LINUX
网络管理员手册

Olaf Kirch 著
赵炯 译

gohigh@sh163.net

Linux 网络管理员手册

The Linux Network Administrator's Guide

Olaf Kirch 著

赵炯 译

www.plinux.org

gohigh@sh163.net

(gohigh@shtdu.edu.cn)

1999/8/30

(草)

The Linux Network Administrator’s Guide

为 Brilla

Legal Notice

UNIX is a trademark of Univel.
Linux is not a trademark, and has no connection to UNIX ™ or Univel.

Kattreinstr. 38, 64295 Darmstadt, Germany
okir@monad.swb.de

目录

图表列表

第一章 网络绪论

1.1 历史

连网的主意大概与电讯事业本身一样的久远。考虑人们生活的石器时代，那时鼓可能已经用于在人们之间传递消息了。假释穴居人A想邀请穴居人B进行一场互扔石块的游戏，但是他们互相之间居住得太遥远，以至B听不见A的击鼓声。那么，A能够做些什么呢？他可以1）走到B的地方去，2）使用一个更大的鼓，或者3）询问C，C居住在他们中间的地方，来传递消息。这最后一个办法就叫做连网。

当然，我们已经比我们祖先的原始嗜好和设备有了长足的进步。现在，通过大量的线缆的集合，如光纤、微波、等等，我们用计算机进行相互间的对话来作星期六足球比赛的约会。[1] 下面，我们将涉及实现上述的手段和方法，但不考虑线缆，也不考虑足球的部分。

在本手册中，我们将讨论两种网络类型：基于UUCP的，和基于TCP/IP的。这是一些协议集和软件包，它们提供了在两台计算机之间传输数据的方法。在这一章中，我们将考虑这两种网络类型，并且讨论它们的基本原理。

我们定义网络是一个互相之间能够通信主机（*hosts*）的集合，这常常依赖于一些专用（指定）主机的服务，即是在参与者之间中继数据。主机通常就是计算机，但也不一定；也可以将X-终端或者智能打印机当作主机。少量的主机聚合也称为站点（*sites*）。

没有语言或者代码，进行通信是不可能的。在计算机网络中，这些语言共同地被称为协议（*protocols*）。然而，这里你不用关心协议是如何制定出来的，而是要，例如，考虑州长开会时注意到的高度形式化的行为代码。同样，用于计算机网络中的协议仅仅是两台或多台主机之间交换消息所用的非常严格的一些规则。

1.2 UUCP 网络

UUCP是Unix-to-Unix Copy的缩写。刚开始它是作为一个程序包，用于在串行线路上传输文件、确定这些传输的时间、并且在远程站点上启动程序的执行。自从七十年代末它第一次实现以来，已经历了很大的变化，但其提供的服务仍然很简单。他的主要应用仍然是在基于拨号连接的广域网中。

UUCP是贝尔实验室在1977年首先开发出来的，用于在他们的Unix开发站点之间的通信。在1978年中期，这个网络已经连接了80多个站点。它应用于运行电子邮件以及远程打印。然而，这个系统主要用于分发新软件以及调试程序。[2] 现今，UUCP不再被限制于这个环境内了。在许多种类的平台上已有免费的和商业的移植版本了，包括AmigaOS、DOS、Atari的TOS等等。

UUCP网络的主要缺点之一是它的低带宽。一方面，电话设备对最高传输速率有严密的限制。另一方面，UUCP链接很少有固定的连接；而是在有规则的时间间隔上主机拨号来相互连接。因此，大多数时间，是用于在UUCP网络上传输存储于某些主机磁盘上的邮件消息、等待下次连接的建立。

尽管有这些限制，世界各地仍有许多UUCP网络在运转着，主要是由计算机业余爱好者在运行着，它以合适的价格为私人用户提供网络访问。UUCP流行的主要原因是：与将你的计算机连接到大Internet电缆上相比，它是极其便宜的。为了使你的计算机成为一个UUCP节点，你只须有个modem、一个运行着的UUCP程序，以及愿意供你邮件和新闻的其他的UUCP节点。

1.2.1 如何使用 UUCP

UUCP后面的概念是非常简单的：就如同它的名字指出的一样，它基本上是将文件从一台主机上拷贝到另一台上去，但它也允许在远程主机上进行一定的操作。

假设你的机器允许访问名为swim的假想的主机，并且让它为你执行lpr打印命令。那么你可以命令行上键入下面一行在swim上打印出本书来：[3]

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

这使得uux为swim调度了一个作业（*job*）。uux是UUCP组中的一个命令。这个作业由输入文件netguide.dvi、以及馈送该文件到lpr的请求组成。-r标志告诉uux不用立刻访问远程系统，而是将作业存储起来直到稍后时有个连接被建立起来。这叫作假脱机（打印）（*spooling*）。

UUCP的另一个特性是它允许通过几台主机转发作业和文件，假如它们合作的话。假定上面例子中的swim与groucho有一个UUCP链接，groucho中保存着大量的应用程序文档。为了下载文件tripwire-1.0.tar.gz到你的站点上，你可以发出

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

所创建的作业将请求swim从groucho取得该文件，并将文件送到你的站点，这里UUCP将把文件存为trip.tgz并且通过文件到达的邮件来通知你。这将分三步完成。首先，你的站点将作业送至swim。当下次swim与groucho建立了连接，就会下载该文件。最后一步是从swim到你站点的实际的传输。

目前，UUCP网络所提供的最重要的服务就是电子邮件和新闻。稍后我们将会讨论这些，所以这里我们仅给出一个概要的介绍。

电子邮件 – 简称email – 允许你与远程主机上的用户交换消息而无需实际地知道如何访问这些主机。控制一个消息从你的站点到达目的站点的任务是完全由邮件处理系统完成的。在一个UUCP环境中，邮件一般是通过在比邻的主机上执行rmail命令传送的，并把接收者的地址和邮件消息传给rmail。然后rmail将会转发消息到另一台主机上等等，直到消息到达目的主机为止。我们将在第13章中详细地讨论。

News可以最恰当地描述成一类分布式的电子公告板系统。绝大多数情况下，这个术语指的是Usenet News，它是直到目前为止最著名的估计有着120,000-参与站点的新闻交换网络。Usenet的起源可追溯至1979年，那时，在新的Unix-V7版本发布以后，三个研究生有了一个在Unix团体中通用信息交换的点子。他们整理了一些脚本，这成了第一个网络新闻系统。在1980年里，这个网络连接了北卡罗林纳州的两所大学里的duke、unc和phs网络。从这衍生出来，Usenet最终成长起来了。尽管它起初是一个基于UUCP的网络，现已不再限于单种类型的网络了。

信息的基本单元是文章，它可能被投寄到专用于某个特殊主题的新闻组的层次结构中。大多数站点仅仅接收全部新闻组的一个选集，而全部新闻组每天平均传送相当于60MB的文章。

在UUCP的世界中，news通常是按照从请求的组中收集所有的文章，并且打包成几批（*batches*），再通过一个UUCP链接来发送的。这几批文章被发送到接收站点，并在那里被送给了rnews命令来打开这几批数据包以及更进一步的处理。

最后，对于许多拨号上网的供公共访问的文档站点来说，UUCP也同样是一种供选择的方法。你通常可以这样来使用它们：使用UUCP拨号上网、作为来客（guest）用户登录、并从公共访问文档

区域下载文件。这些来客帐号的登录名/口令通常是uucp/nuucp或者是其他一些类似的。

1.3 TCP/IP 网络

尽管UUCP可能是一种廉价的拨号上网链接的选择，但还存在着许多情况，在这些情况下，这种存储-与-转发的技术被证明是不灵活的，例如在局域网（LANs）的情况下。这通常是由位于同一幢建筑物、甚至是位于同一层的少数机器组成，它们互相连接以提供一个相似的工作环境。典型地来讲，你将在这些主机上共享文件，或者在不同的机器上运行分布式应用程序。

这些任务需要一种完全不同的连网途径。所有的数据被分解成更小的块（packets 包、分组），这些块被立刻转发到目的主机上，并再重新组合起来。而不是随同一个作业脚本转发整个文件。这类网络被称为*包[分组]交换（packet-switched）*网络。从其它方面来讲，这允许在网络上运行交互式的应用程序。当然，所付出的代价是大大地增加了软件的复杂性。

这种系统---不一定就是主机---所采用的解决方案就是著名的TCP/IP。在本节中，我们将看一下它的基本概念。

1.3.1 TCP/IP 网络引言

TCP/IP的起源可以上溯到1975年美国DARPA（Defense Advanced Research Projects Agency 国防部远景规划局）支助的研究计划。那是个试验性的网络，ARPANET，在经证实成功以后，于1975年转入正常运行。

1983年，新的TCP/IP协议组被作为标准采用，并且网络中的所有主机必须使用它。当ARPANET最终成长为Internet时（ARPANET本身于1990年停止使用），TCP/IP的使用已经传播到Internet以外的网络上去了。最值得注意的是局域网络，但随着象ISDN等快速数字电话设备的出现，将来肯定也会应用于拨号上网的传输。

在贯穿以下几节的TCP/IP讨论当中，作为一个看待的具体实例，我们将考虑坐落在Fredland某地的Groucho Marx大学（GMU），绝大多数系部运行他们自己的局域网，有些系部共享一个局域网，有些有好几个局域网络。他们都是互连的，并且通过一个高速链接与Internet相连。

假设你的机器在数学系连接到局域网上，并且假设你的机器名为erdos。为了访问物理系的一个名为guark的主机，你键入以下命令：

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

在提示符下，输入你的登录名，假如说是andres，以及你的口令。这时你就进入了quark的shell，在那里你就可以象是在系统的控制台上一样键入各种命令。在你退出shell以后，就会退回到自己机器的提示符下。你刚才即使用了TCP/IP提供的即时的、交互式的应用程序：远程登录。

在你登录进quark时，你也会想要运行一个基于X11的应用程序，就象函数绘图程序，或一个PostScript预览器。为了让这个程序知道你想要把窗口显示在你的主机屏幕上，你就必须设置DISPLAY环境变量：

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

如果你现在运行你的应用程序，它将联系你的X服务器而不是quark的，并在你的屏幕上显示它的所有窗口。当然，这需要X11运行在erdos上。这里的要点是TCP/IP允许quark和erdos来回传送X11分组，并给了你这样一种幻觉，好象你只在单个系统上。在这里，网络几乎是透明的。

TCP/IP网络的另一个重要的应用程序是NFS，它代表*网络文件系统（Network File System）*。这是另一种透明使用网络的形式，因为它基本上允许你从其他主机上直接加载目录分层结构，所以它们就好象本地的文件系统一样。例如，所有用户的登录主目录可能在一台中心服务器机器上，局域网上所有其它主机都可以从它上面加载这个目录。这样做的效果是用户可以登录到任何机器上，并且会发现自己在同一个主目录中。类似地，仅在一台机器上安装需要大量磁盘空间的应用程序（比如象TeX），并且将这些目录导出到其它机器上。我们将在第十一章中再次讨论NFS。

当然，这些仅是在TCP/IP网络上你能做些什么的例子。这种可能性几乎是无限的。

我们现在进一步讨论TCP/IP工作的原理。你需要知道这些以理解如何和为什么要配置你的机器。我们将首先从分析硬件着手，然后慢慢地深入。

1.3.2 以太网

在局域网中广泛使用的硬件类型就是众所周知的*以太网（Ethernet）*。它由主机通过连接器与之连接的单根电缆、三通头或收发器组成。简易以太网安装起来一点不贵，而且其网络传输速率达到每秒10M，这也基本上说明了它为什么如此普及。

以太网可分成三类，分别称为*粗缆（thick）*、*细缆（thin）*和*双绞线（twisted pair）*。细缆和粗缆以太网都使用同轴电缆，只是粗细不同以及与主机连接的方式不同。细缆以太网使用一个T形的“BNC”连接头，两端连电缆，一头旋到计算机背面的一个插头上。粗缆以太网需要在电缆上钻一个小孔，并且使用一个“吸血鬼式的三通头”连接收发器。这样，一个或多个主机就能连接到收发器上。以太网的细缆和粗缆使用的长度最长分别可达200米和500米，因此它们也分别称为10base-2和10base-5。双绞线使用一根由两根铜线做成的线缆，也就是在普通电话装置中用的那种，但通常还需要另外一些硬件。它也以10base-T知名。

尽管将一台主机加入粗缆以太网有些烦琐，但不会断开网络。要往细缆以太网装置中增加一台主机的话，就不得不中断网络服务几分钟，因为需要切断电缆以接入一个连接头。

许多人偏爱细缆以太网，因为它非常便宜：PC网卡售价只有50美元左右，并且电缆每米也只有几美分。然而，对于大规模的安装，粗缆以太网则更适合。例如，GMU数学系的以太网使用的是粗缆，所以每当往网上增加一台主机时就不需要中断网络。

以太网技术的缺点之一是它有限的电缆长度，这妨碍了它在除局域网以外地方的使用。然而，几个以太网段可以使用中继器、桥接器或路由器来相互连接在一起。中继器只是简单地在两个或多个网段之间拷贝信号，所以，所有用其连接在一起的网段将表现出好象是一个以太网。由于时限的要求，在网络上的任何两台主机之间都不能多于四个网段。桥接器和路由器是很复杂的。它们分析输入的数据，并且仅在接收主机不在本地以太网上时才转发数据。

以太网的运作如同一个总线系统，在该总线上，一个主机可以发送最大长度可达1500字节的分组（或*帧 frames*）到同一以太网上的另一台主机中。主机是通过一个六字节地址寻址的，这个地址是固化在以太网板上的固件中的。这些地址通常是使用以冒号分开的两位十六进制数的序列形式写出的，就如同格式aa:bb:cc:dd:ee:ff。

某个站点发送的帧，别的所有站点都能看到，但只有目的主机会拾取并处理该帧。如果两个站点试图同时发送，就会发生*碰撞[冲突]（collision）*。这个问题解决的办法是两个站点都放弃发送，并在片刻以后再次尝试发送。

1.3.3 其它硬件类

在大规模的装备中，例如象Groucho Marx大学，以太网通常并不是唯一使用的设备。在Groucho Marx大学，每个系部的局域网都连接到校园主干网上，那是运行着FDDI（*光纤分布式数据接口 Fiber Distributed Data Interface*）的纤维光缆（fiber optics cable）。FDDI使用一种完全不同的方法来传输数据，它基本上包括向四周发送令牌，仅允许获得令牌的站点发送帧。FDDI的主要优点是有高达100Mbps传输速率、光缆最大长度可达200km。

对于长距离网络连接来讲，常常使用不同种类的设备，这些设备是基于名为X.25的标准的。许多所谓的公用数据网（Public Data Networks），如美国的Tymnet、德国的Datex-P就提供这种服务。X.25需要一种特殊的硬件，即分组组合/分拆（Packet Assembler/Disassembler）或*PAD*。X.25定义了一套自己的网络协议，然而它常常用于连接运行TCP/IP和其它协议的网络。由于IP分组不能简单地映射到X.25上（反之也然），所以IP分组只是简单地包装入X.25分组中在网上传输。

业余无线电爱好者常常使用他们自己的一套装置把计算机连成网；称为*分组无线网（packet radio）*或*业余无线电报务员无线网（ham radio）*。分组无线网所使用的协议称为AX.25，是从X.25派生出来的。

其它技术包括使用慢速但廉价的拨号访问串行线路。这些需要另外一些用于传输分组的协议，例如SLIP或PPP，这些将在下面讨论。

1.3.4 Internet 协议

当然，你不会希望你的网络连接仅限于一个以太网络之中。理想地来说，你希望能够使用一个网络而不管它是运行在什么硬件上的，也不管它是由多少个子单元组成的。例如，象Groucho Marx大学那样的大规模网络装置，你通常有几个用某些方法连接起来的独立的以太网络。在GMU，数学系有两个以太网：一个是教授和研究生的快速机器的网络，另一个是学生用的慢速机器的网络。这两个网络都连到了FDDI校园主干网上。

这种连接是由称为*网关（gateway）*的专用主机来处理的，它通过在两个以太网络以及光缆之间拷贝进入和传出的分组来处理的。例如，如果你在数学系并且想从你的机子上访问物理系局域网上的quark，那么网络连接软件不能够直接将分组发送到quark，因为不在同一个以太网内。因此，必须依靠网关作为一个转发者。然后，网关（给它取名为sophus）利用主干网将这些分组转发到物理系的niels网关上，由niels将分组传递到目的机器中。Erdos与quark之间的数据流见图1.1（对Guy L. Steele致歉）。

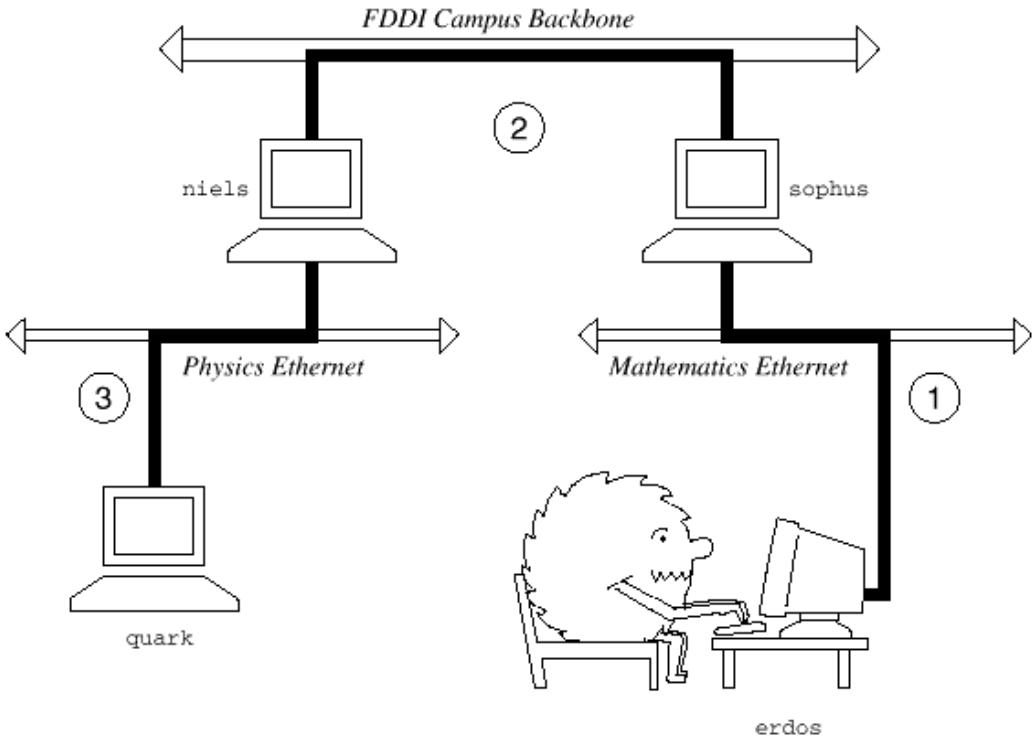

图 1.1 从 erdos 到 quark 发送一个数据报的三个步骤。

这种引导数据至远程主机的方案称为*路由选择或选路（routing）*，并且在此时常将分组称作*数据报（datagrams）*。为使事情简单，数据报交换是由单独一个与所用硬件无关的协议来管理的：IP，或*互连网协议（Internet Protocol）*。在第二章中，我们将极详细地描述IP以及路由选择的方法。

IP的主要好处在于它将物理上不同的网络转换成一个明显同类的网络。这叫做网络互连（internetworking），其所形成的“后网络”称为一个*互连网络（internet）*。在这里请注意*an* internet和*the* Internet的细微差别。后者是特指全球互连网的官方名称。

当然，IP也需要一个硬件独立的寻址方案。这是通过对每个主机分配一个32位的数值来完成的，这个32位数值称为*IP-地址（IP-address）*。一个IP地址通常用四个十进制数来表示，每一个十进制数表示一个8位部分，用点分开。例如，guark主机可能有一个IP地址是0x954C0C04，它可以写成149.76.12.4。这种格式也称为点分四组（*dotted quad*）表示法。

你将注意到我们现在有三种不同的地址类型：首先有象quark这样的主机名、然后我们有IP地址、最后还有硬件地址，如6字节的以太网地址。这些类型的地址要以某种方式加以匹配，这样，当你键入rlogin quark时，网络软件能够给出quark的IP地址；并且当IP发送任何数据到物理系的以太网上时，它会以某种方式找出与指定IP地址相映的以太网地址。这有些令人混淆。

这里我们不再深入讨论下去，而将在第二章中讨论之。现在已足够记住寻址的这些步骤的含义，对于将主机名映像到IP地址的操作成为*主机名解析（hostname resolution）*，对于将IP地址映像到硬件地址的操作称为*地址解析（address resolution）*。

1.3.5 串行线上的 IP

在串行线路上，常常使用一个以SLIP或串行线路IP而知名的“事实上的”标准。一个对SLIP的

修改被称为是CSLIP，或压缩的SLIP，用以进行IP头的压缩以充分利用串行线路提供的低带宽达到更好的性能。[4] 另一个不同的串行线路协议是PPP，或*点对点协议（Point-to-Point Protocol）*。PPP比SLIP有更多的特色，包括一个链接协商步骤。然而，它比SLIP出色的主要优点在于它不限于仅仅传输IP数据报，而是设计成能够传输任何类型的数据报的。

1.3.6 传输控制协议（Transmission Control Protocol）

当然，将数据报从一个主机传送到另一个主机并没有完，如果你登录进 quark，你就想在你的 erdos 上 rlogin 进程和 quark 上的 shell 进程之间有一个可靠的连接。这样，发送者就必须将发送和回应信息分解成分组，并且接收者将接收到的分组再重新组合成字符流。这看上去很琐细，但其中包括许多细微的任务。

关于 IP 要知道的一件重要的事情是，实际上，它不是可靠的。假设在你的以太网上有十个人开始从 GMU 的 FTP 服务器上下载 Xfree86 的最新版本。由此而产生的通信量对于网关处理能力来说可能太多，因为它太慢，并且内存也不多。如果你现在恰巧给 quark 发送了一个分组，网关 sophus 此时正好用完了缓冲空间因此没法转发此分组。IP 解决此问题的方法只是简单地丢弃了这个分组。这个分组就完全失去了。因此，检查数据的完整性，并且在遇到出错时重传数据是进行通信的主机的责任。

这是由另一个协议来完成的，即 TCP，或*传输控制协议（Transmission Control Protocol）*，该协议在 IP 之上建立了一个可靠的服务。TCP 的基本特性是，它利用 IP 给你一个在你的主机进程和远端主机进程之间一个简单连接的幻觉，这样你就不用关心你的数据是如何以及通过哪个路由器传输的。一个 TCP 的连接操作基本上象个双向的管道，两个进程都可以对其进行读写操作。可以把它想象成一次电话对话。

TCP 是通过两台相关主机的 IP 地址以及称为端口（port）的数值来识别一个连接的末（尾）端的。端口可以看作是网络连接的附加点。如果我们对电话的例子夸张一点讲，可以把 IP 地址比作地区码（对应城市的代码），把端口数值比作本地码（对应于某人的电话号码）。

在 rlogin 的例子中，客户应用程序（rlogin）在 erdos 上打开了一个端口，并且连接到 quark 上的端口 513 上，该端口是 rlogind 服务器一直在监听的。这样就建立了一个 TCP 连接。使用这个连接，rlogind 执行授权程序，并且然后产生 shell 进程。该 shell 的标准输入和输出被重定向至 TCP 连接上，所以在你的机器上你对 rlogin 所键入的任何字符将通过 TCP 流被当作标准输入给予 shell。

1.3.7 用户数据报协议（User Datagram Protocol）

当然，TCP 并不是 TCP/IP 连网中唯一的用户协议。尽管它适合象 rlogin 这样的应用程序，其所用的总开销阻止了它用于象 NFS 这样的应用程序中。取而代之的是，NFS 使用一个 TCP 的同属协议称为 UDP，或*用户数据报协议（User Datagram Protocol）*。正如 TCP 一样，UDP 也同样允许一个应用程序连接到远程机器上一个端口的服务上，但它不会为此建立一个连接。而是，你可以使用它来发送单个分组到目的服务上—正如它的名字表示的一样。

假设你将系中心的 NFS 服务器，galois，上的 TeX 目录结构加载到了你的机器上，并且你想查看描述如何使用 LaTeX 的文档。你启动了编辑器，该编辑器首先读入整个文件。然而，需要很长的时间与 galois 建立一个 TCP 连接、传送文件、并且再释放该连接。取而代之的是，向 galois 发送一个请求，galois 以几个 UDP 分组的形式发送出该文件，这是非常快的。然而，UDP 是不处理分组的

丢失与坏分组的。这要依靠应用程序—在这里是指 NFS—来处理的。

1.3.8 关于端口

端口可以看作是网络连接的附属件。如果一个应用程序想要提供一个服务，它就将自身附加到一个端口上并且等待客户的到来（这也称为在端口上*侦听 listening*）。想要使用该服务的客户在自己的本地主机上分配一个端口，并且将其连接到远程主机的该服务的端口上。

端口的一个重要特性是，一旦在客户和服务器之间建立了一个连接，服务器的另一个拷贝就会附加到服务器端口上并且侦听其它的客户。这准许，例如，在同一台主机上的几个并发远程登录操作，都使用端口 513。TCP 能够在这些连接之中加以区别，因为它们都来自于不同的端口或主机。例如，你从 erdos 两次登录到 quark 上，那么第一个 rlogin 客户就会用本地端口 1023，第二个会用端口 1022。然而，两者将连至 quark 上的同一个端口 513 上。

这个例子示出了作为汇聚点的端口的使用，在那里，客户联络一个指定的端口以获得一给定的服务。为了让客户知道正确的端口号，双方系统管理员必须在这些端口号分配上达成一个协定。对于广泛使用的服务，例如 rlogin，这些数值必须集中管理。这是由 *IETF（或互连网工程任务团体 Internet Engineering Task Force）*来管理的，它定期地发表一个标题为*分配的数值（Assigned Numbers）*的 RFC。在其中，它描述了分配给众所周知的服务的端口号，以及其它一些事情。Linux 使用一个文件来映像服务到号码，这个文件是/etc/services。它将在 *The services and protocols Files* 一节中加以叙述。

尽管 TCP 和 UDP 的连接都依赖于端口，这些数值并不会冲突。这表示，例如，TCP 的端口 513 与 UDP 的端口 513 是不同的。实际上，这些端口是作为两个不同服务的访问点的，如 rlogin（TCP）和 rwho（UDP　　）。

1.3.9 套接字（Socket）库

在 UNIX 操作系统中，执行所有上述任务和协议的软件通常是内核的一部分，或是在内核里的。世界上最通用的编程接口是*伯克利套接字库（Berkeley Socket Library）*。[译者注：socket 原意是*插座、窝、孔*] 这个名称得自于通俗的比喻，即将端口比作插座，并且将连接到一个端口比作插入插座。它提供（bind(2)）调用来指定一个远程主机、一个传输协议、以及一个程序能够连接或侦听的服务（使用 connect(2)、listen(2)、以及 accept(2)）。无论如何，这个套接字库还是比较通用的，因为它不仅提供了基于 TCP/IP 类的套接字（AF_INET 套接字），而且也提供了处理本地机器内部连接的类（AF_UNIX 套接字）。有些套接字库的实现同样还能处理其它的类，象 XNS（*Xerox Networking System*）协议，或 X.25。

在 Linux 系统中，套接字库是属于标准 libc C-library 中的。目前，它仅支持 AF_INET 和 AF_UNIX 套接字，但已在努力将 Novell 的网络协议并入，所以最终向这样的一类或几类套接字将被加进来。

1.4 Linux 连网

作为全世界编程者协作努力的结果，如果没有全球网络系统要编制出 Linux 是不可能的。所以

在开发的早期，就有几个人开始做提供网络能力的工作，这一点也不用奇怪的。几乎从一开始，一个 UUCP 的实现就运行于 Linux 上，并且在 1992 年的秋季开始着手基于 TCP/IP 的网络工作，那时，Ross Biro 以及其他人创建了如今以 Net-1 而知名的网络实现。

1993 年五月，在 Ross 停止了开发活动以后，Fred van Kempen 开始了一个新的实现的工作，改写了代码的主要部分。这个不断进行的努力以 Net-2 而知名。第一个公开发行的版本，Net-2d，是于 1992 年夏季编制的（作为 0.99.10 内核的一部分），并且从那以后经有几个人的维护和扩充，值得一提的是 Alan Cox，形成了 Net-2Degugged。在对代码大量的调试和许多改进之后，在 Linux1.0 发行之后，他将其更名为 Net-3。这是目前包括在官方内核发行版中的网络代码版本。

Net-3 给出了大量的以太网卡的设备驱动程序、SLIP（用于在串行线路上传输网络信息）设备驱动程序、以及 PLIP（用于并行线路）设备驱动程序。使用 Net-3，Linux 在局域网环境中有一个运行得很好的 TCP/IP 实现，显示出在正常运行时其性能可与一些商业 PC UNIX 媲美。开发的重点目前已面向在 Internet 主机上运行的稳定性和可靠性上。

除了这些功能外，还有一些正在进行的计划项目将增强 Linux 的多功能性。一个 PPP（点对点协议，另一个在串行线路上进行网络传输的方法）驱动程序，目前正处于 Beta 测试阶段，一个 ham 无线电的 AX.25 驱动程序正处于 Alpha 测试阶段。Alan Cox 同样还完成了 Novell 的 IPX 协议的驱动程序，但是对与 Novell 兼容的完整的网络套件的研究工作此刻没有开展下去，因为 Novell 公司不愿意提供所需的文档资料。另一个非常有希望的许诺是 *samba*，一个免费的 UNIX NetBIOS 服务器，由 Andrew Tridgell 编制的。[5]

1.4.1 不同的发展方向

目前，Fred 正继续着开发工作，正在进行 Net-2e 的工作，这是个具有非常精简的网络层设计特色的。在写本书的同时，Net-2e 仍是个 Beta 的软件。最值得注意的是 Net-2e 并入了 DDI，*设备驱动程序接口（Device Driver Interface）*。DDI 为所有的网络设备和协议提供了统一的访问和配置方法。

还有一个 TCP/IP 的实现来自于 Matthias Urlichs，他为 Linux 和 FreeBSD 编制了一个 ISDN 驱动程序。在这里，他集成了一些 BSD 网络代码到 Linux 内核中。

然而，可以预测，Net-3 将会在此停滞不前了。Alan 目前正在进行用于业余无线电爱好者的 AX.25 协议的实现的工作。毫无疑问，仍需要进行开发的内核的“模块化”将对网络代码带来新的冲击。模块化允许你在运行时刻往内核中增加驱动程序。

尽管这些不同的网络实现都努力提供同样的服务，但在内核和设备层它们之间有很大的不同。因此，你不能够使用 Net-2d 或 Net-3 的工具来配置一个运行 Net-2e 内核的系统，反之也不行。这仅能应用于与内核紧密相关的命令；应用程序以及通用的网络命令，如 *rlogin* 或 *telnet*，可以在它们任何一个上面运行。

不过，你并不用担心这些不同的网络版本。除非你参加实际的开发活动，你不用担心运行那一个版本的 TCP/IP 代码。官方的内核发行版总是带有一套与内核中的网络代码兼容的网络工具。

1.4.2 从何处获得代码

Linux 网络代码的最新版本可以通过各个站点的匿名 FTP 获得。Net-3 的官方站点是 **sunacm.swan.ack.uk**，其镜象在 **sunsite.unc.edu** 的 system/Network/sunacm 下。最新的 Net-2e 的补丁工具和执行代码在 **ftp.aris.com** 有。Matthias Urlichs 的源自 BSD 的网络代码能够从**ftp.ira.uka.de**的

/pub/system/linux/netbsd 下取得。

最新的内核能够在 **nic.funet.fi** 的/pub/OS/Linux/PEOPLE/Linux；**sunsite** 和 **tsx-11.mit.edu** 有这个目录的镜象。

1.5 维护你的系统

贯穿本书，我们将主要涉及安装与配置问题。然而，管理要做的比这多—在设置好一个服务以后，你也还需要使它运行。对于大多数这些问题来说，你仅需要很少的维护工作，而对于有一些问题，象 mail 和 news，就需要执行例程任务以使你的系统保持最新。我们将在后面章节里讨论这些任务。

最少的维护是定期地检查系统以及每个应用程序的日志文件是否有出错情况和不正常的事件发生。通常，你需要写上几行管理用的 shell 脚本并且从 *cron* 中周期地运行它们。有些主要应用程序的源程序发行，如 smail 或 C News，含有这样的脚本，你只需修改它们以适应你的需求以及偏好。

你的任何 *cron* 作业的输出应该邮递到一个管理员帐号中。缺省地，许多应用程序将送出出错报告、应用统计值、或者日志文件的概要到 **root** 帐号。这只有在你经常以 root 登录时才有意义；一个更好的办法是将 root 的邮件转发到你个人的帐号中，见第 14 章中设置邮件别名的描述方法。

然而，不管你已经如何仔细地配置了你的站点，墨菲法则确认有些问题最终*将*会暴露出来。因此，维护一个系统也意味着会有抱怨存在。通常，人们会期待系统管理员起码能够通过 root 的 email 来接触，但是，同样还有其它一些地址，这些地址常用来联系到负责维护某一方面的人。例如，有关错误的邮件配置的抱怨信息常常邮递到 postmaster；news 系统的问题可以报告到 newsmaster 或 Usenet。到 hostmaster 的邮件应该重定向到负责主机基本网络服务以及 DNS 名字服务（如果运行了名字服务器）的人那里去。

1.5.1 系统安全

网络环境中系统管理的另一个非常重要的方面是保护你的系统与用户免受入侵者的干扰。粗心管理的系统给怀有恶意的人许多攻击目标：攻击的范围从猜测口令到以太网探听，导致的损害可能从伪造的邮件消息到数据的丢失或用户权限的侵犯。当讨论相关方面时我们将论及一些特殊的问题，以及针对它们的一些常用的防范措施。

本节将讨论有关系统安全的一些例子和基本技术。当然，所覆盖的主题不能详尽地涉及到你可能要面对的所有安全问题；它们仅仅举例说明可能碰到的问题。因此，阅读一本有关安全的好书是绝对必要的，尤其是在一个连网的系统中。Simon Garfinkel 的“实用 UNIX 安全”（见[Spaf93]）是极为值得推荐的。

系统安全开始于好的系统管理。这包括检查所有重要文件和目录的所有权和权限，监视特权帐号的使用等等。例如，COPS 程序将检查你的文件系统以及常用的配置文件的不寻常的权限和异常情况。它也能够聪明地使用一组口令来迫使用户的口令遵循一定规则以使得它们难以猜出。例如，影子口令集需要口令至少有五个字符，并且含有大小写字符和数字。

当使一个服务在网络上可以访问时，确信给它“最低权限”，这意味着你不允许它做在设计时对于它的工作不需要的事情。例如，仅仅在它们真的需要的情况下，你才应该使程序 setuid 到 root 或一些其他特权帐号。同样，如果你想对仅仅非常有限的应用程序使用一个服务时，不要犹豫，就你的特定应用程序所允许的条件下尽量严格地配置它。例如，如果你想要允许无盘主机从你的机器上

进行引导，你必须提供 TFTP（直接文件传输服务（trivial file transfer service)），这样它们就可以从/boot 目录中下载基本配置文件。然而，如果不加限制地使用，TFTP 就允许世界上的任何用户从你的系统中下载任何可读的文件。如果这并不是你想做的，为什么不限制 TFTP 服务只能访问/boot 目录呢？[6]

同样的想法，你可能也想要来自某些主机的用户限制某些服务，比如说来自你的本地网络的用户。在第 9 章，我们将介绍 tcpd，它会针对各种网络应用程序来做这个事情。

另一个重要点是避免运行“危险的”软件。当然，你所用的任何软件都可能很危险，因为软件都可能有错误，聪明的人可能会利用它来取得对你的系统的访问。这种事情发生过，而且对于此并没有完全的保护措施。这个问题同样影响自由软件以及商业软件产品。[7] 然而，需要特殊权限的程序天生就比其它程序更危险，因为任何漏洞都可能带来强烈的后果。[8] 如果你为了网络的目的而安装 setuid 程序，要有双倍的小心，不要遗漏文档上所说的任何事情，以便于你不会意外地建立了一个安全上的裂口。

不管你是如何地小心，决不能排除你的防范可能会失效。因此，必须尽早地察觉入侵者。检查系统日志文件是一个良好的开端，但是，入侵者也会是同样的聪明，会删除他或她留下的任何明显的痕迹。然而，有一种象 tripwire 一样的工具[9] ，它允许你检查重要的系统文件，以发现它们的内容或权限是否改动过。Tripwire 在这些文件中计算各种强壮的检查和并且存储在一个数据库中。在以后的运行期间，会重新计算检查和并与存储的检查和比较以检测出任何改动的情况。

1.6 以下各章展望

以下几章将涉及 TCP/IP 网络的配置，以及一些主要程序的运行。在开始编辑文件等工作之前，我们将在第 2 章更进一步地讨论 IP。如果你早已知道 IP 路由选择的工作方法，以及地址解析是如何进行的，你可以跳过这一章。

第 3 章讨论最基本的配置问题。比如象建立一个内核以及设置你的以太网卡。串行口的配置在一个独立的一章中讨论，因为它的讨论不仅仅适用于 TCP/IP 网络，同样也与 UUCP 相关。

第 5 章帮助你针对 TCP/IP 网络设置你的机器。它包括只有回环使能的独立的主机的安装提示，以及连接至以太网的主机的安装提示。它也将介绍一些有用的工具，用以测试和调试你的安装设置。下一章将讨论如何配置主机名解析，并解释如何安装设置一个名字服务器。

接下来的两章分别着重介绍了 SLIP 和 PPP 的配置和使用。第 7 章解释了如何建立 SLIP 连接，并给出了一个详细的工具 *dip* 的参考。这个工具允许你自动操作大多数必要的步骤。第 8 章涵盖 PPP 以及 PPP 所需的的后台处理程序 *pppd*。

第9章给出了设置一些很重要的网络应用程序的简短介绍，比如rlogin、rcp等等。这一章也介绍了服务是如何通过inetd super来管理的，以及你如何可以限制与安全相关的服务到一组（受托）可信的主机上。

接下来的两章讨论NIS，即网络信息系统（Network Information System），和NFS，即网络文件系统（Network File System）。对于象在一个局域网中的用户口令等的分布管理的信息来说，NIS是一个有用的工具。NFS允许你在你的网络中的几台主机上共享文件系统。

第12章给出了对Taylor UUCP管理的更进一步的介绍，Taylor UUCP是UUCP组件的一个免费实现。

本书的剩余章节详细讨论了电子邮件和Usenet News。第13章介绍了电子邮件的基本概念，如邮件地址看上去是怎样的，以及邮件处理系统是如何管理将你的消息送至接收者的。

第14章和第15章涵盖了smail和sendmail的设置，两个你可以使用的邮件传送代理。本书对这两者都进行了介绍，因为smail对初学者来说很容易安装，而sendmail则更具有灵活性。

第16章和第17章介绍了在Usenet中news管理的方法，以及你如何安装和使用C news，一个流行的管理Usenet news的软件包。第18章简要地叙述了对于你的本地网络如何设置一个NNTP后台程序来提供新闻阅读访问。第19章最后示出了如何配置和维护各种各样的newsreaders。

注释

[1] 上面这段话的本意，仍然在欧洲时有发生。

[2] 时间并没有对此改变多少。

[3] 当使用bash时，即GNU Bourne Again Shell，你可能需要避免使用感叹号，因为bash使用它作为它的历史字符。

[4] 在RFC-1055中对SLIP进行了讨论。头部压缩CSLIP的基本原理在RFC-1144中进行了叙述。

[5] NetBIOS是一个协议，象lanmanager以及Windows for workgroups就是基于它的。

[6] 在第9章我们将再回到这里来。

[7] 曾经有个你需要花很多钱购买的商业Unix，带有一个setuid-**root**的shell脚本，使用一个简单标准的诡计，该脚本程序就可允许用户获得**root**特权

[8] 在1988年，RTM蠕虫使得觉大多数Internet停滞，部分是通过利用一些sendmail程序的裂孔。这个漏洞早已修复。

[9] 有Gene Kim和Gene Spafford编写。

第二章 TCP/IP 网络的问题

我们将转至讨论一些你将遇到的一些细节上，当你将你的 Linux 机器连接到 TCP/IP 网络时就会用到这些细节，它涉及到 IP 地址、主机名、以及有时是路由选择问题。这一章给出了你所需的背景资料，以用于理解你的设置需求，下一章将讨论涉及这些的一些工具。

2.1 网络接口

为了隐藏可能用于网络环境的设备的差异性，TCP/IP 定义了一个抽象的*接口（interface）*，通过这个接口来访问硬件。这个接口提供了一组操作，该组操作对所有的硬件类型是一样的，并且基本上是涉及发送和接收分组。

对于每种你想用于连网的外围设备来说，必须在内核中有一个相应的接口。例如，Linux 中以太网的接口称作 *eth0* 和 *eth1*，SLIP 的接口是 *sl0*、*sl1* 等等。这些接口名称是用于配置目的，那时你想给内核命名一个特定的物理设备。除了这别无它意。

为了对 TCP/IP 连网有用，一个接口必须分配一个 IP 地址，当与其它地方通信时这个地址起到它的鉴定识别的作用。这个地址与上面提到的接口名称是不同的；如果你将接口比作一扇门的话，那么这个地址就象是门上钉的名牌一样。

当然，可能还有其它一些设备参数需要设定；其中之一就是特定硬件能够处理的最大数据报的大小，也称为*最大传输单元（Maximum Transfer Unit）*，或 MTU。别的属性将在后面介绍。

2.2 IP 地址

如上章所提到的，IP 网络协议所能理解的地址是一个 32 比特的数字。每一台机器必须分配一个对于网络环境来说唯一的一个数字。如果你运行一个不与其它网络有 TCP/IP 通信的本地网络，你可以依据个人的爱好来分配这些数字。然而，对于 Internet 上的站点来说，这些数字是由一个中心权威机构，网络信息中心（Network Information Center 或 NIC）来分配的。[1]

为了容易阅读，IP 地址被分解成四个 8 比特数字称为*八位位组（octets）*。例如，quark.physics.groucho.edu 有一个 IP 地址为 0x954C0C04，它可以写成 149.76.12.4。这个格式通常称作为*点分四组表示法（Dotted quad notation）。[还有一种称呼为点分十进制表示法 Dotted decimal notation—译者注]*

这种表示法的另一个原因是 IP 地址可以分成包含在开头（左面）的八位位组中的一个*网络*号（*network* number）和包含在剩余（右面）的八位位组中的一个*主机*号（*host* number）。当 IP 地址遵循 NIC 的规则使用时，并不是为你计划要使用的每台主机分配一个地址，而是给你一个网络号，并且允许你依照你的喜好，在你的网络上给各台主机分配在该网络号范围内的所有有效的任意 IP 地址。

根据组成的网络规模的大小，IP 地址的主机部分可大可小。为适应不同的需要，定义了几类网络，给出了 IP 地址的不同分割法。

A 类　A 类由 1.0.0.0 到 127.0.0.0 的网络构成。网络号包含在第一个八位位组中。这样就提供了 24 比特位的主机部分，每个网络允许大约有 160 万台主机。

B 类　B 类包括从 128.0.0.0 到 191.255.0.0 的网络；网络号由前两个八位位组构成。这样就可以有 16320 个网络，而每个网络可以有 65024 台主机。

C 类　C 类网络的范围从 192.0.0.0 到 223.255.255.0。网络号由前三个八位位组组成。可以有近 200 万个网络，而每个网络能容纳 254 台主机。

D、E、F 类的地址范围从 224.0.0.0 到 254.0.0.0，用于试验用途或保留给将来使用，而没有分配给任何网络用。

如果我们再参考前一章的例子，我们发现 quark 的地址 149.76.12.4 对应 B 类网络 149.76.0.0 的 12.4 的主机。

你可能已经注意到，上面列出来的主机部分不是每个八位位组的所有数值都可以使用的。这是因为八位位组为全 0 或 255 的主机号是留作它用的。主机部分的所有比特位为全零的地址引用为一个网络，而主机部分的所有比特位为全 1 的地址称为广播地址。它同时指一个网络上的所有主机。因此，149.76.255.255 不是一个有效地址，而是指网络 149.76.0.0 上的所有主机。

还有两个网络地址 0.0.0.0 和 127.0.0.0 也是保留的。这第一个称为*缺省路由（default route）*，后一个称为*回送地址（loopback address）*。缺省路由与 IP 路由数据报的方式有关，这将在下面讨论。

网络地址 127.0.0.0 保留作你的主机内部 IP 通信用。通常，地址 127.0.0.1 将分配给主机上的一个特殊接口使用，称为*回送接口（loopback interface）*，它的行为象一个闭合电路。从 TCP 或 UDP 送来的任何 IP 分组都将返送给它们，就好象这个分组是刚从某个网上传过来的。这允许你不必使用一个“真正”的网络就能开发并测试网络软件。它的另一种很有用的应用是在当你想在单机上使用网络软件时。这并不象听起来那么罕有；例如，许多 UUCP 站点根本就没有 IP 连通，但仍然要运行 INN 新闻系统。为了要正常的运作，INN 需要回送接口。

2.3 地址解析

现在，你已经知道 IP 地址是如何形成的，你可能会觉得奇怪，它们在以太网上是如何用于编址不同的主机的。毕竟，以太网协议是用一个与 IP 地址完全不同的六个八位位组的数值来识别主机的，不是吗？

对。这就是为什么需要一种将 IP 地址映射到以太网地址的机制了。这就是所谓的*地址解析协议（Address Resolution Protocol）*，或 ARP。实际上，ARP 根本不限于以太网，它也可以应用于其它类型的网络，如业余无线电爱好者等。ARP 的基本概念正象人们在一个 150 人的人群中寻找 X. Ample 先生时大多数人的做法一样：他们环绕着人群，叫喊着他的名字，并确信如果他在那里的话一定会应答的。

当 ARP 想找出与给定 IP 地址相应的以太网地址时，它就利用了以太网的“广播”特性，此时一个数据报在网上同时寻址所有的站点。ARP 所发送的广播数据报含有对相应 IP 地址的查询。每台主机将接收到的 IP 与自己的 IP 地址相比较，如果一致，就返回一个 ARP 响应给发出查询的主机。此时发出查询的主机就可以从响应数据报中提取发送者的以太网地址了。

当然，你会觉得惊讶，一台主机怎么能知道在全世界数不清的以太网上它能找到所期望的主机呢？并且为什么一定是以太网呢？这些问题都涉及称为路由选择的操作，也即在网络上寻找出一台

主机的物理位置。这将是下一节的主题。

此时，让我们再多讨论一会 ARP。一旦一台主机发现了一个以太网地址，该主机就将它存贮在 ARP 的缓冲中，这样主机在下一次想要发送数据报给所讨论的主机时就不要再次查询它的以太网地址了。然而，永久地保存这些信息是不明智的；例如，远程主机上的以太网卡有可能由于技术上的问题而被更换掉，此时 ARP 的登记项就变得无效了。为了强迫进行另一次 IP 地址的查询，ARP 缓冲中的登记项因此在一段时间后就被丢弃。

有时候，需要找出与给定以太网地址相对应的 IP 地址。当一台无盘机器想从网上的一台服务器上引导时就会有这种情况，这在局域网上是常有的情形。但是，一个无盘客户实际上并没有有关自身的信息—除了知道自己的以太网址外！所以它主要要做的是广播一条请求引导服务器告诉它它的 IP 地址的消息。针对于此，有另一个协议，命名为*反向地址解析协议（Reverse Address Resolution Protocol）*，或 RARP。与 BOOTP 协议一起，它用于规定在网上引导无盘站的过程。

2.4 IP 路由选择

2.4.1 IP 网络

当你给某人写了一封信，通常会在信封上写上完整的收信人地址，列出了国家、州和邮政编码等等。当你把信放入邮箱时，邮政服务会把它送到目的地：它将被送到指定的国家中去，该国的邮政服务会把它分发到正确的州和地区，等等。这个分级方案的优点是显而易见的：不管你在哪里发出了这封信，当地的邮递员将会大概地知道该封信要投递的方向，但并不关心这封信在送达的国家中以什么方式来投递。

IP 网络的结构与之相似。整个互连网（Internet）是由许多专有网络构成，称为*自治系统（autonomous systems）*。每个这样的系统在其内部成员主机之间执行着各种路由选择，所以分发一个数据报的任务被简化成寻找一条到达目的主机网络的路径。这意味着，一旦数据报被传到那个特定的网络上的*任何（any）*主机上，网络本身会专门进行进一步的处理。

2.4.2 子网（Subnetworks）

这个结构是通过前面所述的将 IP 地址分成主机部分和网络部分来反映出来的。默认地，目的网络得自 IP 地址的网络部分。因此，有同样 IP 网络号的主机在同一个网络上，反之也然。[2]

在网络内部使用类似的方案也同样合理，由于它本身可能有许多更小的网络组成，最小的单元可以是一个象以太网一样的物理网络。因此，IP 允许你将一个 IP 网络分割成一些*子网（subnets）*。

在子网所属的 IP 网上，子网接管了分发数据报到某个 IP 地址范围的职责。对于 A 类、B 类或 C 类网，它是由 IP 地址的网络部分确定的。然而，现在把网络部分扩展成也包括主机部分的几个比特位。被解释为子网号的这几个比特位的位数是由称为*子网掩码（subnet mask）*，或*网络掩码（netmask）*的数值指定的。这同样是一个 32 位的数值，它指定了 IP 地址网络部分的位掩码。

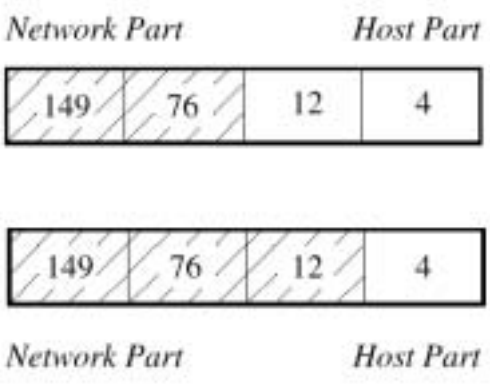

图 2.1 将 B 类网分成子网

Groucho Marx 大学的校园网就是这样一个网的例子。它的 B 类网络号是 149.76.0.0，它的网络掩码也就是 255.255.0.0。

就内部而言，GMU 的校园网由几个更小的网络组成，如各个系的局域网。所以 IP 地址的范围被分成 254 个子网，149.76.1.0 到 149.76.254.0。例如，理论物理系的分配的子网号是 149.76.12.0。校园主干网本身就是一个网络，并给予子网号 149.76.1.0。这些子网号共享同一个 IP 网络号，而第三个八位位组用于区别它们。因此，他们将使用一个子网掩码 255.255.255.0。

图 2.1 示出了 guark 的地址 149.76.12.4 作为一个普通的 B 类网和当使用子网技术时是如何有不同的解释的。

子网（正如生成子网的技术声称）仅是网络的*内部分割（internal division），*并没有什么用。子网是由网络的拥有者（或管理员）生成的。常常，子网的建立是为了反映现存的物理上（如两个以太网之间）的、管理上（在两个系之间）的、或地理上的分界线，并且在这些子网上的权限被授权给某些联系人。然而，这个结构仅反映出网络的内部行为，对于外部世界它是完全不可见的。

2.4.3 网关

子网技术不仅有组织上的益处，它常常是一个自然的硬件分界线的结果。在一个给定的物理的（实际的）网络上，如以太网上，主机的通信范围是有限的：能够直接与之通信的主机是它所在网络上的主机。所有其它的主机仅能通过称为*网关（gateways）*的设备访问。网关是一台同时连接两个或多个物理网络的主机并配置成在这些网络之间交换分组。

由于 IP 能够很容易地识别一台主机是否在一个本地物理网络上，不同的物理网络需要属于不同的 IP 网络。例如网络号 149.76.4.0 是保留给数学系局域网上的主机的。当给 quark 发送一个数据报时，erdos 上的网络软件立刻会从 IP 地址 149.76.12.4 上看出，目的主机在另一个不同的物理网络上，因此可以通过一个网关（缺省的是 sophus）到达那里。

Sophus 本身连接到两个不同的子网上：数学系和校园主干网上。它分别通过不同的接口，eth0 和 fddi0 来访问这两个子网。现在，我们要给它分配那种 IP 地址呢？我们是否要给它一个在子网 149.76.1.0 的 IP，或 149.76.4.0 网上的 IP 呢？

答案是：两个都要。当与数学系局域网上的主机通信时，sophus 就使用 IP 地址 149.76.4.1，当与主干网上的主机通信时，它将使用 ID 地址 149.76.1.4。

因此，网关所连的每一个网络都要给网关分配一个 IP 地址。这些地址---及相应的网络掩码---被约束在相应的接口上，通过该接口子网可以进行访问。因此，sophus 的接口与地址映射就看上去象这样的：

iface	address	netmask
eth0	149.76.4.1	255.255.255.0
fddi0	149.76.1.4	255.255.255.0
lo	127.0.0.1	255.0.0.0

最后一个条目描述了回送接口 *lo*，这在上面已介绍过。

图 2.2 显示了 Groucho Marx 大学（GMU）的部分网络拓扑图。同时在两个子网上的主机有两个地址。

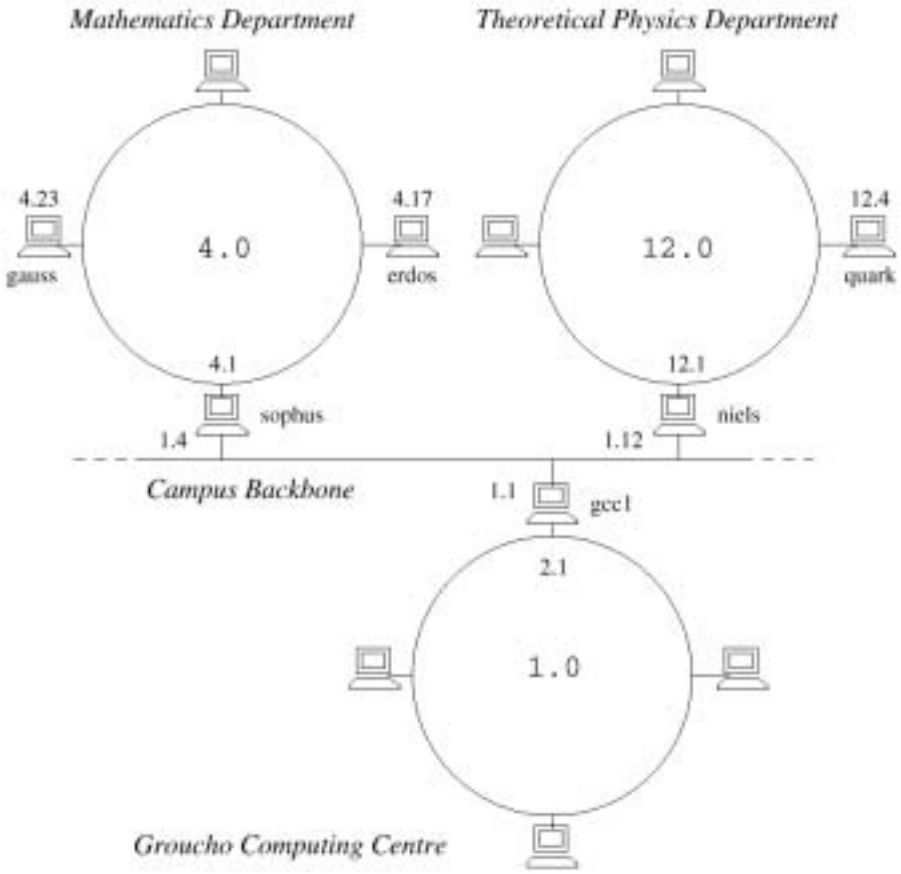

图 2.2 Groucho Marx 大学（GMU）的部分网络拓扑图。

一般来说，你可以忽略将地址附加到主机或它的接口之间的区别。对于仅在一个网上的主机，如 erdos，你通常只需以这个以及那个地址来谈及主机，尽管严格地来说，是以太网接口有这个 IP 地址。然而，这种区别仅在谈论网关时显得重要。

2.4.4 路由选择表

我们现在将注意力集中在当分发一个数据报到远程网络时，IP 是如何选择使用一个网关的。

我们前面已经看到，当给 quark 发送一个数据报时，erdos 检查目的地址并且发觉它不在本地网络上。Erdos 因此将该数据报发送到缺省网关，sophus，它现在基本上面临同样的任务。Sophus 注

意到 quark 不在它直接连接的任何网络上，所以它必须找到另一个网关并通过该网关来转发数据报。正确的选择是到物理系的网关 niels。此时，sophus 需要一些信息将目的网络与适当的网关相联系。

为此，IP 所使用的路由选择信息基本上是一张表，该表将各网络与到达该网络要经过的网关链接在一起。还必须提供一个捕获所有分组的表项（缺省路由）；这是与网络 0.0.0.0 相关的网关。所有发送到未知网络的分组都是通过该缺省路由器发送的。在 sophus 上，这张表看上去象这样：

Network	Gateway	Interface
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

路由到 sophus 直接连接的网络是不需要路由器的；因此它们将网关条目显示为“-”。

路由选择表可以通过各种方法来建立。对于小型局域网，在引导主机时通过手工使用路由器命令构造路由表，并将此表供给 IP 的做法通常是最有效的（见第五章）。对于大规模的网络，它们是通过路由选择后台服务程序在运行时刻建立并调整的；它们运行在网络的中央主机上并且与成员网络之间交换路由选择信息来确定“最佳”路由。

根据网络大小的不同，将使用不同的路由选择协议。对于在自治系统内进行的路由选择（比如 Groucho Marx 校园），要使用*内部路由协议（internal routing protocols）*。最著名的一个是 RIP，即路由信息协议（或称选路信息协议），它是由 BSD 的 routed 后台程序实现的。对于在自治系统之间的路由选择，必须使用*外部路由协议（external routing protocols）*象 EGP（外部网关协议（External Gateway Protocol））, 或 BGP（边界网关协议（Border Gateway Protocol））；这些（以及 RIP）已经在 Cornell 大学的 gated 后台程序中实现。[3]

2.4.5 度量值（Metric Values）

基于 RIP 的动态选路（或路由选择）根据“跳数”选取到达某些目的主机或网络的最佳路径，也即，一个数据报在到达目的之前所经过的网关个数。路径越短，RIP 性能越好。跳数多于或等于 16 的很长的路径被看作是无用的，并且被舍弃。

要使用 RIP 来管理本地网络内部的路由信息，你必须在所有主机上运行 *gated*。在启动引导时，*gated* 会检测所有活动的（处于运行状态的）网络接口。如果有不止一个活动的接口（回送接口不算），就可假设主机在几个网络之间交换信息包，并且会积极地交换和广播路由信息。否则的话，主机只是被动地接收任何 RIP 更新信息，并且更新本地路由表。

当在广播本地路由表中的信息时，*gated* 从与路由表条目关联的称之为*度量值（Metric value）*的数值中计算路径的长度。这个度量值是系统管理员在配置路由时设定的，并且应该反映出这个路由的实际代价。因此，到主机直接连接的子网的路由的度量一定是零，而通过两个网关的路由的度量应该是 2。然而，当你不使用 RIP 或 gated 时，就不必为度量而烦恼。

2.5 互连网控制报文协议

IP 有一个相伴的协议我们至今还没有讨论，这就是*互连网控制报文协议（Internet Control Message Protocol）*（ICMP），用于内核中的网络代码与其它主机进行出错报文及其它的通信。例如，假设你又在 erdos 上并且想远程登录（telnet）到 quark 上的 12345 端口，但是并没有进程在侦听那个端口。当第一个 TCP 信息包到达 quark 的这个端口时，主机内的网络层将其识别出并且立刻给 erdos 返回一个 ICMP 报文，指出“端口不可达”。

有相当多 ICMP 可以理解的报文，其中许多涉及到出错的情况。然而，有一个非常有趣的报文称为重定向报文（Redirect message）。它是由路由选择（选路）模块在检测到有其它主机正使用它作为一个网关时产生的，尽管可能存在一个更短的路由。例如，在主机引导后，sophus 的路由选择表可能并不完整，含有到数学系网络的路由和到 FDDI 主干网的路由，并且缺省路由是指向 Groucho 计算中心的网关（gcc1）。因此，任何发送到 quark 的信息包将被送至 gcc1 而不是送到物理系的网关 niels。当接收到这样一个数据报时，gcc1 将注意到这是一个不良的路由，并且会将这个信息包转发到 niels，同时给 sophus 返回一个 ICMP 重定向报文告诉它这个更好的路由。

现在，除了最基本的路由外，这好象是避免手工设置其它路由的聪明方法。然而，这里要给出警告，如果要依赖于 RIP 或 ICMP 重定向报文的动态选路方案，并不总是一个好主意。ICMP 重定向以及 RIP 在验证某个选路信息是否需要权限方面很少提供或几乎不提供选择。这使得怀有恶意的无用之人能够中断你的整个网络的通信，或者更糟。由于这个原因，有些网络代码的版本将影响网络路由的重定向报文看作仅是主机路由的重定向。

2.6 域名系统

2.6.1 主机名解析

如前所述，TCP/IP 网络中的编址是围绕着 32 比特的数字的，然而，想多记住几个也是很难的。因此，通常主机都用“普通”的名字命名，如 gauss 或 strange。这时，找出与名字相关的 IP 地址就是程序的职责了。这个过程称为*主机名解析（host name resolution）*。

一个想要找出给定主机名的 IP 地址的应用程序不一定要提供自己的例程来查找主机以及 IP 地址。相反地，它依赖于显然是做这个工作的几个库函数，称为 *gethostbyname(3)*和 *gethostbyaddr(3)*。传统上，这些以及一些相关的过程被组合成一个独立的库，称为解析库（resolver library）；在 Linux 上，这些是标准 *libc* 的部分。通俗地说，这个函数的集合因而被称为“解析器”（the resolver）。

如今，在一个象以太网一样的小网上，或者甚至是一族以太网上，维护一张映射主机名到地址的表并不是十分困难的事。该信息通常被存放在文件*/etc/hosts* 中。当增加或移走主机，或重新分配地址时，你所要做的就是更新所有主机上的 *hosts* 文件。很明显，当网络不仅仅是由少数几台机器组成时，这将成为繁重的负担。

针对这个问题的一个解决方案是 NIS，即有 Sun Microsystems 开发的*网络信息系统（Network Information System）*，通俗地称为 YP，或者*黄页（Yellow Pages）*。NIS 在主主机上将 hosts 文件（以及其它信息）存储在一个数据库里，客户可以从中检索所需的信息。不过，这个方法只适应中等大小的网络如局域网，因为它涉及到集中地维护整个主机数据库，并且把它分发到所有服务器上。

在互连网上，地址信息最初也是存储在一个 HOSTS.TXT 数据库中的。这个文件原是由网络信

息中心（Network Information Center），即 NIC，来维护的，并且所有参与站点都需要下载并安装。当网络增长时，这种方案带来了几个问题。除了涉及定期安装 HOST.TXT 的额外管理方面的开销外，分发这个库的服务器负荷变得太高了。更为严重的问题是所有的名字必须在 NIC 登记，它必须确定没有重名存在。

这就是为什么在 1984 年采用了一个新的名字解析方案，即*域名系统（Domain Name System）*。DNS 是由 Paul Mockapetris 设计的，同时能解决这两个问题。

2.6.2 进入 DNS

DNS 用域的层次结构来组织主机名。一个域是在某些方面相关的站点的一个集合---由于他们组成了一个特有的网络（例如，校园内的所有机器，或 BITNET 上的所有主机），因为它们都属于某个机构（就象美国政府），或者因为它们只是在地理位置上比较靠近。例如，大学被（组合）分组在 edu 域，每个大学或学院使用一个独立的*子域（subdomain）*，在子域下面包含它们的主机。Groucho Marx 大学可以给予 groucho.edu 域，数学系的局域网指定为 maths.groucho.edu。系部网络上的主机将有这个域名附加在它们的主机名后；所以 erdos 被认为是 erdos.maths.groucho.edu。这被称为*全资域名（fully qualified domain name）*，或 FQDN，它在世界范围内唯一地标识出该主机。

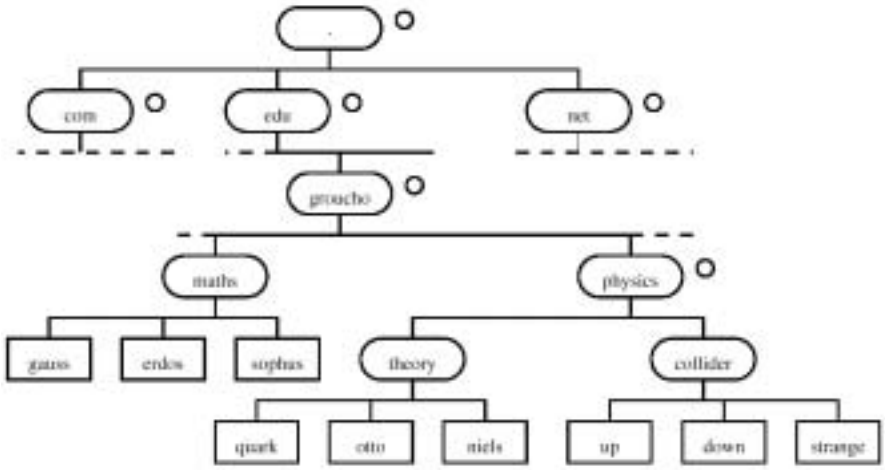

图 2.3 部分域名空间。

图 2.3 示出了名字空间的一部分。这棵树的根的入口是用一个点（dot）来表示的，它被很恰当的称作*根域（root domain）*，并且包含了所有其它域。为了指出一个主机名是一个全资域名，而不是一个与某个（隐含的）本地域相关的名字，有时会写上一个附加点。这表示这个名字的最后的组成部分是根域。

依赖它在层次结构中的位置，一个域可以被称为是顶层、第二层、或第三层的。还可以再划分出分层，但很好这样做。你会常看到以下几个顶层域：

edu （大部分在美国） 教育机构如大学等。

com 商业机构、公司。

org 非商业机构。常常是私有 UUCP 网络在这个域中。

net 网络上的网关以及其它的管理主机。

mil 美国军事机构。

gov 美国政府机构。

uucp 具官方称，以前用作无域名的 UUCP 名字的所有站点名，已被移入该域。

技术上来说，以上的头四个属于 Internet 的美国部分，但你可能在这几个域中还是能见到不是美国的站点。Net 域尤其是这样。然而，mil 和 gov 是美国专有的。

在美国以外的地方，每个国家通常在两字符国家代码（在 ISO-3166 中定义）后面使用一个她自己命名的顶层域。例如，芬兰（Finland）使用 fi 域，fr 由法国（France）使用，de 由德国（Germany）使用，au 由澳大利亚（Australia）使用，cn 由中国（China）使用等等。在这个顶层域下面，每个国家的 NIC 可以自由地以他们想要的方式组织主机名。例如，澳大利亚有一个与国际顶层域相似的次层域，命名为 com.au，edu.au，等等。其他国家，如德国，不使用这一额外层，而是使用直接引用运行一特定域的机构的稍长些的名字。例如，象ftp.informatik.uni-erlangen.de这样的主机名并非少数。由此可见德国人的效率如何了。

当然，这些国家的域并不意味着在该域下的主机实际上一定位于那个国家之中；这仅仅说明这台主机是在那个国家的 NIC 注册登记的。一个瑞典（Swedish）的厂商可能在澳大利亚有一分支机构，但他的所有主机仍然注册为 se 顶层域。

现在，以域名的层次结构来组织名字空间很好地解决了名字的唯一性问题；利用 DNS，一台主机的名字只需在它的域中是唯一的，就可以在世界范围内有一个与所有其它主机不同的名字了。此外，全资名称是很容易记忆的。就其本身而言，这些已是将大的域分割成几个子域的很好的理由了。

但是，DNS 甚至为你比这做得更多：它允许你将子域权限授权给它（子域）的管理者。例如，在 Groucho 计算中心的维护者可以为每个系部创建一个子域；在上面我们已经遇见了 maths 和 physics 子域了。当他们发现物理系的网络太大了，而且从外界来看混乱而难以管理（总之，物理学家是一类不受拘束的人），他们就可以简单地将 physics.groucho.edu 域的控制权交给这个网络的管理员。这样，他们就可以自由地使用他们喜欢的无论什么样的主机名了，并且可以以各种方式在他们的网络中分配 IP 地址了，而不需要外界的干涉。

在本小节结束时，我们还要说一下，名字空间可以分成*区（zones）*，每个区根于一个域。请注意区与域的细微差别：域 groucho.edu 包括 Groucho Marx 大学的所有主机，而区仅包括计算中心直接管理的主机，例如那些在数学系的主机。在物理系的主机就属于一个不同的区，也即 physics.groucho.edu。在图 2.3 中，区的开始是在域名的右侧用小圆圈标出的。

2.6.3 用 DNS 进行名字查找

第一眼看上去，所有这些域和区的麻烦似乎使得名字解析成为一件非常复杂的事情了。毕竟，如果没有中心权威控制什么名字分配给哪台主机的话，那么推知它将是一个多么粗陋的程序啊？！

现在讨论有关 DNS 的真正精华（自然，坦白，直率）的部分了。如果你想找出 erdos 的 IP 地址，那么，DNS 会说，去问管理它的人，他们会告诉你。

实际上，DNS 是一个巨大的分布式数据库。它是依靠所谓的名字服务器来实现的，名字服务器为一个给定的域或一组域提供信息。对于每一个区，起码有两个、最多有几个的名字服务器，在那个区的主机上掌握着所有的权威信息。为了得到 erdos 的 IP 地址，你所要做的只是联系 groucho.edu

区的名字服务器，名字服务器将返回你所期望的数据。

你可能会想，说起来容易做起来难。那么我怎样知道如何达到 Groucho Marx 大学的名字服务器呢？如果你的计算机没有配备一个地址解析程序，DNS 也能提供这个能力。当你的应用程序想要在 erdos 上查找信息时，它就会与本地名字服务器联系，该名字服务器会为它处理一个迭代查询。它首先向根域的名字服务器发出一个查询，询问 erdos.maths.groucho.edu 的地址。根名字服务器注意到这个名字不属于它的管理权限范围内，而是属于 edu 域下的某个。因此，它告诉你与 edu 区的名字服务器联系以取得更详细的信息，并且给了所有一张 edu 的名字服务器及其地址的列表。此时，你的本地名字服务器将继续执行并查询那些 edu 的名字服务器之一，例如 a.isi.edu。和根名字服务器同样的方式，a.isi.edu 知道 groucho.edu 运行于他们自己的区里，并且使你指向他们的服务器。现在，本地名字服务器将自己的查询发送到这些服务器之一上，这将最终认可这个属于它的区的名字，并且返回相应的 IP 地址。

现在，看上去为了查找一个小小的 IP 地址为产生许多的通信量，但是，如果我们仍然使用 HOSTS.TXT 的话，所产生的数据传送量要比上述方法大的多。但是这个方案还有需要改进的地方。

