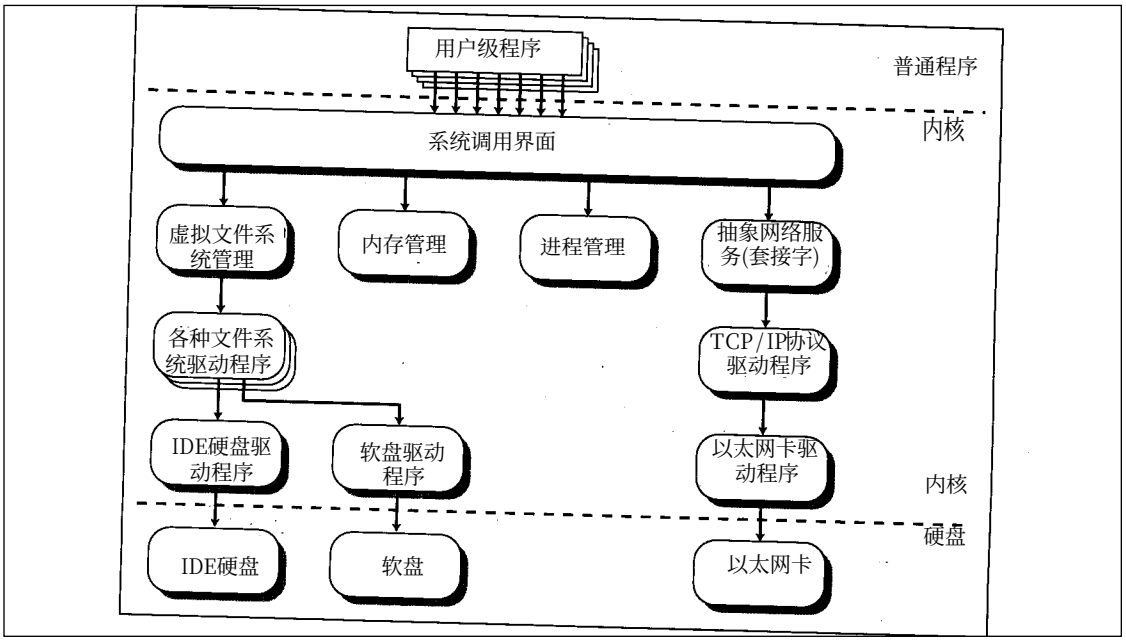

图1-1 Linux内核的几个重要部件

内核本身提供的某些软件服务也有类似的属性，因此，也可将具有类似属性的服务归入一类。例如，各种网络协议已被归入一个编程接口——BSD套接字库。另一个例子是虚拟文件系统（VFS）层，它把文件系统操作从其实施过程中提了出来。每个文件系统类型都提供各个文件系统操作的实施。在有些实体打算使用文件系统时，请求就会到达 VFS，然后VFS再把请求路由到恰当的文件系统驱动程序。

1.3 Unix系统提供的主要服务

本小节将概括解释一些重要的Unix服务，其详情参见后续章节。

1.3.1 init

Unix系统中，最重要的服务是由 init提供的。init是每个Unix系统中第一个启动的进程，也是内核在启动时所进行的最后一件事情。 init启动时，它会处理各种启动“杂活儿”（检查和装入文件系统、启动后台程序等），继续执行启动进程。

init具体做的事和个人目的有关。init通常提供单用户模式，这种模式下，没有人能够登录，只有root才能在控制台使用外壳；普通模式称为多用户模式。有的人把它统称为运行级别；单用户和多用户模式都被认为运行级别是 2，还有别的运行级别，比如X运行级别。

普通操作中，init确定getty正在运行（允许用户登录）并接收“孤儿”进程（其父进程已经死了的进程；在 Unix系统中，所有进程都必须在一个单独的树中，所以 init必须接收孤儿进程）。

系统关闭时，init就负责杀死其他所有进程、卸载所有的文件系统、中止处理期以及处理

已配置好的各种杂事。

1.3.2 从终端登录

从终端（通过串行线路）和控制台（在没有运行 X时）登录是由getty程序提供的。init为允许登录的每个终端提供一个独立的 getty实例。getty读取登录用户名，并运行 login（登录）程序，该程序便开始读取密码。如果用户名和密码都正确，login程序就开始运行shell（外壳）。由于用户名和密码不匹配而引起 shell中断时（比如，用户注销或登录中断等），init会注意到这一点，并发起一个新的 getty实例。内核不会注意到这一点，这一切都是由系统程序来控制的。

1.3.3 syslog

内核和许多系统程序都会发出错误、警告和其他消息。这些消息可稍后进行查看，所以，把它们写入一个文件是非常重要的。这就是系统日志。可以根据消息的重要性或写入者，对消息进行分类。比如，内核消息通常被直接写入一个不同于其他消息文件的文件内，因为内核消息通常更为重要，需要定期读取以探测问题的起因。

1.3.4 周期性执行的命令：cron和at

对用户和系统管理员来说，通常都需要周期性地运行某些命令。例如，系统管理员可能想运行一个命令，从老文件中清空带有临时文件（ /tmp和/var/tmp）的目录，以防止磁盘填满数据，因为并非所有的程序都能正确善后。

cron服务就是为此而设计的。每个用户都有一个 crontab文件，他们在该文件内列出自己想执行的命令及其命令执行次数。 cron后台程序负责在指定时间运行指定的命令。 at服务和cron服务类似，但它只执行一次：这个命令是在指定时间执行的，而不是重复执行的。

1.3.5 图形化用户接口

Unix和Linux都没有将用户接口结合在内核内；而是令其由用户级程序来实施。这种方式适用于文本模式和图形化环境。

这样的安排令系统更为灵活，但有个缺点：为每个程序实施不一样的用户接口令系统更难以掌握。

Linux系统主要使用的图形化环境叫作“ X Window System ”（简称X）。X也没有实施用户接口；它只实施一个窗口系统，也就是能够实施图形化用户接口的若干个工具。X窗口系统上，最常见的三类用户接口是 Athena、Motif和Open Look（最近还有KDE和Gnome，两者都是建于X之上的用户环境）。

1.3.6 连网

“ 连网 ”是指把两台或两台以上的计算机连接在一起，以便它们可以进行沟通。实际应用中的连网和沟通稍微有些复杂，但非常有用。

Unix操作系统提供了许多连网功能。大多数基本服务（文件系统、打印、备份等）都可

在网络中进行。这样一来，系统管理就变得更为简单，因为它允许集中管理，而且还可享受到微型计算处理和分布式计算的好处，比如降低成本和容错性更佳等。

但是，此处只是蜻蜓点水般地提一下连网；有关详情，还请参考本书第一部分。

1.3.7 网络登录

网络登录和普通登录稍有不同。每个终端都有一个独立的物理串行线路，通过这条线路，用户就可以登录了。对每一个通过网络登录的用户来说，有一条独立的虚拟网络连接，而且连接的多少是任意的（至少可以说有许多。网络带宽仍然是很宝贵的，所以通过一条网络连接的并发登录数实际上是有上限的）。所以，不能为每一条可能的虚拟连接运行一个独立的getty程序。通过网络登录还有几种方式，比如，正在成为 TCP/IP网络主流的 telnet和rlogin。

实现网络登录时，取代运行一大堆 getty程序的是一个独立的后台程序（ telnet和rlogin都有独立的后台程序），该后台程序负责监听所有的进入登录请求。一旦它注意到有请求进入，就会启动自己的一个新实例，处理这个登录请求；原来的实例仍然继续监听其他的登录请求。新实例的作用和getty类似。

1.3.8 网络文件系统

采用连网服务的一大优点是能够通过网络文件系统共享文件。最常用的网络文件系统称为“ 网络文件系统 ”（简称NFS），是Sun公司开发的。

有了网络文件系统，一台计算机上某个程序执行的任何文件操作通过网络，被发送到另一台机器上。这样可欺骗程序，使其认为另一台机器上的所有文件实际上在自己正在运行的机器上。这样用不着修改程序，就能方便地实现信息的共享。

1.3.9 邮件

对计算机之间的交流来说，电子邮件通常是最重要的方式。电子邮件保存在一个采用特殊格式的文件内，而特殊的邮件程序用于发送和阅读电子邮件。

每个用户都有一个接收邮件的信箱（一个采用特殊格式的文件），所有的新邮件都保存在这个信箱内。有人在发送邮件时，邮件程序就找出收件人的信箱，并把这封邮件添加到信箱文件内。如果收件人的信箱在另一台机器上，这封邮件就会被发送到另一台机器上，再由它采用最佳路由，把邮件转发到收件人的信箱。

邮件系统由许多程序组成。把邮件投递到本地或远程邮箱是通过一个程序来实现的，这个程序就是邮件传输代理或 MTA，例如sendmail和smail。而用户使用的程序（邮件用户代理或MUA，例如pine和elm）也是举不胜举的。信箱通常保存在 /var/spool/mail内。

1.3.10 打印

虽然一次只允许一个用户使用打印机，但用户间如果不能共享打印机的话，就会多花许多钱。因此，打印机就交给软件来管理，这类软件用于实施打印队列：所有的打印作业都进入队列等候打印，打印机一次只能处理一个作业，完成之后，下一个作业自动跟上。这样便减轻了用户重新组织及管理打印作业的负担。

打印队列软件还将打印输出假脱机在磁盘上，也就是说，打印作业在排队等候期间，打印文本是保存在一个文件内的。这样便允许一个应用程序迅速向打印队列软件供给打印作业。这的确非常方便，因为它允许用户在得到新的完全修订本之前，打印一份样本，而不是被动地等待打印。

1.3.11 文件系统布局

文件系统分为许多部分； root文件系统通常有 /bin、/lib、/etc、/dev以及其他几个部分；/usr文件系统内包含的是程序和一些没有发生变化的数据； /var文件系统内包含的是正在发生变化的数据（比如日志文件）；而 /home文件系统则保存每个用户的私人文件。根据硬件配置和系统管理员的决定，文件系统的布局可以不一样；甚至还可把所有的数据统统装入一个文件系统内。

第3章将详细讨论文件系统的布局；这方面的详情，还可参考 Linux文件系统标准。

第2章　目录树简介

本章将讨论标准Linux目录树的重要部件，它们是基于FSSTND文件系统标准的（“文件系统分层结构标准”[fhs]http://www.pathname.com/fhs/的延续。另外，还可参见“Linux标准基础知识”。其地址是http://www.linuxbase.org）。该标准描述了划分目录树的普通方式并指出这种划分方式的动机。出于不同的目的，我们通常需要把目录树分成若干个独立文件系统。此外，还将描述划分目录树的几种方式。

2.1　背景知识

本章没有严格参照Linux文件系统标准——FSSTND 1.2版（参见参考书目），这个版本尽力为如何组合Linux系统内的目录树建立一个标准。这样的标准有它的优点，即便于为Linux系统编写或移植软件，便于管理配置Linux的机器，因为Linux需要的一切东西都将放在“固定”的某个地方。这个标准的背后没有机构要求每个人都必须遵从它，但它正越来越多地得到许多Linux版本的认可。没有特别充足的理由，就拒绝采纳FSSTND不是件好事情。FSSTND尽力遵循Unix的习惯和当前的发展趋势，使Linux系统更加接近其他的Unix系统，Unix系统也更加接近Linux系统。

本章的重点不是讨论FSSTND。为了对其有全面的认识，系统管理员还应该好好看看FSSTND。

本章也不详细讨论全部文件。其目的不是在于细细剖析每个文件，而是从文件系统的角度出发，总览整个系统。有关各个文件的详情，可参考本书其他章节或手册页。

完整的目录树将被分为若干部分，各部分在其自己的磁盘或分区内，更适应于磁盘空间，更易于进行备份和其他系统管理。目录树中，主要的几个部分是root、/usr、/var和/home（如图2-1所示）。各部分用途不同。经过对目录树的设计，它已经能够正常运行于Linux机器组成的网络环境中，这些机器通过只读设备（比如光盘）或采用NFS的网络共享某些文件系统。

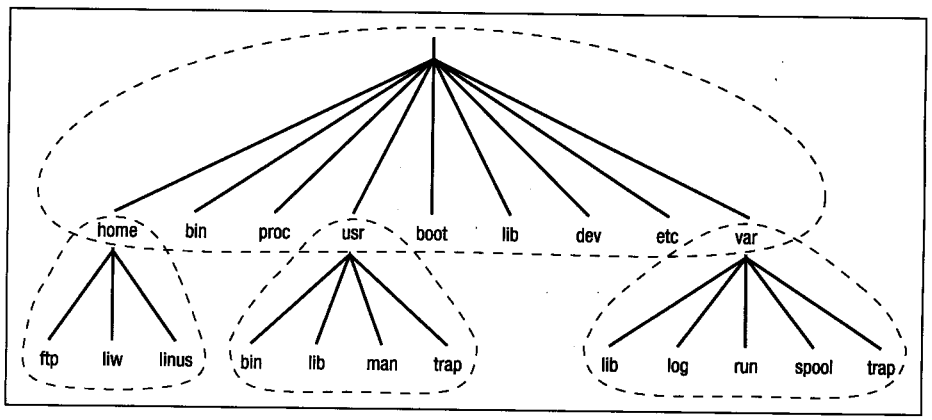

图2-1　Unix目录树中的各部分。虚线表示分区限制

目录树中，各部分扮演的角色是不同的。具体描述如下：

■ root文件系统是每台机器特有的（它一般保存在本地磁盘上，但也可保存在RAM磁盘或网络驱动器上），其中包含启动系统和引导装入文件系统所需的文件。因此，对单用户模式来说，root文件系统的内容足之够矣。另外，该文件系统内还包含一些工具，用于修复损坏的系统和从备份中恢复丢失的文件。

■ /usr文件系统内包含所有的命令、库、手册页和其他没有变动过的文件（这些文件是普通操作期间所需要的）。/usr文件系统内没有专门针对哪台机器的文件，也没有普通操作期间应该修改的文件。这样一来，便可以通过网络实现文件共享，从而有效地节约成本，因为这样可节省磁盘空间（要知道/usr文件系统起码也有几百个MB），使管理工作更容易（在更新应用程序时，只有master/usr需要改动，而不是逐一在每台机器上改动）。即使文件系统是在本地磁盘上，也可以采用只读方式装入它，减小系统崩溃时文件系统受损的可能性。

■ /var文件系统中包含有变动的文件，比如假脱机目录（用于邮件、新闻、打印机等）、日志文件、格式化的手册页和临时文件。 /var内原来的所有东西都已经转移到/usr下，但其结果是不能采取只读装入/usr。

■ /home文件系统中包含用户的根目录，也就是系统上的所有真实数据。把根目录和用户的其他目录树或文件系统区分开的目的是为了方便备份；其他部分通常是不需要备份的，或者说至少不需要经常备份（因为它们几乎没有变动）。一个大型的/home文件系统可能必须分为若干个小型的文件系统，这就需要在/home下面加一个额外的命名级别，比如/home/students和/home/staff。