为了改善将来查询的响应时间，名字服务器将在其本地缓冲中存储所获得的信息。所以当下次你的本地网络上有任何人想要查找在 groucho.edu 域上主机的地址时，你的名字服务器就不用再次经历整个过程，而是会直接到 groucho.edu 名字服务器上。[4]

当然，名字服务器并不会永远保留这个信息的，而是会在一定时间后放弃的。这个到期间隔时间称为*存活期（time to live）*，或 TTL。DNS 数据库中的每一数据都有相应责任区的管理员指定的 TTL。

2.6.4 域名服务器

拥有一个区内所有主机信息的名字服务器被称为该区*授权的（authoritative）*，并且有时称之为*主名字服务器（master name servers）*。对该区内主机的任何查询都将最终绕回到这些主名字服务器之一上。

为了提供与一个区一致的描述，它的主服务器必须很好地同步。这是通过使他们其中之一成为*主要（最初、原始）的（primary）*服务器来达到的，它从数据文件中装入区信息，并且使别的服务器成为*次要（第二位）的（secondary）*服务器，并从主要服务器中周期性地传入区数据。

有几个名字服务器的一个原因是为了分散工作负荷，其它的原因是为了有容余备份。当一个名字服务器机器毫无准备地出了故障，如崩溃了或失去了网络连接，所有的查询都将回送到其它的服务器上。当然，这个方案并不会保护你免受服务器故障而产生对所有 DNS 请求的错误应答，例如，由于服务器程序本身的软件错误。

当然，你也可以运行一个没有为任何域授权的名字服务器。[5] 不过这类服务器也是有用的，因为它还是可以为运行于本地网络上的应用程序管理 DNS 查询的，并缓冲所得信息。因此，它被称为*只缓冲（caching-only）*服务器。

2.6.5 DNS 数据库

通过上面我们已经知道，DNS 不仅仅是只处理主机的 IP 地址，而且还交换名字服务器上的信息。实际上，DNS 数据库可以有许多不同类型条目。

DNS 数据库上的一条信息称为一个*资源记录（resource record）*，或简写成 RR。每一条记录都

有一种与之关联的类型，描述了它所表示的数据，以及一个指明它所适用的网络类型的类。后者用于适应（或调节）不同编址方案的需要，象 IP 地址（IN 类）、或 Hesiod 网络的地址（在 MIT 使用）、以及其它一些。原型资源记录类型是 A 记录，它将一个全资域名与一个 IP 地址相联合。

当然，一台主机可以有不止一个名字。然而，其中一个名字必须指定为正式的，或者是*正规主机名（canonical host name）*，而其它的名字只是前者的别名。它们之间的区别在于正规主机名是与一个 A 记录关联的，而其它名字只有一个指向正规主机名的 CNAME 类型的记录。

这里我们将不讨论所有的记录类型，而是留在后面的章节中讨论。这里只给出一个简短的例子。图 2.4 显示出了装入 physics.groucho.edu 区的名字服务器中的域名数据库的一部分。

```
;
; 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
...
```

图 2.4 物理系 named.hosts 文件的摘录。

除了 A 和 CNAME 记录以外，在该文件的顶部还可以看到一条特殊的记录，延伸了几行。这是 SOA 资源记录，意思是*授权的开始（Start of Authority）*，它保留着服务器授权的区的常用信息。例

如，它包括所有记录缺省的存活期信息。

注意，例子文件中所有不以点结尾的名字将被解释成是相对于 groucho.edu 域的。用于 SOA 记录的特别名字“@”是指域名本身。

从上面我们已经知道，groucho.edu 域的名字服务器必需知道有关物理系的区，这样它们就可以将查询传给它们的名字服务器了。这常常是通过一对记录来完成的：给出服务器的 FQDN 的 NS 记录、以及一个关联地址和那个名字的 A 记录。由于这两个记录一起控制着名字空间，它们常被称为*粘合记录（glue records）*。它们是仅有的记录实例，在该记录中父辈区保存着下属区中主机的信息。指向 physics.groucho.edu 的名字服务器的粘合记录见图 2.5 所示。

```
;
; 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
...
```

图 2.5 GMU 的 named.hosts 文件的摘录。

2.6.6 逆向查找（Reverse Lookups）

除了查找属于一个主机的 IP 地址以外，有时也需要找出一个与地址相应的正规主机名来。这称为*逆向映射（reverse mapping）*，被一些网络服务用于验证客户的身份。当使用单个 *hosts* 文件时，逆向查找只是简单地包括在文件中搜索拥有指定 IP 地址的主机。有了 DNS，对名字空间进行一次彻底的搜索当然就不可能了。取而代之的是，已经创建了一个特殊的域，in-addr.arpa，它以 reverted dotted-quad 表示法包括所有主机的 IP 地址。例如，与 IP 地址 149.76.12.4 相应的名字是 4.12.76.149.in-addr.arpa。连接这些名字到它们的正规主机名的资源记录类型是 PTR。

建立一个授权的区通常意味着给予它的管理员分配地址给名字的完全控制。由于他们手头上常常有一个或更多个 IP 网络或子网，在 DNS 区与 IP 网络之间是一个一对多的映射。例如，物理系是由子网 149.76.8.0、149.76.12.0、149.76.14.0 组成的。

作为结果，必须在 in-addr.arpa 域中随同物理区一起建立新的区并且委派该系的网络管理员为代表：8.76.149.in-addr.arpa、12.76.149.in-addr.arpa、以及 14.76.149.in-addr.arpa。否则的话在碰撞实验

室（Collider Lab）安装一台新主机就需要与他们的上级域联系，将这个新的地址加入他们的 in-addr.arpa 区文件中。

子网 12 的区数据库见图 2.6 所示。他们父辈区数据库中相应的粘合记录见图 2.7 所示。

```
;
; 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.
```

图 2.6 子网 12 的 named.rev 文件的摘录。

```
;
; 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
...
```

图 2.7 网络 149.76 的 named.rev 文件的摘要。

如此的一个重要结论是，只能以 IP 网络的超集来建立区，并且，更严重的是，这些网络的网络掩码必须以字节为界。Groucho Marx 大学的所有子网的掩码都是 255.255.255.0，据此，可以为每个子网建立一个 in-addr.arpa 区。然而，如果网络掩码是 255.255.255.128，那么为子网 149.76.12.128 建立区是不可能的，因为没有办法告知 DNS 12.76.149.in-addr.arpa 域已被分成了两个授权的区，两个区的主机名范围分别是 1 到 127，和从 128 到 255。

注释

[1] 通常，IP 地址是由你出资的 IP 连接提供者那里分配给你的。然而，通过给hostmaster@internic.net 发一个邮件，你也可以为你的网络直接向 NIC 申请一个 IP 地址。

[2] 然而，自治系统稍微更通用些，他们可以由多于一个 IP 网络组成。

[3] 许多人认为 routed 已停用。由于 geted 也支持 RIP，因此最好选用它。

[4] 如果它不这样，那么 DNS 就会象其它方法一样的糟糕，因为每次的查询都将包括根名字服务器。

[5] 唔，几乎是这样。名字服务器至少应该为 localhost 以及 127.0.0.1 的反向查找提供名字服务。

第三章 配置网络硬件

3.1 设备、驱动程序等等

直到现在，我们已经讨论了许多有关网络接口以及一般 TCP/IP 问题，但是并没有真正地包括当内核中的“网络代码”访问一个硬件时会确切地发生什么事情。对此，我们将简要地讨论一下有关接口和驱动程序的概念。

当然，首先是硬件本身，例如一块以太网卡：这是一块环氧树脂板，上面散布着许多有着糊涂数字的很小的芯片，这块板插在你的 PC 机的一个插槽中。这就是我们通常所称的设备。

为了让你能够使用这块以太网卡，你的内核中必须含有一些特殊函数，这些函数知道如何访问这个设备的特定方法。这些就是所谓的设备驱动程序。例如，Linux 有几种以太网卡牌子的设备驱动程序，它们在功能上很相似。它们以“Becker 系列驱动程序”而知名，与它们的作者同名，唐纳德贝克（Donald Becker）。另一个例子是处理连接至并行口的 D-Link 袖珍适配器的 D-Link 驱动程序。

但是，当我们说一个驱动程序“处理”一个设备是什么意思呢？让我们回到上面讨论的以太网板卡上。驱动程序必须能够与外围设备板上的某些逻辑电路通信：它必须往板卡上发送命令和数据，而板卡应该能够传递任何收到的数据给驱动程序。

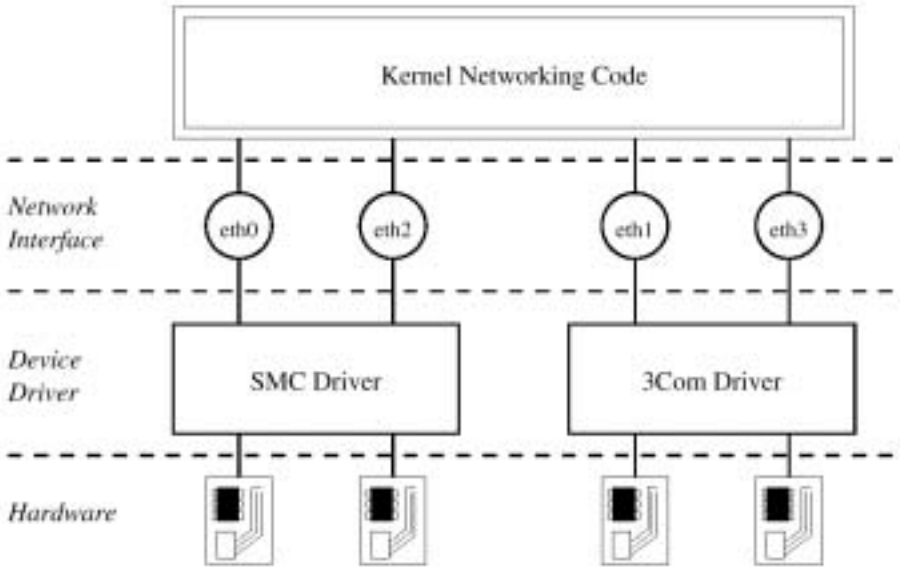

图 3.1 驱动程序、接口、以及硬件之间的关系。

在 PC 机中，通信是通过映射到板卡上的寄存器等的 I/O 存储器（内存）进行的。内核发送到板卡上的所有命令和数据都要经过这些寄存器。I/O 存储器通常是用给定的开始地址或*基地址（base address）*来描述的。以太网板卡的典型基地址是 0x300，或 0x360。

通常，你不用担心有关基地址之类的任何硬件问题，因为内核会在引导时试图检测到板卡上基地址的位置。这称为自动探测（autoprobing），它意味着内核读取几个存储器位置并且将读取的数据与如果安装了某个以太网卡所应有的数据作比较。然而，有些以太网卡不能够自动地被检测出来；

当使用其它生产厂商的不是完全按照标准板卡复制的便宜的以太网板卡时有时就会碰到这种情况。同样，在引导时，内核将只试图检测一块以太网设备。如果你使用了不止一块网卡，你就必须明确地将这块网卡告知内核。

你必须告诉内核的另一个这样的参数是中断请求通道（interrupt request channel）。当硬件部件需要得到照料时，常常中断内核的操作，例如，当数据来到时、或者一个特殊情况发生时。在一台 PC 机中，中断可以发生在 15 个中断通道之一上，编号从 0、1、以及 3 到 15。分配给一个硬件部件的中断号被称为是中断请求号（interrupt request number），或 IRQ。[1]

正如第二章中所述，内核是通过一个所谓的接口访问一个设备的。接口提供了一个对于所有硬件类型都一样的抽象的函数（功能）集，比如发送或接收一个数据报。

接口是通过名字识别的。这些名字是在内核里定义的，而不是在/dev 目录下的设备文件中定义的。对于以太网接口来说，典型的名字有 eth0、eth1 等等。对设备的接口分配通常依赖于配置设备次序；例如，第一块安装的以太网卡将成为 eth0，下一个将成为 eth1，等等。这个规则的一个例外是 SLIP 接口，它是动态分配的；也即，无论何时只要建立了一个 SLIP 连接，就为该串行端口分配一个接口。

图 3.1 中给出的图片试图显示出硬件、设备驱动程序以及接口之间的关系。

当引导时，内核会显示出检测到什么设备，以及它安装了什么接口。下面是一个典型引导屏幕的摘录：

```
.
.
This processor honours the WP bit even when in supervisor mode. Good.
Floppy drive(s): fd0 is 1.44M
Swansea University Computer Society NET3.010
IP Protocols: ICMP, UDP, TCP
PPP: version 0.2.1 (4 channels) OPTIMIZE_FLAGS
TCP compression code copyright 1989 Regents of the University of California
PPP line discipline registered.
SLIP: version 0.7.5 (4 channels)
CSLIP: code copyright 1989 Regents of the University of California
Dl0: D-Link DE-600 pocket adapter, Ethernet Address: 00:80:C8:71:76:95
Checking 386/387 coupling… Ok, fpu using exception 16 error reporting.
Linux version 1.1.11 (okir@monad) #3 Sat May 7 14:57:18 MET DST 1994
```

这表明内核是在 TCP/IP 激活的状态下编译的，并且包括了 SLIP、CSLIP 和 PPP 的驱动程序。倒数第三行说明检测到一个 D-Link 袖珍适配器，并作为接口 dl0 安装了。如果你有一块不同类型的以太网卡，内核通常将打印出以 eth0 开始的一行信息来，接下来是所检测到的卡的类型。如果已经安装了一块以太网卡，但是没有看到任何这样的消息，这表示内核不能正确地检测出你的网卡。这将在稍后的小节中加以讨论。

3.2 内核的配置

大多数的 Linux 版本带有引导启动盘，它可以在所有普通 PC 机类型的硬件上使用。这表明在那些盘片上的内核，配置进了所有种类的驱动程序，其中一些你永远不会用到，但是它们却会浪费宝贵的系统内存，因为部分内核是不能被交换出去的。因此，你通常将配置你自己的内核，只包括

那些你实际需要或想要的驱动程序。

当运行一个 Linux 系统时，你应该熟悉内核的建立。这个工作的基本原理在 Matt Welsh 的“安装与入门”手册中作出了解释，这本手册同样也是文档计划系列中的一本。因此，在本节中，我们将只讨论那些影响网络的配置选项。

当运行 **make config** 时，你首先将被问及一些普通配置，例如你是否需要内核数学仿真，等等。这些询问之一是问你是否要 TCP/IP 网络支持。你必须用 y 作答以得到一个支持连网的内核。

3.2.1 Linux 1.0 及以上版本的内核选项

在普通选项部分完成以后，配置将继续进行询问你各种特性，如 SCSI 驱动程序等等。随后列出了与网络支持有关的问题。由于正在进行中的研制开发，确切的配置选项集是在不停的变动中。大多数 1.0 到 1.1 内核版本所提供的一个典型的选项列表看上去象这样（注解用斜体给出）：

```
*
* Network device support
*
Network device support? (CONFIG_ETHERCARDS) [y]
```

不管方括号中宏名字（macro name）是什么，如果你想要使用任何类型的连网设备的话，你就必须用 y 回答这个问题，而不管这是否是以太网、SLIP、或 PPP。当用 y 回答这个问题时，对以太网类型设备的支持是自动激活的。对其它网络驱动程序类型的支持必须分别进行激活：

```
SLIP (serial line) support ?  (CONFIG_SLIP) [y]
SLIP compressed headers  (SL_COMPRESSED) [y]
PPP (point-to-point) support  (CONFIG_PPP) [y]
PLIP (parallel port) support  (CONFIG_PLIP) [n]
```

这些问题涉及到 Linux 支持的各种链路层协议。SLIP 使得你能够在串行线路上传输 IP 数据报。压缩首部选项提供了对 CSLIP 的支持，一种将 TCP/IP 首部压缩成只有三个字节的技术。这个内核选项不会自动打开 CSLIP，这仅仅为它提供所需的内核函数。

PPP 是另外一种在串行线路上传输网络通信量的协议。它比 SLIP 更灵活，而且并不限于 IP，一旦开发完成也将支持 IPX。由于 PPP 的支持只是在最近才完成，这个选项可能没有出现在你的内核中。

PLIP 提供了在并行端口连接上发行 IP 数据报的方法。它主要是用于与运行 DOS 的 PC 机进行通信。

下面的问题涉及到各个供应商的以太网卡。随着更多的驱动程序开发出来，你很可能看到更多的问题被加入这部分。如果你想建立一个能用于许多不同机器上的内核，你可以激活不止一个驱动程序。

```
NE2000/NE1000  support  (CONFIG_NE2000) [y]
WD80*3 support  (CONFIG_WD80x3)  [n]
SMC Ultra support  (CONFIG_ULTRA)  [n]
3c501 support  (CONFIG_EL1)  [n]
```

```
3c503 support (CONFIG_EL2) [n]
3c509/3c579 support (CONFIG_EL3) [n]
HP PCLAN support (CONFIG_HPLAN) [n]
AT1500 and NE2100 (LANCE and Pcnet-ISA) support (CONFIG_LANCE) [n]
AT1700 support (CONFIG_AT1700) [n]
DEPCA support (CONFIG_DEPCA) [n]
D-Link DE600 pocket adaptor support (CONFIG_DE600) [y]
AT-LAN-TEC/RealTek poket adaptor support (CONFIG_ATP) [n]
*
* CD-ROM drivers
*
…
```

最后，在文件系统部分中，配置脚本将询问你是否需要支持 NFS，网络文件系统。NFS 可以让你输出文件系统到几个主机上，这使得文件显现出好象它们是在主机的一个普通硬盘上一样。

```
NFS filesystem support (CONFIG_NFS_FS) [y]
```

3.2.2 Linux 1.1.14 及以上版本的内核选项

从 Linux 1.1.14 开始，（其中加入了对 IPX 的 alpha 支持），配置过程稍有改变。在普通选项部分中现在会问你是否需要通用的网络支持。这问题后立刻就是几个各种连网选项问题。

```
*
* Networking options
*
TCP/IP networking (CONFIG_INET) [y]
```

为了使用 TCP/IP 网络，你必须用 y 回答这个问题。然而，如果你回答 n，你将仍能够编译出支持 IPX 的内核。

```
IP forwarding/gatewaying (CONFIG_IP_FORWARD) [n]
```

你必须激活这个选项，如果你的系统将在两个以太网之间、或在任何以太网与 SLIP 链路等之间作为网关运行的话。尽管缺省地激活这个选项并无害处，你也许想要禁用这个选项以配置主机成为一个所谓的防火墙（firewall）。防火墙是连接两个或多个网络的主机，但并不在网络之间路由通信量。它们通常是用于为公司网络的用户以对内部网最小的危险性提供 Internet 的访问。用户将被允许登录到防火墙并且使用 Internet 服务，但公司的机器将受到从外界攻击的保护，因为任何进入的连接都不能通过防火墙。

```
*
* (it is safe to leave these untouched)
```

*

PC/TCP compatibility mode (CONFIG_INET_PCTCP) [n]

这个选项用于弥补与某些 PC/TCP 版本的不兼容性，一个用于基于 DOS 的 PC 机上的 TCP/IP 商业版本。如果你激活这个选项，你将仍然能够与正常的机器通信，但在慢连接上性能可能将受到影响。

Reverse ARP (CONFIG_INET_RARP) [n]

这个功能将激活 RARP，逆向地址解析协议。RARP 被用于无盘客户以及 X 终端，即用于在引导时查询它们的 IP 地址。只有在你计划想要为这些客户服务时才激活这个选项。最新的网络工具包（net-0.32d）中包括一个小工具名字为 rarp，它允许你将系统加到 RARP 缓冲中。

Assume subnets are local (CONFIG_INET_SNARL) [y]

当通过 TCP 传送数据时，内核必须在递送给 IP 之间将数据流分割成几个信息包。对于通过本地网络，如以太网，就能到达的主机，将使用大的信息包，而对于要通过很长距离连接的主机则使用稍小一些的信息包。[2] 如果你不激活 SNARL，内核将假设只有实际上有接口连接的网络才是本地的。然而，如果你考虑 Groucho Marx 大学的 B 类网络，那么整个 B 类网络是本地的，但是，大多数主机只有一个或两个子网接口。如果你激活 SNARL，内核将会假设当谈及校园网上的主机时所有的子网都是本地的并且都使用大的信息包。

如果你确实想使用最大包长度更小一些的包来传送数据到特殊的主机（例如，因为数据要经过 SLIP 连接），你可以使用 route 的 mtu 选项来做到，这在本章结束部分将概要讨论之。

Disable NAGLE algorithm (normally enabled) (CONFIG_TCP_NAGLE_OFF) [n]

Nagle 规则一个启发式的避免发送特别小的 IP 包，也成为微型报（tinygrams）。微型报通常是由交互式的网络工具创建的，它通常只传送单个击键，比如象 telnet 或 rsh。微型报在低带宽的连接上，如 SLIP，可能会变得特别浪费。Nagle 算法试图在某些情况下暂时抑制 TCP 数据的传输来避免这个问题。你只有在遇到严重的信息包丢失问题时才禁用 Nagle 算法。

The IPX protocol (CONFIG_IPX) [n]

这激活了对 IPX 的支持，--用于 Novell 网络的传输协议。这个功能仍在开发中，至今并没有什么用。使用它的好处是某一天你可以与基于 IPX 的 DOS 实用程序交换数据，并且通过一个 PPP 链接在基于 Novell 的网络之间路由通信量。对 Novell 网络高层协议的支持还没有眉目，因为这些协议的说明书只有付出可怕的高价并且在不能泄露的协议下才能得到。

从 1.1.16 内核开始，Linux 支持另外一种驱动程序类型，虚拟[伪]（dummy）驱动程序。下面的问题出现在设备驱动程序节的开始部分。

Dummy net driver support (CONFIG_DUMMY) [y]

虚拟驱动程序实际上并没有做很多事情，但对于单机（不连网的）或是 SLIP 主机来说是很有

用的。它基本上是一个伪装的回送接口。要有这类接口的原因是在有 SLIP 而没有以太网的主机上，你要有一个一直负担你的 IP 地址的接口。这将在第五章的虚拟接口一小节中进行进一步的讨论。

3.3 Linux 网络设备一览

Linux 内核支持许多各种设备类型的硬件驱动程序。这一节给出了现有驱动程序族的一个简要概述，以及它们所使用的接口名称。

在 Linux 中，接口有许多标准名称，现列在下面。许多驱动程序支持多个接口，在这种情况下，接口被编上了号，就如 *eth0*、*eht1* 等等。

lo 本地回送接口。它用于测试目的，以及几个网络程序。它工作起来象一个闭合电路，任何发送给它的数据报将被立即返回给主机的网络层。在内核中总有一个回送设备，并且有几个或多个几乎是没有意义的。

ethn 第 n 个以太网卡。这是大多数以太网卡的普通接口名。

dln 这些接口访问 D-Link DE-600 袖珍适配器，是另一种以太网设备。它有一个特别之处，就是 DE-600 是通过并行口驱动的。

sln 第 n 个 SLIP 接口。SLIP 接口以他们被分配给 SLIP 的次序，与串行线路相关；例如，配置成 SLIP 的第一条串行线路成为 *sl0*，等等。内核最多支持四个 SLIP 接口。

PPPn 第 n 个 PPP 接口。正如 SLIP 接口。一个 PPP 接口是与一条转换成 PPP 模式的串行线路相关联的。目前，最多支持四个接口。

plipn 第 n 个 PLIP 接口。PLIP 在并行线上传输 IP 数据报。同时可以支持三个 PLIP 接口。它们在系统引导启动时由 PLIP 驱动程序分配，并且被映射到并行端口上。

对于将来可能要加进来的其它接口驱动程序，象 ISDN、或 AX.25，将引进其他名称。IPX 的驱动程序（Novell 的网络协议），以及 AX.25（用于业余无线电爱好者）正处于开发阶段，仍处于 alpha 阶段。

在下面的部分中，我们将讨论使用上述驱动程序的细节。

3.4 以太网的安装

目前的 Linux 网络代码支持各种牌子的以太网卡。绝大多数的驱动程序是由 Danald Becker（becker@cesdis.gsfc.nasa.gov）编制的，他为基于国家半导体-8390 芯片的网卡编制了一族的驱动程序; 这些已经作为 Becker 系列驱动程序而知名。还有几个 D-Link 的产品的驱动程序，在其中，D-Link 的袖珍适配器允许你通过一个并行端口访问一个以太网。这个驱动程序是由 Bjørn Ekwall (bj0rn@blox.se)编制的。DEPCA 驱动程序是由 David C. Davies(davies@wanton.lkg.dec.com)编制的。

3.4.1 以太网电缆

如果你是平生第一次安装以太网的话，这里有关电缆连接的几句话也许是比较合适的。以太网连网对电缆是十分挑剔的。电缆的两端必须用 50 欧姆的电阻端接，而且不能有任何分支（也即，三根电缆连成星型）。如果你使用细同轴电缆和 T 型 BNC 接头，这些接头必须直接旋在网卡的接头上；你不能在其中插入一段电缆。

如果你连接到粗电缆网上，你必须通过一个收发器将主机连在网上（有时称作以太网附件单元）。你可以直接将收发器插入网卡的 15 针 AUI 端口上，但也可以使用一屏蔽电缆。

3.4.2 支持的板卡

在以太网 HOWTOS 中有已支持板卡的一个完整的列表，由 Paul Gortmaker 每月投递到 comp.os.linux.announce 上来。[3]

下面是 Linux 所支持的广为所知的板卡一张列表。在 HOWTO 中列表的实际长度大约有这个的三倍长。然而，即使你在这张列表中找到了你的网卡，还是要首先检查 HOWTO 文档；因为有时候它包括对这些网卡操作的重要的详细资料。一个相关的场合是某些基于 DMA 的以太网卡使用与 Adaptec 1542 SCSI 控制器的缺省 DMA 通道相同的 DMA，此时该以太网卡会将信息包数据往你的硬盘上乱写一气，使你神经紧蹦！☹

3Com EtherLink 3c503 和 3c503/16 两者都支持，同样也支持 3c507 和 3c509。也支持 3c501，但它太慢了，不值得去购买。

Novell Eagle NE1000 和 NE2000，以及各种各样的仿制品。也支持 NE1500 和 NE2100。

Western Digital/SMC WD8003 和 WD8013（与 SMC Elite 和 SMC Elite Plus 相同）两者都支持，同样支持新的 SMC Elite 16 Ultra。

Hewlett Packard HP 27252，HP 27247B，和 HP J2405A。

D-Link DE-600 袖珍适配器，DE-100，DE-200，和 DE-220-T。对 DE-650-T 还有一个补丁程序，这是一个 PCMCIA 卡[4]。

DEC DE200（32K/64K），DE202，DE100，和 DEPCA rev E。

Allied Teliesis AT1500 和 AT1700。

要在 Linux 中使用这些网卡中的一块，你可以从主要 Linux 发行版中使用一个预编译的内核。通常这些驱动程序都被编译进了内核。就长远观点来说，最好配置你自己的内核并且仅将你实际所使用的驱动程序编译进内核。

3.4.3 以太网卡的自动检测（探测）

在引导时，以太网程序代码将试图定位你的网卡并确定它的类型。网卡是以下面的地址和顺序进行检测的。

Board	Addresses probed for
WD/SMC	0x300, 0x280, 0x380, 0x240
SMC 16 Ultra	0x300, 0x280
3c501	0x280
3c503	0x300, 0x310, 0x330, 0x350, 0x250, 0x280, 0x2a0, 0x2e0
NEx000	0x300, 0x280, 0x320, 0x340, 0x360
HP	0x300, 0x320, 0x340, 0x280, 0x2C0, 0x200, 0x240
DEPCA	0x300, 0x320, 0x340, 0x360

自动检测代码存在两个局限性。第一，它不能够正确地识别出所有的网卡。尤其是对于那些非常便宜的刻隆（仿制）出来的大路货板子会识别不出来。有些 WD80x3 板子也检测不出来。第二个问题是内核此时不能自动检测多于一块的网卡。这是一个特色，因为内核假设你想控制哪一块卡被分配哪一个接口。

如果你正在使用多块网卡，或者自动检测没有检测出你的网卡，你必须把网卡的基地址和名称明确地告知内核。

在 Net-3 中，你可以有两种不同的方案来完成这个操作。一个方法是在内核原代码的包括所有有关驱动程序信息的 *drivers/net/Space.c* 文件中更改或增加信息。只有你是很熟悉网络代码时才推荐使用这个方法。更好的方法是在系统引导时给内核提供这些信息。如果你用 *lilo* 启动你的系统，你可以在 *lilo.conf* 中通过使用 append 选项指定这些参数来传递给内核。为了通知内核有关一个以太网设备，你可以传递以下的参数：

```
ether=irq, base_ addr, param1, param2, name
```

前四个参数是数字型的，而最后一个是设备名。所有数字型的值都是可选的；如果省略或被置零，内核将会试着去探测这些值，或使用一缺省值。

第一个参数设置分配给设备的 IRQ。默认地，内核将自动探测设备的 IRQ 通道。3c503 驱动程序有一个特殊的特性，它可以从 5、9、3、4 中断中选择一个未用的 IRQ，并配置网卡使用这个中断。

Base_addr 参数给出网卡的 I/O 基地址；零值表示要内核探测列于上面的地址。

剩余的两个参数对不同的驱动程序有不同的用途。对于共享内存的板子，如 WD30x3，它们指定共享内存区域的起始和终了地址。其它的网卡通常使用 *param1* 来设置要显示的调试信息的级别。值 1 到 7 表示逐步增加显示信息的长度，而 8 却是关闭所有信息的显示；0 表示使用缺省值。3c503 使用 *param2* 来选择内部收发器（默认的）或一个外部收发器（置为 1）。前者使用板上的 BNC 接头；后者使用板上的 AUI 端口。

如果你有两块以太网卡，你可以让 Linux 自动检测一块，而用 *lilo* 来传递第二块板的参数给内核。然而，你必须确信驱动程序没有意外地首先发现第二块板子，否则的话另一块网卡将完全不会被登记了。可以通过给 *lilo* 加上一个 *reserve* 选项，该选项明确地告知内核避免检测由第二块网卡所占用的 I/O 空间。

例如，为了让 Linux 将第二块以太网卡作为 *eth1* 安装在 *0x300*，你应该将下面的参数传给内核：

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

reserve 选项确定当检测某个设备时，没有设备驱动程序会访问这个板子的 I/O 空间。你也可以使用内核参数来覆盖对 eth0 的自动探测。

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

为了完全关闭自动检测，你可以指定 base_addr 为-1：

```
ether=0, -1, eth0
```

3.5 PLIP 驱动程序

PLIP 代表*并行线路 IP（Parallel Line IP）*,并且当你只想连接两台机器时，这是一个廉价的方法。它使用一个并行端口和一个特殊的电缆，能达到 10kBps 至 20kBps 的速率。

PLIP 最初是由 Crynwr 公司开发的。它的设计是相当直率的（ingenuous）(或者说是有创造性的)：长期以来，PC 机上的并行端口只用来作为单向打印机端口；也即，8 根数据线只能用来从 PC 机向外设发送数据，而不是反向使用。PLIP 通过使用端口的五根状态线作为输入来达到目的，这限制了它只能以半个字节（half bytes）来传输所有的数据。这个操作模式称为模式零 PLIP。今天，这些单向端口不再被广泛使用了。因此，还有一个 PLIP 扩展称为模式 1，它使用了全 8 比特的 接口。

目前，Linux 只支持模式 0。不象早期版本的 PLIP 代码，它现在试图与 Crynwr 的 PLIP 实现、以及 NCSA telnet 中的 PLIP 驱动程序相兼容。[5] 为了用 PLIP 连接两台机器，你需要一根特殊的电缆，在某些商店里作为“Null Printer”或“Turbo Laplink”出售的电缆。然而，你可以很容易地自己做出一根来。附录 A 示出了如何制作。

Linux 的 PLIP 驱动程序几乎是无数人的工作结果。它目前由 Niibe Yutaka 维护。如果把它编译进内核，它就会为每个可能的打印机端口设置一个网络接口，plip0 相应于并行端口 lp0、plip1 相应于 lp1 等等。目前接口到端口的映射是这样的：

Interface	I/O Port	IRQ
plip0	0x3BC	7
plip1	0x378	7
plip2	0x278	5

如果你已经将你的打印机配置成了不同的方式，你就必须改变在 Linux 内核原代码 *drivers/net/Space.c* 中的值，并建立一个新内核。

然而，这个映射并不是说你就不能象通常一样使用这些并行端口了。仅当相应的接口被配置成那样，它们才会被 PLIP 驱动程序访问。

3.6 SLIP 和 PPP 驱动程序

SLIP（串行线路 IP）（Serial Line IP），和 PPP（点对点协议）（Point-to-Point Protocol）是一种在串行线路链接上传输 IP-信息包的广泛使用的协议。许多机构提供 SLIP 和 PPP 拨号访问到互连网上的机器中，这样，为个人提供了 IP 连通性（这是用别的方法很难做到的）。

为了运行 SLIP 或 PPP，无需修改硬件；你可以使用任何一个串行端口。由于串行端口的配置是与 TCP/IP 连网无关的，因此将在独立的章节中专门讨论之。详细信息请参见第四章。

注释

[1] IRQs 2 和 9 是相同的，因为 PC 机有两个层叠的中断处理器，每个处理器有八个 IRQs；第二个处理器连接在第一个处理器的 IRQ-2 上。

[2] 这是为了避免具有很小最大包长度的连接产生的碎片。

[3] 可以用gpg109@rsphysse.anu.edu.au联系到 Paul。

[4] 它可以从 tsx-11.mit.edu 的 packages/laptops 上得到，连同其它一些与便携机有关的软件。

[5] NCSA *telnet* 是 DOS 的一个流行程序，在以太网上或 PLIP 上运行 TCP/IP，并且支持 *telnet* 和 FTP。

第四章 设置串行硬件

有这样的传言，在网络世界中有这样一些人，他们只有一台 PC 机并且钱去花在一根 T1-互连网连接上。然而，为了进行他们的新闻（news）和邮件（mail）的日常工作，他们说是用公共电话网络，依靠 SLIP 连接、UUCP 网络、和[电子]公告牌系统（bulletin board systems BBS's）来施行的。

本章打算帮助所有那些依靠 modem 来维持他们的连接的人。然而，有许多细节本章将不会加以讨论，例如，如何为拨入配置你的 modem。所有这些话题都会包括在 Greg Hankins 的即将发表的 HOWTO 系列中，[1] 它将定期地投递到 comp.os.linux.announce 上。

4.1 Modem 连接的通信软件

Linux 有许多通信软件包，其中许多是终端程序（terminal programs）,用以让一个用户拨接到另一台计算机上，就好象她正坐在一个普通的终端面前一样。传统的拨号终端程序是 kermit。然而，这个软件有些简单。现已有许多支持电话号码薄的、含有拨号和登录远程计算机系统脚本语言等等的更适用的程序。这种软件之一是 minicom，它与先前 DOS 用户可能很习惯的某些终端程序相近似。也有一些基于 X 的通信软件包，例如，seyon。

同样，也有许多基于 Linux 的 BBS 软件包，用于那些想运行[电子]公告牌系统的人。这些软件包有些可以在 sunsite.unc.edu 的*/pub/Linux/system/Network* 中找到。

除终端程序以外，还有一些非交互式地使用串行连接的软件，用于你的计算机收发数据。这种技术的优点在于能够比某些需要在线阅读邮件的 mailbox 程序和查找有趣文章而浏览公告牌所需的时间，花费更少的时间来自动下载几十 KB 的数据。另一方面来讲，因为你常常得到的一些无用信息的装入，需要更多的磁盘存储空间。

这类通信软件的摘要[体现]是 UUCP。这是从一台主机拷贝文件到另一台、在远程主机上执行程序等的一个程序组。它常用于在私人网络中传送 mail 或 news。能运行在 Linux 下的 Ian Taylor 的 UUCP 软件包将在后面章节中进行讨论。其它非交互式的通信软件是，例如，用于闻名于 Fidonet。也有象 ifmail 这样的 Fidonet 应用程序的端口。

SLIP，串行线路互连网协议，相对来说有些属于中间类型的，它允许交互式的或非交互式的使用。许多人使用 SLIP 拨号上到他们的校园网络或一些其他的公共 SLIP 网络服务器上来运行 FTP 会话等等。然而 SLIP 同样也能用于网到网的固定的和半固定的连接，尽管这实际上只对使用 ISDN 的才感兴趣。

4.2 串行设备概述

UNIX 内核为访问串行设备所提供的设备[驱动程序]典型地称为 *ttys*。这是 *Teletype* ™的缩写，它曾经是 UNIX 早期主要终端生产厂商之一。现今这个术语用于指任何字符型数据终端。贯穿本章，我们都将使用该术语专指内核设备[驱动程序]。

区别三种类型的 ttys：（虚拟）控制台、伪终端（类似于一个双向的管道，用于象 X11 这样的应用程序）、以及串行设备。后一种也称作 ttys，因为它允许在一个串行连接上进行交互式的会话操作；

而不管它是固定布线连接的终端，还是通过电话线的一个远程主机。

Ttys 有许多可配置的参数，这些参数可以使用系统调用 ioctl(2)来设置。这些参数中的许多只对串行设备有用，因为它们需要有非常大的灵活性来处理各种类型的连接。

最突出的线路参数是线路速率和奇偶性。但是同样还有大小写字符的转换标志、回车转换成换行等等。tty 驱动程序也可以支持各种*线路规范（line disciplines）*，它使得设备驱动程序的表现完全不同。例如，Linux 的 SLIP 驱动程序是按照特殊的线路规范实现的。

关于如何测试线路速度有些含糊不清。正确的术语是比特率，这是与用每秒比特数（或简写作 bps）测量的线路传输速度相关的。有时你听会到人们以*波特率（Baud rate）*来谈到它，这并不是很正确的。然而，这两个术语是不可互换的。波特率指的是某些串行设备的物理特性，也即发出脉冲的时钟速率。比特率则更恰当地表示了两点之间的一个已知连接的当前状态，也即每秒钟传输的平均比特数。知道这两个值通常是不同的很重要，因为大多数设备在每个电脉冲中起码多编码了一个比特。

4.3 访问串行设备

正象 UNIX 系统中的所有设备一样，串行端口是通过与设备相关的位于/dev 目录中的特定文件进行访问的。有两种与串行驱动程序相关的设备文件，而且对于每一个端口，都有这两种的一个设备文件。根据设备所访问的文件，设备的表现将有所不同。

第一种用于当端口用作拨入时；它有一个主号码 4，并且文件被命名为 ttyS0、ttyS1 等等。第二种用于当端口用作拨出时；文件被称为 cua0 等等，并且有个主号码 5。

次号码对两种类型是同样的。如果你在端口 COM1 到 COM4 中的一个上面连了个 modem 的话，那么它的次号码就是 COM 端口号加 63。如果你的设置与此不同，例如，当使用一块支持多串行线的板子时，请参阅 Serial Howto。

假设你的 modem 在 COM2 上。这样，它的次号码将是 65，对于是拨出时主号码将是 5。将会有一个设备文件 cua1 有这些号码。对/dev 目录中的串行 ttys 进行列表。第五列和第六列应该分别显示主号码和次号码：

```
$ 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
```

如果没有这样的设备，你就必须创建一个：作为超级用户，键入

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

有些人建议做一个符号联接*/dev/modem* 到你的 modem 设备，这样临时用户不需要记住这个不太直觉的 cua1。然而，你不能在一个程序中使用 modem 名称，而在另一个程序中使用实际的设备文件名。这是因为这些程序使用所谓的*锁定文件（lock file）*来通知该设备已被占用。按照惯例，cua1 的锁定文件，例如，是 *LCK..cua1*。给同一个端口使用不同的设备文件意味着程序将不能识别出其它的锁定文件，并且大家同时使用这个设备。结果，两个应用程序都完全不能工作。

4.4 串行硬件

目前 Linux 支持许多种类型的使用 RS-232 标准的串行板卡。RS-232 是目前 PC 世界串行通信中最通用的标准了。它使用了一些电路来传输单个比特数据和进行同步。另外一些连线可以用于载波信号（用于 modems）以及握手信号。

尽管硬件握手信号仅是供选用的，但是很有用的。它使得两个站点的任何一方都能通知对方它是否已准备好接收更多的数据，或者另一个站点是否将要暂停直到接收方已经处理完接收到的数据。用于此目的的连线分别称为“Clear to Send”（CTS）和“Ready to Send”（RTS），它说明了硬件握手信号的俗称，也即“RTS/CTS”。

在 PC 机中，RS-232 接口通常是由 UART 芯片驱动的，该芯片源自于国家半导体公司的 16450 芯片，或它的一个新版本 NSC16550A。[2] 有些牌子（多数显著的内置 modems 装配有 Rockwell 芯片集）也使用完全不同的芯片集，这些芯片被编程成好象它们就是 16550 芯片。

16450 与 16550 的主要不同之处在于后者有一个 16 字节的 FIFO 缓冲区，而前者只有 1 个字节的缓冲区。这使得 16450 适合于速度最高为 9600 Baud，而更高的速度需要一个与 16550 兼容的芯片。除了这些芯片以外，Linux 还支持 8250 芯片，该芯片是 PC-AT 的原始芯片。

在缺省配置情况下，内核检查四个标准的串行端口 COM1 至 COM4。次号码 64 到 67 将被分配给这些端口，正如上面所述。

如果你要正确地配置你的串行端口，你应该安装 Ted Tso 的 setserial 命令以及 rc.serial 脚本。这个脚本应该在系统引导启动时从/etc/rc 中调用。它使用 setserial 配置内核的串行设备。一个典型的 rc.serial 脚本看上去象这样：

```
# /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*
```

请参阅 *setserial* 附带的文档对其有关参数的解释。

如果你的串行卡没有被检测出来，或者 *setserial –bg* 命令显示出设置不正确，你必须通过明确地给出正确的参数值来进行强行的配置。报导称配备有 Rockwell 芯片的内置 modems 的用户会碰到这个问题。例如，如果 UART 芯片被检测出是 NSC16450，而实际上它是 NSC16550 时，你就必须将被错置端口的配置命令改成

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

```
uart 16550
```

对 COM 端口、基地址、以及 IRQ 的设置有相似的选择操作。请参阅 setserial 的手册页。

如果你的 modem 支持硬件握手的话，你必须确信激活了它。令人感到奇怪的是，多数通信程序缺省地并没有试图激活它；因而你必须手工地设置它。这最好使用 stty 命令在 rc.serial 脚本中进行操作：

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

要检查硬件握手是否起作用，使用

```
$ stty –a < /dev/cua1
```

这将给出那个设备所有标志的状态；如果标志前带有一个减号，如-crtscts，表示这个标志被关闭了。

注释

[1] 可以用gregh@cc.gatech.edu联系到。

[2] 也有一个 NSC16550，但它的 FIFO 从没有正常工作过。

第五章 配置 TCP/IP 网络

在本章中，我们将讨论在机器上设置 TCP/IP 网络所要经历的所有步骤。我们将从 IP 地址的分配开始，逐步描述 TCP/IP 网络接口的配置过程，并且介绍几个在解决网络安装问题时非常有用的工具。

本章所述的大多数工作通常你只需要做一次。而后，仅当你要向网络中增加新系统时，或者当你完全重新配置你的系统时，你才会接触许多配置文件。然而，有些用于配置 TCP/IP 的命令必须在系统每次引导时都要执行之。这通常是通过在系统的/etc/rc 脚本中调用它们来做的。

一般地，这个过程的网络专有部分包括在称为 rc.net 或 rc.inet 的脚本中。有时，你也会看到名为 rc.inet1 和 rc.inet2 的两个脚本文件，前一个用于初始化网络的核心部分，而后者启动基本的网络服务和应用程序。在本章下面，我将注重讨论后者的概念。

下面，我将讨论执行 rc.inet1 的作用结果，而应用程序将在后面章节中讨论。在读完本章以后，你将建立起在计算机上正确配置 TCP/IP 的一个命令顺序。然后，你应该替换计算机上 rc.inet1 里的任何例子命令、确信 rc.inet1 在系统引导时被执行，并重新引导你的机器。随着你中意的 Linux 版本而来的网络 rc 脚本会给你一个很好的例子。

5.1 安装 proc 文件系统

有些 Net-2 版本的配置工具要依赖 *proc* 文件系统来与内核进行通信。这是一个使用像文件系统似的机制以允许对内核运行时信息进行访问的接口。当加载时，你可以象使用任何其它文件系统一样地列出文件、或显示它们的内容。典型的项包括含有系统平均负载的 *loadavg* 文件、或显示当前核心内存以及交换使用情况的 *meminfo*。

对于此，网络代码添加了 *net* 目录。它含有许多文件，这些文件显示象内核 ARP 表格、TCP 连接状态，以及路由[选择]表。许多网络管理工具从这些文件中取得它们的信息。

Proc 文件系统（或者也以 procfs 而著称）通常在系统引导时被加载到/proc 目录上。最好的办法是将下面几行增加到/etc/fstab 之中:

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

并且从你的*/etc/rc* 脚本中执行“mount /proc”。

缺省地，现在 procfs 被配置进大多数的内核中。如果 procfs 不在你的内核中，你会得到象这样的消息:“mount: fs type procfs not supported by kernel”(“加载：内核不支持文件系统类型 procfs”)。你就必须重新编译内核并且在问及 procfs 支持时回答“yes”。

5.2 安装执行文件

如果你正在使用一个打包之前的（未打包的）Linux 发布版本，它将很可能包括主要的网络应

用程序以及实用工具和附带的一组例子文件。你可能必须得到和安装新工具的唯一情况是当你安装了一个新的内核版本。由于它们有时在内核的网络层中含有更改的情况，你将需要更新基本配置工具。这起码包括重新编译，但有时也可能需要获得最新的执行文件组。这些通常和内核一起发行，以文档的形式打包并切称为 net-XXX.tar.gz，这里 XXX 是版本号。与 Linux 1.0 相匹配的是 0 .32b，在本书写作时的最新内核（1.1.12 及以后）需要 0.32d。

如果你想自己编译并且安装标准 TCP/IP 网络应用程序的话，你可以从许多 Linux FTP 服务器上获得原程序。这些是或多或少从 Net-BSD 或其它原程序经过大量修正的程序版本。其它的应用程序，比如 Xmosaic、xarchie、或 Gopher 以及 IRC 客户程序必须分别地获得。其中大多数程序如果按照说明编译，则与盒装版本的一样。

官方的 Net-3 的 FTP 站点是 sunacm.swan.ac.uk，镜象站点在 sunsite.unc.edu 下的 system/Network/sunacm。最新的 Net-2e 补丁程序以及执行程序在ftp.aris.com。Matthias Urlichs 的起源于 BSD 网络代码的程序在ftp.ira.uka.de.in /pub/system/linux/上有。

5.3 另一个例子

在本书的余下部分，让我介绍一个新的例子，这个例子要比 Groucho Marx 大学的例子简单，并且可能更加接近你实际要遇到的问题。考虑虚拟酿酒厂（Virtual Brewery），一个小型的酿造—正如名称指出的那样--虚拟啤酒的公司。为了更有效地管理好他们的生意，虚拟酿酒人想要将他们的计算机连网，这些正好都是运行 bright and shiny 1.0（明亮与闪耀 1.0）的 PC 机。

在同一楼层上恰好穿过大厅的地方，有一家工作与之相近的葡萄酿酒厂。他们有一个自己的以太网。很自然地，一旦他们的网络开始正常运作时，这两家公司就想将他们的网络连接起来。作为第一步，他们想要设置一台用于在这两个子网之间转发数据报的网关主机。接下来，他们也想有一个与外界世界联系的 UUCP 链接，通过这个连接，他们就能交换邮件和 news 了。最后，他们也希望安装一个 SLIP 连接以便有时能连接到 Internet 上。

5.4 设置主机名（hostname）

绝大多数情况下—如果不是全部的话，网络应用程序要依赖于合理地设置本地主机名。这通常是在引导过程通过执行 *hostname* 命令来完成的。如要将主机名设置成 ***name*** 的话，它的调用如

```
# hostname name
```

使用一个与任何域名都无关的任意的主机名也是常有的事。例如，虚拟酿酒厂的主机可能叫做 vale.vbrew.com、vlager.vbrew.com 等等。这些是它们正式的、全资域名。它们的本地主机名将是这个名字的第一部分，如 vale。然而，由于本地主机名常常用来查找主机的 IP 地址，你就必须确信解析库能够查找到该主机的 IP 地址。这通常意味着你必须将这个名字写入*/etc/hosts* 中（见下面）。

有些人建议使用 *domainname* 命令来设置内核意义上的域名为 FQDN 的余下部分。这样的话，你可以组合 *hostname* 和 *domainname* 的输出来再次得到 FQDN。然而，这最多对了一半。Domainname 一般用来设置主机的 NIS 域，这可能与你的主机所属的 DNS 域完全不同。NIS 将在第十章中讨论。

5.5 分配 IP 地址

如果在你的主机上为单机操作配置连网软件（例如，为了能够运行 INN 网络新闻软件），你可以安全地跳过本节，因为你只需要为回送（loopback）接口分配一个接口，它总是 127.0.0.1。

而对于象以太网那样的真实网络来说，事情就有一些复杂。如果你想将你的主机连接到一个现成的网络上，你就必须要求该网的管理员给你一个这个网络的 IP 地址。当整个网络都是由你自己来设定时，你就必须按如下描述自己来分配 IP 地址。

在一个本地网络内的主机通常应该有相同的逻辑 IP 网络地址。因此，你必须分配一个 IP 网络地址。如果你有几个物理网络，你或者必须给它们分配不同的网络号（网络地址），或者使用子网技术将你的 IP 地址范围分割成几个子网。

如果你的网络没有连接到 Internet 上，你可以自由地选择任何（合法的）网络地址。你只须确信选择了 A 类、B 类或 C 类中的一种，否则的话事情可能会工作得不正常。然而，如果你打算近期连接 Internet，你现在就得获取一个正式的（官方的）IP 地址。进行的最佳方法是是请求你的网络服务提供商帮助你。如果你想获取一个网络地址只是因为万一某天你可能连上 Internet，从 hostmaster@internic.net要一张网络地址申请表。

为了操作几个以太网（或其它网络，一旦有了驱动程序），你必须将你的网络分隔成子网。注意，只有当你有多个*广播网络（broadcast network）*时，你才需要应用子网技术；点对点的连接不算在内。例如，如果你有一个以太网，并且有一个或多个 SLIP 链接到外部世界，你无需分割你的网络。其理由将在第 7 章中给出。

作为一个例子，酿酒厂的网络管理人员向 NIC 申请了一个 B 类网络号，得到 191.72.0.0。为适应两个以太网，她决定使用主机部分的 8 个比特作为额外的子网比特位。主机部分剩余的 8 个比特，允许每个子网有 254 台主机。然后，她将子网号 1 给了酿酒厂、子网号 2 给了葡萄酒厂。这样他们的网络地址分别为 191.72.1.0 和 191.72.2.0。子网掩码是 255.255.255.0。

在两个网络之间的网关 vlager，在它们两边都分配给它主机号 1，这分别给了它 IP 地址 191.72.1.1 和 191.72.2.1。图 5.1 示出了这两个子网和网关。

注意，在这个例子中，我使用了一个 B 类网络来使得事情简单化；而一个 C 类网络将更现实些。有了新的网络代码，子网的分割并不限于字节边界，所以，即使是一个 C 类网络也可以分割成几个子网。例如，你可以使用 2 比特的主机部分作为网络掩码，就可得到四个可能的子网，每个子网可以有 64 台主机。[1]

5.6 编写 *hosts* 和 *networks* 文件

在把你的网络分成子网以后，你应该使用*/etc/hosts* 文件为某些简单的主机名解析作些准备。如果你不打算使用 DNS 或 NIS 来作地址解析，你就必须将所有的主机名写入 *hosts* 文件中。

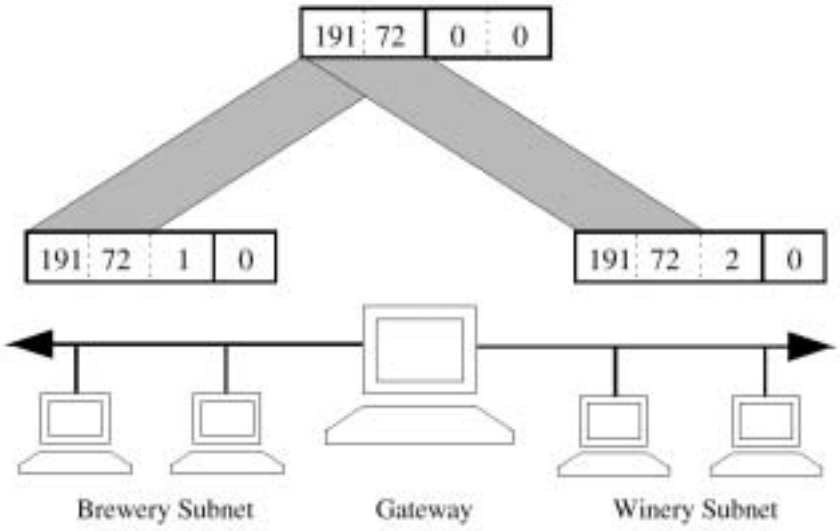

图 5.1 虚拟酿酒厂和虚拟葡萄酒厂—两个子网。

即使你在正常操作期间想运行 DNS 或 NIS，你仍然会将一些主机名写入*/etc/hosts* 文件中。一个原因是，即使没有网络接口正在运行，你还是会想有一些名字解析功能，例如，在系统引导期间。这不但是为了方便，而且也允许你在 *rc.inet* 脚本中使用符号主机名。这样，当更改 IP 地址时，你只需将更新过的 hosts 文件拷贝到所有机器上并且重新启动机器即可。而无需去分别编辑大量的 *rc* 文件。通常，你应该将所有本地主机名和地址放入 *hosts*，加入用到的任何网关和 NIS 服务器的名称和地址。[2]

同样，在最初的测试阶段，你应该确信你的解析器只使用到 *hosts* 文件中的信息。你的 DNS 或 NIS 软件可能附带有样本文件，当使用这些文件时就可能产生奇怪的结果。为了让所有的应用程序在查找一个主机的 IP 地址时只使用*/etc/hosts* 文件，你必须编辑*/etc/host.conf* 文件。用井字符注释掉以关键字 order 开始的所有行，并插入一行

```
order hosts
```

解析器库的配置将在第六章中详细讨论。

Hosts 文件中每一行包含一项，每项由一个 IP 地址、一个主机名、和任选的该主机的一个别名列表组成。各个域用空格或制表符分开，并且地址域必须从第一列开始。任何以井字符开始的行都看作是注释行，被忽略掉。

主机名可以是全资的、或者是与本地域相关的。对与 vale 来说，你通常可以往 hosts 文件中输入全资名 vale.vbrew.com，以及 vale 名本身，这样该主机就有正式名和短小的本地名两个名字了。

这是虚拟酿酒厂的 hosts 文件样式的一个例子。其中包括了两个特别的名字，**vlager-if1** 和 **vlager-if2**，它给出了 **vlager** 所用的两个接口的地址。

```
#
# 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
```

正如对待主机的 IP 地址一样，有时对于网络号你也可能想使用一个符号名称。因此，hosts 文件有一个相伴的称为/etc/networks 的文件，用于映射网络名到网络号以及反之。在虚拟酿酒厂，我们可以安装一个象下面的 networks 文件：[3]

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

5.7 IP 的接口配置

在如前章里解释那样设置好你的硬件以后，你必须让内核的网络软件知道这些设备。有几个命令是用于配置网络接口和初始化路由选择表的。这些任务通常是在系统每次启动引导时由 rc.inet1 脚本来做的。做这些任务的基本工具称为 *ifconfig*（这里“if”表示接口的意思 interface），和 *route*。

Ifconfig 用于使得一个接口能够被内核的网络层访问。这包括 IP 地址的分配和其它参数的指派、激活接口—也以“起用”（“taking up”）而知名。这里激活表示内核将通过接口发送和接收 IP 数据报。调用它的最简单的办法是

```
ifconfig interface ip-address
```

它分配 IP 地址 *ip-address* 给接口 *interface* 并激活它。所有其它的参数都设置成缺省值。例如，缺省的子网掩码是从 IP 地址的网络类型中获得的，如 255.255.0.0 是 B 类地址的掩码。*Ifconfig* 将在本章末给予详细的讨论。

Route 允许你对内核的路由表进行增加或删除路由的操作。它可以象这样调用

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

这里 add 和 dell 是决定增加还是删除到目的（target）网络或主机的路由。

5.7.1 回送（loopback）接口

最早被激活的接口是回送接口（loopback interface）：

```
# ifconfig lo 127.0.0.1
```

有时，你也会看到使用伪主机名 **localhost** 替代 IP 地址的用法。Ifconfig 将在 hosts 文件中查找这个名字，在 hosts 文件中应该有一项申明 localhost 是 127.0.0.1 的主机名：

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

要查看一个接口的配置，你要用接口名作为参数调用 ifconfig：

```
$ 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
```

如你所见，lookback 接口被赋予 255.0.0.0 掩码，这是因为 127.0.0.1 是一个 A 类地址。该地址没有一个广播地址集，但这在 loopback 中并没有什么用。然而，如果你在主机上运行了 rwhod 后台服务程序的话，你就必须设置 loopback 设备的广播地址，以便 rwho 能够正确地运行。如何设置广播地址会在下面的“关于 ifconfig”一节中描述。

现在，你几乎可以启动运行你的米你－“网络”了。最后所缺的是路由选择表里的一项，该项将告诉 IP，它可以使用这个接口作为到达目的 127.0.0.1 的路由。这是通过键入下面的命令来完成的

 # route add 127.0.0.1

再次，你可以使用 localhost 来代替这个 IP 地址。

接下来，你要检查是否每样事情正常工作着，例如通过使用 *ping* 来检查。*ping* 是声纳设备的网络等价物[4] 用于验证一个给定的地址是否真正能达到，并且测量发送一个数据报到给定地址然后返回到主机的延迟时间。所需的时间常称为来回程时间。

```
# 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
```

当如上调用 *ping* 时，它将继续不停地发送信息包，除非被用户中断掉。上面的^C 标记出我们按 Ctrl-C 的地方。

上面这个例子显示出 127.0.0.1 的信息包正确地发送了出去并且一个应答几乎是瞬时就返回到 *ping* 的。这表示你已经成功地设置好第一个网络接口。

如果你从 *ping* 得到的输出信息与不象上面所示的，你就碰到问题了。查看出错信息看看是否它指出了某些文件没有正确地被安装。查看你所使用的 *ifconfig* 和 route 执行文件是否与你运行的内核版本兼容，并且，最重要的是检查所编译的内核是网络使能的（激活的）（你可以从是否存在*/proc/net* 目录看出）。如果你得到出错信息说“网络不可达，”那么你很可能用错了 *route* 命令。请确信你使用了与给 *ifconfig* 相同的地址。

上面所描述的步骤对于在单机上使用网络应用程序来说已足够了。在将上面的几行添加到 *rc.inet1* 并确信这两个 *rc.inet* 脚本都已在*/etc/rc* 中执行以后，你可以重新引导你的机器并且试验各种应用程序。例如，“*telnet localhost*”将会建立一个到你主机的 *telnet* 连接，给你一个登录提示画面。

然而，loopback 接口是有用的，不仅是在网络书本中作为一个例子，或作为开发期间的一个测试台，而且实际上在正常操作时也被用于某些应用程序。[5] 因此，不管你的机器是否连接到一个网络上，你总是必须要配置它的。

5.7.2 以太网接口

配置一个以太网接口与配置 loopback 接口非常相象。它只是在你使用子网技术时需要稍多的参数。

在虚拟酿酒厂，我们已经将 IP 网络分割成了 C 类子网，这个 IP 网络原是一个 B 类网络。为了使得接口能够识别这个变化，*ifconfig* 的参数应该看上去象这样：

```
# ifconfig eth0 vstout netmask 255.255.255.0
```

它将 vstout（191.72.1.2）IP 地址分配给了 eth0 接口。如果我们已经省略了网络掩码，那么 ifconfig 将推论出该 IP 地址类的掩码来，它将会已有 255.255.0.0 这样的网络掩码。现在，一个快速查看显示出：

```
# 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
```

你可以看出，ifconfig 自动地将广播地址（上面的 Bcast 域）设置成常用的值，即将主机的网络号的主机部分的所有比特位置为 1 。同样，消息传输单元（message transfer unit）（内核为该接口生成的最大以太网帧的大小）已被设置成最大值 1500 字节。所有这些值都可以用特定的选项覆盖掉，这将在下面讨论之。

同 loopback 的情况非常相似，你现在必须安装一个路由选择项，它会通知内核有关通过 eth0 能够到达的网络。对于虚拟酿酒厂来说，你应该按如下调用 route

```
# route add –net 191.72.1.0
```

起先，这看上去有点象变魔术，因为并不清楚 route 是如何探测到要通过哪个接口进行路由。然而，窍门是很简单的：内核检测已配置的所有接口并且将目的地址（此时为 192.72.1.0）与接口地址的网络部分相比较（也即将接口地址和网络掩码进行逐位与）。只有 eth0 接口与之匹配。

现在，-net 选项是干什么用的呢？使用这个选项是因为 route 既可以处理路由到网络也可以处理路由到单机的情况（正如上面你已看到有关 localhost 的情况）。当给出的地址是用点分四组表示法表示时，route 通过查看主机部分比特位来试图猜测它是个网络号还是个主机地址。如果地址的主机部分是零的话，route 假定它表示一个网络，否则的话 route 把它当作一个主机地址。因此，route 会认为 192.72.1.0 是一个主机地址而不是一个网络号，因为它并不知道我们使用了子网技术。因此，我们必须给出-net 标志，明确地告之 route 它代表一个网络。

当然，上面的 route 命令键入时很冗长，并且很容易输错。一个更简便的方法是使用我们已经在/dev/networks 中定义的网络名字。这使得这个命令非常易读；而且也可以将-net 标志省略掉了，因为 route 现在知道 191.72.1.0 表示一个网络。

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

现在既然我们已经完成了基本的配置步骤，我们要确信该以太网接口确实能正常运行。从你的以太网络上任选一台主机，例如 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
```

如果你看不到任何象这样的输出，那么，很明显有地方出了问题。如果你碰到不正常的包丢失速率，这意味着是个硬件问题，如坏的或丢失了终端头等等。如果你一点也没收到任何信息包，你应该用 *netstat* 来检查接口配置。*Ifconfig* 的信息包统计信息会告诉你接口是否发送了任何信息包。如果你也可以访问那台远程的主机，你也应该过去检查它的接口的统计信息。用这种方法，你就可以正确地确定这些信息包是在哪丢失的。另外，你应该用 *route* 显示选路（路由选择）信息，来看看两台主机的路由项目是否正确。当不使用任何参数调用 *route* 时，会打印出整个内核选路表（-n 选项只是使得它用点分四组表示法打印出地址来，而不是用主机名）：

```
# route -n
Kernel routing table
Destination    Gateway        Genmask        Flags Metric Ref Use
```

```
127.0.0.1       *               255.255.255.255 UH    1      0
191.72.1.0      *               255.255.255.0   U     1      0
```

这些域的详细含义将在下面的“使用 netstat 检查”一节中给出。Flag 列包含每个接口标志的一个列表。U 对于活动的接口总是置位的，H 是指目的地址表示一台主机。如果 H 标志是为一个作为网络路由器的路由器置位的，那么你就必须为 *route* 命令指定-net 选项。为了测试你加入的一个路由器是否被使用，检查倒数第二列中的 Use 域在 *ping* 的对话期间会增加。

5.7.3 通过网关进行路由

在前一节中，我只描述了在单个以太网上设置一台主机的情况。然而经常性地，一个人会遇到网络通过网关连接到另一个网络的情况。这些网关可能只是简单地连接两个或多个以太网络，但也可能提供到外部世界的 Internet 连接。为了使用一个网关的服务，你必须在网络层提供额外的选路信息。

例如，虚拟酿酒厂和虚拟葡萄酒厂的以太网络是通过这样的一个网关，也即主机 vlager,连在一起的。假设 vlager 早已被配置好，我们只需在 vstout 的选路表中添加另一个项，这个项告诉内核，它可以通过 vlager 到达葡萄酒厂网络上的所有主机。适当的 route“咒语”显示如下；关键字 gw 告诉 route 下一个参数表示一个网关。

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

当然，你所希望对话的葡萄酒厂网络上的任何主机必须有一个酿酒厂网络的相应选路项，否则的话，你将只能将数据从 vstout 发送到 vbardolino，但后者的任何响应都将丢失（go into the great bit bucket）。

这个例子只描述了一个网关，它在两个隔离的以太网之间交换信息包。现在假设，vlager 也有一个到 Internet 的连接（例如，通过一个另外的 SLIP 连接）。那么我们希望除了到达酿酒厂的数据报，到达任何其他网络的数据报都将交由 vlager 处理。这可以通过使 vlager 成为 vstout 缺省的路由器来做到。

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

网络名 default 是 0.0.0.0 的缩写，它表示缺省路由器。你无需把这个名字添加到*/etc/networks* 中,因为它内建于 *route* 中。

当你 ping 一台隔着一个或几个网关的主机时看到很高的包丢失率，这可能意味着网络很拥挤。包丢失并不主要是技术上不足的原因，如由于转发主机暂时的超负荷运行，使得它们延迟甚至丢失了输入的数据报。

5.7.4 配置网关

配置一台机器使其在两个以太网之间交换信息包是非常直接明了的。假设我们回到 vlager 上，它配备了两块以太网卡，每块网卡连接一个网络。全部你所要做的是分别配置这两个接口，给它们

各自的 IP 地址，并且就这些了！

将两个接口的信息按如下方式添加到 hosts 文件中是非常有用的，我们也有唾手可得的名字：

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

设置这两个接口的命令序列也就成为：

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

5.7.5 PLIP 接口

当使用 PLIP 链接来连接两台机器时，事情就与使用以太网的稍微有些不同。前者是所谓的点对点的链接，因为相对于通信网络来说，它们只包括两台主机（“点”）。

作为一个例子，我们考虑虚拟酿酒厂的某个员工的膝上型电脑（笔记本电脑），它通过 PLIP 与 vlager 连接。笔记本电脑本身叫做 vlite，并且只有一个并行端口。在引导期间，这个端口将被注册为 plip1。为了激活这个连接，你必须使用下列命令配置这个 plip1 接口：[6]

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

第一个命令配置这个接口，告诉内核这是个点对点的链接，远端一边的地址是 vlager。第二个命令安装缺省路由器，vlager 用作网关。在 vlager 上，需用一个类似的 ifconfig 命令来激活连接（路由器启用是不需的）：

```
# ifconfig plip1 vlager pointopoint vlite
```

这里有趣的一点是 vlager 上的 plip1 接口不需要有一个独立的 IP 地址，而同样是地址 192.72.1.1。[7]

现在，我们已经配置好了从笔记本电脑到酿酒厂网络的路由；还缺的是酿酒厂的任何主机到 vlite 的路由的方法。一个特别笨的方法是在指定 vlager 为网关的各个主机的选路表中增加一特别的到 vlite 的路由项。

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

在面对临时路由的一个更好的选择是使用动态路由。这样做的一个方法是使用 gated，一个路由后台服务程序，它需要你安装在网络上的每台主机上来动态地发布选路信息。然而，最简单的方法是使用代理（proxy）ARP（地址解析协议），使用代理 ARP，vlager 将通过发出自己的以太网地址来响应任何对 vlite 的 ARP 请求。这个的作用是所有到 vlite 的信息包都送至 vlager，然后 vlager 将信息包转发到笔记本电脑。我们将在“检查 ARP 表”一节中再讨论代理 ARP。

以后发行的 Net-3 将包含一个称为 plipconfig 的工具，它允许你设置所使用的打印机端口的 IRQ。今后，这将由更通用的 ifconfig 命令来代替。

5.7.6 SLIP 和 PPP 接口

尽管 SLIP 和 PPP 链接只是简单的象 PLIP 一样的点对点连接，但对它们有很多要讨论的。通常，建立一个 SLIP 连接包括通过 modem 拨号到远程站点，并设置串行线路成 SLIP 模式。PPP 使用同样的方式。设置 SLIP 或 PPP 所需的工具将在第七章和第八章中描述。

5.7.7 哑（Dummy）接口

哑接口实在是有一些特殊的，但却是非常有用的。它的主要好处是，对于独立的主机以及那些仅有的 IP 网络连接是通过拨号连接才有的机器。实际上，后者在大多数时间也是一台独立的主机。

令独立主机进退维谷的是它们只有一个网络设备是激活的，即回送（loopback）接口，该接口通常分配了地址 127.0.0.1。然而在某些情况下，你需要将数据发送到“正式的”的本地主机 IP 地址上去。例如，考虑笔记本电脑 vlite，假设此时它没有连接任何的网络。Vlite 上的一个应用程序现在可能需要发送一些数据到同一个主机的另一个应用程序中。在/etc/hosts 中查看 vlite，找到它的 IP 地址是 191.72.1.65，所以这个应用程序试图往这个地址发送数据。由于回送接口是该机器上目前唯一活动的接口，所以内核就根本不知道该地址实际上就是自己的！作为结果，内核就会丢弃这个数据报，并且给应用程序返回一个出错信息。

这里就是哑设备需要起作用的地方。它通过简单地作为回送接口的一个密友来解决这个难题。对于 vlite 的情况，你只须简单地给它地址 191.72.1.65 并且加入一个指向它的主机路由。此时，到 191.72.1.65 的每个数据报都将在本地投递了。正确的调用是：

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

5.8 关于 ifconfig

ifconfig 还有比我们在上面所讨论的更多的参数。它通常的调用是：

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

interface 是接口名字，address 是分配给该接口的 IP 地址。它可以是一个点分四组表示的 IP 地址；或者是一个 ifconfig 可以在/etc/hosts 和/etc/networks 中查到的名字。-net 和-host 选项分别迫使 ifconfig 将地址作为网络号或主机地址来对待。

如果 ifconfig 只带一个接口名来调用，它就显示出该接口的配置。当不待任何参数来调用，它就显示你目前已设置的所有接口的配置；-a 选项迫使 ifconfig 同时也显示不活动的接口。对以太网接口 eth0 的一个调用样本可能看上去象这样：

```
# 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
```

MTU 和度量域显示出该接口的当前 MTU 和度量值。传统上，度量值被用于某些操作系统计算路由的代价。Linux 还没有使用这个值，但为了兼容性而定义了它。

RX 和 TX 行显示出有多少个包已被无错地接收和发送了、发生了多少个错误、有多少个包丢失了（通常是由于内存不足）、有多少个包由于超限而丢失了。接收器超限发生通常是由于到来的包的速率快于内核可以对最后一个中断的响应。Ifconfig 所打印出的标志值或多或少与它的命令行选项的名字相对应；它们将在下面给出解释。

下面是 ifconfig 所能识别的参数的一个列表，相应的标志名在括号中给出。简单地打开一个特性的选项同样也能在该选项的前面加上一短划（-）再来关闭这个特性。

Up

这标记一个接口为“up”，也即可以访问 IP 层。在命令行上给出地址时，就隐含了该选项。它也可以用于重新使能被 down 选项临时关闭的接口。（这个选项与标志 UP RUNNING 相对应。）

down

这标记一个接口为“down”，也即不可以访问 IP 层。这有效地禁止了通过该接口的任何通信。注意，这并不删除自动使用该接口的所有选路项。如果你要永远地停掉该接口，你应该删除这些路由项并且提供可能的其它选路信息。

Netmask mask

这指派了用于该接口的一个子网掩码。它可以以一个前面加有 0x 的十六进制的 32 比特数给出，或以点分四组十进制数给出。

Pointopoint

这个选项只用于包含两台主机的点对点 IP 链接。例如，这个选项需要被用于 SLIP 或 PLIP 接口的配置。（如果设置了一个点对点的地址，ifconfig 会显示 POINTOPOINT 标志。）

broadcast address

广播地址通常是通过将网络号的主机部分所有比特位置位产生的。有些 IP 实现使用一个不同的方案；这个选项是用于适应那些特殊的环境。（如果设置了一个广播地址，ifconfig 就显示一个 BROADCAST 标志。）

metric number

这个选项可用于为一个接口建立的选路表项分配一个度量值。这个度量用于路由信息协议（Routing Information Protocol RIP）为网络建立选路表。[8] ifconfig 所用的缺省的度量值是零。如果你没有运行一个 RIP 后台程序，你一点也不需要这个选项；如果你用了，你也很少需要改变度量的值。

Mtu bytes

这设置最大传输单元（Maximum Transmission Unit），这是接口能够在一次传输中处理的最大 8 比特组的数目。对于以太网，MTU 的缺省值是 1500；对于 SLIP 接口，它是 296。

arp

这个选项特别用于象以太网或无线电包的广播型网络的。它启动 ARP 的使用，地址解析协议，

来侦测连接到网络上的主机的物理地址。对于广播型网络，它总是缺省启用的。

-arp

在这个接口上禁止 ARP 的使用。

Promisc

将接口置为混合模式。在一个广播型网络上，这使得该接口接收所有的包，而不管它们是否是到达其它主机的信息包。这使得可以使用包过滤器等来分析网络交通流量，也称为以太网侦听。通常，这是查出用其它方法难以克服的网络问题的一个很好的技术。而另一方面，这也使得攻击者从你的网络流量中得到密码并且做其它肮脏的事。针对这类攻击的一个保护措施是不要让任何人可以随便将他们的计算机插入你的以太网中。另一个选择是使用安全认证协议（secure authentication protocols），如 Kerberos、或 SRA 登录组件。[9] （这个选项与标志 PROMISC 相对应。）

-promisc

禁止混合模式。

Allmulti

组播[多址通信，多点传送]地址是对无需在一个子网上的的一组主机的广播。目前，内核不支持组播地址。（这个选项对应于标志 ALLMULTI。）

-allmulti

禁止组播地址。

5.9 使用 netstat 检查

下面，我们转过来讨论一个对检查网络配置和行为很有用的工具。它称为 netstat 并且实际上是几个工具的汇总。我们将在下面几节中讨论它的每一个功能。

5.9.1 显示路由选择表

当使用-r 标志调用 netstat 时，它以我们在上面已经用 route 做过的方式显示内核选路表。在 vstout，它产生：

```
# 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 选项使得 netstat 以点分四组 IP 数的方式打印出地址，而不是用主机名或网络名。当你想避免在网络上查找地址时这是特别有用的（例如，对于一个 DNS 或 NIS 服务器）。

Netstat 输出的第二列示出了选路选项指向的网关。如果没用到网关，就只打印一个星号。列三显示出路由的“一般性”。当给定一个 IP 地址而要找出适当的路由，内核查询选路表的所有项，在与路由的目的地址比较之前将该地址与 genmask 进行**位与**（bitwise AND）。

第四列显示了描述路由的各种标志：

G　　路由使用了网关。
U　　所用的接口已启动。
II　　通过路由只能到达单个主机，例如，loopback 项 127.0.0.1 就是这种情况。
D　　如果该表项是由一个 ICMP 重定向消息产生的，就设置该标志（见 2.5 节）。
M　　如果该表项被一个 ICMP 重定向消息修改过，就设置该标志。

Netstat 输出的 Ref 列显示了对这个路由的参考数，也即，有多少个其它的路由（例如，通过网关）依赖于这个路由的存在。最后两列显示出了选路表项已被使用的次数，以及数据报通过其分发的接口。

5.9.2 显示接口统计信息

当带-I 标志调用时，netstat 将显示当前配置的网络接口的统计信息。如果再另加上一个-a 标志，它将打印出内核中存在的所有接口，不仅是那些目前已经配置了的。在 vstout 上，netstat 的输出看上去象这样：

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

MTU 和 Met 域那个接口的当前 MTU 和度量值。RX 和 TX 列显示有多少信息包是无错接收和发送的（RX-OK/TX-OK）、出错的（RX-ERR/TX-ERR）、丢失了多少包（RX-DRP/TX-DRP）、以及有多少包是由于超限而丢失的（RX-OVR/TX-OVR）。

最后一列显示出该接口被设置的标志。这些是当你用 ifconfig 显示接口配置的长标志名字的一字符版本。

B
　一个广播地址已被设置。
L
　这个接口是一个回送（loopback）接口。
M
　所有的包都将被接收（混合模式）。
N
　避免包尾。
O
　对于该接口，ARP 被禁止。
P
　这是一个点对点连接。
R
　接口正在运行。
U

接口被起用。

5.9.3 显示连接状态

netstat 支持一组选项来显示主动的或被动的套接字。选项-t、-u、-w 和-x 显示主动的 TCP、UDP、RAW 或 UNIX 套接字连接。如果你另外增加-a 标志，那么，等待连接的（例如，正在倾听的）套接字也被显示出来了。这将给出你的系统上目前运行着的所有服务器的一张列表。

在 vlager 上调用 netstat –ta 产生：

```
$ 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
```

这显示出大多数的服务器只是简单地在等待输入连接。然而，第四行显示从 vstout 来的一个 SMTP 连接，第六行告知你有一个输出的到 vbardolino 的 telnet 连接。[10]

使用-a 标志将显示出各类的所有套接字。

5.10 检查 ARP 表

在某些情况下，观察甚至改动内核的 ARP 表的内容是很有用的，例如，当你怀疑重复使用的 Internet 地址是某些间歇性的网络问题的原因时。*arp* 工具就是为此类事情而编制的。它的命令行选项是

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

所有的 hostname 参数可以是符号主机名或是以点分四组表示的 IP 地址。

第一个调用显示指定 IP 地址或主机的 ARP 项，或者如果没有给出 hostname 时显示所有已知的

主机的 ARP 项。例如，在 vlager 上执行 arp 会产生

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

它显示出了 vlager、vstout 和 vale 的以太网地址。

使用-t 选项你可以限制显示信息为指定的硬件类型的。这可以是 ether、ax25、或者是 pronet，分别表示 10Mbps 的以太网、AMPR-AX.25、和 IEEE-802.5 令牌环网。

-s 选项用于永久性地将 hostname 的以太网地址加入到 ARP 表中。Hwaddr 参数指定硬件地址，它缺省地指的是以太网地址、由冒号分隔的六个十六进制字节指定。使用-t 选项，你也可以为其它类型硬件设置硬件地址。

有一个问题，它可能需要你手工地将一个 IP 地址加入到 ARP 表中，是当由于某些原因对远端主机的 ARP 请求失败了的时候，例如当它的 ARP 驱动程序有错或网络上有另外一台主机错误地使用了那个主机的 IP 地址。ARP 表中硬配置的 IP 地址也是一个（非常强烈的）方法来保护你自己免受你的以太网上其它主机冒充别人的主机。

使用-d 开关调用 arp 将删除与给定主机相关的所有 ARP 项。这可以用于迫使该接口重新尝试获处理中的 IP 地址的以太网地址。这在当错误配置的系统已经广播了错误的 ARP 信息时很有用（当然，你要首先重新配置这个出错的主机）。

选项-s 也可以用于实现代理（proxy）ARP。这是一个特殊的技术，这里一个主机，比如说 gate，通过假装这两个地址引用同一台主机 gate，而为另一台名为 fnord 的主机担当一个网关。它是通过公布一个指向自己（gate）以太网接口的 fnord 的 ARP 项。现在，当一台主机发出一个对 fnord 的 ARP 请求时，gate 将返回一个包含自己以太网地址的响应。此时，发出请求的主机将会送出所有的数据报到 gate，gate 将有责任地转发这些数据报到 fnord。

这些带拐弯的处理有时是需要的，例如，当你想从一个不能很好地理解路由的 TCP 实现的 DOS 机器上访问 fnord 时。当你使用代理 ARP 时，对于 DOS 机器看来，fnord 好象救灾本地子网上，所以它就不需要知道如何路由通过一个网关了。

代理 ARP 的另一个非常有用的应用是，当你的主机之一对于某些主机只是临时充当一个网关时，例如，通过一个拨号链接。在前一个例子当中，我们已经遇到了偶尔通过 PLIP 连接 vlager 的笔记本电脑 vlite。当然，只有当你想要提供代理 ARP 的主机的地址与你的网关有相同的 IP 子网。例如，vstout 可以为酿酒厂子网（192.72.1.0）上的任何主机作为代理 ARP，但却决不能为葡萄酒厂子网（191.72.2.0）上的任何主机做代理 ARP。

为 fnord 提供代理 ARP 的正确的调用如下；当然，所给出的以太网的地址必须是 gate 的。

```
# arp -s fnord 00:00:c0:a1:42:e0 pub
```

代理 ARP 项也可以通过下面的调用移去：

```
# arp -d fnord
```

5.11 展望

Linux 的网络技术仍在发展。内核层中的主要改变将带来一个非常灵活的配置方案，它将允许你在运行时配置网络设备。例如，ifconfig 命令将含有设置 IRQ 号和 DMA 通道的参数。

另一个即将到来的变化是 route 命令的另加的 mtu 标志，它将可以针对特殊的路由设置最大传输单元。这个路由特有的 MTU 覆盖了接口指定的 MTU。你将典型地通过一个网关使用路由的这个选项，那里在网关和目的主机之间的连接需要一个非常小的（低的）MTU。例如，假设主机 wanderer 是通过一 SLIP 链接连接到 vlager 的。当从 vstout 发送数据到 wanderer 时，wanderer 上的网络层将使用最大 1500 字节的包，以为这些信息包是通过以太网发送的。而另一方面，SLIP 链接是以 296 大小的 MTU 工作的，所以 vlager 的网络层必须将 IP 包分解成更小的块以能适应 296 个字节。然而，如果你能配置 vstout 上的路由从一开始就使用 296 大小的 MTU 的话，这个相对耗费较大的分段处理就可以避免：

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

注意，这个 mtu 选项也允许你有选择地取消“子网是本地的”（“Subnets Are Local”）策略（SNARL）。这个策略是一个内核的配置选项并且在第 3 章中进行了描述。

注释

[1] 每个子网的最后一个号被保留用作广播地址，所以实际上每个子网只能有 63 台主机。

[2] 只有在使用了 Peter Eriksson 的 NYS 时，你才会需要所有 NIS 服务器的地址。其它 NIS 的实现仅通过使用 *ypbind* 在运行时间定位它们的服务器。

[3] 注意，networks 中的名字不能与 hosts 文件中的名字冲突，否则有些程序会产生奇怪的结果。

[4] 还有人记得 Pink Floyd 的“Echoes”吗？

[5] 例如，基于 RPC 的所有应用程序在启动时使用 loopback 接口在 portmapper 后台程序上注册它们自己。

[6] 注意，pointtopint 并不是打字错误。它实际上就是这样拼的。

[7] 然而作为一个警告，你应该只有在已经完全为你的以太网设置好选路表项以后，再配置 PLIP 或 SLIP 链接。否则，对于某些老内核，你的网络路由可能会指向点对点连接。

[8] RIP 对一给定的主机基于路径的“长度”选择一个优化的路径。它是通过将每个单独的主机到主机链接的度量值总和起来得出的。缺省地，一跳（hop）的长度是 1，但这个值可以是任何小于 16 的正整数。（一个长度为 16 的路径等于是不可达的。这样的路径被认为是没有用的。）度量参数设置这一跳的代价，然后它会被选路后台程序广播出去。）

[9] 可以从ftp.tamu.edu的/pub/scc/TAMU 中取得 SRA。

[10]你可以知道一个连接是否是输出的或者不是从调用的端口来的。呼叫的主机的端口号总是一个简单的数字，而在被呼叫的主机上，一个众所周知的服务端口将被使用，对于此端口，netstat 使用在/etc/services 中找到的一个符号名。

第六章 名字服务和解析器配置

正如第二章所述，TCP/IP 网络可以依赖于不同的方案来将名字转换成地址。如果不利用名字空间被分裂成区的好处的话，最简单的方法是用存储于*/etc/hosts* 中的主机表。这仅对由一个管理员管理的小型局域网有用，而且这个局域网要与外界没有 IP 通信。*hosts* 文件的格式早已在第 5 章中描述过。

作为选择，你可以使用 BIND—伯克里互连网名字域服务（Berkeley Internet Name Domain Service）--来解析主机名到 IP 地址。配置 BIND 可能很是繁杂，但是一旦你完成它，那么网络拓扑的变化很容易做到。在 Linux 上，正如在许多其它 UNIX 样的系统中，名字服务是通过一个称为 *named* 的程序提供的。在启动时，它将一组主要文件装入缓冲中，等待从远程或本地用户进程来的请求。设立 BIND 有不同的方法，并且不是所有的方法都需要你在每个主机上都运行名字服务器的。

虽然本章可以叙述的更详细一些，但却只是给出了如何操作一个名字服务器的粗略概况。如果你计划在不只是一个小型局域网并且可能有一个 Internet 连接的环境下使用 BIND，你应该针对 BIND 取得一本好书，例如 Crichet Liu 的“DNS 和 BIND”（见[AlbitzLiu92]）。对于当前的信息，你也可以查阅包括在 BIND 原程序中的发行注释（release notes）。还有一个 DNS 问题新闻组（newsgroup）称为 comp.protocols.tcp-ip.domains。

6.1 解析器库

当谈及“解析器”，我们并不是指任何特殊的应用程序，而是指*解析器库（resolver library）*，是一个能在标准 C 库中找到的函数的集合。主要的例程是 *gethostbyname(2)*和 *gethostbyaddr(2)*，它们查寻属于一个主机的所有的 IP 地址，并且反之也然。它们可以被配置成只是简单地在 *hosts* 中查询信息、请求名字服务器的一个数、或使用 NIS 的 *hosts* 数据库（Network Information Service）。其它的应用程序，象 *smail*，可能包括这些中的任何不同的驱动程序，并且需要特别的照料。

6.1.1 *host.conf* 文件

控制你的解析器设置的主要文件是 *host.conf*。它存储于*/etc* 中并且告知解析器使用哪个服务、以及用什么顺序。

Host.conf 中的选项必须出现在不同的行上。各个域要用空格（空格或制表符）隔离。一个“#”号表示一个注释行。

有以下一些选项：

order

这确定了解析服务试验的顺序。有效的选项是：bind 用于请求名字服务器、hosts 用于在/etc/hosts 中查找、nis 用于 NIS 查寻。可以指定其中的任何一个或所有。它们出现在一行上的顺序决定了各个相关服务试验的顺序。

multi

以 *on* 或 *off* 做为选项。这决定了在*/etc/hosts* 中的一个主机是否可以有几个 IP 地址，它通常指的

是作为“多宿主的”。这个标志对 DNS 或 NIS 请求是没有作用的。

nospoof

就如前章所解释的，DNS 通过使用 in-addr.arpa 域，允许你找到属于一个 IP 地址的主机名。名字服务器提供一个假主机名的企图被称为“哄骗”（“*spoofing*”）。为了防止这个做法，解析器可以配置成检查是否一个原始 IP 地址实际上是与一个获得的主机名相关的。如果不是，这个名字将被丢弃并且返回一个出错。这个行为是通过设置 *nospoof* 为 *on* 来打开的。

alert

这个选项使用 *on* 或 *off* 作为参数。如果它被打开，任何哄骗企图（见上面）将导致解析器将信息写进 *syslog* 日志文件中。

trim

这个选项将一个域名作为参数，在查寻之前它将被从主机名中删去。这对于 *hosts* 项是很有用的，那里你可能只想指定无本地域的主机名。附带有本地域名的一个主机的查寻将被移去本地域，这样就使得在*/etc/hosts* 中的查找获得成功。

Vlager 的一个样本文件显示如下：

```
# /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.
```

6.1.2 解析器环境变量

host.conf 中的设置可以通过使用环境变量来覆盖。这些环境变量是

RESOLV_HOST_CONF

这指定读一个文件而不是读*/etc/host.conf*。

RESOLV_SERV_ORDER

覆盖 host.conf 中给出的顺序选项。服务器以 *hosts*、*bind*、以及 *nis* 顺序给出，用空格、逗号、冒号、或分号来分隔。

RESOLV_SPOOF_CHECK

确定对待哄骗的方法。可以用 *off* 来完全禁止它。值 warn 和 warn off 启用哄骗检查，但分别打开或关闭日志。值*启用哄骗检查，但留下在 *host.conf* 中定义的日志选项。

RESOLV_MULTI

值 *on* 或 *off* 可用于覆盖 *host.conf* 中的 *multi* 选项。

RESOLV_OVERRIDE_TRIM_DOMAINS

这个环境变量指定了一个修整域的列表，它覆盖 host.conf 中给出的修整域。

RESOLV_ADD_TRIM_DOMAINS

这个环境变量指定了一个修整域的列表，它对 host.conf 中的修整域作了增加。

6.1.3 配置名字服务器查寻—*resolv.conf*

当配置解析器库以使用 BIND 名字服务进行主机查找，你也必须告知它使用哪个名字服务器。对此有一个独立的文件，称为 *resolv.conf*。如果这个文件不存在或是空的，那么解析器就假设名字服务器在你本地的主机上。

如果在你的本地主机上运行一个名字服务器，就象在下面一节中解释的那样，你必须单独地设置它。如果你在本地网络上并且有机会使用一个现存的名字服务器的话，这将总是一种推荐的做法。

resolv.conf 中最重要的选项是 *nameserver*，它给出了要使用的名字服务器的 IP 地址。如果你通过几次给出 *nameserver* 选项指定了几个名字服务器，那么它们会以给出的顺序试用。因此，你应该首先给出最可靠的服务器。目前，至多支持三个名字服务器。

如果没有给出 *nameserver* 选项，那么解析器试图连接本地主机上的名字服务器。

其它两个选项，*domain* 和 *search* 涉及到如果 BIND 不能用第一个请求解析主机名时附加在主机名上的缺省域。*search* 选项指定了一个试用的域名列表。列表项是用空格或制表符分开的。

如果没有给出 *search* 选项，就会通过使用域名本身从本地域名以及直至 root 的父域中建立一个搜寻列表。本地域名可以使用 *domain* 语句给出；如果一个也没有给出，那么解析器就通过系统调用 *getdomainname(2)*来获取。

如果这使得你感到混淆，考虑虚拟酿酒厂的 *resolv.conf* 样本文件:

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

当解析名字 vale 时，解析器将查询 vale，并且 vale.vbrew.com 和 vale.com 都会失败。

6.1.4 解析器的稳固性

如果你在一个大型网络中运行一个局域网，你无疑地应该使用主要名字服务器如果它们存在的话。这样做的优点是它们会有丰富的缓冲，因为所有的请求都转发给了它们。然而，这个方案也有缺点：当一把火毁坏了我校的主干网电缆时，我们系的 LAN 就做不了什么工作了，因为解析器不再能到达任何名字服务器了、在 X-终端上不再能登录了、也没了打印等等。

尽管校园主干网在火中焚毁不是常有的事，但我们必须针对这种情况采取防范措施。

一种选择是设置一个从本地域解析主机名的本地名字服务器，并且转发对其它主机名的所有请求到主服务器上去。当然，这只适用于你运行在自己的后台程序时。

另一种选择是，你可以在你的*/etc/hosts* 中维护一个你的域或 LAN 的一个备份主机表。然后，在*/etc/host.conf* 中要包含“order bind hosts”使得解析器在主名字服务器不在时后退到 *hosts* 文件。

6.2 运行 *named*

在大多数 UNIX 机器上提供域名服务的程序通常称为 *named*（发音为 name-dee）。这是一个最初为 BSD 开发的用于为客户提供名字服务的服务器程序，其它的名字服务器程序也可能是这样的。目前在多数 Linux 安装中所使用的版本好象是 BIND-4.8.3。现在 BIND-4.9.3 正处于 Beta 测试阶段，不久就会有了。

本节需要对域名系统工作原理有一些理解。如果下面的讨论你一点也看不懂，你可以重新阅读第 2 章，那里有关于 DNS 的更多的信息。

named 通常在系统引导时启动的，并且一直运行到机器再次关闭为止。它从一个称为 */etc/named.boot* 的配置文件中以及各种包含域名到地址映射数据的文件等中取得信息。后者称为*区域文件（zone files）*。这些文件的格式和语义将在下一节中描述。

要运行 *named*，只需在提示符下简单地键入

 # /usr/sbin/named

named 将启动，读取 named.boot 文件以及其中指定的任何区域文件。它将它的进程 id 以 ASCII 写入/var/named.pid 中、如有需要就从主服务器中下载任何区域文件，并且开始在端口 53 侦听 DNS 请求。[1]

6.2.1 *named.boot* 文件

named.boot 文件通常很小并且只包括指向含有区域信息的主文件的指针、以及指向名字服务器的指针。该 boot 文件中的注释行以一个分号开始一直延续到下一个新行开始。在我们详细讨论 named.boot 的格式之前，我们将看一下图 6.1 中给出的 vlager 的样本文件。[2]

```
;
; /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
```

图 6.1 vlager 的 named.boot 文件。

这个例子中的 *cache* 和 *primary* 命令将信息装入 *named*。这个信息是从第二个参数指定的主文件中取得的。它们含有 DNS 资源记录的文本表示，下面我们来看看。

在这个示例中，我们把 *named* 配置成三个域的主要（primary）名字服务器，就如该文件末尾的 *primary* 语句指出的那样。其中第一个 *primary* 语句指示 *named* 作为 **vbrew.com** 的主服务器，并从 *named.hosts* 文件中取得区域数据。*directory* 关键字告诉它所有的区域文件位于*/var/named* 目录中。

cache 是非常特殊的项并且实际上应该在所有运行名字服务器的机器上存在。它的功能有两个：它指示 *named* 激活它的缓冲，并从指定的缓冲文件中（在该示例中是 *named.ca*）装入*根名字服务器提示（root name server hints）*。我们将在下面回过来讨论名字服务器提示。

下面是用于 *named.boot* 中的非常重要的选项的一张列表：

directory 它指定了区域文件存储的目录。文件名可以用与该目录相关的形式给出。通过重复使用 *directory* 可以指定几个目录。根据 Linux 文件系统标准，这应该是*/var/named*。

primary 它将一个域名（*domain name*）和一个文件名（*file name*）作为参数，宣布了对命名的域的本地服务器的授权。作为一个主要服务器，*named* 从主文件中装入区域信息。一般地，在每个.boot 文件中总是有至少一个 *primary* 项，即用于网络 127.0.0.0 的逆向映射，该网络是本地回送（loopback）网络。

secondary 这个语句将域名（*domain name*）、地址列表（*address list*）和一个文件名（*file name*）作为参数。它为指定的域宣布本地服务器为二级主服务器。一个二级服务器同样也掌有该域的授权数据，但它不是从文件中得到的，而是试着从主服务器中下载来的。因此至少一个主服务器的 IP 地址必须在 *named* 的地址列表中给出。本地服务器将联系每个主服务器直到它成功地将区域数据库传输过来，这个区域数据库然后被保存在第三个参数指定的备份文件中。如果主服务器一个都没有响应，就从备份文件中取回区域数据。此后，*named* 将试图定期刷新区域数据。这将在下面连同 SOA 资源记录类型一起讨论。

cache 它将域（*domain*）和文件名（*file name*）作为参数。这个文件包含了根服务器提示，它是一个指向根名字服务器的一张列表。只有 NS 和 A 记录将被识别。*domain* 参数通常是根域名“.”。对 *named* 来说这个信息是绝对至关重要的：如果.boot 文件中没有 *cache* 语句，*named* 将完全不会产生一个本地缓冲。如果下一个服务器请求不在本地网上，这将严重地降低性能以及增加网络负荷。更为严重的是，*named* 将不能到达任何根名字服务器，因而，除了那些它授权的，它将不能解析任何地址。这个规则的一个例外是使用转发服务器（cf. 下面的 *forwarders* 选项）。

forwarders 这个语句将一个地址列表（*address list*）作为参数。这个列表中的 IP 地址指定了在 *named* 不能从本地缓冲中解析一个查询时可能会查询的名字服务器的一个列表。它们会被顺序地试用直到其中一个服务器对查询作出响应。

slave 该语句使得名字服务器成为一个从服务器。也即，它本身将永不执行递归查询，但只是将查询转发到 forwarders 语句指定的服务器上。

还有两个选项，sortlist 和 domain，我们在这里将不对它们进行讨论。另外，还有两个可以用于区域数据库文件中的指令。它们是$INCLUDE 和$ORIGIN。因为它们很少用到，这里我们将同样不对它们进行描述。

6.2.2 DNS 数据库文件

named 包括的主文件，如 *named.hosts*，总有一个域与它们相关联，称之为 *origin*。这是一个用 *cache* 和 *primary* 命令指定的域名。在一个主文件中，允许你指定与该域相关的域和主机名。在配置文件中的一个名字如果以一个点结尾就被认为是绝对的（*absolute*），否则的话它被认为是与 *orgin* 相关的。*orgin* 本身可以用“@”来引用。

一个主文件中包含的所有数据被分裂成*资源记录（resource records）*，或简称 RRs。它们构成

DNS 中的最小的信息单元。每一资源记录有一个类型。例如，A 记录将一个主机名映射到一个 IP 地址、一个 CNAME 记录将一个主机的别名与它的正式主机名相关联。作为一个示例，请观看 115 页是的图 6.3，它显示了虚拟酿酒厂的 named.hosts 主文件。

主文件中的资源记录表示法共享一个通用的格式，它是

[*domain*] [*ttl*] [*class*] *type rdata*

各域用空格或制表符分开。如果一个左大括号在第一个新行之前，一个项可以连续跨过几行，并且最后一个域后跟一个右大括号。在分号和新行之间的任何信息都被忽略。

domain 条目所适用的域名。如果没有给出域名，那么该 RR 假定应用于前一个 RR 的域。

ttl 为了迫使解析器在一段时间后放弃（废弃）信息，每个 RR 有一个相应的“存活期”（“*time to live*”），或简称 *ttl*。*ttl* 域用秒来指定从服务器取得的信息有效的时间。它是一个最多八位的十进制数。如果没有给出 ttl 值，那么它的值缺省为前面 SOA 记录的最小域的值。

class 这是一个地址类，正如 IP 地址的 IN，或 Hesiod 类中对象的 HS。对于 TCP/IP 网络来说，你必须使它为 IN。如果没有给出 class 域，就假定使用前一个 RR 的类。

type 它描述了 RR 的类型。最普通的类型是 A、SOA、PTR 以及 NS。下一节描述了 RR 的各种类型。

rdata 它保存与 RR 相关的数据。这个域的格式依赖于 RR 的类型。下面对每种 RR 进行单独的描述。

下面是用于 DNS 主文件的 RR 的一个不完整列表。还有许多 RR 的类型，我们将不作讨论。它们是试验性的，并且通常很少有用。

SOA 它描述了一个授权区域（SOA 是指“Start of Authority”）。它表示紧跟在 SOA RR 后面的记录包含有对该域的授权信息。每一个用 primary 语句包括的主文件在此区域必须含有一个 SOA 记录。资源数据含有以下几个域：

- *orign* 这是该域的主要名字服务器的规范主机名。它通常用一个完全的名字给出。
- *contact* 这是维护该域的负责人的 email 地址。用一个点替换掉了“@”字符。例如，虚拟酿酒厂域的负责人是 janet，那么这个字段域将含有 *janet.vbrew.com*。
- *serial* 这是一个用单个十进制数表示的区域文件的版本号数。每当区域文件中的数据改变时，这个数值将增加。
  这个序列（serial）号值是用于次要名字服务器识别何时区域信息改变了。为了保持最新，次要服务器在一定间隔时间后就请求主要服务器的 SOA 记录，并且将该序列号值与缓冲的 SOA 记录的序列号值相比较。如果数值改变了，次要服务器就从主要服务器将整个区域数据库传输过来。
- *refresh* 它指定了次要服务器将等待检查主要服务器的 SOA 记录的一个以秒计的间隔时间，同样地，这是一个最多八位的十进制数。通常网络的拓扑结构不会经常性地改变，所以对于大型网络来说，这个数应该指定为大约一天左右。

retry 这个数值确定了一个间隔时间，它是一个请求或一个区域刷新失败时次要服务器将重试与主要服务器联系的间隔时间。它不应该太小，否则一个临时的服务器失败或网络问题将导致次要服务器浪费网络资源。一个小时、或者一个半小时，可能是一个好的选择。

expire 它以秒指定了一个时间值，在这个时间过后，如果服务器还不能联系到主要服务器的话，它最终将丢弃所有的区域数据。这个时间值通常应该很大。Graig Hunt（[GETST 92]）推荐为 42 天。

minimum 这是对于没有明确指定 ttl 的资源记录的一个缺省的 ttl 值。这需要其它的名字服务器在一段时间后丢弃 RR。然而，它与次要服务器开始试图更新区域信息所等待的时间无关。Minimum 应该是一个很大的值，尤其是对于网络拓扑结构几乎不变的局域网。一个大约是一周或一个月的值大概是一个很好的选择。对于单个 RR 可能经常改变的情况，你仍可以给它们一个不同的 ttl 值。

A 这将一个 IP 地址与一个主机名相关联。资源数据域包含点分四组表示的地址。对于每一台主机，只能有一个 A 记录。用于这个 A 记录的主机名被认为是正式的或规范的主机名。所有其它主机名都是别名，并且必须使用一个 CNAME 记录映射到这个规范的主机名上。

NS 这指向一个从属区域的主名字服务器。对于为什么要有 NS 记录的解释，请参见 3.6 节。资源数据域包含名字服务器的主机名。为了解析这个主机名，另外需要一个 A 记录，即给出名字服务器 IP 地址的所谓的粘合记录。

CNAME 它将一个主机的别名与它的正规（或规范）主机名相关联。规范主机名就是主文件为其提供一个 A 记录的主机名；别名只是通过一个 CNAME 记录简单地联结到规范主机名上，但它们本身没有任何其它的记录。

PTR 这个类型的记录用于将 in-addr.arpa 域中的名字与主机名相关联。这用于 IP 地址到主机名的逆向映射。所给出的主机名必须是规范主机名。

MX 这个 RR 对于一个域申明了一个*邮件交换器（mail exchanger）*。有邮件交换器的原因将在第 13 章的“互连网上的邮件路由”一节中讨论。MX 记录的句法是

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

host 为 ***domain*** 命名邮件交换器。每个邮件交换器有一个整数 ***preference***（优先权）与之关联。一个想要将邮件分发到 ***domain*** 的邮件传输代理会试用所有针对这个域的 MX 记录的主机，直到成功为止。具有最低优先权值的将首先试用，然后按着优先权值的增大顺序试用其它主机。

HINFO 这个记录提供了有关系统硬件和软件的信息。它的句法是

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

hardware（硬件）字段确定主机所使用的硬件。存在特别的约定来指定它。在“Assigned Numbers”（RFC 1340）中给出了一张有效名字的列表。如果字段中含有任何空格字符，就必须用双引号括住。***Software***（软件）字段指定系统所用的操作系统软件。同样，应该选用“Assigned Numbers”RFC 中的有效名字。

6.2.3 编写主文件

图 6.2、6.3、6.4 和 6.5 给出了在酿酒厂 *vlager* 上的一个名字服务器的样本文件。考虑到所讨论网络的特性(单个局域网),这个例子是非常直观的。如果你的要求更复杂些,并且你运行不了 *named*,参见 Cricket Liu 和 Paul Albitz（[AlbitzLiu92]）的“DNS and BIND”。

图 6.2 示出的 *named.ca* 缓冲文件显示出一个根名字服务器的样本提示记录。一个典型的缓冲文件通常描述了大约一打左右的名字服务器。你可以使用本章末描述的 *nslookup* 工具获得根域的当前名字服务器的列表。[3]

```
;
; /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
```

图 6.2 named.ca 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
```

图 6.3 named.hosts 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.
```

图 6.4 named.local 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.
```

图 6.5 named.rec 文件

6.2.4 验证名字服务器的设置

有一个很好的工具用于检查名字服务器设置的操作。它称为 *nslookup*，即可以交互式地使用，也可以从命令行上使用。在后一种情况下，你只需象这样简单地调用它

nslookup *hostname*

它将对 *hostname* 查询 *resolv.conf* 中指定的名字服务器。（如果这个文件指定了多个服务器，*nslookup* 将随机地选择一个）。

然而，交互模式则更令人兴奋。除了能查询单个的主机，你可以查询 DNS 记录的任何类型，并且传输一个域的整个区域信息。

当不加参数地调用，*nslookup* 将显示它所用的名字服务器，并且进入交互模式。在'>'提示符下，你可以键入任何想要查询的域名。缺省地，它请求类 A 记录，这些是包含与域名相关的 IP 地址的。

你可以通过发出“set type=*type*”来改变这个类型，这里 type 是上面 6.2 节中描述的资源记录名，或 ANY。

例如，你可以与它进行下面的对话：

```
$ 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
```

如果你试者去查询一个没有相应 IP 地址的名字，但 DNS 数据库中能找到其它的记录，nslookup 将返回一个错误信息说“No type A records found”（“没有类型 A 记录发现”）。然而，你可以通过发出“set type”命令来查询不是类型 A 的其它记录。例如，要得到 unc.edu 的 SOA 记录，你要发出：

```
> 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
```

以同样的方式你可以查询 MX 记录，等等。使用一个 ANY 类型将返回与一个给出的名字关联的所有资源记录。

```
> 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
```

除了调试，*nslookup* 的一个实际应用是为 *named.ca* 文件获取根名字服务器的当前列表。你可以通过查询与根域相关的所有 NS 类型记录来做到：

```
> 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
```

nslookup 完整的命令集可以通过 *nslookup* 中的 **help** 命令得到。

6.2.5 其它有用工具

还有几个工具能帮助你完成作为一个 BIND 管理员的任务。这里我将概要地描述其中的两个。请查阅这些工具的文档来获取如何使用它们的信息。

hostcvt 通过将你的*/etc/hosts* 文件转换成 *named* 的主文件来帮助你进行初始 BIND 配置。它产生前向（A）和逆向映射（PTR）条目，并且管理别名等。当然，它不可能为你做整个工作，正如你仍想要，例如，调整 SOA 记录的超时值、增加 MX 记录等等这类工作。不过，它仍能帮助你节约几片阿司匹林。*hostcvt* 是 BIND 原代码的一部分，但是你也能在几个 Linux FTP 服务器上找到作为一个独立的软件包的。

在设置好你的名字服务器以后，你可能想测试一下你的配置。做这件事的理想的（并且，就我所知而言）唯一的工具是 dnswalk，一个基于 perl 的软件包，它通览你的 DNS 数据库，查找常见的错误并且验证信息的一致性。近来，dnswalk 已在 comp.sources.misc 上发布，并且将会在所有对该组存档的 FTP 站点上存储(如果你不知道靠近你的任何这样的站点，ftp.uu.net将是一个可靠的地方)。

注释

[1] 在 Linux 的 FTP 站点上有各种 *named* 执行文件，每个的配置稍有不同，有些将它们的 pid 文件放在*/etc* 中，有些将 pid 文件存储在*/tmp* 或*/var/tmp* 中。

[2] 注意，这个例子中的域名是以无轨迹点给出的。Named 的早期版本好象将轨迹点当作错误看待，并且悄无声息地舍弃了该行。BIND-4.9.3 据称以修正。

[3] 注意，如果没有安装任何根服务器提示，你就不能向你的名字服务器查询根服务器。Catch-22! 为了避免这个难题，你可以或者使得 *nslookup* 使用一个不同的名字服务器，或者你可以使用图 6.2 中的样本文件作为一个起点，然后获取一张有效服务器的完全列表。

第七章 串行线路 IP

串行线路协议 SLIP 和 PPP 为资金缺乏者提供 Internet 连接。除了需要一个 modem 和一块配有 FIFO 缓冲的串行板（卡）外，不再需要其它的硬件了。使用它并不比使用一个邮箱复杂，并且不断增长的私人机构以可以接受的价格为大家提供拨号上网 IP。

Linux 有 SLIP 和 PPP 两种驱动程序。SLIP 已存在相当长的时间了，并且工作的很稳定。PPP 驱动程序是由 Michael Callahan and Al Longyear 最近开发的。PPP 将在下一章中讨论。

7.1 一般需求

要使用 SLIP 或 PPP，你当然必须配置一些如前章所讨论的那些基本网络特性。起码，你必须设置回送（loopback）接口，并且提供名字解析。当连接到 Internet 上时，你当然要用到 DNS。最方便的方法是将某个名字服务器的地址写入 resolv.conf 文件中；SLIP 链接一旦被激活，就会查询这个服务器。名字服务器离你拨入的地方越近越好。

然而，这个方法并不是最佳的，因为所有的名字查找仍然都要通过你的 SLIP/PPP 链接。如果你担心这样做所耗费的带宽，你也可以设置一个*只缓冲（caching-only）*名字服务器。它并不真的服务于一个域，而只是作为你的主机所产生的所有 DNS 查询的一个中继。这个方案的优点在于它建立了一个缓冲，大多数的查询只需往串行线路上发送一次。一个只缓冲服务器的 named.boot 文件看上去象这样：

```
; 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
```

除了这个 name.boot 文件，你也还需要设置 db.cache 文件，其中含有有效根名字服务器的一张列表。这些在解析器的配置一章中的最后讨论。

7.2 SLIP 操作

拨号（上网）IP 服务器通常使用特殊的用户帐号提供 SLIP 服务。在登陆进这样一个帐号以后，你不会进入普通的 shell 程序；而是执行一个程序或 shell 脚本，激活串行线路的服务器 SLIP 驱动程序并且配置适当的网络接口。然后你需要在链接的你这端做同样的工作。

在某些操作系统上，SLIP 驱动程序是用户空间程序；在 Linux 下，它是内核的一部分，这使得它更快一些。然而，这需要将串行线路明确地转换成 SLIP 模式。这是通过特殊的 tty 线路规程，SLIPDISC，来做到的。当 tty 为普通线路规程（DISCO）时，它将使用普通的 read(2)和 write(2)调用只与用户进程交换数据，而 SLIP 驱动程序不能从 tty 读取或写进数据。在 SLIPDISC 规程里，规则

正好相反：现在任何用户空间进程从 tty 的读取和写入被阻止，此时，从串行端口来的所有数据将被直接地传递到 SLIP 驱动程序。

SLIP 驱动程序本身可以识别 SLIP 协议的多种变化。除了普通的 SLIP，它也能理解 CSLIP，这在输出的 IP 包上执行所谓的 Van Jacobson 头压缩。[1] 这显著地改进了交互式会话的吞吐量。另外，对于这些协议的每一种有六比特的版本。

将串行线路转换为 SLIP 模式的一种简单方法是通过使用 *slattach* 工具。假设你的 modem 连在 /dev/cua3 上，并且成功地登录上 SLIP 服务器。然后执行：

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

这将 cua3 的线路规程转换成 SLIPDISC，并把它连接到 SLIP 网络接口之一。如果这是你的第一次激活的 SLIP 链接，该线路将连至 sl0；第二个将连至 sl1，以此类推。目前的内核同时支持多达八个 SLIP 链接。

Slattach 所选择的缺省的压缩封装是 CSLIP。你可以使用-p 开关来选择任何其它的模式。要使用常规的 SLIP（无压缩的），可以使用

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

其它的模式有 cslip、slip6、cslip6（SLIP 的六比特版本）、以及适应性 SLIP（adaptive SLIP）的 adaptive。后者让内核找出远端所用的是那一种 SLIP 压缩封装类型。

注意，你必须使用与你的对等点同样的压缩封装。例如，如果 cowslip 使用 CSLIP，你也必须使用它。如果选择不匹配，那么将会出现 *ping* 到远程主机将收不到任何返回信息包的现象。如果其它的主机 *ping* 你，那么在你的控制台上也会出现象“不能建立 ICMP 头”（“Can’t build ICMP header”）的信息。避免这些问题的一种方法是使用适应性 SLIP。

实际上，*slattach* 并不仅仅允许你使能 SLIP，同样也可以激活串行线路的其它协议，如 PPP 或 KISS（另一个由无线电爱好者使用的协议）。详细信息，请参考 *slattach(8)*再线手册。

在将线路转至 SLIP 驱动程序以后，你必须配置这个网络接口。再一次，我们使用标准的 *ifconfig* 和 *route* 命令来做这个配置。假设从 vlager，我们拨号到一个名为 cowslip 的服务器。那么你要执行

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

第一个命令将接口配置成到 cowslip 的点对点链接，而第二、第三个命令增加到 cowslip 的路由以及使用 cowslip 作为一个缺省的网关。

当拆卸一个 SLIP 链接时，你首先必须使用带 del 选项的 *route* 命令移去所有通过 cowslip 的路由，将接口关闭，并向 *slattch* 发送一个 hangup 信号。然后，你必须再次使用你的终端程序挂断 modem：

```
# route del default
# route del cowslip
# ifconfig sl0 down
# kill -HUP 516
```

7.3 使用 *dip*

现在来看，上面讲的是非常简单的。然而，你也许想使上面的步骤自动地执行，这样你就可以只调用一个简单的命令来执行上面给出的所有步骤了。这也就是 *dip* 所要做的。[2] 在写作这本手册时它的当前发行版本是 3.3.7。它已被许多人大大地修改过了，所以它已不再是原来的 dip 程序了。这些不同的开发变化有希望在今后的版本中合并。

Dip 为简单的脚本语言提供了一个解释器，它能为你处理 modem，将线路转变为 SLIP 模式，并配置接口。这是非常基本的和有局限性的，但对于大多数情况已足够有效了。某天一个新的 *dip* 版本将能适用于更为广泛的语言。

为了配置 SLIP 接口，dip 需要 root 权限。现在可以临时使用 dip 将 uid 置为 root，因此所有的用户能够拨号到某个 SLIP 服务器而不需要给这些用户 root 权限。这是非常危险的，因为用 dip 设置假的接口和默认路由可能会严重地破坏网络的路由。更糟的是，这将给你的用户连接到任何 SLIP 服务器的能力，并在你的网络上带来危险的攻击。所以如果你想允许你的用户建立一个 SLIP 连接，就为每个期望的 SLIP 服务器写一个小的包装程序，并且让这些包装程序调用包括建立连接用的特定脚本的 dip。那么，这些程序就可以安全地置成 root 的 uid。[3]

7.3.1 一个简单的脚本程序

图 7.1 列出了一个简单的脚本程序。通过用脚本程序的名字作为参数调用 dip，它可以用于连接 cowslip：

```
# 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
#
```

当激活了 SLIP 并连接到 cowslip 以后，dip 将从终端上脱开并运行于后台。此时你就可以在 SLIP 连接上使用通常的网络服务了。要想终止连接，只需用-k 选项调用 dip。这使用/etc/dip.pid [4]中记录的进程 id *dip* 给 dip 进程发送了一个挂断信号：

```
# kill –k
```

在 dip 的脚本语言中，前加美元符号的关键字表示变量名。Dip 有一个预定义的变量集，将在下面列出。例如，$remote 和$local 含有与 SLIP 连接有关的本地以及远程主机的主机名。

```
# 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 41988
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 Svlager\n
wait ssword: 5
if $errlvl != 0 goto error
send hey-jude\n
wait running 30
if $errlvl != 0 goto error

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

error:
print SLIP to $remote failed.
```

图 7.1 一个 *dip* 脚本样本

样本脚本程序中的头两句是 *get* 命令，这是 dip 设置变量的方法。这里，本地和远程主机名分别设置成了 vlager 和 cowslip。

接下来的五句用来设置终端线路和 modem。Reset 向 modem 发送了一个复位串；对于 Hayes 兼容的 modem，这是个 ATZ 命令。下一个语句刷出 modem 的响应，以使得接下来的几行中的登录会话能够正常的工作。这个对话是非常直观的：它简单地拨出 cowslip 的号码 41988，并且使用口令

hey-jude 登录进 Svlager 帐号。Wait 命令使得 dip 等待给出的字符串作为它的第一个参数；作为第二个参数的秒数使得在这些秒数之后如果还没有收到这样的字符串时使等待超时。散布在登录过程中的 If 命令检查执行命令时是否有错。

在登录后最后执行的命令是 default --它使得 SLIP 连接成为所有主机的缺省路由，和 mode –它在线路上激活 SLIP 模式并且为你配置接口和路由选择表。

7.3.2 dip 参考

尽管 dip 被广泛地使用着，但它至今没有很好的文档。因此在这一节，我们将给出大部分 dip 命令的参考。你可以使用测试模式调用 *dip* 来得到所有命令的一个概观，并且进入帮助命令。要找出一个命令的句法，你可以不加任何参数地键入它；当然这对不用参数的命令是没有效的。

```
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>
```

下面，所有显示有 DIP>提示符的例子示出了如何在测试模式下输入一个命令，以及该命令所产生的输出。没有这个提示符的示例将作为脚本引用。

modem 命令

dip 提供了许多配置你的串行线路和 modem 的命令。有些是显而易见的，如 *port*—它选择一个串行口，以及 *speed*、*databits*、*stopbits* 和 *parity*，这些是用来设置通常的线路参数的。

modem 命令用来选择一个 modem 类型。目前，所支持的唯一类型是 *HAYES*（需大写）。你必须给 *dip* 提供一个 modem 类型，否则的话，它将拒绝执行 *dial* 和 *reset* 命令。*reset* 命令给 modem 发送一复位字符串；所用的字符串依赖于选择的 modem 类型。对于 Hayes 兼容 modem，它是 *ATZ*。

flush 命令用于刷新 modem 迄今为止所发出的所有响应。否则的话跟在 *reset* 后面的会话脚本可能会混淆，因为它会读取早些命令的 *OK* 响应。

init 命令用于在拨号之前向 modem 选择发送一条初始化串。Hayes modem 的缺省值是*“ATE0 Q0 V1 X1”*，它启动命令的回应和长结果代码，并且选择盲拨号（对拨号音不作检查）。

dial 命令最终向 modem 发出初始化串并且向远程系统拨号。Hayes modem 的缺省 *dial* 命令是 *ATD*。

echo 和 *term*

echo 命令是用作调试目的的，当 *echo on* 时，就会使得 *dip* 在控制台上回显发送给串行设备的所有命令信息。通过调用 *echo off* 可以再次关闭回显。

dip 同样允许你暂时离开脚本模式并进入终端模式。在这种模式下，你可以象任何其它常用的终端程序那样使用 dip，写至串行线路并从串行线路读取信息。要离开这种模式，键入 Ctrl-]。

get 命令

get 命令是 *dip* 设置变量的方法。最简单的形式是将变量设置成一个常数，正如用于上面例子中的一样。然而，你也可以通过指定关键字 ask 代替一个值来提示用户输入：

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

第三种方式是试着从远程主机取得这个值，这在某些情况下是非常有用的：有些 SLIP 服务器不允许你在 SLIP 连接上使用自己的 IP 地址，而是在你拨入时从一个地址池中取得一个分配给你，显示出一些信息通知你有关给你分配的地址。如果该信息看上去象这样“Your address: 193.174.7.202”，那么下面一段 *dip* 代码将让你获取地址：

```
… login ….
wait address: 10
get $locip remote
```

print 命令

这个命令是向 *dip* 启动的控制台上回显文字。任何 *dip* 变量都可以用于 *print* 命令中，如

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

变量名

dip 只认识一组预定义的变量。变量名总是以美元符号开头并且必须用小写字符。

$local 和*$locip* 变量含有本地主机的名字和 IP 地址。设置主机名使得 *dip* 将规范主机名存入*$local*，同时将相应的 IP 地址存入*$locip*。设置*$locip* 也类似于设置*$local*。

$remote 和*$rmtip* 变量对远程主机名和地址做同样的事情。*$mtu* 含有连接的 MTU 值。

这五个变量是仅有的能够使用 *get* 命令直接进行赋值的。主机的其它变量只能通过相应的命令来设置，但可以用于 *print* 语句；这些是*$modem*、*$port* 和*$speed*。

通过*$errlvl* 变量你可以获得上一条命令执行的结果。一个 0 错误号表示成功，而非零值表示有错。

if 和 goto 命令

if 命令相对于平常所称的 if 来说更是一个条件分支。它的语法是

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

这里表达式必须是与变量*$errlvl*、*$locip*、和*$rmtip* 之一的一个简单的比较关系。第二个操作数必须是一个整数；运算符 *op* 可以是==、!=、<、>、<=、以及>=之一。

goto 命令使得脚本跳转至 label 标号后继续执行。一个标号必须是一行中头一个记号，并且必须紧跟一个分号。

send、*wait* 和 *sleep*

这些命令在 *dip* 中帮助实现简单的会话脚本。 *send* 将它的参数输出到串行线路上。它不支持变量，但是支持所有 C-风格的反斜杠字符序列如*\n* 和*\b*。发音字符（~）用作回车/换行的缩写。

wait 将一个单词作为参数，扫描串行线路上的所有输入直到识别出这个单词。这个单词中不能含有任何空格。作为选项，你可以给 *wait* 一个超时值作为第二个参数；如果在该期间内没有收到期望的单词，命令将返回并置*$errlvl* 值为 1。

sleep 语句可以用来等待一段时间，例如耐心地等待登录序列的完成，再次，间隔时间是以秒计的。

mode 和 *default*

这些命令用于将串行线路转换至 SLIP 模式和配置接口。

mode 命令是 *dip* 在进入后台执行前最后执行的命令。除非出错，否则这个命令没有返回。

mode 将一个协议作为参数。目前 *dip* 只能识别 SLIP 和 CSLIP 作为有效的名字。然而当前版本的 *dip* 不能识别适应性 SLIP。

当在串行线路上激活 SLIP 以后，*dip* 执行 *ifconfig* 将接口配置成点对点连接，并且调用 *route* 将路由设置至远程主机。

另外，如果脚本在 *mode* 之前先执行 *default*，*dip* 也会使得缺省路由指向 SLIP 连接。

7.4 运行于服务器模式

设置你的 SLIP 客户是最艰难的部分。相反地，也即配置你的主机作为 SLIP 服务器，就很容易了。

一种方法是以服务器模式使用 *dip*，可以通过以 *diplogin* 方式调用它来完成。它的主要配置文件是*/etc/diphosts*，它将登录名与分配给这台主机的地址相关联。另一种方法是，你也可以使用 *sliplogin*，这是一个源自于 BSD 的工具，具有更灵活的配置方案，能在主机连接和脱开时让你执行 shell 脚本。目前这是个 Beta 版本。

这两个程序都需要你为每个 SLIP 客户设置一登录帐号。例如，假设你为在 **dent.beta.com** 的 Arthur Dent 提供 SLIP 服务，你可以通过在你的 passwd 文件中增加下面一行来建立一个名为 **dent** 的帐号：

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

此后，你要使用 *passwd* 工具设置 **dent** 的口令。

现在，当 **dent** 登录时，*dip* 将作为一个服务器启动。为了查出他确实允许使用 SLIP，*dip* 将查找*/etc/diphosts* 中的用户名。这个文件详细列出了每个 SLIP 用户的访问权限和连接参数。**dent** 的样本条目就象这样：

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

冒号分隔的第一个字段是用户登录用的名字。第二个字段可以含有一个另加的口令（见下面）。第三个字段是拨号的主机名或 IP 地址。下一个字段是没有任何特殊意义的信息字段（至今）。最后一个字段描述了连接参数。这是一个用逗号分隔的列表，指定协议（目前只有 *SLIP* 或 *CSLIP*），后跟 MTU。

当 **dent** 登录时，*diplogin* 从 *diphosts* 文件中抽取出他的有关信息，并且，如果第二个字段不空的话，提示输入一个“外部安全口令”。用户输入的字符串将与 *diphosts* 中的相应口令（未加密的）相比较。如果它们不匹配的话，这个登录企图将被拒绝。

否则的话，*diplogin* 通过将串行线路转换成 CSLIP 或 SLIP、并且设置接口和路由来处理。这个连接将保持建立状态直到用户断开并且 modem 撤消连线为止。此时，*diplogin* 将线路返回到普通线路规程并且退出。

diplogin 需要超级用户特权。如果你没有让 *dip* 运行 setuid 为 **root**，那么你应该使得 *diplogin* 是 *dip* 的一个独立的拷贝，而不是一个简单的连接。然后，*diplogin* 能够安全地被 setuid，而不会影响 *dip* 本身的状态。

注释

[1] 在 RFC1441 中描述了 Van Jacobson 头压缩。

[2] *dip* 意思是 Dialup IP。它是由 Fred van Kempen 编制的。

[3] *diplogin* 也能够（并且必须）运行 setuid。见本章最后一节。

[4] 对于使用三字符的首字母缩写词的更多的回文乐趣，参见新闻组 alt.tla。

第八章 点对点协议

8.1 揭开 P 字母

正像 SLIP 一样，PPP 是在串行连接上发送数据报的协议，但却改进了前者的几个不足之处。它让通信的双方在开始时协商诸如 IP 地址和最大数据报大小等选项，并且为客户提供授权（权限）。对于每个这样的功能，PPP 都有一个独立的协议。下面，我们将概要地讨论 PPP 的这些基本创建框图。这里的讨论远不是完整的；如果你想对 PPP 知道的更多些，极力推荐你阅读 RFC-1548 中它的规格说明，以及许多相关的 RFCs。[1]

PPP 的最底层是*高级数据链路控制协议*（*High-Level Data Link Control Protocol*），缩写为 HDLC，[2] 它定义了单个 PPP 帧的分界，并提供了 16 比特的检查和。相对于非常原始的 SLIP 包装来说，一个 PPP 帧能够容纳除 IP 以外的其它协议，如 Novell 的 IPX、或 Appletalk。PPP 通过在基本的 HDLC 帧上加上一个协议字段来做到这功能，这个字段用于识别帧所携带的包的类型。

LCP，链路控制协议（the Link Control Protocol），用于 HDLC 的上层，用于协商适合于数据链路的选项，如指出链路的一边同意接收的最大数据报大小的最大接收单元（MRU）。

在 PPP 连接的配置阶段重要的一步是客户授权（权限）。尽管不是强制的，但对于拨号线路几乎是必要的。通常，被呼叫的主机（服务器）通过验证客户是否知道某个秘密的键值来要求客户认证自己。如果呼叫者不能给出正确的秘密键值，连接就中断了。使用 PPP，授权工作是双方面的；也即，呼叫者也可以要求服务器认证自己。这些认证过程对于双方是完全独立的。对于不同的认证类型有两个协议，我们将在下面更进一步地讨论。它们被命名为口令认证协议（Password Authentication protocol），或 PAP 和质询握手认证（Challenge Handshake Authentication Protocol）,或 CHAP。

路由通过数据链路的每一个网络协议，如 IP、Appletalk 等等，使用相应的网络控制协议（Network Control Protocol）（NCP）被动态地配置。例如，要通过链路发送 IP 数据报，PPP 的双方首先必须协商双方使用的 IP 地址。用于此目的的控制协议是 IPCP，即互连网协议控制协议（Internet Protocol Control Protocol）。

除了通过链路发送标准的 IP 数据报，PPP 也支持 IP 数据报的 Van Jacobson 头压缩。这是一项将 TCP 包头缩小到仅有三个字节的技术。它也用于 CSLIP，并且常常通俗地称为 VJ 头压缩。是否使用压缩同样可以在开始时通过 IPCP 协商来决定。

8.2 Linux 上的 PPP

在 Linux 中，PPP 的功能被分成了两个部分，一个是位于内核的低级的 HDLC 驱动程序部分，另一部分是处理各种控制协议的用户空间的 *pppd* 后台程序。Linux 的 PPP 当前发行版是 *linux-ppp-1.0.0*，包含有内核 PPP 模块、*pppd*、和一个用于拨号至远程系统的称为 *chat* 的程序。

PPP 内核驱动程序是由 Michael Callahan 编制的。*pppd* 源自于为 Sun 和 386BSD 机器的一个免费 PPP 实现，它是由 Drew Perkins 和其他人编制的，并且由 Paul Mackerras 维护。是由 Al Longyear[3] 移植到 Linux 上的。*chat* 是由 Karl Fox 编制的。[4]

正如 SLIP，PPP 是通过一个特殊的线路规程来实现的。要以 PPP 连接使用某个串行线路，你首先要象往常那样通过 modem 建立一个连接，随后将线路转换成 PPP 模式。在这种模式下，所有的进入的数据都传给了 PPP 驱动程序，该驱动程序检查传入的 HDLC 帧的有效性（每个 HDLC 帧带有一个 16 比特的检验和），并且解开并分发它们。目前，它能够处理 IP 数据报，可选地使用 Van Jacobson 头压缩。为了支持 IPX，PPP 驱动程序也将被扩展成能处理 IPX 包。

内核的驱动程序是由 *pppd*，PPP 后台程序，协助工作的，在实际的网络通信能在链路上进行之前，它执行必要的整个初始化和认证过程。*pppd* 的行为可以使用一些选项来调整。由于 PPP 非常复杂，不可能在一章中解释所有的东西。因此本书不打算涵盖 *pppd* 的所有方面，而只是给你一个介绍。详细信息，请参考在线手册页和 *pppd* 原始发行版中的 *README*s，它将帮助解决在这章中没有讨论过的大多数问题。如果在阅读了所有的文档之后你的问题还没有得到解决，你应该到新闻组 comp.protocols.ppp 寻求帮助，在那里你可以接触到包括 *pppd* 开发者的大多数人。

8.3 运行 *pppd*

当你想通过一个 PPP 连接连到 Internet 上，你必须设置好基本的网络功能，如回送设备和解析器。这两者已在前面章节中讨论过了。还有些有关在串行链路上使用 DNS 的解释；请参考 SLIP 一章中对此的描述。

作为一个使用 pppd 如何建立一个 PPP 连接的入门例子，假设你再次在 **vlager**。你已经拨号到 PPP 服务器，**c3po**，并且登录进 ppp 帐号。**c3po** 已经启动它的 PPP 驱动程序。在退出用于拨号的通信程序以后，你执行下面的命令：

```
# pppd /dev/cua3 38400 crtscts defaultroute
```

这会将串行线路 *cua3* 转换到 PPP 模式并建立一个到 **c3po** 的 IP 连接。用于串行端口的传输速度将是 38400bps。*crtscts* 选项打开端口的硬件握手功能，这对于高于 9600bps 的速度是绝对必要的。

在启动之后 *pppd* 所做的首件事情是使用 LCP 与远端协商几种连接特性，通常，*pppd* 所试用的缺省的选项集将能工作，所以我们不打算在这里考虑这些。我们将在后面几节中再回过来详细讨论 LCP。

此时，我们也假设 **c3po** 不需要从我们这边取得任何认证，所以配置阶段成功地完成了。

然后 *pppd* 将使用 IPCP，IP 控制协议，与它的对等点协商 IP 参数。由于上面我们没有对 *pppd* 指定任何特殊的 IP 地址。它将试着使用通过解析器查找本地主机名获得的地址。此后，两者将向对方宣告他们的地址。

通常，这些缺省设置并没有什么错误。即使你的机器是在一个以太网上，你可以对以太网和 PPP 接口使用同一个 IP 地址。当然，*pppd* 允许你使用不同的地址，或者请求对方使用某个特定的地址。这些选项将在以后章节中描述。

在通过了 IPCP 设置阶段以后，*pppd* 将准备你的主机的网络层来使用 PPP 连接。它首先配置 PPP 网络接口作为一个点对点连接，对第一个活动的 PPP 连接使用 *ppp0*，对第二个使用 *ppp1*，依次类推。下一步，它将设置一个指向链路另一端主机的路由表条目。在上面示出的例子中，*pppd* 将使得缺省网络路由指向 **c3po**，因为我们已给它 *defaultroute* 选项。[5] 这使得所有到不在本地网络上主机的数据报都被发送到 **c3po**。*pppd* 还支持几个不同的路由选择方案，这将在本章后面详细讨论。

8.4 使用选项文件

在 *pppd* 分析它的命令行参数之前，为了查找缺省选项它扫描几个文件。这些文件可能含有有效的命令行参数，它们分布在任意的行上。注释语句是由“#”开头的。

第一个选项文件是*/etc/ppp/options*，当 *pppd* 启动时总会扫描它。使用它设置一些全局缺省值是个好主意，因为它允许你阻止你的用户做某些危及安全的事情。例如，要使得 pppd 要求对方某种授权认证（*PAP* 或 *CHAP*），你应该在该文件中加入 *auth* 选项。用户覆盖不了这个选项，所以与不在我们的授权数据库中的任何系统建立一个 PPP 连接变成不可能的事。

在*/etc/ppp/options* 文件以后读取的其它选项文件，是用户主目录中的*.ppprc*。它允许每个用户指定她自己的缺省选项集。

一个样本*/etc/ppp/options* 文件可以是象这样的：

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

这些选项的头两个用于权限认证并且将在下面给出解释。*lock* 关键字使得 *pppd* 遵守标准的 UUCP 设备锁定方法。根据这个惯例，每个访问串行设备的进程，如*/dev/cua3*，在 UUCP spool 目录中创建一个名为 *LCK..cua3* 的锁定文件，用于指示该设备正在使用。这是避免任何其它程序如 *minicom* 或 *uucico* 去打开 PPP 正在使用的串行设备。

在全局配置文件中提供这些选项的原因是如上显示的那些选项是不能被覆盖掉的，因而提供了一个合理的安全级。然而，请注意有些选项可以在后面被覆盖掉；这样的一个例子是 *connect* 串。

8.5 使用 *chat* 拨出

在上例中让你感到不便的事情之一是在你能够启动 *pppd* 之前必须手工地建立连接。不象 dip，*pppd* 对于拨号至远程系统以及登录没有自己的脚本语言，而是需要依赖于某些外部程序或 shell 脚本来做这些事。要执行的命令可以用 *connect* 命令行选项给予 *pppd*。*pppd* 将重定向该命令的标准输入和输出到串行线路上。针对此一个有用的程序是 *expect*，是由 Don Libes 编写的。它有一个基于 Tcl 的非常强大的语言，并且是针对此类应用明确地设计出来的。

pppd 软件包带有一个同样称为 chat 的程序，是用于指定 UUCP-风格的会话脚本的。基本上说，一个会话脚本是由我们期望从远程系统收到的交互式的字符串序列以及我们所发送的应答字符串序列。我们将分别称之为期望字符串和发送字符串。这是从一个典型会话脚本中的摘录：

```
ogin: b1ff ssword: s3kr3t
```

这告诉 chat 等待远程系统发送来登录提示，并且返回登录名 b1ff。我们只是等待 ogin: 所以登录提示是大写还是小写没什么关系，也不用管它是否完全正确。接下来的又是一个期望字符串它使得 chat 等待口令输入提示，并且随后发出你的口令。

这基本上，上面是 chat 脚本所要做的。当然，拨号到一个 PPP 服务器的完整脚本也必须包括适

当的 modem 命令。假设你的 modem 使用 Hayes 命令集，并且服务器的电话号码是 318714。那么与 c3po 创建一个连接的完整的会话请求是

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

根据定义，头一个字符串必须是期望字符串，但是在我们启动 modem 之前，modem 是不会发出任何东西的，所以我们通过指定一个空串来跳过第一个期望字符串。然后我们继续发出 ATZ，即 Hayes 兼容 modem 的复位命令，并等待它的响应（OK）。下一个串向 chat 发送出一个拨号命令和电话号码，并且期望得到 CONNECT 消息的响应。接下来又是一个空串，因为我们现在还不想发送出任何信息，而是等待登录提示的出现。余下的 chat 脚本所做的工作完全象我们上面描述的一样。

-v 选项使得 chat 将所有的活动记录在 syslog 后台程序的 local2 中。[6]

在命令行上写出会话脚本会承担一定的风险，因为用户可以使用 ps 命令观察进程的命令行。你可以将会话脚本放入一个文件，比如说是 dial-c3po，来避免这个风险。通过给 chat 一个-f 选项后跟这个文件名，你可以使得 chat 从该文件中读取脚本取而代之从命令行读取。现在完整的 pppd 命令就象这样：

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

除了指定拨号脚本的 connect 选项，我们在命令行上多加了两个选项：-detach—告知 pppd 不要从控制台分离并且变成为后台进程。Modem 关键字使得 pppd 在串行设备上执行某些 modem 专用的动作，就如在拨号之前和之后挂断线路。如果你不使用这个关键字，pppd 将不会监视端口的 DCD 线，因而将不会检测是否远端出呼意料地挂断了。

上面给出的例子是非常简单的；chat 允许有更为复杂的会话脚本。一个非常有用的特性是指定中止含有错误的的会话。典型的中止字符串是象消息 BUSY、或 NO CARRIER，这是当所拨的号码是忙音、或没有提起电话时，你的 modem 产生的。为了使得 chat 迅速地识别出这些中止字符串，而不是等到超时，你可以使用 ABORT 关键字在脚本的开头指定它们：

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

以同样的方式，你可以通过插入 TIMEOUT 选项来改变会话部分中的超时值。详细资料，请参见 chat(8)的手册页。

有时，你也想有条件地执行会话脚本的一部分。例如，当你没有接收到远端的登录提示时，你可能想发送一个 BREAK，或一个回车键。你可以通过给期望字符串附加一个子脚本来做到。它是由一系列的发送和期望字符串组成，正象整个脚本本身，它是用连字号分开的。每当附加的期望字符串没有及时收到时，该子脚本就会被执行。在上面的例子中，我们要象下面那样修改会话脚本：

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

现在，当 chat 没有收到远程系统发送的登录提示时，子脚本就被执行，首先发送一个 BREAK，然后再次等待登录提示的出现。如果现在提示出现时，脚本就会象往常那样继续，否则的话，它将带错而终止。

8.6 调试你的 PPP 设置

缺省地，*pppd* 将把任何警告和出错消息记录到 *syslog* 的 *daemon* 设施中。你必须往 *syslog.conf* 中增加一个条目，以使得这些消息重定向到一个文件中，或者甚至到控制台上，否则的话 *syslog* 就会轻易地丢弃这些消息。下面这个条目将所有的消息送到*/var/log/ppp-log*：

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

如果你的 **PPP** 设置不能立刻工作，查看这个日志文件会给你什么地方出错的一点提示。如果这没有什么帮助，你可以使用 **debug** 选项打开额外的调试输出信息。这使得 *pppd* 将所有发送和接收的控制数据包的内容记录到 *syslog* 中。所有的消息将传入 *daemon* 设施中。

最后，最猛烈的方法是通过使用选项 *kdebug* 调用 *pppd*，以激活内核层的调试。在该选项之后是下面数值的比特位或运算的数值：1 表示一般调试消息，2 表示打印所有传入的 HDLC 帧的内容，4 是使得驱动程序打印出所有传出的 HDLC 帧。要捕获内核的调试消息，你必须或者运行读取 */proc/kmsg* 文件的 *syslogd* 后台程序，或者是 *klogd* 后台程序。这两者都将内核的调试信息定向到 *syslog* 的 *kernel* 设施中。

8.7 IP 配置选项

在连接配置期间，IPCP 是用于协调几个 IP 参数的。通常，每个端点都会发出一个 IPCP 配置请求包，指出它想要改变哪些缺省值，改成什么值。在接收方，远端依次检查每个选项，并且或者同意改变或者拒绝。

关于 *pppd* 将尝试协调哪些 IPCP 选项，*pppd* 给了你许多的控制权。你可以通过命令行选项来调节这些选项，我们将在下面讨论它们。

8.7.1 选择 IP 地址

在上面的例子中，我们用 *pppd* 拨号到 **c3po** 并且建立了一个 IP 连接。在连接的两端没有预备要选择一个特定的 IP 地址。而是使用 **vlager** 的地址作为本地 IP 地址，并让 **c3po** 自己给出自己的。然而，有时候在连接的一端或两端控制要使用的地址是很有用的。*pppd* 支持这种控制的几种变异。

要请求特定的地址，通常你要给 *pppd* 提供下面的选项：

```
local_addr:remote_addr
```

这里 *local_addr* 和 *remote_addr* 可以用点分四组表示法表示，或用主机名表示。[7] 这使得 *pppd* 试图使用第一个地址作为它自己的 IP 地址，而第二个作为对等点的。如果在 IPCP 协商时，对方拒绝了其中之一的地址，那么就不能建立起 IP 连接了。[8]

如果你只想设置本地地址，并接受任何对等点使用的地址，你只需不使用 *remote_addr* 部分就行了。例如，要使得 **vlager** 使用 IP 地址 130.83.4.27 而不使用自己的 IP 地址，你需要在命令行上给出 130.83.4.27:即可。类似地，如果只想设置远端点的地址，你只需让 *local_addr* 字段为空。此时，

缺省地，*pppd* 将使用与你的主机相应的地址。

有些处理许多客户站点的 PPP 服务器动态地为客户分配地址：只有当拨号进来时才给客户系统分配地址，而当客户退出时就再次清除。当拨号到这种服务器时，你必须确信 *pppd* 没有要求从服务器要求任何特定的 IP 地址，而是接受服务器让你使用的地址。这意味着你不必指定 *local_addr* 参数。另外，你必须使用 *noipdefault* 选项，该选项使得 *pppd* 等待对等点提供 IP 地址，以替代使用本地主机的地址。

8.7.2 通过 PPP 连接进行路由

在设置好网络接口以后，*pppd* 通常将主机的路由只设置到它所连接的远端点上。如果远端主机是在一个 LAN 上，你当然也希望能够连接到远端主机所在网络的其它主机上；也即，必须设置一个网络路由。

上面，我们已经知道可以使用 ***defaultroute*** 选项请求 *pppd* 设置默认的路由。如果你所拨号连接的服务器将作为你的 Internet 网关话，那么这个选项将是非常有用的。

相反的情况是，你的系统作为孤独的主机的一个网关，也同样很容易做到。例如，假设虚拟酿酒厂的某个雇员，他家的机器叫做 **loner**。当通过 PPP 连接到 **vlager** 时，他使用酿酒厂子网的地址。在 **vlager** 端，我们现在可以给 *pppd* 一个 ***proxyarp*** 选项，它将为 **loner** 安装一个代理 ARP 条目。这将自动地使得 **loner** 可以访问酿酒厂和葡萄酒厂的所有主机了。

然而事情并不是总是那样的简单，例如，当连接两个局域网时。这常常需要增加一个特殊的网络路由，因为这些网络都可以有它们自己默认的路由。另外，使得两个端点使用 PPP 连接作为默认路由将产生一个循环，到未知目的地的包将在这两个端点之间来回传送，直到它们过了存活期。

作为一个例子，假设虚拟酿酒厂在某个其它城市开了一个分支机构。这个附属机构使用 IP 网络号 191.72.3.0 运行一个他们自己的以太网，它是酿酒厂的 B 类网络的子网 3。他们想通过 PPP 连接到酿酒厂的主以太网上以更新客户的数据库等等。再次，**vlager** 将作为一个网关；它的远端点被称作 **sub-etha** 且 IP 地址为 192.72.3.1。

当 **sub-etha** 连接到 **vlager** 时，它将象平常一样使得默认路由指向 **vlager**。然而，在 **vlager** 上，我们必须为通过 **sub-etha** 的子网 3 安装一个网络路由。为此，我们使用一个 *pppd* 的至今还没有讨论过的特性—*ip-up* 命令。这是一个位于*/etc/ppp* 中的 shell 脚本或程序，是在 PPP 接口配置完以后执行的。当使用时，它是用下列参数调用的：

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

这里，*iface* 指定所使用的接口，*device* 是所使用的串行设备文件（如果使用了 stdin/stdout，那么是*/dev/tty*）的路径名，*speed* 是设备的速度。*local_addr* 和 *remote_addr* 给出了以点分四组表示的用于连接两端点的 IP 地址。在我们这种情况下，*ip-up* 脚本可以包含下列代码段：