虽然前面的各部分被称为文件系统，但不要求它们真的在一个独立的文件系统上。如果系统是一个小型的单用户系统，而且用户想一切简单明了的话，以上各部分统统可合成一体。根据磁盘空间的大小和为不同用途分配空间的多少，目录树也可按照不同的方式分成若干个文件系统。可是，目录树中的重要部分就是使所有标准目录名正常运作的部分；也就是说，即使/var和/usr真的同在一个分区，/usr/lib/libc.a和/var/log/message这两个目录名也必须能用，具体做法是把/var下面的文件移入/usr/var，再令/var成为/usr/var的符号链接。

Unix文件系统结构根据文件的用途，把它们分为各个组，也就是所有的命令归入一个地方，所有的数据文件归入另一个地方，文档又另行存放等等。另外，就是根据文件所属的程序来分组，也就是所有的Emacs文件归入一个目录，所有的TeX文件归入另一个目录等等。后一种分组方法的不足就是很难实现文件的共享（程序目录中，通常既有静态和可共享的文件，又有动态和不能共享的文件），而且有时还不方便文件的查找（比如，手册页可能散布在许多个地方，对用户或管理员来说，不得不找出自己需要的手册页，无疑是场恶梦）。

2.2 root文件系统

一般说来，root文件系统应该较小，因为其中包含非常关键的文件和一个小型的、非频繁变动的文件系统。受损的root文件系统一般意味着系统不能启动，除非借助于特殊的启动设备（比如说软盘），所以一般不要轻易更改它。

root目录一般不包含任何文件，系统的标准启动镜像除外，这个镜像通常称为/vmlinuz。其他的所有文件都保存在root文件系统的子目录下：

1. /bin

启动期间，可供普通用户使用的命令（也可能在启动之后）。

2. /sbin

和/bin一样，但不是供普通用户使用的，虽然在必要或经过允许的情况下，普通用户也可使用它们。

3. /etc

某台机器专用的配置文件。

4. /root

用户\texttt{root}的根目录。

5. /lib

root文件系统上的程序所需的共享库。

6. /lib/modules

可装载的内核模块，特别是从灾难中恢复时，启动系统所需的那些模块（比如，网络和文件系统驱动程序）。

7. /dev

设备文件。

8. /tmp

临时文件。启动后的程序运行应该采用 /var/tmp，而不是/tmp，因为前者可能在空间较大的磁盘上。

9. boot

启动装载程序所用文件，比如 LILO。内核镜像通常保存在这里，而不是 root目录中。如果有多个内核镜像，这个目录就可能增长得很快，所以最好把它单独保存在一个文件系统内。这样做的另一个原因是确保内核镜像在 IDE磁盘的前1024个磁道内。

10. /mnt

系统管理员临时装入的装入点。程序不会自行装入 /mnt。/mnt也可以分为若干个子目录（比如/mnt/dosa可能是使用MS-DOS文件系统的软驱，而/mnt/extra则可能和ext2文件系统如出一辙）。

11. /proc、/usr、/vsr和/home

其他文件系统的装入点。

2.3 /etc文件系统

/etc目录中包含许多文件。下文将讨论其中的一部分。另外的文件，则应该由你决定它们属于哪个程序，并参考那个程序的手册页。许多连网配置文件也被包含在 /etc内，详情参见本书第一部分。

1. /etc/rc或/etc/rc.d或/etc/rc?.d

启动时或运行级别发生变化时运行的脚本或脚本的目录。有关详情，参考 init一章。

2. /etc/passwd

用户数据库，其中有一些字段指定用户名、用户真名、根目录、加密密码以及该用户的其他信息。其格式已编入\man{passwd}手册页。

3. /etc/fdprm

软盘参数表。描述各种软盘的不同格式。供setfdprm使用。更多详情参考setfdprm手册页。

4. /etc/fstab

列出启动时由mount -a命令（在/etc/rc或等同的启动文件内）自动装入的文件系统。 Linux系统中，这个文件还包含一些信息，这些信息和 swapon -a自动采用的交换区有关。更多详情参考第3章中的“ 装入和卸载 ”小节和mount手册页。

5. /etc/group

类似于/etc/passwd，但它描述的不是用户，而是组。更多详情，请参考 group手册页。

6. /etc/inittab

init配置文件。

7. /etc/issue

登录提示出现之前的 getty输出。通常包含对系统的简短说明或欢迎消息。其内容由系统管理员决定。

8. /etc/magic

文件的配置文件。其中包含不同文件格式的说明，以便根据格式猜测出文件的类型。更多详情，请参考magic和file手册页。

9. /etc/motd

日期消息，是在成功登录之后自动输出的。其内容由系统管理员决定。通常用来提示每个用户，比如既定的系统关闭警告等。

10. /etc/mtab

列出当前已装入的文件系统。最初是由启动脚本设置，由 mount命令自动更新的。用于需要已装入文件系统列表时（比如说在运行 df命令时）。

11. /etc/shadow

在已安装影子密码软件的系统上的影子密码文件。影子密码把已加密的密码从 /etc/passwd移入/etc/shadow；后者只有root才能读取。这样可进一步保证密码的安全性。

12. /etc/login.defs

login命令的配置文件。

13. /etc/printcap

类似于/etc/termcap，但对象是打印机。而且采用的语法也不同。

14. /etc/profile、/etc/csh.login和/etc/cshrc

登录或启动时，由Bourne或C外壳执行的文件。这些文件允许系统管理员为所有的用户设置全局默认设置。各外壳的详情，请参考手册页。

15. /etc/securetty

标识安全终端，也就是允许root通过哪些终端登录。一般说来，只列出了虚拟控制台，如此一来，恶意用户不可能通过 modem或网络攻击系统，从而获得超级用户特权（至少说很难）。

16. /etc/shells

列出受托（信得过的）外壳。 chsh命令允许用户把他们自己的登录外壳改成这个文件内列出的受托外壳。为计算机提供 FTP服务的 .ftpd服务器进程，将复查用户的外壳是否在/etc/shells内，如果在，将允许用户登录，如果不在，就不会让用户登录。

17. /etc/termcap

终端性能数据库。描述按照什么样的“ 转义序列 ”来控制不同的终端。编写程序时，不

是直接输出转义序列（只运行于特定品牌的终端），而是查找正确的序列，以执行自己打算在/etc/termcap内执行的操作。这样做的结果是，多数程序都可利用各种各样的终端。更多详情，请参考termcap、curs_termcap和terminfo手册页。

2.4 /dev文件系统

/dev目录下包含所有设备的特定设备文件。设备文件的命名有特殊的约定；对这些约定的描述包括在Device列表中。设备文件是在安装期间或后期，利用 /dev/MAKEDEV脚本创建的。/dev/MAKEDEV.local是由系统管理员编写的一个脚本，它创建只用于本地的设备文件或链接（也就是那些不属于标准 MAKEDEV的设备文件，比如用于某些非标准设备驱动程序的设备文件）。

2.5 /usr文件系统

/usr文件系统通常较大，因为所有的程序都是保存在这个文件系统中的。 /usr内的文件通常来自Linux系统；本地安装的程序和其他东西都保存在 /usr.local下面。这样一来，就能够通过该文件系统的新版本或全新版本来升级系统，根本不需要再次安装所有的程序。下面列出了部分/usr子目录（有些不重要的目录已被删除；更多详情请参考 FSSTND）。

1. /usr/X11R6

X Window System包含所有的文件。为了简化 X的开发和安装， X文件尚未和系统的其他部分集成。/usr/X11R6下面有一个目录树，它类似于 /usr本身的目录树。

2. /usr/X386

类似于/usr/X11R6，但针对的是X11 5版本。

3. /usr/bin

几乎包含所有的用户命令。有些命令在 /bin或/usr/local/bin内。

4. /usr/sbin

root文件系统上不需要的系统管理命令，例如，大多数服务器程序。

5. /usr/man、/usr/info和/usr/doc

分别包含手册页、 GNU信息文档和名目繁多的其他文档文件。

6. /usr/include

C编程语言的头文件。实际上，为了保持数据的一致，这个文件应该保存在 /usr/lib下面，但过去一直都采用这个名称。

7. /usr/lib

程序和子系统所用的未变动过的数据文件，其中包括一些和站点有关的配置文件。 lib这个名称源于库（library）；最初用来编写子例程的库都保存在 /usr/lib内。

8. /usr/local

本地安装的软件和其他文件的地方。

2.6 /var文件系统

/var文件系统内包含系统正常运行时所改动的数据。它是各个系统专有的，也就是说，不能通过网络和其他计算机共享。

1. /var/catman

按需格式化的手册页之缓冲区。手册页的原件通常保存在 /usr/man/man*内；有的手册页还可能有一个预先格式化版本，这个版本保存在 /usr/man/cat*内。至于其他手册页，则需要在初次查看时，进行格式化；格式化过后的版本保存在 /var/man内，以便下一个查看该手册页的用户无须等待其格式化（/var/catman的清空方式和临时目录的清空方式一样）。

2. /var/lib

系统正常运行期间发生变化的文件。

3. /var/local

针对安装在/usr/local内的程序的变化数据（也就是已经由系统管理员安装的程序）。注意，即使是本地安装的程序，如果适当的话，也应该使用其他的 /var目录（比如/var/lock）。

4. /var/lock

锁文件。许多程序都习惯在 /var/lock内建立一个锁文件，借以表明它们正在使用某个特定的设备或文件。其他程序将注意到这个锁文件，并不再尝试使用这个特定的设备或文件。

5. /var/log

日志文件，它源于各个程序，特别是 login（/var/log/wtmp，记录所有的系统登录和注销活动）和syslog（/var/log/messages，通常保存所有的内核和系统程序消息）。/var/log内的文件通常增长较快，需要定期清空。

6. /var/run

文件，其中包含系统相关信息，在系统下一次启动之前，都是有效的。例如，/var/run/utmp内包含和当前登录用户有关的信息。

7. /var/spool

用于邮件、新闻、打印机队列和其他队列作业的目录。对每个不同的假脱机来说，在/var/spool下面都有其自己的子目录，比如用户信箱就在 /var/spool/mail内。

8. /var/tmp

临时文件，通常较大，或需要保存的时间比 /tmp长的文件（虽然系统管理员可能不允许/var/tmp内也保存一些非常老的文件）。

2.7 /proc文件系统

/proc文件系统内包含一个有误的文件系统。它不存在于磁盘上。相反地，内核是在内存中创建它的。它用于提供和系统相关的信息（最初是进程相关信息，并由此得名）。下面将对有些比较重要的文件和目录进行解释。 /proc文件系统的更多详情，请参考 proc手册页。

1. /proc/1

目录，其中有1号进程的相关信息。每个进程在 /proc下面都有一个子目录，这个子目录名就是该进程的编号。

2. /proc/cpuinfo

其中保存关于中央处理器的信息，比如型号、制造商、模型和性能等。

3. /proc/devices

其中列出了已经配置到当前正在运行的内核之中的设备驱动程序。

4. /proc/dma

展示当前正在使用的DMA通道。

5. /proc/filesystems

已配置到内核中的文件系统。

6. /proc/interrupts

展示哪些中断号正在使用中，以及各中断号使用了多少次。

7. /proc/ioports

展示此时哪些I/O端口正在使用中。

8. /proc/kcore

系统物理内存的镜像。其大小完全和你的物理内存一样，但事实上占不了多少内存；它是在程序访问它时，即时生成的（记住，除非你把它复制到别的地方，否则， /proc根本就不占用任何磁盘空间）。

9. /proc/kmsg

内核输出的消息。同时，也被路由到 syslog。

10. /proc/ksyms

内核的符号表。

11. /proc/loadavg

系统的“ 装载平衡 ”；无意义的三个识别符，表示此时系统应该做多少操作。

12. /proc/meminfo

包含和内存使用相关的信息，其中既包括物理内存，又有交换空间。

13. /proc/modules

表明此时正在装载哪些内核模块。

14. /proc/net

和网络协议相关的状态信息。

15. /proc/self

指向一个程序进程目录的符号链接，这个程序此刻正在查看 /proc。如果有两个程序都在查看/proc，它们就会得到两个不同的符号链接。这主要是为了方便程序更容易得到自己的进程目录。

16. /proc/stat

关于系统的各种统计数据，比如自系统启动以来出现的页故障次数统计。

17. /proc/uptime

表明系统已启用多久。

18. /proc/version

内核版本号。

注意，上面的文件越来越发展成为易于理解的文本化文件，但有时，它们采用的格式却是难以理解的。所以，目前有许多命令将上面的文件转换为更便于理解的格式。比如，有个免费程序读取/proc/meminfo，并将指定的字节转换为千字节（同时，还增加了少许信息）。

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。

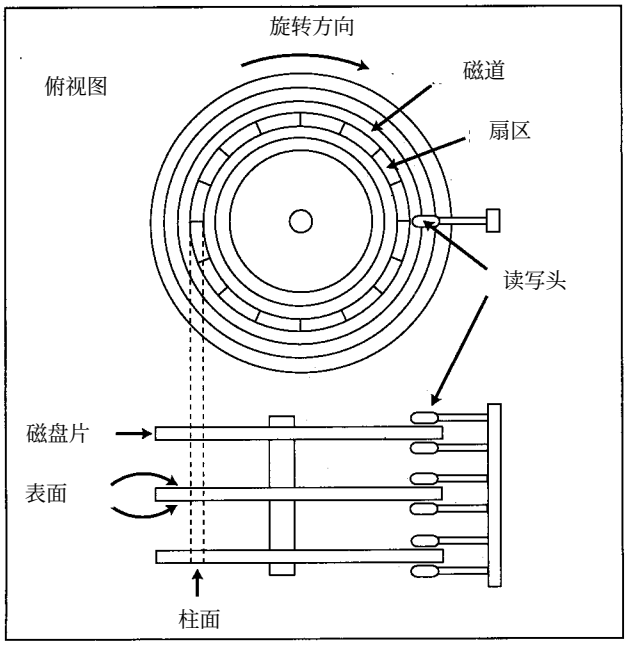

图3-1 硬盘图解

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。

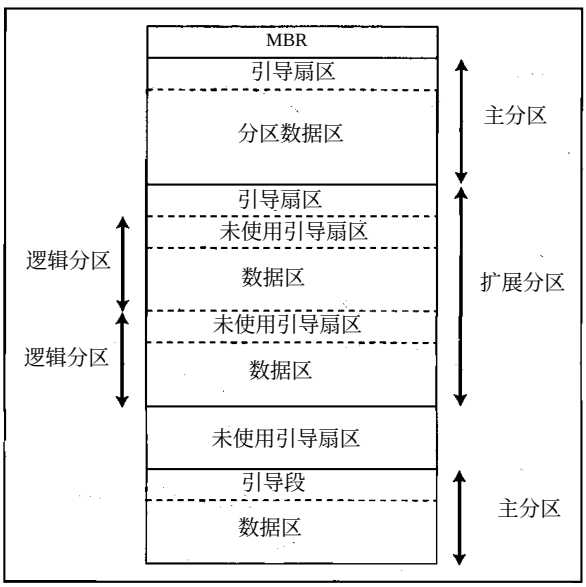

图3-2 硬盘分区示例

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。

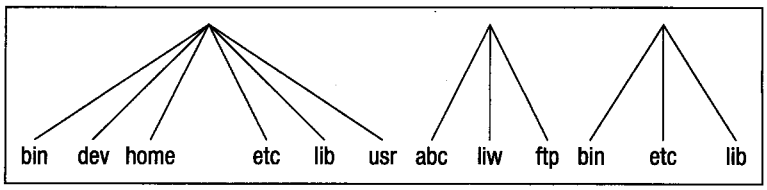

图3-3 三个独立的文件系统

可像下面的示例一样，装入文件系统：

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

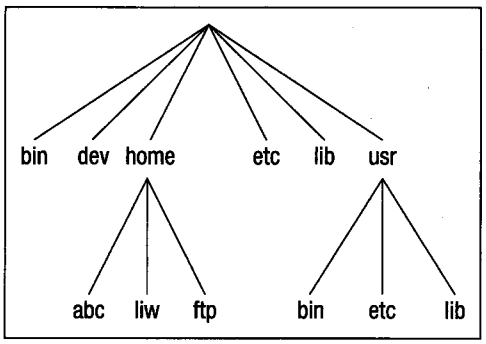

图3-4 已经装入的/home和/usr

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）。

第4章 内存管理

这部分主要讨论Linux的内存管理特性，也就是虚拟内存和缓冲区。重点讨论系统管理员需要考虑的内存管理的目的、工作原理和具体操作。

4.1 何谓虚拟内存

Linux支持虚拟内存，也就是说，把磁盘当作扩充RAM使用，进而有效增大可用内存空间。内核将把目前未用的内存块中的内容写入磁盘，以便把内存用作其他用途。再次需要原来的内容时，再将它们写回内存。这一切对用户来说，是完全透明的；运行于 Linux系统下的程序只能看见还有大量的内存空间可以使用，而不会注意到有部分内容有时会被写入磁盘。当然，和使用真正的内存相比，读写磁盘的速度是比较慢的（甚至慢上千倍），所以程序的运行不会很快。被用作虚拟内存的那部分磁盘被称作“ 交换空间 ”（swap space）。