```
#!/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
```

以类似的方式，*/etc/ppp/ip-down* 是用于在 PPP 连接再次断开以后取消所有 *ip-up* 的活动。

然而，路由选择方案还没有完成。我们已经在双方的 PPP 主机上设置好了路由选择表条目，但至今，在两个网上的所有其它主机关于 PPP 连接的任何事情。如果在附属机构的所有主机有指向 **sub-etha** 的默认路由，并且酿酒厂的所有主机都默认地路由到 **vlager** 的话，
这就不是个大问题了。如果不是这样的话，那么你唯一的选择是使用一个象 *gated* 的路由选择后台程序。在 **vlager** 上创建了网络路由以后，路由选择后台程序将会对所有子网上的主机广播这个新路由。

8.8 链路控制选项

上面，我们已经遇到过 LCP，也即链路控制协议，它是用于协调链路特性以及测试链路的。

通过 LCP 协调的两个重要的选项是最大接收单元，和异步控制字符映射表。还有许多其它的 LCP 配置选项，但它们太特殊，这里就不给予描述了。请参考 RFC1548 中对它们的解释。

异步控制字符映射表，俗称异步表，是用于象电话线路的异步链路上确定必须被换码的控制字符（用一特殊的二字符序列替换）。例如，你可能想要避开用于软件握手信号的 XON 和 XOFF 字符，因为某些配置不当的 modem 在接收到一个 XOFF 时可能会卡住。其它的控制字符包括 Ctrl-]（telent 的换码字符）。通过在异步表中指定这些控制字符，PPP 允许你对 ASCII 码 0 到 31 中的任何字符进行换码。

异步表是一个 32 比特宽的位图，最低比特位对应于 ASCII 的 NUL 字符，最高比特位对应于 ASCII 码 31。如果一个比特位置位，它表示相应的字符在发送到链路之前必须被转义（换码）。最初，异步表被设置为 0xffffffff，它表示所有的控制字符都将被转义。

为了告诉你的对等点不需要对所有的控制字符换码，而只需要对其中的几个进行转义，你可以使用 asyncmap 选项给 pppd 指定一个新的异步映射。例如，如果只有^S 和^Q（ASCII 17 和 ASCII 19，通常用作 XON 和 XOFF）必须被转义，那么就使用下面的选项：

```
asyncmap 0x000A0000
```

最大接收单元（Maximum Receive Unit），或 MRU，通知对等点我们想接收的 HDLC 帧的最大长度。尽管这会使你想起 MTU 的值（最大传输单元 Maximum Transfer Unit），这两者很少有共同之处。MTU 是内核网络设备的一个参数，描述了接口能够处理的最大帧的大小。而 MRU 只是对远端点不要产生任何大于 MRU 值的一个建议；不管怎样，接口必须能够接收最大为 1500 字节的帧。

因此，如何选择一个 MRU 的大小就与链路的传输能力关系不大了，而是与如何能达到最大的吞吐量有关。如果你打算在链路上运行交互式的应用程序，那么将 MRU 值设置为小到 296 是个不错的主意，这样一个偶然的大数据包（比如，从一个 FTP 来的会话）就不会使得你的光标“跳动”了。要告知 pppd 请求一个 296 的 MRU，你必须给它一个选项 mru 296。然而，小的 MRUs 只有在你没有禁止 VJ 头压缩（它默认是激活的）时才有意义。

pppd 也能够理解一些设置协调过程整体行为的 LCP 选项，比如在链路终止前可以交换的最大请求配置数。除非你明确的知道你在做什么，否则不要改动它们。

最后，还有两个用于 LCP 回显消息的选项。PPP 定义了两类消息，回显请求（Echo Reguest）和回显响应（Echo Response）。Pppd 使用这个特性来检查一个链路是否还在运行。你可以通过使用 lcp-echo-interval 选项和一个以秒计的时间值来激活这个特性。如果在这个时间间隔内没有从远程主机收到帧，那么 pppd 生成一个 Echo Reguest，并切期待对等点返回一个 Echo Response。如果对等点没有产生一个响应，那么在发送了一定数量的请求以后链路即被中止。这个数量可以用

lcp-echo-failure 选项来设置。缺省地，这个特性是全部禁止的。

8.9 常规安全考虑

一个配置不当的 PPP 后台程序能成为一个破坏性的安全缺口。它糟糕到象允许任何人能够将他的机器接入你的以太网上（这是非常糟糕的）。在本节中，我们将讨论一些能使你安全配置 PPP 的方法。

Pppd 的一个问题是，要配置网络设备和路由选择表需要 root 权限。通常解决这个问题的办法是运行它 setuid root。然而，pppd 允许用户设置各种与安全相关的选项。为了免受用户可能通过操作这些选项而发起的攻击，建议你在全局/etc/ppp/options 文件中设置一些默认值，就象**使用选项文件**一节例子中显示的。其中有些，比如授权认证选项用户是覆盖不了的，所以对操作起到了一定的保护作用。

当然，你也需要针对你用 PPP 连接的系统来保护自己。要挡开假冒他人的主机，你应该总是检查你的对等点的授权认证。另外，你不应该允许外部主机能使用它们选择的任何 IP 地址，而是限制他们只能使用几个地址。下面一节将涉及到这些问题。

8.10 PPP 授权认证

8.10.1 CHAP 与 PAP

对于 PPP 来讲，每个系统可以要求它的对等点使用两种授权认证协议之一来认证自己。这两个协议是口令认证协议（Password Authentication Protocol）（PAP）和质询握手认证协议（Challenge Handshake Authentication Protocol）（CHAP）。当创建了一个连接以后，两端都可以请求对方认证自己，而不管它是呼叫者还是被呼叫者。下面当我想区别认证系统和认证者时，我将宽松地用“客户”和“服务器”来讲。一个 PPP 后台程序能够请求它的对等点通过发送另一个确定期望的认证协议的 LCP 配置请求来进行认证。

PAP 与通常的登录过程的工作原理基本上一样。客户通过向服务器发送一个用户名和一个（可以是加密的）口令来认证自己，服务器用它们与自己的秘密数据库相比较。对于那些通过监听串行线路以获得口令的偷听者和对于重复试验出错（trial and error）的攻击来说，这项技术是易受攻击的。

CHAP 就没有这些不足之处。对于 CHAP，认证者（也即是服务器）向客户发送一个随机生成的“质询”字符串和自己的主机名。客户使用该主机名来查询适当的秘码（口令），并将其与质询合在一起形成一个字符串，并且使用一个单向（杂凑）哈希函数加密该串。结果值与自己的主机名一起返回给服务器。现在服务器执行相同的计算，如果得到相同的结果就认可该客户。

CHAP 的另一个特性是它不仅在开始时请求客户认证自己，而且每隔一定的时间就发送一个质询，以确信客户并没有被入侵者替换掉，例如通过切换电话线路。

pppd 将 CHAP 和 PAP 的秘密键值分别存放在两个不同的文件中，称为*/etc/ppp/chap-secrets* 和 *pap-secrets*，通过使用一个或另一个文件来进入远程主机，你可以很好地控制是使用 CHAP 还是 PAP 来与对方认证我们自己，反之也然。

默认地，*pppd* 无须来自远端的认证，但是同意来自远端的认证请求。由于 CHAP 比 PAP 来得

更为稳固和强壮，*pppd* 会尽可能地使用前者。如果对方不支持它，或者 *pppd* 在它的 *chap-secrets* 文件中找不到该远程系统的一个 CHAP 秘密（密码），它就回过来使用 PAP。如果它也没有针对于对等点的 PAP 密码，它就拒绝做任何认证。结果，连接被关闭。

这种行为可以从几个方面进行修改。例如，当给出 **auth** 键值时，*pppd* 将请求对方来认证自己。如果 *pppd* 分别在 CHAP 或 PAP 数据库中有一个对于该对等点的密码时, *pppd* 将同意使用 CHAP 或 PAP 来做。还有一个可以打开或关闭特定认证协议的选项，但这里我们就不再描述它们了。详细信息请参考 *pppd*(8)手册页。

如果所有和你进行 PPP 对话的系统同意为你认证它们自己，你应该将 **auth** 选项放入全局 */etc/ppp/options* 文件中并且为每个系统在 *chap-secrets* 文件中定义口令。如果一个系统不支持 CHAP，那么就在 *pap-secrets* 文件中为其加入一个条目。这样，你就可以确信连接到你的主机上的任何系统都是认证过的。

下面两节描述了这两个 PPP 密码文件，*pap-secrets* 和 *chap-secrets*。它们位于*/etc/ppp* 中，包含客户、服务器和口令三为一组的值，还可以后接任选的一个 IP 地址列表。客户和服务器字段的解释对于 CHAP 和 PAP 是不同的，这也起决于我们是否要为对方认证我们自己，或者要求服务器为我们认证它们自己。

8.10.2 CHAP 的秘密文件

当 *pppd* 必须为某些使用 CHAP 的服务器认证自己时，*pppd* 就在 *pap-secrets* 文件中寻找客户字段与本地主机名相同、服务器字段与在 CHAP 质询中发送来的远程主机名相同的条目。当要求对等点认证它自己时，该规则只是简单地反一下：此时 *pppd* 寻找客户字段与远程主机名相同（经由客户的 CHAP 响应发送过来的）、服务器字段与本地主机名相同的条目。

下面是 **vlager** 的一个样本 *chap-secrets* 文件：[9]

```
# 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.
```

当与 **c3po** 建立了一个 PPP 连接时，**c3po** 要求 **vlager** 通过发送一个 CHAP 质询用 CHAP 来认证自己。然后 *pppd* 扫描 *chap-secrets* 文件中的客户字段等于 **vlager.vbrew.com**、服务器字段等于 **c3po.lucas.com** 的条目，[10] 这样就找到了上面的第一行。然后它就从质询字符串和密码（**Use The Source Luke**）生成一个 CHAP 响应，并发送给 **c3po**。

与此同时，*pppd* 为 **c3po** 组合成一个 CHAP 质询，其中包含一个唯一的质询字符串和它的全资主机名 **vlager.vbrew.com**。**c3po** 以刚才我们讨论的方式构成一个 CHAP 响应，并将其返回给 **vlager**。现在，*pppd* 从响应中取得客户的主机名（**c3po.vbrew.com**），并且查找 *chap-secrets* 文件中客户字段为 **c3po**、服务器字段为 **vlager** 的一行。这正是上面的第二行，所以 *pppd* 将 CHAP 质询和该行的密码（**riverrun, pasteve**）组合在一起并进行加密，并将结果与 **c3po** 的 CHAP 响应相比较。

可选的第四个字段列出了在第一个字段的客户的 IP 地址的一个列表。这个地址可以用点分四组

表示法给出也可以用解析器查找出的主机名给出。例如，如果 **c3po** 在 IPCP 协商中请求使用一个不在这个列表中的 IP 地址，请求将被拒绝，并且 IPCP 将被关闭。因此在上面所示的样本文件中，**c3po** 被限制于只能使用自己的 IP 地址。如果地址字段是空的，那么就允许使用任何地址；如果该字段不空，就限制了该客户使用的 IP 地址。

样本 *chap-secrets* 文件的第三行允许任何主机与 **vlager** 建立一个 PPP 链接，因为一个为*的客户或服务器字段与任何主机名匹配。唯一的限制是这些主机必须知道秘码，并且使用地址 **pub.vbrew.com**。以通配符作主机名的任何条目可以在秘密文件中的任何地方，因为 *pppd* 总是使用服务器/客户对最为匹配的条目。

关于 *pppd* 在秘密文件中查找获取主机名的方式，还有一些要作说明。就象前面所解释的，远程主机名总是通过对等点的 CHAP 质询或响应包提供的。本地主机名缺省的是通过调用 *gethostname(2)* 函数来得到的。如果你已经将系统名设置为你的非限制的主机名，这样的话，你就必须使用 **domain** 选项为 *pppd* 另外给出一个域名：

```
# pppd …domain vbrew.com
```

这将使得对于所有有关授权认证的活动，都将把酿酒厂的域名添加到 **vlager** 上。其它修改本地主机名的 progpppd 的想法是 **usehostname** 和 **name**。当你使用“*local*:varremote”（*local* 不是点分四组表示的名字）在命令行上给出本地 IP 地址，*pppd* 将使用它作为本地主机名。详细信息请参见 *pppd(8)* 的手册页。

8.10.3 PAP 秘密文件

PAP 秘密文件与 CHAP 所用的非常相似。头两个字段总是包含一个用户名和一个服务器名；第三个字段含有 PAP 密码。当远端发送来一个认证请求时，*pppd* 使用服务器字段等于本地主机名、用户字段等于请求中发送过来的用户名的条目。当为对等点认证自己时，*pppd* 从用户字段等于本地用户名、服务器字段等于远程主机名的行中取得密码（secret）。

一个样本 PAP 秘密文件就象这一样：

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

第一行用于和 **c3po** 对话时认证我们自己。第二行说明一个名为 **c3po** 的用户必须为我们认证它自己。

第一列的名字 **vlager-pap** 是我们发送给 **c3po** 的用户名。缺省地，*pppd* 将用本地主机名作为用户名，但是你也可以通过给出 **user** 选项后跟一个名字来指定一个不同的名字。

当为了与对等点进行认证而从 *pap-secrets* 文件中选取一个条目时，*pppd* 必须知道远端主机的名字。由于它没法找出该名字，你必须在命令行上使用 **remotename** 关键字后跟对等点的主机名来指定它。例如，要使用上面的条目与 **c3po** 认证，我们必须在 *pppd* 的命令行上加入下面的选项：

```
\#{} pppd … remotename c3po user vlager-pap
```

在第四个字段（以及所有后面的字段），你可以指定特定的主机所允许的 IP 地址，正如 CHAP 秘密文件中的一样。这样，对等点就可以从那个列表中只请求地址。在样本文件中，我们要求 **c3po** 使用它的真实 IP 地址。

注意，PAP 是一个非常弱的认证方法，所以建议尽可能使用 CHAP。因此我们在这里对 PAP 就不再详述了；如果你对 PAP 的使用感兴趣的话，你可以在 *pppd(8)*的手册页中找到许多有关 PAP 的特性。

8.11 配置一个 PPP 服务器

以服务器方式运行 *pppd* 只需在命令行上加上一些适当的选项。原则上讲，你要创建一个特殊的帐号，比如说是 ppp，并给它一个脚本或程序作为用这些选项调用 *pppd* 的登录 shell。例如，你要在 */etc/passwd* 中加入下面一行：

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

当然，你可以使用与上面所示不同的 uids 和 gids。你还必须使用 *passwd* 命令为上面的帐号设置口令。

然后，*ppplogin* 脚本可以象这样：

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

mesg 命令用于禁止其他用户往 tty 写信息，比如使用 *write* 命令。*stty* 命令关闭字符回显功能。这是必须的，否则的话对等点发送来的任何信息都将回显回去。上面给出的最重要的 *pppd* 选项是 `-detach`，因为它防止 *pppd* 脱离控制 tty。如果我们不指定这个选项，它就会进入后台，使得 shell 脚本退出。随之而来的是串行线路将被挂断，连接中止。*silent* 选项使得 *pppd* 在开始发送前进行等待直到它从呼叫系统接收到一个数据包。这可以防止当呼叫系统很缓慢地启动它的 PPP 客户时传输的超时。*modem* 选项用于使得 *pppd* 监视 DTR 引线用以观察对等点是否已掉线，而 `crtscts` 选项用以打开硬件握手信号。

除了这些选项以外，你还可以迫使使用某些授权认证，例如通过在 *pppd* 的命令行上或在全局选项文件中指定 **auth**。有关开启和禁用各种认证协议的更多选项的描述，可参见在线手册。

注释

[1] 相关的 RFCs 列在本书后面的指定参考书目中。

[2] 实际上，HDLC 是一个由国际标准化组织（ISO）设计的非常通用协议。

[3] 两位作者都已经说过今后会比较忙。如果你有任何有关 PPP 的一般问题，你最好询问 Linux 积极分子 mailing list 的 NET 通道上的人。

[4] karl@morningstar.com 。
[5] 缺省的网络路由只是在还没有时才被建立。
[6] 如果你编辑 syslog.conf 将这些登录信息重定向到一个文件中，请确信这个文件是不可读的，因为 chat 缺省地记录下整个会话脚本—包括口令和所有其它的信息。
[7] 对于 CHAP 权限认证方法在该选项中使用主机名表示是有其因果关系的。请参见下面的 CHAP 一节。
[8] 通过将选项 ipcp-accept-local 和 ipcp-accept-remote 给予 pppd，你可以允许远端点覆盖你对使用 IP 地址的设想。详细信息请参阅手册。
[9] 双引号不是口令的一部分，它们只是用来标明口令中的空格而已。
[10] 这个主机名是从 CHAP 质询中得到的。

第九章 各种网络应用程序

在成功地设置好 IP 和解析器以后，我们现在转过来讨论你想在网络上提供的服务。这一章讲解了一些简单网络应用程序的配置，包括 *inetd* 服务器、以及 *rlogin* 族中的程序。象网络文件系统（NFS）以及网络信息系统（NIS）所基于的远程过程调用接口也将概要地给予讨论。然而，NFS 和 NIS 的配置内容太多，将用独立的章节进行描述。电子邮件和网络新闻（netnews）也将用独立的章节给予讨论。

当然，我们不可能在本书中讨论所有的网络应用程序。如果你要安装一个这里没有讨论过的程序，比如说，*talk*、*gopher* 或 *Xmosaic*，请参考它们的手册以获得详细信息。

9.1 inetd 超级服务器（super-server）

通常，服务是由所谓的后台程序{或称端口监控程序}*daemons* 执行的。一个端口监控程序是一个程序，它打开一个端口，并且等待进入的连接。如果来了一个连接，它就创建一个子进程来接纳这个连接，而父进程继续监听更多的请求。这个概念有个缺点，即对于所提供的每个服务，就必须运行一个监听某个端口上连接发生的端口监控程序，这通常意味着象交换空间那样的系统资源的浪费。

因而，几乎所有的 UN*X 安装程序运行一个“超级服务器”，它为许多服务创建套接字，并且使用 *select(2)*系统调用同时监听所有这些端口。当远程系统请求一个服务时，超级服务器注意到这点并且会产生该端口的服务器程序。

常用的超级服务器是 *inetd*，即 Internet Daemon（Internet 端口监听程序）。它是在系统引导时启动的，并且从名为*/etc/inetd.conf* 的启动文件中取得它所要管理的服务的列表。除了上面所述的那些调用的服务器程序以外，还有许多由 *inetd* 自己处理的一般服务，称作内部服务（*internal services*）。这些包括简单地产生字符串的 *chargen*、返回系统日期时间的 *daytime*。

该文件中的一行是由以下几个字段组成的：

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

每个字段的含义如下：

service

给出服务的名称。通过查找*/etc/services* 文件，服务名称可以转换成一个端口号。这个文件将在 *services 和 protocols* 文件一节（10.3 节）中给予描述。

type

指定一个套接字类型，或者是 *stream*（对于基于连接的协议）（for connection-oriented protocols）或者是 *dgram*（对于基于数据报的协议）（for datagram protocols）。因此对于基于 TCP 的服务总是使用 *stream*，而基于 UDP 的服务总是使用 *dgram*。

protocol

命名服务所使用的传输协议。这必须是 *protocols* 文件中能找到的有效协议名称，也将在下面给出解释。

wait

该选项只用于 *dgram* 套接字。它可以是 *wait* 或 *nowait*。如果指定了 *wait*，那么对于指定的端口 *inetd* 任何时候只执行一次服务器程序。否则的话，在执行了指定的服务器以后，它将立刻在这个端口上进行监听。

这对于读取所有进入的数据报，然后退出的“单线程”服务器程序来讲是很有用的。许多 RPC 服务器程序是这种类型的，因而对它们需指定为 *wait*。对于相反的类型，“多线程”服务器程序允许无限数量的例程并发运行；这是很少用到的。这种服务器程序需指定为 *nowait*。*stream* 套接字应该总是使用 *nowait*。

user

这是用户的登录 id，用户的进程就在其下运行。这通常是 **root** 用户，但某些服务可以使用不同的帐号。这里使用最小特权原则是个很好的主意，这是指你不应该在一个享有特权的帐号下运行一个命令，如果该程序不需要特权也能正常工作的话。例如，NNTP 新闻服务器程序将作为 **news** 运行，而可能引起安全危险问题的服务（比如 *tftp* 或 *finger*）经常作为 **nobody** 运行。

server

给出将被执行的服务器程序的全路径。内部服务用关键字 *internal* 标记。

cmdline

这是传给服务器程序的命令行（参数）。这包括参数 0 –命令名。通常，这是是服务器程序名，除非当使用一个不同的名字调用时，该程序有不同的行为。对于内部服务该字段是空的。

```
#
# 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
```

图 9.1 一个/etc/inetd.conf 样本文件。

图 9.1 给出了 *inetd.conf* 的一个样本文件。*finger* 服务被注释掉了，所以这个服务就不存在了。这样做通常是为了安全方面的原因，因为它可能被入侵者用来获取你系统上的用户名。

tftp 同样也被注释掉了。*tftp* 实现原始文件传输协议（*Primitive File Transfer Protocol*）它允许无须口令检查直接从你的系统上传输任何可读的文件等等。这对于/etc/passwd 文件尤其有害，当没有使用影子口令文件时就更没的说了。

TFTP 通常用于无盘客户以及 X 终端从一个引导服务器上下载代码。如果由于这个原因你需要运行 *tftpd* 的话，请通过在 *tftpd* 的命令行上增加那些目录名来限制它们的范围到客户获取文件的目录中。见例子中的第二个 *tftp* 行所示。

9.2 *tcpd* 访问控制工具

由于对网络访问开放一个计算机包含许多安全方面的风险，所以应用程序被设计成能够防卫几种类型的攻击。然而，其中的某些防卫可能有缺陷（最为彻底的例子是 RTM Internet 蠕虫），或者并不区分请求能被接受的特定服务的安全主机与请求将被拒绝的不可靠的主机。上面我们已经概要地讨论过了 *finger* 和 *tftp* 服务。因此，人们想限制这些服务只有“可信的主机”才能访问，而使用通常的设置是不可能做到的，对于这些服务 *inetd* 要么提供给所有的客户，要么一个客户都不能使用。

为了达到此目的，一个有用的工具是 *tcpd*，[1] 即所谓的端口监控程序包装器（daemon wrapper）。对于你想监视和保护的 TCP 服务，将调用这个 *tcpd* 而不是服务器程序。*tcpd* 将请求记录到 *syslog* 后台程序，检查远程主机是否允许使用那个服务，并且只有检查通过时才会执行真正的服务器程序。注意，这对于基于 UDP 的服务不起作用。

例如，要包装 finger daemon 时，你必须改变 indetd.conf 中的相应行成为

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

不增加任何访问控制，这将如通常的 *finger* 设置一样展现在客户面前，除了任何请求将被记录到 *syslog* 的 *auth* 设施中。

访问控制是通过两个文件*/etc/hosts.allow* 和*/etc/hosts.deny* 来实现的。它们分别含有对特定服务和主机允许和禁止访问的条目。当 *tcpd* 处理来自名为 **biff.foobar.com** 的客户主机的服务请求时，比如说是 *finger* 服务请求，它将 *hosts.allow* 和 *hosts.deny*（以这个次序）文件中扫描匹配这个服务和主机的条目。如果在 *hosts.allow* 中找到了匹配条目，就准予访问，而不管 *hosts.deny* 中的任何条目了。如果在 *hosts.deny* 中发现了匹配的条目，该请求将通过关闭这个连接而被拒绝。如果都没有找到匹配的条目，也准予访问。

在访问文件中的条目看上去象这样：

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

servicelist 是来自于*/etc/services* 中的服务名称的一个列表，或者是关键字 *ALL*。要匹配除了 *finger* 和 *tftp* 的所有服务时，使用“*ALL EXCEPT finger, tftp*”。

hostlist 是一个主机名或 IP 地址的列表，或者是关键字 *ALL*、*LOCAL* 或 *UNKNOWN*。*ALL* 匹配任何主机，而 *LOCAL* 匹配不含有点的主机名。[2] *UNKNOWN* 匹配于任何名字或地址查询失败的主机。一个以点开头的名字与域名等于这个名字的主机匹配。例如，**.foobar.com** 与 **biff.foobar.com** 相

匹配。也有对 IP 网络地址和子网号的规定。详细信息请参见 *hosts_access(5)*手册页。

除了本地主机以外，要限制所有主机对 *finger* 和 *tftp* 服务的访问，就将下面一行加入到 */etc/hosts.deny* 文件中，并且*/etc/hosts.allow* 文件为空：

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

可选的 shellcmd 字段可以含有一个 shell 命令，当条目匹配时将被调用。这对于设置可以使潜在的入侵者暴露的陷阱是很有用的：

```
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* 参数被 *tcpd* 分别展开成客户的主机名和服务名称。详细信息请参见 *hosts_access(5)*手册页。

9.3 services 和 protocols 文件

确定的“标准”的服务端口号是在“Assigned Numbers”RFC 中定义的。为了使得服务器和客户程序能够将服务名称转换成这些号码（数值），每个主机起码要保存这个表的一部分；它被存于文件 /etc/service 中。每个条目是由象下面这行一样组成的：

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

这里，*service* 指定服务的名称、*port* 定义提供该服务的端口、*protocol* 定义使用何种传输协议。通常 *protocol* 是 *udp* 或 *tcp* 两者之一。为多种协议提供一个服务是可能的，同样，只要协议不同，在相同的端口上提供不同的服务也是可能的。Aliases 字段用于为相同的服务指定另一个可选用的名称。

通常，你无须更改你的 Linux 系统上的随同网络软件而来的 services 文件。不过，我们下面文件中给出一个小例外。

```
# 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，例如，对于 TCP 和 UDP，都在端口 7 上提供了 echo 服务，并且端口 512 被用于两个不同的服务，即在 UDP 上是 COMSAT daemon（它用于提醒用户有新邮件到达，见 *xbiff(1x)*），在 TCP 上是用于远程执行（*rexec(1)*）。

与 *services* 文件相类似，网络库文件需要有一个方法来将协议名—例如，那些用于 *services* 文件中的—转换成为能够被别的主机的 IP 层理解的协议号。这是通过在*/etc/protocols* 文件中查找协议名来做到的。这个文件每行包含一个条目，每个条目包括一个协议名和一个相关的号码。与/etc/services 文件相比，就更不用对这个文件作任何改动了。下面给出一个样本文件：

```
#
# Internet (IP) protocols
#
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
udp    17     UDP           # user datagram protocol
raw    255    RAW           # RAW IP interface
```

9.4 远程过程调用

客户-服务器应用程序的一个非常普通的机制是有 RPC 提供的，远程过程调用（*Remote Procedure Call*）软件包。RPC 是由 Sun Microsystems 开发出来的，是工具和库函数的一个汇集。建立在 RFC 上的重要应用程序是 NFS—网络文件系统、以及 NIS—网络信息系统，这两者将在后面的章节中给予介绍。

一个 RPC 服务器是由一个过程的集合组成，客户可以通过向服务器发出一个带有过程参数的 RPC 请求来调用这些过程。服务器将代表客户调用指定的过程，如果有任何返回值时就返回它。为

了与机器无关，所有在客户和服务器之间交换的数据都被发送者转换成所谓的*外部数据表示格式*（*External Data Representation format*）（XDR），并有接收者在转回机器本地表示方式。

有时候，对一个 RPC 应用程序的改进会给过程调用的接口带来不兼容性的改变。当然，只是简单地改变服务器将导致所有期望原先行为的应用程序不能使用。因此，RPC 程序有相应的版本号，通常是从 1 开始的，并且随着每一个新版本的 RPC 接口这个计数值将增加。经常，一个服务器可能同时提供几个版本；因此客户通过在他们请求中的版本号来指定他们想使用服务的哪个实现。

在 RPC 服务器和客户之间的网络通信有些奇特。一个 RPC 服务器提供一个或多哥过程的集合；每个集合被称作一个程序（*program*），并且唯一地由一个程序号（*program number*）指定。映射服务名到程序号的一个列表通常存于*/etc/rpc* 中。它的一个摘要在下面图 9.2 中给出。

```
#
# /etc/rpc – miscellaenous 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
```

图 9.2 一个/etc/rpc 样本文件

在 TCP/IP 网络中，RPC 的作者所面临的问题是将程序号映射到普通的网络服务上。他们为每个程序的每个版本选择同时提供 TCP 和 UDP 端口服务。一般来讲，RPC 应用程序将使用 UDP 来发送数据，并且仅当数据不能放入单个 UDP 数据报时才转回来使用 TCP 传输数据。

当然，客户程序必须有一个方法来找出哪个程序号映射到哪个端口上。如果使用一个配置文件来做这事将显得太不灵活了；因为 RPC 应用程序不使用保留的端口，无法保证一个原先被我们的数据库应用程序使用的端口没有被某个其它进程占用。因此，RPC 应用程序抓取它能得到的任何端口，并且用所谓的 *portmapper daemon* 进行登记注册。后者为所有运行在它的机器上的 RPC 服务器充当代理的角色：一个希望用给出的程序号联系一个服务的客户将首先询问服务器主机上的 *portmapper*，*portmapper* 将返回能够到达这个服务的 TCP 和 UDP 端口号。

这个方法有一特别的缺点，它会带来一个失败点，正象在标准 Berkeley 服务中 *inetd* 端口监控程序一样。然而，这个情况甚至有些更糟，因为当 *portmapper* 死去时，所有的 RPC 端口信息都将丢失；这通常意味着你必须手工重新启动所有的 RPC 服务器程序，或者干脆重新启动整个机器。

在 Linux 中，portmapper 被称作 *rpc.portmap* 并且在*/usr/sbin* 中。它是从 *rc.inet2* 中启动的，其它可以确定的是 portmapper 不用做任何配置工作。

9.5 配置 *r* 命令

有许多命令用于在远程主机上执行命令。这些是 *rlogin*、*rsh*、*rcp* 以及 *rcmd*。它们都在远程主机上产生一个 shell 并允许用户执行命令。当然，客户需要在执行命令的主机上有一个帐号。因此所有这些命令都要执行授权认证过程。通常，客户将告知服务器用户的登录名，服务器随之会请求一个使用常用方法鉴定的口令。

然而，有时候希望放宽对特定用户的认证检查。比如，如果你必须频繁地登录进你的 LAN 上的其它机器上，你可能希望不要每次都键入密码而被接纳。

禁用授权认证仅在有很少一些口令数据库是同步的主机的情况下、或是对于很少部分特权用户他们由于管理方面的原因需要访问许多主机的情况下才是可取的。每当你想要允许人们不用指定登录 id 或口令就能登录进你的主机时，确信你没有意外地授权准许别人访问你的机器。

对于 *r* 命令有两种方法来禁用认证检查。一个方法是超级用户允许特定的或所有的用户在特定的或所有的主机上（后者肯定是个坏点子）登录而无须键入口令。这种访问是由*/etc/hosts.equiv* 文件控制的。它含有等同于在本地主机上的用户的一张主机和用户名列表。另一个方法是一个用户在特定的主机上准许其他用户访问她的帐号。这些可以在用户的主目录中列于*.rhosts* 文件里。考虑到安全方面的原因，这个文件必须由该用户或超级用户掌管，并且不能是一个符号连接，否则的话它将被忽略掉。[3]

当一个客户请求一个 *r* 服务时，就会在*/etc/hosts.equiv* 文件中搜寻她的主机和用户名，然后会在她想作为其登录的用户的*.rhosts* 中搜寻。作为一个例子，假设 **janet** 是在 **gauss** 上工作并且试图在 **euler** 上登录进 **joe** 的帐号中。在下面，我们将 Janet 看作为一个*客户*用户，而将 Joe 看作是一个*本地*用户。现在，当 Janet 在 **gauss** 上键入下面一行时

```
$ rlogin –l joe euler
```

服务器将首先检查 *hosts.equiv* 如果 Janet 被准许自由访问，并且如果这个检查失败了，就会在 joe 主目录的*.rhosts* 文件中查找她。

euler 上的 *hosts.equiv* 文件看上去象这样：

```
gauss
euler
-public
quark.physics.groucho.edu      andres
```

每一个条目由一个主机名和一个可选的用户名组成。如果一个条目上只有一个主机名，那么那个主机上的所有用户使用他们的本地帐号都将被接受而无须作任何检查。在上面的例子中，当从 **gauss** 过来时，Janet 将被允许登录到她的帐号 janet 中，对于任何其他用户也是这样的，除了 **root** 用户。然而，如果 Janet 想以 joe 登录时，她将象平常一样得到一个输入口令的提示信息。

如果一个主机名后跟一个用户名的话，就如上面例子文件中的最后一行，那么除了 **root** 帐号以外，这个用户将可以无口令地访问其他所有的帐号。

主机名前也可以有一个减号，就象条目“**-public**”。这表明对于 **public** 上的所有帐号都需要授权认证，而不管个别用户在它们的*.rhosts* 文件中被准许访问。

.rhosts 文件的格式与 *hosts.equiv* 的格式是一样的，但是意思稍微有些不同。考虑 **euler** 上 Joe 的*.rhosts* 文件：

```
chomp.cs.groucho.edu
gauss     janet
```

第一个条目准许 **joe** 在从 **chomp.cs.groucho.edu** 登录进来时能自由地访问，但并不影响（侵犯）**euler** 或 **chomp** 上的任何其它帐号。第二个条目稍微有些不同，它同意 **janet** 在从 **gauss** 登录进来时可以自由访问 Joe 的帐号。

注意，客户的主机名是通过逆向映射呼叫者的地址到名字而获得的，所以这个特性对于解析器未知的主机来说是没有用的。在以下几种情况下，客户的主机名被认为是与 hosts 文件中的名字匹配的：

- 客户的规范主机名（不是一个别名）与文件中的主机名相匹配。
- 如果客户主机名是一个全资域名(比如当你正运行着 DNS 时解析器的返回值)，并且与 hosts 文件中的主机名不是逐字符匹配的，那么就与用本地域名扩展的主机名相比较。

注释

[1] 由 Wietse Venema 编制，wietse@wzv.win.tue.nl。

[2] 通常只有从/etc/hosts 中查到的本地主机名不含有点。

[3] 在一个 NFS 环境当中，你可能需要给它一个 444 的属性保护，因为超级用户对通过 NFS 加载的磁盘上文件的访问经常是非常严格的。

[4] 注意，当某人试图以 root 登录时，*hosts.equiv* 文件是不会被搜寻的。

第十章 网络信息系统

当你运行着一个局域网时，你的整个目标通常时为你的用户提供一个清晰透明的网络环境。做到这一步的一个重要步骤是在所有主机之间保持重要数据（比如用户帐号信息）的同步。我们在前面已经看到，对于主机名的解析，存在一个强大而复杂的服务，即 DNS。对于其他任务没有这种特殊的服务。此外，如果你只是管理一个没有 Internet 连接的小 LAN 的话，那么对于许多管理员来说安装设置 DNS 是不值得的。

这就是为什么 Sun 开发出了 NIS，即*网络信息系统（Network Information System）*。NIS 提供了通用数据库访问设施，它可用来想你的网络上的所有主机分发信息，比如象 *passwd* 和 *groups* 文件所包含的信息。这使得网络看起来象一个独立系统，在所有的主机上有着相同的帐号。以同样的方式，你可以用 NIS 向网络上的所有机器分发来自*/etc/hosts* 中的主机名信息。

NIS 是基于 RPC 的，是由一个服务器、一个客户端库以及几个管理工具组成。起初，NIS 被称作*黄页（Yellow Pages）*，或 YP，现在仍然使用这个名称来非正式地指这项服务。另一方面，Yellow Pages 是英国电信的商标，英国电信一直要求 Sun 更换这个名字。随着事态的发展，某些名称已与人们分不开了，所以 YP 一直以与 NIS 相关命令的前缀形式继续存在着，比如象 *ypserv*、*ypbind* 等等。

今天，几乎所有的 UN*X 都包括 NIS，而且甚至有它的免费实现版本。一个是来自 BSD 的 Net-2 发行版，源自于 Sun 捐赠的公众域参考实现。该版的客户库代码已经存在于 GNU 的 *libc* 中很长时间了，而管理程序只是在最近才由 Swen Thümmler [1] 移植到 Linux 上。在这个参考实现中漏掉了一个 NIS 服务器程序。Tobias Reber 已经编制出了另外一个 NIS 软件包，其中包括所有的工具和一个服务器；该软件包称作 *yps*。[2]

目前，一个完全重写的称为 NYS 的 NIS 代码已由 Peter Eriksson [3]编制出来，它支持普通的 NIS 和 Sun 的经过许多修正的 NIS+。NYS 不仅提供了一个 NIS 工具集和一个服务器，而且还增加了一个全新的库函数集，这个库函数集可能最终会被加入到标准 *libc* 中。这包括替换目前使用 *host.conf* 的主机名解析的一个新设置方案。这些函数的特性将在下面讨论。

这一章将集中讨论 NYS 而非另外两个软件包，对于这两个软件包我将称它们为“传统的”NIS 代码。如果你确实要想运行任何这些软件包的话，本章中的说明也许已经足够也许还不够。要获取另外的信息，请取得一本有关 NIS 的标准（权威）书本，比如象 Hal Stern 的 *NFS 和 NIS*（见[Stern92]）。

目前，NYS 仍处于开发阶段，因此标准的 Linux 工具如网络程序或 login 程序还没有注意到 NYS 的配置方案。只有到 NYS 合并进主流 *libc* 中时，如果你想使得所有这些执行程序使用 NYS 时，你才需要重新编译它们。在任何这些应用程序的 Makefiles 中，在 *libc* 之前，指定*-lnsl* 作为 linker 的最后一个选项。这将有关函数从 libnsl—NYS 库中连接过来，取代从标准 C 库的连接。

10.1 理解 NIS

NIS 在所谓的包含键-值对的 maps 中保存数据库信息。Maps 被存储于运行 NIS 服务器的中央主机中，从该主机中，客户可以通过各种 RPC 调用检索信息。最频繁地，maps 是存于 DBM 文件中的。[4]

Maps 本身是从主要文本文件（比如*/etc/hosts* 或*/etc/passwd*）中生成的。对于某些文件，会生成几个 maps，每个搜寻键类型对应一个。例如，你可以为主机名和 IP 地址搜查 *hosts* 文件。相应地，

从中会生成两个 NIS maps，分别称为 *hosts.byname* 和 *hosts.byaddr*。表 10.1 列出了通用 maps 和它们生成的文件。

Master File	**Map(s)**
/etc/hosts */etc/networks* */etc/passwd* */etc/group* */etc/services* */etc/rpc* */etc/protocols* */usr/lib/aliases*	*Hosts.byname* *hosts.byaddr* *Networks.byname* *networks.byaddr* *Passwd.byname* *passwd.byuid* *Group.byname* *group.bygid* *Services.byname* *services.bynumber* *Rpc.byname* *rpc.bynumber* *Protocols.byname* *protocols.bynumber* *Mail.aliases*

表 10.1 一些标准的 NIS maps 以及相应的文件。

在某些 NIS 软件包或其它软件中，还有一些你可能会觉得有用的别的文件和 maps。这些文件和 maps 可能含有没在这本书中讨论过的应用程序的信息，比如可能用于某些 BOOTP 服务器中的 *bootparams* maps，或者在 Linux 中目前不含有任何函数的文件（就象 *ethers.byname* 和 *ethers.byaddr* maps）。

对于某些 maps，人们通常使用*绰号*（*nicknames*），它们很短因而易于键入。要想获得一个你的 NIS 工具能够理解的绰号的完整列表，运行下面的命令：

```
$ ypcat –x
NIS map nickname translation table:
       “passwd” -> “passwd.byname”
       “group” -> “group.byname”
       “networks” -> “networks.byaddr”
       “hosts” -> “hosts.byname”
       “protocols” -> “protocols.bynumber”
       “services” -> “services.byname”
       “aliases” -> “mail.aliases”
       “ethers” -> “ethers.byname”
       “rpc” -> “rpc.bynumber”
       “netmasks” -> “netmasks.byaddr”
       “publickey” -> “publickey.byname”
       “netid” -> “netid.byname”
       “passwd.adjunct” -> “passwd.adjunct.byname”
       “group.adjunct” -> “group.adjunct.byname”
       “timezone” -> “timezone.byname”
```

NIS 服务器传统地称为 *ypserv*。对于一个中等大小的网络来说，单个服务器通常就足够了；大型的网络可能需要在不同的网段以及不同的机器上运行几个服务器，以减轻服务器机器和路由器的负荷。通过将这些服务器之一作为*主服务器*（*master server*），其它的服务器作为*次服务器*（*slave servers*），使得这些服务器同步。Maps 将只在主服务器上建立。从主服务器上将它们分发到所有次

服务器上。

你可能已经注意到，我们一直很含糊地论及“网络”；当然引用这样一个网络的 NIS 存在着与众不同的概念，也即通过 NIS 共享它们部分系统配置数据的所有主机的一个集合：NIS 域。不幸的是，NIS 域与我们在 DNS 中遇到的域绝对没有一点共同之处。为了在本章中避免含糊不清的情况，我将总是指出我说的哪一类型的域。

NIS 域只具有纯粹的管理功能。对于用户来说它们主要是不可见的，除了在域中所有机器之间口令的共享。因此，给 NIS 域取的名字仅与管理员有关。通常，可以使用任何名字，只要该名字与你的本地网络上的其它 NIS 域名不同就行。例如，虚拟酿酒厂的管理员可以选择建立两个 NIS 域，一个是给酿酒厂本身用的，另一个是个葡萄酒厂的，她分别将其命名为 **brewery** 和 **winery**。另一个很普遍的方案是简单地用 DNS 域名也作为 NIS 的域名。为了设置和显示你的主机的 NIS 域名，你可以使用 *dommainname* 命令。当不加任何参数调用时，它打印出当前 NIS 域名；如要设置这个域名的话，你必须成为超级用户并键入：

```
# domainname brewery
```

NIS 域决定了一个应用程序将查询哪个 NIS 服务器。例如，在葡萄酒厂（Winery）的主机上的 login 程序（当然）将只向葡萄酒厂的 NIS 服务器（或者是它们其中之一，如果存在多个服务器的话）查询用户的口令信息；而酿酒厂主机上的应用程序将只查询酿酒厂的服务器。

现在还有一个疑点要解决，也即一个客户如何知道要连接到哪一台服务器上去。最简单的途径是有一个配置文件，它给出了要在其上查找服务器的主机名。然而，这个办法非常不灵活，因为它不允许客户依据这些服务器存在与否使用不同的服务器（当然是指从同一个域）。因此，传统的 NIS 实现依赖于一个称作 *ypbind* 的特殊后台程序在它们的 NIS 域中来侦测一个适当的 NIS 服务器。在能够执行任何 NIS 查询之前，任何应用程序首先要从 *ypbind* 找出要使用哪个服务器。

ypbind 通过向本地 IP 网络广播来探测服务器；第一个响应的服务器假设基本上是最快的一个并将用于随后的 NIS 查询。在某个间隔时间过去以后，或者如果服务器不工作了，*ypbind* 将再次探测运行着的服务器。

现在，关于动态绑定的争论点是你很少需要它，并且它会带来安全方面的问题：*ypbind* 盲目地相信任何应答者，而这个应答者可能会是一个谦逊的 NIS 服务器也可能是一个怀有恶意的入侵者。不用说如果你在 NIS 上管理你的口令数据库的话，这将变成特别麻烦的事。为了防范这个问题，NYS 缺省地不使用 *ypbind*，而是从一个配置文件中取得服务器的主机名。

10.2 NIS 与 NIS+

NIS 和 NIS+除了在名字上和有共同的目标以外，很少有相同之处。NIS+是用一个完全不同的方法构成的。它使用一个类似于 DNS 的分级名字空间，而不是一个平面的名字空间和松散脱节的 NIS 域。它使用一个由行和列组成的所谓的*表*（*tables*）而不是 maps，在 NIS+数据库中表的每一行表示一个对象，而列表示 NIS+所知所关心的对象的那些属性。一个给定的 NIS+域的每个表由那些它们的父域组成。另外，表中的一个条目可以包含到另一个表的链接。这些特性使得用许多方法构造信息成为可能。

传统的 NIS 的 RPC 版本号是 2，而 NIS+的是版本 3。

NIS+至今似乎还没有被广泛地使用，而且我实际上对它也知道不多。（唔，几乎一窍不通）。由于这个原因，这里我们将不涉及它了。如果你对它感兴趣并想多学一点的话，请参阅 Sun 的 NIS+管理手册（[NISPlus]）。

10.3 客户边的 NIS

如果你熟悉编制或移植网络应用程序的话，你将会注意到上面所列出的许多 NIS maps 与 C 库中的库函数相对应。例如，要获得 *passwd* 信息，你通常使用 *getpwnam(3)*和 *getpwuid(3)*函数，它们分别返回与给定的用户名或数值用户 id 相对应的帐号信息。在通常的环境下，这些函数将在标准文件（比如*/etc/passwd*）中执行请求的查找。

然而，这些函数的基于 NIS（NIS-aware）的实现将更改这种行为，并且会启用一个 RPC 调用让 NIS 服务器查询用户名或 id。对于应用程序来说这个操作是完全透明的。这个函数可以将 NIS map “附加”或“替换”掉原始的文件。当然，这并没有对文件进行实际的修改，它只是让应用程序*看上去*好象该文件已经被替换或附加上去了。

对于传统的 NIS 实现来讲，对于那些 maps 替换掉以及那些被添加到原始信息中，曾有某些惯例。有些 maps（比如 *passwd* maps）需要对 passwd 文件进行杂凑地修改，当做错时，就会打开安全方面的缺口。为了避免这个缺陷，NYS 常规的配置方案，该方案确定了一个特定的客户函数集是否使用原始文件、NIS、NIS+，并且以什么次序使用。这将在本章后续小节中加以讨论。

10.4 运行一个 NIS 服务器

在这么多理论方面的喋喋不休之后，现在开始动手做实际的配置工作。在本节中，我们将讨论 NIS 服务器的配置。如果在你的网络上已经有一个 NIS 服务器在运行，你就不必设置你自己的服务器了；在这种情况下，你可以安全地跳过本节。

> 注意，如果你只是准备对服务器做试验，请确信你没有设置一个已经在你网络上使用的 NIS 域名。因为这会使整个网络服务瘫痪并使得许多人不高兴和恼怒。

对于 Linux 目前有两个现存的免费 NIS 服务器，一个包含在 Tobias Reber 的 *yps* 软件包中，另一个在 Peter Eriksson 的 *ypserv* 软件包中。至于你运行哪一个是无关紧要的，也不管你使用 NYS 还是目前在 *libc* 中的标准 NIS 客户代码。在写作本书时，*yps* 中的 NIS 次服务器处理的代码似乎更完善一些。所以如果你要涉及到次要服务器的话，*yps* 可能是一个更好的选择。

当在*/usr/sbin* 中安装好服务器程序（*ypserv*）以后，你应该建立一个目录，用于存放你的服务器分发的 map 文件。当为 **brewery** 域设置好一个 NIS 域时，maps 将存于*/var/yp/brewery* 中。服务器通过检测是否存在一个 map 目录来确定它是否在为一个特定的 NIS 域服务。如果你对某些 NIS 域禁用了服务，请确信同时也删除那个目录。

Maps 通常储存于 DBM 文件中以加速查询。它们是用一个称为 *makedbm*（对于 Tobias 的服务器）或 *dbmload*（对于 Peter 的服务器）的程序从主文件中创建的。它们是不可互换的。将主文件转换成 *dbmload* 可分析的形式通常需要一些 *awk* 或 *sed* 技巧，这对于录入有些乏味并且难于记忆。因此，Perter Eriksson 的 *ypserv* 软件包含有一个 *Makefile*（称为 *ypMakefile*），它将为你做所有这些工作。你应该将它作为 *Makefile* 安装在你的 map 目录中，并且编辑它，以反映你要分发的 maps。在文件的头部，你会发现 all 目标，它列出了 ypserv 将要提供的服务。缺省地，该行看上去象这样：

```
all: ethers hosts networks protocols rpc services passwd group netid
```

例如，如果你不想生成 *ethers.byname* 和 *ethers.byaddr* maps，只须简单地从这条规则中去掉 *ethers*

先决条件。为了测试你的设置，开始只使用一二个 maps，比如 *services.** maps，就已经足够了。

在 map 的目录里，在编辑好 *Makefile* 以后，键入“**make**”。这将自动地生成并安装 maps。你必须确信每当你改变了主文件之后，一定要更新 maps，否则所做的改变对网络仍然是不可见的。

下一节解释如何配置 NIS 客户代码。如果你的安装设置不工作的话，你应该查出有没有任何请求到达你的服务器。如果你对 NYS 服务器指定-D 命令行标志，它将在控制台上打印出有关所有进入的 NIS 查询的调试信息，并且返回结果。这些将给你一个提示来确定问题到底出在哪里。Tobias 的服务器没有这个选项。

10.5 使用 NYS 设置一个 NIS 客户

在本章的余下部分，我们将讨论 NIS 客户的配置。

你的第一步应该是告知 NYS 对于 NIS 服务使用哪个服务器，并在*/etc/yp.conf* 配置文件中设置好。对于在葡萄酒厂（Winery）网络上一台主机上的简单样本文件看上去象这样：

```
# yp.conf – YP configuration for NYS library.
#
domainname winery
server vbardolino
```

第一条语句告诉所有 NIS 客户，他们属于 **winery** NIS 域。如果你省略这一行，NYS 将使用你通过 *domainname* 命令指派给你系统的域名。*server* 语句指定所使用的 NIS 服务器。当然，与 **vbardolino** 相应的 IP 地址必须在 *hosts* 文件中设置；另外，你也可以在 *server* 语句中使用 IP 地址本身。

在上面所示的表单中，*server* 命令告诉 NYS 使用指定的服务器而不管目前的 NIS 域是什么。然而，如果你频繁地在不同的 NIS 域中移动你的机器的话，你可能想要在 yp.conf 文件中保存几个域的信息。你可以通过在 server 语句中增加 NIS 域名获得几个 NIS 域的服务器的信息。例如，你可以为一个便携机改变上面样本文件成这样：

```
# yp.conf – YP configuration for NYS library
#
server vbardolino winery
server vstout    brewery
```

这允许你在系统引导时通过 domainname 命令设置期望的 NIS 域来在两个域的任何一个域中使用便携机。

在创建了这个基本的配置文件并确信它是可读的以后，你应该运行的第一次测试来检查你是否能连接到你的服务器上。确信选择你的服务器分发的任何 map，如 *hosts.byname*，并试着使用 *ypcat* 工具来检索它。*ypcat*，就象所有其它的 NIS 管理工具一样，应该存在于*/usr/sbin* 中。

```
# ypcat hosts.byname
191.72.2.2    vbeaujolais   vbeaujolais.linus.lxnet.org
191.72.2.3    vbardolinovbardolino.linus.lxnet.org
191.72.1.1    vlager    vlager.linus.lxnet.org
```

```
191.72.2.1    vlager    vlager.linus.lxnet.org
191.72.1.2    vstout    vstout.linus.lxnet.org
191.72.1.3    vale        vale.linus.lxnet.org
191.72.2.4    vchianti    vchianti.linus.lxnet.org
```

你所得到的输出应该与上面显示的相象。如果你得到了一条错误信息指出“Can’t bind to server which serves domain”或者某些类似的信息，那么或者是你设置的 NIS 域名在 *yp.conf* 中没有匹配的服务器，或者是由于某些原因服务器找不到。在后一种情况下，请确信 *ping* 到那个主机产生正确的结果，并且它确实正在运行一个 NIS 服务器。你可以使用 rpcinfo 来验证后者，它将生成以下输出：

```
# rpcinfo –u serverhost ypserv
program 100004 version 2 ready and waiting
```

10.6 选择正确的 maps

在确信能够与 NIS 服务器联系之后，你必须决定要用 NIS maps 替换或添加哪个配置文件。一般地，你将会对主机和口令查找函数使用 NIS maps。前者对于没有使用 BIND 时特别有用。后者允许所有用户在 NIS 域的任何系统上登录进他们的帐号；这通常要求通过 NFS 在所有的主机之间共享一个中央*/home* 目录。这将在 10.7 节中详细讨论。其它的 maps，如同 *services.byname*，并没有如此有戏剧性的效能，但能为你省去某些编辑工作如果你安装了任何网络应用程序而该应用程序使用了一个不在标准 *services* 文件中的服务名。

通常，对于一个查找函数何时使用本地文件、何时询问 NIS 服务器，你会想有某些自由的选择。NYS 允许你配置函数访问这些服务的顺序。这是通过*/etc/nsswitch.conf* 文件来控制的，该文件名是指*名称服务交换*（*Name Service Switch*），当然其并不限制于名称服务。对于 NYS 支持的任何数据查找函数，它都包含指定所用服务的一行。

服务的正确顺序是与数据的类型有关的。并无必要让 *services.byname* 的 map 一定要含有与本地 *services* 文件中不同的条目；它可以包含更多的条目。所以，一个好的选择可以是首先查询本地文件，并且只有当服务名称没有找到时才查找 NIS。另一方面，主机名信息可能会非常频繁地改变，所以 DNS 或 NIS 服务器应该总是有非常正确的信息，而本地的 *hosts* 文件只作为在 DNS 和 NIS 不可用时的一个备份而已。在这种情况下，你可能想最后查询本地文件。

下面的例子显示出了如何以上面描述的方式配置 *gethostbyname(2)*、*gethostbyaddr(2)*和 *getservbyname(2)*函数。它们将依次试用列出的服务；如果一个查找成功，结果就返回，否则试用下一个服务。

```
# small sample /etc/nsswitch.conf
#
hosts:      nis dns files
services:   files nis
```

可以在 *nsswitch.conf* 文件中有一个条目的完整服务的列表如下面所示。实际被查询的 maps、文件、服务器和对象依赖于条目名。

nisplus 或 *nis+*

对这个域使用 NIS+服务器。服务器的位置从*/etc/nis.conf* 文件中获得。

`nis` 使用这个域的当前 NIS 服务器。被查询的服务器的位置在 *yp.conf* 文件中设置，见前节所示。对于 *hosts* 条目，要查询 *hosts.byname* 和 *hosts.byaddr maps*。

`dns` 使用 DNS 名字服务器。这个服务类型只对 *hosts* 条目有用。要被检索的名字服务器仍然由标准 *resolv.conf* 文件确定。

`files` 使用本地文件，比如对于 hosts 条目使用/etc/hosts 文件。

`dbm` 从位于*/var/dbm* 内的 DBM 文件中查找信息。文件所使用的名字与 NIS map 相对应。

目前，NYS 支持下面这些 *nsswitch.conf* 条目：*hosts*、*networks*、*passwd*、*group*、*shadow*、*gshadow*、*services*、*protocols*、*rpc* 和 *ethers*。以后还会增加更多的条目。

图 10.1 显示了一个更完整的例子，它引入了 *nsswitch.conf* 的另一个特性：*hosts* 条目中的*[NOTFOUND=return]*关键字通知 NYS，如果在 NIS 或 DNS 数据库中没有找到所要的项就返回。也即，只有在向 NIS 和 DNS 服务器的呼叫由于某些其它原因失败时，NYS 才将继续搜寻本地文件。因此，本地文件只在启动引导期间使用并且当 NIS 服务器关闭时起一个备份的作用。

```
# /etc/nsswitch.conf
#
hosts:     nis dns [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files

services:   files nis
protocols:  files nis
rpc:       files nis
```

图 10.1 nsswitch.conf 样本文件。

10.7 使用 *passwd* 和 *group* Maps

NIS 的一个主要应用是在一个 NIS 域中的所有主机上同步用户以及帐目信息。关于这方面，你通常只保存了一个小的本地*/etc/passwd* 文件，对于这个文件，从 NIS maps 获得的站点范围的信息被添加了进去。然而，只是简单地在 *nsswitch.conf* 中为这个服务启用 NIS 查找还不很够。

当引用 NIS 描述的口令信息时，你必须首先确信在你本地 *passwd* 文件中任何用户的数值用户 id 与 NIS 服务器的用户 id 匹配。同样对于其它目的你也会需要这样的，比如从你的网络中其它主机上加载 NFS 卷时。

如果*/etc/passwd* 或*/etc/group* 中的任何数值 id 与 maps 中的相偏离，你必须为属于那个用户的所有文件调整文件的所有权。首先你必须将 *passwd* 和 *group* 中的 uid 和 gid 改成一个新值；然后找出属于刚改变的用户的所有文件，最后改变这些文件的所有权。假设 **news** 曾有一个 id 为 9，而 **okir** 有一个 id 为 103，它们将被改成其它值；那么你可以发出以下的命令：

```
# find / -uid  9 –print >/tmp/uid.9
# find / -uid 103 –print >/tmp/uid.103
# cat /tmp/uid.9  | xargs chown news
# cat /tmp/uid.103 | xargs chown okir
```

必须针对*新*安装的 *passwd* 文件执行这些命令，并且在改变任何文件的所有权之前收集所有文件的名字，这点很重要。为了更新文件的组所有权，你将使用一个类似的命令。

在做完这些工作之后，你系统上的数值 uid 和 gid 将与你的 NIS 域中所有其它主机上的相匹配。下一步将是在 nsswitch.conf 中增加配置行，它为用户和组信息启用 NIS 查找：

```
# /etc/nsswitch.conf – passwd and group treatment
passwd: nis files
group:  nis files
```

这使得在一个用户试图登录时，login 命令和所有其它类似命令首先查询 NIS maps，如果这个查找失败时，再返回使用本地文件。一般来讲，你将从你的本地文件中删除所有的用户，而只留下 **root** 和象 **mail** 一样的通用帐目。这是因为某些至关重要的系统任务可能需要将 uid 映射到用户名上或者反之。例如，管理用的 *cron* 作业可能会执行 su 命令来临时变成 **news**，或者 UUCP 子系统可能要邮寄一个状态报告。如果 **news** 和 **uucp** 在本地 *passwd* 文件中没有条目了，那么在 NIS 不能使用期间这些作业将糟糕地失败。

这里有两个大告戒：一方面，上面所描述的设置在这里只适应于没有使用影子（shadow）口令的登录状况，象那些包括在 *util-linux* 软件包中的。与 NIS 一起使用影子口令的复杂问题将在下面论及。另一方面，登录命令并不是仅有的访问 *passwd* 文件的命令—请看许多人几乎一直使用的 *ls* 命令。每当进行一次长列表时，*ls* 将显示一个文件的用户和组的宿主的符号名；也即，对于它遇到的每个 uid 和 gid，它就要查询 NIS 服务器一次。如果你的本地网络受到阻塞时将严重地拖延进行的工作，或者更糟糕的是，当 NIS 服务器不在同一个物理网络上时，数据报还必须通过路由器传输。

事情还没结束。想象以下如果一个用户想要更改她的口令时会发生什么情况。通常，她会执行 *passwd*，它将读入新的口令并更新本地 *passwd* 文件。对于 NIS 来说，这是不可能的，因为这个文件已不再存在于本地了，但是每当用户想要改变他们的口令时就让他们登录进 NIS 也不是个选择。因此，NIS 提供了一个对 *passwd* 的混入替换称为 *yppasswd*，它用来在目前的 NIS 中做类似的工作。为了改变服务器主机上的口令，它通过 RPC 联系那个主机上的 *yppasswdd* 后台程序，并向它提供更新过的口令信息。通常，你通过象这样做在常规程序上安装 *yppasswd*：

```
# cd /bin
# mv passwd passwd.old
# ln yppasswd passwd
```

与此同时你必须在服务器上安装 *rpc.yppasswdd* 并从 *rc.inet2* 中启动它。这将对你的用户有效地隐藏 NIS 所带来的任何扭曲。

10.8 使用支持影子（shadow）的 NIS

至今还没有对使用影子登录程序组的站点的 NIS 支持。John F. Haugh，影子程序组的作者，最

近往 **comp.sources.misc** 发布了一个受 GNU 库的 GPL 保护的影子库函数的一个版本。它对 NIS 已经有了一些支持，但还不完整，并且这些函数还没有加入到标准 C 库中。另一方面来讲，通过 NIS 之类公布来自于*/etc/shadow* 中的信息是与 shadow 组件的目的相违背的。

尽管 NYS 口令查找函数不使用 *shadow.byname* map 或任何这类 map，NYS 还是支持透明地使用一个本地*/etc/shadow* 文件的。当 *getpwnam* 的 NYS 实现被调用来查找与给定的登录名相关的信息时，*nsswitch.conf* 中的 *passwd* 条目所指定的设施被检索。*nis* 服务将简单地在 NIS 服务器的 *passwd.byname* map 中查找这个名字。而 *files* 服务将检查*/etc/shadow* 是否存在，并且如果存在的话，就试着打开它。如果不存在的话，或者如果用户没有 **root** 特权的话，它就返回到只在*/etc/passwd* 中查找用户信息的传统的处理方法中。然而，如果 *shadow* 文件存在，并且能被打开的话，NYS 将从 *shadow* 中抽取用户的口令。*getpwuid* 函数也是这样实现的。在这种方式下，用 NYS 编译的执行文件将透明地处理本地影子组件的安装。

10.9 使用传统的 NIS 代码

如果你使用目前在标准 C 库中的客户代码的话，那么配置一个 NIS 客户就稍微有些不同。一方面，它使用一个 *ypbind* 后台程序（daemon）来广播查询运行着的服务器而不是从一个配置文件中取得（服务器）信息的。因此，你必须确信在启动期间开始运行 *ypbind*。它必须在 NIS 域已被设置好并且 RPC portmapper 已启动后才被调用。此时，上面所示的调用 *ypcat* 进行对服务器测试才能工作。

最近，有许多有关 NIS 出错报告（bug reports），出错信息说“`clntudp_create: RPC: portmapper failure – RPC: unable to receive`”。这是由于对 ypbind 与库函数有关绑定信息的通信（沟通）方式的不兼容的改动。取得最新有关 NIS 工具的最新源程序并重新编译之可以解决这个问题。[5]

同样，传统的 NIS 确定是否要和如何将 NIS 信息与本地文件中的信息合并的方法与 NYS 中所使用的方法是有偏差的。例如，为了使用 NIS 口令 maps，你必须在*/etc/passwd* map 中包含下列行：

```
+:*:0:0:::
```

这标记出口令查找函数“插入”NIS maps 的地方。往*/etc/group* 中插入类似的一行（去掉最后两个冒号）会对 *group.** maps 做出同样的事。为了使用 NIS 分发的 *hosts.** maps，只要改动 *host.conf* 文件中的 *order* 一行。例如，如果你要使用 NIS、DNS 以及/etc/hosts 文件（以这个顺序），你必须将这行改成

```
order yp bind hosts
```

目前，传统的 NIS 实现不支持任何其它的 maps。

注释

[1] 可以用swen@uni-paderborn.de与他联系。NIS 客户程序以 yp-linux.tar.gz 的形式在 matlab.unc.edu

上的 *system/Network* 中有。

[2] 当前的版本（在写作本书时）是 yps-0.21 并且可以从ftp.lysator.liu.se的*/pub/NYS* 目录中取得。

[3] 可以用pen@lysator.liu.se与他联系。

[4] DBM 是一个简单的数据库管理库，它使用杂凑技术（hashing technigues）来加速查询操作。GNU 计划有它的一个免费实现，称为 *gdbm*，它已包括在大多数 Linux 发行版中。

[5] 可以从ftp.uni-paderborn.de的*/pub/Linux/LOCAL* 目录中取得 *yp-linux* 源程序。

第十一章 网络文件系统

NFS，网络文件系统（network filesystem），或许是使用 RPC 最突出的网络服务了。它允许你以访问任何本地文件一样的方法来访问远程主机上的文件。这是通过将客户端的内核功能（它使用远程文件系统）与服务器端的 NFS 服务器功能（它提供文件数据）相混合而成为可能的。这种文件访问对客户来说是完全透明的，并且可在各种服务器和各种主机结构上工作。

NFS 提供的许多优点：

- 被所有用户访问的数据可以存放在一台中央主机上，由客户在引导启动时加载这个目录。例如，你可以将所有用户的帐目存放在一台主机上，让你的网络上的所有用户从那台主机上加载*/home* 目录。如果也安装了 NIS 的话，用户就可以登录进任何系统上，而始终在一组文件上工作。
- 需要耗费大量磁盘空间的数据可以被保存在一台主机上。例如，所有有关 LaTeX 和 METAFONT 的文件和程序可以在一个地方保存和维护。
- 管理用的数据可以存放在单个主机上。不再需要使用 *rcp* 将相同的文件安装到 20 个不同的机器上。

NFS 在很大程度上是由 Rick Sladkey 做出的，[1] 他编制了 NFS 的内核代码和 NFS 服务器的大部分。后者源自于由 Mark Shand 编制的 *unfsd* user-space NFS 服务器和 Donald Becker 编制的 *hnfs* Harris NFS 服务器。

现在让我们观看一下 NFS 是如何工作的：一个客户可以请求将一台远程主机的目录加载到一个本地目录上，正如用加载一个物理设备同样的方法。然而，用于指定远程目录的句法是不同的。例如，为了主机 **vlager** 的*/home* 加载到 **vale** 的*/users* 上，管理员在 **vale** 上要发出以下的命令：[2]

```
# mount –t nfs vlager:/home /users
```

此时，*mount* 将试图通过 RPC 联系 **vlager** 上的 *mountd* mount 后台程序。服务器将检查 **vale** 是否允许加载所考虑的目录，如果行的话，就返回一个文件句柄（file handle）。这个文件句柄将被用于所有随后对*/users* 下的文件的请求。

当某人通过 NFS 访问一个文件时，内核就将一个 RPC 调用送至服务器机器上的 *nfsd*（NFS 后台程序）。该调用将文件句柄、要访问的文件名以及用户的 user 和 group id 作为参数。这些用于确定对指定文件的访问权限。为了避免非授权用户读取或修改文件，user 和 group id 在两台主机上必须相同。

在许多 UN*X 实现中，NFS 的客户和服务器功能是作为内核层的后台程序实现的，是在系统引导时从用户空间启动的。NFS 后台程序（*nfsd*）在服务器主机上，*Block I/O* 后台程序（*biod*）在客户主机上。为了提高吞吐率，biod 使用预读（read-ahead）和后写（迟写）（write-behind）来执行同步 I/O；同样，几个 *nfsd* 后台程序通常是并发运行的。

NFS 的 Linux 实现稍微有些不同，客户代码被紧密地集成到内核的虚拟文件系统中（VFS）并且不需要通过 *biod* 进行另外的控制。另一方面，服务器代码完全在用户空间运行，所以同时运行该服务器的几个拷贝几乎是不可能的—因为这将涉及到同步问题。目前 Linux 的 NFS 还缺乏预读和后写机制，但 Rick Sladkey 计划今后将添加进去。[3]

Linux 的 NFS 代码的最大问题是 Linux 的 1.0 版内核不能分配大于 4K 的内存块；其结果是，网络代码不能处理除去头大小等数据后大于 3500 左右字节的数据报。这意味着对于使用默认的大的

UDP 数据报的系统（例如，在 SunOS 上是 8K），从其上运行的 NFS 后台程序上传回的数据，需要人工地切成小块。在某些环境下这会严重地损害系统性能。[4] 这个限制在最近的 Linux-1.1 内核中已不复存在，并且客户代码也已进行了修改以克服这个问题。

11.1 准备 NFS

在你能够使用 NFS 之前，不管是作为服务器还是客户，你必须确信你的内核已将 NFS 支持编译进去了。对此，较新的内核在 proc 文件系统上有一个简单的接口，即*/proc/filesystems* 文件，你可以使用 *cat* 来显示它：

```
$ cat /proc/filesystems
minix
ext2
msdos
nodev   proc
nodev   nfs
```

如果这个列表中没有 nfs，那么你必须编译你自己的启用了 NFS 的内核。配置内核网络选项在第三章的“内核配置”一节中作过解释。

对于 Linux 1.1 以前的内核，要找出你的内核是否启用了 NFS 支持的最容易的方法是实际地试着加载一个 NFS 文件系统。对于此，你可以在/tmp 下建立一个目录，并试着往上加载一个本地目录：

```
# mkdir /tmp/test
# mount localhost:/etc/ tmp/test
```

如果这个加载尝试失败了，并有一段出错信息指出“`fs type nfs no supported by kernel`”，那么你就必须制作一个启用了 NFS 的新内核。任何其它的出错信息都完全无关紧要，因为你还没有在你的主机上配置 NFS 后台程序。

11.2 加载一个 NFS 卷

NFS 卷[5]的加载方法与普通文件系统的加载方式是一样的。你使用下面的句法调用 *mount*：

```
# mount –t nfs nfs_volume local_dir options
```

`nfs_volume` 是以 *`remote_host:remote_dir`* 形式给出。由于这个表示法对 NFS 文件系统来说是唯一的，你可以省略`-t nfs` 选项。

在加载一个 NFS 卷时，你可以指定许多别的选项。这些选项可以在命令行上在`-o` 开关后面给出，或者在*/etc/fstab* 该卷条目的选项字段内给出。在这两种情况下，多个选项是用逗号彼此分开的。在命令行上的选项总是覆盖 *fstab* 文件中给出的选项。

在/etc/fstab 中的一个样本条目可以是

```
# volume            mount point      type  options
news:/usr/spool/news  /usr/spool/news   nfs   timeo=14,intr
```

然后，这个卷可以使用下面命令被加载

```
# mount news:/usr/spool/news
```

如果 *fstab* 中没有相应的条目，那么 NFS 的 *mount* 调用看上去就有点乱。例如，假如你从一台称为 **moonshot** 的机器上加载你的用户主目录，**moonshot** 机器使用一个默认的 4K 大小的块来进行读/写操作。你可以发出如下命令将块的大小减至 2K 以适应 Linux 的数据报大小的限制