Linux即可采用文件系统内的普通文件，又可采用用于交换空间的独立分区。交换分区要快一些，但交换文件的大小更容易更改（没有必要对整个磁盘重新分区，一切都可从头安装）。在得知自己需要多少交换空间时，你应该选择交换分区，但如果你不确定自己需要多少交换空间，可先采用交换文件，暂时选用系统，以便了解自己需要多少交换空间，在确定空间大小之后，再安排交换分区。

此外，你还应该知道Linux允许一个用户同时使用若干个交换分区和／或交换文件。意思是如果你只是偶尔需要一个非常大的交换空间，可为此设置一个特殊的交换文件，而不是一直保留这样大的分配空间。

注意操作系统上的术语：计算机科学中，交换（把整个进程提出，转入交换空间）和页面调度（实现虚拟内存的一种技术。一次只写入固定大小的块，通常只有几 KB。一块称之为一页）之间是有区别的。一般说来，页面调度更为有效，它也是 Linux常采用的方法，但过去的Linux操作系统总称之为交换。

4.2 创建交换空间

交换文件是一个普通文件；对内核来说，它没有什么特别之处。对内核而言，唯一值得注意的是，它没有漏洞，而且是为 mkswap的使用而准备的。但是，该文件必须驻留在本地磁盘上；如果由于实施时的具体考虑，文件系统已经通过 NFS得以装入，那么，它就不能存在于这样的文件系统中。

交换文件中绝对不能出现漏洞（即不连续地存储）。交换文件保留了磁盘空间，所以在为文件分配一个磁盘扇区时，内核可以快速交换出一个页，无须重写整个交换文件。内核只采用已经分配给该文件的扇区。由于文件中的漏洞意味着那里没有分配相应的磁盘扇区，所以这将为内核带来不便。

要创建一个没有漏洞的交换文件，最好通过下面的命令来进行：

```
$ dd if=/dev/zero of=/extra-swap bs=1024 count=1024
```

```
1024+0 records in
1024+0 records out
$
```

/extra-swap指的是交换文件名，count=后面则是为其指定的文件大小。这个值最好是4的倍数，因为内核交出的内存页的大小是4K。如果该值不是4的倍数，最后2KB就不能得以采用。

交换分区也没有任何特别之处。你可像对待其他的分区一样，创建它；唯一的区别是它被用作原始分区，也就是说，交换分区内根本不包含任何文件系统。把交换分区标记为82类型（Linux交换）倒是个好办法；这样一来，分区清单将更为清楚（虽然内核并没有这方面的要求）。

在已创建交换文件或交换分区之后，需要在其开始之处，写上签名；签名中应该包含一些管理信息，而且这个签名供内核采用。执行此项任务的命令是\cmd{mkswap}，其用法如下：

```
$ mkswap /extra-swap 1024
Setting up swapspace, size = 1044480 bytes
$
```

注意，交换空间还没有开始使用：它是存在的，但内核还没有用它来提供虚拟内存。

在使用mkswap时，应该非常小心，因为它不会对没有用的交换文件或分区进行检查。利用mkswap，将轻易地改写重要文件或分区！遗憾的是，在你安装系统的时候，必须采用mkswap。

Linux内存管理器将每个交换分区的大小限制在127KB左右（由于技术上的原因，实际上的限制是(4096-10)×8×4096=133890048$字节或127.6875MB）。但是，可同时使用16个交换分区，其总字节数可达2GB（很快，我们就要谈到真正的内存了）。

4.3 交换空间的使用

利用swapon，就可开始使用一个经过初始化的交换空间了。这个命令向内核报告可使用的交换空间。到该交换空间的路径作为参数给出，所以对一个临时交换文件开始执行交换时，将采用下面的命令：

```
$ swapon /extra-swap
$
```

在/etc/fstab文件内列出需要的交换空间，就可以自动使用这些交换空间了。

```
/dev/hda8    none   swap   sw 0 0
/swapfile    none   swap   sw 0 0
```

startup（开始）脚本将开始运行swapon -a命令，该命令将开始对/etc/fstab内列出的所有交换空间执行交换。因此，swapon命令通常只有在需要额外交换空间时才使用。

可以用free命令来监视交换空间的使用情况。它会告诉我们总共使用了多大的交换空间。

```
$ free
             total       used       free     shared    buffers
Mem:         15152      14896        256      12404       2528
-/+ buffers:            12368       2784
Swap:        32452       6684      25768
$
```

输出的第一行（Mem:）显示出物理内存的容量。 total列并没有显示出由内核使用的物理内存量，但这通常都在1MB左右。used列显示出已经使用的内存容量（第二行没有把缓冲考虑在内）。free列显示的是完全没有使用的内存容量。 shared列显示出由几个进程共享的内存量：数字越大越好。buffers列显示出当前的磁盘高速缓存容量有多大。

最后一行（Swap:）显示交换空间的一些类似信息。假如这一行全为零，表明你的交换空间尚未激活。

同样的信息也可以通过top命令得到，或使用文件/proc/meminfo内的proc文件系统来得到。目前，很难获得与一个特定交换空间的使用有关的信息。

为了删除一个交换空间，可使用swapoff命令。但通常都没必要这样做，除那些临时交换空间外。交换空间内使用的每个页都会首先得以交换；假如没有足够多的物理内存来容纳它们，它们就会被“交换出去”（到其他的一些交换空间）。假如没有足够多的虚拟内存来容纳全部页，Linux就会出现混乱；经过一段时间，这种混乱状况也许会结束，但在此期间，整个系统是完全无法使用的。因此，删除一个正在使用的交换空间之前，首先应检查是否有足够多的剩余内存（比如用free命令来检查）。

通过swapon -a命令自动使用的所有交换空间都可以用 swapoff -a命令来删除；它会在/etc/fstab文件中检查，判断自己该删除哪些交换空间。注意，任何人工使用的交换空间仍然会继续保持使用状态。

在某些情况下，即使有大量空闲的物理内存，也可能会使用大量的交换空间。例如，假如某个时候需要交换，但在过后的某个时候，一个占用了大量物理内存的进程终止运行，并释放出它占用的所有内存，就会出现这样的情况。交换出去的数据不会自动交换回来，除非有这方面的需要。因此，物理内存可能会长时间保持闲置状态。尽管你不必过多操心这方面的问题，但搞清楚其中的原理，仍然是很有必要的。

4.4 与其他操作系统共享交换空间

许多操作系统都内建了虚拟内存机制。由于只有在操作系统运行时，才需要使用虚拟内存，所以，除了当前正在运行的操作系统之外，其他操作系统的交换空间都被浪费了。因此，一种更有效的策略是让它们共享一个独立的交换空间。尽管这种方法理论上是可行的，但事实上需要一些技巧。Tips-HOWTO内包含相关的实用性建议。

4.5 交换空间的分配

有些人会告诉你，交换空间的大小应该是物理内存的两倍，但这种说法未免太空泛，正确作法是：

1) 首先估计需要的内存容量。这应该是一次可能需要的最大内存容量，它是你打算同时运行的所有程序需要的内存量的总和。要想计算出这一数字，请实际运行你打算同时使用的全部程序。

例如，如果你想运行X，就应该为其分配8MB，gcc需要为它分配数MB的空间（有些文件甚至需要更多的内存，多达数十 MB），等等。注意，内核本身需要占用约 1MB的内存空间，普通外壳和其他小工具各自需要占用几百 KB的空间。注意，这个数字无须特别精确，估计一下就行了，但宁多勿少。

要记住这一点：如果几个人打算同时使用系统，那么他们都会消耗内存。但是，如果两个人同时运行相同的程序，总共消耗的内存却用不着加一倍，这是由于代码页和共享库只存在着一个副本实例。

free和ps命令特别适合用来估算你需要的内存容量。

2) 根据步骤1算出的结果，我们再增加一些保险系数。这是由于你可能会忘记自己需要运行的某些程序，造成前面的估算有误。这样做也是为了保证你最终有足够的额外空间供你调用。这个保险系数不能太大， 1或2兆就够了（交换空间只能多，不能少，但也不必太多，该不会把整个磁盘都用作交换空间吧！那些闲置的交换空间最终还是被浪费掉了；参考后文，了解如何增加额外的交换空间）。除此以外，注意尽可能采用整数，将你估算出来的值四舍五入，换算成一个整数的MB。

3) 根据上面的计算结果，你已经知道了自己总共需要多少内存空间。所以，为了分配交换空间，你只需从这个所需空间总数内减去物理内存的大小即可，这样就能得出自己需要的交换空间啦（有些Linux版本中，还需要为物理内存映像分配空间，所以步骤 2中计算出来的结果就是你所需的交换空间，不应该减去物理内存大小）。

4) 如果你计算出来的交换空间比自己的物理空间大得多（甚至超过两倍），可能应该分配更多的物理内存，不然，系统性能就会有所损失。

至少得保留交换空间总是错不了，即使你的计算表明你根本不需要它。 Linux会主动采用交换空间，所以可让你的物理内存尽可能地空闲出来。 Linux将交换出没有使用的内存页，即使程序并不需要内存。这样一来，需要内存时，就可不必再等待交换了。交换的工作早在之前的磁盘空闲时进行了。

同一个交换空间可跨越几个磁盘。根据磁盘速度和访问磁盘的方式，这样做有时还可改进性能。你可能想尝试几种方案，但一定要知道各种方案是相当不同的。不要相信某某人声称某方案是最好的，因为这类话通常都不可信。

4.6 缓冲区

和访问真正的内存相比，从磁盘上读取数据显然非常之慢（显然，RAM磁盘除外）。另外，在相当短的时间内，多次读取磁盘上的同一部分是常有的事。例如，有人可能会先读取电子邮件消息，然后在答复这封邮件时，把它读入编辑器，再在把它复制到文件夹时，令电子邮件程序再次读取它。或者，以ls命令为例，这个命令非常频繁地运行于一个多用户系统上。只从磁盘上读取一次信息，然后把它保存在内存内，直到不再需要它为止，通过这种方式，除了第一次读取较慢以外，后面的运行速度明显得以加快。这种技术叫作“ 磁盘缓冲 ”，用于该目的的内存叫作“ 缓冲区。”

遗憾的是，由于内存是一个有限的、宝贵的资源，缓冲区通常不可能非常之大（它容纳不下人们想使用的所有数据）。缓冲区充满时，长时间内没有使用的数据就会被丢弃，从而为新数据释放内存空间。

磁盘缓冲对写入数据来说，非常有用。一方面，写入的数据很快又会被再次读取（比如，一个源代码文件被保存为一个文件，然后由编辑器对它进行读取），所以把写入的数据放入缓冲区内是个妙计。另一方面，只把数据放入缓冲区，不立即将其写入磁盘，通过这一方式，程序的运行就要快得多。然后，写入磁盘的操作被移到后台进行，丝毫没有影响其他程序的

运行速度。

许多操作系统都有缓冲区（虽然它们可能也被称为别的什么），但并不表示所有的操作系统都遵循上面的规则。有的“即时写”：数据立即被写入磁盘（当然，同样是保存在缓冲区内）。如果写入操作在稍后进行的话，这个缓冲区就叫作“后写”（write back）。和即时写相比，后写更为有效，但也更容易出错：如果缓冲区内等待写操作的数据在被写入磁盘之前，出现了这类情况：机器崩溃，或电源被切断，或者软盘从软盘驱动器中取走了，缓冲区内的改动通常都会丢失。这意味着文件系统（如果有的话）不再完整，其原因可能是未写入的数据对管理操作员信息来说，是非常关键的。

鉴于此，绝对不应该在没有采用正确的关机进程的情况下，关掉电源，或在未完全装入软盘中数据之前，或它向正在使用它的某某程序发出已完成信号，且软驱指示灯不再闪之前，抽出软盘。sync命令刷新缓冲区，也就是说，迫使所有未写入的数据写入磁盘，供希望确定所有数据都已安全写入的用户使用。在过去的 Unix系统中，有一个程序名为 update，它运行于后台模式，每隔 30s，就执行一次同步处理，所以一般说来，没必要使用 sync命令。Linux系统中有一个额外的后台程序，叫作 bdflush，它更频繁地实行更不完全的同步刷新，以避免由于负荷过大的磁盘I/O导致的宕机，sync有时会出现这样的情况。

Linux系统下，bdflush是由update来启动的。通常情况下，不用过多操心它，但如果出于某种原因，bdflush死了，内核将对此发出警告，而你则应该手工启动它（/sbin/update）。

缓冲区实际上不对文件进行缓冲处理，文件是最小的磁盘 I/O单元（Linux系统下，它们的大小通常是1K）。缓冲处理的对象是目录、超级数据块、其他文件系统管理操作数据和非文件系统磁盘。

缓冲区的效力基本上是由其大小来决定的。小型缓冲区没什么用：其中将容纳极少的数据，以致所有的缓冲数据都是在再次使用它之前，通过缓冲区得以刷新的。临界大小和读写数据的多少有关，与同一个数据的访问频率有关。具体情况只有试过以后，才能知道。

如果缓冲区的大小是固定的，最好不要将它设得很大。因为那样可能令空闲的内存空间太小，导致交换减速。为了能更有效地使用真正的内存，Linux针对缓冲区，自动采用所有空闲RAM，但与此同时，在程序需要更多内存时，也会自动令缓冲区变小。