```
# mount moonshot:/home /home -o rsize=2048,wsize=2048
```

在随 Rick Sladkey 的基于 NFS 的 *mount* 工具（可以在 Rik Faith 的 *util-linux* 软件包中找到）而来的 *nfs(5)*手册页中对所有有效选项的列表进行了完整的描述。下面是你可能要用到的这些选项的一个不完整的列表：

rsize=n and wsize=n
: 这分别指出了用于 NFS 客户读写请求的数据报的大小。由于上述的 UDP 数据报大小的限制，它们目前的缺省值是 1024 字节。

timeo=n
: 这用于设置 NFS 客户将等待一个请求完成的等待时间（以十分之一秒计）。缺省值是 0.7 秒。

hard
: 明确地标记出这个卷是硬加载的（hard-mounted）。缺省值是 on。

soft
: 软加载（Soft-mount）驱动程序（与硬加载相反）。

intr
: 允许通知中断一个 NFS 调用。对于服务器没有响应时的异常中止很有用。

除了 *rsize* 和 *wsize* 以外，所有这些选项应用于当服务器临时变得不可访问时客户的相应动作。它们一起以如下的方式行事：每当一个客户想 NFS 服务器发送了一个请求，它就期望操作在一给定的时间间隔内（由 *timeout* 选项指定）完成。如果在该时间内没有收到确认，就会发生一个所谓的*次超时*（*minor timeout*），操作被重试并且超时间隔时间翻倍。在到达 60 秒钟的最大超时时间时，就发生一个*主超时*（*major timeout*）。

缺省地，一个主超时将导致客户程序在控制台上打印出一条消息并且再次重新开始，这一次，时间值再次翻倍。潜在地，这将会一直继续下去。顽固地重复一个操作直到服务器再次可用时的卷称为*硬加载*（*hard-mounted*）。相反地，每当一个主超时发生时，软加载（*soft-mounted*）的卷就为调用进程生成一个 I/O 出错信息。由于缓冲引入了后写，所以在进程下一次调用 *write(2)*函数之前，这个出错情形并没有传播到进程本身，因此一个程序就完全不能确知对一个软加载卷的写操作是否已经完成了。

硬加载还是软加载一个卷不仅只是一个感觉（习惯）问题，而且也是与从该卷上你要访问什么信息有关。例如，如果你通过 NFS 加载你的 X 程序，你当然不希望你的 X 会话仅仅因为有人同时

开启 7 个 *xv* 致使网络几乎中断而抓狂，或者由于片刻插拔下以太网插头而使得 X 会话瞬时停止。通过硬加载这个卷，你可以确信你的计算机将会等待，直到能够重新与你的 NFS 服务器建立连接。另一方面，非关键数据比如 NFS 加载的 news 分区或 FTP 文档程序也可以软加载，那么在远程机器暂时连接不上或关闭时就不会挂起你的会话过程。如果到服务器的网络连接很脆弱或者要经过一个重负荷的路由器的话，那么你或者使用 *timeo* 选项增大初始的超时值，或者硬加载这个卷，但是要允许信号中断 NFS 调用，这样你仍然可以终止任何挂起的文件访问。

通常，*mountd* 后台程序以某种方法保持哪个目录已被什么主机加载的轨迹。这个信息能够使用 *showmount* 程序显示出来，该程序也包括在 NFS 服务器软件包中。然而现在 Linux *mountd* 还不能做到这件事。

11.3 NFS 后台程序（Daemons）

如果你想为其它主机提供 NFS 服务，那么必须在你的机器上运行 *nfsd* 和 *mountd* 后台程序。作为基于 RPC 的程序，它们不是由 *inetd* 管理的，而是在引导期间启动的，并且用 portmapper 注册自身的。因此你必须确信只在 *rpc.portmap* 运行之后才开始运行它们。通常，在你的 rc.inet2 脚本中要包括下面两行：

```
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
```

NFS 后台程序为它的客户所提供的文件的属主信息通常仅包含数值的 user 和 group id。如果客户和服务器两者所用的user和group的名字与这些数值id的相关联，那么就称它们共享相同的uid/gid空间。例如，当你在你的 LAN 上对所有的主机使用 NIS 分发 *passwd* 信息就是这种情况。

然而，在某些情况下它们并不匹配。你可以不用更新客户的 uid 和 gid 以与服务器上的匹配，你可以使用 *ugidd* 映射后台程序来处理这事。使用下面描述的 *map_daemon* 选项，在客户 *ugidd* 的协助下你可以告知 *nfsd* 将服务器的 uid/gid 空间映射到客户的 uid/gid 空间上。

ugidd 是一个基于 RPC 的服务器，并且如果 *nfsd* 和 *mountd* 一样是从 *rc.inet2* 中启动的。

```
if [ -x /usr/sbin/rpc.ugidd ]; then
       /usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

11.4 *exports* 文件

上面的各个选项是用于客户的 NFS 配置的，在服务器一端有不同的选项集用于设置每一客户的行为。这些选项必须在/etc/exports 中设置。

缺省地，mountd 不允许任何人从本地主机加载目录，这是一种非常敏感的态度。为了许可一台或多台主机用 NFS 加载一个目录，这个目录必须 *exported*（输出），也即，它必须在 *exports* 文件中

被指定。一个样本文件看上去象这样：

```
# 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)
```

每一行定义了一个目录，以及允许加载该目录的主机。主机名通常是一个全资域名，但又可以含有*和?通配符，它们的作用与在 Bourne shell 中的一样。例如，**lab*.foo.com** 与 **lab01.foo.com** 匹配，也与 **laber.foo.com** 匹配。如果主机名没有给出，就象上列中的*/home/ftp* 目录，那么任何主机都允许加载这个目录。

当使用 *exports* 文件检查一台客户主机时，*mountd* 将使用 *gethostbyaddr(2)*调用查找客户的主机名，所以你必须确信没有在 *exports* 中使用别名。当不使用 DNS 时，那么返回的名字是在 *hosts* 文件中找到的与客户地址匹配的第一个主机名。

主机名字后跟一个可选的、用逗号分开的标志的列表，这些标志是用括号括住的。这些标志可以使用下面这些值：

insecure 允许从这台机器过来的非授权访问。

unix-rpc 要求对这台机器进行 UNIX 域 RPC 认证。这简单的要求请求要从一个保留的 internet 端口生成（也即，端口号必须小于 1024）。缺省地，这个选项是启用的（on）。

secure-rpc 要求对这台机器进行安全的 RCP 认证。这个选项还没有实现。参见 Sun 的有关安全 RPC 的文档。

kerberos 要求对这台机器来的访问进行 Kerberos 认证。这个选项还没有实现。参见 MIT 有关 Kerberos 认证系统的文档。

root-squash 这是一个安全特性，它通过将客户的 uid 0 请求映射到服务器的 uid 65534(-2)来拒绝指定主机上的超级用户的任何特殊访问权限。这个 uid 应该与 user **nobody** 相关联。

no_root_squash
不映射来自 uid 0 的请求。这个选项的缺省值为 on。

ro 将文件结构加载为只读。这个选项缺省值为 on。

rw 将文件结构加载为读-写。

link-relative 通过装扮所需的几个**../**来从服务器上含有连接到根的目录上将绝对符号连接（以一个斜杠开始的连接形式）转换成为相对连接。只有当主机的整个文件系统被加载时，这个选项才有意义，否则的话，某些连接将不指向任何地方，或者更糟的是，指向它们根本就没有想指的文件上。

这个选项缺省的为 on。

link-absolute 保留所有符号连接为原样（这是 Sun 提供的 NFS 服务器的常规行为）。

map_daemon 这个选项告知 NFS 服务器假设客户和服务器不共享相同的 uid/gid 空间。此时，*nfsd* 将通过查询客户的 *ugidd* 后台程序来建立一个在客户和服务器之间映射 id 的列表。

在 *nfsd* 或 *mountd* 启动时，解析 *exports* 文件所得的出错信息在 *notice* 层上被报告到 *syslogd* 的后台程序设施中。

注意，主机名字是通过逆向映射从客户的 IP 地址获取的，所以你必须正确地配置好解析器。如果你使用 BIND 并且特别注重安全性，你应该在你的 *host.conf* 文件中激活 spoof 检查。

11.5 Linux 自动加载器（Automounter）

有时候，加载用户可能要访问的所有 NFS 卷是很浪费时间的；一方面是由于要被加载的卷数量，另一方面是在启动时要占用很多时间。对此一个可选择的方案是所谓的*自动加载器*（*automounter*）。这是一个 daemon，它能自动地和透明地加载任何需要的 NFS 卷，并且在一定时间没有用到时自动卸载它们。automounter 的一个聪明之处是它可以从另外一个地方加载某个卷。例如，你可能在两到三台主机上保存有你的 X 程序和支持文件的拷贝，并且所有其它主机通过 NFS 加载它们。使用 automounter，你可以指定加载所有这三个到*/usr/X386* 上；此时 automounter 将尝试加载其中任何一个，直到有一个加载尝试成功。

Linux 常用的 automounter 称为 *amd*。它最初是由 Jan-Simon Pendry 编制的并且由 Rick Sladkey 移植到 Linux 上。当前的版本是 *amd-5.3*。

对 amd 作讨论已超出本书的范围；要想有一本好手册，请参阅源程序；其中含有一个有详细信息的文本文件。

注释

[1] 可以通过jrs@world.std.com联系到 Rick。

[2] 注意，你可以省略`-t nfs` 参数，因为 *mount* 从冒号上看出这里指定了一个 NFS 卷。

[3] 后写的问题是内核高速缓冲是用 device/inode 对索引的，因此，不能用于 NFS 加载的文件系统上。

[4] 正如 Alan Cox 向我解释的：NFS 规范要求服务器在返回应答之前将每次写入的数据刷新到磁盘上。因为，BSD 内核只能进行页大小的写入（4K），因此写入 4 块 1K 的数据将导致基于 BSD 的 NFS 服务器执行 4 次每块 4K 的写操作。

[5] 我们没有说文件系统，因为这些不是适当的文件系统。

第十二章 管理 Taylor UUCP

12.1 历史回顾

UUCP 是 AT&T 贝尔实验室的 Mike Lesk 在七十年代末期设计的，用于在公共电话线路上提供简单的拨号上网服务。由于许多想在自己的机器上有 email 和 Usenet News 的人仍然使用 modem 进行通信，所以 UUCP 仍然很流行。尽管有运行于各种类型的硬件平台和操作系统上的许多实现版本，然而它们在很高的程度上是兼容的。

然而，尽管在过去的这些年中有许多软件以各种方式已成为“标准”，还没有一个 UUCP 软件被称为 UUCP 的。自从在 1976 年第一个版本实现以来，它经历了一个稳固的演变过程。目前，存在着两个主要种类，它们在硬件支持和配置上是不同的。它们都有各式各样的实现，每种实现都有一些细微的差别。

其中一类就是所谓的“版本 2 UUCP”，它是 Mike Lesk、David A. Novitz 和 Greg Chesson 于 1977 年实现的。尽管这是一个很老的版本，但仍然被经常使用。版本 2 的近期实现提供了更新的 UUCP 种类的易用性。

第二种是于 1983 年开发的，并且通常被称为 BNU（基本连网工具）、HoneyDanBer UUCP，或简称为 HDB。这个名称产生自作者的名字，P. Honeyman、D. A. Novitz 和 B. E. Redman。HDB 考虑到了排除版本 2 UUCP 的某些不足之处。例如，增加了新的传输协议并且针对每个与之有 UUCP 通信的站点都有一个独立的目录。

目前随同 Linux 发行的 UUCP 实现是 Taylor UUCP 1.04，[1]本章即基于这个版本进行讨论。Taylor UUCP 版本 1.04 是于 1993 年 2 月发布的。除了传统的配置文件以外，Taylor UUCP 也可被编译成使用新的样式 – a.k.a.“Taylor”—配置文件。

最近发行了 1.05 版，并且不久就将融入大多数 Linux 发行版中。这些版本的不同之处主要在于你不太会使用到的特性上，所以你可以使用本书中的信息来配置 Taylor UUCP 1.05 版。

对于包含在许多 Linux 发行中的 Taylor UUCP，它通常被编译成 BNU 兼容的，或者是使用 Taylor 配置方案的，或者间而有之。由于后者更具灵活性，并且可能比经常是晦涩的 BNU 配置文件易于理解，所以下面我将介绍 Taylor 配置方案。

本章的目的不是给你一个对 UUCP 命令的命令行选项是什么和怎么使用的详尽描述，而是给你一个对如何设置一个可使用的 UUCP 站点的概要介绍。第一部分给出了有关 UUCP 是如何实现远程执行和文件传输的一个简要说明。如果对于 UUCP，你不是一个完全的新手的话，你可以跳过这一部分而直接到 UUCP 的配置文件部分，该部分解释了用于设置 UUCP 的各种文件。

然而，我们将假设你对 UUCP 套件的用户程序很熟悉。这些程序是 *uucp* 和 *uux*。有关这两个命令的介绍，请参见在线手册页。

除了通常使用的 *uux* 和 *uucp* 程序，UUCP 套件还包含了仅用于管理目的的一系列命令。它们用于监视通过你的节点的 UUCP 通信、删除老的日志文件或者汇总统计参数。这里将不对它们进行任何说明，因为它们与 UUCP 的主要任务是并行的。而且，它们有很好的文档可作参考并且很容易理解。不过，还有一类，它们是由 UUCP 实际的“工作机器”组成。它们被称为 *uucico*（这里 cico 代表 copy-in copy-out）和 *uuxqt*—用于执行远程系统发送来的作业。

12.1.1 有关 UUCP 的更多信息

对于不能在本章中找到所要信息的人，应该阅览随该软件包而来的文档。这是描述使用 Taylor 配置方案进行设置的一打 texinfo 文件。可以分别使用 *tex* 和 *makeinfo* 将 texinfo 转换成 DVI 和 GNU 信息文件。

如果你想使用 BNU 或者甚至是（令人战栗的）版本 2 配置文件的话，这里有一本很好的书，“管理 UUCP 和 Usenet”（[Oreilly89]）。我发现它非常有用。其它有关 Linux 上 UUCP 的很好的信息来源是 Vince Skahan 的 UUCP-HOWTO，它是定期地投递到 comp.os.linux.announce 上的。

当然还有一个专门讨论 UUCP 的新闻组，叫做 comp.mail.uucp。如果你有针对 Taylor UUCP 的问题，你最好在那里去问他们，而不要在 comp.os.linux 组中。

12.2 概述

12.2.1 UUCP 传输和远程执行的概要

对于理解 UUCP 至关重要的概念是*作业*（*jobs*）。用户使用 *uucp* 或 *uux* 发起的每一个传输被称作一个作业。它是由在远程系统上执行的*命令*，以及将要被在站点间传输的*文件*集构成。当然，可以省略其中一部分。

作为一个例子，假设你在你的主机上发出了下面的命令，该命令使得 UUCP 将文件 *netguide.ps* 拷贝到主机 **pablo** 上，并且使得它执行 *lpr* 命令来打印这个文件。

```
$ uux –r pablo!lpr !netguide.ps
```

UUCP 通常不会立刻调用远程系统来执行一个作业（不过你可以使用 kermit 来做到）。而是临时地将该作业描述存储起来。这称为*假脱机操作（spooling）*。作业所存放的目录树因此也就称为*假脱机目录（spool directory）*并且通常位于*/var/spool/uucp* 中。在我们的例子中，该作业描述含有将被执行的远程命令（*lpr*）的有关信息、要求进行该操作的命令以及其它一些项目。除了这个作业描述，UUCP 也需要存储输入的文件 *netguide.ps*。

假脱机文件所在的确切位置和命名方法是可以不同的，这依赖于一些编译时的选项。HDB 兼容的 UUCP 通常将假脱机文件存储于命名为*/var/spool/uucp/**site*** 的目录中，这里 ***site*** 是远端站点的名称。当以 Taylor 配置方式编译时，UUCP 将针对不同类型的假脱机文件在指定站点的假脱机目录下再创建子目录。

在规定的时间间隔，UUCP 将向远程系统拨号。当与远程系统的连接建立后，UUCP 就会传输描述作业的文件，加上所有的输入文件。输入的作业不会立刻被执行的，而要到连接结束关闭之后。这是用 *uuxqt* 来执行的，如果有指定到其它站点的作业，它也处理这些作业的转发工作。

为了区分重要的和不很重要的作业，UUCP 给每个作业指定了一级别（*grade*）。这是一单个字母，范围从 0 到 9、A 到 Z 以及 a 到 z，级别从大到小。Mail 通常以级别 B 或 C 进行假脱机操作，而 news 则以级别 N 进行假脱机操作。级别越高的作业传输的越早。级别可以在调用 *uucp* 或 *uux* 时用-g 标志来指定。

你也可以在一定时间内禁止低于某级别的作业的传输。这称为在对话期间所允许的*最大假脱机*

级别（maximum spool grade），缺省值是 z。这里请注意术语上的含糊不清：一个文件将被传输当且仅当它的级别等于或高于最大假脱机级别。

12.2.2 *uucico* 的内部工作机制

要理解为什么 *uucico* 需要知道某些事情，这里给出了它实际上是如何连接至远程系统的一个快速描述。

当你在命令行上执行 *uucico* –s system 时，它首先必须进行物理连接。所进行的操作起决于所打开的连接类型 – 例如，当使用电话线时，它必须找到一个 modem，并且进行拨号。而在 TCP 上，它就必须调用 *gethostbyname(3)*将名字转换为一个网络地址、找出要打开那一个端口，并且将该地址绑定于相应的套接字（socket）上。

在这个连接被建立起来后，接下来必须通过一个认证过程。这常常是由远程系统询问一个登录名字以及一个可能的密码组成。这通常被称为*登录对话（login chat）*。认证过程或者是通过通常的 *getty/login* 套件执行的，或者是由 *uucico* 自身在 TCP 套接字上完成的。如果认证成功的话，远端系统就会启动 *uucico*。初始化连接的本地 uucico 拷贝被视作主端（*master*），远端的则作为从端（*slave*）。

接下来是握手阶段（*handshake phase*）；主端现在发出自己的主机名，加上几个标志，从端检查这个主机名的登录许可，发送和接收文件等等。这些标志用于描述（以及在其它一些事情中）被传输的假脱机文件的最大级别。如果使能的话，这里将会进行一个对话计数，或调用序列号的检查。使用这个特性，两端站点就维持有一个成功连接的计数，可用于进行比较。如果它们不匹配的话，握手过程就失败了。这对于保护你免受冒充者是很有用的。

最后，两个 *uucico* 试着达成一个共同的传输协议。这个协议指导数据被传输的方法、检查一致性并且在出错时进行重传操作。针对所支持的不同的连接类型需要有不同的协议。例如，电话线路要求有一个对于出错保守的“安全”协议，而 TCP 传输天生就是可靠的因此可以使用一个更为有效的无须附加出错检查的协议。

在握手阶段完成以后，就开始进行实际的传输阶段。传输两端开启所选的协议驱动程序。驱动程序一般还要进行与该协议相关的初始化过程。

首先，主端将发送假脱机级别足够高的排于队列中的所有文件到远程系统中。当它完成传输后就会通知从端，此时从端就可以挂断了。现在从端可以或者同意挂断，或者将对话控制权接过来。这是一个规则的变化：现在远程系统变成了主端，而本地则变成了从端。新的主端现在发送它的文件。当完成后，两个 *uucico* 互相交换传输消息，并关闭连接。

我们将不再进行更为详细的描述：请参阅代码或任何针对于此的有关 UUCP 的好书。在网上也还流传着一篇很古老的文章，是由 David A. Novitz 写的，它给出了 UUCP 协议的详细描述。Taylor UUCP FAQ 也讨论了 UUCP 实现方法的某些细节。它被定期地投递到 comp.mail.uucp 上。

12.2.3 *uucico* 命令行选项

本节描述 *uucico* 的一些最重要的命令行选项。有关完整的命令行选项列表，请参阅 uucico(1) 手册页。

`-s` *`system`*　　呼叫指定的系统（*system*）除非受到呼叫时间的限制。

`-S system` 无条件地呼叫指定的（*system*）。

`-r1` 以主端（master）模式启动 *uucico*。这是在给出`-s` 或`-S` 时的缺省值。如果只有该选项，该`-r1` 选项将致使 *uucico* 试尝呼叫 *sys* 中的所有系统（systems），除非受到呼叫限制或重试次数限制。

`-r0` 以从端（slave）模式启动 *uucico*。这是在没有`-s` 或`-S` 给出时的缺省值。在从端模式中，或者是标准的输入/输出被假设连接至一个串行端口，或者是使用由`-p` 指定的 TCP 端口。

`-x type,-X type` 开启指定类型的调试。可以用逗号分开的一个列表指定几种类型。以下类型是有效的：*abnormal，chat，handshake，uucp-proto，proto，port，config，spooldir，execute，incoming，outgoing*。为了与其它 UUCP 实现方式的兼容，可以用一个数字代替，该数字开启上面列表中的头 *n* 项。

调试信息将被记录在*/var/spool/uucp* 文件 *Debug* 中。

12.3 UUCP 的配置文件

与简单的文件传输程序相比，UUCP 被设计成能够自动地处理所有的传输操作。一旦被正确地设置好，就无须管理员每日进行干预了。操作所需要的信息被保存在*/usr/lib/uucp* 目录下的几个配置文件中。这些文件大多数用于拨出操作中。

12.3.1 Taylor UUCP 简介

要是说 UUCP 的配置很难，这只是一种保守的说法。实际上它是一个错综复杂的问题，并且有时候配置文件简明的格式并没有使得问题变得更容易些。（尽管与 HDB 或版本 2 中的老式格式相比较，Taylor 格式几乎是很容易阅读的）。

为了给你一个这些文件是如何相互作用的感觉，我们将向你介绍一个非常重要的文件，并对这些文件的样本进行观察。现在我们将不解释各种细节问题；更精确的描述将在下面各个独立小节中给出。如果你想在你的机器上设置 UUCP，你最好是从某些样本文件开始，并逐渐地对样本文件加以调整。你可以或者采用下面示出的，或者采用那些包含在你的 Linux 发行版本中的样本文件。

本节描述的所有文件都存储在*/usr/lib/uucp* 或其中的一个子目录中。有些 Linux 发行版含有 UUCP 执行文件，它们对 HDB 和 Taylor 配置都支持，并且对每种配置文件集使用不同的子目录。在*/usr/lib/uucp* 中通常有一个 README 文件。

为了让 UUCP 能正常地工作，这些文件必须为 **uucp** 用户所有。其中某些文件含有密码和电话号码，因此应该有 600 的许可权限。[2]

最主要的 UUCP 配置文件是*/usr/lib/uucp/config*，并用于设置通常的参数。其中最重要的（并且到现在为止，也是仅有的），是你的主机的 UUCP 名字。在虚拟酿酒厂，他们使用 **vstout** 作为他们的 UUCP 网关：

```
# /usr/lib/uucp/config – UUCP main configuration file
hostname      vstout
```

另一个重要的配置文件是 *sys* 文件。它含有你将连接站点的所有系统方面的信息。这包括站点名字以及连接本身的信息，比如当使用 modem 连接时的电话号码。一个以 modem 连接的称为 **pablo** 站点的典型内容将是

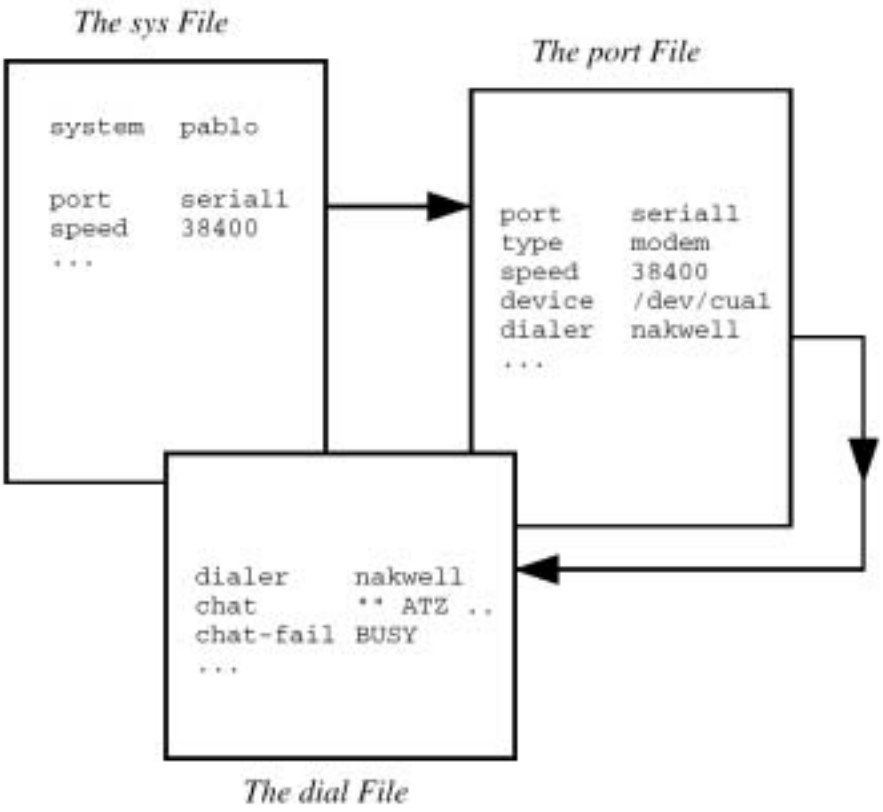

图 12.1: Taylor UUCP 配置文件之间的相互关系。

```
# /usr/lib/uucp/sys – name UUCP neighbors
# system: pablo
system    pablo
time         Any
phone     123-456
port         serial1
speed     38400
chat         ogin: vstout ssword: lorca
```

port 指定了所使用的端口，而 *time* 指出了它将被呼叫的时刻。Chat 描述了登录对话脚本-必须相互交换以允许 *uucico* 登录进 **pablo** 的字符串序列。我们将稍后讨论登录脚本。*port* 命令并不指定一个设备相关的文件比如*/dev/cua1*，而是指定了 *port* 文件中的一个入口。你可以随心所欲地指定分配这些名字只要这些名字在 *port* 文件中有有效的引用。

port 文件掌握着与连接相关的信息。对于 modem 连接，它描述了所使用的设备相关文件、所支持的速度范围以及连接至端口的拨号设备的类型。下面的入口条目描述/dev/cua1（即已知的 COM2），

一个 NakWell modem 连接到它上面，该 modem 可以运行在最高 38400bps 的速度上。入口条目的命名方法是选择与 *sys* 文件中端口名匹配的名字。

```
# /usr/lib/uucp/port – UUCP ports
# /dev/cua1 (COM2)
port            serial1
type            modem
device       /dev/cua1
speed        38400
dialer       nakwell
```

属于拨号器（dialer）本身的信息则被保存在另一个文件中，称为-你猜得到的：*dial*。对于每种拨号器类型，它基本上含有用于发出拨号到一个远程站点的命令序列、和给出的电话号码。再一次，这是作为一个对话脚本给出的。例如，对于上面的 NakWell 的入口条目看上去可以象这样：

```
# /usr/lib/uucp/dial – per-dialer information
# NakWell modems
dialer    nakwell
chat         “” ATZ OK ATDT\T CONNECT
```

以 chat 开头的一行指出了 modem 对话，它是发送到 modem 和从 modem 接收的初始化 modem 并使它拨出所期望号码的命令串。”\T”序列将被 *uucico* 替换成电话号码。

为了给你一个 *uucico* 是如何处理这些配置文件的大概思路，假设你在命令行上发出命令

```
# uucico –s pablo
```

uucico 所做的第一件事是在 *sys* 文件中查找 **pablo**。从 sys 文件的 **pablo** 入口条目中它看到它应该使用 *serial1* 端口来建立连接。*port* 文件告知它这是一个 modem 端口，并且有一个 NakWell modem 连在该端口上。

uucico 现在在 *dial* 文件中查找描述 NakWell modem 的入口条目，并且找到一个，并打开串行端口*/dev/cua1* 并执行拨号器对话。也即，它发送出“ATZ”，等待响应“OK”，等等。当遇到字符串“\T”时，就用从 *sys* 文件中获得的电话号码（123-456）取代之。

在 modem 返回 `CONNECT` 以后，连接就被建立好了，并且 modem 对话也就结束了。*uucico* 现在返回到 sys 文件并执行登录对话。在我们的例子中，它将等待提示“`login:`”，然后送出它的用户名（`neruda`），等待“`password:`”提示，并送出它的密码，“`lorca`”。

在完成了权限认证以后，远端即会启动它自己的 uucico。然后，这两个 uucico 将进入前节所述的握手阶段。

配置文件之间的相互关系也在图 12.1 中示出了。

12.3.2 UUCP 需要知道些什么

在你开始编写 UUCP 配置文件之前，你必须收集它需要知道的一些信息。

首先，你必须知道你的 modem 连接在哪个串行端口上。通常，（DOS）端口 COM1 至 COM4

映射到设备文件*/dev/cua0* 至*/dev/cua3* 上。对于大多数发行版，比如象 Slackware，创建了一个链 */dev/modem* 作为到适当 cua*设备文件的一个链，并配置 *kermit*、*seyon* 等等，以使用这个通用文件。在我们这个情况下，你也应该在你的 UUCP 配置中使用*/dev/modem*。

这样做的原因是所有拨号程序在串行端口被占用时使用所谓的 *lock* 文件来作出通知。这些锁定文件的名称是字符串 *LCK..*和设备文件名的串联，例如 *LCK..cua1*。如果程序对于同样的设备使用不同的文件名，它们将不能识别出各自的锁定文件。结果，当它们同时启动时，将毁坏对方的进程。在你使用 crontab 确定你的 UUCP 呼叫时间表时，这并非不常发生的事。

关于设置你的串行端口的详细信息，请参见第四章。

下一步，你必须找出在什么速度上你的 modem 将与 Linux 通信。你必须将它设置成你所期望获得的最大有效传输速率上。有效的传输速率可能比你的 modem 的原始物理传输速率高得多。例如，许多 modem 以 2400bps(每秒比特数)发送和接收数据。如果使用了压缩协议如 V.42bis，实际的传输速率可能爬升至 9600bps。

当然，如果 UUCP 要做任何事的话，你将需要一个被呼叫的系统的电话号码。同样，对与远程系统，你还需要一个有效的登录 id 和一个可能的密码。[3]

你也必须明确地知道如何登录进系统。例如，在登录提示出现之前你是否需要按下 BREAK 键？它是显示 `login:`还是 `user:`？这对于编制对话脚本（*chat script*）是必须的，这个对话脚本告知 *uucico* 如何进行登录。如果你不知道，或者如果常用的对话脚本失败了，就尝试使用象 *kermit* 或 *minicom* 的终端程序，并明确地写下来你必须做些什么。

12.3.3 站点命名

对于基于 TCP/IP 的网络，使用 UUCP 连网时，你的主机必须有一个名字。要是你只是想简单地使用 UUCP 来进行你直接拨号的站点间或在在本地网上文件的传输工作，那么这个名字就不需要符合任何标准。[4]

然而，如果你将 UUCP 用于 mail 或 news 连接，你应该考虑把你的名字注册在 UUCP 映射计划项目中。UUCP 映射计划将在第 13 章加以描述。即使你只是在一个域中，你也应该考虑为你的站点获取一个官方的 UUCP 名字。

人们常常选择他们的 UUCP 名字与他们的全资域名相匹配。假如你的站点的域名地址是 **swim.twobirds.com**，那么你的 UUCP 主机名就应该是 **swim**。考虑那些互相熟知的基于第一个名字的 UUCP 站点。当然，你也可以使用一个完全与你的全资域名无关的 UUCP 名字。

然而，请确信不要在邮件地址中使用不合格的站点名称除非你已经将其注册为你的正式 UUCP 名字。在最好的情况下，邮递到没有注册的 UUCP 主机的所有信息将只是悄无声息地消失掉了。如果你使用了一个早已属于其它站点的名字，那么邮件将会路由到那个站点去，并会给那个站点的邮件管理者带来无穷无尽的烦恼。

默认地，UUCP 站点使用由 *hostname* 设置的名字作为该站点的 UUCP 名字。这个名字通常是在 */etc/rc.local* 脚本中设置的。如果你的 UUCP 名字与你的主机名的设置不同的话，你就必须在 *config* 文件中使用 *hostname* 选项来告知 *uucico* 你的 UUCP 的名字。这将在下面讲解。

12.3.4 Taylor 配置文件

现在我们回过来讨论配置文件。Taylor UUCP 从下列文件中获取信息：

`config` 这是个主要的配置文件。你可以在这里定义你的站点的 UUCP 名字。

`sys` 这个文件描述了你所知道的所有站点。对于每个站点，它指定了站点的名字、呼叫时间、拨号号码（如果有的话）、所使用的设备类型以及如何登录。

`port` 含有描述各个存在端口的入口条目，以及线路所支持的速率和所用的拨号器。

`dial` 描述用于建立电话线路连接的拨号器。

`call` 含有呼叫一个系统时所用的登录名和密码。很少用到。

`passwd` 含有在登录时系统可能用到的登录名和密码。仅当 uucico 自己进行密码检查时才用到这个文件。

Taylor 配置文件通常由含有关键字-值对的行组成。一个‘#’符号指定本行直到行尾都为注释。要使用‘#’符号本身，你就要和反斜杠一起使用它。

你可以使用很多的选项来调节这些配置文件。这里我们不能讨论所有这些参数，而将只是讨论几个很重要的参数。利用它们，你就可以配置一个基于 modem 的 UUCP 连接。其它的小节将描述当你想在 TCP/IP 或直接串行线上使用 UUCP 时所须作的必要修正。完整的参考资料已随同 Taylor UUCP 原代码以 Texinfo 文档给出。

当你认为你已经完全配置好你的 UUCP 系统以后，你可以使用 *uuchk* 工具（位于*/usr/lib/uucp*）来检查你的配置情况。*uuchk* 读入你的配置文件，并打印出用于每个系统的配置值的详细报告。

12.3.5 常用配置选项 – *config* 文件

除了你的 UUCP 主机名以外，这个文件通常并不包含其它信息。默认地，UUCP 将使用你用 hostname 命令设置的名称，但是明确地设置 UUCP 名字将是一个好注意。下面示出了一个例子文件：

```
# /usr/lib/uucp/config – UUCP main configuration file
hostname        vstout
```

当然在这里也可以设置许多各种各样的参数，比如假脱机的目录、或匿名 UUCP 访问的权限。后者将在后面小节中讨论。

12.3.6 如何告知 UUCP 有关其它系统的信息 – *sys* 文件

sys 文件描述了你的机器要了解的远端系统的信息。一个条目由 *system* 关键字引入；随后的几行直到下一个 *system* 指令之间的信息详细描述了有关那个站点的参数。通常，一个系统条目将定义电话号码和登录对话等参数。

在头一个 *system* 行以前的各个参数设置了用于所有系统的缺省值。一般地，你要在缺省部分中

设置协议参数等的信息。

下面，较详细地描述了几个很重要的字段。

系统名称（System Name）

System 命令指定了远程系统。你必须指定远程系统的正确名字，而不是你虚构的别名，因为在你登录时 *uucico* 将拿它与远程系统所给出的作检查比较。[6]

每个系统名字只能出显一次。如果对同一个系统你想使用几种配置（比如 *uucico* 应该尝试的几个不同的电话号码），你可以指定 *alternates*。Alternates 将在下面讨论。

电话号码（Telephone Number）

如果远程系统是通过电话线路联系的，*phone* 字段将指定 modem 将拨号的号码。它可以含有由 *uucico* 的拨号过程解释的标记。一个等于符号意思是指等待第二个拨号音，一个破折号产生一秒的暂停时间。比如，有些安装的电话当你在前导码与电话号码之间不暂停时就会止住。

[对此我并不知道适当的英语术语，就象一个公司内部私有安装的电话，当你要接外线时你首先必须拨一个 0 或 9。]

任何内嵌的字符串可以用于隐藏与站点相关的信息，比如区位号。任何象这样的字符串使用 *dialcode* 文件都被转换成拨号代码。假如你有以下的 *dialcode* 文件：

```
# /usr/lib/uucp/dialcode – dialcode translation
Bogoham          024881
Coxton        035119
```

利用这些转换，你可以在 *sys* 文件中使用 *Bogoham7732* 这样的电话号码，这使得号码变得清晰易读些。

端口与速率（Port and Speed）

port 和 *speed* 选项用作选择用于呼叫远程系统的设备以及设备应该设置的最高速率。[7]一个 *system* 条目可以单独地使用这两个选项，或同时使用它们。当在 *port* 文件中查找适当的设备时，只有那些端口名以及/或者速度范围匹配的端口被选中。

通常，使用 *speed* 选项就足够了。如果在 *port* 中只定义了一个串行设备的话，*uucico* 无论如何将总能选择到一个正确的，所以你只需给它一个所期望的速度即可。如果你的系统上连接了几个 modem 的话，你通常仍然无须指定一个特定的端口，因为在 *uucico* 发现有几个匹配时，它回逐个尝试每个设备直到找到一个未用的设备为止。

登录对话（The Login Chat）

上面，我们已经遇到过登录对话脚本，它告知 *uucico* 如何登录进远程系统。它由本地 *uucico* 进程指定的期望字符串和发送字符串的一个标记列表组成。目的是为了让 *uucico* 等待远程机器发送过来一个登录提示，然后返回登录名称，再等待远程系统发送密码提示，并发送密码。所期望的和发

送的字符串是交替给出的。*uucico* 自动地将回车符（carriage return character `\r`）附加到任何发送的字符串上。这样，一个简单的对话脚本将象这样

```
ogin:  vstout ssword:  catch22
```

你会注意到所期望的字段并没有包含完整的提示。这是为了确信即使远程系统广播了 `Login:` 而不是 `login:`时也能登录成功。

uucico 同样也允许某些条件执行，例如，在发出提示之前远程机器的 *getty* 需要被复位的情况。对于此，你可以将一个子对话附加到一期望字符串上，用一破折号补偿。只有当主要期望失败时，例如超时，子对话才会被执行。使用这个特性的一种方法是在远程站点没有显示一登录提示时发送一 BREAK。下面的例子给出了一个多用途的对话脚本，对于你必须在登录显示出来之前按回车的情况也同样能用。`""`告诉 UUCP 不要等待任何信息而立刻继续进行下一字符串发送。

```
"" \n\r\d\r\n\c ogin:-BREAK-ogin: vstout ssword: catch22
```

在对话脚本中会存在几个特殊的字符串和逃逸字符。下面是在期望字符串中合法的字符的不完整列表：

`""`　　空字符串。它告知 uucico 不要等待任何事情，而立刻进行下个字符串的发送处理。

`\t`　　Tab 字符。

`\r`　　回车（Carriage return）字符。

`\s`　　空格字符。你需要它在对话串中加入空格。

`\n`　　换行符。

`\\`　　反斜杠字符。

对于发送的字符串，除了上面的字符以外，下面这些字符和字符串也是合法的：

EOT　　传输结束字符（^D）（End of transmission character）。

BREAK　　中断字符。

`\c`　　在字符串尾压缩回车的发送。

`\d`　　延迟 1 秒发送。

`\E`　　允许响应（回送 echo）检查。在 uucico 能继续进行对话之前，它要等待写出的所有信息的来自设备的响应被读取到。当用于 modem 对话时，这是非常有用的（我们将在下面遇到）。缺省地，响应检查是关闭的（off）。

\e 禁止响应检查。

\K 同 BREAK。

\p 暂停几分之一秒。

备用（Alternates）

有时，非常希望对于单个系统有多个条目，比如如果系统能通过不同的 modem 线路到达。对于 Taylor UUCP，你可以通过定义一个所谓的 *alternate* 来做到。

为了对 **pablo** 使用两个电话号码，你要用以下方法修改它在 *sys* 中的条目：

```
system    pablo
phone     123-456
… entries as above …
alternate
phone     123-455
```

当呼叫 **pablo** 时，*uucico* 现在将首先拨出号码 123-456，如果失败了，就会试用备用的。备用条目维持与主要系统条目所有的设置，仅是电话号码不同而已。

限定呼叫时间（Restricting Call Times）

Taylor UUCP 提供了许多方法允许你限制对远程系统呼叫的时间。这样做的原因或者是因为远端主机只在上班时间才提供此类服务，或者简单地只是为了避免高呼叫时间段的费用。注意，通过给 *uucico* 选项-S 或-f，总是可以覆盖呼叫时间限制。

默认地，Taylor UUCP 不允许任意时间的连接，所以你必须在 *sys* 文件中使用某些时间的规格说明。如果你不在乎呼叫时间限制的话，那么你可以在你的 *sys* 文件中把值 *Any* 指定给时间 *time* 选项。

限定呼叫时间的最简单的办法是利用 *time* 条目，它是由一个日子和时间子字段组成的字符串指定的。日子可以是任何 *Mo*、*Tu*、*We*、*Th*、*Fr*、*Sa*、*Su* 的组合，或者是 *Any*、*Never*、或者平日 *Wk*。时间由两个 24 小时时钟值组成，由短划线分隔。它们指定了呼叫的时间范围。这些标记的组合之间没有空格。任何数量的日子和时间的说明可以用逗号组合在一起。例如，

```
time              MoWe0300-0730,Fr1805-2000
```

允许在星期一以及星期三从早上 3 点到 7 点 30 分，以及星期五在 18 点 05 分到 20 点之间进行呼叫活动。当时间字段跨越午夜时，比如 Mo1830-0600，它实际上表示星期一，在午夜到早上 6 点之间，以及在下午 6 点 30 分到午夜之间。

特殊的时间字符串 *Any* 和 *Never* 意思即是它们的本意：分别是指在任何时候都可以进行呼叫和在任何时候都不可以呼叫。

time 命令有一个可选的第二个参数，用于描述以分钟计的重试时间。当建立一个连接的企图失败时，*uucico* 在一定时间间隔内将不允许另一次拨号到远程主机的尝试。缺省地，*uucico* 使用一种指数后退方案，这是指重复失败次数越多，那么重试间隔的时间就越长。例如，当你指定 5 分钟的重试时间时，*uucico* 在距上次失败 5 分钟之内将拒绝呼叫远程系统。

timegrade 命令允许你往时间表上附加一个最大假脱机级别。例如，假设在 *system* 条目中你有以下的 *timegrade* 命令：

```
timegrade       N Wk1900-0700,SaSu
timegrade       C Any
```

这允许只要呼叫一被建立，那么假脱机级别 C 或更高的作业（通常，mail 是以级别 B 或 C 排队的）即可立刻传输，而 news（通常以级别 N 排于队列中）将只能在夜晚和周末传输。

和 *time* 一样，*timegrade* 命令将一个以分钟计的重试间隔时间作为第三个可选的参数。

然而，有关假脱机级别的一个告戒是：首先，*timegrade* 选项只应用于你的系统的发送；而远程系统仍然可以随心所欲地传输任何东西。你可以使用 *call-timegrade* 选项明确地请求远程系统只发送那些高于假脱机级别的作业；但并不能保证它一定会遵循这个请求。[8]

同样地，当远程系统呼叫进来时，并不会检查 *timegrade* 字段，所以任何排队等待这个远程系统的作业都将被发送出去。然而，远程系统可以明确地请求你的 uucico 来限制自身只发送一定假脱机级别的作业。

12.3.7 有些什么设备 –*port* 文件

port 文件告知 *uucico* 现有的端口。这些可能是 modem 端口，但其它类型的端口比如直接串行线路连接和 TCP 套接字也同样得到支持。

象 *sys* 文件一样，*port* 文件由以关键字 *port* 开始后接端口名的各个条目构成。这个端口名可以被用于 *sys* 文件中的 *port* 语句中。这个名字无须是唯一的；如果几个端口有同样的名字，*uucico* 将按顺序地试用每一个端口直到它找到一个空闲的端口。

port 命令后必须紧跟描述端口类型的 *type* 语句。有效的类型有 *modem*、直接连接的 *direct*、和用 TCP 套接字的 *tcp*。如果 *port* 命令不存在，那么端口类型的缺省值是 modem。

在这一小节中，我们将仅讨论 modem 端口；TCP 端口和直接线路连接将在后面的小节中讨论。

对于 modem 和直接端口，你必须使用 device 指令指定用于呼出的设备。通常，这是*/dev* 目录中一个设备文件的名字，象*/dev/cua1*。[9]

对于 modem 设备的情况，端口条目也确定了那种类型的 modem 连接在端口上。不同类型的 modem 要求不同的配置。即使是那些声称 Hayes 兼容的 modem 也不一定是互相真正兼容的。因此，你必须告知 *uucico* 如何初始化 modem 以及如何让它拨出所希望的号码来。Taylor UUCP 将所有拨号器的描述保存在一个称为 *dial* 的文件中。为了要使用其中任何一个，你必须使用 *dialer* 命令指定拨号器的名字。

有时候，你会想以不同的方式使用一个 modem，这依赖于你呼叫的系统。例如，当一个高速 modem 试图以 14400bps 来连接时，某些老式的 modem 就不能理解；它们只是简单地中断线路而不是协商一个连接速度，比如说，9600bps。当你知道站点 **drop** 使用这样一个不灵光的 modem 时，那么在呼叫它们时，你就必须以不同的方式设置你的 modem。针对于此，在 *port* 文件中你需要一个指定一个不同的拨号器的额外的端口条目。现在你可以给这个新端口一个不同的名字，比如说 *serial1-slow*，并在 *sys* 文件的 **drop** 系统条目中用 port 指令。

一个更好的办法是通过他们所支持的速度来区分它们。例如，上面情况的两个端口入口条目看上去象这样：

```
# NakWell modem; connect at high speed
```

```
port            serial1          # port name
type            modem         # modem port
device       /dev/cua1        # this is COM2
speed        38400         # supported speed
dialer       nakwell          # normal dialer

# NakWell modem; connect at low speed
port            serial1          # port name
type            modem         # modem port
device       /dev/cua1        # this is COM2
speed        9600             # supported speed
dialer       nakwell-slow     # don't attempt fast connect
```

对于站点 drop 的系统入口条目将给出 serial1 作为端口名，但只请求以 9600bps 使用它。此时，uucico 将会自动地使用第二个端口入口条目。在系统条目中有 38400bps 速度的所有站点都将被使用第一个端口条目呼叫。

12.3.8 如何拨号 – *dial* 文件

dial 文件描述了各种拨号装置（拨号器）的使用方法。传统上，UUCP 只谈及拨号器而非 modem，因为在早期，只拥有一个（昂贵的）的拨号器来服务于一组 modem 是很现实的事。如今，大多数 modem 都有内置的拨号支持，因此这种区别有一些使人混淆。

然而，不同的拨号器或 modem 需要不同的配置。你可以在 *dial* 文件中来描述它们的每一种。*dial* 中的每一个条目都以给出拨号器名字的 *dialer* 命令开始。

除此之外最重要的条目是由 *chat* 命令指定的 modem 对话条目。与登录对话类似，它是由一系列 uucico 发送到拨号器和它所期望的返回响应字符串组成。它通常用于将 modem 复位到某种已知状态并且进行拨号。下面的拨号器条目样本示出了 Hayes 兼容 modem 的一个典型 modem 对话：

```
# NakWell modem; connect at high speed
dialer       nakwell      # dialer name
chat            "" ATZ OK\r ATH1EOQO OK\r ATDT\T CONNECT
chat-fail    BUSY
chat-fail    ERROR
chat-fail    NO\sCARRIER
dtr-toggle   true
```

modem 对话过程以空字符串“”开始。接下来 uucico 将送出第一个命令（ATZ）。ATZ 是 Hayes 命令用于复位 modem。然后等待 modem 送回 OK，接着送出下一个命令来关闭本地回显，等等。当 modem 再次返回 OK 以后，*uucico* 发出拨号命令（ATDT）。该字符串中的逃逸字符序列\T 将被从 sys 文件中的系统条目内的电话号码替换掉。此后，*uucico* 等待 modem 返回字符串 CONNECT，这个字符串表示与远程的 modem 已经成功地建立了连接。

常常，modem 与远程系统的连接会失败，例如，远程系统正在与其它人通话并且线路忙的话。在这种情况下，modem 将返回某些出错信息指出失败的原因。Modem 对话是没有能力来识别出这

种消息的；*uucico* 将继续等待期望的字符串的到来直到超时。因此 UUCP 的日志文件将只显示出一模糊的“对话脚本超时（timed out in chat script）”信息而非真正的原因。

然而，Taylor UUCP 允许你使用上面的 *chat-fail* 命令告知 *uucico* 有关此类出错消息。在执行 modem 对话时，当 uucico 检测出一个对话出错（chat-fail）字符串，它就会放弃这次呼叫，并且将错误消息记录在 UUCP 的日志文件中。

上面样本例子中所示的最后一个命令告知 UUCP 在开始 modem 对话之前转换 DTR 线的信号。大多数 modem 能够配置成当检测到 DTR 线路信号转换时挂断线路，并进入命令模式。[10]

12.3.9 TCP 上使用 UUCP

初听起来有些荒谬，然而使用 UUCP 在 TCP 上传输数据并不是一个坏主意，尤其是当传输象 Usenet news 这样的大量数据时。在基于 TCP 的链接上，news 通常使用 NNTP 协议来进行交换的，请求和发送文章是分别进行的，没有压缩也没有进行任何的优化。尽管这适用于有着几个并行的喂信功能的大站点，这种技术对于使用 ISDN 等慢速连接来接收他们的 news 的小型站点并不合适。这些站点通常希望能结合 TCP 大批量发送 news 的优点，能够将数据压缩传输而只有非常小的过载。批量传输这些数据的一个标准方法就是在 TCP 上使用 UUCP。

在 *sys* 中，你要以下面的方法来指定一个系统通过 TCP 来呼叫：

```
system        gmu
address           news.groucho.edu
time              Any
port              tcp-conn
chat              ogin; vstout word; clouseau
```

address 命令给出了主机的 IP 地址，或者是它的全资域名。相应的 *port* 条目将是：

```
port              tcp-conn
type              tcp
service           540
```

这个条目表示当一个 *sys* 条目参考使用 tcp-conn 时将使用一个 TCP 连接，并且 *uucico* 将试图连接到远端主机的 TCP 网络端口 540 上。这个端口是 UUCP 服务的默认端口号。除了端口号，你也可以给 *service* 命令一个符号端口名。而与这个名字对应的端口号将在*/etc/services* 中找到。UUCP 服务的通用名称是 *uucpd*。

12.3.10 使用直接连接

假设你使用线缆把你的系统 **vstout** 和 **tiny** 直接连接起来。与使用 modem 的情况非常相似，你必须在 sys 文件中写一个系统条目。确定串行端口 *tiny* 的 *port* 命令被连通起来。

```
system        tiny
```

```
time            Any
port            direct1
speed        38400
chat            ogin; cathcart word; catch22
```

在 *port* 文件中，你必须对直接连接描述这个串行端口。不需要 *dialer* 条目，因为无须拨号。

```
port            direct1
type            direct
speed        38400
```

12.4 UUCP 能做与不能做的 – 调整权限

12.4.1 命令执行

UUCP 的任务是将文件从一个系统拷贝至另一个系统，并且请求在远程主机上执行适当的命令。当然，你作为一个管理员会想要控制给予其它系统的权限 – 允许他们能够执行你的系统上的任何命令肯定不是一个好主意。

默认地，Taylor UUCP 所允许其他系统在你的机器上执行的仅有的命令是 *rmail* 和 *rnews*，它们通常是用于使用 UUCP 交换 email 和 Usenet news。用于 *uuxqt* 的默认搜索路径是一个编译时的选项，通常包含有*/bin*、*/usr/bin* 和*/usr/local/bin*。如果想要改变一个特定系统能执行的命令集的话，你可以在 *sys* 文件中使用 *commands* 关键字。类似地，可以用 *command-path* 语句改变搜索路径。例如，除了 *rmail* 和 *rnews* 命令，你可能还想允许 **pablo** 系统能执行 *rsmtp* 命令：[11]

```
system       pablo
…
commands        rmail rnews rsmtp
```

12.4.2 文件传输

Taylor UUCP 也允许你非常详细地微调文件的传输。在一个极端情况下，你可以禁止向/从某一特定系统的传输。只需设置 *request* 为 *no*，远程系统就不能从你的系统中汲取文件也不能向你的系统发送任何文件。同样地，通过把 *transfer* 设置成 *no*，你可以禁止你的用户向一个系统传输文件或从一个系统中传进文件。默认地，本地的和远程的用户是允许上载和下载文件的。

另外，你可以配置进行文件拷贝的目录。通常，你会想要限制远程系统的访问到某一个目录结构中，但仍然允许你的用户从他们的主目录中发送文件。常常，远程用户只被允许从公共的 UUCP 目录*/var/spool/uucppublic* 中接收文件。这是放置公共文件的传统地方；非常象 Internet 上的 FTP 服务。它通常用~字符指示。

因此，Taylor UUCP 提供了四种不同的命令用来配置发送和接收文件的目录。它们是 *local-send*，

它指定了用户可以要求 UUCP 从中发送文件的目录列表；*local-receive*，它给出了用户可以请求 UUCP 接收文件的目录列表；*remote-send* 和 *remote-receive*，它们针对外部系统做类似的工作。考虑下面的例子：

```
system        pablo
…
local-send       /home ~
local-receive       /home ~/receive
remote-send         ~ !~/incoming !~/receive
remote-receive         ~/incoming
```

local-send 命令允许你的主机上的用户将*/home* 和公共 UUCP 目录中的任何文件发送到 **pablo**。*local-receive* 命令允许用户将接收到的文件放在 *uucppublic* 中的完全可写的 *receive* 目录中或*/home* 下任何完全可写的目录中。*remote-send 命*令允许 **pablo** 从*/var/spool/uucppublic* 中获取文件，而除了 *incoming* 和 *receive* 目录下的文件。这是通过在相应的目录名前放置感叹号来指出的。末了，最后一行允许 **pablo** 往 *incoming* 中上载任何文件。

使用 UUCP 传输文件的严重问题之一是它仅允许将接收到的文件放到完全可写的目录中。这可能会引诱某些用户对其他用户设置陷阱等等。然而，并没有任何方法来阻止这个问题，除非完全禁止 UUCP 的文件传输。

12.4.3 转发（Forwarding）

UUCP 提供了让其它系统依你的的要求执行文件传输的机制。例如，这允许你使得 **seci** 为你从 **uchile** 获取一个文件，并将它发送到你的系统。下面的命令将完成这个任务：

```
$ uucp –r seci!uchile!~/find-ls.gz ~/uchile.files.gz
```

这种将一个作业通过几个系统传送的技术叫做转发（*forwarding*）。在上面的例子中，使用转发的原因可能是 **seci** 有对 **uchile** 的 UUCP 访问，而你的系统却没有。然而，如果你运行了一个 UUCP 系统，你会想要将转发服务限制到你可信任的很少几台主机，以免会为他们去下载一个最新的 X11R6 源程序版本而花费惊人的电话费用。

缺省地，Taylor UUCP 完全不允许转发活动。为了对某一特定系统启动转发，你可以使用 *forward* 命令。这个命令指定了能够请求你的系统进行转发作业的站点的一个列表。例如，为了允许 **pablo** 能够从 **uchile** 请求文件，**seci** 的 UUCP 管理员必须将下列几行内容加入到 *sys* 文件中：

```
####################
# pablo
system        pablo
…
forward          uchile
####################
# uchile
system        uchile
```

…

```
forward-to    pablo
```

uchile 的 *forward-to* 条目是必须的，这样任何由 **uchile** 返回的文件才能传到 **pablo**。否则的话，UUCP 将丢弃它们。这个条目使用了一个 *forward* 命令的一个变种，它只允许 **uchile** 能通过 **seci** 将文件发送到 **pablo**；而不能反之。

如果要许可到任何系统的转发，可以使用特殊关键字 *ANY*（需大写）。

12.5 为电话拨入设置你的系统

如果你想将你的站点设置成电话可拨入的，你必须允许登录到你的串行端口上，并且调整某些系统文件以提供 UUCP 的帐号。这将是本节的主题内容。

12.5.1 设置 *getty*

如果你想使用串行线路作为拨入端口，你就必须在这个端口上启动一个 *getty* 进程。然而某些 *getty* 的实现并不适用于此，因为你通常希望使用一个串行端口既可以拨入也可以拨出。因此你必须确信使用一个能够与其它程序（如 *uucico* 或 *minicom*）共享的这条线路的 *getty*。能够这样做的一个程序是 *getty_ps* 软件包中的 *uugetty*。大多数 Linux 发行版有这个软件包；请在你的*/sbin* 目录中检查 *uugetty* 是否存在。我所知道的另一个程序是 Gert Doering 的 *mgetty*，这个程序也支持接收传真。你可以从 sunsite.unc.edu 上获取该程序执行文件或源程序的最新版。

解释 uugetty 与 mgetty 处理登录方法的不同之处已不属本小节；有关更详细的信息，请参见 Grag Hankins 的 Serial HOWTO，以及附随 *getty_ps* 和 *mgetty* 的文档。

12.5.2 提供 UUCP 帐号

下一步，你必须设置用户帐号以允许远程站点能够登录进你的系统并且建立一 UUCP 连接。一般来讲，你要为每一个查询你的系统分别提供独立的登录名。当为系统 **pablo** 设置一个帐号时，你通常可以给它一个 **Upablo** 作为用户名称。

对于通过串行端口拨入的系统，你通常必须将这些帐号加入到系统密码文件*/etc/passwd* 中。一个好的做法是将所有 UUCP 登录都放进一个特殊的组中，如 **uuguest** 组。这个帐号的主目录应该设置成公共假脱机目录*/var/spool/uucppublic*；它的登录 shell 必须是 *uucico*。

如果已安装了影子密码组件，那么你可以使用 *useradd* 命令来做：

```
# useradd –d /var/spool/uucppublic –G uuguest –s /usr/lib/uucp/uucico
Upablo
```

如果你没有安装影子密码组件的话，你可能就需要手工编辑*/etc/passwd*，加入象下面的一行文字，其中 5000 和 150 是分别分配给用户 **Upablo** 和组 **uuguest** 的数字 uid 和 gid。

```
    Upablo:x:5000:150:UUCP
Account:/var/spool/uucppublic:/usr/lib/uucp/uucico
```

在建立了帐号以后，你还必须使用 passwd 命令来设置它的密码以激活这个帐号。

为了对通过 TCP 连接到你的站点的 UUCP 系统提供服务，你必须设置 *inetd* 来处理 *uucp* 端口上的连入。这是通过在/etc/inetd.conf 中增加下面所示的一行来做到的：[12]

```
    uucp       stream tcp nowaitroot                          /usr/sbin/tcpd
/usr/lib/uucp/uucico –l
```

-l 选项使得 *uucico* 执行它自己的登录认证。如同标准的 *login* 程序一样，它也将提示输入一个登录名和一个密码，但是它将依赖于自己的密码数据库而不是*/etc/passwd*。这个私有的密码文件是*/usr/lib/uucp/passwd*，内含登录名和密码对：

```
    Upablo IslaNegra
    Ulorca co’rdoba
```

当然，这个文件由 **uucp** 拥有并且具有权限 600。

如果这个数据库让你感觉到可以在通常的串行登录上使用它将是一个好点子的话，那你会大大地失望的，因为在没有重大调整之前是不可能做到的。首先，你需要使用 Taylor UUCP 1.05，因为它允许 *getty* 使用-u 选项将呼叫用户的登录名传递给 *uucico*。[13]然后，你必须哄骗你所用的 *getty* 来调用 *uucico* 而不是通常的*/bin/login*。这用 *getty_ps* 并通过在配置文件中设置 *LOGIN* 选项，你就可以做到。然而，这样就完全取消了交互式的登录。而另一方面，*mgetty* 就有个很好的特性，它允许你基于用户所提供的名字而调用不同的登录命令。例如，你可以告知 *mgetty* 对所有以大写字母 U 开头的登录名使用 *uucico*，而所有其他的用户仍用标准 *login* 命令来处理。

为了保护你的 UUCP 用户免受利用假的系统名称的呼叫者的攻击而造成用户邮件的混乱，你应该在 *sys* 文件中的每个系统条目中增加 *called-login* 命令。这种情况在下节保护自己不上骗子的当中描述。

12.5.3 保护自己不上骗子的当

UUCP 最严重的问题之一是呼叫系统可能会假冒别的系统的名称；在登录进之后它向被呼叫的系统宣称自己的名字，但是这个服务器没有任何办法来验证它。因而，攻击者能够登录进他/她自己的 UUCP 帐号，然而却假装是另外一个人，并取得其他站点的邮件。这对于提供了匿名 UUCP 登录的情况特别有问题，此时登录密码是公开的。

除非你能够相信呼叫你的系统的所有站点都是诚实的，否则你就必须防范这类冒名顶替者。为了处理这种问题，需要在 *sys* 中应用 *called-login* 要求每个系统使用一个特别的登录名。一个系统条目的样本看上去象这样：

```
    system       pablo
    … usual options …
    called-login     Upablo
```

这样做的结果是每当一个系统登录进来并声称它是 **pablo** 时，*uucico* 就会检测它是否是以 **Upablo** 登录进来的。如果不是，那么呼叫的系统就将被拒绝，连接也会断开。你应该养成一个习惯，就是在 *sys* 文件的每个系统条目中都加入 *called-login* 命令。对所有的系统都这样做是很重要的，而不管他们是否会呼叫你的站点。对于那些从不会呼叫你的站点，你应该将 *called-login* 设置成完全伪造的用户名，比如说 **neverlogsin**。

12.5.4 像患偏执狂的 – 呼叫序列检查（Call Sequence Checks）

避开和侦测出冒名顶替者的另一种办法是使用呼叫序列检查。呼叫序列检查能够帮助你保护自己免受入侵者想方设法偷取你用来登录 UUCP 系统的密码。

当使用呼叫序列检查时，两端的机器都将保留有至今为止所建立的连接次数。随着每次连接，这个次数会随每次连接而逐渐增加。在登录进来后，呼叫者会发送出自己的呼叫序列次数，而被呼叫者将检查这个数与它自己的数字作比较。如果他们不匹配，那么连接的企图将被拒绝。如果初始的数字是随机选取的一个数字，那么攻击者就很难猜出正确的呼叫序列次数。

而且呼叫序列检查能为你做得更多：即使某些非常聪明的家伙能够猜出你的呼叫序列次数（呼叫序列号）和你的密码来，你就会察觉出来的。当攻击者呼叫你的 UUCP 喂信和偷取你的邮件时，这会将喂信的呼叫序列次数增一。当下一次你呼叫你的喂信系统并试图登录时，远程的 *uucico* 将拒绝你登录，因为序列号已不再匹配了！

如果你已建立了呼叫序列检查，你应该定期地检查你的日志文件以检测出可能的攻击错误信息。如果你的系统拒绝了呼叫系统提供的呼叫序列号，*uucico* 将在日志文件中放置一条信息指出象“序列不匹配，呼叫被拒绝”（“Out of sequence call rejected”）。如果你的系统由于序列号不同步而被它的喂信者拒绝的话，它将在日志文件中放置一条信息说“握手失败（RBADSEQ）”（“Handshake failed（RBADSEQ）”）。

为了开启呼叫序列检查功能，你必须将下列命令放入系统条目中：

```
# enable call sequence checks
sequence      true
```

而且，你还必须创建一个放置序列号的文件。Taylor UUCP 在远程站点的假脱机目录中保存序列号文件，称为*.Sequence*。它必须属于 **uucp**，并且有属性 600（也即仅有 **uucp** 可读写）。最好用随意的、意见一致的起始值来初始化这个文件。否则的话，攻击者可能会通过试用所有的比如说小于 60 的数来设法猜出这个数字来。

```
# cd /var/spool/uucp/pablo
# echo 94316 > .Sequence
# chmod 600 .Seguence
# chown uucp.uucp .Sequence
```

当然，远端站点同时也要开启呼叫序列检查的功能，并且使用完全一样的序列号来开始。

12.5.5 匿名 UUCP

如果你想对你的系统提供匿名 UUCP 的访问，那么首先你必须象上述设置一个特别的帐号。一个通用实际的办法是将登录名和密码都指定为 **uucp**。

另外，针对不明系统你必须设置一些安全方面的参数。例如，你可能想要禁止它们执行你系统上的任何命令。然而，你不能在 *sys* 文件的条目中设置这些参数，因为 *system* 命令要求有系统的名字，而你却没有这些名字。Taylor UUCP 通过 *unknown* 命令解决了这个问题。*unknown* 可以被用于 *config* 文件中来指定任何可以出现在系统条目中的命令：

```
unknown         remote-receive “/incoming
unknown         remote-send “/pub
unknown         max-remote-debug none
unknown         command-path /usr/lib/uucp/anon-bin
unknown         commands rmail
```

这将限制未知系统只能从 *pub* 目录下下载文件和往*/var/spool/uucppublic* 目录中上载文件。下一行将使得 *uucico* 忽略远程系统要求打开本地调试功能的任何请求。最后两行允许未知系统执行 *rmail*；但是所指定的命令路径使得 *uucico* 仅能在一个名为 *anon-bin* 的私有目录中查找 *rmail* 命令。这允许你提供某些特殊的 *rmail*，比如将所有邮件转发给超级用户以作检查，但同时防止向其它站点注入任何邮件。

为了开启匿名 UUCP 功能，你必须在 *config* 文件中指定至少一条 *unknown* 语句。否则的话 *uucico* 将拒绝任何未知的系统。

12.6 UUCP 低层协议

为了与远端协商进程控制和文件传输，uucico 使用了一个标准的信息（消息）集。通常称它们为高层（高级）协议。在初始化阶段和挂断期间，这些通常是以字符串发送的。然而，在真正的传输阶段，使用了对于高层协议几乎透明的一个另外的低层协议。这使得查错成为可能，例如，当使用不可靠的线路时。

12.6.1 协议综述

就如同 UUCP 被用于象串行线路或 TCP、甚至 X.25 等的不同类型的连接上一样，还需要特定的低层协议。另外，UUCP 的几种不同实现也引进了功能类似的几种不同协议。

协议可以被分成两类：流式的和基于包的协议。后一类的协议将一个文件的传输作为一个整体来处理，很可能还为其计算出一个校验和。这几乎无须任何额外开销，但要求有一个可靠的连接，因为任何传输中的错误都将导致整个文件被重传。这些协议通常用于 TCP 连接上，但并不适用于电话线路。尽管现代 modem 在处理纠错方面已做的很好了，但并不是非常完善的，对于计算机与 modem 之间也没有任何的检错功能。

在另一方面，包协议将文件分割成大小相等的数块数据块。每个包的收发都是分别进行的，校验和也被计算出来，传输的响应被送回给发送者。为了使这种传输更有效率，发明了滑动窗口（sliding-window）协议，它允许在任何时候的有限数量的显著响应。这极大地降低了 uucico 在传输期间的等待时间。然而，与流式协议相比显得较大的额外开销使得包协议在 TCP 上的使用不够有效。

数据宽度也同样造成区别。有时候，在一串行连接上发送八比特的字符是不可能的，比如，如果连接通过一台笨拙的终端服务器时。在这种情况下，宽度为八比特的字符集在传输时必须加引号。当你在一个七比特的连接上传输八比特的字符时，它们不得不考虑最坏的情形，这将把数据的传输量加倍，尽管此时硬件所做的压缩可能对此作出一定的补偿。对于能够随意地传输八比特字符的线路通常称为纯八比特的（eight-bit clean）。所有 TCP 连接都是这种情况的，包括大多数的 modem 连接。

以下的协议是 Taylor UUCP 1.04 所含有的：

g 这是一个非常普通的协议，几乎所有的 *uucico* 都必须理解它。它进行完全的错误检测因此非常适用于多噪声的电话线链接。*g* 要求一个纯八位的连接。它是一个使用滑动窗口技术的基于包的协议。

i 这是一个双向的包协议，能够同时用于收发文件。它需要一个全双工的连接以及一个纯八比特数据宽度。目前它只适用于 Taylor UUCP.

t 这是一个打算在 TCP 连接上使用的协议，或者在其它无差错网络上使用。它使用 1024 字节的包，要求纯八位连接。

e 这个协议基本上同 *t* 所做类似的事情。主要的不同之处是 *e* 是一个流式协议。

f 这是打算在可靠的 X.25 连接上使用的。它是个流式协议并期望一个七比特数据宽度。八比特字符将被引用，使得效率下降。

G 这个 System V Release 4 的 g 协议版本。某些其它的 UUCP 同样也能使用该协议。

a 这个协议与 ZMODEM 类似。它要求有一个八比特连接，但会引用某些象 XON 和 XOFF 的控制字符。

12.6.2 调整传输协议

所有的协议在包的大小、超时长短等等上允许有某些变化。通常，所提供的缺省设置在标准环境下工作的很好，但对于你的情况并不一定是最优的。例如，g 协议使用的窗口的大小为 1 到 7、包的大小为 2 的幂次从 64 到 4096。[14] 如果你的电话线路噪声太大，使得有大于 5%的包丢失的话，你就应该减小包的大小和收缩窗口。另一方面，在非常好的电话线路上，对于每 128 字节发送一个 ACK 响应的协议开销就可证明是非常浪费了，所以对于这种情况，你可以将包的大小增大至 512 或者甚至是 1024。

Taylor UUCP 提供了一种机制，通过在 *sys* 文件中用 *protocol-parameter* 命令调整这些参数来满足你的需求。例如，为了在与 pablo 的对话时将 g 协议的包大小设置为 512，你需要加入：

```
system       pablo
…
protocol-parameter g packet-size 512
```

各个协议的可调整参数及它们的名称不尽相同。有关这些参数的完整的列表，请参阅 Taylor UUCP 原代码所带的文档。

12.6.3 选择特定的协议

并不是每一个 *uucico* 的实现都能识别（理解）每种协议的，所以在初始握手阶段，两端的进程就必须使用同一个公共协议。主 *uucico* 通过向次 *uucico* 发送 `Pprotlist` 来提供一个所支持的协议的列表，次 *uucico* 从中选择一个进行对话。

根据所使用的端口的类型（modem、TCP、或直接连接），*uucico* 将构成一个协议列表。对于 modem 和直接连接来说，这个列表通常由 *i*、*a*、*g*、*G* 和 *j* 组成。对于 TCP 连接，列表为 *t*、*e*、*I*、*a*、*g*、*G*、*j* 和 *f* 组成。你可以用 *protocols* 命令来覆盖这个缺省列表，这个命令可以在系统条目和端口条目中指定。例如，你可以如下对你的 modem 端口编辑 port 文件中的条目：

```
port             serial1
…
protocols     igG
```

这会要求所有通过这个端口的接入或呼出连接都使用 *i*、*g* 或 *G*。如果远端系统不支持其中的任何一个，那么对话就会失败。

12.7 故障诊断

本节描述了 UUCP 连接可能出错的地方，并给出查错的建议。然而，这个问题并不是我最注重的。有很多方面会导致出错。

在任何情况下，使用`-xall` 开启调试，并观察假脱机目录 *Debug* 中的输出。这将帮助你快速地识别出问题的所在。同样，我一直发现在 modem 不能连接时将 modem 的喇叭打开是很有帮助的。对于 Hayes 兼容 modem 来讲，这是通过往 *dial* 文件 modem 对话中加入“`ATL1M1 OK`”做到的。

首先要检查的总是是否所有文件的权限设置正确了。*uucico* 应该 setuid **uucp**，并且所有在 */usr/lib/uucp*、*/var/spool/uucp* 和*/var/spool/uucppublic* 中的文件应该属于 **uucp**。在假脱机（spool）目录中还有一些隐含文件也必须属于 **uucp**。

***uucico* 不断指出“Wrong time to call”（呼叫时间不对）：**这可能是指在 *sys* 中的系统条目里，你没有用 *time* 命令指定远程系统被呼叫的具体时间，或者你给出的时间禁止在当前时间呼叫。如果没有给出时间表，*uucico* 便会假定系统将永远不会被呼叫。

***uucico* 抱怨站点已被锁定：**这表示 *uucico* 在*/var/spool/uucp* 中检测到远程系统的一个锁定（lock）文件。这个锁定文件可能是上一次呼叫系统没成功或被杀死而留下的。然而，也可能是有另一个 *uucico* 进程正在试图拨到远程系统并死在对话脚本中等等。如果在连接至远程系统时这个 *uucico* 进程没有成功，那么就用 hangup 信号杀死它，并删除任何遗留下来的锁定文件。

我能够连接到远端站点，但是对话脚本失败了：观察你从远端站点接收到的文字。如果它们是混乱的，这可能是速度不匹配的问题。否则的话，请确信收到的是否是你的对话脚本所期望的。记住，对话脚本以一期望的字符串开始的。如果你接收到登录提示，并发出了你的名字，但是再也没

有收到输入密码的提示，那么在发送它之前插入某些延迟，或者甚至在字符之间插入延迟。对于你的 modem 来讲你可能太快了。

我的 modem 不拨号：在 *uucico* 呼叫出时，如果你的 modem 没有指示出 DTR 线上有信号，你很可能没有将正确的设备给予 *uucico*。如果你的 modem 识别出了 DTR，那么用一个终端程序来检验一下你能否对它进行写。如果可以的话，那么在 modem 对话开始处用\E 打开回显功能。如果在 modem 对话期间不能回显你的命令，请检查你的线路速度对于你的 modem 来讲是否太高或太低。如果你看见了回显，请检查你是否已经禁止了 modem 响应，或将它们设置成数字代码了。验证对话脚本本身是否正确。记住，你需要写上两个反斜杠才能向 modem 发送一个反斜杠。

我的 modem 尝试着拨号，但是无法进行下去：在电话号码中插入延迟。这对于从公司内部电话网上拨出尤其有用。对于在欧洲的人来说，通常是脉冲拨号的，请试试音频拨号。在某些国家中，邮政服务近来已经更新了它们的邮电网。音频拨号有时是有帮助的。

我的日志文件说我有很高的包丢失率：这看上去象是一个速度问题。计算机与 modem 之间的链接可能太慢了（记住将它调整为最高的有效速率）？或者是否是你的硬件速度太慢而不能及时地处理中断响应？对于串行端口的一个 NSC 16550A 芯片来讲，38kbps 是能够很好地工作的；然而，如果没有 FIFO（象 16450 芯片），9600bps 是一个上限。同样，你应该确信串行线路的硬件握手功能是开启的（enabled）。

另外一个可能原因是端口上硬件握手没有开启。Taylor UUCP 1.04 对于 RTS/CTS 握手的调整是没有规定的。你必须使用下面的命令从 *rc.serial* 中明确地开启这个功能：

```
$ stty crtscts < /dev/cua3
```

我可以登录，但是握手失败了：那么，可能有几种问题。日志文件中的输出会告诉你很多信息。观察远端站点提供了什么协议（在握手期间它送来了字符串 `Pprotlist`）。可能它们不含有普通的协议（你在 *sys* 或 *port* 中选择了任何协议了吗？）。

如果远端系统发送了 `RLCK`，那么在远端系统上就有一个你的陈旧的锁定文件（lockfile）。如果不是因为你已经在不同的线路上连接至这个远端系统上的话，请请求远端系统管理员删除它。

如果它发送来 `RBADSEQ`，说明其它站点对你开启了对话计数功能，但是数字不匹配。如果它发送来 `RLOGIN`，那么说明不允许你用这个 id 登录。

12.8 日志文件

当编译 UUCP 套件以使用 Taylor 形式的日志时，你只有三个全局日志文件，都存于假脱机目录中。主要的日志文件被称为 *Log* 并含有有关连接建立和文件传输的所有信息。典型的片段看上去象这样（为了能够适合页面进行了小小的重排列）：

```
   uucico pablo - (1994-05-28 17:15:01.66 539) Calling system pablo (port
cua3)
   uucico pablo – (1994-05-28 17:15:39.25 539) Login successful
   uucico pablo – (1994-05-28 17:15:39.90 539) Handshake successful
               (protocol ‘g’ packet size 1024 window 7)
   uucico  pablo  postmaster  (1994-05-28  17:15:43.65  539)  Receiving
D.pabloB04aj
   uucico  pablo  postmaster  (1994-05-28  17:15:46.51  539)  Receiving
```

```
X.pabloB04ai
   uucico pablo postmaster (1994-05-28 17:15:48.91 539) Receiving
D.pabloB04at
   uucico pablo postmaster (1994-05-28 17:15:51.52 539) Receiving
X.pabloB04as
   uucico pablo postmaster (1994-05-28 17:15:54.01 539) Receiving
D.pabloB04c2
   uucico pablo postmaster (1994-05-28 17:15:57.17 539) Receiving
X.pabloB04c1
   uucico pablo – (1994-05-28 17:15:59.05 539) Protocol ‘g’ packets; sent
15,
                resent 0, received 32
   uucico pablo – (1994-05-28 17:16:02.50 539) Call complete (26 seconds)
   uuxqt pablo postmaster (1994-05-28 17:16:11.41 546) Executing
X.pabloX04ai
                (rmail okir)
   uuxqt pablo postmaster (1994-05-28 17:16:13.30 546) Executing
X.pabloX04as
                (rmail okir)
   uuxqt pablo postmaster (1994-05-28 17:16:13.51 546) Executing
X.pabloX04c1
                (rmail okir)
```

下一个重要的日志文件是 *Stats*，它列出了文件传输的静态参数。与上面传输相应的 *Stats* 部分看上去象这样：

```
   postmaster pablo (1994-05-28 17:15:44.78)
                received 1714 bytes in 1.802 seconds (951 bytes/sec)
   postmaster pablo (1994-05-28 17:15:46.66)
                received 57 bytes in 0.634 seconds (89 bytes/sec)
   postmaster pablo (1994-05-28 17:15:49.91)
                received 1898 bytes in 1.599 seconds (1186 bytes/sec)
   postmaster pablo (1994-05-28 17:15:51.67)
                received 65 bytes in 0.555 seconds (117 bytes/sec)
   postmaster pablo (1994-05-28 17:15:55.71)
                received 3217 bytes in 2.254 seconds (1427 bytes/sec)
   postmaster pablo (1994-05-28 17:15:57.31)
                received 65 bytes in 0.590 seconds (110 bytes/sec)
```

同样地，各行为了适合页面进行了分割。

第三个文件是 *Debug*。这是调试信息被写入的地方。如果你使用了调试功能，你应该确信这个文件有保护模式 600。根据你所选择的调式模式，它可能含有用于连接至远程系统的登录名和密码。

某些 Linux 发行版中所包含的 UUCP 执行文件是以使用 HDB 形式日志被编译的。HDB UUCP 使用了存储在*/var/spool/uucp/.Log* 下的整族日志文件。这个目录含有至少三个子目录，有 *uucico*、

uuxqt 和 *uux* 目录。它们含有相应命令产生的日志输出信息，针对每个站点分类进不同的文件。因此，当呼叫站点 **pablo** 时 *uucico* 的输出将进入*.Log/uucico/pablo*，而随后的 uuxqt 的运行将写入*.Log/uuxqt/pablo* 中。但是，写入各个日志文件中的行信息与 Taylor 的日志文件的相同。

当你对使用 HDB 形式日志编译的 UUCP 开启调试功能时，信息将被写入*/var/spool/uucp* 下的*.Admin* 目录。在向外呼叫期间，调试信息将送至*.Admin/audit.local*，而当某人呼叫进来时，*uucico* 的输出将被写入*.Admin/audit*。

注释

[1] 由 Ian Taylor 编制和版权所有，1993。

[2] 注意，尽管绝大多数 UUCP 命令必须 setuid 为 uucp，你必须确定 uuchk 程序却不是的。否则的话，尽管用户有模式 600，他们仍将可以显示出密码来。

[3] 如果你只是打算试用 UUCP，那么找一个你就近的 archive 站点的号码。写下登录名和密码 – 为了允许匿名下载它们是公开的。在绝大多数情况下，它们是 **uucp/uucp** 或 **nuucp/uucp**。

[4] 仅有的限制是这个名字不能长于 7 个字符，以防混淆对文件名的长度有限制的文件系统的主机。

[5] UUCP 映射计划在世界范围内注册所有的 UUCP 主机并确信这些主机名是唯一的。为了注册你的 UUCP 名字，询问处理你的邮件的该站点的维护者；他们会在这方面帮助你的。

[6] 老式的版本 2 UUCP 当被呼叫时并不广播自身的名字；然而，更新的实现版本常常是这样做的，Taylor UUCP 也是这样。

[7] tty 的波特率起码要与最高传输速率一样高。

[8] 如果远程系统运行 Taylor UUCP 的话，它就会遵循请求。

[9] 有些人代之以使用 ttyS*设备，这种设备只能用于拨入。

[10] 你也可以将 modem 配置成在检测到 DTR 上信号变化时复位自己。然而，某些 modem 不能这样做到，并且有时会挂起（无响应）。

[11] *rsmtp* 使用批处理 SMTP 来分发邮件。这在邮件一章中有描述。

[12] 注意，通常 *tcpd* 具有模式（mode）700，所以你必须作为 **root** 用户来调用它，而不是你通常所用的 **uucp**。

[13] –u 选项也存在于 1.04 版中，但仅是个空操作（no-op）。

[14] Linux 发行版中包括的绝大多数执行程序的缺省值设置为窗口大小是 7，包大小是 128。

第十三章 电子邮件

自从第一个网络被设计出来，连网最显著的用途之一就是电子邮件（eletronic mail）。它是从将一个文件从一台机器拷贝到另一台机器的简单服务开始的，并将该拷贝的文件添加到接收者的 *mailbox* 文件中。本质上来说，这仍然是 email 所做的全部工作，尽管不断增长的网络及其复杂的路由需求以及它的不断增大的消息负载量已经促使要求更加精心制作的方案。

已设计出了各种邮件交换的标准。Internet 上的站点始终坚持 RFC 822 中的设计，并被描述与机器无关的传输特殊字符方法的某些 RFC 所扩充，等等。对于“多媒体邮件”的更多思考目前也已作出，这涉及到有关在邮件消息中包含图形和声音。另一个标准，X.400，也已被 CCITT 定义。

有许多 UN*X 系统的邮件传输程序被实现。其中最为著名的是 Berkeley 大学的 *sendmail*，它被用于很多平台上。最初的作者是 Eric Allman，他现在又再次积极地工作于 *sendmail* 小组中。有两个 *sendmail-5.56c* 的 Linux 移植版本，其中之一将在第 15 章中讨论。目前正在开发的 *sendmail* 版本是 8.6.5。

Linux 最常用的邮件代理程序是 *smail-3.1.28*，是由 Curt Landon Noll 和 Ronald S.Karr 编写和版权所有。在大多数 Linux 发行版中都包含这个程序。下面，我们将简称它为 *smail*，尽管它还有其它完全不同的版本，但我们在这里并不讨论它们。

与 *sendmail* 相比，*smail* 就显得非常简单。当对于一个没有复杂路由要求的小站点处理邮件时，这两者的性能就非常相近。然而，对于大型站点，*sendmail* 总能高出一筹，因为它的配置方案是非常灵活的。

smail 和 *sendmail* 两者够支持一族配置文件，它们都需要进行自定义（定制）。除了邮件子系统运行所必要的信息以外（比如象本地主机名），还有许多参数需要调整。Sendmail 的主要配置文件开始是非常难理解的。它看上去就好象你的猫在按下了 shift 键的键盘上打过盹。*smail* 的配置文件就非常结构化并且比 *sendmail* 的容易理解，但没有给予用户很强的调整邮件箱性能的能力。然而，对于小型的 UUCP 或 Internet 站点，它们的配置所需要的工作基本上是一样的。

在本章中，我们将讨论什么是邮件以及作为一个管理员你要涉及什么问题。第 14 章和第 15 章将对第一次设置 *smail* 和 *sendmail* 给出指导。那里所提供的信息已足够让一个小型站点工作起来，但是还有许多的选项，你可以在你的计算机旁花费很多快乐的时间来配置这些奇妙的特性。

在本章的最后，我们将概要地讨论 *elm* 的设置，这是一个许多 UN*X 系统（包括 Linux）非常通用的邮件用户代理程序。

有关 Linux 上电子邮件方面问题的更多信息，请参考 Vince Skahan 的 Electronic Mail HOWTO，这是定期投递到 **comp.os.linux.announce** 上的。Elm、smail 和 sendmail 的源程序发行版同样也包含了非常广的文档资料，能够解答设置它们时所遇到的大多数问题。如果你在寻找有关电子邮件的一般性信息，有许多 RFC 涉及这个方面。它们列于本书后的参考书目中。

13.1 什么是邮件消息？

一个邮件消息是由消息体—它是发送者所写的文本、以及指明收信者的特定数据、传输介质等等组成，非常象信封上所见的信息。

这些管理用的数据可以分成两类；第一类是与特定传输介质相关的数据，就象发信者和收信者的地址。因此它们被称为 *envelope*。在消息传输途径中，它们可能会被转换。

第二类是处理邮件消息所需的任何数据，它们并不是针对任何传输机制的，比如消息的主题行

（subject line）、所有接收者列表、以及消息发送的日期。在许多网络中，将这些数据添置到邮件消息中已成为了标准，形成所谓的邮件头（*mail header*）。它与邮件体（*mail body*）之间相隔一空行。[1]

UN*X 世界中的很多邮件传输软件使用一个 RFC 822 中指定的头[标题]格式。它的最初目的是在 ARPANET 上的使用指定一个标准，但是由于它被设计成是与任何环境都独立的，它很容易地被其它网络采纳，包括许多基于 UUCP 的网络。

然而，RFC 822 仅是最伟大的公共奠基石。现有更多的标准已被构想出来以应付不断增长的需求，比如，数据加密、国际字符集的支持、以及多媒体邮件扩展（MIME）。

在所有这些标准中，标题[头]由几行组成，并由换行符来分割。每行是由从第一列开始的字段名、和由一个冒号和一个空格分割的字段本身组成。每个字段的格式和语义是随字段名的不同而有变化的。如果下一行是以一个 TAB 开始的，表示是标题字段的续行。各字段的顺序是随意的。

一个典型的邮件标题看上去象这样的：

```
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 MET
DST
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 21:47
MEST
Received: by ruby.ora.com (8.6.8/8.6.4) id RAA26438; Tue, 12 Arp 94 15:56
-0400
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，所有所需的标题字段都是由你所使用的邮件程序界面产生，象 *elm*、*pine*、*much* 或 *mailx*。然而有些字段是可选的，可以由用户自己加上去。例如，*elm* 允许你编辑部分消息头[标题]。而其它的则是由邮件传输软件加上去的。常用的标题字段以及它们的含义如下所示：

`From:` 含有发送者的 `email` 地址，以及可能还有“真实名字”。这里使用了一个格式的完整 `zoo`。

`To:` 这是接收者的地址。

`Subject:` 用几个词描述邮件的内容。起码这是*应该*做的。

`Date:` 邮件发送的日期。

`Reply-To:` 发送者指定接收者回信所发往的地址。对于你有几个帐号，但是只想在你常用的帐号下接收大量的邮件，此时这个字段是很有用的。该字段是可选的。

`Organization:` 产生邮件的机器所属的组织。如果你的机器是属于你个人的，你可以空着该字段，或者插入“个人”（“private”）或某些完全无意义的东西。这个字段是可选字段。

`Message-ID:` 产生邮件的系统的邮件传输程序生成的字符串。对于这条消息来讲是唯一的。

`Received:` 处理你的邮件的每一个站点（包括发送者和接收者的机器）在标题中插入的字段，给出它的站点名称、一个消息 `id`、收到该消息的时间和日期、从哪个站点来的以及使用了哪个传输软件。根据此，你可以跟踪消息走过的路由，并且如果出了差错你可以抱怨的负有责任的相关人士。

`X-`*anything*`:` 以 `X-`开头的标题行对任何邮件相关的程序都是不起作用的。它是用于实现还没有写入或不可能写入 RFC 的额外的特性的。例如，它用于 Linux Activists 邮件列表中，其中，是用 `X-Mn-Key:` 标题字段 来选择频道的。

这个结构的一个例外是最开头的一行。它以关键字 `From` 开始，紧接着是一个空格而是一个冒号。为了与普通的 `From:`字段相区别，它通常被引用为 `From_`。它包含消息所参与的 UUCP 大宗路径形式（下面将作解释）的路由、最后一台接收消息的机器处理消息的时间和日期以及一指明从哪台主机接收来的可选部分。由于每个处理过这个消息的系统都会生成这个字段，所以它通常包括在信封数据下。

因此 `From_`字段与某些老式的邮件程序是向后兼容的，但已不再经常使用，除了邮件用户界面程序还要依靠它来确定用户邮件箱中一个消息的开始处。也是为了消除以“From ”开始的消息体中的行所带来的潜在问题，在它之前放置“>”以避免任何这样的现象出现，这已经成为一标准方法。

13.2 邮件是如何投递的？

一般来讲，你要使用一个邮件界面程序象 *mail* 或 *mailx* 来编写邮件；或者使用更为复杂的象 *elm*、*mush* 或 *pine*。这些程序统称为邮件用户代理（*mail user agents*），或简称 MUA。如果你发出了一个邮件消息，那么在大多数情况下界面（接口）程序会将它传递给另一个程序去进行投递。这个程序称为邮件传输代理（*mail transport agent*），或简称 MTA。在某些系统中，对于本地的和远程的投递有不同的邮件传输代理程序；在其它系统上，仅有一个。远程投递的命令通常称为 *rmail*，其它的称为 *lmail*（如果存在的话）。

当然，本地邮件的投递仅仅是将进来的消息附加到接收者的邮件箱里。通常，本地 MTA 是懂得别名（将接收者的地址设置成指向其它的地址），和转发的（将用户的邮件重定向到某个其它的目的地）。同样，不能投递的消息通常必须反弹回来（*bounced*），也即，附带某些出错信息返回给发送者。

对于远程投递操作，所用的传输软件依赖于链接的属性。如果一个邮件必须在使用 TCP/IP 的网络上投递的话，常常会使用 SMTP。SMTP 表示简单邮件传输协议（Simple Mail Transfer Protocol），是在 RFC 788 和 RFC 821 中定义的。SMTP 通常直接连接至接收者的机器上，与远端的 SMTP 后台程序（daemon）协商消息的传输。

在 UUCP 网络中，邮件通常不是直接投递的，而是通过一系列的中间系统转发到目的主机上的。为了在 UUCP 链接上发送一个消息，发送 MTA 通常将使用 *uux* 在转发的系统上执行 *rmail*，并且将

消息在标准输入上喂信给转发系统。

由于这是对每个消息分别操作的，这将在主要的邮件 hub 上产生可观的工作负载，以及耗费不成比例的大量磁盘空间的数百个小文件构成的混乱的 UUCP 假脱机队列。[2] 因此某些 MTA 允许你以一个批处理文件从远程系统收集几个消息。如果使用了直接 SMTP 连接，那么这个批处理文件通常含有本地主机要发出的 SMTP 命令。这称为 BSMTP，或 *batched* SMTP。此时这个批处理会被发送给远程系统中的 *rsmtp* 或 *bsmtp* 程序，远程系统将如正常的 SMTP 连接一样来处理输入。

13.3 Email 地址

对于电子邮件，地址起码是由处理该人邮件的机器的名字，和这个系统能够识别的用户的代号组成。这可以是接收者的登录名，但也可以是任何别的代号。其它的邮件地址方案，象 X.400，使用一个更为一般的“属性”集，用于在一个 X.500 目录服务器中查询接收者的主机。

一个机器名字被解释的方法，也即，在哪个站点你的消息将最终结束，以及如何将这个名字与接收者的用户名结合在一起，这很大程度上依赖于你上的网络。

Internet 站点是符合 RFC 822 标准的，它需要**user@host.domain**这样的表示法，这里 **host.domain** 上主机的全资域名（fully qualified domain name）。当中的符号称作为“在”（“at”）符号。因为这个表示法不包含有到目的主机的路由，而是给出了（唯一的）主机名，这称为一绝对（*absolute*）地址。

在初始的 UUCP 环境中，流行的形式是 **path!host!user**，这里 **path** 描述了在到达目的 **host** 之前消息经过的一系列的主机。这种结构称为 *bang path* 表示法，因为一个感叹号近似地被称为一个“bang”。今天，许多基于 UUCP 的网络已经采用了 RFC 822，并且能够理解这种地址类型。

如今，这两种地址类型还不能很好地融合在一起。假设有一个地址**hostA!user@hostB**。这并不清楚在这个路径上‘@’符号是否是优先的；是否我们必须将消息发送到 **hostB**，再邮递到 **hostA!user**，还是先发送到 **hostA**，再转发到**user@hostB**？

然而，存在一种以 RFC 822 一致的方法指定路由：**<@hostA,@hostB:user@hostC>**表示在 **hostC** 上 **user** 的地址，这里 **hostC** 将经过 **hostA** 和 hostB 到达（以这个次序）。这种地址类型常常称为 *route-addr* 地址。

而且，还有一个‘**%**’地址操作符：**user%hostB@hostA**将首先发送到 **hostA**，它将最右边的（仅在这里的情况下）百分符号扩展成为一个‘**@**’符号。现在这个地址成为了**user@hostB**，邮件程序将很顺利地将你的消息转发到**hostB**，再而投递给 **user**。这种类型的地址有时被称为“Ye Olde ARPANET Kludge”，它的使用是不赞成的。然而，许多邮件传输代理产生这种地址类型。

其它的网络还有不同的地址方案。例如，基于 DECnet 的网络使用两个冒号作为地址的分割符，产生一个 `host::user` 地址。[3] 最后，X.400 标准使用了一个完全不同的方案，通过使用一族属性-值对描述一个接收者，就象国家和组织一样。

在 FidoNet 上，每个用户的确定是用一个代码象 **2;320/204.9**，由四个数字组成，表示区 zone（2 表示欧洲）、网络 net（320 代表巴黎和郊区）、节点 node（本地 hub）和点 point（个人的用户 PC）。FidoNet 的地址可以被映射到 RFC 822 上；上面的地址可以写成 **Thomas.Quinot@p9.f204.n320.z2.fidonet.org**。现在我没有说过域名好记吧？

使用这些不同的地址类型有某些隐含方面，这将在以下几节中讨论。然而，在一个 RFC 822 环境中，除了象`user@host.domain`这样的绝对地址，你很少会用到其它的地址类型。

13.4 邮件路由是如何工作的？

定向一个消息到接收者主机的过程称为路由选择（*routing*）。除了寻找到一条从发送站点到目的地的路径以外，它还包括错误检测以及速度和代价优化。

UUCP 站点处理路由选择的方法与 Internet 站点的有很大的不同。在 Internet 上，定向数据到接收者的主机（由它的 IP 地址确定）的主要任务是由 IP 的网络层来完成的，而在 UUCP 区中，路由必须由用户来提供，或者是由邮件传输代理生成的。

13.4.1 Internet 上的邮件路由选择

在 Internet 上，它完全依赖于目的主机是否要执行任何特定的邮件路由选择。默认的是通过查找目的主机的 IP 地址，直接将消息投递到目的主机去，而将实际数据的路由选择工作给 IP 传输层去做。

许多站点通常会想要指引所有入站的邮件到一个高效稳定的邮件服务器中，这个服务器能够处理所有这些数据流量，并让它在本地分发邮件。为了宣告这种服务，站点在 DNS 数据库中为它们的本地域公布一个所谓的 MX 记录。MX 代表邮件交换器（*Mail Exchanger*）基本的意思是表明服务器主机很愿意为本域中的所有机器充当一邮件转发器。MX 记录也可以用于处理自己没有连接到 Internet 上的主机的交通流量，比如 UUCP 网络，或者是携带着机密信息的公司网络上的主机。

MX 记录也有一个与之相关的优先权（*preference*）。这是一个正整数。对于一台主机如果存在几个邮件交换器的话，那么邮件传输代理将会试图把消息传输到具有最低优先权值交换器上，仅当这样做失败时才会尝试一台具有高一级优先权值的主机。如果本地主机本身就是目的地址的一个邮件交换器，它就不必将消息转发到优先权值比自己高的任何主机去；这是避免邮件循环（loops）的一个安全方法。

假设有一个叫 Foobar Inc.的公司，想要他们所有的邮件由他们的称为 **mailhub** 的机器来处理。那么他们就应该在 DNS 数据库中有一个象这样的 MX 记录：

```
foobar.com       IN  MX     5   mailhub.foobar.com
```

这宣告了 **mailhub.foobar.com** 是一个 **foobar.com** 上的有优先权值 5 的邮件交换器。一个希望把消息投递到**joe@greenhouse.foobar.com**的主机将检查 **foobar.com** 的 DNS，会发现 MX 记录指向 **mailhub**。如果此时没有优先权值小于 5 的其它 MX 存在，这个消息将被投递到 **mailhub**，然后被分发给 **greenhouse**。

上面实际上仅是 MX 记录如何工作的一个轮廓。有关 Internet 上邮件路由选择的更多信息，请参考 RFC 974。

13.4.2 UUCP 世界中的邮件路由选择

UUCP 网络上的邮件路由选择就比 Internet 上的复杂得多，因为传输软件本身不执行任何的路由选择功能。在早期，所有的邮件得用 bang paths 来寻址。Bang paths 指定了一张主机的列表，各主机名用感叹号分开，通过这些主机来转发消息，后接用户名。为了将一封信寻址到名为 **moria** 的

机器上的 **Janet** 用户，你就得使用路径 **eek!swim!moria!janet**。这将会把邮件从你的主机发送到 **eek**，再从 **eek** 发送到 **swim** 最后到 **moria**。

这种技术明显的不足之处是它需要你记住很多有关网络拓扑的细节、快速链接等等。更糟的是，网络拓扑的变化—象链接被删除了或主机被移走了—可能导致消息传输的失败，简单的原因只是由于你不知道网络结构变化了。还有，如果你移到了不同的地方，你很可能就要更新所有这些路由了。

然而，还有一件事使得源路由选择成为必需的是不明确的（多义的）主机名的存在：例如，假设有两个站点名字都叫 **moria**，一个在 U.S.，另一个在法国。那么 **moria!janet** 指的是哪一个呢？这可以通过指明到达 **moria** 的路径弄清楚。

消除多义主机名的第一步是 UUCP 映射计划组（*The UUCP Mapping Project*）的成立。它位于 Rutgers 大学，对所有官方的[正式]的 UUCP 主机进行登记注册，并记录下他们的 UUCP 邻居和他们的地理位置，确信没有那个主机名被使用了两次。由映射计划组收集的信息作为 *Usenet Maps* 公布出来，它会定期地通过 Usenet 进行发布。[4] Map 中一个典型的系统条目看上去象这样的（在删除了注释语句之后）。

```
Moria
        bert(DAILY/2),
        swim(WEEKLY)
```

这个条目说明 **moria** 有一个至 bert 的连接--**moria** 每天对它呼叫两次、和一个到 **swim** 的连接—**moria** 每星期对它呼叫一次。下面我们将更详细地讨论映射文件的格式。

使用映射文件中提供的连接信息，你可以自动地生成从你的主机到达任何目的站点的全路径。这个信息一般存储在 *paths* 文件中，有时也被称为路径别名数据库（*pathalias database*）。假设映射表明你可以通过 **ernie** 到达 **bert**，那么从上述映射中为 **moria** 生成路径别名条目看上去象这样的：

```
moria          ernie!bert!moria%s
```

如果你现在给出一个目的地址**janet@moria.uucp**，你的 MTA 就会取得上面的路由，并将消息用 **bert!moria!janet** 作为信封地址发送到 **ernie**。

然而，从完整的 Usenet 影射来建立一个 *paths* 文件并不是一个好主意。其所提供的信息通常是有误导的，而且有时是过时的。因此，只有很少几个主要的主机使用完整的 UUCP 世界映射来建立它们的 *paths* 文件。大多数的站点仅仅维护着它们周邻站点的路由选择信息，并将需要发送到它们数据库中找不到的站点的任何邮件发到一个有更多完整路由选择信息的灵敏主机上。这种方案被称作灵敏-主机路由选择（*smart-host routing*）。只有一个 UUCP 邮件连接的主机（即所谓的页站点（*leaf sites*））本身不做任何的路由选择；它们完全依赖于它们的灵敏-主机。

13.4.3 混合 UUCP 和 RFC 822

至今为止针对 UUCP 网络邮件路由选择问题最好的解决方案是在 UUCP 网络中采用域名系统。当然，你不能在 UUCP 上查询一个名字服务器。然而，许多 UUCP 站点已经在自己内部组成了与路由选择相协调的小型域。在映射中，这些域宣称一台或两台主机作为他们的邮件网关，所以在域中不需要对每台主机都有一个映射条目。网关将处理所有流入和流出域的邮件。在域中的路由选择方案对于外界世界来说是完全看不见的。

这个方法与上述的灵敏-主机路由选择方案配合的很好。全局路由选择信息仅由网关来维护；一

个有很少主机的域将只有一个很小的手写 paths 文件，其中列出了他们域内部的路由，以及到邮件 hub 的路由。即使是邮件网关也不再需要有世界上每一台 UUCP 主机的路由选择信息，除了他们所服务的域的完整路由选择信息以外，现在仅需要在它们的数据库中含有至整个域的路由。例如，下面所示的路径别名条目（pathalias entry）将会把 **sub.org** 域中的所有站点的邮件路由到 **smurf**：

```
.sub.org        swim!smurf!%s
```

任何寻址到**claire@jones.sub.org**的邮件将被用 **smurf!jones!claire** 作为信封地址发送到 **swim**。

域名空间的层次化组织结构允许邮件服务器将非常特定的路由与一般的路由相混合。例如，一个在法国的系统可能有一个对 **fr** 子域的特定路由，但是将会把 us 域中任何主机的邮件路由到 U.S. 中的某个系统上。用这种方法，基于域的路由选择（就如这种技术所称的）大大地减小了路由选择数据库的大小和所需的管理负担。

然而，在 UUCP 环境中使用域名的主要好处是与 RFC 822 的兼容性使得 UUCP 网络和 Internet 之间的网关变得简单。现今许多 UUCP 域与 Internet 网关都有一个链接以作为它们的灵敏-主机。通过 Internet 发送消息更快，路由选择信息也更可靠，因为 Internet 主机能够使用 DNS 而不是 Usenet 映射。

为了从 Internet 能够传过来，基于 UUCP 的域通常有自己的 Internet 网关宣告了一个 MX 记录（上面已对 MX 作过讨论）。例如，假设 **moria** 属于 **orcnet.org** 域。**gcc2.groucho.edu** 作为他们的 Internet 网关。因而 **moria** 使用 **gcc2** 作为它的灵敏-主机，所以发往外部域的所有邮件都通过 Internet 投递了出去。另一方面，**gcc2** 会为 **orcnet.org** 宣告一个 MX 记录，并将进入 **orcnet** 站点的所有入站邮件投递到 **moria**。

还有一个仅剩的问题是 UUCP 传输程序不能处理全资域名。大多数 UUCP 站点设计成应付多至八个字符的站点名，有些则更少，而使用非字母数字的字符比如象点（dot）则完全不可能了。

因此，就需要 RFC 822 名称与 UUCP 主机名之间的某些映射。映射的方法完全依赖于实现。映射 FQDN 到 UUCP 名字的一个常用方法是使用路径别名：

```
moria.orcnet.org ernie!bert!moria!%s
```

这将从指明一全资域名的地址中产生一个纯 UUCP 风格的 bang path。有些邮件程序为此提供了一个特殊的文件；例如，*sendmail* 为此使用了 *uucpxtable*。

当需要从 UUCP 网络发送邮件到 Internet 时，有时就需要反向转换（通俗地称为域名化）。由于邮件发送程序在目的地址中使用了全资域名，在转发消息给灵敏-主机时，通过不从信封地址上删除域名就可以避免这个问题。然而，仍有某些 UUCP 站点不属于任何域。它们通常通过附加伪域（pseudo-domain）**uucp** 来进行域名化。

13.5 路径别名和映射文件格式

路径别名数据库在基于 UUCP 的网络中提供了主要路由选择信息。一个典型的条目看上去象这样（站点名和路径用制表符分开）：

```
moria.orcnet.org ernie!bert!moria!%s
moria        ernie!bert!moria!%s
```

这使得到 **moria** 的任何消息被通过 **ernie** 和 **bert** 进行投递。如果邮件程序没有在这些名字之间映射的独立方法，**moria** 的全资名称和它的 UUCP 名字就都必须给出。

如果你想把发到某个域中主机的所有消息定向到它的邮件中继，你也需在路径别名数据库中指定一个路径，给出域名作为目标，域名前放置一个点。例如，如果在 **sub.org** 中的所有主机可以通过 **swim!smurf** 到达，那么路径别名就看上去象这样：

```
.sub.org        swim!smurf!%s
```

仅当你运行的是一个没有很多路由选择的站点时，编写一个路径别名文件才是可行的。如果你要为大量的主机进行路由选择的话，那么更好的方法是使用 *pathalias* 命令从映射文件中创建这个文件。映射能够很容易地维护，因为通过编辑系统的映射条目，你可以很容易地加入和移去一个系统，并重新创建映射文件。尽管由 Usenet Mapping Project 发布的映射不再用于路由选择，更小的 UUCP 网络仍可以在他们自己的映射集中提供路由选择信息。

一个映射文件主要是由站点的列表组成，列出每个系统查询和被查询的站点。系统名字从第一列开始，后跟用逗号分开的链接列表。列表可以跨行继续如果下一行以一个制表符开始。每个链接由站点名后跟一个用括号括住的代价组成。代价是一个算术表达式，由数字和符号代价构成。以 hash 符号开始的行将被忽略。

作为一个例子，考虑 **moria**，它每天查询 **swim.twobirds.com** 两次，每星期查询 **bert.sesame.com** 一次。更有甚者，到 bert 的链接只使用一个低速 2400bps 的 modem。Moria 会发布下面的映射条目：

```
moria.orcnet.org
       bert.sesame.com(DAILY/2),
       swim.twobirds.com(WEEKLY+LOW)

moria.orcnet.org = moria
```

最后一行也使得它在它的 UUCP 名字下已知。注意，它必须是 *DAILY/2*，因为每天呼叫两次实际上将这个连接的代价减半。

使用 *pathalias* 这样的映射文件中的信息可以计算出到达列于路径文件中任何目的站点的优化路径，并产生一个路径别名数据库，然后这个数据库可以用于到这些站点的路由选择。

pathalias 还提供了象站点隐藏（也即，让站点只能通过网关来访问）等几个特性。详细信息和链接代价列表请参见 *pathalias* 手册页。

映射文件中的注释通常含有所描述站点的额外信息。注释有一固定的格式，所以这些信息可以从映射文件中抽取出来。例如，一个称为 *uuwho* 的程序使用从映射文件创建的数据库用精细格式化的方式来显示这些信息。

当你在会给自己的会员发布映射文件的组织登记你的站点时，通常你要填写这样一个映射条目。

下面是一个映射条目的样本（实际上，它是我的站点）：

```
#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 1993
#
monad  brewhq(DALLY/2)
# Domains
monad = monad.swb.de
monad = monad.swb.sub.org
```

在头两个字符后面的空间是个制表符 TAB。大多数字段的意思都是很显然的；你会从你注册的域那里收到详细的说明。L 字段是非常有趣的：它以经纬度给出了你的地理位置用于画出显示每个国家所有站点的地图以及全世界的地图来。[5]

13.6 配置 *elm*

elm 代表“电子邮件”（“electronic mail”）的意思，是非常合理命名的 UN*X 工具之一。它提供了一个全屏幕界面并有一个很好的帮助特性。这里我们不讨论如何使用 *elm*，而是详细叙述它的配置选项。

理论上讲，无须配置你就可以运行 *elm*，并且样样工作的都很好 --- 如果你幸运的话。但是有几个选项是必须配置的，尽管只是有必要而已。

当 *elm* 开始运行时，它从*/usr/lib/elm/*中的 *elm.rc* 文件里读取一族配置变量。然后，它将试图在你的主目录中读取文件*.elm/elmrc*。这个文件一般不是你自己写的。它是当你从 *elm* 的选项菜单中选择了“save options”时被创建的。

私有 *elmrc* 文件的选项集也存在于全局 *elm.rc* 文件中。你的私有 *elmrc* 文件中的许多选项将覆盖全局文件中对应的选项。

13.6.1 全局 *elm* 选项

在全局 *elm.rc* 文件中，你必须设置属于你的主机名的选项。例如，在虚拟酿酒厂，**vlager** 的这个文件将含有下面信息：

```
#
# The local hostname
hostname = vlager
#
# Domain name
hostdomain = .vbrew.com
#
# Full quallified domain name
hostfullname = vlager.vbrew.com
```

这些选项设置 *elm* 的本地主机名的概念。尽管这个信息很少用到，但是你应该设置这些选项。注意，这些选项只有在全局配置文件中才有效；当存在于私有的 *elmrc* 中时，它们将被忽略掉。

13.6.2 国家字符集

近来，已有提议对 RFC 822 标准进行修正以支持各种类型的报文，比如无格式文本（plain text）、二进制数据、Postscript 文件等等。有关这些方面的标准集和 RFCs 通常称为 MIME，或多用途互连网邮件扩展（Multipurpose Internet Mail Extensions）。在其它方面，这也使得接收者在写报文时知道是否使用了非标准 ASCII 码，例如，使用法语重音符或德语变音符号。*elm* 在某些程度上支持这些扩展。

Linux 内部用来表示字符所使用的字符集通常称为 ISO-8859-1，这是它所遵守的标准的名字。它也以 Latin-1 知名。任何使用这个字符集的报文应该在标题部分有下面这一行：

```
Content-Type:    text/plain; charset=iso-8859-1
```

接收系统会识别出这个字段并在显示信息时采取一定的措施。*text/plain* 信息的默认值是一个值为 *us-ascii* 的 *charset*。

为了显示非 ASCII 字符集的信息，*elm* 必须知道如何打印这些字符。默认地，当 *elm* 接收到 *charset* 字段为非 *us-ascii*（或者 conten type 不是 *text/plain*）的信息时，它试着使用称为 *metamail* 的命令来显示信息。需要使用 *metamail* 来显示的信息在观察窗口的第一列显示一个‘M’。

由于 Linux 的本身的字符集是 ISO-8859-1，调用 *metamail* 使用这个字符集来显示信息是不必要的。如果 *elm* 被告知显示能够理解 ISO-8859-1，它就不会使用 *metamail*，而是直接显示信息。这可以通过在全局 *elm.rc* 中设置下面选项来做到：

```
displaycharset = iso-8859-1
```

注意，即使当你从不会发送和接收含有任何非 ASCII 的信息，你也应该设置这个选项。这是因为人们确实发送这样的信息时通常会配置他们的邮件程序将适当的 `Content-Type:` 字段默认地放入邮件的标题中，而不管他们是否只发送 ASCII 信息。

然而，在 *elm.rc* 中设置这个选项还不够。问题是当用它内建的分页程序显示信息时，*elm* 针对每个字符调用一库函数来确定该字符是否可打印的。缺省地，这个函数将只能识别 ASCII 字符为可打印字符，并将所有其它字符显示为“^?”。你可以通过设置环境变量 *LC_CTYPE* 为 ISO-8859-1 来克服这个问题，该变量告知库函数接受 Latin-1 字符为可打印的。库从 *libc-4.5.8* 起支持这个和其它特性。

当发送含有 ISO-8859-1 中特殊字符的信息时，你应该确信在 *elm.rc* 文件中设置了起码两个变量：

```
charset = iso-8859-1
textencoding = 8bit
```

这使得 *elm* 在邮件标题中报告字符集是 ISO-8859-1，并将它作为 8 比特值来发送（缺省的是将所有字符剥成 7 比特）。

当然，这些选项中的任何一个除了可以在全局 *elm.rc* 文件中设置以外，都可以在私有 *elmrc* 文件中设置。

注释

[1] 通常习惯上会给邮件消息附加上一个签名（*signature*）或*.sig*，通常包含有关作者的信息，以及一个笑话或者一座右铭。它与邮件消息之间相隔一含有“--”的一行。

[2] 这是因为磁盘空间的分配通常是以 1024 字节的块为单位的。所以，即使一个最多 400 字节的消息也要吃掉整整一 KB 的空间。

[3] 当从一个 RFC 822 环境试图到达一个 DECnet 地址时，你可以使用**“host::user”@relay**，这里 relay 是一个已知的 Internet-DECnet 中继。

[4] 在 UUCP 映射计划登记的站点的映射是通过新闻组 **comp.mail.maps** 发布的；其他的组织也会公布他们网络的映射。

[5] 它们定期地投递到 **news.lists.ps-maps**。注意，它们非常巨大。

第十四章 配置和运行 *smail*

这一章将给你一个设置 *smail* 的快速入门，以及它所提供的功能概述。尽管 *smail* 的行为在很大程度上与 *sendmail* 相兼容，但是它们的配置文件却是完全不同的。

主要的配置文件是*/usr/lib/smail/config*。你一定要编辑这个文件以反映你的站点的特定的值。如果你只是一个 UUCP 的末端站点（leaf site），那么相应地你很少需要改动的。配置路由选择和传输选项的其它文件当然也可以使用；这些文件也将概要地论及。

缺省地，*smail* 会立刻处理并分发所有入站的邮件。如果你有相应较大的流量，你可以先让 *smail* 将信息收集到所谓的队列（*queue*）中，并且仅在一定的间隔期间来处理它们。

当在 TCP/IP 网络上处理邮件时，*smail* 会以后台模式（daemon mode）频繁地运行：在系统引导启动时，它是从 *rc.inet2* 被调用的，并且将自己置入后台以等待 SMTP 端口（通常是端口 25）进入的 TCP 连接。当你可能会遇上较大信息流量时，这是非常有用的，因为 *smail* 不会针对每个入站连接而立即运行的。另外一种方法是让 *inetd* 来管理 SMTP 端口，并且每当这个端口有连接时，由它来调用 *smail*。

smail 有许多标志以控制自己的行为；在此详细讨论它们对你并不会有太大的帮助。幸运的是，*smail* 支持一些标准的操作模式，当你通过一特定的命令名调用 *smail* 时这些模式会开启，如 *rmail* 和 *smtpd*。通常，这些别名本身是对 smail 执行文件的符号连接。在讨论 smail 的各种特性时我们会遇到其中大多数特性。

在所有环境下，你应该有两个到 *smail* 的连接；它们是*/usr/bin/rmail* 和*/usr/sbin/sendmail*。[1] 当你使用一个用户代理程序（如 *elm*）撰写和发送一个邮件信息时，该邮件信息将输送给 *rmail* 去进行投递，而接收者列表要在命令行上给出。对于通过 UUCP 接收的邮件也会有同样的情况。然而，*elm* 的某些版本会调用*/usr/sbin/sendmail* 而不是 *rmail*，所以你需要它们两者。例如，如果你将 smail 放在/usr/local/bin 中，那么在 shell 提示下键入下面两行：

```
# ln -s /usr/local/bin/smail /usr/bin/rmail
# ln -s /usr/local/bin/smail /usr/sbin/sendmail
```

如果你想更深入地研究配置 *smail* 的细节，请参阅手册页 *smail(1)*和 *smail(5)*。如果它没有包括在你中意的 Linux 发行版本中，你可以从 smail 的源程序中得到。

14.1 UUCP 的设置

要想在只有 UUCP 的环境下使用 *smail*，基本的安装过程是非常简单的。首先，你必须确信你已经有了上面所提到的两个符号连接 *rmail* 和 *sendmail*。如果你还希望从其它站点接收到 SMTP 批处理信息，你也需要设定 *rsmtp* 为一个到 *smail* 的连接。

在 Vince Skahan 的 *smail* 发行版中，你会找到一个样本配置文件。它被命名为 *config.sample* 并存在于*/usr/lib/smail* 中。你必须拷贝它到 *config* 并且编辑它以适用于你的站点。

假设你的站点名称是 *swim.twobirds.com*，并在 UUCP 映射中以 *swim* 登记注册。你的灵敏主机是 *ulysses*。此时你的 *config* 文件应该看上去象这样的：

```
#
```

```
# 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
```

第一条语句告知 *smail* 有关你的站点所属的域。在这里插入它们的名字并用冒号分开。如果你的站点名在 UUCP 映射中注册过，那么你也应该加上 *uucp*。在处理一个邮件消息时，*smail* 使用 *hostname(2)*系统调用来确定你的主机的名字，并且将接收者的地址和这个主机名作比较检查，依次添加上这个列表中的所有名字。如果该地址与这些名字或非正规主机名中的任何一个相匹配时，接收者就被认为是本地的，并且 *smail* 将试图将这个消息投递给本地主机上的一个用户或别名。否则的话，接收者被认为是远程的，并开始尝试投递到目的主机去。

visible_name 应该含有单个、用于出站邮件上的你的站点的全资域名。当在所有出站邮件上生成发送者的地址时，将使用这个名字。你必须确信使用了一个 *smail* 能够识别为代表本地主机的名字（也即，列于 *visible_domain* 属性中域之一的主机名）。否则的话，对你的邮件作出的回复将弹出你的站点。

最后一条语句设置用于灵敏主机路由选择的路径（已在 13.4 节中作了描述）。对于这个样本设置，*smail* 会把所有到远程地址的邮件转发给灵敏主机。由于消息将通过 UUCP 来投递，该属性必须指定一个你的 UUCP 软件认识的系统。请参阅第 12 章的让站点为 UUCP 知晓。

还有一个上面文件中用到的选项我们至今还没给出解释；这就是 *uucp_name*。使用这个选项的理由是：默认地，*smail* 使用 *hostname(2)*返回的值给 UUCP 方面使用，比如在 *From_*标题行中给出的返回路径。如果你的主机名没有在 UUCP 映射计划组注册过，你应该告诉 *smail* 另外使用你的全资域名取代之。[2] 这可以在 *config* 文件中加入 *uucp_name* 选项来做到。

在*/usr/lib/smail* 中还有一个文件，叫 *paths.sample*。它是 *paths* 文件的一个例子。然而，除非你有到多于一个站点的邮件连接，否则的话你并不需要它。如果你确实有多个邮件连接，你就需要自己写一个这个文件，或者从 Usenet 映射中生成一个。*paths* 文件将在本章稍后讨论。

14.2 为局域网（LAN）进行设置

如果你正运行一个站点，具有两台或以上的主机构成了一个 LAN，那么你就必须指定一台主机来处理你的 UUCP 与外部世界的连接。在你的 LAN 上的主机之间，你很可能会用 SMTP 在 TCP/IP 上交换邮件。假设我们现在再次回到虚拟酿酒厂，而且 **vstout** 被设置成为 UUCP 的网关。

在一个连网的环境中，最好将所有用户的邮件箱放在单个文件系统上，这个文件系统在所有其它主机上可以以 NFS 加载的。这允许用户从一台机器换到另一台机器，而不需要将他们的邮件带来带去（或者更糟的是，每天早晨检查三四台机器看看有没有新到的邮件）。因此你也希望发信者的地址是与编写邮件的机器无关的。在发信者的地址中一直使用域名而非主机名，就是一个非常实际有用的方法。例如，Janet 用户将指定地址为**janet@vbrew.com**而不是**janet@vale.vbrew.com**。下面我们将解释如何让服务器将域名识别为一个你的站点的有效名字。

将所有邮件箱保持在一台中央主机上的另一种不同的方法是使用 POP 或者 IMAP。POP 代表*邮局协议*（*Post Office Protocol*）它能让用户通过一简单 TCP/IP 连接访问他们的邮件箱。IMAP，*交互式邮件访问协议*（*Interactive Mail Access Protocol*），与 POP 类似，但更通用。IMAP 和 POP 的客户以及服务器程序都已经移植到 Linux 上，可以从 **sunsite.unc.edu** 中的*/pub/Linux/system/Network* 下取得。

14.2.1 编写配置文件

酿酒厂的配置是按如下方式工作的：除了邮件服务器 **vstout** 本身的所有主机使用灵敏主机路由选择将所有出站邮件传递给服务器。**vstout** 本身则将所有出站邮件发送给用以传递所有酿酒厂邮件的真正灵敏主机；这个主机叫作 **moria**。

除了 **vstout**，所有其它主机的标准 *config* 文件看上去象这样：

```
#
# Our domain:
visible_domain=vbrew.com
#
# Whaat we name ourselves
visible_name=vbrew.com
#
# Smart-host routing: via SMTP to vstout
smart_path=vstout
smart_transport=smtp
```

这同我们用于 UUCP 站点的非常相似。主要的不同之处是用于发送邮件到灵敏主机的传输是 SMTP。*visible_domain* 属性使得 *smail* 在所有出站邮件上使用域名来代替本地主机名。

在 UUCP 邮件网关 **vstout** 上，*config* 文件看上去稍有不同：

```
#
# Our hostnames:
hostnames=vbrew.com:vstout.vbrew.com:vstout
#
# What we name ourselevs
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
```

这个 *config* 文件使用了一个不同的方案来告诉 *smail* 本地主机叫什么。不是给它一个域列表并让它使用一个系统调用来找出主机名，而是明确地给出了一个列表。上面的配置本身含有*全资和自由的主机名*（fully qualified and the unqulified hostname），以及域名。这使得 *smail* 能将**janet@vbrew.com**识别为一个本地地址，并将消息投递给 **janet**。

auth_domains 变量命名一个域，对于这个域，**vstout** 被认为是授权的。也即，在 *smail* 接收到任何到地址 *host***.vbrew.com** 的邮件时，如果其中的 *host* 不是任何本地机器的名字，那么它就会拒绝这个邮件消息并将它返回给发信者。如果没有这个条目，那么任何这样的消息都将被送到灵敏主机，而灵敏主机将把它返回给 **vstout**，一直这样循环下去直到该消息由于超过最大跳数而被丢弃。

14.2.2 运行 *smail*

首先，你要决定是否将 *smail* 作为一个独立的后台程序（daemon）运行，或者是让 *inetd* 管理 SMTP 端口并且仅当某些客户请求一个 SMTP 连接时才调用 *smail*。通常，在邮件服务器上，你将更喜欢后台程序操作的方式，因为这样不会对每个单独的连接不停地产生 *smail* 子进程而增加机器的负荷。由于邮件服务器也将大多数入站邮件直接投递给用户，在许多其它主机上你将选择 *inetd* 的操作方式。

对每台单独的主机，不管你选择了哪种操作模式，你必须确信在*/etc/services* 文件中有下面的条目：

```
smtp          25/tcp        # Simple Mail Transfer Protocol
```

这定义了 *smail* 用于 SMTP 连接的 TCP 端口号。25 是定义于 Assigned Numbers RFC 中的标准端口号。

当运行于后台模式时，*smail* 将把自己放入后台，并且等待发生在 SMTP 端口的连接。当出现一个连接时，它用对等进程产生并引导一个 SMTP 对话。*smail* 后台程序通常是使用下面命令通过从 *rc.inet2* 脚本中被调用而启动的：

```
/usr/local/bin/smail –bd –q15m
```

`-bd` 标志打开后台模式，`-q15m` 使得它每隔 15 分钟就处理一次堆积在队列中的消息。

如果你想另外使用 inetd 方式，那么你的/etc/inetd.conf 文件中应该含有象这样的一行：

```
smtp       stream  tcp nowait  root  /usr/sbin/smtpd smtpd
```

smtpd 应该是一个到 *smail* 执行文件的符号链接。记住，在作过改变之后你必须给它发送一个 *HUP* 信号让 *inetd* 重读 *inetd.conf*。

后台模式（Daemon mode）和 inetd 模式是相互排斥的。如果你以后台模式运行了 *smail*，你应该确信注释掉了 *inetd.conf* 中任何 *smtp* 服务的行。同样地，当让 *inetd* 管理 *smail* 时，请确信 *rc.inet2* 没有启动 *smail* 后台程序。

14.3 如果你没有顺利完成____

如果你在安装时碰到了问题，有几个特征可能可以帮助你找出问题的根源。第一个需要检查的地方是 *samil* 的日志文件。它们是放在*/var/spool/smail/log* 中的，名字分别为 *logfile* 和 *paniclog*。前者列出了所有事项，而后者仅用于存储与配置错误等相关方面的出错信息。

logfile 中的典型条目看上去象这样的：

```
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
```

这显示出从 **root** 到**root@vstout.vbrew.com**的消息已经通过 SMTP 正确地投递到主机 **vstout** 上去了。

smail 不能投递的消息在日志文件中也产生类似的条目，但是用一错误信息取代了 `delivered` 部分：

```
04/24/94 07:12:04: [mOpuwU8-00023UB] received
|          from: root
|       program: sendmail
|          size: 1468 bytes
04/24/94 07:12:04: [mOpuwU8-00023UB] root@vstout.vbrew.com … deferred
 (ERR_148) transport smtp: connect: Connection refused
```

上面的错误对于*smail* 已能正确地识别出消息应该投递到**vstout** 但是却连不到**vstout** 上的 SMTP 服务的情况是一种典型的错误。如果发生了这种情况，你或者是有个配置问题，或者是因为你的 *smail* 执行程序不支持 TCP。

这个问题并没有人们想象的那么特殊。存在着很多已经编译好的 smail 执行程序都不支持 TCP/IP 连网，甚至是在某些 Linux 发行版中。如果你碰到的是这种情况，你就必须自己编译 smail。如果已经安装了 smail，你可以通过 telnet 远程登录到你的机器的 SMTP 端口来检查它是否支持 TCP 连网。到 SMTP 服务的一个成功连接如下面显示的（你的输入用*这样表示的*）：

```
$ telnet localhost smtp
Trying 127.0.0.1…
Connected to localhost.
Escape character is ‘^]’.
220 monad.swb.de Smail3.1.28 #6 ready at Sun, 23 Jan 94 19:26 MET
QUIT
```

```
221 monad.swb.de closing connection
```

如果这个测试没有产生 SMTP 标题（以 220 码开头的行），首先在你自己进行 *smail* 编译之前确信你的配置是确实正确的，这在下面讨论。

如果你使用 *smail* 遇到了一个问题，而且从 *smail* 产生的出错信息不能确定问题的所在，那么你可以开启调试信息功能。你可以使用-d 标志来做到，后跟一个指定调试信息长度的数字选项（在标志和数字之前可以不加空格）。*smail* 将在屏幕上打印出它的操作报告，这可能会给你更多有关出错的线索。

[不知道，…人们可能不会觉得这有趣：] 如果一点也没帮助，你可能想要通过在命令行上给出 -bR 选项以 Rogue 模式调用 *smail*。关于这个选项手册页说：“往敌方域发送巨量的邮件信息，以及 RFC 标准的资料。试图让它下到协议层 26 然后返回。”尽管这个选项不能解决你的问题，它可能为你提供某些舒适度和安慰。[3]

14.3.1 编译 *smail*

如果你确实认为你的 *smail* 还没有 TCP 网络支持，你就必须取得源码。如果你是从 CD-ROM 中得到的，那么一般它是包括在你的发行版中的，否则的话你可以从网上 FTP 站点获得。[4]

当编译 *smail* 时，你最好用从 Vince Skahan 的 *newspak* 发行版的配置文件集开始。为了用 TCP 网络驱动程序编译，你必须设置 *conf/EDITME* 文件中的 *DRIVER_CONFIGURATION* 宏为 *bsd-network* 或 *arpa-network*。前者适用于 LAN 的安装，但是 Internet 需要 *arpa-network*。这两者的不同之处在于后者是能够识别 MX 记录的 **BIND** 服务专用的驱动程序，而前者却不是。

14.4 邮件投递模式

正如上面提到的，*smail* 能够立刻将消息投递出去，或者排入队列稍后再作处理。如果你选择将消息排入队列，*smail* 将把所有的邮件存储在*/var/spool/smail* 目录下。*smail* 不会去处理这些邮件直到明确地告知它去处理（这也称为“运行该队列（running the queue）”）。

通过在 *config* 文件中设置 *delivery_mode* 属性为 *foreground*、*background* 或 *queued*，你可以选择三种投递模式之一。这将投递模式选择为前台的（对入站消息进行立即处理）、后台的（消息将有接收进程的子进程进行投递，而父进程则会在派生了子进程后立即退出）、以及排队的。如果 *config* 文件中设置了布尔变量 *queue_only*，那么不管这个选项怎么设置，入站邮件将总是被排入队列中。

如果你打开了排队选项，你必须确信队列会被定期检查；一般是每个 10 或 15 分钟。如果你以 daemon 模式运行 *smail*，你就必须在命令行上加上选项-q10m 用以每隔 10 分钟处理一次队列。另外，你可以以这些间隔时间从 *cron* 来调用 *runq*。*runq* 应该是一个到 *smail* 的链接。

你可以通过用选项-bp 来调用 *smail* 来显示当前的邮件队列。同样地，你可以作一个到 *smail* 的链接 *mailq*，并调用 *mailq*：

```
$ mailq –v
mOpvB1r-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
```

这显示了消息队列中的一个消息。事项日志（仅在你给 *mailq* 一个`-v`选项时才会显示）可能会给出为什么还在等待投递的额外原因。如果至今都没有试图投递这个消息，那么将不显示事项日志。

即使你不使用队列，那么当 smail 发现由于短暂的原因立即投递失败时偶然也会将消息放入队列中。对于 SMTP 连接来说，这可能是一个不可达的主机；但是当文件系统满时消息也会被延期投递的。因此你应该放置一个队列并且每隔 1 小时左右运行处理队列一次（使用 *runq*），否则的话，任何延期的消息都将永远待在队列中了。

14.5 各种其它 *config* 选项

还有许多可以在 *config* 文件中设置的选项，这些选项尽管很有用，但对运行 *smail* 并不是必须的，所以我们在这里不讨论它们。而是仅提及几个你可能由于某种原因会使用的选项：

error_copy_postmaster

如果设置了这个布尔变量，任何错误都会产生一条给邮件管理者的消息。通常，这样做只是为了检测配置中的错误。通过将它放入 config 文件，并在前面放一个加号（+）来开启这个功能。

max_hop_count

如果一个消息的跳数（也即，经过的主机数目）等于或超过这个数字时，在远程投递消息的企图将产生一个错误信息并被返回给发送者。这是用于防止消息不停地循环传输。跳数的计数一般是从邮件标题中的 *Received:*字段的个数计算出的，但是也可以在命令行上使用-h 选项手工设置的。

这个变量的缺省值是 20。

postmaster 邮件管理者的地址。如果 **Postmaster** 不能解析成一个有效的本地地址，那么作为最后的手段。缺省值是 **root**。

14.6 消息（报文）路由选择和投递

smail 将邮件的投递分成了三个不同的任务，路由器、导向器和传输器模块（router，director，transport module）。

路由器模块用于解析所有远程地址，确定消息将被送到下一台哪台主机，以及必须使用哪种传输器。根据链接的特性，将使用不同的传输器，比如 UUCP 或 SMTP。

本地地址将给予导向器任务，用于解析任何转发和别名使用。例如，地址可能是一别名或一个邮件列表，或者用户可能想将她的邮件转发到另外一个地址。如果所得到的地址是远程的，它将被传送给路由器模块进行额外的路由选择，否则的话它将进行本地投递传输。到目前为止，最普通的情况是投递到一个邮件箱中，但消息也可能被传送给一个命令，或者是附加到某些任意的文件中。

最后，传输器模块负责选择投递的方法。它试图投递这个消息，并且如果失败就将消息弹回，或将消息延迟投递。

使用 *smail*，在配置这些任务时你有很大的自由度。对于每个任务，有许多驱动程序，你可以从中选取你所需的。要使用几个文件来向 *smail* 描述它们，它们是位于*/usr/lib/smail* 中的 *routers*、*directors* 和 *transports*。如果这些文件不存在的话，那么将采用合理的默认值，这些默认值对于使用 SMTP 或 UUCP 传输的许多站点来讲都是合适的。如果你想改变 *smail* 的路由选择策略，或者要修改某个传输，你应该从 *smail* 的源发行版中取得样本文件[5]，将样本文件拷贝到*/usr/lib/smail* 目录下，并且按照你的需要来修改它们。在附录 B 中也给出了样本配置文件。

14.7 消息（报文）的路由选择

当给出一个消息，*smail* 首先会检查它的目的地是否是本地的，还是一远程站点。如果目标主机地址是 *config* 中配置的本地主机名之一，就将消息传送到导向器模块。否则的话，*smail* 将把目的地址送到一系列路由器驱动程序以找出将消息转发到的主机。可以在 *routers* 文件中对它们进行描述；如果这个文件不存在的话，就会使用一族缺省值。

接下来目的主机（名）将传给所有的路由器，并且将选择找到最为确定路由的路由器。考虑一个到**joe@foo.bar.com**的消息。那么，某个路由器可能知道一个到 **bar.com** 域中所有主机去的缺省路由，而另一个路由器却有 **foo.bar.com** 本身的信息。由于后者更为确切，就会选择后者。如果有两个路由器都提供了“最佳匹配”，那么就会选择 *routers* 文件中先出现的那个。

现在，这个路由器指定所使用的传输器，比如 UUCP，并且生成一个新的目的地址。这个新的地址与主机（名）一起传给这个传输器以将消息转发过去。在上面的例子中，*smail* 可能会发现通过使用路径 **ernie!bert** 的 UUCP 可以到达 **foo.bar.com**。此时它会产生一个新的目标 **bert!foo.bar.com!user**，并且让 UUCP 传输器将此目标用作信封地址传送到 **ernie**。

当使用缺省设置时，就有下列路由器：

- 如果目的主机地址可以使用 *gethostbyname(3)*或者 *gethostbyaddr(3)*库调用解析出来，那么消息就会用 SMTP 进行投递。唯一的例外是，如果发现该地址引用（涉及）到本地主机，那么消息也会被传到导向器模块。

  smail 也能识别点分四组写的 IP 地址作为一个合法的主机名，只要它们能够使用 *gethostbyaddr(3)*调用来解析。例如，**scrooge@[149.76.12.4]**将是一个有效的地址，尽管在 **quark.physics.groucho.edu** 上这是一个非常与众不同的邮件地址。

  如果你的机器是在 Internet 上的，那么这些路由器就不是你所要的，因为它们不支持 MX 记录。对于这种情况请看下面。

- 如果路径别名数据库*/usr/lib/smail/paths* 存在，那么 *smail* 将试图在这个文件中查找目标主机（减去任何**.uucp** 的结尾）。邮递到这个路由器匹配的地址去的邮件将使用 UUCP 和数据库中找到的路径来投递。

- 主机地址（去除任何**.uucp** 结尾）将与 *uuname* 命令的输出相比较，以检查目标主机实际上是否是一个 UUCP 邻居。如果真是这种情况，那么消息将使用 UUCP 传输器投递。

- 如果地址用上面任何一个路由器都不能匹配的话，那么它将被投递到灵敏主机。到灵敏主机的路径以及所使用的传输器是在 *config* 文件中设置。

这些缺省设置对于许多简单设置来说是可工作的，但是如果路由选择的要求稍微复杂一些时就会失败。如果你面临下面描述的任何问题，那么你就需要安装你自己的 *routers* 文件以覆盖缺省的文件。附录 B 中给出了一个样本 *routers* 文件，你可以以它作为开始。某些 Linux 发行版也携带有一族配置文件，它们被编辑成克服这些困难的。

当你的主机有着双重的拨号 IP 和 UUCP 链接时，或许会产生最糟糕的问题。此时在你的 hosts 文件中有着你仅通过 SLIP 链接很少论及的主机名，所以 *smail* 将试图通过 SMTP 为这些主机投递任何邮件。这通常并不是你想要的，因为即使 SLIP 链接是定期激活的，SMTP 要比通过 UUCP 发送邮件慢得多。对于使用缺省设置的情况，没有任何方法来逃避 *smail* 的。

通过在查询解析器之前让 *smail* 检查 *paths* 文件，你就可以避免这个问题，并且将所有你想迫使 UUCP 投递到的主机放入 *paths* 文件中。如果你再也不想通过 SMTP 发送任何消息，那么你也可以完全注释掉基于解析器的路由器。

另一个问题是缺省的设置并不是为真正的 Internet 邮件路由选择提供的，因为基于解析器的路由器没有估计（evaluate）MX 记录。为了启用对 Internet 邮件路由选择的全面支持，注释掉这个路由器，并且去掉对使用 BIND 的路由器的注释。然而，包含在某些 Linux 发行版中的 *samil* 执行程序并没有编译进对 BIND 的支持。如果你启用 BIND，但却在 *paniclog* 文件中得到一个信息指出“路由器 inet_hosts：驱动程序 bind 没有找到”（“router inet_hosts: driver bind not found”），那么你就必须取得源代码并重新编译 *smail*（见上面 14.2 节）。

最后，使用 *uuname* 驱动程序一般并不是一个好主意。首先是当你没有安装 UUCP 时，它将会产生一个配置错误，因为会找不到 *uuname* 命令。其次是当你有比实际有邮件链接多的站点列于你的 UUCP *Systems* 文件时。这些可能是你仅仅进行 news 交换的站点、或者是你偶尔使用匿名 UUCP 下载文件的站点，但除此以外没有任何交通流量了。

为了解决第一个问题，你可以用只做一简单 *exit* 0 的 shell 脚本来替换 *uuname*。然而，更加常规的解决方案是编辑 routers 文件并且完全删除这个驱动程序。

14.7.1 *paths* 数据库

smail 期望在*/usr/lib/smail* 下的 *paths* 文件中找到路径别名数据库（pathalias database）。而这个文件是可选的，所以如果你完全不需要执行任何路径别名路由选择的话，只需简单地删除任何存在的 *paths* 文件。

paths 必须是一个排过序的 ASCII 文件，包含有映射目的站点名到 UUCP bang paths 的条目。因为 *smail* 使用二叉树搜索来查找一个站点，所以这个文件必须是排过序的。文件中不得含有注释，并且站点名与路径之间必须用制表符 TAB 分开。路径别名数据库已在第 13 章中详细讨论过。

如果这个文件是你自己编写的，你应该确信其中包含了一个站点的所有合法的名称。例如，如果一个站点已知有无格式 UUCP 名称和一个全资域名（fully qualified domain name），那么你就必须为这两个名字都加入一个条目。可以通过将此文件传给 *sort(1)*命令来排序这个文件。

然而，如果你的站点仅仅是个页站点（a leaf site），那么就完全不需要 *paths* 文件；只要在你的 config 文件中设置灵敏主机属性，并让你的邮件馈送者来处理所有的路由选择。

14.8 往本地地址投递消息（报文）

非常一般地，一个本地地址只是一个用户的登录名，在这种情况下，消息被投递到她的邮件箱中，*/var/spool/mail/**user***。其它的情况包括别名和邮件列表名、以及用户作的邮件转发。在这些情况下，本地地址被扩展成一个新的地址列表，这个新地址列表或者是本地的，也可能是远程的。

除了这些“普通的“地址以外，*smail* 还可以处理其它本地消息目的的类型，象文件名和管道命令。这些本身并不是地址，所以你不可以将邮件发送到**/etc/passwd@vbrew.com**这样的地址去；仅当它们是从转发和别名文件中取得时，它们才是有效的。

一个*文件名*（*file name*）是以斜杠（/）或破折号（~）开始的任何字符串。后者表示用户的主目录，而且仅当文件名是从一个*.forward* 文件或从一个在邮件箱（见下面）的转发条目中取得时才是可能的。当投递到一个文件时，*smail* 将消息附加到这个文件上，必要时就创建这个文件。

一个 *pipe* 命令可以是前面加有管道符号（|）的任何 UN*X 命令。这使得 *smail* 将命令及其参数传递给 shell，但是不包括前导‘|’。消息本身在标准输入上被馈送给这个命令。

例如，为了将邮件列表送入一个本地新闻组（newsgroup）中，你可以使用一个名为 *gateit* 的 shell 脚本，并且设置一个本地别名，这个别名会使用“*—gateit*”从这个邮件列表中投递所有的消息。

如果引用中包括有空格，就必须使用双引号包住。由于所涉及的安全因素，如果地址是以某些不可靠的方法获得的（例如，如果从中获取地址的别名文件是任何人可写的），就要避免执行这个命令。

14.8.1 本地用户

一个本地地址最普通的情况是表示一个用户的邮件箱。这个邮件箱位于*/var/spool/mail* 中并且具有这个用户的名字。它是属于这个用户的，并有组名 **mail** 和属性 600。如果它不存在，*smail* 就会创建它。

注意，尽管*/var/spool/mail* 目前是放置邮件箱文件的标准位置，某些邮件软件可能编译进了不同的路径，例如*/usr/spool/mail*。如果往你机器上的用户进行投递一直会失败，你应该试着作一个到 */var/spool/mail* 的符号链接。

smail 要求有两个地址存在：**MAILER-DAEMON** 和 **Postmaster**。当为一个不能投递的邮件产生一个反弹消息时，一个副本将被发送给 **postmaster** 帐号以作检查（以防万一这可能是由于配置方面的问题）。MAILER-DAEMON 是在反弹消息上作为发送者地址的。

如果这些地址在你的系统上并没有有效的帐号，*smail* 分别隐含地将 MAILER-DAEMON 映射到 **postmaster**、**postmaster** 到 **root**。通常你应该通过为 postmaster 帐号建立别名来覆盖这个映射，将别名建立到负责维护邮件软件的用户去。

14.8.2 转发

用户可以使用 smail 支持的两种方法之一通过将邮件转发到另外一个地址来重定向她的邮件。一种选择是将

```
Forward to recipient, …
```

放在她的邮件箱文件的第一行。这会把所有入站的邮件发送到指定的接收者列表去。另一种方法是，她可以在她的主目录中建立一个*.forward* 文件，该文件含有用逗号分开的接收者列表。对于各类转发，该文件的各行将被读取并作出说明。

注意，可以使用任何类型的地址。因此，对于休假的*.forward* 文件的一个实际例子可以是

```
janet, "|vacation"
```

第一个地址不管怎样将会把入站消息投递到 `janet` 的邮件箱中，而 vacation 命令则会给发送者返回一个简短的通告。

14.8.3 别名文件

samil 能够处理那些与 Berkeley 的 *sendmail* 兼容的别名文件。在别名文件中条目可以有如下的形式