Linux操作系统中，不必做任何事，就可利用缓冲区，缓冲区的采用完全是自动的。除了遵循正确的关机过程和抽出磁盘外，实在没必要为它操心。

第5章　引导和关机

这部分主要讲解Linux系统引导和关机过程中做了些什么，以及如何才能做到正确引导和关机。如果采取的作法不正确，可能导致文件被讹用或丢失。

5.1　概论

打开计算机，导致加载其操作系统的过程就叫作“引导”（早期的计算机中，只打开计算机是不够的，还必须手工加载操作系统。现在的计算机好多了，一切引导行为都是自动的）。“引导”一词源于计算机通过引导程序，把自己拔起，但自举这一动作更为形象。

引导过程中，计算机先加载一小段被称为“自举程序”的代码，该程序依次加载并启动操作系统。自举程序通常保存在硬盘或软盘上的固定位置。采用两个步骤的原因是操作系统大而复杂，但计算机载入的第一段代码又必须非常之小（只能有几百个字节），以避免固件过于复杂。

不同的计算机执行引导的方式是不同的。对个人电脑来说，计算机（其 BIOS）在软盘或硬盘上的第一个扇道（被称为引导区）进行读取。自举程序就包含在这个扇道内。它从磁盘上的某个地方（或别的地方）载入操作系统。

Linux系统已经载入之后，它便初始化硬件和设备驱动程序，然后运行 init。init启动别的进程，以便允许用户登录和开始工作。关于这部分的详情，参见随后的讨论。

为了关闭Linux系统，首先要求所有的进程终止（其目的是为了让它们关闭所有的文件，执行必要的操作保持干净退出），然后，取消装入文件系统和交换区，最后向控制台打印出一条消息，声明可以关机。如果不遵循正确的关机进程，可能会导致非常严重的后果；更为重要的是，文件系统缓冲区可能没有得到刷新，这意味着该缓冲区内的所有数据会被丢失，磁盘上的文件系统不一致，从而导致文件系统不能使用。

5.2　深入研究引导过程

Linux系统的引导，既可从软盘，又可从硬盘开始。 Linux文档项目（http://MetaLab.unc.edu/LDP/LDP/gs/gs.html）中，由Matt Welsh等编写的《安装和启动指南》内的安装部分将告诉你如何安装Linux系统，以便按自己的方式启动它。

启动PC机时，BIOS将进行各种各样的测试，检查一切是否正常（这称为电源自检，或简称POST），然后开始真正的引导进程。它将选择磁盘驱动（如果已插入软盘的话，一般是第一个软区；不然的话，如果计算机内已安装硬盘的话，就选择第一个硬盘；但其顺序是可以配置的），然后开始读取最前面的第一个扇区。这就是引导区；对硬盘而言，还可把它称为“主引导记录”，因为一个硬盘中可包含若干个分区，每个分区都有自己的引导区。

引导区内包含一个很小的程序（小得以至于一个扇区就能装下它），其责任是从磁盘上读取并启动真正的操作系统。通过软盘引导 Linux系统时，引导区内包含有一小段代码（即引导程序，它从内存中的预定好的位置读取第一个只有几百个字节的数据块，数据块的大小和实际上的内核有关）。Linux启动盘上，没有文件系统，内核只保存在数个连续的扇区内，因为

这样可以简化启动进程。但是，要通过带有文件系统的软盘来启动操作系统，也是可行的，只要你用LILO（即Linux装载程序）。

从硬盘启动时，主引导记录中的代码将检查分区表（同样位于主引导记录内），鉴定活动分区（该分区已被标注为可引导的），再从这个活动分区内读取引导区，然后启动引导区内的代码。分区引导区内的代码和软盘引导区的作用是一样的：从分区内读取并启动内核。但是其细节不同，因为用一个单独的分区来保存内核映像通常是没有用的，所以该分区引导区内的代码不能连续读取硬盘。对于这个问题，有几种解决办法，但最常用的是利用 LILO（如何使用这个程序不在我们的讨论之列；有关详情，参见 LILO文档，该文档讲得非常全面）。

利用LILO启动系统时，它一般会直接读取并启动默认内核。此外，对 LILO进行配置，使其能够启动多个内核之一，甚至除了 Linux之外的其他操作系统也是可能的，对用户来说，也可在启动时，选择准备启动的内核和操作系统。 LILO是可以配置的，因此，如果在启动时（装载LILO时），同时按下Alt、Shift和Ctrl这三个键时，LILO会问你准备启动什么，而不是立即启动默认设置。另外，你可以配置 LILO，令其无论如何都询问，同时指定一个可选的超时值，一旦超过这个时间，就启动默认内核。

利用LILO时，还可以在内核或操作系统名之后，指定一个内核命令行参数。

软盘启动和硬盘启动都各有千秋，但一般情况下从硬盘启动更好一些，因为这样可避免在一大堆软盘中找启动盘。而且，从硬盘启动要快得多。但是，从硬盘安装准备启动的系统是非常麻烦的，所以许多人一般先从软盘启动，然后在采取别的方式安装并使用系统时，安装LILO，从硬盘上开始启动进程。

Linux内核已经被读入内存之后，真正的启动进程开始了，大致会出现下面这些情况：

■ Linux内核是压缩过后安装的，所以第一步是自行解压。内核映像的开始处有一个专门为此设计的小程序。

■ 如果你有Linux能够识别的超级VGA卡，而且有某些特殊的文本模式（比如 100列×40行），Linux将询问你，准备采用哪种模式。内核编辑期间，如果预先设置一个映像模式，它就不会提出此类问题。这同样也是通过 LILO或rdev来完成的。

■ 之后，内核检查还存在哪些其他的硬件（硬盘、软盘、网络适配器等等），并配置相应的设备驱动程序；在它进行这个步骤期间，它将输出自己找到的信息。例如，我在启动它时，它的输出信息就是这样的：

```
LILO boot:
Loading linux.
Console: colour EGA+ 80x25, 8 virtual consoles
Serial driver version 3.94 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
lp_init: lp1 exists (0), using polling driver
Memory: 7332k/8192k available (300k kernel code, 384k reserved, 176k data)
Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M
Loopback device init
Warning WD8013 board not found at i/o = 280.
Math coprocessor using irq13 error reporting.
Partition check:
  hda: hda1 hda2 hda3
VFS: Mounted root (ext filesystem).
Linux version 0.99.pl9-1 (root@haven) 05/01/93 14:12:20
```

不同系统上，文本的具体模式是不一样的，这和硬件、所用的 Linux版本以及模式的配置方法有关。

■ 然后，内核将试着装入根文件系统。装入点是在编辑时间或任何时候，利用 LILO或rdev配置的。文件系统类型可以自动侦测。如果文件系统装入失败了，比如说因为你忘了把相应的文件系统驱动程序包括在内核内，内核就会手足无措，停止不前了（只能这样了）。

■ 根文件系统通常采用只读装入（可以采用设置装入点的方式来设置）。这样一来，就可以令其在装入期间检查文件系统；检查采用只写装入的文件系统，不见得是个好主意。

■ 在此以后，内核便在后台启动 init程序（位于/sbin/init）。init从事各种各样的启动杂活。其具体活动和它的配置有关；关于这方面的详情，可参考第 6章。有一点是确定的，那就是它起码会启动某些基本的后台程序。

■ 然后，init切换到多用户模式，启动虚拟控制台和串行线路的 getty。getty是一个程序，让人们通过虚拟控制台和串行终端进行登录。 init还可能启动其他一些程序，这和它的具体配置有关。

■ 之后，启动完成，系统激活并开始正常运行。

5.3 关机详情

关闭Linux系统时，遵循正确的规则是非常重要的。如若不然，你的文件系统就可能混乱不堪。这是因为Linux有一个磁盘缓冲区，这个缓冲区没有将所有数据立即写入磁盘，而是隔段时间后，再将数据写入磁盘。这样可极大地提高性能，但同时也意味着：如果你一时兴起，随手关掉电源，缓冲区内可能还包含有大量的数据，所以磁盘上的文件系统可能不完整（因为只有其中的部分数据已被写入磁盘）。

反对冒然关掉电源的另一个理由是：在多任务系统中，可能有许多程序正置于后台运行，而关掉电源的后果将是不堪设想的。通过正确的关机顺序，你可以保证所有后台进程都能保存自己的数据。

用于正确关闭Linux系统的命令是shutdown。这个命令通常通过两种方式来使用。

如果你运行于单用户系统上，使用 shutdown的通常方式是退出所有正在运行的程序，注销所有虚拟控制台，以root的身份登录到其中一个控制台（或仍然以 root的身份保持登录状态，但应该转而采用 root的根目录或 root目录，以避免取消装入的问题），然后，给出命令 shutdown -h now（可是，你通常不会只运行于单用户系统之上，如果你想延迟，就用一个“ +”号和一个以min为单位的数来代替now）。

另一种方法是，如果你的系统有多名用户，就采用命令 shutdown -h +time message，time指的是系统停止之前，等待的时间，以 min为单位。而message是对为什么关闭系统的简要说明。

```
# shutdown -h +10 'We will install a new disk.   System should
> be back on-line in three hours.'
#
```

它将警告各位用户，系统将在 10min之后关闭，他们最好马上注销，不然就会丢失所有未保存的数据。这条警告消息将在有人登录的每台终端上打印出来，其中包括所有的 X终端：

```
Broadcast message from root (ttyp0) Wed Aug  2 01:03:25 1995...

We will install a new disk.  System should
be back on-line in three hours.
The system is going DOWN for system halt in 10 minutes !!
```

关机之前，这条警告消息会自动出现若干次，随着时间的推近，其出现的时间间隔越来越短。

任何延迟之后，开始真正的关机时，所有文件系统（除开根文件系统）都将卸装，用户进程（如果有人仍在登录状态的话）将被杀死，后台程序将被关闭，所有的文件系统都被卸装，所有的一切活动都将停止下来。事后， init打印出你可以关机的消息。然后，而且只有在这以后，你才能把手伸向电源开关关机。

有些时候，我们不能正确关闭系统，尽管这在好的系统上很少出现。例如，如果内核出错或崩溃，而且通常在出现异常时，我们就完全不可能执行任何新命令。因此，正确的关闭系统，有时会显得很难。我们只能指望不要出现太严重的数据损毁现象，并正确关闭电源。假如出现的问题并不严重（比如有人拿斧头砍你的键盘），而且内核和更新程序仍然在正常运行，那么一种更好的做法是稍等几分钟，让更新程序有机会刷新你的缓冲区。最后，再关闭电源。

有人喜欢连续执行三次sync命令（sync命令用于刷新缓冲区），等待磁盘I/O终止，再关掉电源。假如当前没有程序运行，利用这个命令，其效果便等同于 shutdown命令。但是，由于它不会卸载任何文件系统，所以有可能在 ext2fs的“清除文件系统”标记中出现问题。因此，我们一般不推荐使用三次sync命令的方法。

假如你想打破砂锅问到底的话，这里便告诉你：之所以会出现“三次 sync”这种看似可笑的做法，原因在于，在早期年代的 Unix系统中，磁盘I/O操作的速度非常慢，所以假如你慢慢地连续键入三次sync命令的话，便能为I/O操作留下充裕的时间，让它“刚好”能够完成。

5.4 重新启动

重新启动意味着再一次启动系统。整个过程是：首先将系统完全关闭，切断电源，再重新加电开机。一个较简单的方法是直接要求 shutdown命令重新启动系统，而非只是简单地关闭。要想做到这一点，方法是为 shutdown命令增加一个-r参数。举个例子来说，你可执行下述命令，要求立即重启：

```
shutdown -r now
```

大多数Linux系统都能在用户用键盘同时按下 Ctrl＋Alt＋Del的时候，自动执行shutdown -r now命令。这样一来，便能立即开始系统的重启。但是，用户按下 Ctrl＋Alt＋Del时系统的反应也是可能修改的。事实上，一种更好的做法是，一旦用户按下该组合键，便预留出一段延迟时间，以便完成一个多用户系统的重启。至于那些任何人都能亲自操作的机器（谁都能用它的键盘），则最好配置成按下Ctrl＋Alt＋Del后，什么事情也不做！

5.5 单用户模式

shutdown命令亦可用于将系统切换为单用户模式。在这种模式中，任何人都不能登录，只有root才能使用控制台。这对系统管理员来说特别有用，因为对他们的某些任务而言，在系

统正常运行的时候，是根本无法进行的。

5.6 应急盘

显然，并非总能从硬盘正常启动一台计算机。举个例子来说，假如你在配置 LILO的时候犯了一个错误，那么系统也许会根本无法启动。在这样的情况下，我们需要通过另一种方式，来完成系统的启动（只要硬件正常，这种方式就一定是可行的）。对典型的PC来说，这便意味着要从软盘启动！

在目前发布的大多数Linux系统中，都允许操作者在安装时创建一张应急启动软盘。这是一个不错的选项。但请注意，在这样的启动盘中，可能只包含了内核，而且会假定你会用安装盘上的程序来修正以后碰到的任何问题。但某些情况下，那些程序并不够用。例如，可能需要恢复一些以后制作的文件，而那些文件并非是用安装盘上的程序（软件）来制作的。

所以，最好同时再创建一张自定义的根软盘。在由 Graham Chapman编写的《Bootdisk HOWTO》中，给出了一系列如何制作应急盘的指示，极具参考价值。网址是 http://MetaLag.unc.edu/LDP/HOWTO/Bootdisk-HOWTO.html。当然，应急盘也不能制作完毕后便搁在一边不管了，还得注意经常更新它的内容。

对于用来装入根软盘的软盘驱动器来说，不得把它用于其他任何目的。因此，假如你只有一部软盘驱动器的时候，便显得非常不便。但是，假如你的内存足够大，便可考虑配置你的启动盘，将根盘载入一个 RAM虚拟盘里（启动软盘的内核需要为此专门配置）。一旦根软盘正确装入RAM盘，软盘驱动器便可自由地装入其他软盘了。

第6章 init

本章的主题是init进程，它是一个由内核启动的用户级进程。init有许多重要的职责，比如启动getty（以便用户能够登录），实施运行级和照顾孤儿进程等。本章将讨论如何配置 init，如何利用不同的运行级。

6.1 init的重要作用

init是Linux系统操作中绝不可少的程序之一，但你大多数时间都可以忽略它。一个编得好的Linux系统中，都会附带适用于多数系统的init，而且在这些系统上，根本不用为init操心。通常，只有在你挂上串行终端、拨入（注意非拨出） modem或打算更改默认的运行级时，才有必要去关心init。

内核自行启动（已经被载入内存，开始运行，并已初始化所有的设备驱动程序和数据结构等）之后，就通过启动一个用户级程序 init的方式，完成了自己的引导进程。所以， init始终是第一个进程（其进程编号始终为1）。

内核会在过去曾使用过init的几个地方查找它，但它的正确位置（对 Linux系统来说）是/sbin/init。如果内核找不到init，它就会试着运行/bin/sh，如果运行失败，系统的启动也会失败。

init启动时，先执行大量的管理任务，比如检查文件系统，清除 /tmp，启动各种服务以及针对用户将能够登录的每台终端和虚拟控制台，启动 getty（有关详情，参见第7章），完成启动进程。