```
alias: recipients
```

recipients 是一个用逗号分开的地址列表，它将被别名替代。接收者列表可以续行如果下一行以一个制表符开始。

有一个特殊的属性，允许 *smail* 从别名文件中来处理邮件列表：如果你指定"`:include:`*filename*"作为接收者，*smail* 将读取指定的文件，并且替换它的内容作为一个接收者的列表。

主要的别名文件是*/usr/lib/aliases*。如果你选择使得这个文件是人人可写的，smail 将不会把任何消息投递给该文件给出的 shell 命令。一个例子文件见如下所示：

```
# 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"
```

在投递给从 *aliases* 文件中产生的一个地址去时如果出现了一个错误，*smail* 将试图把出错消息

的一个拷贝发送到“别名拥有者”（“alias owner”）。例如，在把一个消息投递给 **development** 邮件列表时，如果投递给 **biff** 失败了，那么一个出错消息的拷贝将被邮递给发送者、以及 **postmaster** 和 **owner-development**。如果拥有者的地址不存在，那么就不会生成额外的出错消息。

当投递给文件或者当调用 *aliases* 文件中给出的程序时，*smail* 将成为 **nobody** 用户以避免任何安全方面的问题。特别是当投递给文件时，这将是非常麻烦的事。例如，在上面给出的文件中，**nobody** 必须拥有这个日志文件并且对其可写，否则向他们进行的投递将会失败。

14.8.4 邮件列表

除了使用 *aliases* 文件，邮件列表也可以通过*/usr/lib/smail/lists* 目录中的文件来管理。一个命名为 *nag-bugs* 的邮件列表是由 *lists/nag-bugs* 文件描述的，它应该含有用逗号分开的组员的地址。列表可以在多行上给出，注释使用 hash 符号来引出。

对于每个邮件列表，应该存在一命名为 **owner-***listname* 的用户（或别名）；在解析一个地址时发生的任何错误会向这个用户报告。这个地址在所有出站消息的 `Sender:` 标题字段中也用作发送者的地址。

14.9 基于 UUCP 的传输器

有许多利用 UUCP 程序集的编译进 *smail* 的传输器。在一个 UUCP 环境中，消息通常通过在下一台主机上调用 *rmail* 来传递的，并需要在标准输入上给 *rmail* 消息和在命令行上给 *rmail* 信封地址。在你的主机上，*rmail* 应该是一个到 *smail* 命令的链接。

当把一个消息递给 UUCP 传输器时，smail 将目标地址转换为一个 UUCP bang 路径。例如，**user@host**将被转换成 **host!user**。任何存在的‘%’地址操作符都将保留，所以**user%host@gateway** 将转换成 **gateway!user%host**。然而，*smail* 自己不会生成这样的地址。

另一方面，*smail* 可以通过 UUCP 发送和接收 BSMTP 批处理。使用 BSMTP，一个或多个消息被包裹成单个批处理，这个批处理含有本地邮件程序在一个实际 SMTP 连接被建立时将发出的命令。BSMTP 频繁地使用于存储-转发（例如，基于 UUCP 的）网络以节约磁盘空间。附录 B 中的样本 *transports* 文件含有一个在一队列目录中能够生成部分 BSMTP 批处理的配有 *bsmtp* 的传输器。以后，使用一个加入适当的 *HELO* 和 *QUIT* 命令的 shell 脚本，它们必须被合并入最终的批处理中。

对于指定的 UUCP 连接，为了启用 *bsmtp* 传输器，你必须使用所谓的 *method* 文件（详细信息请参阅 *smail(5)*手册页）。如果你仅有一个 UUCP 链接并且使用灵敏主机路由器，你可以通过将配置变量 *smart_transport* 设置成 *bsmtp* 而非 *uux* 来启用发送 SMTP 批处理。

要在 UUCP 上接收 SMTP 批处理，你必须确信你有*解批处理*（unbatching）命令，远程站点会将自己的批处理发送到这个命令。如果远程站点也使用 *smail*，那么你需要使得 *rsmtp* 为一个到 *smail* 的链接。如果远程站点运行 *sendmail*，那么你应该额外地安装一个名为*/usr/bin/bsmtp* 的 shell 脚本，这个脚本执行一个简单的“*exec rsmtp*”（符号链接是不能工作的）。

14.10 基于 SMTP 的传输器

目前 *smail* 支持一个 SMTP 驱动程序以在 TCP 连接上投递邮件。[6] 在主机名指定为一个能够使用网络软件解析的全资域名，或是用方括号括住的点分四组表示法时，在一单个主机上将消息投递到任何数量的地址去是可以的。通常，由任何 BIND、*gethostbyname(3)*、或 *gethostbyaddr(3)*路由器驱动程序解析的地址都将被投递给 SMTP 传输器。

SMTP 驱动程序通过列于*/etc/services* 中的 *smtp* 端口将试图立即连接至远程主机。如果它不能到达，或者连接超时，那么将在以后时间再次尝试投递。

在 Internet 上的投递，要求到目的主机的路由指定为第 13 章中描述的 *route-addr* 格式，而不是作为一个 bang 路径。[7] 因此，smail 将转换 **user%host@gateway**（这里 **gateway** 是通过 **host1!host2!host3** 到达的）成为源-路由地址**<@host2,@host3:user%host@gateway>**，这将被作为消息的信封地址发送到 **host1**。要开启这些转换（与内建的 BIND 驱动程序一起），你必须编辑 *transports* 文件中的有关 *smtp* 驱动程序的条目。附录 B 中给出了一个样本 *transports* 文件。

14.11 主机名限定（qualification）

有时找出在发送者或接收者地址中指定的无限定的主机名（也即，那些没有域名的主机名）是值得做的。例如，当在两个网络之间进行网关连接时，其中，一个网络要求全资域名。在一个 Internet-UUCP 中继上，缺省地无限定主机名应该被映射到 **uucp** 域。除此之外的其它地址修改都是不可靠的。

/usr/lib/smail/qualify 文件告诉 *smail* 哪个域名附加到哪个主机名上。*qualify* 文件中的条目由一个从第一列开始的主机名、和后跟的域名组成。以 hash 符号作为首个非空格字符的行被看作是注释行。各条目是根据它出现的先后次序进行搜索的。

如果 *qualify* 文件不存在，那么就完全不会执行任何主机名限定操作。

特殊的主机名*****与任何主机名匹配，因而允许你在进入一个缺省的域之前映射所有未提及的主机。它应该仅用于最后一个条目。

在虚拟酿酒厂，所有的主机已被设置成在发送者地址中使用全资域名。无限定接收者的地址被认为是在 **uucp** 域中的，所以在 *qualify* 文件中只需要一单个条目。

```
# /usr/lib/smail/qualify, last changed Feb 12, 1994 by janet
#
*         uucp
```

注释

[1] 依照 Linux 文件系统标准，这是 *sendmail* 新的标准位置。另一个常用的位置是*/usr/lib*。

[2] 理由是：假设你的主机名是 *monad*，但是并没有在映射中注册。然而，在映射中有一个站点叫做 monad，所以每个到 *monad!root* 的邮件、甚至是从你的直接 UUCP 邻居来得邮件，都将被送到另一个 *monad* 去了。这对任何人来讲都是令人讨厌的。

[3] 如果你的情绪很坏时就不要使用它。

[4] 如果你是从销售商那里随同 Linux 发行版买来的，那么依照 *smail* 的拷贝条件，你就有资格“以最低的（极小的）运输费”得到源码。

[5] 可以从源目录的 samples/generic 下面找到缺省的配置文件。

[6] 作者称这种支持为“简单的”。对于 smail 的将来版本，他们预告了一个完整的能更有效处理的后端程序。

[7] 然而，在 Internet 中使用路由器是完全不赞成的。取而代之应该使用全资域名。

第十五章 Sendmail+IDA

15.1 Sendmail+IDA 概述

有人曾说过只有编辑过一个 *sendmail.cf* 文件，你才是一个*真正的* Unix 系统管理员。也有人说如果你试图这样编辑两次，那你一定是疯了☺

Sendmail 是一个难以置信的功能强大的程序。对大多数人来说它也是不可思义的难学。具有 792 页长的权威性参考资料（*Sendmail*，由 O'Reilly and Associates 出版）的任何程序不可辩驳地会吓跑大多数人。

Sendmail+IDA 则不同，它删除了对编辑总是隐晦的 *sendmail.cf* 文件的需要并且允许管理人员通过相对简单易懂的称为 *tables* 的支持文件来定义与站点有关的路由选择和寻址配置。切换到 sendmail+IDA 上，可以节约你的许多工作时间和精力。

与其它邮件传输代理相比，使用 sendmail+IDA 几乎没有什么不可以做的更快和更简单的了。运行常规的 UUCP 或 Internet 站点所需的典型工作变得很容易完成。通常非常困难的配置也变得简单，易于建立和维护。

在编写本书的这个时候，当前版本 *sendmail5.67b+IDA1.5* 可以通过匿名 FTP 从 **vixen.cso.uiuc.edu** 取得。在 Linux 下，它的编译不需要任何补丁程序。

对于在 Linux 下的 sendmail+IDA 源程序编译、安装和运行所需的所有配置文件包括在 *newspak-2.2.tar.gz* 中，该压缩文件可以通过匿名 FTP 从 **sunsite.unc.edu** 的*/pub/Linux/system/Mail* 下载。

15.2 配置文件—概述

传统的 sendmail 是通过一个系统配置文件进行设置的（典型的是*/etc/sendmail.cf* 或者是 */usr/lib/sendmail.cf*），这个文件不是与你所见过的任何程序语言相近的。编辑 *sendmail.cf* 文件以提供定制的性能可能是一种卑下的经验技巧。

Sendmail+IDA 通过将所有配置选项做成句法易于理解的表格驱动方式使得这种痛苦基本上成过时的情况了。这些选项经由源代码提供的制作文件（Makefiles）通过在许多数据文件上运行 *m4*（一个宏处理程序）或者 *dbm*（一个数据库处理程序）来配置的。

sendmail.cf 文件仅仅定义了系统缺省的性能。事实上，所有特殊的定制操作是通过很多可选的表格做到的，而不是直接编辑 *sendmail.cf* 文件来做的。图 15.1 中列出了 *sendmail* 所有的表格。

mailertable	为远程主机或域定义特殊的性能。
uucpxtable	迫使 UUCP 邮件投递到具有 DNS 格式的主机。
pathtable	定义到远程主机或域的 UUCP bang-paths。
uucprelays	短路路径别名路径到著名的远程主机。
genericfrom	将内部地址转换成外部世界可见的普通地址。

xaliases 将普通地址转换成有效的内部地址/或将内部地址转换成普通地址。

decnetxtable 将 RFC-822 地址转换成 DECnet 形式的地址。

图 15.1: *sendmail* 的支持文件。

15.3 *sendmail.cf* 文件

用于 sendmail+IDA 的 *sendmail.cf* 文件不用直接编辑的，而是从一个本地系统管理员提供的 *m4* 配置文件中生成的。下面，我们将称它为 *sendmail.m4*。

这个文件含有几个定义和其它一些仅仅是指向进行真正工作表格的连接。一般地，仅需要指明：

。在本地系统上所使用的路径名和文件名。

。用于 e-mail 目的的站点名称。

。哪个缺省邮件程序（也可以是灵敏主机）是所期望的。

有大量的参数可以被定义，用来设立本地站点的性能或覆盖编译进的配置项目。这些配置选项是在源代码目录中的 *ida/cf/OPTIONS* 文件中确定的。

对于最小配置的 *sendmail.m4* 文件（具有所有非本地邮件被中继到直接连接的灵敏主机去的 UUCP 或 SMTP）如果不算注释行的话，能够短到只有 10 或 15 行。

15.3.1 一个 sendmail.m4 的例子文件

下面示出了在虚拟酿酒厂的 **vstout** 的 *sendmail.m4* 文件。**vstout** 用 SMTP 与酿酒厂局域网上的所有主机对话，而将所有其它目的地的所有邮件通过 UUCP 发送到它的 Internet 中继主机 **moria**。

15.3.2 用于 *sendmail.m4* 的典型参数

sendmail.m4 文件中有几项是一直需要的；其它的可以被省略如果你使用缺省值侥幸成功的话。下面几节详细描述了 *sendmail.m4* 例子文件中的每个项。

定义路径的项（Items that Define Paths）

```
dnl #define(LIBDIR,/usr/local/lib/mail)dnl  # where all support files
go
```

LIBDIR 定义了一个目录，在这个目录中，sendmail+IDA 期望能找到各个配置文件、

```
dnl #------------------ SAMPLE SENDMAIL.M4 FILE ------------------
```

```
dnl # (the string ‘dnl’ is the m4 equivalent of commenting out a line)
dnl # you generally don’t want to override LIBDIR from the compiled in paths
dnl #define(LIBDIR,/usr/local/lib/mail)dnl    # where all support files go
define(LOCAL_MAILER_DEF, mailers.linux)dnl    # mailer for local delivery
define(POSTMASTERBOUNCE)dnl                  # postmaster gets bounces
define(PSEUDODOMAINS, BITNET UUCP)dnl        # don’t try DNS on these
dnl #---------------------------------------------------------------
dnl #
define(PSEUDONYMS, vstout.vbrew.com  vstout.UUCP vbrew.com)
dnl                                      # names we’re known by
define(DEFAULT_HOST, vstout.vbrew.com)       # our primary ‘name’ for mail
define(UUCPNAME, vstout)dnl                  # our uucp name
dnl #
dnl #---------------------------------------------------------------
define(UUCPNODES, |uuname|sort|uniq)dnl      # our uucp neighbors
define(BANGIMPLIESUUCP)dnl                   # make certain that uucp
define(BANGONLYUUCP)dnl                      # make is treated correctly
define(RELAY_HOST, moria)dnl                 # our smart relay host
define(RELAY_MAILER, UUCP-A)dnl              # we reach moria via uucp
dnl #
dnl #-----------------------------------------------------------------------
dnl #
dnl # the various dbm lookup tables
dnl #
define(ALIASES, LIBDIR/aliases)dnl           # system aliases
define(DOMAINTABLE, LIBDIR/domaintable)dnl    # domainize hosts
define(PATHTABLE, LIBDIR/pathtable)dnl       # paths database
define(GENERICFROM, LIBDIR/generics)dnl      # generic from addresses
define(MAILERTABLE, LIBDIR/mailertable)dnl    # mailers per host or domain
define(UUCPXTABLE, LIBDIR/uucpxtable)dnl     # paths to hosts we feed
define(UUCPRELAYS, LIBDIR/uucprelays)dnl     # short-circuit paths
dnl #
dnl
#-------------------------------------------------------------------------
dnl #
dnl # include the ‘real’ code that makes it all work
dnl # (provided with the source code)
dnl #
include(Sendmail.mc)dnl                      # REQUIRED ENTRY !!!
dnl #
dnl #------------- END OF SAMPLE SENDMAIL.M4 FILE -------
```

图 15.2：用于 **vstout** 的 *sendmail.m4* 例子文件。

各种 dbm 表格和特殊的本地定义。在一个典型的执行程序发行版中，这被编译进了 sendmail 的执行程序中，不需要在 *sendmail.m4* 文件中明确地设置。

上面的例子有一前导 *dnl*，这表示该行基本上是一个作为信息的注释行。

为了将支持文件的位置改变到一个不同的地方去，从上面一行中去掉 *dnl* 注释，将路径设置到期望的位置，并且重建和重安装 *sendmail.cf* 文件。

定义本地邮件程序（Defining the Local Mailer）

```
define(LOCAL_MAILER_DEF, mailers.linux)dnl    # mailer for local delvery
```

许多操作系统提供了处理本地邮件投递的程序。对于许多主要的 Unix 变体来说，典型的程序早已制作进 sendmail 执行程序中了。

在 Linux 中，就需要明确地定义合适的本地邮件程序，因为本地投递程序并无必要一定会包括在你已经安装的发行版中。这是通过在 *sendmail.m4* 文件中指明 *LOCAL_MAILER_DEF* 来做到的。

例如，为了让通用的 *deliver* 程序[1]提供这个服务，你应该将 *LOCAL_MAILER_DEF* 设置成 *mailers.linux*。

然后，下面的文件应该作为 *mailers.linux* 被安装在由 *LIBDIR* 指定的目录中。它明确地用适当的参数在内部 *Mlocal* 邮件程序中定义了 *deliver* 程序，以使得 *sendmail* 能正确地投递目标面向本地系统的邮件。除非你是一个 sendmail 专家，否则一般你不需要改变下面的例子。

```
# -- /usr/local/lib/mail/mailers.linux –
#    (local mailers for use on Linux )
Mlocal, P=/usr/bin/deliver, F=SlsmFDMP, S=10, R=25/10, A=deliver $u
Mprog,  P=/bin/sh,      F=lsDFMeuP,   S=10, R=10, A=sh –c $u
```

在包括在 *sendmail.cf* 文件中的 *Sendmail.mc* 文件里对于 *deliver* 也有一内建缺省值。为了指明它，你不能使用 *mailers.linux* 文件，而是要在你的 *sendmail.m4* 文件中进行如下定义：

```
dnl --- (in sendmail.m4) ---
define(LOCAL_MAILER_DEF, DELIVER)dnl      # mailer for local delivery
```

不幸的是，*Sendmail.mc* 假定 deliver 被安装在*/bin* 中，而对于 Slackware1.1.1（它将其安装在了*/usr/bin 中*）并不是这种情况。在那种情况下，你就必须用链接伪装它或者从源代码中重建 deliver 以使它驻留在*/bin* 中。

处理反弹的邮件（Dealing with Bounced Mail）

```
define(POSTMASTERBOUNCE)dnl             # postmaster gets bounces
```

许多站点发现，确信邮件是以接近 100%成功率发送和接收是很重要的。正如检查 *syslogd(8)*日志是很有帮助的一样，本地邮件管理者通常需要观察反弹回来的邮件的标题信息，以确定邮件的不能投递是否是因为用户错误还是相关系统之一的配置问题。

定义 *POSTMASTERBOUNCE* 会导致对每个反弹回来的消息作一份拷贝，并被送到作为系统**邮件管理者（Postmaster）**的人那里。

不幸的是，设置这个参数也将导致消息的文本内容被送到了邮件管理者 Postmaster，这潜在地会涉及到系统上使用邮件者的隐私问题。

一般地，站点邮件管理者应该尽力严格律己（或者通过使用 shell 脚本的技术方式来删除他们所收到的反弹回来的消息的内容），不阅读不是自己邮件的内容。

与域名服务相关的项（Domain Name Service Related Items）

```
define(PSEUDODOMAINS, BITNET UUCP)dnl      # don't try DNS on these
```

由于历史原因有些著名的网络通常在邮件地址中参考引用，但它们对于 DNS 来说却是无效的。定义 *PSEUDODOMAINS* 是用于防止不必要的且总是失败的 DNS 查询企图。

定义本地系统公开的名称（Defining Names the Local System is Known by）

```
define(PSEUDONYMS, vstout.vbrew.com  vstout.UUCP vbrew.com)dnl
                                    # names we're known by
define(DEFAULT_HOST, vstout.vbrew.com)dnl   # our primary 'name' for mail
```

常常，系统希望隐藏它们的真实身份、作为邮件网关的服务、或者是接收和处理寻址到曾经已知的‘老’名字的邮件。

PSEUDONYMS 指定了本地系统将接收其邮件的所有主机的列表。

DEFAULT_HOST 指定了本地主机产生的将出现在消息标题上的主机名。将这个参数设置为一个有效的值是很重要的，否则的话所有返回邮件都将不能被投递。

与 UUCP 相关的项（**UUCP-Related Items**）

```
define(UUCPNAME, vstout)dnl                 # our uucp name
define(UUCPNODES, |uuname|sort|uniq)dnl      # our uucp neighbors
define(BANGIMPLIESUUCP)dnl                  # make certain that uucp
define(BANGONLYUUCP)dnl                     # mail is trated correctly
```

常常，对于用于 DNS 时系统使用一个名字，而用于 UUCP 时系统则使用另外一个名字。*UUCPNAME* 准许你定义出现在出站 UUCP 邮件标题中的不同的主机名。

UUCPNODES 定义了为系统返回主机名列表的命令，而该系统是我们通过 UUCP 连接直接连接的。

BANGIMPLIESUUCP 和 *BANGONLYUUCP* 确保用 UUCP‘bang’句法寻址的邮件依照 UUCP 的性能来对待，而不是以现今用于 Internet 上最近的域名服务来对待。

中继系统和邮件程序（Relay Systems and Mailers）

```
define(RELAY_HOST, moria)dnl                # our smart relay host
define(RELAY_MAILER, UUCP-A)dnl             # we reach moria via UUCP
```

许多系统管理员不想为确保他们的系统能够到达世界范围内所有的网络（因而还有系统）而需

做的工作所捆扰。他们宁愿将所有的出站邮件中继到另一台已知确实“灵敏”的系统去，而不愿那样做。

RELAY_HOST 定义了这样一个灵敏的邻接系统的 UUCP 主机名。

RELAY_MAILER 定义了用于将消息中继到那里去的邮件程序。

请注意，设置这些参数将导致你的出站邮件都将被转发到这个远程系统去，这将影响他们系统的负荷，这点很重要。在配置你的系统使用另一个系统作为一般用途的中继主机之前，一定要从远程邮件管理者（Postmaster）那里取得明确协定。

各种配置表格（The Various Configuration Tables）

```
define(ALIASES, LIBDIR/ALIASES)dnl         # system aliases
define(DOMAINTABLE, LIBDIR/domaintable)dnl  # domainize hosts
define(PATHTABLE, LIBDIR/pathtable)dnl      # paths database
define(GENERICFROM, LIBDIR/generics)dnl     # generic from addresses
define(MAILERTABLE, LIBDIR/mailertable)dnl  # mailers per host or domain
define(UUCPXTABLE, LIBDIR/uucpxtable)dnl    # paths to hosts we feed
define(UUCPRELAYS, LIBDIR/uucprelays)dnl    # short-circuit paths
```

使用这些宏，你可以改变 sendmail+IDA 查找各种 dbm 表格的位置，这些表格定义了系统的“真正”性能。通常，将它们留在 LIBDIR 中是明智的。

主要的 Sendmail.mc 文件（The Master Sendmail.mc File）

```
include(Sendmail.mc)dnl                    # REQUIRED ENTRY !!!
```

sendmail+IDA 的作者提供了 *Sendmail.mc* 文件，它含有成为 sendmail.cf 文件的真正的“内脏”。周期性地，新的版本会发行出来以修正错误或增加新功能，但并不需要有一个完整的版本和从源代码中对 sendmail 进行重新编译。

请不要编辑这个文件，这点是很重要的。

那么那些条目是真正必须的呢？

当不使用任何可选的 dbm 表格时，sendmail+IDA 通过定义在 *sendmail.m4* 文件中的 *DEFAULT_MAILER*（还可能有 *RELAY_HOST* 和 *RELAY_MAILER*）来投递邮件。*sendmail.m4* 是用于生成 *sendmail.cf* 文件的。通过 *domaintable* 或 *uucpxtable* 中的条目可以很容易的覆盖这个特性。

在 Internet 上使用域名服务的一般站点，或者一个仅是 UUCP 的并且使用 UUCP 通过一个灵敏 *RELAY_HOST* 转发所有邮件的站点，很可能完全不需要任何特定的表格。

事实上，所有系统都应设置 *DEFAULT_HOST* 和 *PSEUDONYMS* 宏（这定义了公众已知的规范站点的名字和别名），以及 *DEFAULT_MAILER* 宏。如果所有的只是一个中继主机和中继邮件程序，那么你就不需要设置这些缺省值，因为它是自动工作的。

UUCP 主机很可能也会需要将 *UUCPNAME* 设置成他们官方的（正式的）UUCP 名字。他们可能也要设置 *RELAY_MAILER* 和 *RELAY_HOST*，这通过一个邮件中继启用了灵敏-主机路由选择。所使用的邮件传输器是定义在 *RELAY_MAILER* 中并且对于 UUCP 站点来讲，通常应该是 *UUCP-A*。

如果你的站点仅是 SMTP 的并且用‘域名服务’交谈，那么你应该将 *DEFAULT_MAILER* 改变

成 *TCP-A* 并且还可以删除 *RELAY_MAILER* 和 *RELAY_HOST* 行。

15.4 Sendmail+IDA 表格通览

Sendmail+IDA 提供了几个表格，这些表格允许你覆盖 sendmail（在 sendmail.m4 中指明）的缺省设置并且针对独特（单一）情况、远程系统、以及网络定义特定的行为，这些表格使用随发行版提供的 Makefile 利用 *dbm* 后处理的。[？]

绝大多数站点如果需要用到这些表的话，也将只需要很少几个。如果你的站点不需要这些表的话，那么最简单的方法很可能是让它们成为长度为零的文件（使用 *touch* 命令）并且使用在 *LIBDIR* 中的缺省 Makefile 而不是去编辑 Makefile 本身。

15.4.1 *mailertable*

mailertable 根据远程主机或网络名称，对特定的主机定义特殊的处理。它常常用于在 Internet 的站点上选择一个中间的邮件中继主机或通往远程网络的网关，并且指定所用的特定协议（UUCP 或 SMTP）。UUCP 站点通常不需要使用这个文件。

次序是很重要的。Sendmail 从上至下读取文件并依照它所匹配的第一条规则来处理消息。所以通常是将非常明确的规则放在文件的开头部分，而一般的规则则放在文件的后面。

假设你想通过 UUCP 将 Groucho Marx 大学计算机系的所有邮件转发到一个中继主机 **ada**。为了这样做，你需要有个看上去象下面这样的 *mailertable*：

```
# (in mailertable)
#
# forward all mail for the domain .cs.groucho.edu via UUCP to ada
UUCP-A,ada        .cs.groucho.edu
```

假设想要将更大的 **groucho.edu** 域中所有邮件送到一个不同的中继主机 bighub 中用以进行地址解析和邮件投递。那么扩展后的 mailertable 条目看上去很象下面的样子。

```
# (in mailertable)
#
# forward all mail for the domain cs.groucho.edu via UUCP to ada
UUCP-A,ada        .cs.groucho.edu
#
# forward all mail for the domain groucho.edu via UUCP to bighub
UUCP-A,bighub     .groucho.edu
```

正如上面所提到的，顺序是很重要的。将上面的规则顺序反一下的话将导致所有邮递到**.cs.groucho.edu** 的邮件穿过更一般的 **bighub** 路径而不是真正期望的确定的 **ada** 路径。

```
# (in mailertable)
```

```
#
# forward all mail for the domain .groucho.edu via UUCP to bighub
UUCP-A,bighub     .groucho.edu
#
# (it is impossible to reach the next line because
#   the rule above will be matched first)
UUCP-A,ada        .cs.groucho.edu
#
```

在上面的 mailertable 例子中，*UUCP-A* 邮件程序使得 sendmail 用指定的标题通过 UUCP 进行投递。

在邮件程序（mailer）和远程系统之间的逗号通知它将消息转发到 **ada** 以进行地址解析和投递。

Mailertable 中的条目具有如下格式：

mailer delimiter relayhost *host_or_domain*

有一些可用的 mailer。它们之间的不同之处通常在于处理地址的方式。典型的 mailer 是 *TCP-A*（TCP/IP 和 Internet 形式的地址）、*TCP-U*（TCP/IP 和 UUCP 形式的地址）、*UUCP-A*（UUCP 和 Internet 形式的地址）。

在 mailertable 的一行上面将 mailer 从左边主机部分分割开来的字符定义了 mailertable 是如何修改地址的。要认识到的重要事情是这仅仅重写了信封（以使得邮件到远程系统中去）。要重写除信封以外的任何部分是不赞成的，因为这及有可能破坏邮件的配置。

! 一个感叹号使得在转发到 mailer 之前剥离接收者的主机名。当你想要迫使邮件进入一个配置错误的远程站点时，就可以使用这个符号。

, 逗号对地址不作任何改变。消息仅通过指定的 mailer 转发到指定的中继主机去。

: 仅在你与目的地之间有中间主机的情况下，冒号用于删除接收者的主机名。因此，对于 **foo!bar!joe**，**foo** 会被删除掉，而 **xyzzy!janet** 则将保持不变。

15.4.2 *uucpxtable*

通常，到具有全资域名主机的邮件是通过使用域名服务（DNS）的 Internet 形式（SMTP）邮递方式进行投递的，或通过中继主机进行投递。而 uucpxtable 则通过将有域的名字转换成 UUCP 形式的非域的远程主机名迫使使用 UUCP 路由选择来投递。

当你是一个站点的或一个域的邮件转发器时，或者当你希望通过直接的和可靠的 UUCP 链接来发送邮件而不想通过可能是多跳数的经过缺省 mailer 以及任何中间系统和网络来发送邮件时，这个表是经常使用的。

与使用有域邮件标题的 UUCP 邻居对话的 UUCP 站点将使用这个文件来迫使邮件的投递通过两系统之间直接的 UUCP 点对点链接，而不是使用间接的路由通过 *RELAY_MAILER* 和 *RELAY_HOST* 或者通过 *DEFAULT_MAILER* 来进行邮件的投递。

不使用 UUCP 对话的 Internet 站点通常不会用到这个 *uucpxtable*。

假设你为一个在 DNS 中称为 **sesame.com** 而在 UUCP 映射中称为 **sesame** 的系统提供邮件转发服务。那么你会需要下面的条目以迫使它们主机的邮件通过你的直接 UUCP 连接来投递。

```
#============== /usr/local/lib/mail/uucpxtable ============
# Mail sent to joe@sesame.com is rewritten to sesame!joe and
# therefore delivered via UUCP
#
sesame    sesame.com
#
#-----------------------------------------------------------
```

15.4.3 *pathtable*

pathtable 用于定义到远程主机或网络的明确的路由选择。*pathtable* 文件应该使用路径别名形式的句法，并以字母顺序排过序。每一行上的两个字段必须用实际的制表符 TAB 来分隔，否则的话 *dbm* 可能会有问题。

大多数系统都不需要任何 *pathtable* 条目。

```
#=============== /usr/local/lib/mail/pathtable ================
#
# this is a pathalias-style paths file to let you kick mail to
# UUCP neighbors to the direct UUCP path so you don't have to
# go the long way through your smart host that takes other traffic
#
# you want real tabs on each line or m4 might complain
#
# route mail through one or more intermediate sites to a remote
# system using UUCP-style addressing.
#
sesame!ernie!%s          ernie
#
# forwarding to a system that is a UUCP neighbor of a reachable
# internet site.
#
swim!%s@gcc.groucho.edu      swim
#
# The following sends all mail for two networks through different
# gateways (see the leading '.' ?).
# In this example, "uugate" and "byte" are specific systems that serve
# as mail gateways to the .UUCP and .BITNET pseudo-domains respectively
#
%s@uugate.groucho.edu              .UUCP
bye!%s@mail.shift.com              .BITNET
```

```
#
#==================== end of pathtable ========================
```

15.4.4 *domaintable*

domaintable 一般地用于在一个 DNS 查询发生后迫使进行一定的操作。它允许管理员通过自动地将简写名替换为合适的名字来为经常引用的系统或域建立一个简写名称。它也可以用于将不正确的主机或域名替换成“正确的”信息。

大多数站点不会需要任何 *domaintable* 条目。

```
# ============= /usr/local/lib/mail/domaintable =================
#
#
brokenhost.correct.domain    brokenhost.wrong.domain
#
#
#=================== end of domaintable ========================
```

15.4.5 *aliases*

aliases 允许很多事情**发生**：

。它们为邮件寻址提供了一个简写名或一个众所周知的名称以便发送给一个或多个人。
。它们调用一个程序并将邮件消息作为这个程序的输入。
。它们将邮件发送到一个文件。

所有的系统要求 **Postmaster** 和 **MAILER-DAEMON** 的别名是 RFC 兼容的。

当定义能够调用程序或能向程序进行写操作的别名时，要特别注意安全问题，因为 sendmail 通常是以 setuid-root 运行的。

对 *aliases* 文件所作的改变不会立即起作用的，只有在执行了命令

```
# /usr/lib/sendmail –bi
```

以建立要求的 dbm 表后它才起作用。这也可以通过执行 *newaliases* 命令来做到。通常是从 cron 中调用这个命令的。

有关邮件别名的详细信息可以从 *aliases(5)*手册页中找到。

```
#---------------------                    /usr/local/lib/mail/aliases
-------------------
#
# demonstrate commonly seen types of aliases
```

```
#
usenet:        janet                  # alias for a person
admin:         joe,janet               # alias for several people
newspak-users:  :include:/usr/lib/lists/newspak
                                      # read recipients from a file
changefeed:    | /usr/local/lib/gup     # alias that invokes a program
complaints:    /var/log/complaints      # alias that writes mail to a
file
#
# The following two aliases must be present to be RFC-compliant.
# It is important to have them resolve to 'a person' who reads mail routinely.
#
postmaster:    root                    # required entry
MAILER-DAEMON:  postmaster              # required entry
#
#-----------------------------------------------------------------
```

15.4.6 很少使用的表

还有以下一些表，但是很少使用。有关的详细信息请查阅随同 sendmail+IDA 源代码一起的文档。

uucprelays uucprelays 文件是用于“短路”UUCP 到著名站点的路径，而不是使用多跳数的或不可靠的由使用 *pathalias* 处理 UUCP 映射生成的路径。

Genericfrom and *xaliases*

genericfrom 文件通过自动地将本地用户名转换成与内部用户名不匹配的一般发送者地址，来对外部世界隐藏用户名和地址。

相关的 xalparse 工具会自动地生成 genericfrom 和 aliases 文件，这样入站和出站用户名的转换从一个主 xaliases 文件中发生。

decnetxtable *decnetxtable* 将有域地址重写为 decnet 形式的地址，这非常象 domaintable 可用于将非域的地址重写为 SMTP 形式的地址。

15.5 安装 *sendmail*

在本节中，我们将观察如何安装一个典型的 sendmail+IDA 执行程序，并且讨论使它专用化和正常运行需要做些什么。

Linux 的 sendmail+IDA 的最新的执行程序版可以从 **sunsite.unc.edu** 下的*/pub/Linux/system/Mail* 中得到。即使你已有一个 *sendmail* 的早期版本，我强烈希望你使用 *sendmail5.67b+IDA1.5* 版本，因为所有 Linux 要求的补丁程序现在已经在 vanilla 的代码中并且几个重大的安全漏洞也已被补上，这些漏洞存在于约 1993 年 12 月 1 日以前的版本中。

如果你是从源代码来建立 *senmail*，你应该按照包括在源代码发行版中的 *README* 指示来操作。最新的 sendmail+IDA 源代码在 **vixen.cso.uiuc.edu** 上有。要在 Linux 上建立（编译）sendmail+IDA，

你也需要 *newspak-2.2.tar.gz* 中的 Linux 专用配置文件，在 **sunsite.unc.edu** 上的*/pub/Linux/system/Mail* 目录中有这个文件。

如果你以前已经安装过 *smail* 或其它的邮件投递代理，那么出于安全的考虑，你可能需要从 *smail* 中删除（或改名）所有的文件。

15.5.1 提取执行程序发行版

首先你必须在某个安全的地方打开存档（archive）文件：

```
# gunzip –c sendmail5.65b+IDA1.5+mailx5.3b.tgz | tar xvf –
```

如果你有一个“最新的”（“modern”）*tar*（例如从最近的 Slackware 发行版中取得的），那么你还可以只运行 `tar –zxvf filename.tgz` 就能得到相同的结果。

打开这个存档文件会创建一个名为 *sendmail5.65b+IDA1.5+mailx5.3b* 的目录。在这个目录中，你可以找到 sendmail+IDA 完整的安装程序和 *mailx* 用户代理的执行程序。在这个目录下的所有文件路径都对应着文件应该被安装的位置，所以逐步使用 *tar* 命令将它们移过去是安全的：

```
# cd sendmail5.65b+IDA1.5+mailx5.3b
# tar cf - . | (cd /; tar xvvpoof -)
```

15.5.2 创建 *sendmail.cf*

为给的站点创建一个定制的 *sendmail.cf* 文件，你必须写一个 *sendmail.m4* 文件，并且用 *m4* 对它进行处理。在*/usr/local/lib/mail/CF* 中，可以找到一个称为 *sample.m4* 的样本文件。将它拷贝为 ***yourhostname***.*m4*，并对它进行编辑以反映你的站点的情况。

这个样本文件是为只使用 UUCP 的站点设置的。并需要这个站点具有有域标题（domainized headers）并与一个灵敏主机对话的。象这样的站点只需要改动很少的项。

在本节中，我将只对你必须改变的宏，给出一简略的概述。对于它们的功能的完整描述，请参阅前面对 *sendmail.m4* 的讨论。

LOCAL_MAILER_DEF

定义了详细说明本地邮件投递 mailer（邮件程序）的文件。起功能参见上面小节“定义本地邮件程序”。

PSEUDONYMS

定义你的本地主机为人所知的所有名称。

DEFAULT_HOST

放入你的全资域名。这个名字将作为你的主机名出现在所有出站邮件中。

UUCPNAME

放入你的无限定（绝对的）主机名。

RELAY_HOST 和 *RELAY_MAILER*

如果你用 UUCP 与一个灵敏主机对话，将 RELAY_HOST 设置成你的‘灵敏中继’uucp

邻居的名字。如果你想有有域的标题，那么就使用 UUCP-A 邮件程序。

DEFAULT_MAILER

如果你在 Internet 上并且使用 DNS，你应该将它设置成 *TCP-A*。这将通知 sendmail 使用 TCP-A 邮件程序，该邮件程序通过 SMTP 在信封上使用正规的 RFC 形式的寻址来投递邮件。Internet 站点一般不需要定义 *RELAY_HOST* 或 *RELAY_MAILER*。

要创建 sendmail.cf 文件，执行命令

```
# make yourhostname.cf
```

这将对 *yourhostname.m4* 文件进行处理并从中创建出 *yourhostname.cf*。

下一步，应该测试你所创建的配置文件是否如你期望的去工作。这将在下两节中作出解释。

一旦你对它的性能感到满意，使用下面命令将它拷贝到*/etc* 下。

```
# cp yourhostname.cf /etc/sendmail.cf
```

此时，你的 sendmail 系统已经为运行准备好了。将下面一行放入适当的启动文件中（一般是 */etc/rc.inet2* 中）。你现在也可以手工执行它让程序现在就启动。

```
# /usr/lib/sendmail –bd –q1h
```

15.5.3 测试 *sendmail.cf* 文件

使用`-bt` 标志调用 sendmail 将它置入‘测试’模式。安装在系统上的缺省配置文件是 *sendmail*.cf 文件。你可以使用`-C`*`filename`* 选项来测试一个备用的文件。

在下面的例子中，我们对 *vstout.cf* 进行测试，这个配置文件是从图 15.2 中示出的 *vstout.m4* 文件生成的。

```
# /usr/lib/sendmail –bt –Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
>
```

下面的测试确保 sendmail 能够将所有邮件投递给你的系统上的用户。在所有的情况下测试的结果都应该是一样的并且 *LOCAL* mailer 指向本地系统名。

首先测试一个到本地用户的邮件是如何被投递的。

```
# /usr/lib/sendmail –bt –Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 me
```

```
rewrite: ruleset  3   input: me
rewrite: ruleset  7   input: me
rewrite: ruleset  9   input: me
rewrite: ruleset  9 returns: < me >
rewrite: ruleset  7 returns: < > , me
rewrite: ruleset  3 returns: < > , me
rewrite: ruleset  0   input: < > , me
rewrite: ruleset  8   input: < > , me
rewrite: ruleset 20   input: < > , me
rewrite: ruleset 20 returns: < > , @ vstout . vbrew . com , me
rewrite: ruleset  8 returns: < > , @ vstout . vbrew . com , me
rewrite: ruleset 26   input: < > , @ vstout . vbrew . com , me
rewrite: ruleset 26 returns: @# LOCAL $@ vstout . vbrew . com $: me
rewrite: ruleset  0 returns: @# LOCAL $@ vstout . vbrew . com $: me
```

输出结果显示出 *sendmail* 内部是如何处理地址的。地址被传给各种规则集（rulesets）并由这些规则集来分析它、轮流调用其它规则集并且将它分裂成为它的各个组件。

在我们的例子中，我们将地址 **me** 传给规则集 3 和 0（这就是在地址前输入的 3,0 的意思）。最后一行显示出由规则集 0 返回的被分析过的地址，包含有用于投递消息的 mailer、以及给予 mailer 的主机名和用户名。

接下来，使用 UUCP 句法测试将邮件投递到你系统上的一个用户。

```
# /usr/lib/sendmail –bt –Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 vstout!me
rewrite: ruleset  3   input: vstout ! me
[...]
rewrite: ruleset  0 returns: $# LOCAL $@ vstout . vbrew . com  $: me
>
```

下一步，使用 Internet 句法用你的全资主机名测试邮递到你的系统上一个用户。

```
# /usr/lib/sendmail –bt –Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 me@vstout.vbrew.com
rewrite: ruleset  3   input: me @ vstout . vbrw . com
[...]
rewrite: ruleset  0 returns: $# LOCAL $@ vstout . vbrew . com $: me
>
```

你应该使用在 *sendmail.m4* 文件中 *PSEUDONYMS* 和 *DEFAULT_NAME* 参数上指定的每个名字，重复进行上面两个测试。

最后，测试你是否可以邮递到你的中继主机上。

```
# /usr/lib/sendmail –bt –Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 fred@moria.com
rewrite: ruleset  3   input: fred @ moria . com
rewrite: ruleset  7   input: fred @ moria . com
rewrite: ruleset  9   input: fred @ moria . com
rewrite: ruleset  9 returns: < fred > @ moria . com
rewrite: ruleset  7 returns: < @ moria . com > , fred
rewrite: ruleset  3 returns: < @ moria . com > , fred
rewrite: ruleset  0   input: < @ moria . com > , fred
rewrite: ruleset  8   input: < @ moria . com > , fred
rewrite: ruleset  8 returns: < @ moria . com > , fred
rewrite: ruleset 29   input: < @ moria . com > , fred
rewrite: ruleset 29 returns: < @ moria . com > , fred
rewrite: ruleset 26   input: < @ moria . com > , fred
rewrite: ruleset 25   input: < @ moria . com > , fred
rewrite: ruleset 25 returns: < @ moria . com > , fred
rewrite: ruleset  4   input: < @ moria . com > , fred
rewrite: ruleset  4 returns: fred @ moria . com
rewrite: ruleset 26 returns: < @ moria . com > , fred
rewrite: ruleset  0 returns: $# UUCP-A $@ moria $: < @ moria . com > ,
fred
>
```

15.5.4 综合 – 集中测试 *sendmail.cf* 和各个表

到现在为止，你已经检验过了邮件系统已有了期望的默认性能，并且你将能够收发具有有效地址的邮件。为了完成安装，还需要创建适当的 dbm 表以取得期望的最终结果。

在创建了你的站点所需的各个表之后，你必须在包含表的目录中运行 *make* 通过 *dbm* 来对这些表进行处理.

如果你是仅使用 UUCP 的，那么你就不需要创建在 *README.linux* 文件中所提到的任何表了。你只需要 touch 这些文件以使得 Makefile 能顺利工作。

如果你是仅使用 UUCP 的并且你除了与你的灵敏主机对话以外还与其它站点有来往，那么你就需要为每个站点增加 *uucpxtable* 条目（否则的话邮个它们的邮件也将通过灵敏主机传递了）并且针对修改过的 *uucpxtable* 运行 *dbm*。

首先你必须确定经过你的 *RELAY_HOST* 的邮件是通过 *RELAY_MAILER* 发送给它们的。

```
# /usr/lib/sendmail –bt –Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 fred@sesame.com
rewrite: ruleset  3   input: fred @ sesame . com
rewrite: ruleset  7   input: fred @ sesame . com
rewrite: ruleset  9   input: fred @ sesame . com
rewrite: ruleset  9 returns: < fred > @ sesame . com
rewrite: ruleset  7 returns: < @ sesame . com > , fred
rewrite: ruleset  3 returns: < @ sesame . com > , fred
rewrite: ruleset  0   input: < @ sesame . com > , fred
rewrite: ruleset  8   input: < @ sesame . com > , fred
rewrite: ruleset  8 returns: < @ sesame . com > , fred
rewrite: ruleset 29   input: < @ sesame . com > , fred
rewrite: ruleset 29 returns: < @ sesame . com > , fred
rewrite: ruleset 26   input: < @ sesame . com > , fred
rewrite: ruleset 25   input: < @ sesame . com > , fred
rewrite: ruleset 25 returns: < @ sesame . com > , fred
rewrite: ruleset  4   input: < @ sesame . com > , fred
rewrite: ruleset  4 returns: fred @ sesame .com
rewrite: ruleset 26 returns: < @ sesame .com > , fred
rewrite: ruleset  0 returns: $# UUCP-A $@ moria $: < @ sesame . com > ,
fred
>
```

除了 *RELAY_HOST* 外，如果你还有 UUCP 邻居，那么你必须确保邮递给它们有着正确的操作。以 UUCP 形式句法寻址的邮件到一个与之用 UUCP 对话的主机必须是直接递送给它的（除非你用 *domaintable* 条目明确地指出不这样做）。假设主机 **swim** 是你的一个直接 UUCP 邻居。那么给 *sendmail* 输入 **swim!fred** 将产生下面的结果：

```
# /usr/lib/sendmail –bt –Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 swim!fred
rewrite: ruleset  3   input: swim ! fred
[...lines omitted ...]
rewrite: ruleset  0 returns: $# UUCP $@ swim $: < > , fred
>
```

如果你有 *uucpxtable* 条目来迫使 UUCP 投递到一个 UUCP 邻居（这个邻居使用 Internet 形式的有域标题发送它们的邮件），就也需要对其进行测试。

```
# /usr/lib/sendmail –bt –Cvstout.cf
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
> 3,0 dude@swim.2birds.com
rewrite: ruleset  3   input: dude @ swim . 2birds .com
[...lines omitted ...]
rewrite: ruleset  0 returns: $# UUCP $@ swim . 2birds $: < > , dude
>
```

15.6 重复性的邮件处理工作和愚蠢的邮件技巧

既然我们已经讨论了 sendmail+IDA 系统的配置、安装和测试的原理，让我们再花一点时间来看一下在邮件管理员的工作中天天会需要做些什么事。

远程系统有时会中断，Modem 或电话线路会不通，由于人为的因素 DNS 的定义可能会不正确，网络会突然失去作用。在这些情况中，邮件管理者需要知道如何迅速、有效、安全地作出反应，以保持邮件能够从备用的路由传递，直到远程系统或服务提供者能恢复正常服务为止。

本章接下来的部分打算为你提供最常碰到的“电子邮件紧急事件”的解决方案。

15.6.1 将邮件转发到一个中继主机

为一个特定的主机或域转发邮件到一个指定的中继系统，你一般要使用 *mailertable*。

例如，为了转发 **backwood.org** 的邮件到它们的 UUCP 网关系统 **backdoor** 去，你要将以下条目放入 *mailertable* 中：

```
UUCP-A，backdoor     backwood.org
```

15.6.2 迫使邮件发送到误配置的远程站点

Internet 上的主机在要将邮件送到配置错误的远程站点时常常会碰到问题。这个问题有几种情况，但一般症状是邮件被远程系统反弹回来或者是完全到不了那里。

这些问题可能会将本地系统管理员置入一个很糟糕的境地，因为你的用户通常是不关心你个人并不是世界上每个系统的管理者（或者是知道如何让远程系统的管理员解决问题）。他们只知道他们的邮件没有被传递到另一端期望的接收者那里去并且你可能就是他们抱怨的人。

远程站点的配置是远程站点管理员的问题，而不是你的。在所有这些情况下，请确保不要中断你的站点，以与误配置的站点保持通信。如果你不能与远程站点的邮件管理员联系上让他们及时地修复他们的配置问题，那么你有两个选择。

。迫使邮件成功地传送到远程系统通常是可以做到的，尽管远程系统是误配置的而使回复邮件可能不能工作...但是这已是远程管理员的问题了：

```
braindead.correct.domain.com     braindead.wrong.domain.com
```

。通常，误配置的站点会将邮件“反弹”回发送系统并振振有辞地说“那个邮件不是到这个站点的”因为在他们的配置中一般没有他们的 *PSEUDONYMNS* 或等效集。从你站点发往他们消息的信封上完全将主机名和域信息剥离是很可能的。
在下面 *mailertable* 中的!将邮件投递给他们的远程站点使得给邮件呈现在他们的 *sendmail* 面前就好象它是在他们系统上本地生成的。所以适当的返回地址仍将出现在消息中。

```
TCP!BRAINdead.correct.domain.com   braindead.wrong.domain.com
```

不管怎样，即使你将邮件发送到他们的系统中，并不能保证他们能回复你的邮件（记住，他们中断了。。。）但此时他们的用户会在他们的管理员面前大喊大叫而不是你的用户在你面前。

15.6.3 迫使邮件通过 UUCP 传递

在一个理想的世界中（从 Internet 观点来看），所有的主机在域名服务（DNS）中都有记录并且将用全资域名来发送邮件。

如果偶尔使用 UUCP 来与这样的站点对话，你可以迫使邮件通过点对点的 UUCP 连接来传输而不是通过潜在的使用 *uucpxtable* 让他们的主机名“不使用域系统”经过你的缺省的 mailer。

为了迫使 UUCP 投递到 **sesame.com**，你应该将下面的信息放入你的 *uucpxtable* 中。

```
# un-domainize sesame.com to force UUCP delivery
sesame    sesame.com
```

结果是 sendmail 此时将确定（通过 *sendmail.m4* 文件中的 *UUCPNODES*）你是直接连接到远程系统的并且将把邮件排入队列中以使用 UUCP 来投递。

15.6.4 避免邮件通过 UUCP 进行投递

也可能发生相反的情况。通常，系统可能有几个经常使用的直接 UUCP 连接或者它们并不是如默认的 mailer 或中继主机那样很可靠和总是存在的。

例如，在西雅图地区有很多系统，当 Linux 发行版发布时，它们通过匿名 UUCP 来交换各种 Linux 发行版。必要时这些系统仅使用 UUCP 对话，所以通过多个非常可靠的转寄和公共的（并且总是存在的）中继主机来发送邮件通常更快。

避免用 UUCP 将邮件投递到一个你直接连接的主机是很容易可行的。如果远程系统有一个全资域名，你可以这样将一个条目加入到 *domaintable* 中：

```
# prevent mail delivery via UUCP to a neighbor
snorkel.com      snorkel
```

这将用全资域名 FQDN 替换任何出现的 UUCP 名字，因而避免了与 *sendmail.m4* 文件中 *UUCPNODES* 一行的匹配。结果通常是邮件将通过 *RELAY_MAIL* 和 *RELAY_HOST*（或 *DEFAULT_MAILER*）来传递。

15.6.5 根据需要运行 sendmail 队列

要立刻处理存于队列中的消息，只需键入‘/usr/lib/runq’。这使用适当的选项调用 sendmail，导致 sendmail 立刻处理悬挂起的作业而不会等到下一次预定的运行。

15.6.6 报告邮件静态参数

许多站点的管理员（和这些管理员的上司）对入站、出站和经过他们站点的邮件量很感兴趣。有许多方法来测量邮件的交通量。

。与 Sendmail 一起有一个称为 *mailstats* 的工具，该程序读取一个称为*/usr/local/lib/mail/sendmail.st* 的文件并且报告出在 *sendmail.cf* 文件中使用的每个 mailer 的邮件的数量和传输的字节数。sendmail.st 这个文件必须要由本地管理员手工创建以让 sendmail 进行日志登记。运行的统计数是通过删除并重新创建 *sendmail.st* 文件来清除的。一种方法是这样做：

```
# cp /dev/null /usr/lib/local/mail/sendmail.st
```

。或许有关谁使用邮件以及发送、接收和经过本地系统的容量大小的最佳方法是使用 *syslogd(8)* 开启邮件调试。一般来讲，这意味着要从系统启动文件（这是你一定要做的）中运行*/etc/syslogd* 后台程序，并将与下面类似的信息加入到*/etc/syslog.conf(5)*中：

```
mail.debug                      /var/log/syslog.mail
```

如果你使用 mail.debug 并且得知任何媒介有很高的邮件容量，那么 syslog 输出将会很大。从 *syslogd* 输出的文件通常需要在 *crond(8)*的一个过程中让该文件被循环使用或清除。

有许多公用的工具可以用来统计从 syslogd 输出的邮件日志。一个非常著名的工具之一是 *syslog-stat.pl*，这是一个 *perl* 脚本，是与 sendmail+IDA 程序一起发布的。

15.7 混合和匹配发布的执行程序

电子邮件传输和投递代理并没有真正的标准配置，也没有“一个真正的目录结构。”

相应地，有必要确保系统的所有各个方面（USENET news、邮件、TCP/IP）都有一致的本地邮件投递程序（*lmail*、*deliver*、*etc.*）、远程邮件投递程序（*rmail*）、以及邮件传输程序（*sendmail* 或 *smail*）的位置。这样的假设通常并没有在文档中说明，尽管使用 strings 命令可以帮助确定什么文件和目录是期望的。下面是我们曾经见过的在 Linux 执行程序发布版和代码中普遍出现的某些问题。

。某些 TCP/IP 的 NET-2 发行版本中有为称为 umail 定义的服务而不是为 sendmail 的。
。有各种 *elm* 和 *mailx* 的端口是寻找*/usr/bin/smail* 投递代理而不是 *sendmail*。
。Sendmail+IDA 对于 *deliver* 有一个内建的本地 mailer，但是期望它是位于/bin 中而不是更为典型的 Linux 位置*/usr/bin*。

通常我们不会去从源代码建立所有邮件客户程序，这样太麻烦了，我们通常用适当的软链接来仿冒它。。。

15.8 从哪里获得更多的信息

有很多地方你可以找到有关 *sendmail* 的更多信息。关于这些信息的列表，见定期投递到 **comp.answers** 上的 Linux MAIL Howto。在匿名 FTP **rtfm.mit.edu** 上也有它。然而，权威性的地方就在 sendmail+IDA 代码中。参见 source 目录下的 *ida/cf* 目录中有关的文件 *DBM-GUIDE*、*OPTIONS* 和 *Sendmail.mc*。

注释

[1] *deliver* 是由 Chip Salzenberg (**chip%tct@ateng.com)**编写的。它是几个 Linux 发行版的一部分，可以在平常的匿名 FTP 档案（如 **ftp.uu.net**）中找到。

第十六章 网络新闻（Netnews）

16.1 Usenet 的历史

网络新闻（network news）的观念产生于 1979 年，当时有两位研究生 Tom Truscott 和 Jim Ellis 考虑在 UN*X 用户之间使用 UUCP 来连接各台机器以进行信息的交换。他们在北卡罗莱纳州用三台机器组成了一个小网。

最初，信息传输是通过很多 shell 脚本来处理的（后来用 C 语言改写），但是这些脚本从未向外界公布过。它们很快就被“A”news 替换掉了，这是第一个向外界发布的 news 软件版本。

“A”news 只设计成处理每天每个组很少的文章。当新闻组的容量继续在增大时，该软件由 Mark Horton 和 Matt Glickman 进行了重写，他两人称其为“B”发行版（又名 Bnews）。Bnews 第一个公开发行版是 1982 年的 2.1 版。它被不断地扩充，增加了几个新特性。目前它的版本是 Bnews 2.11。现在它已慢慢地被舍弃，最后一个正式维护者也转向了 INN。

另一次重写是由 Geoff Collyer 和 Henry Spencer 在 1987 年完成并发行的；这是版本“C”，或称 C News。在接下来的一段时间内出了许多针对 C News 的补丁程序，最为显著的是 C News Performance 版本。在有着大量新闻组的站点上，由于频繁地调用 *relaynews*（该程序负责将入站文章分发到其它站点去）而造成的系统开销是很大的。Performance 版本给 relaynews 加了一个选项，该选项允许它以后台模式（*daemon mode*）来运行，将自己置入后台。

Performance 版本是目前包括在大多数 Linux 版中的 C News 版本。

一直到“C”的所有 news 发行版基本上都是用于 UUCP 网络的。尽管它们也能用于其它的环境。在象 TCP/IP、DECNet 或其它相应的网络上进行有效的新闻传输要求一个新的方案。这就是为什么在 1986 年引进“网络新闻传输协议”（“Network News Transfer Protocol”），NNTP 的原因了。它是基于网络连接的，并且指定了许多进行交互传送和收取文章的命令。

网上有许多基于 NNTP 的应用程序。其中之一就是 Brian Barber 和 Phil Lapsley 的 *nntpd* 软件包，你可以使用这个软件以及其它软件在一个局域网内为许多主机提供新闻阅读服务。*nntpd* 是被设计成补足象 Bnews 或 C News 这样的 news 软件包的，为它们提供 NNTP 的特性。

另一个不同的 NNTP 软件包是 INN，即 Internet News。它不仅仅是一个前端程序，它自己就是一个 news 系统。它是一个能够有效地同时维护几个并行 NNTP 链接的复杂的 news 中继后台程序，因而许多 Internet 站点选择它作为 news 服务器。

16.2 总之，什么是 Usenet？

有关 Usenet 最显著的事实之一是它不属于任何组织，也没有任何集中的网络管理特权。实际上，这是 Usenet 学问的一部分，除了技术说明以外，你定义不了它到底是*什么*，你只能指出它不是什么。如果你手头有本 Brendan Kehoe 的出色的“禅和 Internet 艺术”（网上和 Prentice-Hall 有售，见 [Kehoe92]），你会找到一张有趣的有关不属于 Usenet 的列表。

冒着让人听上去感觉很蠢的风险，人们可以把 Usenet 定义为交换 Usenet News 的独立站点的集合。要成为一个 Usenet 站点，你全部所需做的是找到另一个 Usenet 站点，并与它的拥有者和维护者达成和你交换 news 的协议。为另一个站点提供 news 也称为给它*喂*信，这起源于另一个 Usenet

哲学的公理：“Get a feed and you’re on it.”

Usenet news 最基本的单元是文章。这是用户写作并“投递”到网上的。为了让 news 系统能够处理，这些文章附带了管理信息，即所谓的文章标题（头）。它与符合 Internet 邮件标准 RFC 822 的邮件的标题格式非常相似，它也是有几行文本组成，每行用一个以冒号结尾的字段名开始，其后带有字段的值。[1]

文章被提交给一个或更多个新闻组（*newsgroups*）。人们可以将一个新闻组看作是有关一个共同主题文章的论坛。所有的新闻组以层次结构组成，每个组名指出了自己在这个层次结构中的位置。这使得一个组能够很容易地被看出是有关哪方面的。例如，从新闻组的名称上任何人都可以看出 **comp.os.linux.announce** 是用于有关名为 Linux 的操作系统公告的。

然后这些文章在所有乐意从这个组中传递 news 的 Usenet 站点之间进行交换。当两个站点同意互相交换 news 时，他们可以自由地交换所喜欢的任何新闻组，甚至可以加上他们自己本地 news 层次结构。例如，**groucho.edu** 可能与 **branyard.edu**（这是一个主要的 news 喂信机）有一个 news 链接，还与几个它要喂信的小站点有链接。现在，Barnyard 大学可能接收了所有的 Usenet 组，而 GMU 却只想传送几个象 **sci**、**comp**、**rec** 等的主要的层次结构。某些下游的站点，比如说一个称为 **brewhq** 的 UUCP 站点，甚至想传送更少的组，因为它们没有这种网络或硬件资源。另一方面，**brewhq** 可能想要从 **fj** 层次结构中接收新闻组，而这个层次结构 GMU 没有传送。因此，它就与 **gargleblaster.com** 维持另一个链接，这个站点却传送了所有 **fj** 组，并将这些组喂给 **brewhq**。这些 news 的流动见图 16.1 中所示。

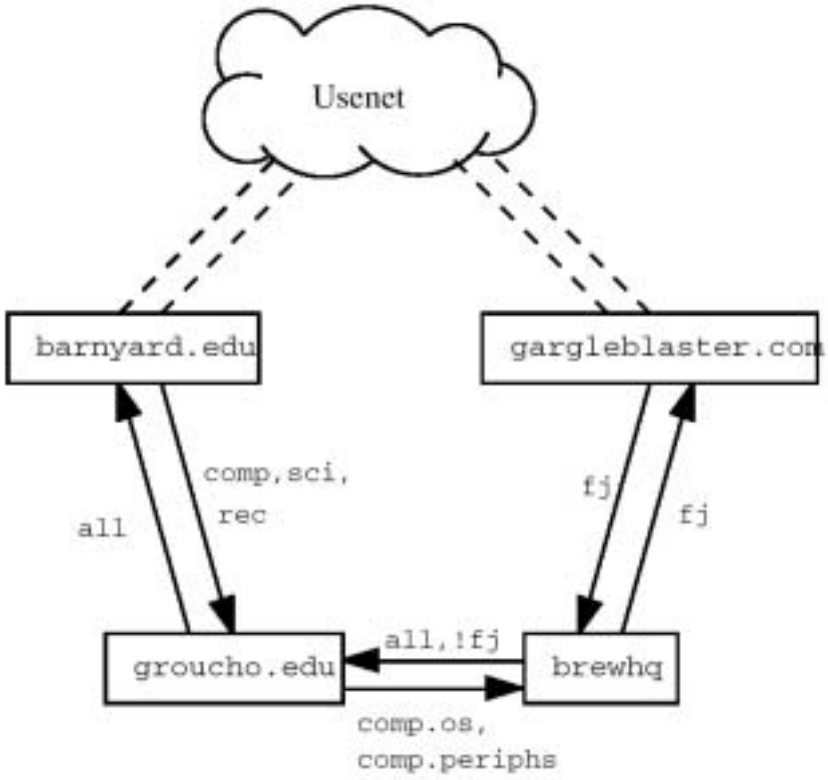

图 16.1 Usenet news 在 Groucho Marx 大学中的流动

尽管很清楚，从 **brewhq** 发出的箭头上的标签可能还是需要作些解释。默认地，它会要所有本地产生的 news 被送到 **groucho.edu**。然而，由于 **groucho.edu** 没有传送 **fj** 组，所以从这些组给它发送任何消息中没有箭头指向。因此，从 **brewhq** 给 GMU 的喂信操作被标上了 **all,!fj**，表示除了 **fj** 下的所有组都发送给它。

16.3 Usenet 是如何处理 News 的？

如今，Usenet 已经增长到占有巨大的比例。传送整个网络新闻的站点通常每天传输六十兆字节已不足为奇。[2] 当然这比任意摆布文件需要更多的处理。所以让我们看看大多数 UN*X 系统处理 Usenet news 的方法。

News 是通过各种传输渠道经由网络来发布的。传统的传输媒介曾经是 UUCP，但是如今主要的数据流量是由 Internet 站点来传送的。所用的路由选择的算法称为扩散法（*flooding*）：每个站点都维护着许多到其它站点的链接（*news feeds*）。任何由本地 news 系统产生的或接收到的文章都被转发给它们，除非文章已在那个站点上，在这种情况下文章将被丢弃。通过观察 Path：标题字段，一个站点可以发现文章已经穿过的所有其它站点。这个标题含有文章已经被其转发过的以 bang path 标记的所有系统的一张列表。

为了区别各篇文章和识别重复文章，Usenet 文章必须要携带一个消息 id（在 `Message-Id`：标题字段中指明），它将投递站点的名字和一个序列号组合在一起形成“`<serial@site>`”。对于所处理的每篇文章，news 系统将这个 id 记录进 history 日志文件中，据此，检查所有新到的文章。

任何两个站点之间的文章信息流动可以通过两个准则来加以限制：其一，文章被指派一个发布信息（在 `Distribution`：标题字段），该信息可用于将该文章归入站点内适当的组中。另一方面，被交换的新闻组可以通过发送或接收系统两者来加以限制。允许往一个站点传送的新闻组和分类的集合通常保存在 *sys* 文件中。

纯粹的文章数量通常要求对上述方案进行改进。在 UUCP 网络上，平常要做的是每隔一段时间收集一次文章，并将收集来的文章组合成一单个文件，进行压缩并发送到远程站点去。这称为批量处理（*batching*）。[3]

另外一种技术是 *ihave/sendme* 协议，这个协议在文章最初的地方就避免重复发送文章，这样就节约了网络带宽。不是把所有的文章放入一个批处理文件中并将它们一起发送出去，而是只将文章的消息 id 组合成一个巨大的“ihave”消息并发送到远程站点。远程站点读取这个消息，与它自己的 history 文件比较，然后返回在“sendme”消息中想要的文章的列表。此后，只有这些文章将被发送。

当然，ihave/sendme 只有在涉及两个大站点时才显得有意义，这两个大站点各自从几个独立的喂信者那里接收 news，并且经常相互选取对方作为有效的 news 流动目的地。

在 Internet 上的站点通常依赖于基于 TCP/IP，使用网络新闻传输协议 NNTP 的软件[4]。它在喂信者之间传送 news 并且为远程主机上的单独用户提供 Usenet 访问。

NNTP 可以用三种方法来传递 news。一种是 ihave/sendme 的实时版，也被称为推信（*pushing* news）操作。第二种技术称为拉信（*pulling* news）操作，在这种方法中，客户就向服务器索取指定新闻组或层次结构中文章的列表清单，并选择那些在它的 history 文件中没有的文章，这些文章是在指定日期以后到达服务器站点的。第三种方式是用于交互式新闻阅读的，允许你或你的新闻阅读者从指定的新闻组中抽取文章，以及投递标题信息不完全的文章。

在每个站点上，news 被放置在/var/spool/news 下的目录结构中，每篇文章为一个文件，而每个新闻组在一个单独的目录中。目录名由新闻组名构成，新闻组名中的每个部分构成路径部分。因此，**comp.os.linux.misc** 上的文章是放在*/var/spool/news/comp/os/linux/misc* 中的。一个新闻组中的文章被按到达的次序分配了一个号，这个号就作为文件名。当前在线文章号的范围放置在一个称为 *active* 的文件中，这个文件同时用作你的站点所已知的新闻组的清单。

由于磁盘空间是有限的资源，[5] 在一段时间以后，你就需要开始丢弃一些文章了。这叫做过期操作（*expiring*）。通常，从一定的组和结构中来的文章在到达站点经过一固定天数以后就过期了。投递者若在文章标题字段 `Expires`：中指定一过期的日期，就可以覆盖固定天数了。

注释

[1] Usenet news 消息的格式是在 RFC 1036 中指定的，“USENET 消息交换的标准”（“Standard for interchange of USENET messages”）。

[2] 等一下，以 9600bps 传输 60 Megs，那是 6 千万乘 1200，那是…咕哝，…嘀咕，咳！那需要 34 小时！

[3] 根据 Geoff Collyer 所言，网络新闻的黄金规则是“你需批处理你的文章。”

[4] 在 RFC 977 中作了讨论。

[5] 某些人宣称 Usenet 是 modem 和硬盘厂商的共同阴谋。

第十七章 C News

用于网络新闻的非常流行的软件包之一是 C News。它被设计成用于在 UUCP 链接上传送 news 的站点。这一章将讨论 C News 的中心思想，以及基本的安装与维护任务。

C News 的配置文件存储在*/usr/lib/news* 中，而它的绝大多数执行文件在*/usr/lib/news/bin* 目录中。文章被放置在*/var/spool/news* 下。事实上，你应该确保这些目录中的所有文件要属于用户 **news**、组 **news**。许多问题起于 C News 访问不了文件。在你触碰里面的任何文件之前使用 *su* 变成为 **news** 用户—让这成为你操作的准则。仅有的例外是 *setnewsids*，用于设置某些 news 程序的真实用户 id 的。它必须由 **root** 拥有，并且必须有 setuid 比特置位。

下面，我们详细地描述 C News 的所有配置文件，并且向你说明要让你的站点运行该做些什么。

17.1 投递 News

文章可以有几种方法馈给 C News。当一个本地用户邮递了一篇文章时，新闻阅读器通常会将该文章传递给 *inews* 命令用以建立标题信息。远程站点来的新闻，不管是单篇文章还是整个批处理文件，都被送至 *rnews* 命令。该命令将它们存储在*/var/spool/newsin.coming* 目录中，以后 *newsrun* 会从中选取文件。然而，不管使用这两种技术中的哪一个，文章最终都将被送至 *relaynews* 命令。

对于每一篇文章，通过在 *histroy* 文件中查询消息 id，*relaynews* 命令首先检查该篇文章是否已经在本地站点上。重复的文章将被抛弃。然后，*relaynews* 观察 `Newsgroups:`标题行以查明本地站点是否向任何这些组请求过文章。如果是的并且该新闻组已列在 *active* 文件中，*relaynews* 就会试着将该文章存储到 news spool 区内相应的目录中。如果这个目录不存在的话，就建立一个。此时，这篇文章的消息 id 将被记录在 *history* 文件中。否则的话，*relaynews* 就会抛弃掉这篇文章。

如果由于文章要投递到的组没有列在你的 *active* 文件中而致使 *relaynews* 存储入站文章失败时，该篇文章将被移入 **junk** 组中。[1] *relaynews* 也会对过时的或填错日期的文章进行检测并拒收它们。由于任何其它原因而失败的任何入站批处理文章将被移入*/var/spool/news/in.coming/bad* 中，并在日志文件中记录一个错误消息。

此后，使用各个站点特定的传输方式，文章将被中继到从这些组中请求 news 的所有其它站点去。为了确保文章没有被送到早已见过这篇文章的站点上，每个目的站点会检查文章的 `Path:`标题字段，该字段含有文章至今为止已经经过的站点的清单，是以 bang path 形式写的。

C News 一般用于在 UUCP 站点之间中继 news，尽管也能将它用于 NNTP 环境中。为了将 news 投递到远程 UUCP 站点去—比如单篇文章或整个批处理—*uux* 被用来在远程站点上执行 *rnews* 命令，并在标准输入上把文章或批处理文件喂给它。

当对一指定站点开启了批处理，C News 不会立刻发送任何入站的文章，而是将文章的路径名添加到一个文件中，通常是 *out.going/`site`/togo*。周期性地，crontab 会执行一个批处理程序，[2] 该程序将文章放入一个或多个文件中，还可选择对他们压缩，并且将它们发送到远程站点的 *rnews*。

图 17.1 示出了流经 *relaynews* 的 news。文章可以被转发到本地站点（由 *ME* 表示）、通过 email 到某个名为 **pronderosa** 的站点、或到一个开启了批处理的名为 **moria** 站点。

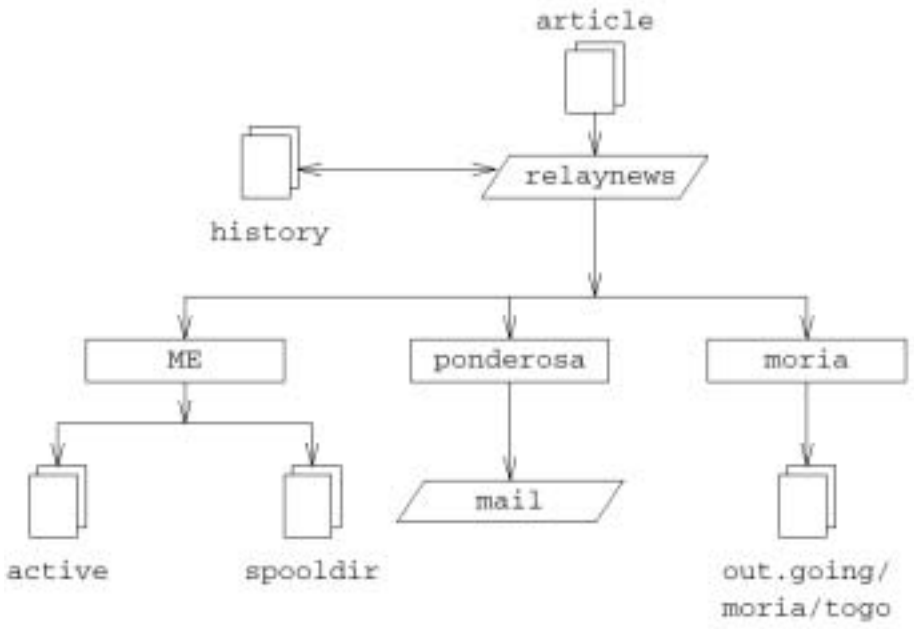

图 17.1 流经 relaynews 的 news。

17.2 安装

为了安装 C News，如果你还没有解开压缩，首先将文件解开到适当的地方，并且编辑下面所列出的配置文件。它们都位于*/usr/lib/news*。它们的格式将在下面几节中描述。

sys

一般你必须修改描述你系统的 *ME* 一行，尽管使用 *all/all* 总是安全的。你也需要为每个你要喂信的站点增加一行配置。
如果你是个页站点，你只需要添加一行将所有本地生成的文章发送给你的喂信者的配置信息。假设你的喂信者是 **moria**，那么你的 *sys* 文件看上去就象这样的：

```
ME:all/all::
Moria/moria.orcnet.org:all/all,!local:f:
```

organization

你的组织的名称。例如，“`Virtual Brewery, Inc.`”。在你自己的机器上，输入“`private site`”，或任何你喜欢的信息。如果你没有定义这个文件，许多人将不能呼叫你的站点。

newsgroups

mailname

你的站点的邮件名称，例如，**vbrew.com**。

wohoami

用于 news 的你的站点名。常常使用 UUCP 站点名，例如 **vbrew**。

explist

一般你应该编辑这个文件来反映某些特定新闻组的过期时间。

为了建立初始的新闻组层次结构，从给你喂信的站点上取得 *active* 和 *newsgroups* 两文件，并将

这两文件安装到*/usr/lib/news* 中，确保它们的宿主是 news 并具有模式 644。从 active 文件中删除所有的 **to.***组，并加入 **to.*mysite*** 和 **to.*feedsite***，以及 **junk** 和 **control**。**to.***组通常用于交换 ihave/sendme 消息的，但不管你是否计划使用 ihave/sendme 都应该建立它们。下一步，在 *active* 的第二和第三字段中使用下面的命令替换掉所有文章的数（号）。

```
# cp active active.old
# sed ‘s/ [0-9]* [0-9]* / 0000000000 00001 /’ active.old > active
# rm active.old
```

第二个命令是对 *sed(1)*的调用，是我最喜欢的 UN*X 命令之一。这个调用分别将两个数字串替换为一个全零串和字符串 `00001`。

最后，建立 news 假脱机目录和用于入站和出站 news 的子目录：

```
# cd /var/spool
# mkdir news news/in.coming news/out.going
# chown –R news.news news
# chmod –R 755 news
```

如果你正在使用一个 C News 的最新发布版，那么你也必须在 news 假脱机目录中建立 *out.master* 目录。

如果你正在使用一个不是已经运行的 C News 中的不同的 newreaders 发布版程序的话，你可能会发现某些 newsreaders 期望 news spool 在*/usr/spool/news* 中而不是在*/var/spool/news*。如果你的 newsreader 好象没有找到任何文章的话，那么就从*/usr/spool/news* 建立一个到*/var/spool/news* 的符号链接。

现在，你已准备好接收 news 了。注意，除了上面示出的目录以外，你不需要建立任何其它目录，因为每次 C News 从一个组接收到一篇至今还没有 spool 目录的文章时，它就会创建该目录的。

特别地，文章被交叉投递的所有组都会发生。所以，过一阵子，你会发现你的 news spool 目录被你从没有订阅的新闻组的目录搞的乱七八糟，象 **alt.lang.teco** 等。你可以通过从 active 中删除所有不想要的组，或通过定期运行一个 shell 脚本来避免出现这种情况。这个 shell 脚本会删除所有在*/var/spool/news*（当然，除了 *out.going* 和 *in.coming*）下的空目录。

C News 需要一个用户来给它发送出错信息和状态报告。默认地，这个用户就是 **usenet**。如果你使用这个缺省的用户，你就必须为它设置一个别名，它会将它的所有邮件中转给一个或多个负责人。（第 14 和 15 章解释了针对 *smail* 和 *sendmail* 是如何做的）。你也可以通过将环境变量 *NEWSMARSTER* 设置成适当的名字来覆盖这个特性。在 **news** 的 crontab 文件中、以及每次你手工调用一个管理工具时，你都必须这样做，所以设置一个别名通常是更容易的。

当你在编辑/etc/passwd 时，请确信每个用户在密码文件的 pw_gecos 字段（这是第四个字段）中都有她真实的名字。这是 Usenet 的网络礼节，发送者的真实姓名必须出现在文章的 `From:`字段中。当然，在使用邮件时你也应该这样做。

17.3 sys 文件

位于*/usr/lib/news* 中的 *sys* 文件控制着你接收和中转到其他站点的层次结构。尽管有着名为 *addfeed* 和 *delfeed* 的维护工具，我想最好亲手来维护这个文件。

sys 文件含有你中转 news 所到的各个站点的条目，以及你接受的组的描述。条目看上去象这样

```
site[/exclusions]:grouplist[/distlist][:flags[:cmds]]
```

条目可以使用一个反斜杠（\）来续行。井字符号（#）表示一个注释。

site 这是条目相应的站点的名称。通常选择站点的 UUCP 名字。在 sys 文件中也必须有你自己站点的条目，否则的话你将收不到任何文章。
特殊的站点名 *ME* 表示你的站点名。*ME* 条目定义了你想在本地存储的所有组。与 *ME* 行不匹配的所有文章都将进入 **junk** 组。
因为 C News 是把 `site` 和 `Path:` 标题字段上的站点名作比较检查的，所以你必须确保它们真正是匹配的。有些站点在这个字段中使用它们的全资域名，或是象 **news**.`site`.*domain* 这样的别名。为了避免任何文章被返回给这些站点，你必须将这些站点加入排除清单中，并用逗号分开。
例如，对于站点 **moria** 的条目，该站点的这个字段将含有 **moria/moria.orcnet.org**。

grouplist 这是一个用逗号分开的组和相应站点层次结构的订阅清单。层次结构可以通过给定层次结构的前缀来指定（比如，名字以 **comp.os** 为前缀的所有组），后面可以跟一个可选用的关键字 **all**（比如，**comp.os.all**）
通过在一个层次结构或组的前面放上一个感叹号就可以排除对这个层次结构或组的转发。如果用这个清单来检查新闻组，那么适用于最长匹配的名称。例如，如果 `grouplist` 含有

```
! comp,comp.os.linux,comp.folklore.computers
```

那么除了 **comp.folklore.computers** 和 **comp.os.linux** 下的所有组以外，**comp** 层次结构下的其它组都不会喂给那个站点。
如果站点请求将你所接收到的所有组都转发给它，那么在 *grouplist* 中输入 *all*。

distlist 是使用一个斜杠的 *grouplist* 的分支，含有要转发的发布信息的一个清单。同样，你可以通过在它们前面加上一个感叹号来排除适当的发布信息。所有的发布信息是用 *all* 来表示的。省略 *distlist* 暗示一个 *all* 清单。
例如，你可以使用一个 *all* 发布清单，*!local* 用于避免只供本地使用 news 发送到远程站点去。
起码有两种发布：*world*，当用户没有指定时，这经常是默认的发布，还有一个是 *local*。还可以有应用于某个区域、州、国家等等的其它发布。总的来说，C News 只使用两个发布；这是 *sendme* 和 *ihave*，并且是用于 sendme/ihave 协议的。
发布的使用是一个有争论的问题。首先，某些新闻阅读器通过简单地使用顶层层次结构建立假的发布，例如，在投递到 **comp.os.linux** 时用 **comp**。应用于区域的发布也常常是有问题的，因为当通过 Internet 发送时 news 可以传输到你的区域之外去。[3] 然而，应用于一个组织机构的发布是非常有意义的，例如，避免机密的信息跑到公司网络外面去。但是，针对这个目的通常最好通过创建一个独立的新闻组或层次结构来做到。

flags 这描述了喂信的适当参数。它可以是空的，或者是下面标志的组合：

F 这个标志开启批量处理。

f 这个标志与 *F* 几乎是相同的，但是允许 C News 更精确地计算出出站批量的大小。

I 这个标志使得 C News 生成一个适用于 ihvae/sendme 的文章清单。还需要对 *sys* 和 *batchparms* 文件额外的修改，以开启 ihave/sendme。

n 这个标志为活跃的 NNTP 传输客户（象 nntpxmit 见第 18 章）建立批处理文件。该批处理文件含有文章的文件名和它的消息 id。

L 这个标志告诉 C News 只传送在你站点上投递的文章。这个标志后面可以跟一个十进制数 *n*，这使得 C News 只传输离你站点 n 跳之内的文章。C News 从 `Path:`字段来确定跳数。

u 这个标志告诉 C News 只从 unmoderated 组中批量处理文章。

m 这个标志告诉 C News 只从 moderated 组中批量处理文章。

你最多可以使用 *F*、*f*、*I*、或 *n* 中的一个。

Cmds 除非开启了批量处理，这个字段域含有对每篇文章所要执行的命令。文章将在标准输入上馈给该命令。这只能用于非常小的输入；否则的话，两个系统的负载都会太大。缺省的命令是

```
uux - -r –z system!rnews
```

该命令在远程系统上调用 *rnews*，在标准输入上给它喂文章。

该字段中给出的命令的默认搜索路径是*/bin:/usr/bin:/usr/lib/news/bin/batch*。最后一个目录中含有许多 shell 脚本，这些脚本的名字以 *via* 开头；将在本章后面讨论。

如果使用了 *F* 或 *f*、*I* 或 *n* 标志开启了批量处理，C News 会期望在这个字段中找到一个文件名而非一个命令。如果这个文件名不是以一个斜杠（/）开头的，就假设是与 */var/spool/news/out.going* 相关的。如果这个字段是空的，它缺省地指向 *system*/*togo*。

在设置 C News 时，你一般肯定要写出自己的 *sys* 文件。为了帮助你编写，下面我们给出了 **vbrew.com** 的样本文件。从中你可以拷贝你需要的部分。

```
# We take whatevr 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
```

17.4 *active* 文件

active 文件位于*/usr/lib/news* 中，其中列出了你的站点所知道的所有组，以及目前在线上的文章。你很少需要改动它，但为了完整起见我们还是对它作些解释。文件中的条目有如下的形式：

```
newsgroup high low perm
```

当然，*newsgroup* 是组的名字。*Low* 和 *high* 是现有的最低和最高文章数量。如果此时一篇文章也没有，那么 *low* 就等于 *high*+1。

起码，这是 low 字段所代表的意思。然而，考虑到效率的问题，C News 并不更新这个字段。在没有新闻阅读器依赖于这个字段时是不会有什么大损失的。例如，*trn* 检查这个字段看它是否能够从它的线程数据库中清除任何的文件。为了更新 *low* 字段，你需要定期地运行 *updatemin* 命令（或者，在早期的 C News 版本中是 *upact* 脚本）。

perm 是一个详细描述用户对组的访问许可。它可以取下列值之一。

y	允许用户往这个组中投递文章。
n	不允许用户往这个组中投递文章。然而，该组仍然是可被阅读的。
x	这个组已被本地禁止掉了。当 news 管理员（或他们的上司）禁止用户向某个组投递文章时有时就会这样做。 往这个组投递的文章并不在本地存储，尽管会将这些文章转发到请求它们的站点去。
m	这是一个调解（仲裁）组。当一个用户企图往这个组邮递文章时，一个智能新闻阅读器会注意到她的这个操作，并将该文章发送到调解人那里去而不是这个组中。调解人的地址取自*/usr/lib/news* 中的 *moderators* 文件。
=real-group	这标志出 *newsgroup* 作为另一个组（即 *real-group*）的本地别名。所有邮递到 *newsgroup* 的文章都将重定向到它。

在 C News 中，一般你不需要直接访问这个文件的。可以使用 *addgroup* 和 *delgroup* 在本地增加或删除组（见下面维护工具和任务小节）。当为整个 Usenet 增加或删除组时，常常是通过分别发送一个 *newgroup* 或 *rmgroup* 控制消息来完成的。*你自己绝对不要发送这样的消息！*有关如何建立一个新闻组的方法，请阅读 **news.announce.newusers** 中的每月投递。

与 *active* 紧密相关的文件是 *active.times*。每当创建了一个组时，C News 就往这个文件中记录一条消息，该消息含有被创建组的名称、创建的日期、是通过 *newgroup* 控制消息建立的还是本地建立的、以及是谁操作的。这是为了便于新闻阅读器通知用户近期建立的任何新组。这个文件也被 NNTP 的 *NEWGROUPS* 命令所使用。

17.5 文章批量处理（Batching）

新闻批量处理遵循着一个特定的格式，这个格式是与 Bnews、C News 和 INN 的相同。每篇文章的前面都有象这样的一行：

```
#! Rnews count
```

这里 *count* 是文章的字节数。当使用批量压缩时，产生的文件是作为一个整体压缩的，并且在前面有另一行，其中的信息是用于解压缩的。标准的压缩工具是 compress，是用以下来标记的

```
#! cunbatch
```

有时候，当需要通过会从所有数据中去除第八比特的邮件软件中发送批量文件时，压缩的批量文件可以使用所谓的 c7-编码进行保护；此时这些批量文件将标记为 *c7unbatch*。

当一个批量文件被喂到远程站点上的 *rnews* 时，它会检查这些标记并进行适当的处理。有些站点也使用其它的压缩工具，比如 gzip，并在它们的用此工具压缩的文件前面加上 *zunbatch* 而不是 *cunbatch*。C News 识别不出象这样的非标准标题来；所以你必须修改源程序来支持它。

在 C News 中，文章批量处理是通过*/usr/lib/news/bin/batch/sendbatches* 来执行的，它从 *site*/*togo* 文件中取得文章的清单，并将它们放入几个新闻批处理中。它应该每隔一小时执行一次或更频繁地执行，这依赖于数据的流量。

它的操作是由*/usr/lib/news* 中的 *batchparms* 文件控制的。该文件描述了每个站点所允许的最大批量处理大小、进行批量处理和可选压缩所用的程序、以及将它们投递到远程站点所用的传输方式。你可以针对每个站点指定批量处理的参数，以及对没有明确提及的站点指定一缺省参数集。

要对一特定站点执行批量处理，象下面那样来调用它

```
# su news –c "/usr/lib/news/bin/batch/sendbatches site"
```

当不带参数进行调用时，*sendbatches* 将处理所有排着队的批处理。对“所有”的解释依赖于 *batchparms* 中是否有缺省条目。如果有，则*/var/spool/news/out.going* 中的所有目录都将被检查，否则的话，它将循环使用 *batchparms* 中的条目。注意，当扫描 out.going 目录时，*sendbatches* 将只取不含有点和@符号作为站点名字的目录。

当安装 C News 时，你肯定会在你的发行版中找到一个 batchparms 文件，其中含有一个合适的缺省条目，所以你一般不需要改动这个文件。不过为了以防万一要改动，所以我们还是描述一下它的格式。每一行含有六个字段，由空格或制表符 tab 分开：

```
site size max batcher muncher transport
```

这些字段的含义如下：

site 是条目所应用的站点。这个站点的 *togo* 文件必须位于 news spool 下的 *out.going/togo* 中。/default/站点名表示缺省的条目。

Size 是所创建的最大文章批量文件的大小（在压缩之前）。对与大于这个值的单个文章，C News 将其作为一个例外并将它放入单独一个批量文件中。

max 是特定站点的最多建立和定期要传输的批量的数目。这在远程站点关闭了很长时间的情况

下是很有用的，因为这可以避免 C News 用成千上万个新闻批量文件将你的 UUCP 假脱机目录搞乱。

C News 使用*/usr/lib/news/bin* 中的 *queulen* 脚本来确定排于队列中的批量文章的数目。Vince Skahan 的 newspak 发行版应该含有一个 BNU 兼容 UUCP 的脚本。假如你用了不同的假脱机目录 Taylor UUCP，那么你可能需要自己来编写它。[4]

batcher 字段含有用于从 *togo* 文件中的文章清单生成批量文件的命令。对于定期的喂信操作，这通常是 *batcher*。对于其它的目的，可以提供另外的 batchers。例如，ihave/sendme 协议要求文章清单被转换成 ihave 或 sendme 控制消息，该消息会被投递到新闻组 **to.site** 中。这是通过 *batchih* 和 *batchsm* 来执行的。

muncher 字段指定用于压缩的命令。通常，这是一个产生压缩批量文件的脚本 **compcun**。[5] 另外地，你也可以提供一个使用 *gzip*，比如 *gzipcun*（为了清楚起见，你必须自己编制它）的 muncher。你必须确信远程站点上的 *uncompress* 已被修改成能识别用 *gzip* 压缩的文件。

如果远程站点没有 *uncompress* 命令，你可以指明 *nocomp* 来不做任何压缩。

最后一个字段 *transport* 描述了所用的传输方式。针对不同的传输方式有许多标准的命令，这些命令的名字都以 *via* 开头。*Sendbatches* 在命令行上将目的站点的名字传给它们。如果 *batchparms* 的条目不是*/default/*，它就会从 site 字段通过剥离在第一个点或斜杠后的任何字符来取得站点名字。如果条目是*/default/*，那么就使用在 *out.going* 中的目录名。

有两个命令使用 *uux* 在远程系统上执行 *rnews*；*viauux* 和 *viauuxz*。后一个命令为（老版本的）uux 设置-z 标志以使它对每一篇投递的文章返回成功消息。另一个命令，*viamail*，通过邮件将批量文章发送到远程系统上的 **rnews** 用户。当然，这需要远程系统会将所有 **rnews** 的邮件喂到它们的本地 news 系统。有关这些传输方式完整的清单，请参见 *newsbatch(8)*手册页。

最后三个字段中的命令必须位于 out.going/site 或/usr/lib/news/bin/batch 中。其中多数的命令是脚本，所以你可以方便地根据你的所需编辑这些新工具。它们是通过管道来调用的。文章清单是在标准输入上喂给 batcher 的，并在标准输出上生成批量文件。而这个输出又通过管道送至 muncher，等等。

下面给出了一个样本文件。

```
# batchparms file for the brewery
# site       | size   |max    |batcher  |muncher    |transport
#-------------+--------+-------+---------+-----------+-----------
/default/      100000  22     batcher   compcun     viauux
swim           10000  10     batcher   nocomp     viauux
```

17.6 过期 News（新闻）

在 Bnews 中，过期操作通常是通过 *expire* 程序来执行的，它将新闻组的清单和文章过期的日期作为参数。为了操作不同的层次结构在不同的时间过期，你就必须编制一个分别针对它们每个层次结构调用 *expire* 的脚本。而 C News 对此提供了一个更加方便的解决办法：在一个称为 *explist* 的文件中，你可以指定新闻组和过期的间隔时间。一个称为 *doexpire* 的命令通常从 *cron* 中每天执行一次，并根据这个清单对所有的组进行处理。

有时候，即使已经过期，你也可能想要保存特定组中的文章；例如，你可能想要保存投递到 **comp.sources.unix** 中的程序。这称作存档操作 *archiving*。*explist* 允许你标记出需要存档的组。

explist 中的条目看上去象这样：

grouplist perm times archive

grouplist 是一个用逗号分开的对应于当前条目的组的清单。层次结构可以通过给出组名前缀来指定，后面可以附加任选的 *all*。例如，对应于所有 **comp.os** 下的组，你可以在 *grouplist* 中输入 **comp.os** 或 **comp.os.all**。

当从组中对新闻进行过期操作时，将把名字与 *explist* 中所有条目以给出的顺序作比较检查。首个匹配的条目将起作用。例如，除了 **comp.os.linux.announce** 中的文章你想保留一个星期外，其它多数文章在四天后就丢弃掉，那么你只需针对前者有一个条目，指定过期时间周期为一个星期，后面跟一个针对 **comp** 的条目，指定过期周期为四天。

perm 字段详细说明了该条目应用于 moderated、unmoderated、还是任何的组。它可以取值 *m*、*u*、或 *x*，分别表示 moderated、unmoderated 或任何类型。

第三个字段 *times* 通常只含有一个数。如果没有在文章标题的 Expires：字段中人为指定了过期日期，这将是文章将过期的天数。注意这个天数的计数是从文章到达你的站点开始的，而不是从文章投递的日期开始算起的。

然而，*times* 字段可能更复杂些。它可以是三个用斜杠分开的数字的组合。第一个数字表示文章被认为是过期的所要经过的天数。除了用数字零以外，这个数字很少要用到其它的数。第二个字段是上面所提到的文章过期缺省的天数。第三个字段是一篇文章无条件地所经过的天数，而不管该文章是否有 Expires:字段。如果只给出了中间一个数字，那么另两个数就取缺省值。这些可以用特定的条目*/bounds/*来指定，见下面描述。

第四个字段 *archive*，表示新闻组是否要进行归档，以及归档到那里。如果并不想要归档，那么就要用一短划线。否则，你或者使用一个全路径名（指向一个目录），或者一个@符号。@符号表示缺省的归档目录,这个缺省目录必须在命令行上用-a 标志给予 *doexpire*。归档目录必须属于 **news**。比如，当 doexpire 从 **comp.sources.unix** 归档一篇文章时，就会将该篇文章存储在归档目录下的 **comp/sources/unix** 下，如果该目录不存在就会创建它。然而，归档目录本身是不会被创建的。

在 *explist* 文件中有两个特殊条目是 *doexpire* 所依赖的。它们有关键字*/bounds/*和*/expired/*而不是一个新闻组清单。/bounds/条目含有上述 *times* 字段中三个值的缺省值。

/expired/字段用来确定 C News 要与 *history* 文件中相应行保持多久时间。这是需要的，因为一旦相应的文章已被过期作废处理后，C News 将不会从 histroy 文件中删除一行，但是会与该行保持着联系以防在该日期后重复的文章的到来。如果你只是由一个站点给你喂信的，你可以保持这个值很小。否则的话，在 UUCP 网络上，可以是几个星期，这要依赖于你经历的从这些站点来的文章的延迟值。

一个具有非常小的过期间隔时间的 *explist* 样本文件见如下所示：

```
# Keep histroy lines for two weeks. Nobody gets more than three months
/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
juck                        x      1      -

# control messages are of scant interest, too
control                     x      1      -

# catch-all entry for the rest of it
all                         x      2      -
```

就 C News 的过期操作来讲，还有一些潜在的问题。首先是你的新闻阅读器可能依赖于 active 文件的第三个字段，其中含有在线文章的最低数量。当对文章进行过期操作时，C News 并不更新这个字段。如果你需要（或想要）这个字段反映真实的情况，那么在每次运行 *doexpire* 以后，你需要运行一个称为 *updatemiin* 的程序。[6]

其次，C News 通过扫描新闻组的目录并不会过期，而只是简单地检查一下 *history* 文件看文章是否要过期。[7] 如果 *history* 文件由于某些原因不同步了，文章将会永存于你的磁盘上，因为 C News 已经忘记它们了。[8] 你可以使用*/usr/lib/news/bin/maint* 中的 *addmissing* 脚本来修复这个问题，*addmissing* 将把遗漏的文件加进 *history* 文件，或者使用 *mkhistroy*，该脚本会从破坏的文件中重建整个 *history* 文件。在调用它之前不要忘记换作 **news** 用户，否则会引起 C News 读不了 *history* 文件的错误。

17.7 其它各类文件

还有许多控制着 C News 的文件，但对 C News 的功能来讲并不是很重要的。它们都在 */usr/lib/news* 中。我们将概要地描述它们。

newsgroups 这是 *active* 的搭档文件，含有新闻组名称的清单和对每个新闻组的主题的单行描述。在 C News 接收到一个 *checknews* 控制消息时（见章节 17.8）这个文件会自动更新。

mailpaths 含有每个 moderated 组的 moderator 地址。每行含有组名，后跟 moderator 的电子邮件地址（由制表符 TAB 分隔）。
提供了两个特殊的条目作为缺省。这些是 *backbone* 和 *internet*。两个都以 bang-path 表示方法提供了到最近的主干站点、以及理解 RFC 822 地址（**user@host**）的站点的路径。缺省的条目为

```
internet      backbone
```

如果已安装了 *smail* 或 *sendmail*，那么你就不需要改变 *internet* 条目，因为它们了解 RFC 822 寻址方式。
每当用户投递到一个 moderator 没有明确列出的 moderated 组，就要用到 *backbone* 条目。如果新闻组的名称是 **alt.sewer**，并且 *backbone* 条目含有 `path!%s`，那么 C News 会将该文章邮递到 `path!alt-sewer`，并希望主干机器会转发这篇文章。为了找出要使用哪个路径，请询问喂信给你的站点的管理员。作为最后一个方法，你也可以使用 **uunet.uu.net!%s**。

distributions 这个文件其实不是一个 C News 的文件，但是某些新闻阅读器和 nntpd 使用它。它含有你的站点能够辨认的 distributions 的清单，以及它所起的作用的描述。例如，虚拟酿酒厂有以下文件：

```
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 这个文件含有 C News 所有活动的记录。通过运行 *newsdaily* 它被定期地采集；老日志文件的拷贝以 *log.o*、*log.oo* 等保存。

errlog 这是 C News 所生成的所有错误消息的日志文件。这些错误不包括由于组的不对而造成的无用文章等。如果这个文件不是空的，就会被 newsdaily 自动地邮寄给新闻管理员（缺省为 **usenet**）。
errlog 由 *newsdaily* 负责清理。旧的拷贝被保存为 *errlog.o* 等相似的文件。

batchlog 此文件记录所有 *sendbatches* 的运行结果。通常是不感兴趣的。它也是由 *newdaily* 来维护的。

watchtime 这是 *newswatch* 每次运行时产生的空文件。

17.8 控制消息

Usenet 新闻协议知道一特殊类型的文章能够引起新闻系统特定的响应或动作。这些特殊类型的文章称为*控制*消息（*control* message）。它们是通过文章标题中存在着 `Control:`字段来辨认的，其中含有所要执行的控制操作的名称。它们可分为几类，但所有都是通过位于*/usr/lib/news/ctl* 中的 shell 脚本来处理的。

在 C News 处理这类文章的同时，大多数这类文章会自动地执行它们的操作，而不会通知新闻

管理员。缺省地，只有 checkgroups 消息会被送给新闻管理员，[9] 但你可以通过编辑脚本来改变这种情况。

17.8.1 *cancel* 消息

最为大家知晓的消息是 *cancel*，使用它用户可以取消她早些时候发送的文章。如果该文章存在的话，这可以有效地从假脱机目录中删除这篇文章。*cancel* 消息会被转发到所有从这个被作用的组中接收新闻的站点去，而不管该文章是否早已有了。这是考虑到原来的文章可能会被取消消息所延误。有些新闻系统允许用户取消其他人的信息；这当然是不可取的。

17.8.2 *newgroup* 和 *rmgroup*

涉及创建和删除新闻组的两个消息是 *newgroup* 和 *rmgroup*。在“普通”层次结构下的新闻组只有在 Usenet 读者中经过讨论和表决之后才可以被建立。而应用于 **alt** 层次结构的规则几乎允许是可以随意创建新闻组的。有关更多的信息，请参见定期投递到 **news.announce.newusers** 和 **news.announce.newgroups** 中的信息。绝对不要发送一个 *newgroup* 或 *rmgroup* 消息除非你确实知道允许你这样做。

17.8.3 *checkgroups* 消息

checkgroups 消息是由新闻管理员发送的用以使得在一个网络中的所有站点的 *active* 文件与真实的 Usenet 同步。例如，商业 Internet 服务提供者可能会向他们的客户站点发出这样一个消息。对于主要层次结构的“官方的” *checkgroups* 消息由它的 moderator 每月一次地投递到 **comp.announce.newgroups**。然而它是作为一个普通文章投递的，而不是作为一个控制消息。为了进行 *checkgroups* 操作，将该文章保存到一个文件中，比如*/tmp/check*，将直到控制消息开始处的所有信息删除，并使用下面的命令将它送至 *checkgroups* 脚本：

```
# su news –c “/usr/lib/news/bin/ctl/checkgroups” < /tmp/check
```

这将更新你的 *newsgroups* 文件，并将列于 *localgroups* 中的组加入其中。而老的 *newsgroups* 文件将被更名为 *newsgroups.bac*。注意，本地投递该消息是不起作用的，因为 *inews* 会拒绝接受这样一个大型文章的。

如果 C News 发现 checkgroups 列表与 active 文件之间不匹配，那么它会生成一个命令清单来让你的站点更新到最新，并将该命令列表邮寄给新闻管理员。典型的输出看上去象这样：

```
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
       com.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.xintrisics
        /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
```

当你从你的新闻系统收到象这样的信息时，不要盲目地相信它。依赖于是谁发送的 *checkgroups* 消息，它可能缺少几个组或着甚至是整个层次结构；所以在删除任何组时你应该小心谨慎。如果发现你想在你的站点上要的组被列为遗漏的，那么你就必须用 *addgroup* 脚本来增加它们。将遗漏组的清单保存到一个文件中并将该文件送给下面的小脚本中：

```
#!/bin/sh
cd /usr/lib/news

while read group; do
   if grep –si “^$group[[:space:]].*moderated” newgroup; then
      mod=m
   else
      mod=y
   fi
   /usr/lib/news/bin/maint/addgroup $group $mod
done
```

17.8.4 *sendsys*、*version* 和 *senduuname*

最后，还有三个消息可以用于找出有关网络的拓扑。这些消息是 *sendsys*、*version* 和 *senduuname*。它们将使得 C News 分别将 *sys* 文件、一个软件版本字符串、以及 *uuname(1)*的输出返回给发送者。对于 *version* 消息 C News 是非常简明的；它返回一简单的明了的“C”。

再者，你绝对不要发出这样的消息，除非你确信该消息不会跑到你的（区域）网络以外去。对 *sendsys* 消息的应答可以很快地使一个 UUCP 网络瘫痪。[10]

17.9 NFS 环境中的 C News

在一个本地网络中发布新闻的一个简单方法是将所有新闻保存在一个中央主机上，并通过 NFS 将相关的目录输出，这样新闻阅读器就可以直接扫描文章。在 NNTP 上这种方法的优点是检索和调度文章的负荷是相当低的。另一方面，在一个主机的设备变化极大的多机种的网络中，或者是用户在服务器机器上没有等效的帐号的情况下，NNTP 是胜任的。

当使用 NFS 时，在本地主机上投递的文章必须被转发到中央主机上，因为否则的话，访问管理用的文件可能会将系统暴露在不同的条件下而导致文件的不一致。同样，你可能想通过将你的也需要转发到中央机器上的新闻假脱机区域公布为只读来加以保护。

C News 能够透明地对此进行处理。当你投递了一篇文章，你的新闻阅读器通常会调用 *inews* 来将这篇文章送入新闻系统。这个命令对该文章进行一系列的检查、完善标题，并检查*/usr/lib/news* 中的文件 *server*。如果这个文件存在并且其中含有不同于本地主机的主机名,此时，通过 *rsh* 在那个服务器主机上 *inews* 被调用。由于 inews 脚本使用了很多的二进制命令和 C News 中的支持文件，所以你必须在本地安装了 C News，或者从服务器将 news 软件加载（mount）过来。

为了让 *rsh* 调用能正常地工作，每个用户必须在服务器系统上有一个相同的帐号，也即，一个不需要再被要求密码的帐号。

请确保在 *server* 中给出的主机名与服务器机器上 *hostname(1)*命令的输出是匹配的，否则在试投该文章时 C News 将永远地循环下去。

17.10 维护工具和任务

尽管 C News 很复杂，但是一个新闻管理员的日常工作可以是非常轻松的，因为 C News 为你提供了大量的维护工具。其中有些是用于定期地从 *cron* 中运行的，象 *newsdaily*。使用这些脚本可以大大地减少 C News 的日常维护和喂信需求。

除非另加说明，这些命令都位于*/usr/lib/news/bin/maint* 中。注意，在调用这些命令之前你必须是 **news** 用户。用超级用户的身份来运行这些命令可能会导致 C News 不能访问这些文件。

newsdaily　　这个名称本身就说明了自己：每天运行一次。这是一个重要的脚本，它帮助你保持日志文件的短小、保存最后三次运行的每个拷贝。它也试图检测出任何的异常情况，比如在入站和出站目录中早已失效的批处理、向未知的或 moderated 新闻组的投递活动，等等。产生的错误消息将被邮寄给新闻管理员。

newswatch 这是一个应该定期运行的脚本程序用以查找新闻系统中的异常情况，一般一个小时左右运行一次。它是用来检测对新闻系统的运行有着即可影响的问题并且将问题报告邮寄给新闻管理员。所要检查的情况包括陈旧被锁定而删除不掉的文件、没有预订的输入批处理、以及磁盘空间的短缺。

addgroup 给你本地站点增加一个组。适当的调用是

```
addgroup groupname y|n|m|=realgroup
```

第二个参数有着与 *active* 文件中的标志同样的含义，意思是任何人都可以向该组中投递（y）、任何人都不可以向该组投递（n）、那是一个 moderated 组（m）、或者这是另外一个组（`=realgroup`）的别名。
当最新建立的组的文章在 *newgroup* 控制消息打算创建该组之前到来时，你也可以使用 *addgroup*。

delgroup 允许你删除本地的一个组。调用方式如

```
delgroup groupname
```

你仍然需要删除余留在新闻组假脱机目录中的相应文章。或者，你也可以任随它自然地消去（也叫作*过期 expire*）。

addmissing 将遗漏的文章加到 *history* 文件中。当发现有文章好象老是存在时，就运行这个脚本程序。[11]

newsboot 这个脚本程序应该在系统启动时运行。它会删除当新闻进程在关机时刻被杀死时所遗留下来的任何锁定的文件，并且会关闭和执行任何从 NNTP 连接余留下来的批处理，这是在系统关闭时所中止的 NNTP 连接。

newsrunning 这个程序在*/usr/lib/news/bin/input* 中，可以用来，例如在工作时间内，禁止对入站新闻的解批处理操作（unbatching）。你可以通过调用

```
/usr/lib/news/bin/input/newsrunning off
```

来关闭解批处理操作。
用 *on* 代替 *off* 就可以开启它。

注释

[1] 在你的站点上的组和那些你的站点要接收的组之间可能会有些不同。例如，订阅清单可能指明 **comp.all**，这表明在 **comp** 结构下的所有新闻组，但是在你的站点上，只有几个 **comp** 组列在 *active* 中。邮递到那些组的文章都将移入 **junk**。

[2] 注意，为了不破坏文件的权限，这应该是 **news** 的 crontab。

[3] 比如对于一篇在 Hamburg 投递的文章通过荷兰的 reston.ans.net、甚至通过 U.S 的某些站点跑到

了 Frankfurt。

[4] 如果你并不关心假脱机文件的数量（因为你是使用你的计算机的唯一的人，并且你不会写出有兆字节大的文章来），你可以用一简单的 *exit 0* 语句来替换脚本中的内容。

[5] 作为与 C News 一起发行，**compcun** 用 12 比特选项使用 **compress**，因为对于很多站点来说，这是最经济实惠的（最小公分母）。你可以生成它的一个拷贝，比如说是 **compcun16**，这里你使用 16 比特压缩。但是改进并不是很明显。

[6] 对于 C News 的早期版本，是用一个称为 *upact* 的脚本来完成的。

[7] 文章到达的日期是保存在 histroy 行的中间字段中，从 1970 年 1 月 1 日算起以秒计。

[8] 我不知道这为什么会发生，但对于我来说，这并不是经常的。

[9] 在 RFC 1036（p.12）中有一个有趣的排印错误：“Implementors and administrators may choose to allow control messages to be carried out automatically, or to queue them for annual processing.”

[10] 我也不会在 Internet 上试用的。

[11] 还想知道如何除去“Help! I can’t get X11 to work with 0.97.2!!!”这样的文章吗？

第十八章 NNTP 说明

由于所使用的不同网络传输方式，NNTP 为 C News 的新闻交换提供了一个非常不同的途径。NNTP 代表“网络新闻传输协议”（“Network News Transfer Protocol”），它并不是一个特定的软件包，而是一个 Internet 标准。[1] 它是基于通常在 TCP 上的流式连接，该连接存在于网络中的客户和在磁盘上保存着网络新闻的服务器主机之间。流式连接允许客户与服务器交互地协商几乎无延迟的文章传输，因而使得重复的文章数量很低。另外，还由于 Internet 的高传输速率，这使得至今新闻的传输胜过原来的 UUCP 网络。而在几年前，让一篇文章到达 Usenet 的最后一个角落往往要两个星期或更长的时间，这在当时是普遍的现象，但现在常常不用两天就能到达；而在 Internet 当中，甚至只需要几分钟就行了。

有各种命令允许客户检索（提取）、发送和投递文章。发送和投递之间的区别在于后者可能涉及带有不完整标题信息的文章。[2] 文章的检索可以被用于新闻传输客户以及新闻阅读器。这使得 NNTP 为许多客户在一个本地网络上提供进行新闻访问的一个极好的工具，而不需要在使用 NFS 时绕弯子。

NNTP 同样也提供主动的和被动的新闻传输方法，相应地称为“推入”（“pushing”）和“拉出”（“pulling”）操作。推入操作基本上与 C News 的 ihave/sendme 协议相同。客户通过“*IHAVE* <varmsgid>”命令向服务器提供文章，而服务器则返回一个响应代码，指出是否已有这篇文章，还是想要这篇文章。如果想要这篇文章，客户就会发送出这篇文章，并且用单个点在一独立行上来结束该篇文章。

推入新闻有一个缺点，它会给服务器系统带来很大的负荷，因为服务器必须针对每篇文章搜索它的历史数据库。

相反的技术是拉出新闻，在这种操作中，客户从一个组中请求一在指定日期以后到达的所有文章的列表清单。这个查询是由 *NEWNEWS* 命令来执行的。从返回的消息标题中，客户挑选出那些自己还没有的文章，然后对每篇文章依次使用 *ARTICLE* 命令。

拉出新闻的问题在于它需要服务器对客户请求的组和发布的信息有严密的控制。例如，它必须确保本地站点新闻组中没有机密的资料被发送给未经认可的客户。

对于新闻阅读器还有许多方便的命令允许它们对文章标题和文章内容分别检索，甚至是检索一系列文章标题行中的一行。这让你能够将所有的新闻都保存在一个中央主机上，让网络上的所有用户（假定都是本地的）使用基于 NNTP 的客户程序进行新闻的阅读和投递。这是第 17 章中讨论过的通过 NFS 输出新闻目录的另一种替代方法。

NNTP 的一个总体问题是它允许有本事者将有着伪造发送者说明的文章插入到新闻流中。这称为伪造新闻（*news faking*）。[3] 一个对 NNTP 的扩展是对特定命令要求用户身份验证。

现存有很多 NNTP 软件包。其中最为有名的是 NNTP daemon，也以参考实现（*reference implementation*）知名。原先，它是由 Stan Barber 和 Phil Lapsley 编写的，用以演示 RFC 977 的细节。目前它的最新版本是 *nntpd-1.5.11*，这将在下面讨论。你或者可以取得源程序并由你自己进行编译，或者使用 Fred van Kempen 的 *net-std* 执行程序包中的 *nntpd*。没有现存的直接可用的 *nntpd*，因为有很多与站点相关的值要编译进去。

nntpd 软件包由一个服务器和两个客户软件以及一个 *inews* 替代程序组成，这两个客户软件分别用于拉出和推入操作。它们原是在 Bnews 环境下运行的，但有些奇怪的是它们在 C News 下也能很好的运行。然而，如果你计划使用 NNTP 不仅仅是给新闻阅读器提供访问你的新闻服务器的功能，那么参考实现并不是一个实用的选择。因此我们将只讨论 *nntpd* 软件包中的 NNTP daemon，而不涉

及其中的客户程序。

还有一个叫做“InterNet News”的软件包，或简称 INN，是由 Rich Salz 编写的。它同时提供基于 NNTP 和 UUCP 的新闻传输，并且更适合于大型的新闻网络中。当它应用于 NNTP 上的新闻传输时，肯定要比 *nntpd* 好。INN 目前的版本是 *inn-1.4sec*。有一个 Arjan de Vet 编制的小软件用于在 Linux 机器上建立 INN；可以从 **sunsite.unc.edu** 的 *system/Mail* 目录中得到它。如果你想设置 INN，请参考随源程序而来的文档，以及定期投递到 **news.software.b** 中的 INN FAQ。

18.2 安装 NNTP 服务器

NNTP 服务器称为 *nntpd*，根据新闻系统上所期望的负荷情况，可以用两种方式编译它。没有现成编译好的版本的，因为某些与站点相关的默认设置是编制在执行文件中的。所有的配置参数是通过在 *common/conf.h* 中的宏定义来实现的。

Nntpd 可以配置成在系统启动时从 rc.inet2 启动的单机服务器，或配置成由 inetd 来管理的后台程序（daemon）。在后一种情况下，必须在/etc/inetd.conf 中有着下面这样一个条目：

```
nntp      stream  tcp nowait      news    /usr/etc/in.nntpd    nntpd
```

如果你将 nntpd 配置成单机形式的（独立的），请确信在 *inetd.conf* 中的任何象上面这样的行都已被注释掉了。在这两种方式下，都必须保证在*/etc/services* 中有下面这一行：

```
nntp    119/tcp    readnews untp    # Network News Transfer Protocol
```

如果要临时存储任何入站的文章等，那么在你的新闻假脱机目录中 *nntpd* 也需要一个*.tmp* 目录。你应该使用如下命令来创建它。

```
# mkdir /var/spool/news/.tmp
# chown news.news /var/spool/news/.tmp
```

18.3 限制 NNTP 的访问

对 NNTP 资源的访问是由*/usr/lib/news* 中的 *nntp_access* 文件来控制的。该文件中的每一行描述了给予外部主机的访问权限。每一行的格式如下：

site `read|xfer|both|no post|no [!`*exceptgroups*`]`

如果一个客户连接至 NNTP 端口，nntpd 就会通过客户的 IP 地址进行反向查询来获得客户主机的全资域名。此时客户的主机名和 IP 地址就会用来与文件中出现的各个条目的 *site* 字段象比较进行检查。可能会是部分匹配也可能是完全匹配。如果有一个条目是完全匹配的，就使用这个条目；如果是部分匹配的，那么在没有任何其他更好的匹配条目时就应用这个条目。*site* 可以用以下的方法之一来指定：

hostname 这是一个主机的全资域名。如果它与客户的规范主机名完全匹配的话，就使用这个条目，其余所有的条目都被忽略。

IP address 这是一个用*.domain 指定的域名。如果客户的主机名与这个域名匹配的话，该条目就是匹配的。

network name 这是*/etc/networks* 中指定的网络的名字。如果客户 IP 地址的网络号与网络名称相应的网络号匹配的话，该条目就是匹配的。

default *default* 与任何客户相匹配。

有着更为一般的站点说明的条目应该在前面指定，因为任何这些匹配条目将会被后面更精确的匹配条目所代替。

第二和第三个字段描述赋予客户的访问权限。第二个字段详细说明了通过拉出（读取）操作来检索新闻和通过推入（*xfer*）操作来传送新闻的许可权限。一个 *both* 值将启用两者，而 *no* 禁止这两种访问。第三个字段赋予客户投递文章的权限，也即，投递通过新闻软件完善过的原来有着不完整标题信息的文章。如果第二个字段含有值 *no*，那么第三个字段就被忽略。

第四个字段是可选的，含有用逗号分开的禁止客户访问的组的清单。

一个样本 *nntp_access* 文件显示如下：

```
#
# 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
```

18.4 NNTP 授权

当用大写字母写 *nntp_access* 文件中的访问记号时，*nntpd* 对相应的操作需要从客户处得到认可权限。例如，当指定一个 *Xfer* 或 *XFER* 权限，那么只有在客户通过权限认可后，*nntpd* 才让客户将文件传输到你的站点。

权限认可过程是依靠一个称为 *AUTHINFO* 的新的 NNTP 命令来完成的。使用这个命令，客户将发送一个用户名和一个密码到 NNTP 服务器，*nntpd* 通过把这个用户名和密码与*/etc/passwd* 数据库的内容作比较来验证它们，并且还要验证这个用户是否是属于 **nntp** 组的。

NNTP 权限认可的目前实现仅仅是实验性的，因此其移植性并不是很好。其结果是它只能用于纯文本形式的密码数据库；影子密码库它是识别不了的。

18.5 *nntpd* 与 C News 的相互作用

当收到一篇文章时，*nntpd* 必须将它投递到新闻子系统中。根据它是用 *IHAVE* 还是 *POST* 命令接收的，文章将分别送至 *rnews* 或 *inews*。除了调用 rnews，你还可以（在编译时）将它配置成把入站文章进行批处理并且将处理结果移入*/var/spool/news/in.coming* 中，在这个地方，将等待下一次队列操作时 *relaynews* 来取它们。

为了能够正确地执行 ihave/sendme 协议，*nntpd* 必须能够访问 *history* 文件。因此，在编译时，你必须确信路径是正确的。你也必须确信 C News 和 *nntpd* 使用同样格式的 history 文件。C News 使用 *dbm* 杂凑(哈希)函数来访问它的；然而，*dbm* 库有许多不同的和略微不兼容的实现。如果 C News 是与标准 *libc* 中不同的 *dbm* 库链接的话，那么你也必须让 *nntpd* 连接那个库。

nntpd 和 C News 在数据库格式上不相容的一个典型征兆是，系统日志中的错误信息指出 *nntpd* 不能正确地打开它，或着是通过 NNTP 收到重复的文章。一个好的测试方法是从假脱机（spool）目录中任取一篇文章，远程登录（telnet）到 nntp 端口，并把这篇文章提供给 nntpd，见如下所示（你的输入是***象这样***标记的）。当然，你必须用你想再次喂给 nntpd 的文章的 message-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
```

这个对话显示了 *nntpd* 的适当响应；信息“`Got it`”告诉你它已经有这篇文章了。如果你得到了信息“`335 OK`”而不是“`Got it`”，这表示由于某种原因对 histroy 文件的查询失败了。可以用 Ctrl-D 来中止对话。你可以通过检查系统日志来核对什么地方出错了；*nntp* 将使用 *syslog* 的 *daemon* 来记录所有类型的消息。*dbm* 库的不兼容通常在一条信息中本身就很明了，指出 *dbminit* 失败了。

注释

[1] 正式说明在 RFC 977 中。

[2] 当通过 NNTP 邮寄一篇文章时，服务器起码要加上一个标题字段，该字段是 `Nntp-Posting-Host:`。其中含有客户的主机名。

[3] SMTP，即简单邮件传输协议也存在同样的问题。

第十九章 Newsreader 的配置

新闻阅读器（Newsreader）是用于为用户提供方便地访问新闻系统功能的方法，比如象邮寄文章、以适当的方式取得一个新闻组的目录。这个界面的品质正面临着无穷无尽的战火。

已经有许多新闻阅读器移植到了 Linux 上。下面，我将叙述名为 *tin*、*trn* 和 *nn* 的三种最为通用的新闻阅读器的基本设置方法。

最有效的新闻阅读器之一是

```
$ find /var/spool/news –name ‘[0-9]*’ –exec cat {} \; | more
```

这是 UN*X 读取它们的新闻的顽固僵硬的方法。

然而，大部分的新闻阅读器是比较复杂的。它们通常提供一个全屏幕的界面并以不同的层次显示用户预订的所有组和显示一个组中所有文章的概貌以及显示单篇文章。

在新闻组这一层，许多新闻阅读器列出了文章的清单，显示出了这些文章的标题和作者。在大型的组中，用户要明了文章之间的关系是不可能的，尽管能够确认出对早期文章的响应文章来。

一篇响应文章通常重复原有文章的题目，只是在题目前加上了“`Re: `”。另外，后面紧跟的文章的 message id 可能在 `References:`标题行给出。根据这两个标准进行排序可以生成小文章族（实际上是树结构），通常被称为*线索*（*threads*）。编写新闻阅读器程序的任务之一就是设计出有效的生成线索的方案，因为其所需的时间是与文章数的平方成正比的。

这里，我们将不再深入讨论用户界面是如何建立的。目前 Linux 下的所有新闻阅读器都有一个很好的帮助功能，所以你应该会摸索出来的。

下面，我们将只涉及管理方面的任务。其中的大部分都是与线索数据库的建立和帐号相关的。

19.1 *tin* 的配置

有关线索操作的最通用的新闻阅读器是 *tin*。它是有 Iain Lea 编写的并且是松散地以一个很老的新闻阅读器 *tass* 为模型的。[1] 当用户进入一个新闻组时，它就作线索操作，除非你通过 NNTP 来进行这个操作，否则的话这个线索操作是很快的。

在一个 486DX50 的机器上，当直接从硬盘上读取文章时，它大约要花 30 秒的时间来索引 1000 篇文章。对于通过 NNTP 到一个已加载的新闻服务器上，这个操作大约要花费 5 分钟左右的时间。[2] 你可以通过使用-u 选项定期地更新你的索引文件或通过使用-U 选项调用 *tin* 来改善这个操作。

通常，*tin* 将它生成的线索数据库放在*.tin/index* 下的用户主目录中。然而，这可能是很费资源的，所以你会想在一个中心位置处保存它们的一个拷贝。例如，这可以通过让 *tin* setuid 为 **news** 或某个完全无特权的帐号来做到。[3] 此时，*tin* 将把所有的线索数据库放在*/var/spool/news/.index* 下。对于任何的文件访问和 shell 出口，它会把他的有效 uid 复位成调用它的用户的真正 uid。[4]

一个更好的解决方案是安装 *tind* 索引后台程序，它会作为一个后台程序运行并且定期地更新索引文件。然而，这个后台程序并不包含在任何一个 Linux 发行版中，所以你必须自行编译它。如果你正在运行一个有着中心新闻服务器的局域网，那么你甚至可以在该服务器上运行 *tind* 并由所有的客户通过 NNTP 来取回这个索引文件。当然，这需要扩展到 NNTP。对于实现了这个扩展的 *nntpd* 的补丁程序已包括在 *tin* 源程序中。

包含在某些 Linux 发行版中的 *tin* 版本没有编译进对 NNTP 的支持，但现在大多数 *tin* 版本是有的。当作为 rtin 或使用-r 选项被调用时，tin 就会试图连接到文件*/etc/nntpserver* 中指定的或由 NNTPSERVER 环境变量说明的 NNTP 服务器上。*nntpserver* 文件只是简单地在一行上含有服务器的名字。

19.2 *trn* 的配置

trn 也是一个更老的名为 *rn*（意思是阅读新闻）的新闻阅读器的继承者。该名字中的“t”代表“线索操作过的”。它是由 Wayne Davidson 编写的。

不象 *tin*，*trn* 并没有提供运行时刻的线索数据库的生成功能。相反，它使用那些由一个称为 *mthreads* 的程序准备的线索数据库。这个程序必须定期地从 *cron* 中更新索引文件。

然而，不运行 *mthreads* 并不意味着你不能访问新的文章了，它只是表示你将会有所有哪些散落在你的文章选择菜单中的“Novell buys out Linux!!”文章，而不是单一的一个你可以很容易跳过的线索。

为了针对特定的新闻组开启线索操作，就要在命令行上使用一新闻组列表来调用 *mthreads*。这个列表清单与 *sys* 文件中的组成方式是完全一样的：

```
mthreads comp,rec,!rec.games.go
```

该命令将开启对 **comp** 和 **rec** 中所有的组的线索操作，除了 **rec.games.go**（玩 Go 的人不需要流行的线索）。此后，你只需简单地不带任何选项地调用它来对新到的文章进行线索操作。对 *active* 文件中的所有组进行线索操作可以通过使用 **all** 的组列表调用 *mthreads* 来打开。

如果你是在晚间接收新闻的，那么你通常可以在早晨运行一次 *mthreads*，但你也可以根据所需运行多次。有着很大传输流量的的站点可能想以后台方式运行 *mthreads*。当它是在系统启动期间用 -d 选项运行时，它就把自己放入后台，每隔 10 分钟醒来一次来检查是否有新到的文章，如果有则对它们进行线索操作。为了以后台方式运行 *mthreads*，将下面一行写入你的 *rc.news* 脚本中：

```
/usr/local/bin/rn/mthreads –deav
```

-a 选项使得 *mthreads* 对新建的组自动开启线索操作；-v 用来对 *mthreads* 的日志文件 *mt.log* 采用详细的日志消息。该日志文件位于你安装 *trn* 的目录中。

不再存在的老文件必须定期地从索引文件中删去。默认地，只有号码在低水线标记以下的文章将被删除。[5] 但不管怎样已经过期的在这个号码以上的文章（因为该最老的文章在 Expires:标题字段中已经赋予了一个长时间的过期日期）可以通过给予 *mthreads* 一个-e 选项来迫使运行一个“加强的”过期操作来删除。当 *mthreads* 是以后台方式运行时，-e 选项使得它一天中在午夜稍过一些时候进行一次加强的过期操作。

19.3 *nn* 的配置

由 Kim F. Storm 编写的 nn 申明是一个最终目标不是为了阅读新闻的新闻阅读器。它的名字代表“No News”，它的格言是“没有新闻就是好新闻，*nn* 则更好。”（“No news is good news. *nn* is better.”）

为了达到这个野心勃勃的目标，*nn* 带有一大套的维护工具，不但可以生成线索，而且还有对这些数据库一致性的检查、帐号管理、使用统计参数的采集、和访问限制功能。也有一个称为 *nnadmin* 的管理程序，该程序允许你交互式地执行这些任务。它是非常有直觉性的，因此我们将不再深入讨论这些方面，而仅涉及索引文件的生成。

Nn 的线索数据库管理程序称为 nnmaster。它通常从 rc.news 或 rc.inet2 脚本中启动并作为后台程序运行。它是以如下方式调用的

```
/usr/local/lib/nn/nnmaster –l –r -C
```

这对 *active* 文件中的所有组开启了线索操作。

同样地，你也可以从 *cron* 中周期性地调用 *nnmaster*，给它一个进行操作的组的清单。这个清单与 *sys* 文件中的订阅清单非常相似，但它使用空格代替了逗号。它使用一个空的参数“”来表示所有的组，而不是使用伪组名 **all**。一个调用的例子如下

```
# /usr/local/lib/nn/nnmaster !rec.games.go rec comp
```

注意，这里的顺序是很重要的：最左边匹配的组说明是首选的。因此，如果我们将*!rec.games.go* 放在 *rec* 后面的话，那么无论怎样，这个组中的所有文章都将被线索操作过。

nn 提供了几种方法来从它的数据库中删除已过期的文章。第一种方法是通过扫描新闻组目录并且丢弃相应文章不再存在的条目来更新数据库。这是使用`-E` 选项调用 *nnmaster* 的默认操作。除非你是通过 NNTP 来做的，否则它是很快的。

第二种方法的行为非常象 mthreads 的默认过期操作，它只删除那些号码在 active 文件中低水线记号以下的文章。它可以用-e 选项开启。

最后，第三种方法是放弃整个数据库并且重新收集所有的文章。这可以通过将`-E3` 给 *nnmaster* 来做到。

即将过期的组的清单是以上面同样的方式通过`-F` 选项给出的。然而，如果你已有作为后台运行的 *nnmaster*，那么在过期操作开始前你必须杀死它（使用-k），过后再用原来的选项重新运行它。这样使用第一种方法对所有组运行过期操作的适当的命令是：

```
# nnmaster –kF “”
# nnmaster -lrC
```

还有很多的标志用来微调 *nn* 的行为。如果你涉及有关坏文章的删除或对文章进行分类，请阅读 *nnmaster* 的手册。

nnmaster 依赖于一个名为 *GROUPS* 的文件，该文件位于*/usr/local/lib/nn* 中。如果它原本不存在的话，就会被创建。对于每一个新闻组，它都含有一行以组名开始的内容，组名后跟一可选的时间标记以及一些标志。你可以针对相应的组，编辑这些标志项以启用一定的性能，但你最好不要改变各个组出现的顺序。[6]所许可的标志和这些标志的作用在 nnmaster 的手册中也有详细的说明。

注释

[1] 由 Rich Skrenta 编写。

[2] 如果让 NNTP 服务器本身来做线索的操作，并且让客户来取回线索数据库的话，事情就会有巨

大的进展；例如，INN-1.4 就是这样做的。

[3] 然而，对于此不要用 **nobody**。作为一个规则，任何文件或命令都不要与这个用户相关联。

[4] 这就是为什么当作为超级用户调用它时你会得到出错信息。但不管怎样，不要用 **root** 来做。

[5] 注意，C News 不会自动地更新这个低水线标记；你必须运行 *updatemin* 来更新。请参见第 17 章。

[6] 这是因为它们的顺序必须与（二进制）*MASTER* 文件中的条目的顺序一致。

附录 A

PLIP 空打印机电缆

为了制作一根用于 PLIP 连接的空打印机电缆，你需要两个 25 针的连接器（称为 DB-25）和一些具有 11 根导线的电缆。这根电缆最长不得超过 15 米。

如果你细看连接器，你应该能够看到在每个引脚根部的小数字，左上角的引脚是 1（如果你将宽的一边朝上的话）右下角的引脚是 25。对空打印机电缆，你必须按下面的引脚将两个连接器连起来：

```
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
```

所有其它的引脚保持为未连接状态。如果这根电缆是带屏蔽的，那么屏蔽层应该只与一端的 DB-25 的金属外壳相连接。

附录 B

smail 样本配置文件

本节示出了局域网上一个 UUCP 末端站点的样本配置文件。它们是基于 *smail-3.1.28* 发行版源程序中包括的样本文件的。尽管我对这些文件是如何工作的作了简单的解释，建议你还是去阅读非常精美的 *smail(8)*手册，其中详细讨论了这些文件。一但你已理解了 *smail* 配置后面的基本原理，它们是值得一读的。这是很容易的！

示出的第一个文件是 *routers*，它描述了 *smail* 的一族路由器。当 *smail* 必须将一个消息投递到一给定地址时，它就将该地址依次送到所有路由器去，直到其中一个与之匹配。这里所谓的匹配是指这个路由器在它的数据库中，即它的 *paths* 文件、*/etc/hosts*、或路由器接口的任何路由选择机制中找到了目的主机。

在 *smail* 配置文件中的条目总是以一个唯一的名字开始，这个名字确定了相应的路由器、传输方式或导向器。它们后跟定义它们行为的属性列表。这个列表清单由一族全局属性组成，比如象所用的驱动程序（*driver*），以及只有相应驱动程序能够理解的专用的属性。属性是用逗号分开的，而全局和专用属性族是用分号来分开的。

为了使这些区别更明显，假设你要维护两个独立的路径别名文件；一个含有你的域的路由选择信息，另一个含有含有全局路由选择信息，很可能是从 UUCP 映射生成的。使用 *smail*，你现在可以在 *router* 文件中指定两个路由器，它们都使用路径别名（*pathalias*）驱动程序。这个驱动程序在一个路径别名数据库中查找主机名。它期望能够在一个私有属性中给予它文件的名称：

```
#
# pathalias database for intra-domain routing
domain_paths:
       driver=pathalias,        # look up host in a paths file
       transport=uux;          # if matched, deliver over UUCP

       file=paths/domain,       # file is /usr/lib/smail/paths/domain
       proto=lsearch,          # file is unsorted (linear search)

       optional,              # ignore if the file does not exist
       required=vbrew.com,      # look up only *.vbrew.com hosts

#
# pathalias database for routing to hosts outside our domain
world_paths:
       driver=pathalias,        # look up host in a paths file
       transport=uux;          # if matched, deliver over UUCP

       file=paths/world,        # file is /usr/lib/smail/paths/world
       proto=bsearch,          # file is sorted with sort(1)
       optional,              # ignore if the file does not exist
```

```
        -required,              # no required domains
        domain=uucp,              # strip ending ".uucp" before searching
```

上面在两个 *routers* 条目中给出的第二个全局属性定义了当路由器与地址匹配时所用的传输方式。在我们的情况中，消息将使用 *uux* 传输来投递。传输方式在 *transports* 文件中定义，这将在下面解释。

如果你指定了一个方法文件而不是 *transports* 属性，你就可以微调使用哪个传输方式来投递一个消息。方法文件提供了从目标主机到传输方式的映射。这里我们不对此进行讨论。

下面的 routers 文件定义了一个局域网的要查询解析器库的路由器。然而，在一个 Internet 主机上，你可能会使用一个处理 MX 记录的路由器。因此，你应该打开被注释掉的另外一个使用 *smail* 内建 BIND 驱动程序的 *inet_bind* 路由器。

在一个混合使用 UUCP 和 TCP/IP 的环境中，你可能会遇到这样的问题，在*/etc/hosts* 文件中有你偶尔用 SLIP 或 PPP 连接的主机。通常，你仍然想要通过 UUCP 来为它们发送任何邮件。为了避免 *inet_hosts* 驱动程序与这些主机的匹配，你必须将它们放入 *paths/force* 文件中。这是另一个路径别名形式的数据库，并在 *smail* 查询解析器之前被查询。

```
# A sample /usr/lib/smail/routers file
#
# force – force UUCP delivery to certain hosts, even when
#       they are in our /etc/hosts
force:
        driver=pathalias,         # look up host in a paths file
        transport=uux;           # if matched, deliver over UUCP

        file=paths/force,         # file is /usr/lib/smail/paths/force
        optional,                # ignore if the file does not exist
        proto=lsearch,           # file is unsorted (linear search)
        -required,               # no required domains
        domain=uucp,             # strip ending ".uucp" before searching

# inet_addrs – match domain literals containing literal
#       IP addresses, such as in janet@[191.72.2.1]
inet_addrs:
        driver=gethostbyaddr,     # driver to match IP domain literals
        transport=smtp;          # deliver using SMTP over TCP/IP

        fail_if_error,           # fail if address is malformed
        check_for_local,         # deliver directly if host is ourself

# inet_hosts – match hostnames with gethostbyname(3N)
#       Comment this out if you wish to use the BIND version instead.
Inet_hosts:
        Driver=gethostbyname,     # match hosts with the library function
        Transport=smtp;          # use default SMTP
```

```
        -required,              # no required domains
        -domain,                # no defined domain suffixes
        -only_local_domain,       # don't restrict to defined domains

# inet_hosts – alternate version using BIND to access the DNS
#inet_hosts:
#       driver=bind,            # use built-in BIND driver
#       transport=smtp;         # use TCP/IP SMTP for delivery
#
#       defnames,               # use standard domain searching
#       defer_no_connect,        # try again if the nameserver is down
#       -local_mx_okay,         # fail (don't pass through) an MX
#                               # to the local host

#
# pathalias database for intra-domain routing
domain_paths:
        driver=pathalias,        # look up host in a paths file
        transport=uux;           # if matched, deliver over UUCP

        file=paths/domain,       # file is /usr/lib/smail/paths/domain
        proto=lsearch,           # file is unsorted (linear search)
        optional,                # ignore if the file does not exist
        required=vbrew.com,       # look up only *.vbrew.com hosts

#
# pathalias database for routing to hosts outside our domain
world_paths:
        driver=pathalias,        # look up host in a paths file
        transport=uux;           # if matched, deliver over UUCP

        file=paths/world,        # file is /usr/lib/smail/paths/world
        proto=bsearch,           # file is sorted with sort(1)
        optional,                # ignore if the file does not exist
        -required,               # no required domains
        domain=uucp,             # strip ending ".uucp" before searching


# smart_host – a partically specified smarthost director
#       If the smart_path attribute is not defined in
#       /usr/lib/smail/config, this router is ignored.
#       The transport attribute is overridden by the global
#       smart_transport variable
```

```
smart_host:
       driver=smarthost,        # special-case driver
       transport=uux;           # by default deliver over UUCP

       -path,                  # use smart_path config file variable
```

对于本地地址的邮件的处理是在 *directors* 文件中配置的。该文件的组成与 *routers* 文件相似，其中有一系列的定义每个导向器的条目。导向器并不投递消息，它们仅仅执行一些可能的重定向（例如通过别名）、邮件转发等等。

当将邮件投递到一个本地地址去时，比如是 **janet**，*smail* 依次将用户名送至各个导向器。如果有一个匹配，那么它或者指定一个该消息投递要用的传输方式（例如，到用户的邮件箱文件），或者生成一个新的地址（例如，在评估了一个别名以后）。

由于所包括的安全问题，导向器通常会做很多的检查，看看它们所使用的文件是否是有损安全的。以有些可疑的方法获取的地址（例如，从一个可写的 *aliases* 文件中）被标上是不安全的。某些传输驱动程序将会拒绝这样的地址，比如将消息投递到一个文件的传输。

除了这以外，*smail* 也将用户与每个地址相关联。任何读写操作是作为用户来执行的。比如对于投递到 **janet** 的邮件箱，地址当然是与 **janet** 相关联的。其它的地址，比如那些从 *aliases* 文件中获得的地址，是与其他用户相关联的，比如 **nobody** 用户。

有关这些特性的详细内容，请参考 *smail(8)*手册页。

```
# A sample /usr/lib/smail/directors file

# aliasinclude - expand ":include:filename" addresses produced
#      by alias files
aliasinclude:
       driver=aliasinclude,        # use this special-case driver
       nobody;                    # access file as nobody user if unsecure

       copysecure,                # get permissions from alias director
       copyowners,                # get owners from alias director

# forwardinclude - expand ":include:filename" addrs produced
#      by forward files
forwardinclude:
       driver=forwardinclude,      # use this special-case driver
       nobody;                    # access file as nobody user if unsecure

       checkpath,                 # check path accessibility
       copysecure,                # get perms from forwarding director
       copyowners,                # get owners from forwarding director

# aliases - search for alias expansions stored in a database
aliases:
```

```
        driver=aliasfile,          # general-purpose aliasing director
        -nobody,                  # all addresses are associated
                                 # with nobody by default anyway
        sender_okay,               # don't remove sender from expansions
        owner=owner-$user;         # problems go to an owner address

        file=/usr/lib/aliases,      # default: sendmail compatible
        modemask=002,              # should not be globally writable
        optional,                  # ignore if file does not exist
        proto=lsearch,             # unsorted ASCII file

# dotforward - expand .forward files in user home directories
dotforward:
        driver=forwardfile,              #  general-purpose  forwarding
    director
        owner=real-$user,          # problems go to the user's mailbox
        nobody,                   # use nobody user, if unsecure
        sender_okay;               # sender never removed from expansion

        file=~/.forward,           # .forward file in home directories
        checkowner,                # the user can own this file
        owners=root,               # or root can own the file
        modemask=002,              # it should not be globally writable
        caution=0-10:uucp:daemon,    # don't run things as root or daemons
        # be extra careful of remotely accessible home directories
        unsecure="~ftp:~uucp:~nuucp:/tmp:/usr/tmp",

# forwardto - expand a "Forward to " line at the top of
#      the user's mailbox file
forwardto:
        driver=forwardfile,
        owner=Postmaster,          # errors go to Postmaster
        nobody,                   # use nobody user, if unsecure
        sender_okay;               # don't remove sender from expansion

        file=/var/spool/mail/${lc:user}, # location of user's mailbox
        forwardto,                # enable "Forward to " check
        checkowner,                # the user can own this file
        owners=root,               # or root can own the file
        modemask=0002,             # under System V, group mail can write
        caution=0-10:uucp:daemon,    # don't run things as root or daemons

# user - match users on the local host with delivery to their mailboxes
user:  driver=user;               # driver to match usernames
```

```
        transport=local,          # local transport goes to mailboxes

# real_user - match usernames when prefixed with the string "real-"
real_user:
        driver=user;             # driver to match usernames

        transport=local,          # local transport goes to mailboxes
        prefix="real-",           # for example, match real-root

# lists - expand mailing lists stored below /usr/lib/smail/lists
lists: driver=forwardfile,
        caution,                 # flag all addresses with caution
        nobody,                  # and then associate the nobody user
        sender_okay,             # do NOT remove the sender
        owner=owner-$user;       # the list owner

        # map the name of the mailing list to lower case
        file=lists/${lc:user},
```

在成功地进行了路由选择或重定向一个消息以后，*smail* 将消息传给与地址匹配的路由器或导向器指定的传输方式。这些传输方式是在 *transporst* 文件中定义的。再一次，一个传输方式是由一族全局和私有选项定义的。

每个条目定义的最为重要的选项是处理传输的驱动程序，例如，*pipe* 驱动程序，它调用在 *cmd* 属性中指定的命令。除了这以外，传输方式还可以用许多的全局属性，它在消息头以及很可能在消息体上执行各种变换。例如 *return_path* 属性使得传输在消息头中插入一个 *return_path* 字段。*unix_from_hack* 属性使得它在以 From 开始的每行的的前面加上一个>符号。

```
# 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
```

附录 C
The GNU General Public License

Printed below is the GNU General Public License (the GPL or copyleft), under which Linux is licensed. It is reproduced here to clear up some of the confusion about Linux's copyright status Linux is not shareware, and it is not in the public domain. The bulk of the Linux kernel is copyright c 1993 by Linus Torvalds, and other software and parts of the kernel are copyrighted by their authors. Thus, Linux is copyrighted, however, you may redistribute it under the terms of the GPL printed below.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

C.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software{to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rightsor to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) o_er you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that

there is no warranty for this free software. If the software is modi_ed by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not re ect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in e_ect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modi_cation follow.

C.2 Terms and Conditions for Copying, Distribution, and Modi_cation

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The \Program", below, refers to any such program or work, and a \work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modi_cations and/or translated into another language. (Hereinafter, translation is included without limitation in the term \modi_cation".) Each licensee is addressed as \you".

   Activities other than copying, distribution and modi_cation are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option o_er warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modi_cations or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   a. You must cause the modi_ed _les to carry prominent notices stating that you changed the _les and the date of any change.
   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
   c. If the modi_ed program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the

Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modi_ed work as a whole. If identi_able sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. b. Accompany it with a written o_er, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the o_er to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an o_er, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modi_cations to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface de_nition _les, plus the scripts used to control com- pilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by o_ering access to copy from a designated place, then o_ering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Pro- gram is void, and will automatically terminate your rights under this License. However, parties who have received copies,

or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may di_er in

detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program speci_es a version number of this License which applies to it and \any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are di_erent, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WAR-RANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLD- ERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM \AS IS" WITHOUT WAR- RANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PER- FORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRIT- ING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CON- SEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSI- BILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

C.3 Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source _le to most e_ectively convey the exclusion of warranty; and each _le should have at least the \copyright" line and a pointer to where the full notice is found.

(one line to give the program's name and a brief idea of what it does.) Copyright c 19yy (name of author)

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FIT-NESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge,MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) 19yy name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items{whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a \copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

(signature of Ty Coon), 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary pro-grams. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

词汇表