系统正常启动之后， init在用户已经注销登录后，针对每个终端重新启动 getty（以便下一名用户的登录）。init还收养“孤儿”（独立的）进程：进程启动一个子进程并先其子进程死掉，这个子进程立即就成了init的子进程。由于各种技术方面的原因，这一点相当重要，因为它有助于我们进一步了解进程列表和进程树图表（ init本身是不能死的。即使用SIGKILL也不能杀死它）。有几个 init的变体是可以用的。许多 Linux版本都采用 sysvinit（Miquel van Smoorenburg编写的），这个程序基于System V init设计。Unix的BSD版本中另有一个init。两者间的主要区别在于运行级： System V有运行级， BSD则没有（至少过去没有）。这点区别不太重要。我们下面只谈谈sysvinit。

6.2 通过init启动getty：/etc/inittab文件

系统启动时， init开始读取/etc/inittab配置文件。系统运行期间，如果系统发出一个 HUP信号“ kill -HUP 1 as root ”，它将再次读取这个配置文件。这个特性令其不必要启动系统，更改init配置。

/etc/inittab文件有点复杂。我们将从配置几个简单的 getty行开始。位于/etc/inittab的这些行由四个用冒号定界的字段组成：

```
id:runlevels:action:process
```

下面是对这四个字段的说明。此外， /etc/inittab中可包含空行和以“ # ”号开头的行；这些行是可被忽略。

1. id

这个字段定义文件内的行。对 getty行来说，它指定的是它运行的终端（设备文件名内/dev/tty后面的字符）。对其他行来说，它没什么作用（只是一个长度限制），但它应该是独一无二的。

2. runlevels

应该为代码行考虑运行级别。运行级别应该以单一的数位来指定，不带定界符（关于运行级别的详情，参见下一小节）。

3. action

代码行应该采取的行动，例如respawn用于在退出时，再次运行下一个字段中的命令。

4. process

准备运行的命令。

为了在第一个虚拟终端上启动 getty（/dev/tty1}），所有普通多用户运行级别（ 2～5）内，都应该写入这样一行：

```
1:2345:respawn:/sbin/getty 9600 tty1
```

第一个字段指这一行用于 /dev/tty1。第二个字段指该行的运行级别是 2、3、4、5。第三个字段是说这个命令应该在其退出之后，再执行一次（以便另外的用户能够登录、注销，然后再次登录）。最后一个字段是命令，这个命令将在第一个虚拟终端上运行 getty（不同版本的getty的运行是不一样的。查看你的手册页并保证其是完全正确的）。

如果你想在系统中增添终端或拨入 modem，最好在/etc/inittab内多增添几行，一行对应一个终端或一条拨入线路。关于这方面的详情，可参考 init、inittab和getty手册页。

如果命令在启动时就失败了，而且 init被配置为重启，它就会使用大量的系统资源： init启动它，不行，再重启，再不行，再重启，如此“ 纠缠不休 ”，最后耗完所有的系统资源为止。为了防止出现这种情况， init将对重启命令的频率进行跟踪，如果频率过快，它将在重启之前，延迟5min。

6.3 运行级别

运行级别是指init和整个系统的状态，该系统定义了对哪些系统服务进行操作。运行级别是按照编号来识别的，如表 6-1所示。

表6-1 运行级别号

0	使系统停止
1	单用户模式（用于特殊管理）
-2	普通操作（用户自定义）
6	重启

有的系统管理员利用运行级别来定义正在使用的自系统，比如是否正在运行 X，网络是否可操作等等。其他管理员则在不更改运行级别的情况下，让所有的子系统单独运行或启动和终止它们，因为对控制他们的系统而言，运行级别太粗略了。具体情况需要你自行决定，但最容易的方式是照你的Linux版本中所说的去做。

运行级别的配置是在/etc/inittab行内进行的，如下所示：

```
12:2:wait:/etc/init.d/rc 2
```

第一个字段是一个任意指定的标签，第二个字段表示这一行适用于运行级别 2。第三个字段表示进入运行级别时， init应该运行第四个字段内的命令一次，而且 init应该等待该命令结束。/etc/init.d/rc命令运行启动和终止输入以便进入运行级别 2时所需的任何命令。

第四个字段中的命令执行设置运行级别时的一切“ 杂活 ”。它启动已经没有运行的服务，终止不应该再在新运行级别内运行的服务。根据 Linux版本的不同，采用的具体命令也不同，而且运行级别的配置也是有差别的。

init启动时，它会在/etc/inittab内查找一个代码行，这一行指定了默认的运行级别：

```
id:2:initdefault:
```

你可以要求init在启动时，进入非默认运行级别，这是通过为内核指定一个“ single ”或“ emergency ” 命令行参数来实现的。比如说，内核命令行参数的指定可通过 LILO来执行。这样一来，你就可以选择单用户模式了（即运行级别 1）。

系统正在运行时， telinit命令可更改运行级别。运行级别发生变化时， init就会从/etc/inittab运行相应的命令。

6.4 /etc/inittab中的特殊配置

/etc/inittab中，有几个特殊的特性，允许 init重新激活特殊事件。这些特殊特性都是用第三个字段中的特殊关键字标记出来的。比如：

1. powerwait

允许init在电源被切断时，关闭系统。其前提是具有 UPS和监视UPS并通知init电源已被切断的软件。

2. ctrlaltdel

允许init在用户于控制台键盘上按下 Ctrl+Alt+Del组合键时，重新启动系统。注意，如果该系统放在一个公共场所，系统管理员可将 Ctrl+Alt+Del组合键配置为别的行为，比如忽略等。

3. sysinit

系统启动时准备运行的命令。比如说，这个命令将清除 /tmp。

上面列出的特殊关键字尚不完整。其他的关键字及其使用详情，可参考你的 inittab手册页。

6.5 在单用户模式下引导

一个重要的运行级别就是单用户模式（运行级别 1），该模式中，只有一个系统管理员使用特定的机器，而且尽可能少地运行系统服务，其中包含登录。单用户模式对少数管理任务（比如在/usr分区上运行fsck）而言，是很有必要的，因为这需要卸载分区，但这是不可能的，除非所有的服务系统已被杀死。

一个正在运行的系统可以进入单用户模式，具体做法是利用 init，请求运行级别 1。内核启动时，在内核命令行指定 single或emergency关键字，就可进入运行级别 1了。内核同时也为init指定命令行， init从关键字得知自己不应该采用默认的运行级别（内核命令行的输入方式

和你启动系统的方式有关）。

有时，以单用户模式进行启动是必要的，这样一来，用户在装入分区之前，或至少在装入分散的/usr分区之前，能手工运行 fsck（在分散的文件系统上，任何活动都可能使其更为分散，所以应该尽可能地运行 fsck）。

如果自动化的 fsck在启动时失败了，启动脚本 init的运行将自动进入单用户模式。这样做是为了防止系统使用不连贯的文件系统，这个文件系统是 fsck不能自动修复的。文件系统不连贯的现象极为少见，而且通常会导致硬盘的不连贯或实验性的内核释放，但最好能做到防患于未然。

由于安全上的考虑，在单用户模式下，启动外壳脚本之前，配置得当的系统会要求用户提供root密码。否则，它会简单地为 LILO输入合适的一行代码，以 root的身份登录（当然，如果/etc/passwd已经由于文件系统的问题而不连贯了，就不适合这里的原则了，为对付这种情况，你最好随时准备一张启动盘）。

第7章　登录和注销

本章将说明用户在登录和注销时发生的行为。另外，还要详细介绍后台进程、日志文件、配置文件等的交互过程。

7.1　通过终端登录

图7-1展示了如何通过终端进行登录。首先， init要确定有一个供该终端连接（或控制台）使用的getty程序。getty在终端上监听并等待用户通知它“他／她准备登录了”（这通常意味着用户必须输入点东西）。它注意到用户后， getty就输出一条欢迎消息（保存在/etc/issue）内，提示用户输入用户名，最后才运行 login（登录）程序。login得到作为参数的用户名后，提示用户输入密码。如果两者相符， login便开始启动为该用户配置的外壳脚本；如果两者不符， login只好退出并中断登录进程（也许要求用户再次提供用户名和密码之后）。init注意到登录进程中断后，就为终端启动一个新的getty。

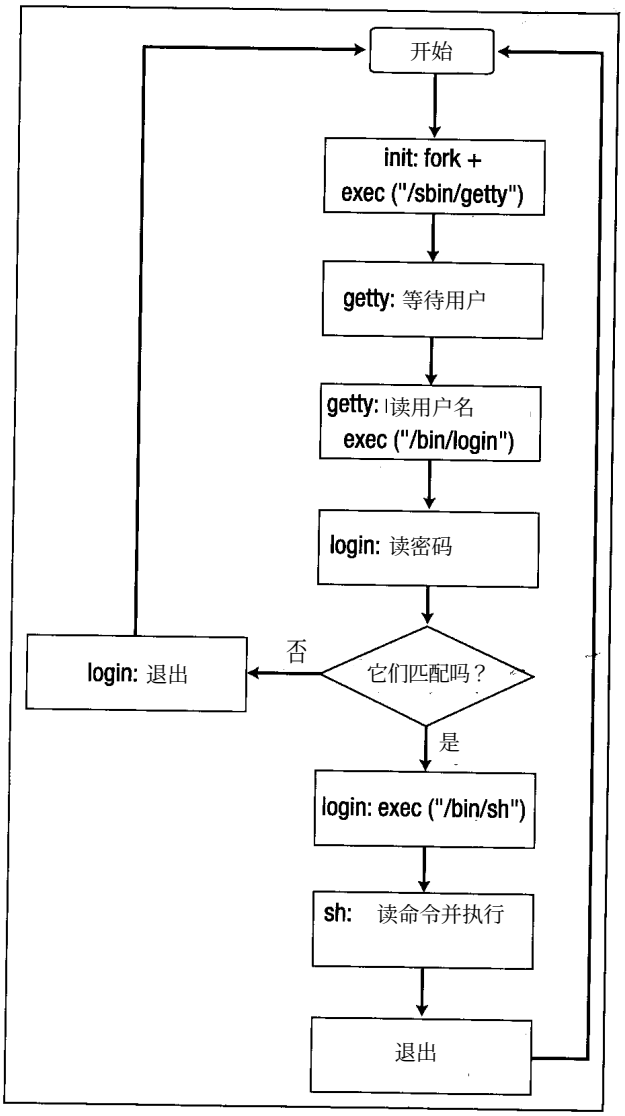

图7-1　通过终端登录：init、getty、login和shell之间的交互活动

注意，唯一的新进程是由 init创建的那个（利用 fork系统调用创建的）；getty和login只能替换进程中正在运行的程序（利用 exec系统调用）。

值得用户注意的是，串行线路需要一个单独的程序，因为在终端进入活动状态时，这个程序可能（过去一直如此）非常复杂。getty还可适应于连接的速率和其他参数设置的变化，这些对拨入连接来说，是特别重要的，因为对拨入连接来说，参数可能会不

时地发生变化（即每次呼叫时的参数都可能不一样）。

getty和init都有好几个版本，它们都有各自的优缺点。因此，你最好要知道自己系统内采用的是什么版本，与此同时，还需要对其他的版本有所了解（可利用 Linux Software Map来搜索它们）。如果没有拨入连接，大可不必太多关注 getty，但仍需了解init。

7.2 通过网络登录

同一个网络内的两台计算机通常通过一条单独的物理线缆连接在一起。当它们通过网络进行通信时，对参与通信的每台计算机上的程序来说，都会通过一条虚拟的连接（一条假想的线缆）链接在一起。只要虚拟连接任何一端上的程序被包含在内，都会有自己专用的“线缆”。但是，由于线缆不是真的，只是假想的，两台计算机上的操作系统都可能出现数条虚拟连接共享同一条物理线缆的情况。通过这种只采用一条物理线缆的方式，数个程序可以在不知道或不担心其他通信的情况下，进行通信。甚至于数台计算机采用同一条物理线缆也是可能的；虚拟连接存在于两台计算机之间，而其他计算机忽略那些它们自己没有参与的连接。

以上说明复杂而且过于抽象。但是，仍然非常易于理解，我们从中能看出网络登录和普通登录之所以不同的重要原因。虚拟连接是在有两个程序希望通信时建立的，这两个程序分别在不同计算机上。由于网络内的任何一台计算机都有可能想登录到另一台计算机，因此可能存在数目较大的虚拟通信。正由于此，要对每一次潜在的登录启动 getty几乎是不现实的。

所以，一个能处理所有网络登录的单独进程应运而生，它就是 inetd（对应于getty）。它注意到进入的网络登录（也就是说，它注意到自己与另一台计算机建立了新的虚拟连接），就启动一个新进程来处理单独的登录。原来的进程仍然存在，并继续监听新的登录。

为了能处理更多的事情，网络登录使用的通信协议不止一种。最重要的两个是 telnet和rlogin。除了登录以外，还可能实行其他许多虚拟连接（针对 FTP、Gopher、HTTP和其他一些网络服务），有一个单独的进程来监听特定类型的连接是非常有效的，所以就出现了一个专门的监听者，它能够识别连接类型并启动相应类型的程序来提供服务。这个单独的监听者就是/cmd{inetd}；有关详情，请参考本书第一部分。

7.3 登录的意义

login程序负责对用户进行身份验证（确定用户名和密码是否匹配），以及为用户设置初始化环境，这是通过为串行线路设置访问许可和启动 shell（外壳）来完成的。

部分初始化设置将输出文件 /etc/motd（message of the day的缩写，意为日期消息）的内容，并检查电子邮件。这些设置是可以禁用的，具体做法是在用户的根目录内，创建一个名为.hushlogin的文件。

如果/etc/nologin文件存在，登录就会被取消。这个文件一般是由 shutdown及其关系词创建的。login检查这个文件，如果这个文件存在的话，它将拒绝接受登录请求。如果这个文件不存在，login就会在退出之前，把自己的内容输出到终端。

login把所有失败的登录尝试记录在系统日志文件中（通过 syslog）。另外，它还记录了root的每次登录。在跟踪入侵者时，这些记录都是非常有用的。

当前已经登录的用户都列在 /var/run/utmp内。只有在系统下一次重启或关闭之前，这个文件才是有效的；系统启动时，就会被清除。该文件列出了每个用户以及他们使用的终端（或

网络连接），另外还有其他有用的信息。 who、w和其他类似的命令将在 utmp内查看登录的用户。

所有成功的登录都记录在 /var/log/wtmp文件内。该文件将无限制地增长，所以必须定期对它进行清除，最好每周执行一次 cron任务，清除它。最后一个命令用于浏览 wtmp文件。

utmp和wtmp均是二进制格式（参见 utmp手册页）；但遗憾的是，没有特定的程序，是不方便对它们进行检查的。

7.4 访问控制

过去，用户数据库包含在 /etc/passwd文件内。有的系统采用影子密码，并且已经把密码挪到/etc/shadow文件内。对具有多台计算机（它们共享账号）的站点来说，利用 NIS或别的方法来保存用户数据库；它们还可能自动把数据库从一个集中地点复制到所有的计算机。

用户数据库内不仅包含密码，还有一些和用户有关的额外信息，比如用户的真名、根目录和登录外壳等。这些额外的信息需要公开，以便任何人都可读取它们。因此，密码要实行加密保存。这样做的确有不足的地方，因为任何访问加密密码的人可采用各种解密方法来猜测真正的密码，根本无须尝试实际登录计算机。影子密码试图通过将密码移入另一个文件（只有root才能读取它，但密码仍采用加密方式保存）内的方式，来避免此类情况的发生。但是，在一个不支持影子密码的系统上安装它不是件简单的事。

不管有无密码，重要的是要确定系统内的所有密码都是可靠的，也就是说，不是轻易就能猜出来的。解密程序能够用来对密码进行解密；它能找出的任何密码都被认为是不可靠的。解密程序可以由入侵者实施，但也可由系统管理员来实施，以避免采用不可靠密码。可靠的密码也可以由 passwd程序来执行；实际上，这在 CPU周期中更为有效，因为解密密码需要进行大量的计算。

用户组数据库保存在 /etc/group内；对安装了影子密码的系统来说，用户组数据库也可能是/etc/shadow.group。

root通常不能通过大多数终端或网络登录，它只能通过 /etc/securetty文件内列出的终端进行登录。这样一来，就有必要与这些终端之一建立物理连接，以便进行访问。但是，它也可以作为任何一名其他的用户，通过任何一个终端进行登录，然后再用 su命令，恢复 root的身份。

7.5 外壳的启动

一个交互性的登录外壳启动时，它将自动执行一个或多个预先定义好的文件。不同的外壳，执行的文件也是不同的；有关详情，参见各个外壳的文档。

许多外壳首先运行某个全局文件，比如 Bourne外壳（/bin/sh）及其衍生程序执行的是/etc/profile；另外，它们还将执行用户根目录下的 .profile文件。/etc/profile允许系统管理员设置好一个普通的用户环境，特别是设置路径，使其除了包含普通目录外，还要包括本地命令目录。另一方面， .profile还允许用户在必要时，通过改写默认设置的方式，自定义他／她的用户环境。

第8章　用户账号x的管理

本章将解释如何建立新的用户账号，如何修改用户账号的属性，以及如何删除账号。不同版本的Linux，将采用不同的工具来对这一切进行管理。

8.1　何谓账号

当一台计算机供多名用户使用时，通常有必要区分各个用户，以便他们的私人文件只供他们私人使用。即使该计算机一次只供一名用户使用时，也同等重要（如果不这样的话，某某人可以看到我的情书，多难为情）。因此，每名用户都将被指定一个独一无二的用户名，这个用户名供他们登录时使用。

但是，对用户来说，还有更重要的。那就是账号。所谓账号，就是属于某个用户的所有文件、资源和信息。在银行和商业系统中，账号通常和钱有关。根据用户的使用程度，账号上的钱以不同的速度减少至零。举个例子来说，用户在磁盘空间上的消费就比较缓慢，但是处理器时间就显得要“宝贵”得多。

8.2　创建用户

Linux内核本身把用户当作数字对待。每个用户都是用一个独一无二的整数来标识的，它就是用户ID或称uid，因为对计算机来说，与处理文字化的用户名相比，对数字的处理要快得多，而且容易得多。

内核外部的一个独立的数据库负责为每个用户 ID分配一个文字化的名称，也就是用户名。这个数据库中也包含额外的一些信息。

为了创建用户，你需要把与之相关的用户信息添加到用户数据库内，并为准备创建的用户创建一个根目录。同时，还必须训练这个新用户，为它设置一个合适的初始化环境。

许多Linux版本都带有用于创建账号的程序。有几个此类的程序是可以采用的。这里，有两个命令行供你选择，它们是 adduser和useradd；另外，可能还有一个 GUI工具。不管采用什么程序，如果采用手工操作的话，其结果都将收效甚微。即使其细节翔实而复杂，这些程序仍使创建用户的操作变得琐碎平常。随后的“手工创建用户”中，将为大家介绍如何手工创建用户。

8.2.1　/etc/passwd和其他的信息性文件

Unix系统中，基本的用户数据库是一个文本文件，名为 /etc/passwd（也称为密码文件），它列出了所有有效用户名及其相关信息。该文件内，每个用户名都有一行，它被分为 7个字段，中间用冒号定界：

- 用户名。
- 密码，采用加密形式。
- 数字式的用户ID。

- 数字式的组ID。
- 账号的全名或其他说明。
- 根目录。
- 登录外壳。

有关其格式的详情，可参考passwd手册页。

系统上的任何一名用户都可能读取这个密码文件，所以他们能够从中得知另一个用户的用户名。这意味着人人都能看见密码（第二个字段）。密码文件对密码进行了加密，所以从理论上讲，以加密形式保存的密码是可靠的。但是，加密也是很容易被解密的，特别是密码比较薄弱的时候（也就是说，如果它很短，或很容易猜测得到的话）。因此，把密码放在密码文件内通常不是个好主意。

许多Linux系统都有影子密码。这是另一种保存密码的方案：加密的密码被保存在一个独立的文件内，该文件名为 /etc/shadow，只有root才能读取这个文件。 /etc/passed文件只在第二个字段内包含一个特殊的标记。任何需要验证用户是 setuid的程序，都可因此而访问影子密码文件。以只能使用密码文件内其他字段的普通程序来说，它们是不能看到这个影子密码文件的（对，这意味着密码文件中有关于用户的所有信息，但密码除外）。

8.2.2 如何选择数字式用户和组ID

大多数系统上，对数字式用户和组 ID没有具体的标准，但是在使用 NFS（网络文件系统）时，所有的系统上都必须采用同一个 uid和gid。这是因为NFS也要利用数字式uid来识别用户。如果你没有用NFS，大可让你的账号创建工具自动选择 ID。

在使用NFS时，将必须发明一种机制，用于同步更新账号信息。有个方案是 NIS系统（详情参考本书的第一部分“ 网络管理员指南 ”）。

但是，你应该尽量避免重复使用数字式 uid（和文字化的用户名），因为uid（或用户名）的新拥有者可能访问旧拥有者的文件（以及邮件等）。

8.2.3 初始化环境：/etc/skel

新用户的根目录建立时，要通过 /etc/skel目录下的文件，对它进行初始化。系统管理员可创建/etc/skel内的文件，使其为用户提供一个好的默认环境。举个例子来说，他可创建一个 /etc/skel/.profile文件，该文件把EDITOR环境参数设置为某个编辑器，对新用户来说，这个编辑器是非常友好的。

但是，通常情况下，最好尽量让 etc/skel尽可能的小，因为要更新现成用户的文件几乎是不可能的。例如，如果友好编辑器的名称发生了变化，全部现成用户都将不得不编辑自己的.profile。系统管理员可试着利用一个脚本自动处理，但毫无疑问，这样做肯定会损毁有些用户的文件。

可能的情况下，最好把全局配置放入诸如 /etc/profile之类的全局文件内。从而可能在不损毁用户本人设置的前提下，更新所有现成用户的文件。

8.2.4 手工创建一个用户

要手工创建一个用户，需遵循下面的步骤：

1) 利用vipw退出/etc/passwd，为新账号增加一个新行。注意语法。千万不能直接用编辑器编辑！vipw锁住了文件，所以其他命令不会同时对该文件进行更新。你应该令密码字段为“*”，以便用户不可能进行登录。

2) 如果你还需要创建一个新组的话，采用类似的做法，利用 vigr编辑/etc/group。

3) 利用mkdir创建用户根目录。

4) 把文件从/etc/skel复制到新建的根目录。

5) 利用chown和chmod，确定拥有权和访问许可。 -R选项是最有用的。对不同站点的访问许可同中存异，但通常都会采用下面的命令：

```
cd /home/newusername
chown -R username.group .
chmod -R go=u,go-w .
chmod go= .
```

最后，利用passwd设置密码。

设置密码之后，账号就开始发挥作用了。注意，在别的操作没有完成之前，一定不要设置密码，如若不然，用户就可能在你正忙于复制文件时，一不小心登录进来。

有时，创建无人使用的伪账号是很有必要的。例如，为了设置一个匿名 FTP服务器（以便任何人无须先获得账号，就可从这里下载文件），你需要建立一个名为ftp的账号。碰到此类情况时，通常没必要设置密码（即上面的最后一步）。实际上，最好不要设置，因为没有人能够用那个账号，除非先拥有root的身份，因为root可以成为任何一个用户。

8.3 更改用户属性

下面有几条命令，是用于更改账号属性（也就是 /etc/passwd内的相关字段）的：

- chfn —— 更改完整用户名字段。
- chsh —— 更改login外壳。
- passwd —— 更改密码。

超级用户可能用这些命令来更改任何一个账号的属性。普通用户只能更改他们自己账号的属性。对普通用户来说，有时可能还必需禁用这些命令（利用 chmod来实现），比如在有许多新用户的环境中。

其他的事情需要手工完成。比方说，要更改用户名，你需要直接编辑 /etc/passwd（记住用vipw）。类似地，要增加或删除用户，需要编辑 /etc/group（用vigr）。但是，这类任务较少执行，而且执行过程中应该谨慎：举个例子来说，如果你更改了用户名，电子邮件再也不能抵达这个用户的手中，除非你另外还建立了一个邮件别名（用户的名字可能会因为结婚而发生变化，而且他希望有一个用户名来体现自己的新名）。

8.4 删除用户

要删除一个用户，必须先删除他的所有文件、邮件别名、打印作业、 cron和at作业，以及其他对该用户的所有引用。然后，才能从 /etc/passwd和/etc/group内删除相关的行（记住，从已经添加该用户名的所有组中，删除这个用户名）。较好的做法是在开始删除用户名之前，先取消其账号（如下所示），以防止该用户在删除期间，仍然使用这个账号。

记住，用户在其根目录外面可能还有文件。利用 find命令可以找出它们：

```
find / -user username
```

但是注意，如果你的硬盘很大的话，上面的命令会花很长的一段时间。如果你装入网络磁盘，就一定要小心行事，以便自己不会丢弃网络或服务器。

有些Linux版本附带某些特殊的命令来执行这项任务；查找 deluser和userdel。然而，用手工执行一样容易，毕竟命令不是万能的。

8.5 临时禁用用户

有时，必须在不删除账号的情况下，临时禁用它。举个例子来说，用户可能没有按时交费或系统管理员怀疑有人盗用账号等。

要禁用账号，最好的方法是将其外壳改为一个特殊的程序，这个程序只打印一条消息。通过这种方式，任何想采用这个账号登录的人都会失败，而且将知道失败的原因。这条消息能够要求用户和系统管理员取得联系，以便解决存在的问题。

另外，把用户名或密码更改为别的东西也是有可能的，但那样的话，用户就不能知道到底发生了什么事。

要创建这样的特殊程序，较简单的方法是编写“ tail脚本 ”，如下所示：

```
#!/usr/bin/tail +2
This account has been closed due to a security breach.
Please call 555-1234 and wait for the men in black to arrive.
```

前两个字符“ #! ”告诉内核该行中的其他部分是一条命令，该命令需要得以运行，以便翻译这个文件。这里的tail命令将在标准输出内输出除了第一行以外的所有消息。

如果用户billg被怀疑违反了安全规则，系统管理员就会像这样：

```
# chsh -s /usr/local/lib/no-login/security billg
# su - tester
This account has been closed due to a security breach.
Please call 555-1234 and wait for the men in black to arrive.
#
```

当然，su的目的是测试改动是否见效。

tail脚本应该保存在一个单独的目录下，以便其命令不和普通用户命令发生冲突。

第9章　备　　份

本章将讨论为什么要备份，怎样备份，何时备份以及如何通过备份恢复数据。

9.1　备份的重要性

谈到备份的重要性，相信大家都有过痛失数据的惨痛经历。而这种丢失数据的心情，比失恋还要难受，甚至让人心灰意冷。因为你的数据是有价值的。一旦丢失，重新建立它，得花你许多时间和精力，有的甚至还要花不少钱；更糟的是，有时甚至不可能重建它。数据包含着你的喜与乐，包含了你的工作成果，所以你应该保护它，采取相应的行动，以避免失去它。解决之道就是备份。

数据可能丢失的原因基本上可归结于四个：硬件故障、软件错误、人为因素和自然灾害（第五个原因就是“其他原因”）。虽然现在硬件越来越可靠，但仍然可能存在例外的情况。保存数据的硬件部分中，最关键的部分是硬盘，除了会发出微小的噪声外，它几乎是最理想的存储设备。现在的软件就不那么可靠了，当然，坚如磐石，固若金汤的软件产品也不是没有，只不过少罢了。人才是最不可靠的；他们不是操作失误，就是有目的地、或者恶作剧地破坏你的数据（比如，前不久，管理员就对我的主页开了个玩笑，让它消失了一晚上）。自然灾害只能怪你运气不好。总的说来，谁也不能绝对保证你的数据不遭破坏或损毁。

所谓备份，就是保护数据的一种手段。通过保留若干个备份的方式，能够在数据受损的情况下，尽快地恢复它（用不着伤心，用不着花太多的时间、精力和钱，只须从备份恢复丢失的数据即可）。

适当地备份是非常重要的。和现实生活中没有永恒的事物一样，备份迟早会无效。适当备份过程中最重要的是保证备份能够发挥作用；你肯定不希望看到自己的备份不能用吧（不要笑，有些人就是这样的）。有时候是屋漏偏遭连夜雨，就在你制作备份的时候，系统却崩溃了；如果你只有一个备份媒体的话，它也有可能遭到损毁，让你气得鼻子冒烟。或者说，在你试图恢复数据时，发现自己居然忘了对一些重要数据进行备份，比如一个有 15 000用户的站点上的用户数据库。最为理想的是，你的所有备份都能用。

9.2　选择备份媒体

关于备份，最重要的是如何选择备份媒体。在选择备份媒体时，需要考虑这几个因素：价格、可靠性、速度、可得性和可用性。

价格是比较重要的因素，因为你可能有许多数据需要多次备份。选择便宜点的存储媒体是必须的。

可靠性相当重要，因为一个被破坏了的备份无异于没有备份，想象吧，面对丢失了的数据，你满心以为用备份来恢复，结果备份也是被破坏了的，这种情形真让人欲哭无泪。在理想情况下，备份媒体能够保存数据的时间长达数年。对备份媒体而言，你使用媒体的方式也会对其可靠性产生一定的影响。一般来说，硬盘是非常可靠的，但如果备份媒体和你正在备

份的磁盘处于同一台机器上，备份媒体就不那么可靠了。

通常情况下，如果备份时不需要交互操作的话，速度就不是十分重要。只要它不需要人监管，花上两小时也无所谓。但另一方面，如果因为计算机空闲，备份不能完成的话，就应该考虑备份速度了。

显然，可得性是必需考虑的因素，总不可能在没有备份媒体的情况下，使用它吧。比如说，你没有光盘刻录机，怎么能将自己的数据备份在光盘上呢？（将来逐渐不需要拥有备份媒体，而是可从别的计算机上获得备份数据）。如果不考虑存储媒体的可得性，就不可能在灾难之后恢复数据。

在考虑备份频率时，可用性最为重要。制作备份的方式越简单越好。备份媒体绝对不应该不好用。

备份媒体一般采用软盘和磁带。软盘非常便宜、相当可靠、速度不是非常快、非常容易获得，但不适合量大的数据。磁带则有贵的、有便宜的，它相当可靠、速度相当快、非常容易获得（和磁带的容量大小有关），相当好用。

另外还有一些备份媒体。它们通常不太容易获得，但它们在其他方面的表现却非常出色。比如，光盘既有软盘的优点（随机访问，很快就能恢复单独的一个文件），又有磁带的优点（保存大量的数据）。

9.3 选择备份工具

用于制作备份的工具有许多。过去用于备份的 Unix工具是tar、cpio和dump。另外，还有许多第三方工具包（即有自由软件，又有商业软件）。备份媒体的选择直接对备份工具的选择产生影响。

从备份角度来说，tar和cpio极为相似。两者都能把数据保存在磁带上，并从磁带恢复数据。两者均适用于任何一种媒体，因为内核设备负责管理低级控制设备和越来越像用户级程序设备。tar和cpio的有些Unix版本可能不能备份非普通文件（比如符号链接、设备文件、带有超长路径名的文件等），但它们的Linux版本则可以轻松胜任所有文件的备份。

dump不同于前两者的地方在于：它直接读取文件系统，而不是通过文件系统读取文件，它也是专门针对备份而设计的；tar和cpio其实是用于对文件进行归档的，但同样可用于备份。

直接读取文件系统有诸多好处。比如说，可在不影响文件时间戳的情况下，备份文件；如果利用tar和cpio，你就必须先以只读方式装入文件系统。此外，如果所有文件都需要备份的话，直接读取文件系统也更为有效，因为这样可减少磁头的“运动量”。直接读取文件系统的缺点在于备份程序只能专用于一种文件系统类型； Linux dump只能理解ext2类型的文件系统。

dump还支持备份级别（稍后着重讨论）；对 tar和cpio来说，必须借助于其他工具，才能实施备份级别。

至于第三方备份工具，不在本书讨论之列。 Linux Software Map列出了许多用于备份的自由软件。

9.4 简单备份

简单备份方案指一次性备份所有的文件，然后再备份上次备份之后所做的修改。这种方

案中，第一次备份称为“完全备份”（full backup），后一次备份称为“新增备份”（incremental backups）。完全备份通常比新增备份费力得多，因为需要写入磁带的数据比后者多得多，而且完全备份需要的磁盘（或软盘）可能不止一盘。但和通过完全备份恢复数据相比，从新增备份中恢复数据所花的时间和精力是前者的数倍。当然，数据的恢复也是可以优化的，这样一来，就可以一直备份自上次完全备份以来的所有数据；这时虽然有些费神，但至少除了恢复一个完全备份和一个新增备份以外，无须再恢复其他的东西了。

如果你打算每天都进行备份，而且手中有 6盘磁带，就可用磁带 1来备份第一次完全备份（比如说，从星期五开始），磁带2和5用于新增备份（星期一到星期四）。然后，用磁带 6开始第二次完全备份（第二个星期五），并再次用磁带2和5进行新增备份。如果你不打算在执行新的完全备份之前，改写磁带 1，就不会有异常情况发生。在用磁带 6进行第二次完全备份之后，要把磁带1保存在某个地方，以便在其他备份磁带受损的情况下，仍然还有起死回生的最后一线希望。在需要进行下一次完全备份时，可采用磁带 1，让磁带6保持原样。

如果你手中的磁带不止6盘，可用多盘磁带来保存完全备份。每次执行完全备份时，都采用最早的那盘。这样，就能够有数周以前的完全备份，从而方便你找出原来的、现已删除了的文件或某文件以前的版本。

9.4.1 如何利用tar进行备份

利用tar，可轻松地进行完全备份：

```
# tar -create -file /dev/ftape /usr/src
tar: Removing leading / from absolute path names in the archive
#
```

上面的例子中，采用了tar的GNU版本及其长选项名。过去的tar版本只能识别单一字符选项。对于一盘磁带或软盘装不下、而且带有超长路径名的备份文件， GNU版本也是能够处理的；但并非所有的版本都能如此（Linux只用GNU tar）。

如果一盘磁带装不下你的备份，你就需要用多卷（-M）选项：

```
# tar -cMf /dev/fd0H1440 /usr/src
tar: Removing leading / from adsolute path names in the archive
Prepare volume \#2 for /dev/fd0H1440 and hit return:
#
```

注意，在开始备份之前，应该格式化软盘，否则就用另一个窗口或虚拟终端，在 tar要求新软盘的时候，开始备份。

做完备份之后，应该利用-compare(d)选项来检查备份是否正确：

```
# tar -compare -verbose -f /dev/ftape
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
....
#
```

不检查备份意味着你可能在已经丢失原始数据之后，才发现自己的备份有误。

新增备份利用-newer(-N)选项，通过tar来执行的：

```
# tar -create -newer '8 Sep 1995' -file /dev/ftape /usr/src -verbose
tar: Removing leading / from absolute path names in the archive
usr/src/
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/modules/
usr/src/linux-1.2.10-includes/include/asm-generic/
usr/src/linux-1.2.10-includes/include/asm-i386/
usr/src/linux-1.2.10-includes/include/asm-mips/
usr/src/linux-1.2.10-includes/include/asm-alpha/
usr/src/linux-1.2.10-includes/include/asm-m68k/
usr/src/linux-1.2.10-includes/include/asm-sparc/
usr/src/patch-1.2.11.gz
#
```

不幸的是，tar注意不到文件的inode信息是何时更改的，比如，文件的访问许可部分或文件名是何时更改的。出现类似情况时，解决办法是用 find，并把当前文件系统状态和以前备份过的文件列表进行对比。Linux ftp站点上可找到所需的脚本和程序。

9.4.2 如何利用tar恢复文件

tar抽取文件所用的选项是-extract(-X)：

```
# tar -extract -same-permissions -verbose -file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

还可通过在命令行指定的方式，抽取特定的文件或目录（其中包含所有文件和子目录）：

```
# tar xpvf /dev/fd0H1440 usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
#
```

如果你只想看备份卷上有些什么文件的话，可用 -list(-t)选项：

```
# tar -list -file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

注意，tar始终依序读取备份卷，所以对于大型卷来说，读取速度是非常慢的。但是，在

使用磁带机或其他媒体时，也不可能采用随机访问数据库的方式。

tar不能正确处理已删除的文件。如果你需要从一个完全和新增备份恢复一个文件系统，而且你已经在两次备份之间删掉了一个文件，那么在完成恢复之后，这个文件又出现了。如果这个文件中是一些不再可得的敏感数据的话，问题就严重了。

9.5 多级备份

前一小节中解释的简单备份通常适用于个人或小型网站。对于机构或大型网站来说，则需要采用多级备份。

简单备份共有两个备份级别：完全备份和新增备份。笼统地讲，还可分为更多的备份级别。完全备份是1级，新增备份的级别分别是1、2、3等。每个新增级别上，都可备份自同级或上一级的上次备份以来的所有变动。

采用多级备份的目的是能够更便宜地保持较长的备份历史。前一小节的示例中，备份历史又回到了前一次完全备份。这是可以通过多盘磁带进行扩展的，但一周用一盘新磁带，可能代价太高了。拥有较长的备份历史是很有用的，因为被删除或损毁的文件通常很长时间都不能被察觉。即使某文件的版本不是很新，但总比根本没有强。

利用多级备份备份历史能够在少花钱的情况下，得以轻松扩展。举个例子来说，如果我们有10盘磁带，就可以把磁带1和2用于月备份（每月的第一个星期五），磁带3到6用于周备份（每月除开第一个星期五的星期五；注意，有时一个月可能有 5个星期五，所有可能还需要4盘磁带），磁带7到10用于日备份（星期一到星期四）。只用4盘磁带，我们已能够把备份历史从两周（所有的日备份磁带用完之后）扩展到了两个月。我们不能恢复每个文件在两个月内的每个版本，但我们能够选择最完整的版本进行恢复，这是假不了的。

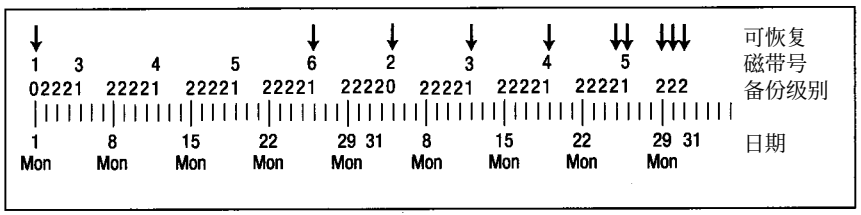

图9-1 多级备份日程表示例

图9-1展示了每天所用的备份级别，以及月末时，可从哪些备份中恢复。

备份级别还可用于使文件系统的恢复时间缩短至最少。如果你有许多新增备份，这些备份的级别号是单向递增的，那么，在你需要重建整个文件系统时，就需要一一恢复所有的新增备份。如果你采用的备份级别不是单向递增的（即是无序的），就可以减少备份数。

为了最小化恢复数据所需的磁带数，可为每个新增磁带采用一个较小的备份级别。但是，执行备份的时间将会增加（每个备份都要复制自前一次完全备份以来的所有更改过的数据。因此，dump手册页建议采用另一种较好的方法，如表 9-1所示（有效的备份级别）。利用其中的备份级别：3、2、5、4、7、6、9、8、9，以此类推。这样一来，就可将备份和恢复所需要的时间缩至最短，只须你执行两天一次的备份工作。恢复所用的磁带数取决于完全备份之间的时间，但它少于简单备份所需的磁带数。

表9-1 利用多级备份的有效备份方案

磁带编号	备份级别	备份（日期）	恢复磁带
1	0	n/a	1
2	3	1	1, 2
3	2	2	1, 3
4	5	1	1, 2, 4
5	4	2	1, 2, 5
6	7	1	1, 2, 5, 6
7	6	2	1, 2, 5, 7
8	9	1	1, 2, 5, 7, 8
9	8	2	1, 3, 5, 7, 9
10	9	1	1, 2, 5, 7, 9, 10
11	9	1	1, 2, 5, 7, 9, 10, 11
...	9	1	1, 2, 5, 7, 9, 10, 11

新的解决之道可能会减少你的工作负担，但意味着你还需要多了解其他的一些东西。究竟采用哪个方案，必须自己拿主意。

dump本身支持备份级别。而对tar和cpio来说，必须通过外壳脚本来实现。

9.6 要备份什么

任何情况下，人们都希望尽可能地保留备份。例外之一：可以轻松重新安装的软件（得了解一下“轻松”一词的含义；有人把利用一打软盘来安装软件也称为轻松），即使备份配置文件非常重要，只需重新配置即可。另一个例外是 /proc文件系统；由于只有它才包含了始终由内核自动生成的数据，要对它进行备份绝不值得尝试。特别是 /proc/kcore文件，最没有必要备份，因为对你而言，它只是当前物理内存的一个镜像而已；它的数量相当大。

灰色区内包含的是新闻假脱机、日志文件和 /var内的其他一些东西。你必须自己决定哪些是最重要的。

很显然，需要备份的东西是用户文件（ /home）和系统配置文件（ /etc，但可能还有散布在文件系统各处的其他文件）。

9.7 压缩备份

备份会占用大量的空间，可能会花很多钱。为了减少所需空间，节省开支，所以要对备份进行压缩。常见的方式有许多。有的程序内置压缩支持；比如 GNU tar的-gzip（-z）选项，它在把备份写入备份媒体之前，通过 gzip压缩程序，把整个备份压缩在一起。

不幸的是，压缩备份通常会带来许多不便。由于压缩的本性，如果其中有丁点错误，其他的压缩数据都将不能用了。有的备份程序有内置的纠错功能，但面对大量的错误，也是无法解决的。这意味着如果备份是像 GNU tar那样压缩的，也就是说，把整个备份压缩成一个整体，其中有一个错误的话，将导致其他所有备份数据的丢失。由于备份必须可靠，因此，这种压缩方式不见得是上上之策。

另一种方法是对每个文件进行单独压缩。这意味着其中一个文件丢失，其他的文件却不会受损。丢失的文件总之已经受损，所以只能说，这种方法比没有使用压缩好一点。 afio程序

（cpio的变体）也用于此。

压缩会花些时间，这样一来，会令备份程序不能以很快的速度，把数据写入磁带驱动器（如果磁带驱动器不能连续收到数据，它就必须停下来等待数据的写入；这样一来，备份程序的速度甚而更慢，停停走走的状态对磁带和驱动器来说，都不是件好事）。但这种情况可以通过缓冲备份输出的方式得以避免，但同样收效甚微。不过，这种情况一般发生在较慢的机器上。

第三部分　附　　录

附录A 词汇表

本词汇表简要列出了和Linux系统和系统管理相关的几个概念及说明。

ambition：佳句

写下一些有趣的句子，希望它们能够进入Linux的Cookie文件。

application program：应用程序

用于真正“干活儿”的软件。之所以买计算机，其目的就是为了运行应用程序。参考“系统程序”和“操作系统”。

daemon：后台程序

潜伏于后台的一个进程（通常不为人所知），只有在符合特定条件时，才会触发，进入前台运行。例如，\cmd{update}这个后台程序每隔30s就运行一次，清空缓冲区。另外，\cmd{sendmail}这个后台程序会在有人发送邮件时，立即启动。

filesystem：文件系统

操作系统用于对磁盘或分区上的文件进行跟踪的一种方法和数据结构：换言之，它是文件在磁盘上进行组织的一种方法。分区或磁盘利用文件系统，以文件的形式来保存数据。

glossary：字典

一系列单字（words），以及解释其用途的一系列文字。注意不要与“辞典”（Dictionary）混淆。因为后者也是一系列词与解释的组合。

kernel：内核

操作系统的一部分，用于实现与硬件的沟通，以及实现资源的共享。同时参考“系统程序”。

operating system：操作系统

和用户及其运行的应用程序一起，共享计算机操作系统资源（处理器、磁盘空间、网络带宽等）的软件。控制对系统的访问以便为系统提供安全保障。参见“内核”、“系统程序”和“应用程序”。

system call：系统调用

由内核为应用程序提供的服务，以及这些服务的调用方式。参见手册页的第二部分。

system program：系统程序

实施操作系统高级功能的程序，例如不直接依赖于硬件的程序。这类程序有时可能需要特殊权限才能运行，但通常情况下，它们都被视为系统的一部分（比如编辑器）。参见“应用程序”、“内核”和“操作系统”。

附录B　Linux文档项目复制许可证

1997年1月6日最后一次修改

下列版权许可证适用于“ Linux文档项目 ”(LDP)的所有作品。

请仔细阅读本许可证—它是类似于 GNU通用公共许可证的东西，但有几个条件不同于你习惯的内容。如果你有任何问题，请往 mdw @ metalab.unc.edu发Emeirl和LDP的负责人联系。

“ Linux文档项目 ”手册的部分或全部内容，依据下列条件，可以自由复制与传播。

所有“ Linux文档项目 ”手册由其作者享有版本，它们不在公共范围之内。

■ 在全部或部分副本中，上述版权声明及本使用许可证必须保持完整；

■ 任何翻译或衍生性的工作，在发布前，必须先征得原作者同意；

■ 如果以部分形式发布这份手册，必须明确提醒读者可取得这份手册的完整版本，并同时说明取得完整版本的方式；

■ 在其他作品中，可引用或评论本手册的一小部分内容。只要保证了引文的完整性，便可不必遵守这里的使用许可证；

针对学术性目的，这些规则可有所例外。详情请致函作者询问。这里的限制只用来保护原作者的权益，并不是用来对学员及教育工作者加以限制。

本文档包含的所有源码都受到GNU“ 通用公共许可证 ”（GPL）的约束。该许可证可通过匿名FTP服务器下载，地址是：ftp://prep.ai.mit.edu/pub/gnu/COPYING。

出版LDP手册

如果您代表的是一家出版社，有兴趣出版任何 LDP手册，请继续向下阅读。

以上面给出的使用许可证为前提，任何人都可出版和发行 Linux文档项目手册的原文拷贝(副本)。如果要做的是这件事情，那么不必事先征得我们的同意。然而，假如您想出版的是翻译稿，或者在原LDP手册的基础上，进行任何衍生性的工作，比如内容修订等等，那么只要该手册的“ 使用许可证 ”(License）对这方面进行了明确规定，那么您事先需要征得原作者的同意。

当然，您可以以赢利为目的，来销售 LDP手册。我们也喜欢您这样做。但要注意的是，由于LDP手册本身是免费发行的，所以任何人都应能免费影印或传播它的拷贝。

对于通过销售LDP手册带来的任何赢利，我们不要求您支付版税。但是，如果您以赢利为目的而销售LDP手册，我们建议您既可支付原作者一定的版税，亦可将收入的一部分捐赠或回报给作者、LDP或者整个Linux开发团体。最好能将出版的一本或几本 LDP手册寄送给原作者以供参考。对于您对LDP和Linux社区的支持，我们致以衷心感谢！

我们希望收取 LDP手册的任何出版或发行计划，以便我们知道它们的传播情况。如果您已经或计划出版任何 LDP手册，请向ldp-l@linux.org.au发一封信。对于您的工作，我们将非常感谢。

我们鼓励Linux的软件发行商随软件一道，发布 LDP手册（比如“ 安装和新手指南 ”这样的东西)。LDP手册有志于成为“ 正式 ”的 Linux文档。我们很高兴地看到，许多邮购分销商已经在软件中配套提供了 LDP手册。LDP手册的成熟，需要众多 Linux爱好者的共同参与。在此说声：“ 谢谢大家！ ”

附录C　GNU通用公共许可证

在本书最后，我们附上这份 GNU通用公共许可证 (GPL),便于您参考。本书讲到的软件，全部是以它为基础发布给公众的。但要注意的是，本书的正文，却不受此 GPL许可证的约束。

前言

大多数软件许可证的目的是限制你共享和修改软件的自由。相反， GNU通用公共许可证则旨在保护你共享和修改软件的自由——即保证软件对所有用户都是自由的。通用公共许可证（GPL）适用于大多数自由软件基金会的软件，以及由使用这些软件而承担义务的作者所开发的软件。（有些自由软件基金会的软件受到 GNU库通用公共许可证的保护。）你也可以将它用到你的程序中。

当我们谈到自由软件时，我们所指的是自由而不是价格。我们的通用公共许可证的目的是保证你有分发自由软件副本的自由（如果愿意的话，你可以对这个服务收费）；保证你能得到源程序代码或者在你需要的时候能得到它，保证你能够修改软件或者将其用到新的自由软件中，保证你知道你能做这些事情。

为保护你的权利，我们规定禁止任何人不承认你的权利或者要求你放弃这些权利。如果你分发了软件的副本或者修改了软件，这些规定就转化为你的责任。

比如说，你分发了这样一个程序的副本，不管是收费的还是免费的，你必须将你具有的一切权利给予这些接受者，必须保证他们收到或能得到源程序，而且必须让他们知道自己所拥有的权利。

我们采取两项措施来保护你的权利： 1) 给软件以版权保护， 2) 给你提供许可证。它给你复制、分发和修改软件提供法律许可。

另外，为了保护每个作者和我们自己，我们需要清楚的让每个人明白，自由软件不承担担保。如果由于其他人修改了软件并继续传播，我们需要它的接受者明白：他们所得到的并不是原来的自由软件。由其他人引入的任何问题，不应损害原作者的声誉。

最后，任何自由软件经常受到软件专利的威胁。我们希望避免这样一种风险：自由软件的再发布者以个人名义获得专利许可证，从而将软件变为私有。为防止这一点，我们必须明确：任何专利必须以允许每个人自由使用为前提，否则就不准拥有专利。

下面是有关复制、发布和修改自由软件的确切条款和条件。

关于复制、分发和修改的条款和条件

这个许可证适用于任何包含版权所有者声明的程序和其他作品，版权所有者在声明中明

确说明程序和作品可以在 GPL条款的约束下发布。下面提到的“ 程序 ”指的是任何这样的程序或作品。“ 基于程序的作品 ”指的是程序或者任何受版权约束的衍生作品。衍生作品是指包含程序或程序一部分的作品，其包含形式可以是原封不动的，也可以是经过修改的 /翻译成其他语言的。（下文中，翻译包含在没有附加限制的“ 修改 ”条款中。）每个领到许可证的人将用“ 你 ”来称呼。

许可证条款不适用复制、发布和修改以外的活动，他们超过了许可证条款的范围。运行程序的活动不受条款的限制。仅当程序的输出构成基于程序作品的内容时，这一条款才适用（如果只运行程序就无关）。是否普遍适用取决于程序具体用来作什么。

1) 只要你在每一副本上明显和恰当地表达版权声明和不承担保证的声明，保持此许可证的声明和没有保证的声明完整无损，并和程序一起给每个其他的程序接受者一份本许可证的副本，你就可以用任何媒体复制和发布你收到的原始的程序的源代码。

你可以为转让副本的实际行动收取一定费用。你也有权选择提供保证以换取一定的费用。

2) 你可以修改程序的一个或几个副本或程序的任何部分，以此形成基于程序的作品。只要你同时满足下面的所有条件，你就可以按前面第一款的要求复制和发布这一经过修改的程序或作品。

a) 你必须在修改的文件中附有明确的说明：你修改了这一文件及具体的修改日期。

b) 你必须使你发布或出版的作品（它包含程序的全部或一部分，或包含由程序的全部或部分衍生的作品）允许第三方作为整体按许可证条款免费使用。

c) 如果修改的程序在运行时以交互方式读取命令，你必须使它在开始进入常规的交互使用方式时打印或显示声明：包括适当的版权声明和没有保证的声明（或者你提供保证的声明）；用户可以按照此许可证条款重新发布程序的说明；并告诉用户如何看到这一许可证的副本。（例外的情况：如果原始程序以交互方式工作，它并不打印这样的声明，你的基于程序的作品也就不用打印声明）。

这些要求使用于修改了的作品的整体。如果能够确定作品的一部分并非程序的衍生产品，可以合理地认为这部分是独立的，是不同的作品。当你将它作为独立作品发布时，它不受此许可证和它的条款的约束。但是当你将这部分作为基于程序的作品的一部分发布时，作为整体它将受到许可证条款约束。准予其他许可证持有人的使用范围扩大到整个产品，也就是每个部分，不管它是谁写的。

因此，本条款的意图不在于要求或争夺对全部由你写成的作品的权利；而是履行权利来控制基于程序的集体作品或衍生作品的发布。

此外，仅将另一个不是基于程序的作品与程序（或与基于程序的作品）一起放在存储体或发布媒体的同一卷上，并不导致将那个作品置于此许可证的约束范围之内。

3) 你可以以目标码或可执行形式复制或发布程序（或符合第 2款的基于程序的作品），只要你遵守前面的第1、2款，并同时满足下列3条中的一条。

a) 在通常用作软件交换的媒体上，随带其相应的、机器可读的、完整的源代码。这些源代码的发布应符合上面第 1、2款的要求。或者

b) 在通常用作软件交换的媒体上，随带一份为第三方提供响应的机器可读的源代码的书面报价。其有效期不少于 3 年，费用不超过实际完成源程序发布的实际成本。源代码的发布这些源代码的发布应符合上面第 1、2款的要求。或者

c) 随带你收到的发布源代码的报价信息。(这一条款只使用于非商业性发布，而且你只收到程序的目标码或可执行代码和按 2）款要求提供的报价)。

作品的源代码指的是对作品进行修改时最优先择取的形式。对可执行的作品来说，完整的源代码包括：所有模块的所有源程序，加上有关的接口定义文件，控制可执行作品的安装和编译的脚本。作为特殊例外，发布的源代码不必包含任何常规发布的供可执行代码在上面运行的操作系统的主要组成部分（如编译程序、内核等），除非这些组成部分和可执行作品结合在一起。

4) 除非你明确按照许可证提出的要求去做，否则你不能复制、修改、转发许可证和发布程序。任何试图用其他方式复制、修改、转发许可证和发布程序是无效的。而且将自动结束许可证赋予你的权利。然而，对那些从你那里按许可证条款得到副本和权利的人们，只要他们继续全面履行条款，许可证赋予他们的权利依然有效。

5) 你还没有在本许可证上签字，因而你没有必要一定要接受它。然而，没有任何其他东西赋予你修改和发布程序及其衍生作品的权利。如果你不接受本许可证，这些行为是法律禁止的。因此，如果你修改或发布程序（或任何基于程序的作品），你就表明你接受这一许可证以及它的所有有关复制、发布和修改程序或基于程序的作品的条款和条件。

6) 每当你重新发布程序（或任何基于程序的作品）时，接受者自动从原始许可证颁发者那里接到受这些条款和条件支配的复制、发布或修改程序的许可证。你不可以对接受者履行这里赋予他们的权利强加其他限制。你也没有强求第三方履行许可证条款的义务。

7) 如果作为法院的判决或专利的侵权争议或任何其他原因（不限于专利问题）的结果，对你强加了一些与本许可证的条件有冲突的条件，它们也不能开脱本许可证条款对你的约束。如果你不能同时满足本许可证规定的义务和其他相关的义务，那么你可以根本不发布程序。例如：如果某一专利许可证不允许所有那些直接或间接从你那里接受副本的人们在不付费的情况下重新发布程序，唯一能同时满足两方面要求的办法是停止发布程序。

如果本条款的任何部分在特定的环境下无效或无法实施，就使用条款的剩余部分 ,并将条款作为整体用于其他环境。

本条款的目的不在于引诱你侵犯专利或其他财产权的要求，或争论这种要求的有效性。本条款的主要目的在于保护自由软件发布系统的完整性。它是通过通用公共许可证的应用来实现的。许多人坚持应用这一系统，已经为通过这一系统发布大量自由软件作出慷慨的贡献。作者／捐献者有权决定他／她是否通过任何其他系统发布软件。许可证持有人不能强制这种选择。

本节的目的在于明确说明许可证其余部分可能产生的结果。

8) 如果由于专利或者由于有版权的接口问题使程序在某些国家的发布和使用受到限制，将此程序置于许可证约束下的原始版权拥有者可以增加限制发布地区的条款，将这些国家明确排除在外,而在这些国家以外的地区发布程序。在这种情况下，许可证包含的限制条款和许可证正文一样有效。

9) 自由软件基金会可能通过随时发布通用公共许可证的修改版或新版。新版和当前的版本在原则上保持一致，但在提到新问题时或有关事项时，在细节上可能出现差别。

每一版本都有不同的版本号。如果程序指定使用于它的许可证版本号以及“ 任何更新的版本 ”，你有权选择指定的版本或自由软件基金会以后出版的新版本；如果程序未指定许可证

版本，你可选择自由软件基金会已经出版的任何版本。

10) 如果你愿意将程序的一部分结合到其他自由程序中，而它们的发布条件不同，写信给作者，要求准予使用。如果是自由软件基金会加以版权保护的软件，写信给自由软件基金会，我们有时会作为例外的情况处理。我们的决定受两个主要目标的指导，这两个主要目标是：我们的自由软件的衍生作品继续保持自由状态；以及从整体上促进软件的共享和重复利用。

不保证

11) 由于程序准予免费使用，在适用法准许的范围内，对程序没有保证。除非另有书面说明，版权所有者和／或其他提供程序的人们“一样”不提供任何类型的保证，不论是明确的，还是隐含的，包括但不限于隐含的适销和使用特定用途的保证。全部的风险，如程序的质量和性能问题都由你来承担。如果程序出现缺陷，你承担所有必要的服务、修复和改正的费用。

12) 除非受适用法的要求或以书面形式达成协议，在任何情况下，任何版权所有者或任何按许可证条款修改和发布程序的人们都不对你的损失负有责任。包括由于使用或不能使用程序引起的任何一般的、特殊的、偶然发生的或重大的损失（包括但不限于数据的损失，或者数据变的不精确，或者你或第三方的持续的损失，或者程序不能与其他程序协调运行等），即使就这种损失的可能性咨询过版权所有者和其他人也不例外。

如何将条款使用于你的新程序

如果你开发了一个新程序，并且希望它得到公众最大限度的利用，达到这一点的最好方法是将它变为自由软件，每个人都能在遵守条款的基础上对它进行修改和重新发布。

为了作到这一点，给程序附上下面的版权声明。最安全的方式是将其放在每个源程序的开头，以便最有效地传递专有权保证信息。每个文件至少应有“版权所有”行以及指出版权说明放在何处。

<用一行空间给出程序的名称和它的用处的简单声明 >

版权所有 (c) 19XX<作者姓名>

这一程序是自由软件,你可以遵照自由软件基金会发布的 GNU通用公共许可证条款 来修改和重新发布这一程序 ,即可参照该许可证的第二版,也可参照(根据你的选择)任何更新的版本进行。

发布这一程序的目的 是希望它有用,但不提供任何保证,甚至没有适销或适合特定目的的隐含的保证。更详细的情况请参阅 GNU通用公共许可证。

你应该已经与程序一起收到一份 GNU通用公共许可证的副本，如果还没有，写信给 The Free Software Foundation,Inc. ,59 Temple Place,Suite 330,Boston,MA02111-1307 USA

还应加上如何用电子邮件和书面邮件与你联系的信息。

如果程序以交互方式进行工作，当它开始进入交互方式工作时，使它输出类似下面的简短声明：

Gnomovision 第69版，版权所有 （C）19XX 作者姓名

Gnomovision 绝对没有保证。要知道详细情况，请输入 show w

这是自由软件，欢迎你遵守一定的条件重新发布它，要知道详细请况，请输入 show c

假设的命令“show c”和 “show w”应显示通用公共许可证的相应条款。当然，你使用

的命令名称可以不同于“ show c ”和 “ show w ”。根据你的程序的具体情况，也可以用菜单或鼠标选项来显示这些条款。

如果需要，你应该取得你的雇主（如果你是程序员）或你的学校签署放弃程序版权的声明。下面是一个例子，你应该改变相应的名称：

Yoyodyne 公司在此放弃James Harker所写的Gnomovision程序的全部版权利益。

<Ty Coon 的签名>,1989月4日1 Ty Coon，副总裁

这一许可证不允许你将程序并入专用程序。如果你的程序是一个子程序库，可以考虑把库链入专用应用程序中。如果这是你想做的事，使用 GNU Library通用公共许可证代替本许可证。